ROCm Documentation

Release 5.7.0

Advanced Micro Devices, Inc.

May 02, 2024

CONTENTS

What is ROCm?

ROCm is an open-source stack, composed primarily of open-source software (OSS), designed for graphics processing unit (GPU) computation. ROCm consists of a collection of drivers, development tools, and APIs that enable GPU programming from low-level kernel to end-user applications. more...

Deploy ROCm

- Deploy ROCm on Linux
- Deploy ROCm Docker containers

Release Info

- Release Notes
- GPU and OS Support
- Known Issues
- Compatibility
- Licensing

APIs and Reference

- Compilers and Development Tools
- HIP
- OpenMP
- Math Libraries
- C++ Primitives Libraries
- Communication Libraries
- AI Libraries
- Computer Vision
- Management Tools
- Validation Tools

Understand ROCm

- Compiler Disambiguation
- Using CMake
- Linux Folder Structure Reorganization
- GPU Isolation Techniques
- GPU Architecture

How to Guides

- System Tuning for Various Architectures
- GPU Aware MPI
- Setting up for Deep Learning with ROCm

CHAPTER

ONE

WHAT IS ROCM?

ROCm is an open-source stack, composed primarily of open-source software (OSS), designed for graphics processing unit (GPU) computation. ROCm consists of a collection of drivers, development tools, and APIs that enable GPU programming from low-level kernel to end-user applications.

With ROCm, you can customize your GPU software to meet your specific needs. You can develop, collaborate, test, and deploy your applications in a free, open-source, integrated, and secure software ecosystem. ROCm is particularly well-suited to GPU-accelerated high-performance computing (HPC), artificial intelligence (AI), scientific computing, and computer aided design (CAD).

ROCm is powered by AMD's Heterogeneous-computing Interface for Portability (HIP), an OSS C++ GPU programming environment and its corresponding runtime. HIP allows ROCm developers to create portable applications on different platforms by deploying code on a range of platforms, from dedicated gaming GPUs to exascale HPC clusters.

ROCm supports programming models, such as OpenMP and OpenCL, and includes all necessary OSS compilers, debuggers, and libraries. ROCm is fully integrated into machine learning (ML) frameworks, such as PyTorch and TensorFlow.

1.1 ROCm on Windows

Starting with ROCm 5.5, the HIP SDK brings a subset of ROCm to developers on Windows. The collection of features enabled on Windows is referred to as the HIP SDK. These features allow developers to use the HIP runtime, HIP math libraries and HIP Primitive libraries. The following table shows the differences between Windows and Linux releases.

Component	Linux	Windows
Driver	Radeon Software for Linux	AMD Software Pro Edition
Compiler	hipcc/amdclang++	hipcc/clang++
Debugger	rocgdb	no debugger available
Profiler	rocprof	Radeon GPU Profiler
Porting Tools	HIPIFY	Coming Soon
Runtime	HIP (Open Sourced)	HIP (closed source)
Math Libraries	Supported	Supported
Primitives Libraries	Supported	Supported
Communication Libraries	Supported	Not Available
AI Libraries	MIOpen, MIGraphX	Not Available
System Management	rocm-smi-lib, RDC, rocminfo	amdsmi, hipInfo
AI Frameworks	PyTorch, TensorFlow, etc.	Not Available
CMake HIP Language	Enabled	Unsupported
Visual Studio	Not applicable	Plugin Available
HIP Ray Tracing	Supported	Supported

AMD is continuing to invest in Windows support and AMD plans to release enhanced features in subsequent revisions.

Note: The 5.5 Windows Installer collectively groups the Math and Primitives libraries.

Note: GPU support on Windows and Linux may differ. You must refer to Windows and Linux GPU support tables separately.

Note: HIP Ray Tracing is not distributed via ROCm in Linux.

1.1.1 ROCm release versioning

Linux OS releases set the canonical version numbers for ROCm. Windows will follow Linux version numbers as Windows releases are based on Linux ROCm releases. However, not all Linux ROCm releases will have a corresponding Windows release. The following table shows the ROCm releases on Windows and Linux. Releases with both Windows and Linux are referred to as a joint release. Releases with only Linux support are referred to as a skipped release from the Windows perspective.

Release version	Linux	Windows
5.5		
5.6		

ROCm Linux releases are versioned with following the Major.Minor.Patch version number system. Windows releases will only be versioned with Major.Minor.

In general, Windows releases will trail Linux releases. Software developers that wish to support both Linux and Windows using a single ROCm version should refrain from upgrading ROCm unless there is a joint release.

1.1.2 Windows Documentation implications

The ROCm documentation website contains both Windows and Linux documentation. Just below each article title, a convenient article information section states whether the page applies to Linux only, Windows only or both OSes. To find the exact Windows documentation for a release of the HIP SDK, please view the ROCm documentation with the same Major.Minor version number while ignoring the Patch version. The Patch version only matters for Linux releases. For convenience, Windows documentation will continue to be included in the overall ROCm documentation for the skipped Windows releases.

Windows release notes will contain only information pertinent to Windows. The software developer must read all the previous ROCm release notes (including) skipped ROCm versions on Windows for information on all the changes present in the Windows release.

1.1.3 Windows Builds from Source

Not all source code required to build Windows from source is available under a permissive open source license. Build instructions on Windows is only provided for projects that can be built from source on Windows using a toolchain that has closed source build prerequisites. The ROCm manifest file is not valid for Windows. AMD does not release a manifest or tag our components in Windows. Users may use corresponding Linux tags to build on Windows.

CHAPTER

TWO

QUICK START (LINUX)

2.1 Add Repositories

Ubuntu

1. Download and convert the package signing key

```
# Make the directory if it doesn't exist yet.
# This location is recommended by the distribution maintainers.
sudo mkdir --parents --mode=0755 /etc/apt/keyrings
# Download the key, convert the signing-key to a full
# keyring required by apt and store in the keyring directory
wget https://repo.radeon.com/rocm/rocm.gpg.key -O - | \
    gpg --dearmor | sudo tee /etc/apt/keyrings/rocm.gpg > /dev/null
```

2. Add the repositories

Ubuntu 20.04

```
# Kernel driver repository for focal
sudo tee /etc/apt/sources.list.d/amdgpu.list <<'EOF'
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/5.7/ubuntu focal␣
↪main
EOF
# ROCm repository for focal
sudo tee /etc/apt/sources.list.d/rocm.list <<'EOF'
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/5.7 focal main
EOF
```

Ubuntu 22.04

```
# Kernel driver repository for jammy
sudo tee /etc/apt/sources.list.d/amdgpu.list <<'EOF'
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/5.7/ubuntu jammy␣
↪main
EOF
# ROCm repository for jammy
sudo tee /etc/apt/sources.list.d/rocm.list <<'EOF'
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/5.7 jammy main
```

(continues on next page)

(continued from previous page)

```
EOF
# Prefer packages from the rocm repository over system packages
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' | sudo tee /etc/apt/preferences.d/rocm-
↪pin-600
```

3. Update the list of packages

```
sudo apt update
```

Red Hat Enterprise Linux

1. Add the repositories

RHEL 8.7

```
# Add the amdgpu module repository for RHEL 8.7
sudo tee /etc/yum.repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/5.7/rhel/8.7/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
# Add the rocm repository for RHEL 8
sudo tee /etc/yum.repos.d/rocm.repo <<'EOF'
[rocm]
name=rocm
baseurl=https://repo.radeon.com/rocm/rhel8/5.7/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

RHEL 8.8

```
# Add the amdgpu module repository for RHEL 8.8
sudo tee /etc/yum.repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/5.7/rhel/8.8/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
# Add the rocm repository for RHEL 8
sudo tee /etc/yum.repos.d/rocm.repo <<'EOF'
[rocm]
name=rocm
```

(continues on next page)

(continued from previous page)

```
baseurl=https://repo.radeon.com/rocm/rhel8/5.7/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

RHEL 9.1

```
# Add the amdgpu module repository for RHEL 9.1
sudo tee /etc/yum.repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/5.7/rhel/9.1/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
# Add the rocm repository for RHEL 9
sudo tee /etc/yum.repos.d/rocm.repo <<'EOF'
[rocm]
name=rocm
baseurl=https://repo.radeon.com/rocm/rhel9/5.7/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

RHEL 9.2

```
# Add the amdgpu module repository for RHEL 9.2
sudo tee /etc/yum.repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/5.7/rhel/9.2/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
# Add the rocm repository for RHEL 9
sudo tee /etc/yum.repos.d/rocm.repo <<'EOF'
[rocm]
name=rocm
baseurl=https://repo.radeon.com/rocm/rhel9/5.7/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

2. Clean cached files from enabled repositories

```
sudo yum clean all
```

SUSE Linux Enterprise Server

1. Add the repositories

SLES 15.4

```
# Add the amdgpu module repository for SLES 15.4
sudo tee /etc/zypp/repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/5.7/sle/15.4/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
# Add the rocm repository for SLES
sudo tee /etc/zypp/repos.d/rocm.repo <<'EOF'
[rocm]
name=rocm
baseurl=https://repo.radeon.com/rocm/zyp/zypper
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

SLES 15.5

```
# Add the amdgpu module repository for SLES 15.5
sudo tee /etc/zypp/repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/5.7/sle/15.5/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
# Add the rocm repository for SLES
sudo tee /etc/zypp/repos.d/rocm.repo <<'EOF'
[rocm]
name=rocm
baseurl=https://repo.radeon.com/rocm/zyp/zypper
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

2. Update the new repository

```
sudo zypper ref
```

2.2 Install Drivers

Install the amdgpu-dkms kernel module, aka driver, on your system.

Ubuntu

```
sudo apt install amdgpu-dkms
```

Red Hat Enterprise Linux

```
sudo yum install amdgpu-dkms
```

SUSE Linux Enterprise Server

```
sudo zypper install amdgpu-dkms
```

2.3 Install ROCm Runtimes

Install the rocm-hip-libraries meta-package. This contains dependencies for most common ROCm applications.

Ubuntu

```
sudo apt install rocm-hip-libraries
```

Red Hat Enterprise Linux

```
sudo yum install rocm-hip-libraries
```

SUSE Linux Enterprise Server

```
sudo zypper install rocm-hip-libraries
```

2.4 Reboot the system

Loading the new driver requires a reboot of the system.

```
sudo reboot
```

CHAPTER

THREE

DEPLOY ROCM ON LINUX

Start with Quick Start (Linux) or follow the detailed instructions below.

3.1 Prepare to Install

Prerequisites The prerequisites page lists the required steps before installation.

Install Choices Package manager vs AMDGPU Installer

Standard Packages vs Multi-Version Packages

3.2 Choose your install method

Package Manager Directly use your distribution's package manager to install ROCm.

AMDGPU Installer Use an installer tool that orchestrates changes via the package manager.

3.3 See Also

- GPU Support and OS Compatibility (Linux)

3.4 ROCm Installation Options (Linux)

Users installing ROCm must choose between various installation options. A new user should follow the Quick Start guide.

3.4.1 Package Manager versus AMDGPU Installer?

ROCm supports two methods for installation:

- Directly using the Linux distribution's package manager
- The amdgpu-install script

There is no difference in the final installation state when choosing either option.

Using the distribution's package manager lets the user install, upgrade and uninstall using familiar commands and workflows. Third party ecosystem support is the same as your OS package manager.

The amdgpu-install script is a wrapper around the package manager. The same packages are installed by this script as the package manager system.

The installer automates the installation process for the AMDGPU and ROCm stack. It handles the complete installation process for ROCm, including setting up the repository, cleaning the system, updating, and installing the desired drivers and meta-packages. Users who are less familiar with the package manager can choose this method for ROCm installation.

3.4.2 Single Version ROCm install versus Multi-Version

ROCm packages are versioned with both semantic versioning that is package specific and a ROCm release version.

3.4.2.1 Single-version Installation

The single-version ROCm installation refers to the following:

- Installation of a single instance of the ROCm release on a system
- Use of non-versioned ROCm meta-packages

3.4.2.2 Multi-version Installation

The multi-version installation refers to the following:

- Installation of multiple instances of the ROCm stack on a system. Extending the package name and its dependencies with the release version adds the ability to support multiple versions of packages simultaneously.
- Use of versioned ROCm meta-packages.

Attention: ROCm packages that were previously installed from a single-version installation must be removed before proceeding with the multi-version installation to avoid conflicts.

Note: Multiversion install is not available for the kernel driver module, also referred to as AMDGPU.

The following image demonstrates the difference between single-version and multi-version ROCm installation types:

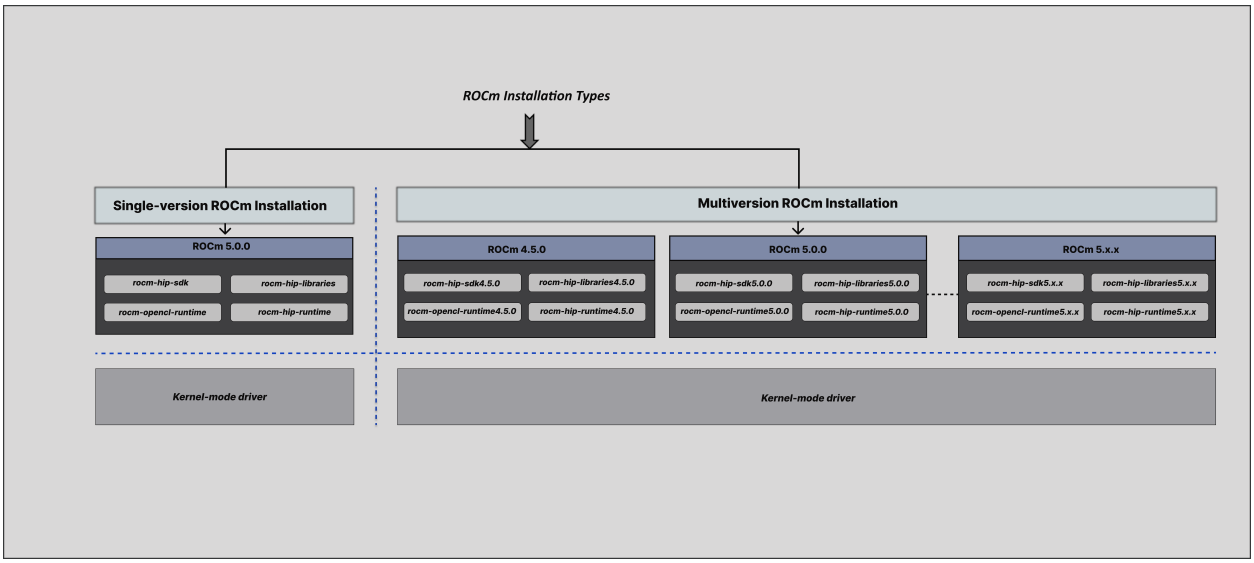


Fig. 3.1: ROCm Installation Types

3.5 Installation Prerequisites (Linux)

You must perform the following steps before installing ROCm and check if the system meets all the requirements to proceed with the installation.

3.5.1 Confirm the System Has a Supported Linux Distribution Version

The ROCm installation is supported only on specific Linux distributions and kernel versions.

3.5.1.1 Check the Linux Distribution and Kernel Version on Your System

This section discusses obtaining information about the Linux distribution and kernel version.

3.5.1.1.1 Linux Distribution Information

Verify the Linux distribution using the following steps:

1. To obtain the Linux distribution information, type the following command on your system from the Command Line Interface (CLI):

   ```
   uname -m && cat /etc/*release
   ```

2. Confirm that the obtained Linux distribution information matches with those listed in Supported Linux Distributions.

   Example: Running the command above on an Ubuntu system results in the following output:

   ```
   x86_64
   DISTRIB_ID=Ubuntu
   DISTRIB_RELEASE=20.04
   ```

(continues on next page)

(continued from previous page)

```
DISTRIB_CODENAME=focal
DISTRIB_DESCRIPTION="Ubuntu 20.04.5 LTS"
```

3.5.1.1.2 Kernel Information

Verify the kernel version using the following steps:

1. To check the kernel version of your Linux system, type the following command:

   ```
   uname -srmv
   ```

   Example: The output of the command above lists the kernel version in the following format:

   ```
   Linux 5.15.0-46-generic #44~20.04.5-Ubuntu SMP Fri Jun 24 13:27:29 UTC 2022 x86_64
   ```

2. Confirm that the obtained kernel version information matches with system requirements as listed in Supported Linux Distributions.

3.5.2 Additional package repositories

On some distributions the ROCm packages depend on packages outside the default package repositories. These extra repositories need to be enabled before installation. Follow the instructions below based on your distributions.

Ubuntu

All packages are available in the default Ubuntu repositories, therefore no additional repositories need to be added.

Red Hat Enterprise Linux

1. Add the EPEL repository

RHEL 8

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
sudo rpm -ivh epel-release-latest-8.noarch.rpm
```

RHEL 9

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
```

2. Enable the CodeReady Linux Builder repository

Run the following command and follow the instructions.

```
sudo crb enable
```

SUSE Linux Enterprise Server

Add the perl languages repository.

Note: Mar 25, 2024: We currently need to install the Perl module from SLES 15 SP5 as a workaround. The module was removed for SLES 15 SP4.

SLES 15.4

```
zypper addrepo https://download.opensuse.org/repositories/devel:/languages:/perl/15.5/devel:languages:perl.repo
```

SLES 15.5

```
zypper addrepo https://download.opensuse.org/repositories/devel:/languages:/perl/15.5/devel:languages:perl.repo
```

3.5.3 Kernel headers and development packages

The driver package uses DKMS (Dynamic Kernel Module Support) to build the amdgpu-dkms module (driver) for the installed kernels. This requires the Linux kernel headers and modules to be installed for each. Usually these are automatically installed with the kernel, but if you have multiple kernel versions or you have downloaded the kernel images and not the kernel meta-packages then they must be manually installed.

To install for the currently active kernel run the command corresponding to your distribution.

Ubuntu

```
sudo apt install "linux-headers-$(uname -r)" "linux-modules-extra-$(uname -r)"
```

Red Hat Enterprise Linux

```
sudo yum install kernel-headers kernel-devel
```

SUSE Linux Enterprise Server

```
sudo zypper install kernel-default-devel
```

3.5.4 Setting Permissions for Groups

This section provides steps to add any current user to a video group to access GPU resources. Use of the video group is recommended for all ROCm-supported operating systems.

1. To check the groups in your system, issue the following command:

   ```
   groups
   ```

2. Add yourself to the render and video group using the command:

   ```
   sudo usermod -a -G render,video $LOGNAME
   ```

To add all future users to the video and render groups by default, run the following commands:

```
echo 'ADD_EXTRA_GROUPS=1' | sudo tee -a /etc/adduser.conf
echo 'EXTRA_GROUPS=video' | sudo tee -a /etc/adduser.conf
echo 'EXTRA_GROUPS=render' | sudo tee -a /etc/adduser.conf
```

3.6 Installation via Package manager

Install How to install ROCm?

Upgrade Instructions for upgrading an existing ROCm installation.

Uninstall Steps for removing ROCm packages libraries and tools.

Package Manager Integration Information about packages.

3.6.1 See Also

- GPU Support and OS Compatibility (Linux)

3.6.2 Installation (Linux)

Warning: ROCm currently doesn't support integrated graphics. Should your system have an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime may crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

3.6.2.1 Understanding the Release-specific AMDGPU and ROCm Repositories on Linux Distributions

The release-specific repositories consist of packages from a specific release of versions of AMDGPU and ROCm. The repositories are not updated for the latest packages with subsequent releases. When a new ROCm release is available, the new repository, specific to that release, is added. You can select a specific

release to install, update the previously installed single version to the later available release, or add the latest version of ROCm along with the currently installed version by using the multi-version ROCm packages.

3.6.2.2 Step by Step Instructions

Ubuntu

1. Download and convert the package signing key

```
# Make the directory if it doesn't exist yet.
# This location is recommended by the distribution maintainers.
sudo mkdir --parents --mode=0755 /etc/apt/keyrings
# Download the key, convert the signing-key to a full
# keyring required by apt and store in the keyring directory
wget https://repo.radeon.com/rocm/rocm.gpg.key -O - | \
    gpg --dearmor | sudo tee /etc/apt/keyrings/rocm.gpg > /dev/null
```

Note: The GPG key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm to the apt repository as mentioned above. The current rocm.gpg.key is not available in a standard key ring distribution but has the following SHA1 sum hash: 73f5d8100de6048aa38a8b84cd9a87f05177d208 rocm.gpg.key

---

2. Add the AMDGPU Repository and Install the Kernel-mode Driver

---

Tip: If you have a version of the kernel-mode driver installed, you may skip this section.

To add the AMDGPU repository, follow these steps:

Ubuntu 20.04

```
# version
ver=5.7

# amdgpu repository for focal
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/$ver/ubuntu␣
↪focal main" \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

Ubuntu 22.04

```
# version
ver=5.7

# amdgpu repository for jammy
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/$ver/ubuntu␣
↪jammy main" \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
# Prefer packages from the rocm repository over system packages
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' | sudo tee /etc/apt/preferences.d/rocm-
↪pin-600
```

Install the kernel mode driver and reboot the system using the following commands:

```
sudo apt install amdgpu-dkms
sudo reboot
```

3. Add the ROCm Repository

To add the ROCm repository, use the following steps:

Ubuntu 20.04

```
# ROCm repositories for focal
for ver in 5.3.3 5.4.6 5.5.3 5.6.1 5.7; do
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver focal␣
↪main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

Ubuntu 22.04

```
# ROCm repositories for jammy
for ver in 5.3.3 5.4.6 5.5.3 5.6.1 5.7; do
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver jammy␣
↪main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

4. Install packages

Install packages of your choice in a single-version ROCm install or in a multi-version ROCm install fashion. For more information on what single/multi-version installations are, refer to Single Version ROCm install versus Multi-Version. For a comprehensive list of meta-packages, refer to Meta-packages and Their Descriptions.

- Sample Single-version installation

```
sudo apt install rocm-hip-sdk
```

- Sample Multi-version installation

```
sudo apt install rocm-hip-sdk5.7 rocm-hip-sdk5.6.1 rocm-hip-sdk5.5.3
```

Red Hat Enterprise Linux

1. Add the AMDGPU Stack Repository and Install the Kernel-mode Driver

Tip: If you have a version of the kernel-mode driver installed, you may skip this section.

RHEL 8.7

```
# version
ver=5.7

sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$ver/rhel/8.7/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8.8

```
# version
ver=5.7

sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$ver/rhel/8.8/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

(continues on next page)

(continued from previous page)

```
EOF
sudo yum clean all
```

RHEL 9.1

```
# version
ver=5.7

sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$ver/rhel/9.1/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 9.2

```
# version
ver=5.7

sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$ver/rhel/9.2/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

Install the kernel mode driver and reboot the system using the following commands:

```
sudo yum install amdgpu-dkms
sudo reboot
```

2. Add the ROCm Stack Repository

To add the ROCm repository, use the following steps, based on your distribution:

RHEL 8

```
for ver in 5.3.3 5.4.6 5.5.3 5.6.1 5.7; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/rhel8/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

RHEL 9

```
for ver in 5.3.3 5.4.6 5.5.3 5.6.1 5.7; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/rhel9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

3. Install packages

Install packages of your choice in a single-version ROCm install or in a multi-version ROCm install fashion. For more information on what single/multi-version installations are, refer to Single Version ROCm install versus Multi-Version. For a comprehensive list of meta-packages, refer to Meta-packages and Their Descriptions.

- Sample Single-version installation

  ```
  sudo yum install rocm-hip-sdk
  ```

- Sample Multi-version installation

```
sudo yum install rocm-hip-sdk5.7 rocm-hip-sdk5.6.1
```

SUSE Linux Enterprise Server

1. Add the AMDGPU Repository and Install the Kernel-mode Driver

Tip: If you have a version of the kernel-mode driver installed, you may skip this section.

SLES 15.4

```
# version
ver=5.7


sudo tee /etc/zypp/repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$ver/sle/15.4/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo zypper ref
```

SLES 15.5

```
# version
ver=5.7

sudo tee /etc/zypp/repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$ver/sle/15.5/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo zypper ref
```

Install the kernel mode driver and reboot the system using the following commands:

```
sudo zypper --gpg-auto-import-keys install amdgpu-dkms
sudo reboot
```

2. Add the ROCm Stack Repository

To add the ROCm repository, use the following steps:

```
for ver in 5.3.3 5.4.6 5.5.3 5.6.1 5.7; do
sudo tee --append /etc/zypp/repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
name=rocm
baseurl=https://repo.radeon.com/rocm/zyp/$ver/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo zypper ref
```

3. Install packages

Install packages of your choice in a single-version ROCm install or in a multi-version ROCm install fashion. For more information on what single/multi-version installations are, refer to Single Version ROCm install versus Multi-Version. For a comprehensive list of meta-packages, refer to Meta-packages and Their Descriptions.

- Sample Single-version installation

```
sudo zypper --gpg-auto-import-keys install rocm-hip-sdk
```

- Sample Multi-version installation

```
sudo zypper --gpg-auto-import-keys install rocm-hip-sdk5.7 rocm-hip-sdk5.6.1
```

3.6.2.3 Post-install Actions and Verification Process

The post-install actions listed here are optional and depend on your use case, but are generally useful. Verification of the install is advised.

3.6.2.3.1 Post-install Actions

1. Instruct the system linker where to find the shared objects (.so files) for ROCm applications.

```
sudo tee --append /etc/ld.so.conf.d/rocm.conf <<EOF
/opt/rocm/lib
/opt/rocm/lib64
EOF
sudo ldconfig
```

Note: Multi-version installations require extra care. Having multiple versions on the system linker library search path is unadvised. One must take care both at compile-time and at run-time to assure that the proper libraries are picked up. You can override ld.so.conf entries on a case-by-case basis using the LD_LIBRARY_PATH environmental variable.

2. Add binary paths to the PATH environment variable.

```
export PATH=$PATH:/opt/rocm-5.7/bin:/opt/rocm-5.7/opencl/bin
```

> Attention: When using CMake to build applications, having the ROCm install location on the PATH subtly affects how ROCm libraries are searched for. See Config Mode Search Procedure and CMAKE_FIND_USE_SYSTEM_ENVIRONMENT_PATH for details.
>
> (Entries in the PATH minus bin and sbin are added to library search paths, therefore this convenience will affect builds and result in ROCm libraries almost always being found. This may be an issue when you're developing these libraries or want to use self-built versions of them.)

3.6.2.3.2 Verifying Kernel-mode Driver Installation

Check the installation of the kernel-mode driver by typing the command given below:

```
dkms status
```

3.6.2.3.3 Verifying ROCm Installation

After completing the ROCm installation, execute the following commands on the system to verify if the installation is successful. If you see your GPUs listed by both commands, the installation is considered successful:

```
/opt/rocm/bin/rocminfo
# OR
/opt/rocm/opencl/bin/clinfo
```

3.6.2.3.4 Verifying Package Installation

To ensure the packages are installed successfully, use the following commands:

Ubuntu

```
sudo apt list --installed
```

Red Hat Enterprise Linux

```
sudo yum list installed
```

SUSE Linux Enterprise Server

```
sudo zypper search --installed-only
```

3.6.3 Upgrade ROCm with the package manager

This section explains how to upgrade the existing AMDGPU driver and ROCm packages to the latest version using your OS's distributed package manager.

Note: Package upgrade is applicable to single-version packages only. If the preference is to install an updated version of the ROCm along with the currently installed version, refer to the Installation (Linux) page.

3.6.3.1 Upgrade Steps

3.6.3.1.1 Update the AMDGPU repository

Execute the commands below based on your distribution to point the amdgpu repository to the new release.

Ubuntu

Ubuntu 20.04

```
# version
version=5.7


# amdgpu repository for focal
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/$version/
↪ubuntu focal main" \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

Ubuntu 22.04

```
# version
version=5.7


# amdgpu repository for jammy
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/$version/
↪ubuntu jammy main" \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

Red Hat Enterprise Linux

RHEL 8.7

```
# version
version=5.7


sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$version/rhel/8.7/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8.8

```
# version
version=5.7


sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$version/rhel/8.8/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 9.1

```
# version
version=5.7


sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$version/rhel/9.1/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 9.2

```
# version
version=5.7


sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$version/rhel/9.2/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

SUSE Linux Enterprise Server

SLES 15.4

```
# version
version=5.7


sudo tee /etc/zypp/repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$version/sle/15.4/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo zypper ref
```

SLES 15.5

```
# version
version=5.7


sudo tee /etc/zypp/repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/$version/sle/15.5/main/x86_64
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo zypper ref
```

3.6.3.1.2 Upgrade the kernel-mode driver & reboot

Upgrade the kernel mode driver and reboot the system using the following commands based on your distribution:

Ubuntu

```
sudo apt install amdgpu-dkms
sudo reboot
```

Red Hat Enterprise Linux

```
sudo yum install amdgpu-dkms
sudo reboot
```

SUSE Linux Enterprise Server

```
sudo zypper --gpg-auto-import-keys install amdgpu-dkms
sudo reboot
```

3.6.3.1.3 Update the ROCm repository

Execute the commands below based on your distribution to point the rocm repository to the new release.

Ubuntu

Ubuntu 20.04

```
# version
version=5.7


echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$version␣
↪focal main" \
    | sudo tee /etc/apt/sources.list.d/rocm.list
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

Ubuntu 22.04

```
# version
version=5.7


echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$version␣
↪jammy main" \
    | sudo tee /etc/apt/sources.list.d/rocm.list
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

Red Hat Enterprise Linux

RHEL 8

```
# version
version=5.7


sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/rhel8/$version/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 9

```
# version
version=5.7


sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/rhel9/$version/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

SUSE Linux Enterprise Server

```
# version
version=5.7

sudo tee /etc/zypp/repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
name=rocm
baseurl=https://repo.radeon.com/rocm/zyp/$version/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo zypper ref
```

3.6.3.1.4 Upgrade the ROCm packages

Your packages can be upgraded now through their meta-packages, see the following example based on your distribution:

Ubuntu

```
sudo apt install --only-upgrade rocm-hip-sdk
```

Red Hat Enterprise Linux

```
sudo yum update rocm-hip-sdk
```

Suse Linux Enterprise Server

```
sudo zypper --gpg-auto-import-keys update rocm-hip-sdk
```

3.6.3.2 Verification Process

To verify if the upgrade is successful, refer to the Post-install Actions and Verification Process given in the Installation section.

3.6.4 Uninstallation with package manager (Linux)

This section describes how to uninstall ROCm with the Linux distribution's package manager. This method should be used if ROCm was installed via the package manager. If the installer script was used for installation, then it should be used for uninstallation too, refer to Installer Script Uninstallation (Linux).

Ubuntu

Uninstalling Specific Meta-packages

```
# Uninstall single-version ROCm packages
sudo apt autoremove <package-name>
# Uninstall multiversion ROCm packages
sudo apt autoremove <package-name with release version>
```

Complete Uninstallation of ROCm Packages

```
# Uninstall single-version ROCm packages
sudo apt autoremove rocm-core
# Uninstall multiversion ROCm packages
sudo apt autoremove rocm-core<release version>
```

Uninstall Kernel-mode Driver

```
sudo apt autoremove amdgpu-dkms
```

Remove ROCm and AMDGPU Repositories

1. Execute these commands:

   ```
   sudo rm /etc/apt/sources.list.d/<rocm_repository-name>.list
   sudo rm /etc/apt/sources.list.d/<amdgpu_repository-name>.list
   ```

2. Clear the cache and clean the system.

   ```
   sudo rm -rf /var/cache/apt/*
   sudo apt-get clean all
   ```

3. Restart the system.

   ```
   sudo reboot
   ```

Red Hat Enterprise Linux

Uninstalling Specific Meta-packages

```
# Uninstall single-version ROCm packages
sudo yum remove <package-name>
# Uninstall multiversion ROCm packages
sudo yum remove <package-name with release version>
```

Complete Uninstallation of ROCm Packages

```
# Uninstall single-version ROCm packages
sudo yum remove rocm-core
# Uninstall multiversion ROCm packages
sudo yum remove rocm-core<release version>
```

Uninstall Kernel-mode Driver

```
sudo yum autoremove amdgpu-dkms
```

Remove ROCm and AMDGPU Repositories

1. Execute these commands:

   ```
   sudo rm -rf /etc/yum.repos.d/<rocm_repository-name> # Remove only rocm repo
   sudo rm -rf /etc/yum.repos.d/<amdgpu_repository-name> # Remove only amdgpu repo
   ```

2. Clear the cache and clean the system.

   ```
   sudo rm -rf /var/cache/yum #Remove the cache
   sudo yum clean all
   ```

3. Restart the system.

   ```
   sudo reboot
   ```

SUSE Linux Enterprise Server

Uninstalling Specific Meta-packages

```
# Uninstall all single-version ROCm packages
sudo zypper remove <package-name>
# Uninstall all multiversion ROCm packages
sudo zypper remove <package-name with release version>
```

Complete Uninstallation of ROCm Packages

```
# Uninstall all single-version ROCm packages
sudo zypper remove rocm-core
# Uninstall all multiversion ROCm packages
sudo zypper remove rocm-core<release version>
```

Uninstall Kernel-mode Driver

```
sudo zypper remove --clean-deps amdgpu-dkms
```

Remove ROCm and AMDGPU Repositories

1. Execute these commands:

   ```
   sudo zypper removerepo <rocm_repository-name>
   sudo zypper removerepo <amdgpu_repository-name>
   ```

2. Clear the cache and clean the system.

   ```
   sudo zypper clean --all
   ```

3. Restart the system.

   ```
   sudo reboot
   ```

3.6.5 Package Manager Integration

This section provides information about the required meta-packages for the following AMD ROCm programming models:

- Heterogeneous-Computing Interface for Portability (HIP)
- OpenCL™
- OpenMP™

3.6.5.1 ROCm Package Naming Conventions

A meta-package is a grouping of related packages and dependencies used to support a specific use case.

All meta-packages exist in both versioned and non-versioned forms.

- Non-versioned packages – For a single-version installation of the ROCm stack
- Versioned packages – For multi-version installations of the ROCm stack

Fig. 3.2 demonstrates the single and multi-version ROCm packages' naming structure, including examples for various Linux distributions. See terms below:

Module - It is the part of the package that represents the name of the ROCm component.

Example: The examples mentioned in the image represent the ROCm HIP module.

Module version - It is the version of the library released in that package. It should increase with a newer release.

Release version - It shows the ROCm release version when the package was released.

Example: 50400 points to the ROCm 5.4.0 release.

Build id - It represents the Jenkins build number for that release.

Arch - It shows the architecture for which the package was created.

Distro - It describes the distribution for which the package was created. It is valid only for rpm packages.

Fig. 3.2: ROCm Release Package Naming

Example: el8 represents RHEL 8.x packages.

3.6.5.2 Components of ROCm Programming Models

Fig. 3.3 demonstrates the high-level layered architecture of ROCm programming models and their meta-packages. All meta-packages are a combination of required packages and libraries.

Example:

- rocm-hip-runtime is used to deploy on supported machines to execute HIP applications.
- rocm-hip-sdk contains runtime components to deploy and execute HIP applications.

Note: rocm-llvm is not a meta-package but a single package that installs the ROCm clang compiler files.

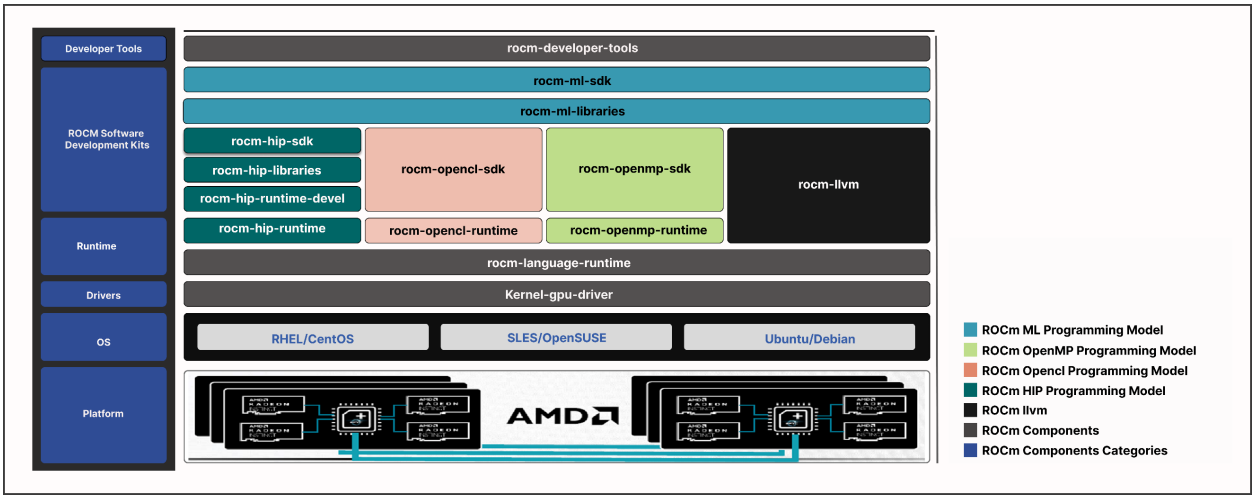


Fig. 3.3: ROCm Meta Packages

Table 3.1: Meta-packages and Their Descriptions

Meta-packages	Description
rocm-language-runtime	The ROCm runtime
rocm-hip-runtime	Run HIP applications written for the AMD platform
rocm-opencl-runtime	Run OpenCL-based applications on the AMD platform
rocm-hip-runtime-devel	Develop applications on HIP or port from CUDA
rocm-opencl-sdk	Develop applications in OpenCL for the AMD platform
rocm-hip-libraries	HIP libraries optimized for the AMD platform
rocm-hip-sdk	Develop or port HIP applications and libraries for the AMD platform
rocm-developer-tools	Debug and profile HIP applications
rocm-ml-sdk	Develop and run Machine Learning applications with optimized for AMD
rocm-ml-libraries	Key Machine Learning libraries, specifically MIOpen
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD platform
rocm-openmp-runtime	Run OpenMP-based applications for the AMD platform

3.6.5.3 Packages in ROCm Programming Models

This section discusses the available meta-packages and their packages. The following image visualizes the meta-packages and their associated packages in a ROCm programming model.

- Meta-packages can include another meta-package.
- rocm-core package is common across all the meta-packages.
- Meta-packages and associated packages are represented in the same color.

Note: Fig. 3.4 is for informational purposes only, as the individual packages in a meta-package are subject to change. Install meta-packages, and not individual packages, to avoid conflicts.

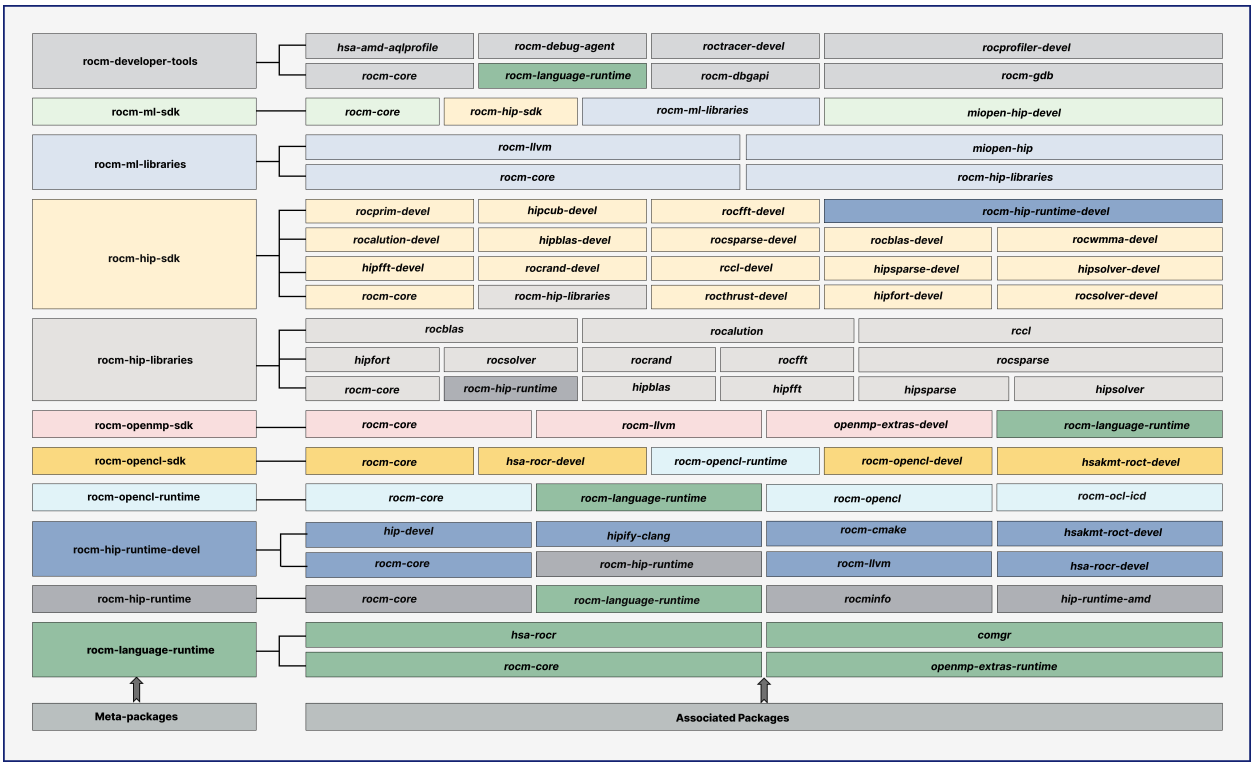


Fig. 3.4: Associated Packages

3.7 AMDGPU Install Script

Install How to install ROCm?

Upgrade Instructions for upgrading an existing ROCm installation.

Uninstall Steps for removing ROCm packages, libraries and tools.

3.7.1 See Also

- GPU Support and OS Compatibility (Linux)

3.7.2 Installation with install script

Prior to beginning, please ensure you have the prerequisites installed.

> Warning: ROCm currently doesn't support integrated graphics. Should your system have an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime may crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

3.7.2.1 Download the Installer Script

To download and install the amdgpu-install script on the system, use the following commands based on your distribution.

Ubuntu

Ubuntu 20.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/5.7/ubuntu/focal/amdgpu-install_5.7.50700-1_all.deb
sudo apt install ./amdgpu-install_5.7.50700-1_all.deb
```

Ubuntu 22.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/5.7/ubuntu/jammy/amdgpu-install_5.7.50700-1_all.deb
sudo apt install ./amdgpu-install_5.7.50700-1_all.deb
```

Red Hat Enterprise Linux

RHEL 8.7

```
sudo yum install https://repo.radeon.com/amdgpu-install/5.7/rhel/8.7/amdgpu-install-5.7.50700-1.el8.noarch.rpm
```

RHEL 8.8

```
sudo yum install https://repo.radeon.com/amdgpu-install/5.7/rhel/8.8/amdgpu-install-5.7.50700-1.el8.noarch.rpm
```

RHEL 9.1

```
sudo yum install https://repo.radeon.com/amdgpu-install/5.7/rhel/9.1/amdgpu-install-5.7.50700-1.el9.noarch.rpm
```

RHEL 9.2

```
sudo yum install https://repo.radeon.com/amdgpu-install/5.7/rhel/9.2/amdgpu-install-5.7.50700-1.el9.noarch.rpm
```

SUSE Linux Enterprise Server

SLES 15.4

```
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/5.7/sle/15.4/amdgpu-install-5.7.
↪50700-1.noarch.rpm
```

SLES 15.5

```
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/5.7/sle/15.5/amdgpu-install-5.7.
↪50700-1.noarch.rpm
```

3.7.2.2 Use cases

Instead of installing individual applications or libraries the installer script groups packages into specific use cases, matching typical workflows and runtimes.

To display a list of available use cases execute the command:

```
sudo amdgpu-install --list-usecase
```

The available use-cases will be printed in a format similar to the example output below.

```
If --usecase option is not present, the default selection is "graphics,opencl,hip"

Available use cases:
rocm(for users and developers requiring full ROCm stack)
- OpenCL (ROCr/KFD based) runtime
- HIP runtimes
- Machine learning framework
- All ROCm libraries and applications
- ROCm Compiler and device libraries
- ROCr runtime and thunk
lrt(for users of applications requiring ROCm runtime)
- ROCm Compiler and device libraries
- ROCr runtime and thunk
opencl(for users of applications requiring OpenCL on Vega or
later products)
- ROCr based OpenCL
- ROCm Language runtime

openclsdk (for application developers requiring ROCr based OpenCL)
- ROCr based OpenCL
- ROCm Language runtime
- development and SDK files for ROCr based OpenCL

hip(for users of HIP runtime on AMD products)
- HIP runtimes
hiplibsdk (for application developers requiring HIP on AMD products)
- HIP runtimes
- ROCm math libraries
- HIP development libraries
```

To install use cases specific to your requirements, use the installer amdgpu-install as follows:

- To install a single use case add it with the --usecase option:

```
sudo amdgpu-install --usecase=rocm
```

- For multiple use cases separate them with commas:

```
sudo amdgpu-install --usecase=hiplibsdk,rocm
```

- For graphical workloads using the open-source driver add graphics. For example:

```
sudo amdgpu-install --usecase=graphics,rocm
```

- For graphical workloads using the proprietary driver add graphics. For example:

```
sudo amdgpu-install --usecase=workstation,rocm
```

3.7.2.3 Single-version ROCm Installation

By default (without the --rocmrelease option) the installer script will install packages in the single-version layout.

3.7.2.4 Multi-version ROCm Installation

For the multi-version ROCm installation you must use the installer script from the latest release of ROCm that you wish to install.

Example: If you want to install ROCm releases 5.5.3, 5.6.1 and 5.7 simultaneously, you are required to download the installer from the latest ROCm release 5.7.

3.7.2.4.1 Add Required Repositories

You must add the ROCm repositories manually for all ROCm releases you want to install except the latest one. The amdgpu-install script automatically adds the required repositories for the latest release.

Run the following commands based on your distribution to add the repositories:

Ubuntu

Ubuntu 20.04

```
for ver in 5.5.3 5.6.1 5.7; do
echo "deb [arch=amd64 signed-by=/etc/apt/trusted.gpg.d/rocm-keyring.gpg] https://repo.radeon.com/rocm/apt/
↪$ver focal main" | sudo tee /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' | sudo tee /etc/apt/preferences.d/rocm-
↪pin-600
sudo apt update
```

Ubuntu 22.04

```
for ver in 5.5.3 5.6.1 5.7; do
echo "deb [arch=amd64 signed-by=/etc/apt/trusted.gpg.d/rocm-keyring.gpg] https://repo.radeon.com/rocm/apt/
↪$ver jammy main" | sudo tee /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' | sudo tee /etc/apt/preferences.d/rocm-
↪pin-600
sudo apt update
```

Red Hat Enterprise Linux

RHEL 8

```
for ver in 5.5.3 5.6.1 5.7; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/rhel8/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

RHEL 9

```
for ver in 5.5.3 5.6.1 5.7; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/rhel9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

SUSE Linux Enterprise Server

```
for ver in 5.5.3 5.6.1 5.7; do
sudo tee --append /etc/zypp/repos.d/rocm.repo <<EOF
name=rocm
baseurl=https://repo.radeon.com/rocm/zyp/$ver/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

(continues on next page)

(continued from previous page)

```
EOF
done
sudo zypper ref
```

3.7.2.4.2 Install packages

Use the installer script as given below:

```
sudo amdgpu-install --usecase=rocm --rocmrelease=<release-number-1>
sudo amdgpu-install --usecase=rocm --rocmrelease=<release-number-2>
sudo amdgpu-install --usecase=rocm --rocmrelease=<release-number-3>
```

Following are examples of ROCm multi-version installation. The kernel-mode driver, associated with the ROCm release 5.7, will be installed as its latest release in the list.

```
sudo amdgpu-install --usecase=rocm --rocmrelease=5.7
sudo amdgpu-install --usecase=rocm --rocmrelease=5.6.1
sudo amdgpu-install --usecase=rocm --rocmrelease=5.5.3
```

3.7.2.5 Additional options

3.7.2.5.1 Unattended installation

Adding -y as a parameter to amdgpu-install skips user prompts (for automation). Example: amdgpu-install -y --usecase=rocm

3.7.2.5.2 Skipping kernel mode driver installation

The installer script tries to install the kernel mode driver along with the requested use cases. This might be unnecessary as in the case of docker containers or you may wish to keep a specific version when using multi-version installation, and not have the last installed version overwrite the kernel mode driver.

To skip the installation of the kernel-mode driver add the --no-dkms option when calling the installer script.

3.7.3 Upgrading with the Installer Script (Linux)

The upgrade procedure with the installer script is exactly the same as installing for 1st time use. Refer to the Installation with install script section on the exact procedure to follow.

3.7.4 Installer Script Uninstallation (Linux)

To uninstall all ROCm packages and the kernel-mode driver the following commands can be used.

Uninstalling Single-Version Install

```
sudo amdgpu-install --uninstall
```

Uninstalling a Specific ROCm Release

```
sudo amdgpu-install --uninstall --rocmrelease=<release-number>
```

Uninstalling all ROCm Releases

```
sudo amdgpu-install --uninstall --rocmrelease=all
```

CHAPTER

FOUR

QUICK START (WINDOWS)

The steps to install the HIP SDK for Windows are described in this document.

4.1 System Requirements

The HIP SDK is supported on Windows 10 and 11. The HIP SDK may be installed on a system without AMD GPUs to use the build toolchains. To run HIP applications, a compatible GPU is required. Please see the supported GPU guide for more details.

4.2 HIP SDK Installation

4.2.1 Download the installer

Download the installer from the HIP-SDK download page.

4.2.2 Launching the installer

To launch the AMD HIP SDK Installer, click the Setup icon shown in Fig. 4.1.

Fig. 4.1: Setup Icon

The installer requires Administrator Privileges, so you may be greeted with a User Access Control (UAC) pop-up. Click Yes.

The installer executable will temporarily extract installer packages to C:\AMD which it will remove after installation completes. This extraction is signified by the “Initializing install” window in Fig. 4.4.

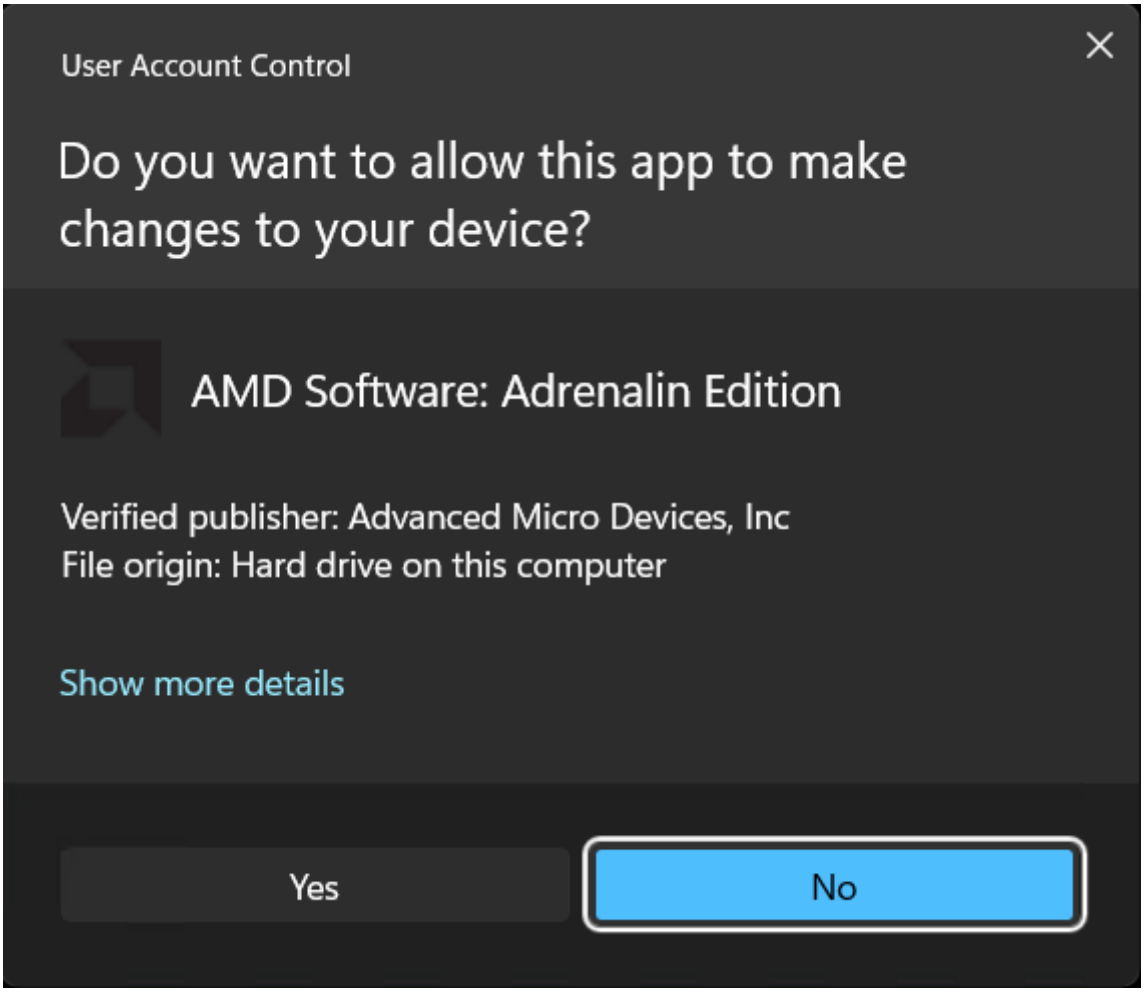


Fig. 4.2: User Access Control pop-up

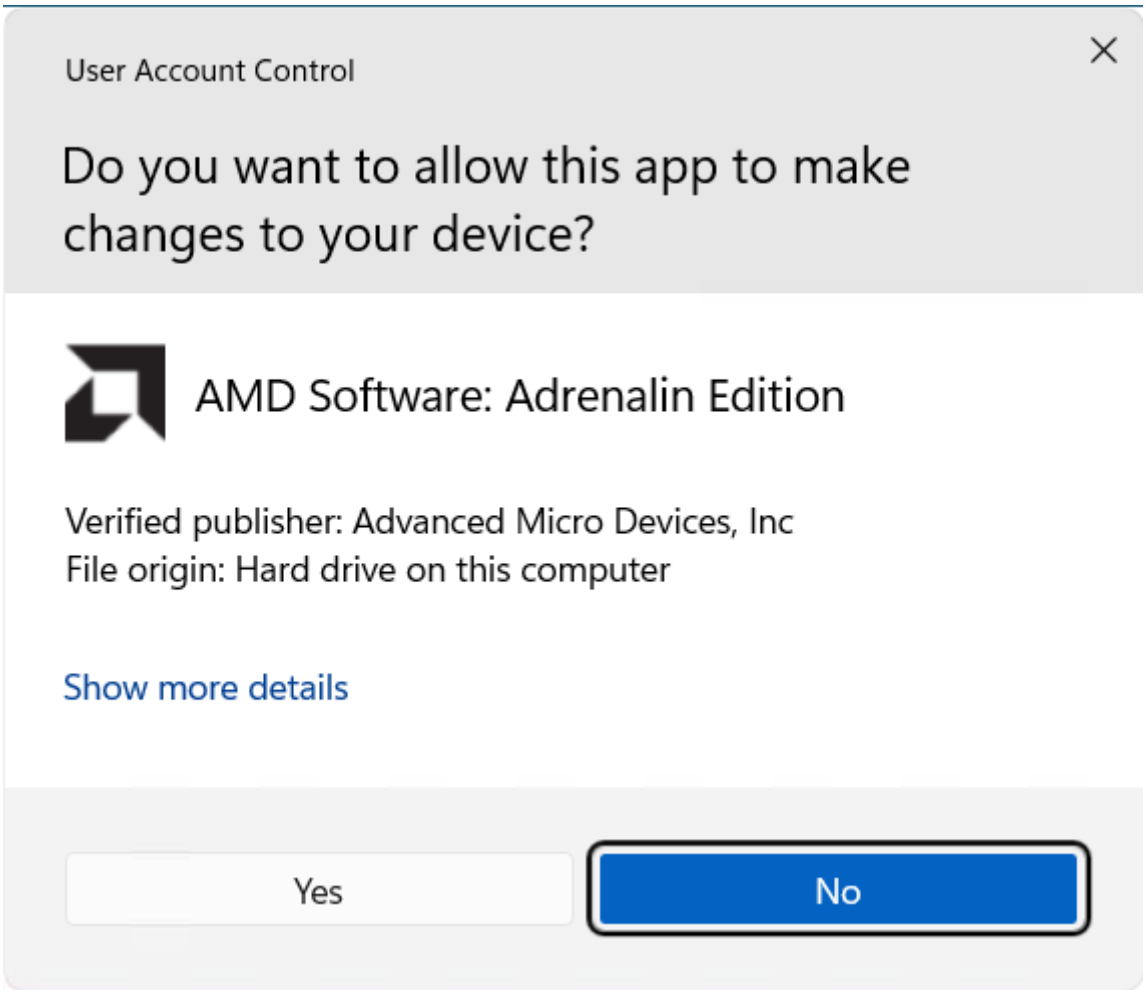


Fig. 4.3: User Access Control pop-up

Fig. 4.4: Installer initialization window

The installer will then detect your system configuration as per Fig. 4.5 to decide, which installable components are applicable to your system.

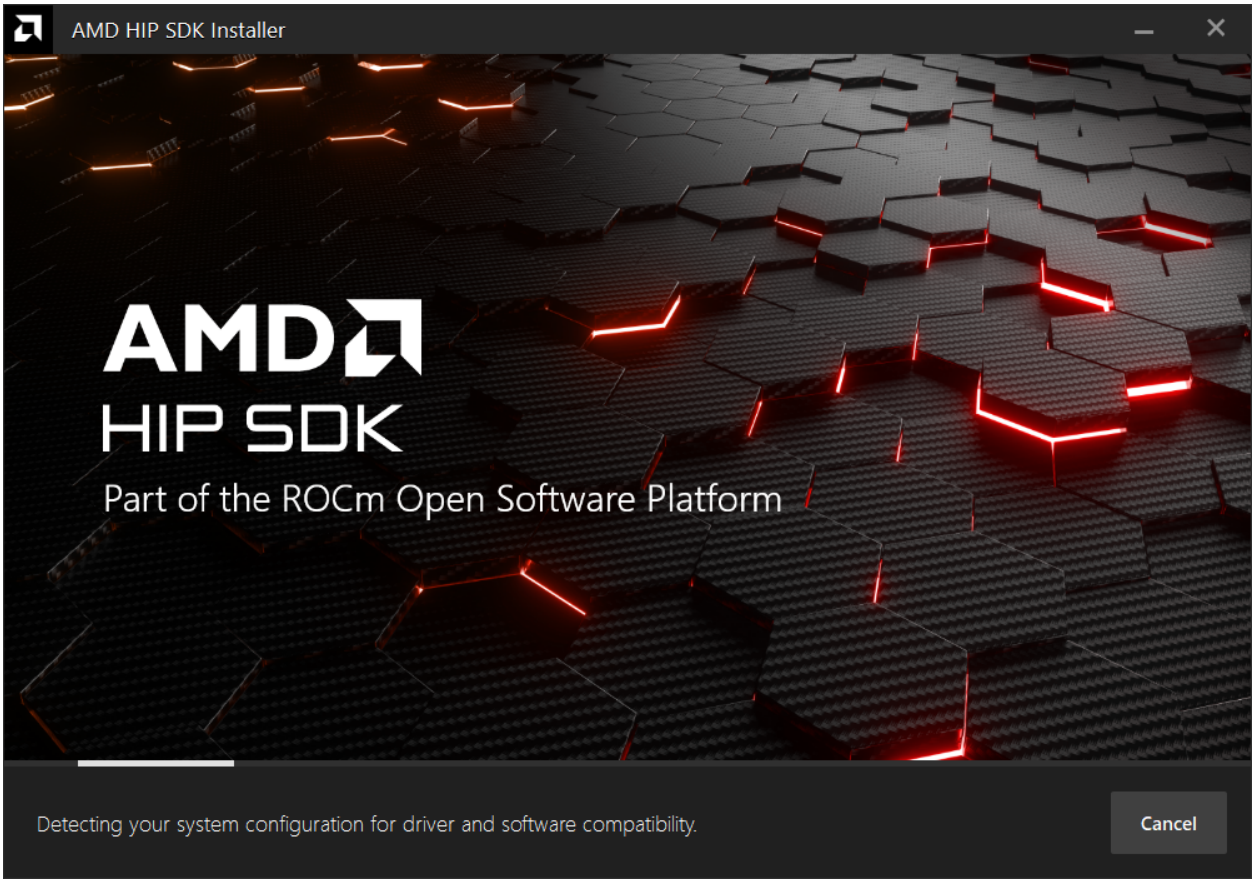


Fig. 4.5: Installer initialization window.

4.2.3 Customizing the install

When the installer launches, it displays a window that lets the user customize the installation. By default, all components are selected for installation. Refer to Fig. 4.6 for an instance when the Select All option is turned on.

4.2.3.1 HIP SDK Installer

The HIP SDK installation options are listed in Table 4.1.

Table 4.1: HIP SDK Components for Installation

HIP Components	Install Type	Additional Options
HIP SDK Core	5.5.0	Install location
HIP Libraries	Full, Partial, None	Runtime, Development (Libs and headers)
HIP Runtime Compiler	Full, Partial, None	Runtime, Development (Headers)
HIP Ray Tracing	Full, Partial, None	Runtime, Development (Headers)
Visual Studio Plugin	Full, Partial, None	Visual Studio 2017, 2019, 2022 Plugin

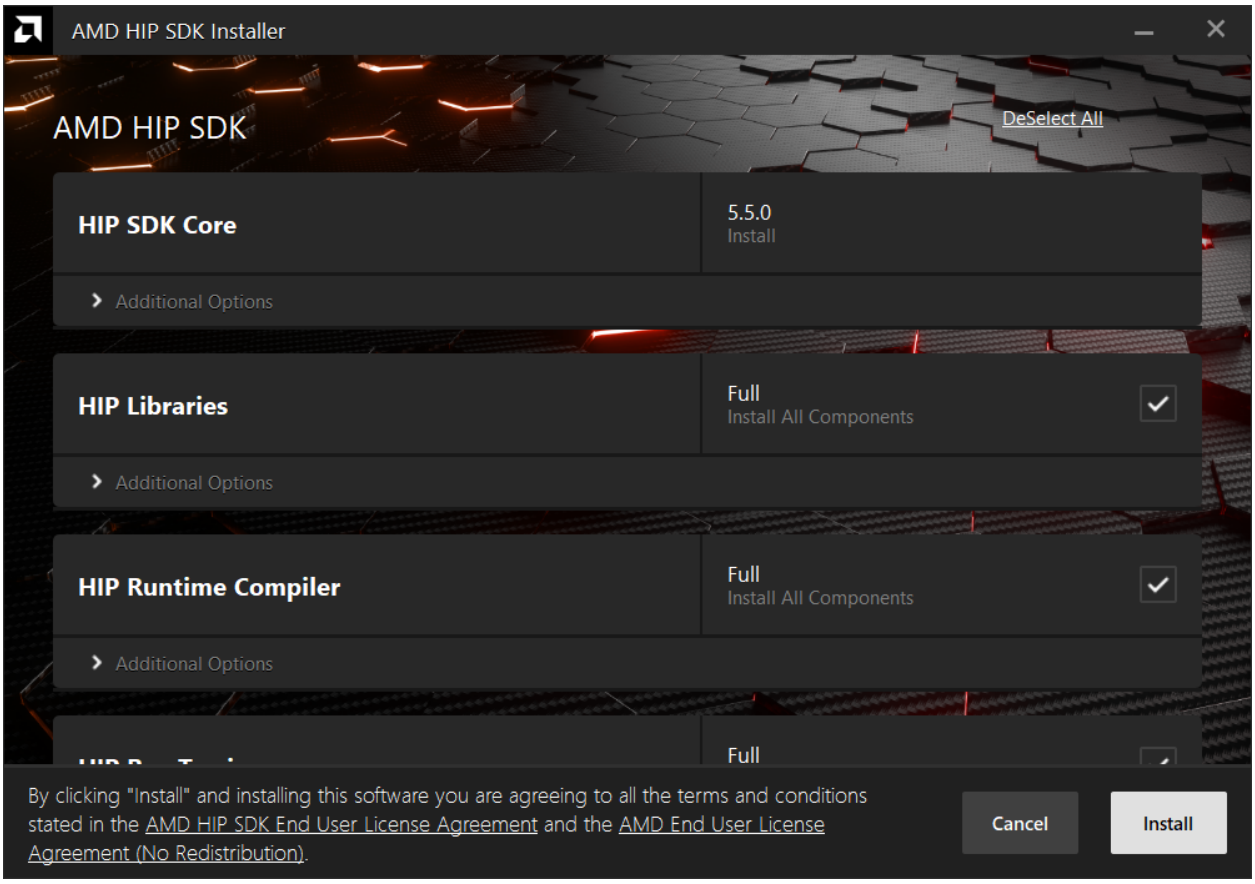


Fig. 4.6: Installer initialization window.

Note: The Select/DeSelect All option only applies to the installation of HIP SDK components. To install the bundled AMD Display Driver, manually select the install type.

Tip: Should you only wish to install a few select components, DeSelecting All and then picking the individual components may be more convenient.

4.2.3.2 AMD Display Driver

The HIP SDK installer bundles an AMD Radeon Software PRO 23.10 installer. The supported install options are summarized by Table 4.2:

Table 4.2: AMD Display Driver Install Options

Install Option	Description
Install Location	Location on disk to store driver files.
Install Type	The breadth of components to be installed. Refer to Table 4.3 for details.
Factory Reset (Optional)	A Factory Reset will remove all prior versions of AMD HIP SDK and drivers. You will not be able to roll back to previously installed drivers.

Table 4.3: AMD Display Driver Install Types

Install Type	Description
Full Install	Provides all AMD Software features and controls for gaming, recording, streaming, and tweaking the performance on your graphics hardware.
Minimal Install	Provides only the basic controls for AMD Software features and does not include advanced features such as performance tweaking or recording and capturing content.
Driver Only	Provides no user interface for AMD Software features.

Note: You must perform a system restart for a complete installation of the Display Driver.

4.2.4 Installing Components

Please wait for the installation to complete during as shown in Fig. 4.7.

Fig. 4.7: Installation Progress

4.2.5 Installation Complete

Once the installation is complete, the installer window may prompt you for a system restart. Click Restart at the lower right corner, shown in Fig. 4.8

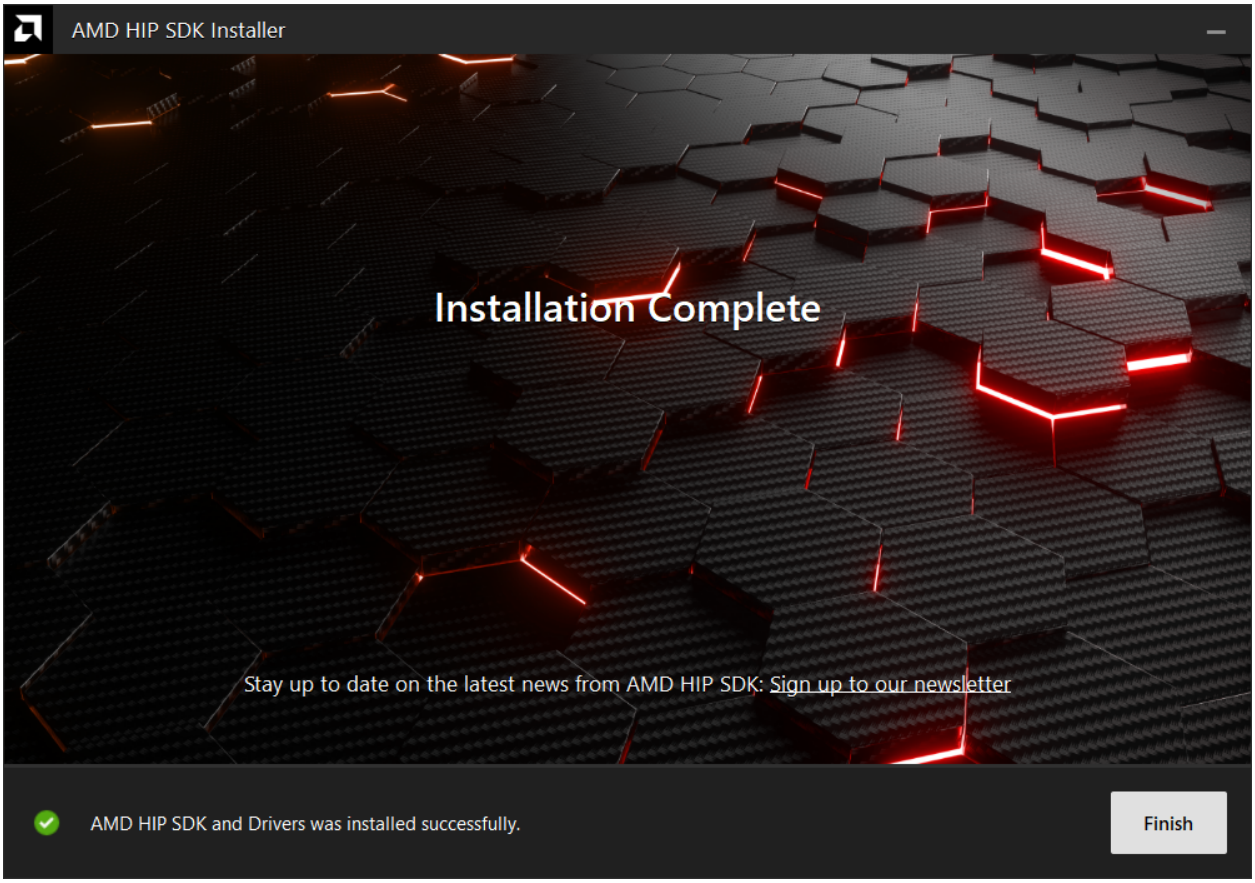


Fig. 4.8: Installation Complete

Error: Should the installer terminate due to unexpcted circumstances, or the user forcibly terminates the installer, the temporary directory created under C:\AMD may be safely removed. Installed components will not depend on this folder (unless the user specifies C:\AMD as an install folder explicitly).

4.3 Uninstallation

All components, except visual studio plug-in should be uninstalled through control panel -> Add/Remove Program. For visual studio extension uninstallation, please refer to https://github.com/ROCm-Developer-Tools/HIP-VS/blob/master/README.md. Uninstallation of the HIP SDK components can be done through the Windows Settings app. Navigate to “Apps > Installed apps”, click the “…” on the far right next to the component to uninstall, and click “Uninstall”.

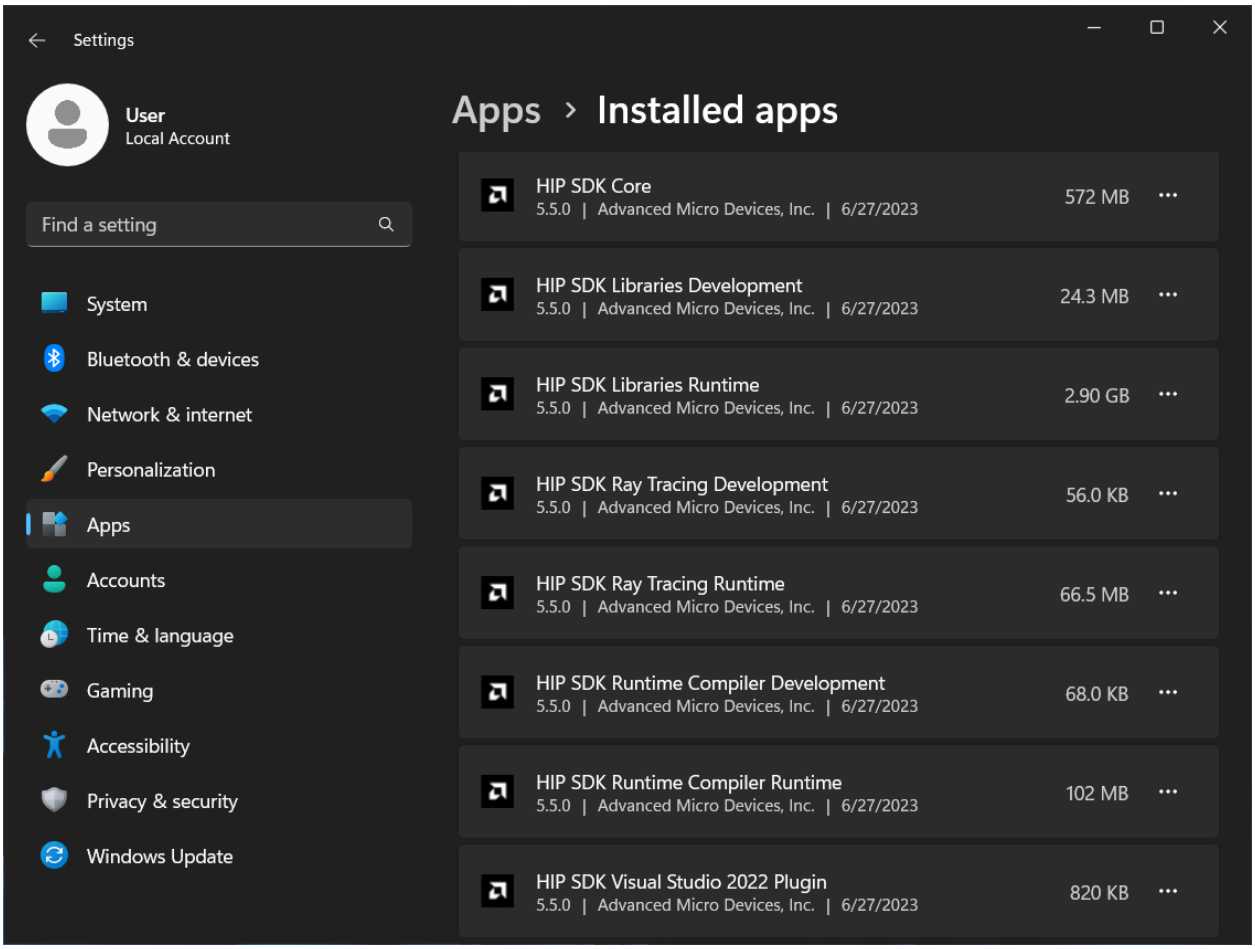


Fig. 4.9: Removing the SDK via the Setting app

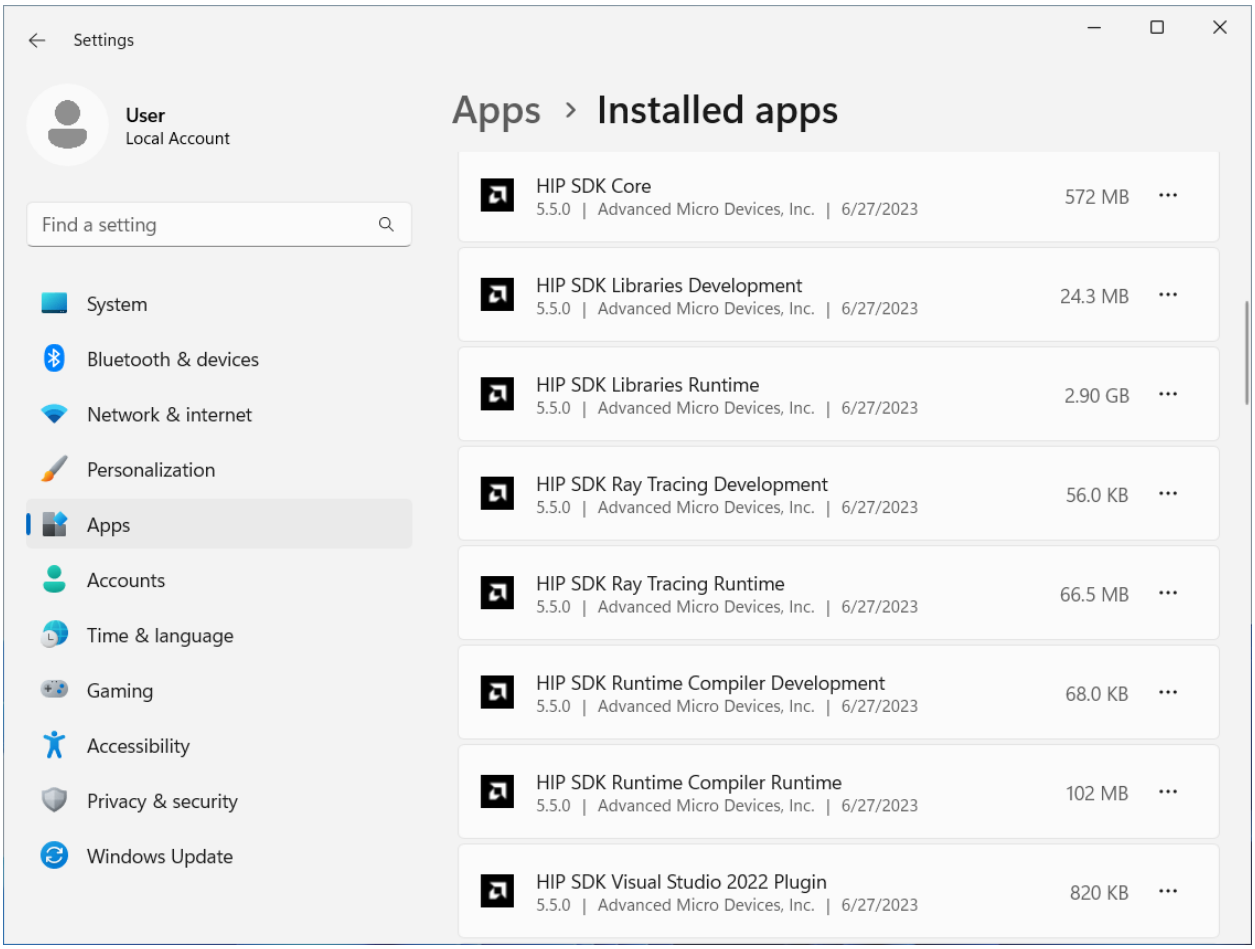


Fig. 4.10: Removing the SDK via the Setting app

CHAPTER

FIVE

INSTALL ROCM (HIP SDK) ON WINDOWS

Start with Quick Start (Windows) or follow the detailed instructions below.

5.1 Prepare to Install

Prerequisites The prerequisites page lists the required steps to verify that the system supports ROCm.

5.2 Choose your install method

Graphical Installation Use the graphical front-end of the installer.

Command Line Installation Use the command line front-end of the installer.

5.3 Post Installation

ROCm-Examples Learn how to use ROCm with descriptive examples for novice to intermediate users.

Windows App Deployment Guidelines Discusses strategies on how to bundle HIP libraries with an end user application.

5.4 See Also

- GPU Support and OS Compatibility (Linux)

5.5 Installation Prerequisites (Windows)

You must perform the following steps before installing ROCm and check if the system meets all the requirements to proceed with the installation.

5.5.1 Confirm the System Is Supported

The ROCm installation is supported only on specific host architectures, Windows Editions and update versions.

5.5.1.1 Check the Windows Editions and Update Version on Your System

This section discusses obtaining information about the host architecture, Windows Edition and update version.

5.5.1.1.1 Command Line Check

Verify the Windows Edition using the following steps:

1. To obtain the Linux distribution information, type the following command on your system from a PowerShell Command Line Interface (CLI):

```
Get-ComputerInfo | Format-Table CsSystemType,OSName,OSDisplayVersion
```

2. Confirm that the obtained information matches with those listed in Supported SKUs.

   Example: Running the command above on a Windows system may result in the following output:

```
CsSystemType OsName                    OSDisplayVersion
------------ ------                    ----------------
x64-based PC Microsoft Windows 11 Pro 22H2
```

5.5.1.1.2 Graphical Check

1. Open the Setting app.

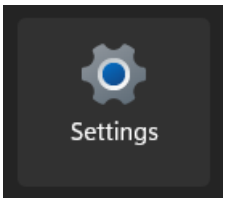


Fig. 5.1: Windows Settings app icon

2. Navigate to System > About.
3. Confirm that the obtained information matches with those listed in Supported SKUs.

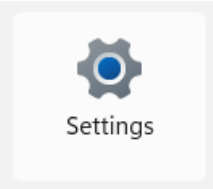


Fig. 5.2: Windows Settings app icon

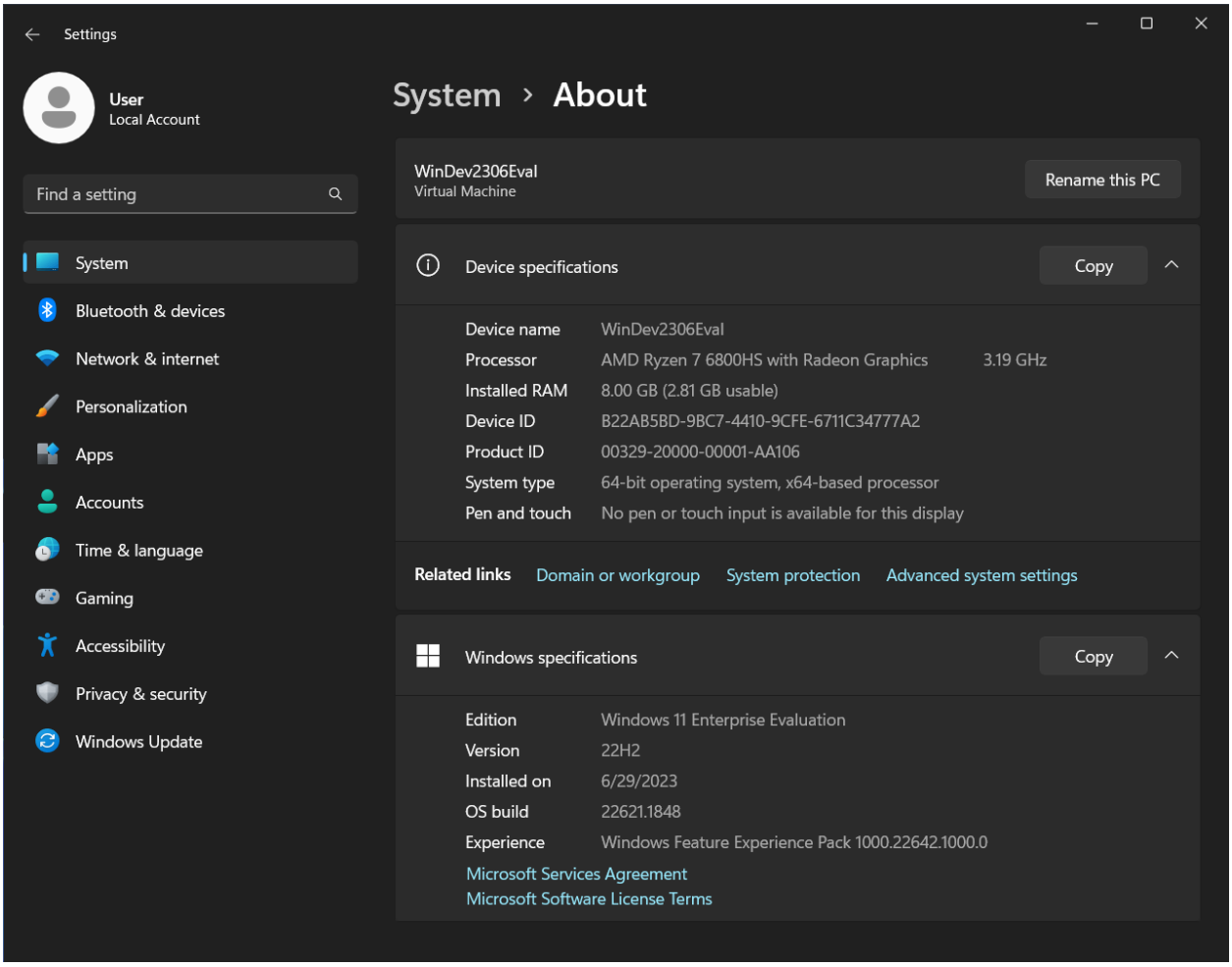


Fig. 5.3: Settings > About page

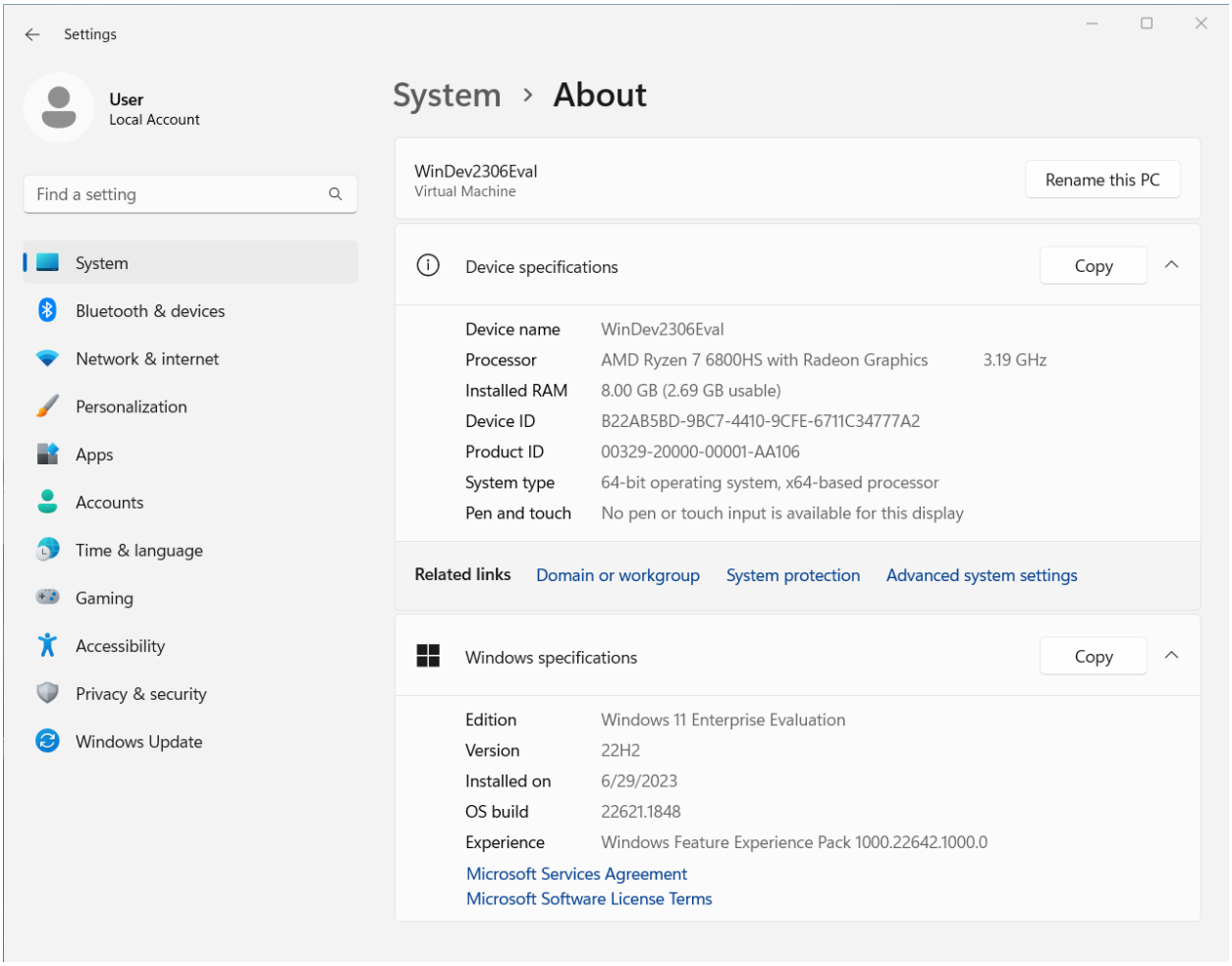


Fig. 5.4: Settings > About page

5.6 Graphical Installation

Install How to install ROCm?

Upgrade Instructions for upgrading an existing ROCm installation.

Uninstall Steps for removing ROCm packages and libraries.

5.6.1 See Also

- GPU Support and OS Compatibility (Linux)

5.6.2 Installation Using the Graphical Interface

The steps to install the HIP SDK for Windows are described in this document.

5.6.2.1 System Requirements

The HIP SDK is supported on Windows 10 and 11. The HIP SDK may be installed on a system without AMD GPUs to use the build toolchains. To run HIP applications, a compatible GPU is required. Please see the supported GPU guide for more details.

5.6.2.2 HIP SDK Installation

5.6.2.2.1 Download the installer

Download the installer from the HIP-SDK download page.

5.6.2.2.2 Launching the installer

To launch the AMD HIP SDK Installer, click the Setup icon shown in Fig. 4.1.

Fig. 5.5: Setup Icon

The installer requires Administrator Privileges, so you may be greeted with a User Access Control (UAC) pop-up. Click Yes.

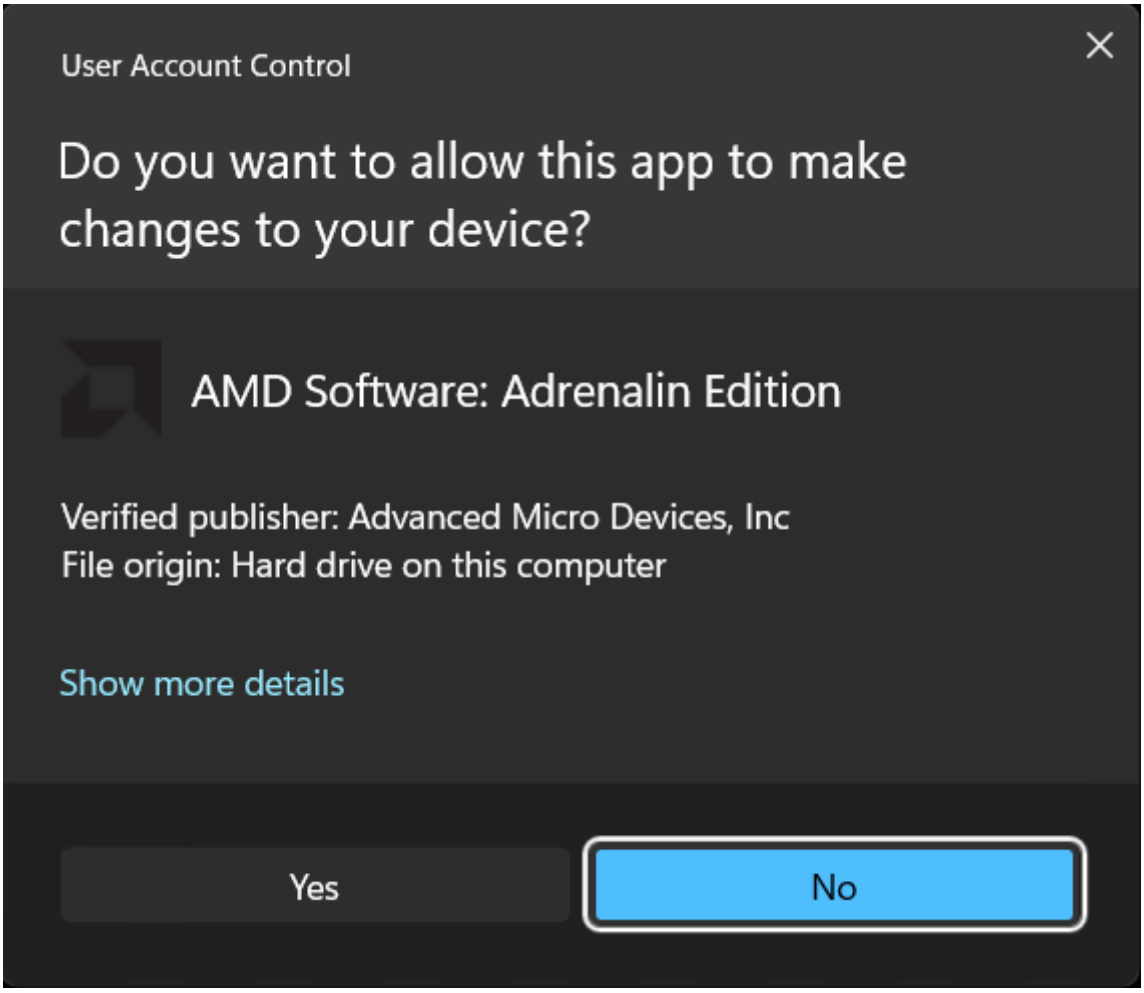


Fig. 5.6: User Access Control pop-up

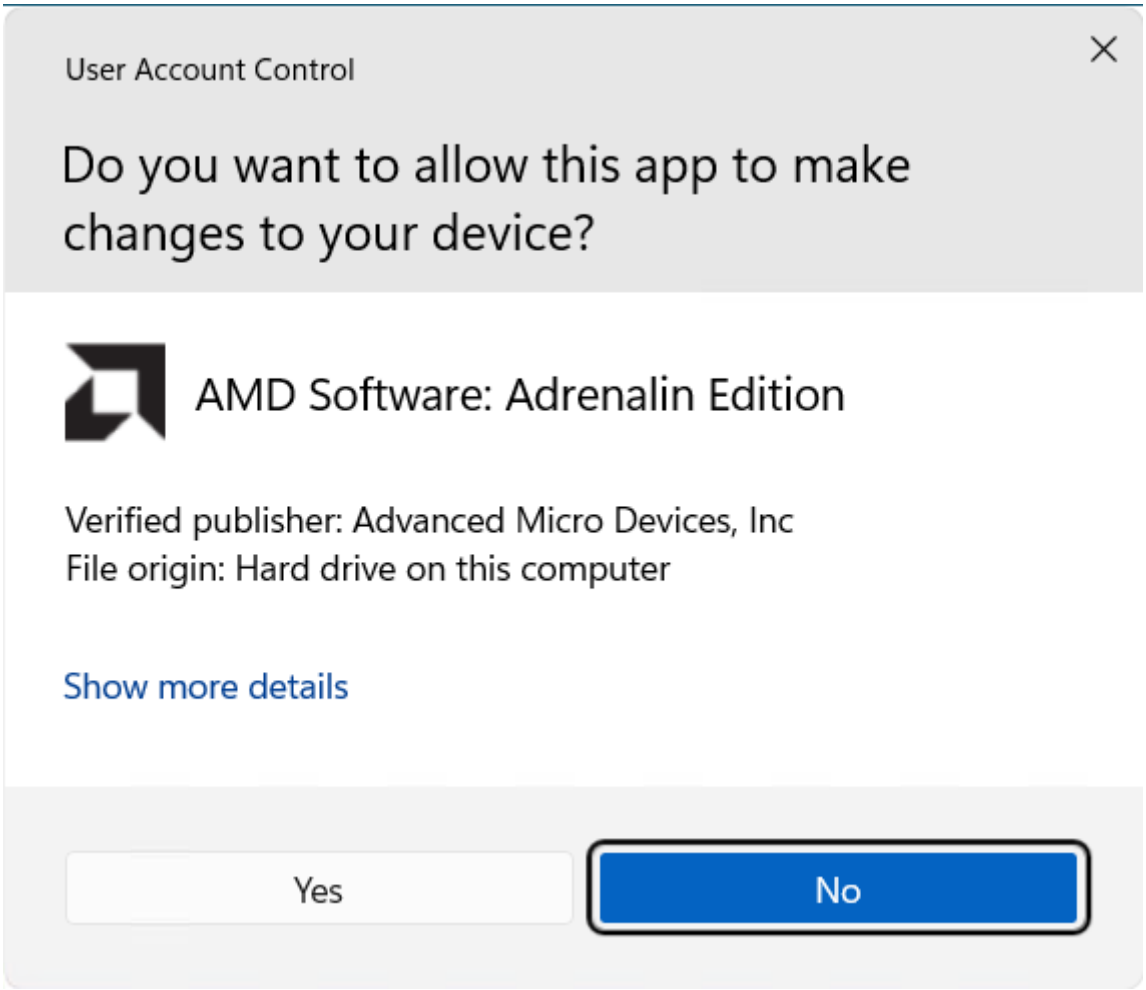


Fig. 5.7: User Access Control pop-up

Fig. 5.8: Installer initialization window

The installer executable will temporarily extract installer packages to C:\AMD which it will remove after installation completes. This extraction is signified by the “Initializing install” window in Fig. 4.4.

The installer will then detect your system configuration as per Fig. 4.5 to decide, which installable components are applicable to your system.

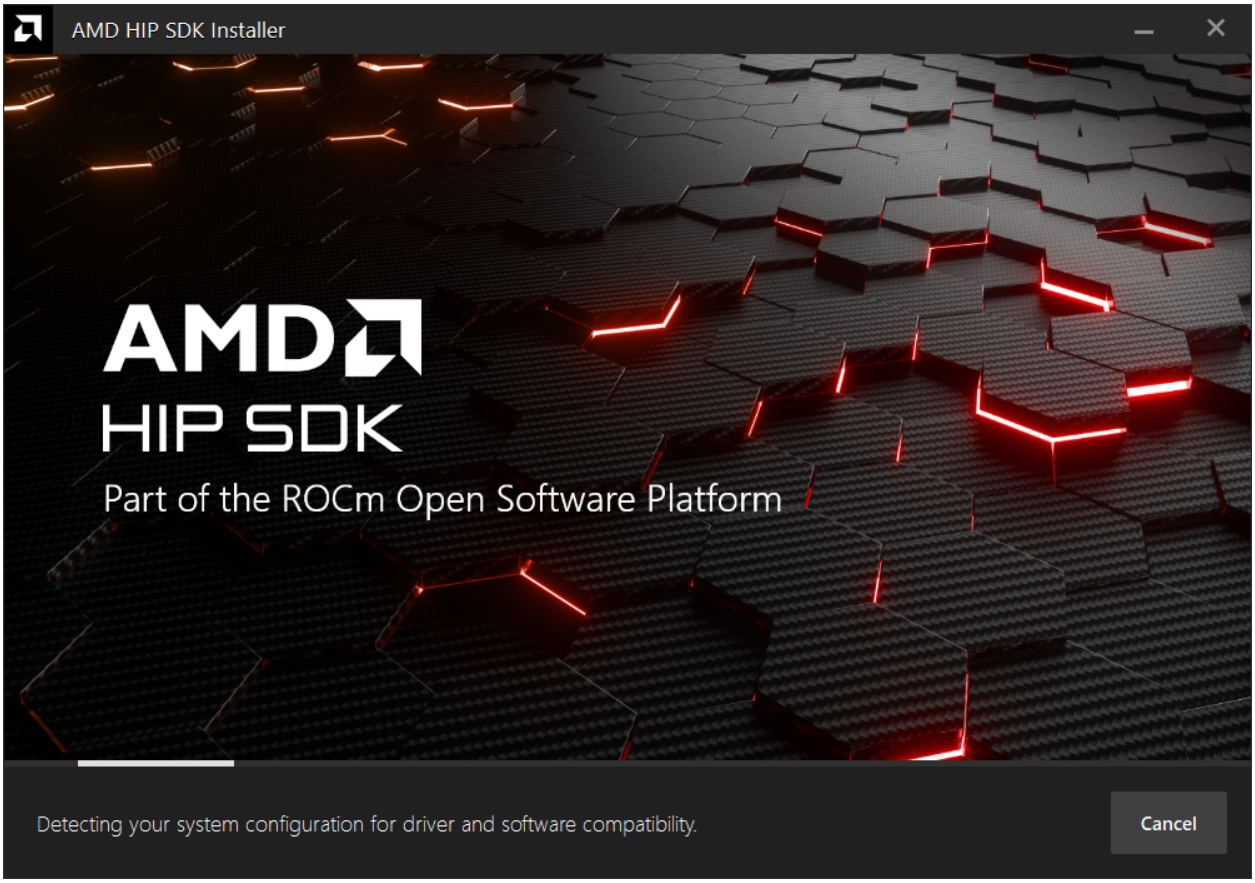


Fig. 5.9: Installer initialization window.

5.6.2.2.3 Customizing the install

When the installer launches, it displays a window that lets the user customize the installation. By default, all components are selected for installation. Refer to Fig. 4.6 for an instance when the Select All option is turned on.

5.6.2.2.3.1 HIP SDK Installer

The HIP SDK installation options are listed in Table 4.1.

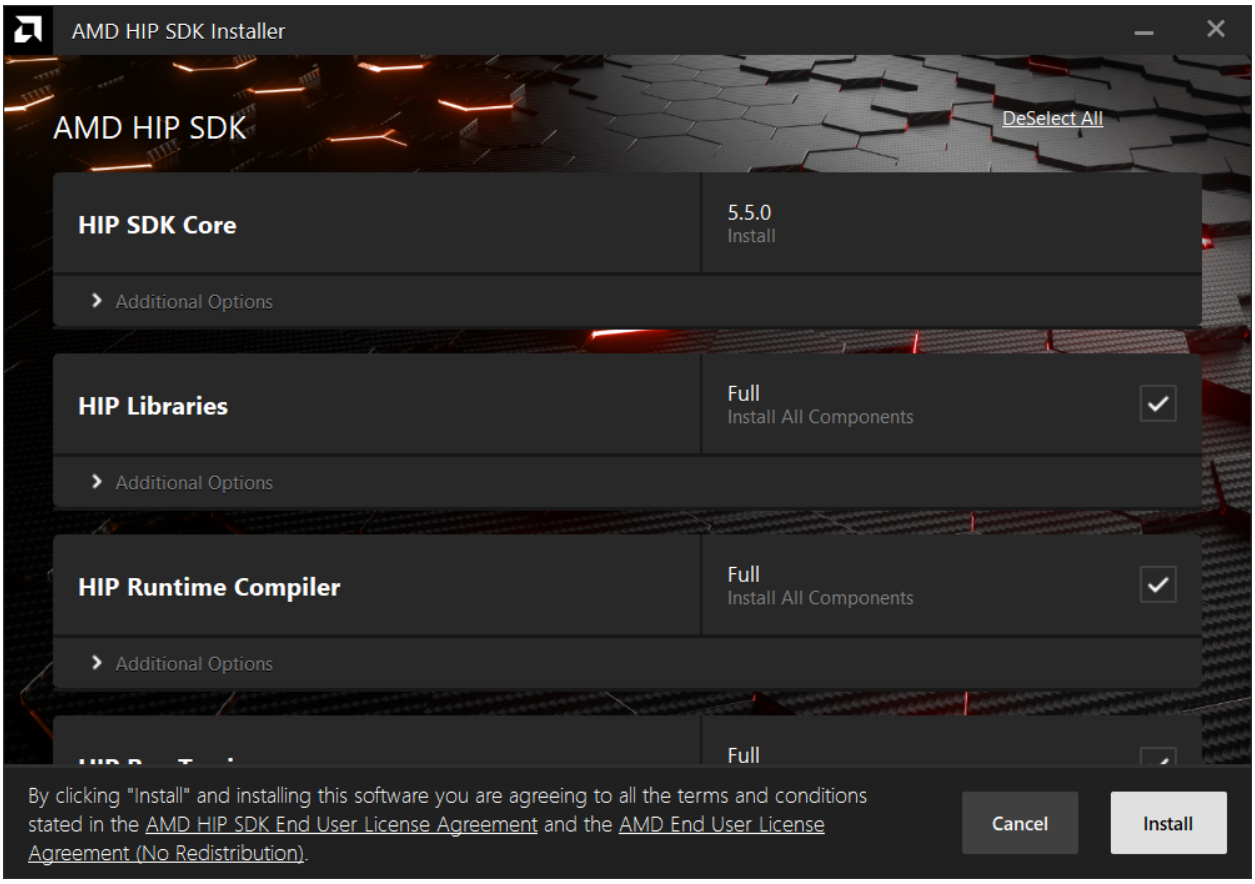


Fig. 5.10: Installer initialization window.

Table 5.1: HIP SDK Components for Installation

HIP Components	Install Type	Additional Options
HIP SDK Core	5.5.0	Install location
HIP Libraries	Full, Partial, None	Runtime, Development (Libs and headers)
HIP Runtime Compiler	Full, Partial, None	Runtime, Development (Headers)
HIP Ray Tracing	Full, Partial, None	Runtime, Development (Headers)
Visual Studio Plugin	Full, Partial, None	Visual Studio 2017, 2019, 2022 Plugin

Note: The Select/DeSelect All option only applies to the installation of HIP SDK components. To install the bundled AMD Display Driver, manually select the install type.

Tip: Should you only wish to install a few select components, DeSelecting All and then picking the individual components may be more convenient.

5.6.2.2.3.2 AMD Display Driver

The HIP SDK installer bundles an AMD Radeon Software PRO 23.10 installer. The supported install options are summarized by Table 4.2:

Table 5.2: AMD Display Driver Install Options

Install Option	Description
Install Location	Location on disk to store driver files.
Install Type	The breadth of components to be installed. Refer to Table 4.3 for details.
Factory Reset (Optional)	A Factory Reset will remove all prior versions of AMD HIP SDK and drivers. You will not be able to roll back to previously installed drivers.

Table 5.3: AMD Display Driver Install Types

Install Type	Description
Full Install	Provides all AMD Software features and controls for gaming, recording, streaming, and tweaking the performance on your graphics hardware.
Minimal Install	Provides only the basic controls for AMD Software features and does not include advanced features such as performance tweaking or recording and capturing content.
Driver Only	Provides no user interface for AMD Software features.

Note: You must perform a system restart for a complete installation of the Display Driver.

5.6.2.2.4 Installing Components

Please wait for the installation to complete during as shown in Fig. 4.7.

Fig. 5.11: Installation Progress

5.6.2.2.5 Installation Complete

Once the installation is complete, the installer window may prompt you for a system restart. Click Restart at the lower right corner, shown in Fig. 4.8

Error: Should the installer terminate due to unexpcted circumstances, or the user forcibly terminates the installer, the temporary directory created under C:\AMD may be safely removed. Installed components will not depend on this folder (unless the user specifies C:\AMD as an install folder explicitly).

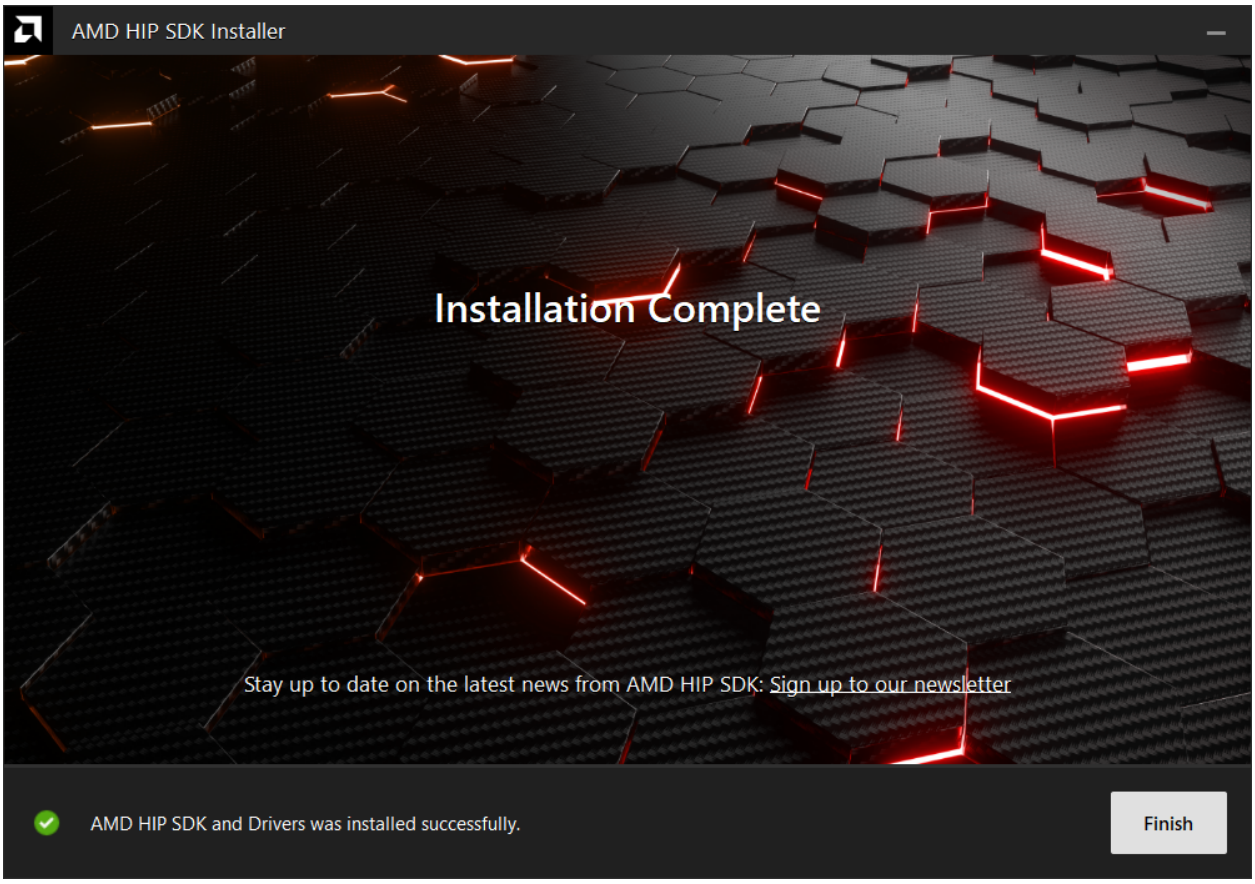


Fig. 5.12: Installation Complete

5.6.3 Upgrading Using the Graphical Interface

The steps to upgrade an existing HIP SDK installation for Windows are described in this document.

5.6.4 Uninstallation Using the Graphical Interface

The steps to uninstall the HIP SDK for Windows are described in this document.

5.6.4.1 Uninstallation

All components, except visual studio plug-in should be uninstalled through control panel -> Add/Remove Program. For visual studio extension uninstallation, please refer to https://github.com/ROCm-Developer-Tools/HIP-VS/blob/master/README.md. Uninstallation of the HIP SDK components can be done through the Windows Settings app. Navigate to "Apps > Installed apps", click the "..." on the far right next to the component to uninstall, and click "Uninstall".

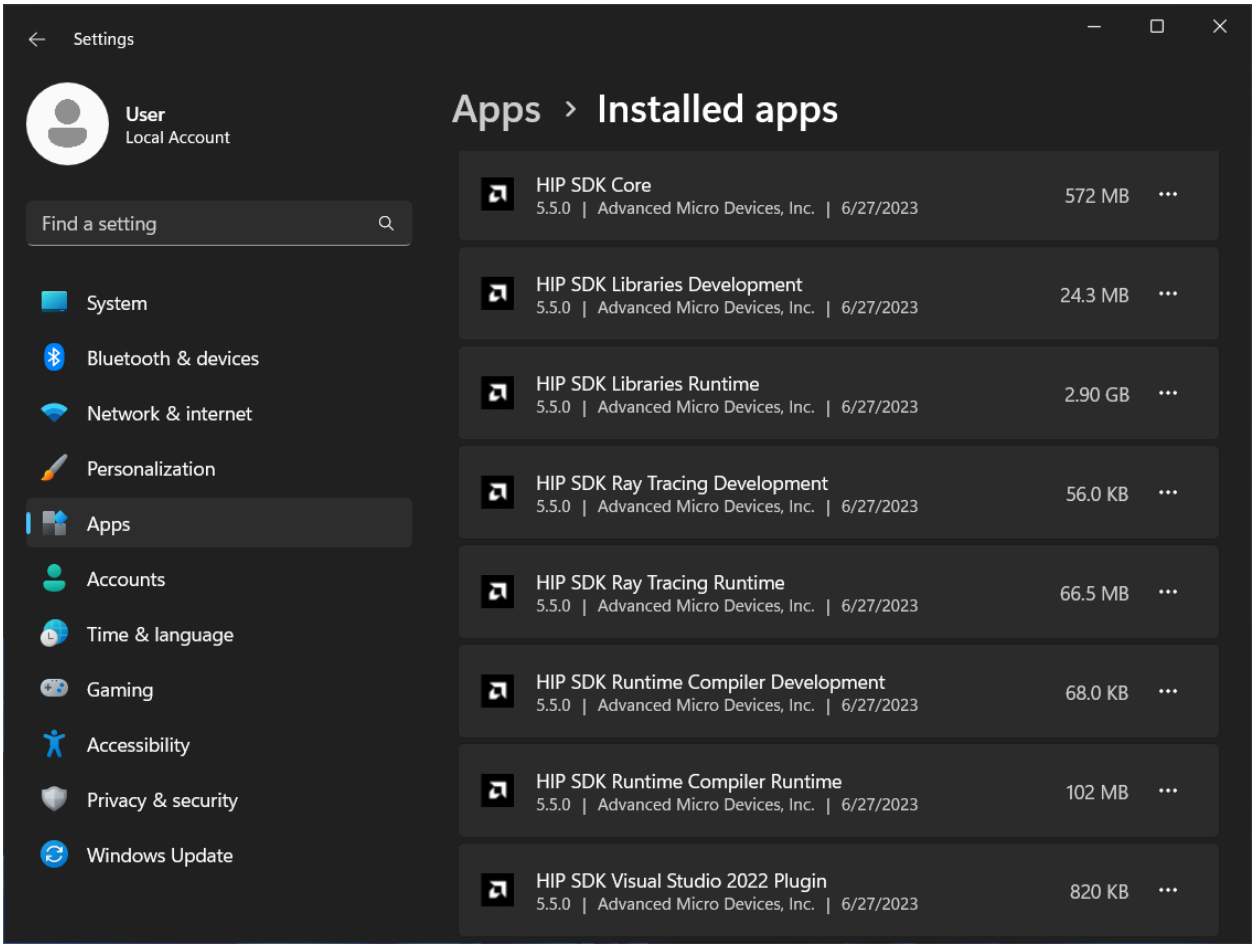


Fig. 5.13: Removing the SDK via the Setting app

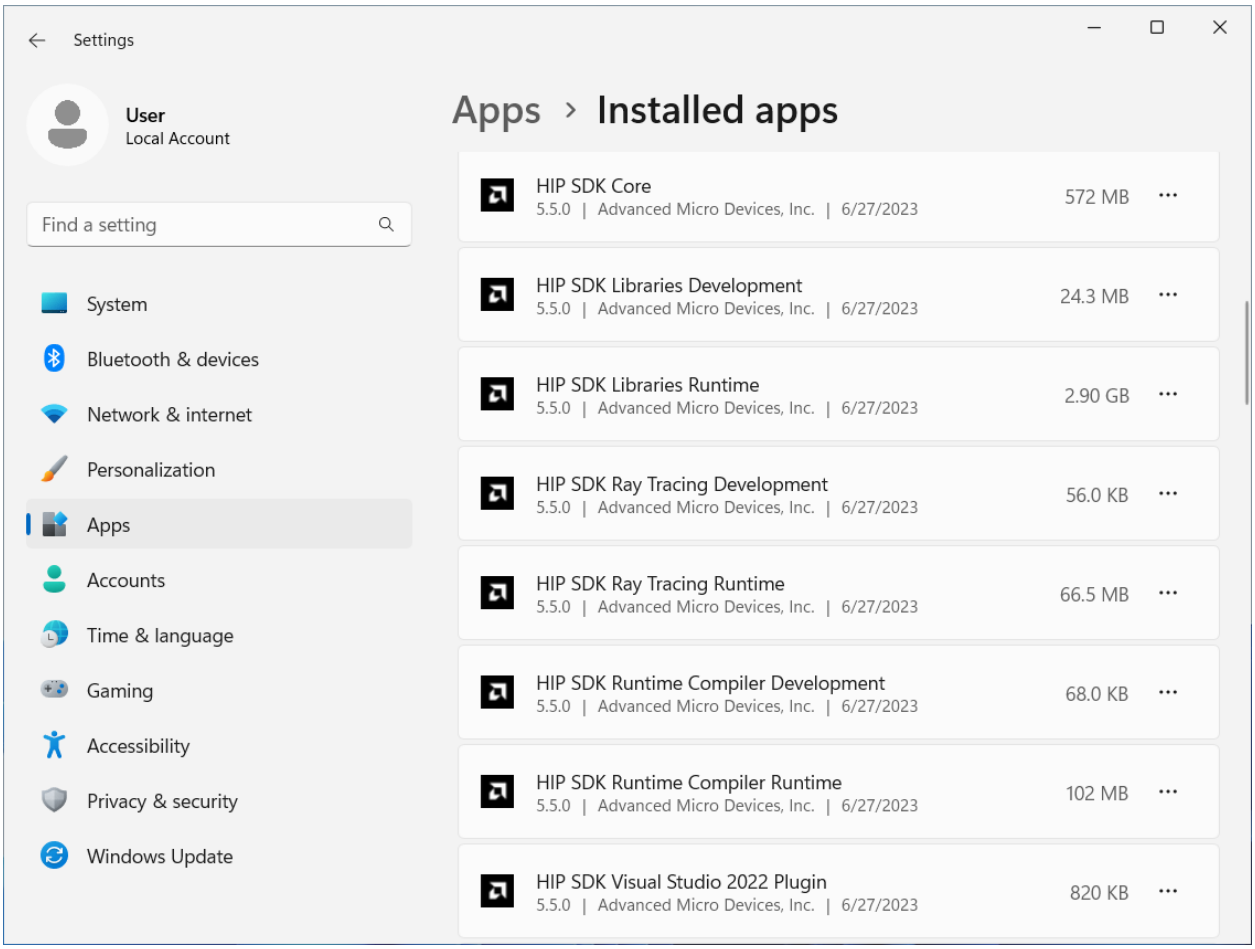


Fig. 5.14: Removing the SDK via the Setting app

5.7 Command Line Installation

Install How to install ROCm?

Upgrade Instructions for upgrading an existing ROCm installation.

Uninstall Steps for removing ROCm packages and libraries.

5.7.1 See Also

- GPU Support and OS Compatibility (Linux)

5.7.2 Installation Using the Command Line Interface

The steps to install the HIP SDK for Windows are described in this document.

5.7.2.1 System Requirements

The HIP SDK is supported on Windows 10 and 11. The HIP SDK may be installed on a system without AMD GPUs to use the build toolchains. To run HIP applications, a compatible GPU is required. Please see the supported GPU guide for more details.

5.7.2.2 HIP SDK Installation

The command line installer is the same executable which is used by the graphical front-end. Download the installer from the HIP-SDK download page. The options supported by the command line interface are summarized in Table 5.5.

Table 5.4: HIP SDK Command Line Options

Install Option	Description
-install	Command used to install packages, both driver and applications. No output to the screen.
-install -boot	Silent install with auto reboot.
-install -log <absolute path>	Write install result code to the specified log file. The specified log file must be on a local machine. Double quotes are needed if there are spaces in the log file path.
-uninstall	Command to uninstall all packages installed by this installer on the system. There is no option to specify which packages to uninstall.
-uninstall -boot	Silent uninstall with auto reboot.
/? or /help	Shows a brief description of all switch commands.

Note: Unlike the graphical installer, the command line interface doesn't support selectively installing parts of the SDK bundle. It's all or nothing.

5.7.2.2.1 Launching the Installer From the Command Line

The installer is still a graphical application with a WinMain entry point, even when called on the command line. This means that the application lifetime is tied to a window, even on headless systems where that window may not be visible. To launch the installer from PowerShell that will block until the installer exits, one may use the following pattern:

```
Start-Process $InstallerExecutable -ArgumentList $InstallerArgs -NoNewWindow -Wait
```

Important: Running the installer requires Administrator Privileges.

For example, installing all components and

```
Start-Process ~\Downloads\Setup.exe -ArgumentList '-install','-log',"${env:USERPROFILE}\installer_log.txt" -
↪NoNewWindow -Wait
```

5.7.3 Upgrading Using the Graphical Interface

The steps to uninstall the HIP SDK for Windows are described in this document.

5.7.3.1 HIP SDK Upgrade

To upgrade an existing installation of the HIP SDK without preserving the previous version, first uninstall it, then install the new version following the instructions in Uninstallation Using the Command Line Interface and Installation Using the Command Line Interface using the old and new installers respectively.

To upgrade by installing both versions side-by-side, just run the installer of the newer version.

5.7.4 Uninstallation Using the Command Line Interface

The steps to uninstall the HIP SDK for Windows are described in this document.

5.7.4.1 HIP SDK Uninstallation

The command line installer is the same executable which is used by the graphical front-end. The options supported by the command line interface are summarized in Table 5.5.

Table 5.5: HIP SDK Command Line Options

Install Option	Description
-install	Command used to install packages, both driver and applications. No output to the screen.
-install -boot	Silent install with auto reboot.
-install -log <absolute path>	Write install result code to the specified log file. The specified log file must be on a local machine. Double quotes are needed if there are spaces in the log file path.
-uninstall	Command to uninstall all packages installed by this installer on the system. There is no option to specify which packages to uninstall.
-uninstall -boot	Silent uninstall with auto reboot.
/? or /help	Shows a brief description of all switch commands.

Note: Unlike the graphical installer, the command line interface doesn't support selectively installing parts of the SDK bundle. It's all or nothing.

5.7.4.1.1 Launching the Installer From the Command Line

The installer is still a graphical application with a WinMain entry point, even when called on the command line. This means that the application lifetime is tied to a window, even on headless systems where that window may not be visible. To launch the installer from PowerShell that will block until the installer exits, one may use the following pattern:

```
Start-Process $InstallerExecutable -ArgumentList $InstallerArgs -NoNewWindow -Wait
```

Important: Running the installer requires Administrator Privileges.

For example, uninstalling all components and

```
Start-Process ~\Downloads\Setup.exe -ArgumentList '-uninstall' -NoNewWindow -Wait
```

CHAPTER

SIX

DEPLOY ROCM DOCKER CONTAINERS

6.1 Prerequisites

Docker containers share the kernel with the host operating system, therefore the ROCm kernel-mode driver must be installed on the host. Please refer to using-the-package-manager on installing amdgpu-dkms. The other user-space parts (like the HIP-runtime or math libraries) of the ROCm stack will be loaded from the container image and don't need to be installed to the host.

6.2 Accessing GPUs in containers

In order to access GPUs in a container (to run applications using HIP, OpenCL or OpenMP offloading) explicit access to the GPUs must be granted.

The ROCm runtimes make use of multiple device files:

- /dev/kfd: the main compute interface shared by all GPUs
- /dev/dri/renderD<node>: direct rendering interface (DRI) devices for each GPU. <node> is a number for each card in the system starting from 128.

Exposing these devices to a container is done by using the --device option, i.e. to allow access to all GPUs expose /dev/kfd and all /dev/dri/renderD devices:

```
docker run --device /dev/kfd --device /dev/renderD128 --device /dev/renderD129 ...
```

More conveniently, instead of listing all devices, the entire /dev/dri folder can be exposed to the new container:

```
docker run --device /dev/kfd --device /dev/dri
```

Note that this gives more access than strictly required, as it also exposes the other device files found in that folder to the container.

6.2.1 Restricting a container to a subset of the GPUs

If a /dev/dri/renderD device is not exposed to a container then it cannot use the GPU associated with it; this allows to restrict a container to any subset of devices.

For example to allow the container to access the first and third GPU start it like:

```
docker run --device /dev/kfd --device /dev/dri/renderD128 --device /dev/dri/renderD130 <image>
```

6.2.2 Additional Options

The performance of an application can vary depending on the assignment of GPUs and CPUs to the task. Typically, numactl is installed as part of many HPC applications to provide GPU/CPU mappings. This Docker runtime option supports memory mapping and can improve performance.

```
--security-opt seccomp=unconfined
```

This option is recommended for Docker Containers running HPC applications.

```
docker run --device /dev/kfd --device /dev/dri --security-opt seccomp=unconfined ...
```

6.3 Docker images in the ROCm ecosystem

6.3.1 Base images

https://github.com/RadeonOpenCompute/ROCm-docker hosts images useful for users wishing to build their own containers leveraging ROCm. The built images are available from Docker Hub. In particular rocm/rocm-terminal is a small image with the prerequisites to build HIP applications, but does not include any libraries.

6.3.2 Applications

AMD provides pre-built images for various GPU-ready applications through its Infinity Hub at https://www.amd.com/en/technologies/infinity-hub. Examples for invoking each application and suggested parameters used for benchmarking are also provided there.

CHAPTER

SEVEN

RELEASE NOTES

The release notes for the ROCm platform.

7.1 ROCm 5.7.0

7.1.1 Release Highlights for ROCm v5.7

ROCm 5.7.0 includes many new features. These include: a new library (hipTensor), and optimizations for rocRAND and MIVisionX. Address sanitizer for host and device code (GPU) is now available as a beta. Note that ROCm 5.7.0 is EOS for MI50. 5.7 versions of ROCm are the last major release in the ROCm 5 series. This release is Linux-only.

Important: The next major ROCm release (ROCm 6.0) will not be backward compatible with the ROCm 5 series. Changes will include: splitting LLVM packages into more manageable sizes, changes to the HIP runtime API, splitting rocRAND and hipRAND into separate packages, and reorganizing our file structure.

7.1.1.1 AMD Instinct™ MI50 End of Support Notice

AMD Instinct MI50, Radeon Pro VII, and Radeon VII products (collectively gfx906 GPUs) will enter maintenance mode starting Q3 2023.

As outlined in 5.6.0, ROCm 5.7 will be the final release for gfx906 GPUs to be in a fully supported state.

- ROCm 6.0 release will show MI50s as “under maintenance” mode for Linux and Windows
- No new features and performance optimizations will be supported for the gfx906 GPUs beyond this major release (ROCm 5.7).
- Bug fixes / critical security patches will continue to be supported for the gfx906 GPUs till Q2 2024 (EOM (End of Maintenance) will be aligned with the closest ROCm release).
- Bug fixes during the maintenance will be made to the next ROCm point release.
- Bug fixes will not be backported to older ROCm releases for gfx906.
- Distro / Operating system updates will continue as per the ROCm release cadence for gfx906 GPUs till EOM.

7.1.1.2 Feature Updates

7.1.1.2.1 Non-hostcall HIP Printf

Current behavior

The current version of HIP printf relies on hostcalls, which, in turn, rely on PCIe atomics. However, PCIe atomics are unavailable in some environments, and, as a result, HIP-printf does not work in those environments. Users may see the following error from runtime (with AMD_LOG_LEVEL 1 and above),

```
Pcie atomics not enabled, hostcall not supported
```

Workaround

The ROCm 5.7 release introduces an alternative to the current hostcall-based implementation that leverages an older OpenCL-based printf scheme, which does not rely on hostcalls/PCIe atomics. Note: This option is less robust than hostcall-based implementation and is intended to be a workaround when hostcalls do not work.

The printf variant is now controlled via a new compiler option -mprintf-kind=. This is supported only for HIP programs and takes the following values,

- "hostcall" – This currently available implementation relies on hostcalls, which require the system to support PCIe atomics. It is the default scheme.
- "buffered" – This implementation leverages the older printf scheme used by OpenCL; it relies on a memory buffer where printf arguments are stored during the kernel execution, and then the runtime handles the actual printing once the kernel finishes execution.

NOTE: With the new workaround,

- The printf buffer is fixed size and non-circular. After the buffer is filled, calls to printf will not result in additional output.
- The printf call returns either 0 (on success) or -1 (on failure, due to full buffer), unlike the hostcall scheme that returns the number of characters printed.

7.1.1.2.2 Beta Release of LLVM Address Sanitizer (ASAN) with the GPU

The ROCm v5.7 release introduces the beta release of LLVM Address Sanitizer (ASAN) with the GPU. The LLVM Address Sanitizer provides a process that allows developers to detect runtime addressing errors in applications and libraries. The detection is achieved using a combination of compiler-added instrumentation and runtime techniques, including function interception and replacement. Until now, the LLVM Address Sanitizer process was only available for traditional purely CPU applications. However, ROCm has extended this mechanism to additionally allow the detection of some addressing errors on the GPU in heterogeneous applications. Ideally, developers should treat heterogeneous HIP and OpenMP applications like pure CPU applications. However, this simplicity has not been achieved yet.

Refer to the documentation on LLVM Address Sanitizer with the GPU at LLVM Address Sanitizer User Guide.

Note: The beta release of LLVM Address Sanitizer for ROCm is currently tested and validated on Ubuntu 20.04.

7.1.1.3 Fixed Defects

The following defects are fixed in ROCm v5.7,

- Test hangs observed in HMM RCCL
- NoGpuTst test of Catch2 fails with Docker
- Failures observed with non-HMM HIP directed catch2 tests with XNACK+
- Multiple test failures and test hangs observed in hip-directed catch2 tests with xnack+

7.1.1.4 HIP 5.7.0

7.1.1.4.1 Optimizations

7.1.1.4.2 Added

- Added meta_group_size/rank for getting the number of tiles and rank of a tile in the partition
- Added new APIs supporting Windows only, under development on Linux
  - hipMallocMipmappedArray for allocating a mipmapped array on the device
  - hipFreeMipmappedArray for freeing a mipmapped array on the device
  - hipGetMipmappedArrayLevel for getting a mipmap level of a HIP mipmapped array
  - hipMipmappedArrayCreate for creating a mipmapped array
  - hipMipmappedArrayDestroy for destroy a mipmapped array
  - hipMipmappedArrayGetLevel for getting a mipmapped array on a mipmapped level

7.1.1.4.3 Changed

7.1.1.4.4 Fixed

7.1.1.4.5 Known Issues

- HIP memory type enum values currently don't support equivalent value to cudaMemoryTypeUnregistered, due to HIP functionality backward compatibility.
- HIP API hipPointerGetAttributes could return invalid value in case the input memory pointer was not allocated through any HIP API on device or host.

7.1.1.4.6 Upcoming changes for HIP in ROCm 6.0 release

- Removal of gcnarch from hipDeviceProp_t structure
- Addition of new fields in hipDeviceProp_t structure
  - maxTexture1D
  - maxTexture2D
  - maxTexture1DLayered

  - maxTexture2DLayered
  - sharedMemPerMultiprocessor
  - deviceOverlap
  - asyncEngineCount
  - surfaceAlignment
  - unifiedAddressing
  - computePreemptionSupported
  - hostRegisterSupported
  - uuid
- Removal of deprecated code -hip-hcc codes from hip code tree
- Correct hipArray usage in HIP APIs such as hipMemcpyAtoH and hipMemcpyHtoA
- HIPMEMCPY_3D fields correction to avoid truncation of "size_t" to "unsigned int" inside hipMemcpy3D()
- Renaming of 'memoryType' in hipPointerAttribute_t structure to 'type'
- Correct hipGetLastError to return the last error instead of last API call's return code
- Update hipExternalSemaphoreHandleDesc to add "unsigned int reserved[16]"
- Correct handling of flag values in hipIpcOpenMemHandle for hipIpcMemLazyEnablePeerAccess
- Remove hiparray* and make it opaque with hipArray_t

7.1.2 Library Changes in ROCM 5.7.0

Library	Version
hipBLAS	1.1.0
hipCUB	2.13.1
hipFFT	1.0.12
hipSOLVER	1.8.1
hipSPARSE	2.3.8
MIOpen	2.19.0
rccl	2.17.1-1
rocALUTION	2.1.11
rocBLAS	3.1.0
rocFFT	1.0.24
rocm-cmake	0.10.0
rocPRIM	2.13.1
rocRAND	2.10.17
rocSOLVER	3.23.0
rocSPARSE	2.5.4
rocThrust	2.18.0
rocWMMA	1.2.0
Tensile	4.38.0

7.1.2.1 hipBLAS 1.1.0

hipBLAS 1.1.0 for ROCm 5.7.0

7.1.2.1.1 Changed

- updated documentation requirements

7.1.2.1.2 Dependencies

- dependency rocSOLVER now depends on rocSPARSE

7.1.2.2 hipCUB 2.13.1

hipCUB 2.13.1 for ROCm 5.7.0

7.1.2.2.1 Changed

- CUB backend references CUB and Thrust version 2.0.1.
- Fixed DeviceSegmentedReduce::ArgMin and DeviceSegmentedReduce::ArgMax by returning the segment-relative index instead of the absolute one.
- Fixed DeviceSegmentedReduce::ArgMin for inputs where the segment minimum is smaller than the value returned for empty segments. An equivalent fix is applied to DeviceSegmentedReduce::ArgMax.

7.1.2.2.2 Known Issues

- debug_synchronous no longer works on CUDA platform. CUB_DEBUG_SYNC should be used to enable those checks.
- DeviceReduce::Sum does not compile on CUDA platform for mixed extended-floating-point/floating-point InputT and OutputT types.
- DeviceHistogram::HistogramEven fails on CUDA platform for [LevelT, SampleIteratorT] = [int, int].
- DeviceHistogram::MultiHistogramEven fails on CUDA platform for [LevelT, SampleIteratorT] = [int, int/unsigned short/float/double] and [LevelT, SampleIteratorT] = [float, double].

7.1.2.3 hipFFT 1.0.12

hipFFT 1.0.12 for ROCm 5.7.0

7.1.2.3.1 Added

- Implemented the hipfftXtMakePlanMany, hipfftXtGetSizeMany, hipfftXtExec APIs, to allow requesting half-precision transforms.

7.1.2.3.2 Changed

- Added –precision argument to benchmark/test clients. –double is still accepted but is deprecated as a method to request a double-precision transform.

7.1.2.4 hipSOLVER 1.8.1

hipSOLVER 1.8.1 for ROCm 5.7.0

7.1.2.4.1 Changed

- Changed hipsolver-test sparse input data search paths to be relative to the test executable

7.1.2.5 hipSPARSE 2.3.8

hipSPARSE 2.3.8 for ROCm 5.7.0

7.1.2.5.1 Improved

- Fix compilation failures when using cusparse 12.1.0 backend
- Fix compilation failures when using cusparse 12.0.0 backend
- Fix compilation failures when using cusparse 10.1 (non-update versions) as backend
- Minor improvements

7.1.2.6 MIOpen 2.19.0

MIOpen 2.19.0 for ROCm 5.7.0

7.1.2.6.1 Added

- ROCm 5.5 support for gfx1101 (Navi32)

7.1.2.6.2 Changed

- Tuning results for MLIR on ROCm 5.5
- Bumping MLIR commit to 5.5.0 release tag

7.1.2.6.3 Fixed

- Fix 3d convolution Host API bug
- [HOTFIX][MI200][FP16] Disabled ConvHipImplicitGemmBwdXdlops when FP16_ALT is required.

7.1.2.7 RCCL 2.17.1-1

RCCL 2.17.1-1 for ROCm 5.7.0

7.1.2.7.1 Changed

- Compatibility with NCCL 2.17.1-1
- Performance tuning for some collective operations

7.1.2.7.2 Added

- Minor improvements to MSCCL codepath
- NCCL_NCHANNELS_PER_PEER support
- Improved compilation performance
- Support for gfx94x

7.1.2.7.3 Fixed

- Potential race-condition during ncclSocketClose()

7.1.2.8 rocALUTION 2.1.11

rocALUTION 2.1.11 for ROCm 5.7.0

7.1.2.8.1 Added

- Added support for gfx940, gfx941 and gfx942

7.1.2.8.2 Improved

- Fixed OpenMP runtime issue with Windows toolchain

7.1.2.9 rocBLAS 3.1.0

rocBLAS 3.1.0 for ROCm 5.7.0

7.1.2.9.1 Added

- yaml lock step argument scanning for rocblas-bench and rocblas-test clients. See Programmers Guide for details.
- rocblas-gemm-tune is used to find the best performing GEMM kernel for each of a given set of GEMM problems.

7.1.2.9.2 Fixed

- make offset calculations for rocBLAS functions 64 bit safe. Fixes for very large leading dimensions or increments potentially causing overflow:
  - Level 1: axpy, copy, rot, rotm, scal, swap, asum, dot, iamax, iamin, nrm2
  - Level 2: gemv, symv, hemv, trmv, ger, syr, her, syr2, her2, trsv
  - Level 3: gemm, symm, hemm, trmm, syrk, herk, syr2k, her2k, syrkx, herkx, trsm, trtri, dgmm, geam
  - General: set_vector, get_vector, set_matrix, get_matrix
  - Related fixes: internal scalar loads with > 32bit offsets
  - fix in-place functionality for all trtri sizes

7.1.2.9.3 Changed

- dot when using rocblas_pointer_mode_host is now synchronous to match legacy BLAS as it stores results in host memory
- enhanced reporting of installation issues caused by runtime libraries (Tensile)
- standardized internal rocblas C++ interface across most functions

7.1.2.9.4 Deprecated

- Removal of STDC_WANT_IEC_60559_TYPES_EXT define in future release

7.1.2.9.5 Dependencies

- optional use of AOCL BLIS 4.0 on Linux for clients
- optional build tool only dependency on python psutil

7.1.2.10 rocFFT 1.0.24

rocFFT 1.0.24 for ROCm 5.7.0

7.1.2.10.1 Optimizations

- Improved performance of complex forward/inverse 1D FFTs (2049 <= length <= 131071) that use Bluestein's algorithm.

7.1.2.10.2 Added

- Implemented a solution map version converter and finish the first conversion from ver.0 to ver.1. Where version 1 removes some incorrect kernels (sbrc/sbcr using half_lds)

7.1.2.10.3 Changed

- Moved rocfft_rtc_helper executable to lib/rocFFT directory on Linux.
- Moved library kernel cache to lib/rocFFT directory.

7.1.2.11 rocm-cmake 0.10.0

rocm-cmake 0.10.0 for ROCm 5.7.0

7.1.2.11.1 Added

- Added ROCMTest module
- ROCMCreatePackage: Added support for ASAN packages

7.1.2.12 rocPRIM 2.13.1

rocPRIM 2.13.1 for ROCm 5.7.0

7.1.2.12.1 Changed

- Deprecated configuration radix_sort_config for device-level radix sort as it no longer matches the algorithm's parameters. New configuration radix_sort_config_v2 is preferred instead.
- Removed erroneous implementation of device-level inclusive_scan and exclusive_scan. The prior default implementation using lookback-scan now is the only available implementation.
- The benchmark metric indicating the bytes processed for exclusive_scan_by_key and inclusive_scan_by_key has been changed to incorporate the key type. Furthermore, the benchmark log has been changed such that these algorithms are reported as scan and scan_by_key instead of scan_exclusive and scan_inclusive.
- Deprecated configurations scan_config and scan_by_key_config for device-level scans, as they no longer match the algorithm's parameters. New configurations scan_config_v2 and scan_by_key_config_v2 are preferred instead.

7.1.2.12.2 Fixed

- Fixed build issue caused by missing header in thread/thread_search.hpp.

7.1.2.13 rocRAND 2.10.17

rocRAND 2.10.17 for ROCm 5.7.0

7.1.2.13.1 Added

- MT19937 pseudo random number generator based on M. Matsumoto and T. Nishimura, 1998, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator.
- New benchmark for the device API using Google Benchmark, benchmark_rocrand_device_api, replacing benchmark_rocrand_kernel. benchmark_rocrand_kernel is deprecated and will be removed in a future version. Likewise, benchmark_curand_host_api is added to replace benchmark_curand_generate and benchmark_curand_device_api is added to replace benchmark_curand_kernel.
- experimental HIP-CPU feature
- ThreeFry pseudorandom number generator based on Salmon et al., 2011, "Parallel random numbers: as easy as 1, 2, 3".

7.1.2.13.2 Changed

- Python 2.7 is no longer officially supported.

7.1.2.14 rocSOLVER 3.23.0

rocSOLVER 3.23.0 for ROCm 5.7.0

7.1.2.14.1 Added

- LU factorization without pivoting for block tridiagonal matrices:
  - GEBLTTRF_NPVT now supports interleaved_batched format
- Linear system solver without pivoting for block tridiagonal matrices:
  - GEBLTTRS_NPVT now supports interleaved_batched format

7.1.2.14.2 Fixed

- Fixed stack overflow in sparse tests on Windows

7.1.2.14.3 Changed

- Changed rocsolver-test sparse input data search paths to be relative to the test executable
- Changed build scripts to default to compressed debug symbols in Debug builds

7.1.2.15 rocSPARSE 2.5.4

rocSPARSE 2.5.4 for ROCm 5.7.0

7.1.2.15.1 Added

- Added more mixed precisions for SpMV, (matrix: float, vectors: double, calculation: double) and (matrix: rocsparse_float_complex, vectors: rocsparse_double_complex, calculation: rocsparse_double_complex)
- Added support for gfx940, gfx941 and gfx942

7.1.2.15.2 Improved

- Fixed a bug in csrsm and bsrsm

7.1.2.15.3 Known Issues

In csritlu0, the algorithm rocsparse_itilu0_alg_sync_split_fusion has some accuracy issues to investigate with XNACK enabled. The fallback is rocsparse_itilu0_alg_sync_split.

7.1.2.16 rocThrust 2.18.0

rocThrust 2.18.0 for ROCm 5.7.0

7.1.2.16.1 Fixed

- lower_bound, upper_bound, and binary_search failed to compile for certain types.
- Fixed issue where transform_iterator would not compile with ___device___-only operators.

7.1.2.16.2 Changed

- Updated docs directory structure to match the standard of rocm-docs-core.
- Removed references to and workarounds for deprecated hcc

7.1.2.17 rocWMMA 1.2.0

rocWMMA 1.2.0 for ROCm 5.7.0

7.1.2.17.1 Changed

- Fixed a bug with synchronization
- Updated rocWMMA cmake versioning

7.1.2.18 Tensile 4.38.0

Tensile 4.38.0 for ROCm 5.7.0

7.1.2.18.1 Added

- Added support for FP16 Alt Round Near Zero Mode (this feature allows the generation of alternate kernels with intermediate rounding instead of truncation)
- Added user-driven solution selection feature

7.1.2.18.2 Optimizations

- Enabled LocalSplitU with MFMA for I8 data type
- Optimized K mask code in mfmaIter
- Enabled TailLoop code in NoLoadLoop to prefetch global/local read
- Enabled DirectToVgpr in TailLoop for NN, TN, and TT matrix orientations
- Optimized DirectToLds test cases to reduce the test duration

7.1.2.18.3 Changed

- Removed DGEMM NT custom kernels and related test cases
- Changed noTailLoop logic to apply noTailLoop only for NT
- Changed the range of AssertFree0ElementMultiple and Free1
- Unified aStr, bStr generation code in mfmaIter

7.1.2.18.4 Fixed

- Fixed LocalSplitU mismatch issue for SGEMM
- Fixed BufferStore=0 and Ldc != Ldd case
- Fixed mismatch issue with TailLoop + MatrixInstB > 1

CHAPTER

EIGHT

RELEASE NOTES

The release notes for the ROCm platform.

8.1 ROCm 5.7.0

8.1.1 Release Highlights for ROCm 5.7

ROCm 5.7.0 includes many new features. These include: a new library (hipTensor), and optimizations for rocRAND and MIVisionX. Address sanitizer for host and device code (GPU) is now available as a beta. Note that ROCm 5.7.0 is EOS for MI50. 5.7 versions of ROCm are the last major release in the ROCm 5 series. This release is Linux-only.

Important: The next major ROCm release (ROCm 6.0) will not be backward compatible with the ROCm 5 series. Changes will include: splitting LLVM packages into more manageable sizes, changes to the HIP runtime API, splitting rocRAND and hipRAND into separate packages, and reorganizing our file structure.

8.1.1.1 AMD Instinct™ MI50 End of Support Notice

AMD Instinct MI50, Radeon Pro VII, and Radeon VII products (collectively gfx906 GPUs) will enter maintenance mode starting Q3 2023.

As outlined in 5.6.0, ROCm 5.7 will be the final release for gfx906 GPUs to be in a fully supported state.

- ROCm 6.0 release will show MI50s as “under maintenance” mode for Linux and Windows
- No new features and performance optimizations will be supported for the gfx906 GPUs beyond this major release (ROCm 5.7).
- Bug fixes / critical security patches will continue to be supported for the gfx906 GPUs till Q2 2024 (EOM (End of Maintenance) will be aligned with the closest ROCm release).
- Bug fixes during the maintenance will be made to the next ROCm point release.
- Bug fixes will not be backported to older ROCm releases for gfx906.
- Distro / Operating system updates will continue as per the ROCm release cadence for gfx906 GPUs till EOM.

8.1.1.2 Feature Updates

8.1.1.2.1 Non-hostcall HIP Printf

Current behavior

The current version of HIP printf relies on hostcalls, which, in turn, rely on PCIe atomics. However, PCIe atomics are unavailable in some environments, and, as a result, HIP-printf does not work in those environments. Users may see the following error from runtime (with AMD_LOG_LEVEL 1 and above),

```
Pcie atomics not enabled, hostcall not supported
```

Workaround

The ROCm 5.7 release introduces an alternative to the current hostcall-based implementation that leverages an older OpenCL-based printf scheme, which does not rely on hostcalls/PCIe atomics. Note: This option is less robust than hostcall-based implementation and is intended to be a workaround when hostcalls do not work.

The printf variant is now controlled via a new compiler option -mprintf-kind=. This is supported only for HIP programs and takes the following values,

- "hostcall" – This currently available implementation relies on hostcalls, which require the system to support PCIe atomics. It is the default scheme.
- "buffered" – This implementation leverages the older printf scheme used by OpenCL; it relies on a memory buffer where printf arguments are stored during the kernel execution, and then the runtime handles the actual printing once the kernel finishes execution.

NOTE: With the new workaround,

- The printf buffer is fixed size and non-circular. After the buffer is filled, calls to printf will not result in additional output.
- The printf call returns either 0 (on success) or -1 (on failure, due to full buffer), unlike the hostcall scheme that returns the number of characters printed.

8.1.1.2.2 Beta Release of LLVM Address Sanitizer (ASAN) with the GPU

The ROCm v5.7 release introduces the beta release of LLVM Address Sanitizer (ASAN) with the GPU. The LLVM Address Sanitizer provides a process that allows developers to detect runtime addressing errors in applications and libraries. The detection is achieved using a combination of compiler-added instrumentation and runtime techniques, including function interception and replacement. Until now, the LLVM Address Sanitizer process was only available for traditional purely CPU applications. However, ROCm has extended this mechanism to additionally allow the detection of some addressing errors on the GPU in heterogeneous applications. Ideally, developers should treat heterogeneous HIP and OpenMP applications like pure CPU applications. However, this simplicity has not been achieved yet.

Refer to the documentation on LLVM Address Sanitizer with the GPU at LLVM Address Sanitizer User Guide.

Note: The beta release of LLVM Address Sanitizer for ROCm is currently tested and validated on Ubuntu 20.04.

8.1.1.3 Fixed Defects

The following defects are fixed in ROCm v5.7,

- Test hangs observed in HMM RCCL
- NoGpuTst test of Catch2 fails with Docker
- Failures observed with non-HMM HIP directed catch2 tests with XNACK+
- Multiple test failures and test hangs observed in hip-directed catch2 tests with xnack+

8.1.1.4 HIP 5.7.0

8.1.1.4.1 Optimizations

8.1.1.4.2 Added

- Added meta_group_size/rank for getting the number of tiles and rank of a tile in the partition
- Added new APIs supporting Windows only, under development on Linux
  - hipMallocMipmappedArray for allocating a mipmapped array on the device
  - hipFreeMipmappedArray for freeing a mipmapped array on the device
  - hipGetMipmappedArrayLevel for getting a mipmap level of a HIP mipmapped array
  - hipMipmappedArrayCreate for creating a mipmapped array
  - hipMipmappedArrayDestroy for destroy a mipmapped array
  - hipMipmappedArrayGetLevel for getting a mipmapped array on a mipmapped level

8.1.1.4.3 Changed

8.1.1.4.4 Fixed

8.1.1.4.5 Known Issues

- HIP memory type enum values currently don't support equivalent value to cudaMemoryTypeUnregistered, due to HIP functionality backward compatibility.
- HIP API hipPointerGetAttributes could return invalid value in case the input memory pointer was not allocated through any HIP API on device or host.

8.1.1.4.6 Upcoming changes for HIP in ROCm 6.0 release

- Removal of gcnarch from hipDeviceProp_t structure
- Addition of new fields in hipDeviceProp_t structure
  - maxTexture1D
  - maxTexture2D
  - maxTexture1DLayered

  - maxTexture2DLayered
  - sharedMemPerMultiprocessor
  - deviceOverlap
  - asyncEngineCount
  - surfaceAlignment
  - unifiedAddressing
  - computePreemptionSupported
  - hostRegisterSupported
  - uuid
- Removal of deprecated code -hip-hcc codes from hip code tree
- Correct hipArray usage in HIP APIs such as hipMemcpyAtoH and hipMemcpyHtoA
- HIPMEMCPY_3D fields correction to avoid truncation of "size_t" to "unsigned int" inside hipMemcpy3D()
- Renaming of 'memoryType' in hipPointerAttribute_t structure to 'type'
- Correct hipGetLastError to return the last error instead of last API call's return code
- Update hipExternalSemaphoreHandleDesc to add "unsigned int reserved[16]"
- Correct handling of flag values in hipIpcOpenMemHandle for hipIpcMemLazyEnablePeerAccess
- Remove hiparray* and make it opaque with hipArray_t

8.1.2 Library Changes in ROCM 5.7.0

Library	Version
hipBLAS	0.54.0 1.1.0
hipCUB	2.13.1
hipFFT	1.0.12
hipSOLVER	1.8.0 1.8.1
hipSPARSE	2.3.7 2.3.8
MIOpen	2.19.0
rccl	2.15.5 2.17.1-1
rocALUTION	2.1.9 2.1.11
rocBLAS	3.0.0 3.1.0
rocFFT	1.0.23 1.0.24
rocm-cmake	0.9.0 0.10.0
rocPRIM	2.13.0 2.13.1
rocRAND	2.10.17
rocSOLVER	3.22.0 3.23.0
rocSPARSE	2.5.2 2.5.4
rocThrust	2.18.0
rocWMMA	1.2.0
Tensile	4.37.0 4.38.0

8.1.2.1 hipBLAS 1.1.0

hipBLAS 1.1.0 for ROCm 5.7.0

8.1.2.1.1 Changed

- updated documentation requirements

8.1.2.1.2 Dependencies

- dependency rocSOLVER now depends on rocSPARSE

8.1.2.2 hipCUB 2.13.1

hipCUB 2.13.1 for ROCm 5.7.0

8.1.2.2.1 Changed

- CUB backend references CUB and Thrust version 2.0.1.
- Fixed DeviceSegmentedReduce::ArgMin and DeviceSegmentedReduce::ArgMax by returning the segment-relative index instead of the absolute one.
- Fixed DeviceSegmentedReduce::ArgMin for inputs where the segment minimum is smaller than the value returned for empty segments. An equivalent fix is applied to DeviceSegmentedReduce::ArgMax.

8.1.2.2.2 Known Issues

- debug_synchronous no longer works on CUDA platform. CUB_DEBUG_SYNC should be used to enable those checks.
- DeviceReduce::Sum does not compile on CUDA platform for mixed extended-floating-point/floating-point InputT and OutputT types.
- DeviceHistogram::HistogramEven fails on CUDA platform for [LevelT, SampleIteratorT] = [int, int].
- DeviceHistogram::MultiHistogramEven fails on CUDA platform for [LevelT, SampleIteratorT] = [int, int/unsigned short/float/double] and [LevelT, SampleIteratorT] = [float, double].

8.1.2.3 hipFFT 1.0.12

hipFFT 1.0.12 for ROCm 5.7.0

8.1.2.3.1 Added

- Implemented the hipfftXtMakePlanMany, hipfftXtGetSizeMany, hipfftXtExec APIs, to allow requesting half-precision transforms.

8.1.2.3.2 Changed

- Added –precision argument to benchmark/test clients. –double is still accepted but is deprecated as a method to request a double-precision transform.

8.1.2.4 hipSOLVER 1.8.1

hipSOLVER 1.8.1 for ROCm 5.7.0

8.1.2.4.1 Changed

- Changed hipsolver-test sparse input data search paths to be relative to the test executable

8.1.2.5 hipSPARSE 2.3.8

hipSPARSE 2.3.8 for ROCm 5.7.0

8.1.2.5.1 Improved

- Fix compilation failures when using cusparse 12.1.0 backend
- Fix compilation failures when using cusparse 12.0.0 backend
- Fix compilation failures when using cusparse 10.1 (non-update versions) as backend
- Minor improvements

8.1.2.6 MIOpen 2.19.0

MIOpen 2.19.0 for ROCm 5.7.0

8.1.2.6.1 Added

- ROCm 5.5 support for gfx1101 (Navi32)

8.1.2.6.2 Changed

- Tuning results for MLIR on ROCm 5.5
- Bumping MLIR commit to 5.5.0 release tag

8.1.2.6.3 Fixed

- Fix 3d convolution Host API bug
- [HOTFIX][MI200][FP16] Disabled ConvHipImplicitGemmBwdXdlops when FP16_ALT is required.

8.1.2.7 RCCL 2.17.1-1

RCCL 2.17.1-1 for ROCm 5.7.0

8.1.2.7.1 Changed

- Compatibility with NCCL 2.17.1-1
- Performance tuning for some collective operations

8.1.2.7.2 Added

- Minor improvements to MSCCL codepath
- NCCL_NCHANNELS_PER_PEER support
- Improved compilation performance
- Support for gfx94x

8.1.2.7.3 Fixed

- Potential race-condition during ncclSocketClose()

8.1.2.8 rocALUTION 2.1.11

rocALUTION 2.1.11 for ROCm 5.7.0

8.1.2.8.1 Added

- Added support for gfx940, gfx941 and gfx942

8.1.2.8.2 Improved

- Fixed OpenMP runtime issue with Windows toolchain

8.1.2.9 rocBLAS 3.1.0

rocBLAS 3.1.0 for ROCm 5.7.0

8.1.2.9.1 Added

- yaml lock step argument scanning for rocblas-bench and rocblas-test clients. See Programmers Guide for details.
- rocblas-gemm-tune is used to find the best performing GEMM kernel for each of a given set of GEMM problems.

8.1.2.9.2 Fixed

- make offset calculations for rocBLAS functions 64 bit safe. Fixes for very large leading dimensions or increments potentially causing overflow:
  - Level 1: axpy, copy, rot, rotm, scal, swap, asum, dot, iamax, iamin, nrm2
  - Level 2: gemv, symv, hemv, trmv, ger, syr, her, syr2, her2, trsv
  - Level 3: gemm, symm, hemm, trmm, syrk, herk, syr2k, her2k, syrkx, herkx, trsm, trtri, dgmm, geam
  - General: set_vector, get_vector, set_matrix, get_matrix
  - Related fixes: internal scalar loads with > 32bit offsets
  - fix in-place functionality for all trtri sizes

8.1.2.9.3 Changed

- dot when using rocblas_pointer_mode_host is now synchronous to match legacy BLAS as it stores results in host memory
- enhanced reporting of installation issues caused by runtime libraries (Tensile)
- standardized internal rocblas C++ interface across most functions

8.1.2.9.4 Deprecated

- Removal of STDC_WANT_IEC_60559_TYPES_EXT define in future release

8.1.2.9.5 Dependencies

- optional use of AOCL BLIS 4.0 on Linux for clients
- optional build tool only dependency on python psutil

8.1.2.10 rocFFT 1.0.24

rocFFT 1.0.24 for ROCm 5.7.0

8.1.2.10.1 Optimizations

- Improved performance of complex forward/inverse 1D FFTs (2049 <= length <= 131071) that use Bluestein's algorithm.

8.1.2.10.2 Added

- Implemented a solution map version converter and finish the first conversion from ver.0 to ver.1. Where version 1 removes some incorrect kernels (sbrc/sbcr using half_lds)

8.1.2.10.3 Changed

- Moved rocfft_rtc_helper executable to lib/rocFFT directory on Linux.
- Moved library kernel cache to lib/rocFFT directory.

8.1.2.11 rocm-cmake 0.10.0

rocm-cmake 0.10.0 for ROCm 5.7.0

8.1.2.11.1 Added

- Added ROCMTest module
- ROCMCreatePackage: Added support for ASAN packages

8.1.2.12 rocPRIM 2.13.1

rocPRIM 2.13.1 for ROCm 5.7.0

8.1.2.12.1 Changed

- Deprecated configuration radix_sort_config for device-level radix sort as it no longer matches the algorithm's parameters. New configuration radix_sort_config_v2 is preferred instead.
- Removed erroneous implementation of device-level inclusive_scan and exclusive_scan. The prior default implementation using lookback-scan now is the only available implementation.
- The benchmark metric indicating the bytes processed for exclusive_scan_by_key and inclusive_scan_by_key has been changed to incorporate the key type. Furthermore, the benchmark log has been changed such that these algorithms are reported as scan and scan_by_key instead of scan_exclusive and scan_inclusive.
- Deprecated configurations scan_config and scan_by_key_config for device-level scans, as they no longer match the algorithm's parameters. New configurations scan_config_v2 and scan_by_key_config_v2 are preferred instead.

8.1.2.12.2 Fixed

- Fixed build issue caused by missing header in thread/thread_search.hpp.

8.1.2.13 rocRAND 2.10.17

rocRAND 2.10.17 for ROCm 5.7.0

8.1.2.13.1 Added

- MT19937 pseudo random number generator based on M. Matsumoto and T. Nishimura, 1998, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator.
- New benchmark for the device API using Google Benchmark, benchmark_rocrand_device_api, replacing benchmark_rocrand_kernel. benchmark_rocrand_kernel is deprecated and will be removed in a future version. Likewise, benchmark_curand_host_api is added to replace benchmark_curand_generate and benchmark_curand_device_api is added to replace benchmark_curand_kernel.
- experimental HIP-CPU feature
- ThreeFry pseudorandom number generator based on Salmon et al., 2011, "Parallel random numbers: as easy as 1, 2, 3".

8.1.2.13.2 Changed

- Python 2.7 is no longer officially supported.

8.1.2.14 rocSOLVER 3.23.0

rocSOLVER 3.23.0 for ROCm 5.7.0

8.1.2.14.1 Added

- LU factorization without pivoting for block tridiagonal matrices:
  - GEBLTTRF_NPVT now supports interleaved_batched format
- Linear system solver without pivoting for block tridiagonal matrices:
  - GEBLTTRS_NPVT now supports interleaved_batched format

8.1.2.14.2 Fixed

- Fixed stack overflow in sparse tests on Windows

8.1.2.14.3 Changed

- Changed rocsolver-test sparse input data search paths to be relative to the test executable
- Changed build scripts to default to compressed debug symbols in Debug builds

8.1.2.15 rocSPARSE 2.5.4

rocSPARSE 2.5.4 for ROCm 5.7.0

8.1.2.15.1 Added

- Added more mixed precisions for SpMV, (matrix: float, vectors: double, calculation: double) and (matrix: rocsparse_float_complex, vectors: rocsparse_double_complex, calculation: rocsparse_double_complex)
- Added support for gfx940, gfx941 and gfx942

8.1.2.15.2 Improved

- Fixed a bug in csrsm and bsrsm

8.1.2.15.3 Known Issues

In csritlu0, the algorithm rocsparse_itilu0_alg_sync_split_fusion has some accuracy issues to investigate with XNACK enabled. The fallback is rocsparse_itilu0_alg_sync_split.

8.1.2.16 rocThrust 2.18.0

rocThrust 2.18.0 for ROCm 5.7.0

8.1.2.16.1 Fixed

- lower_bound, upper_bound, and binary_search failed to compile for certain types.
- Fixed issue where transform_iterator would not compile with ___device___-only operators.

8.1.2.16.2 Changed

- Updated docs directory structure to match the standard of rocm-docs-core.
- Removed references to and workarounds for deprecated hcc

8.1.2.17 rocWMMA 1.2.0

rocWMMA 1.2.0 for ROCm 5.7.0

8.1.2.17.1 Changed

- Fixed a bug with synchronization
- Updated rocWMMA cmake versioning

8.1.2.18 Tensile 4.38.0

Tensile 4.38.0 for ROCm 5.7.0

8.1.2.18.1 Added

- Added support for FP16 Alt Round Near Zero Mode (this feature allows the generation of alternate kernels with intermediate rounding instead of truncation)
- Added user-driven solution selection feature

8.1.2.18.2 Optimizations

- Enabled LocalSplitU with MFMA for I8 data type
- Optimized K mask code in mfmaIter
- Enabled TailLoop code in NoLoadLoop to prefetch global/local read
- Enabled DirectToVgpr in TailLoop for NN, TN, and TT matrix orientations
- Optimized DirectToLds test cases to reduce the test duration

8.1.2.18.3 Changed

- Removed DGEMM NT custom kernels and related test cases
- Changed noTailLoop logic to apply noTailLoop only for NT
- Changed the range of AssertFree0ElementMultiple and Free1
- Unified aStr, bStr generation code in mfmaIter

8.1.2.18.4 Fixed

- Fixed LocalSplitU mismatch issue for SGEMM
- Fixed BufferStore=0 and Ldc != Ldd case
- Fixed mismatch issue with TailLoop + MatrixInstB > 1

8.2 ROCm 5.6.1

8.2.1 What's New in This Release

ROCm 5.6.1 is a point release with several bug fixes in the HIP runtime.

8.3 HIP 5.6.1 (for ROCm 5.6.1)

8.3.1 Fixed Defects

- hipMemcpy device-to-device (intra device) is now asynchronous with respect to the host
- Enabled xnack+ check in HIP catch2 tests hang when executing tests
- Memory leak when code object files are loaded/unloaded via hipModuleLoad/hipModuleUnload APIs
- Using hipGraphAddMemFreeNode no longer results in a crash

8.3.2 Library Changes in ROCM 5.6.1

Library	Version
hipBLAS	0.53.0
hipCUB	2.13.1
hipFFT	1.0.12
hipSOLVER	1.8.0
hipSPARSE	2.3.6 2.3.7
MIOpen	2.19.0
rccl	2.15.5
rocALUTION	2.1.9
rocBLAS	3.0.0
rocFFT	1.0.23
rocm-cmake	0.9.0
rocPRIM	2.13.0
rocRAND	2.10.17
rocSOLVER	3.22.0
rocSPARSE	2.5.2
rocThrust	2.18.0
Tensile	4.37.0

8.3.2.1 hipSPARSE 2.3.7

hipSPARSE 2.3.7 for ROCm 5.6.1

8.3.2.1.1 Bugfix

- Reverted an undocumented API change in hipSPARSE 2.3.6 that affected hipsparseSpSV_solve function

---

8.4 ROCm 5.6.0

8.4.1 Release Highlights

ROCm 5.6 consists of several AI software ecosystem improvements to our fast-growing user base. A few examples include:

- New documentation portal at https://rocm.docs.amd.com
- Ongoing software enhancements for LLMs, ensuring full compliance with the HuggingFace unit test suite
- OpenAI Triton, CuPy, HIP Graph support, and many other library performance enhancements
- Improved ROCm deployment and development tools, including CPU-GPU (rocGDB) debugger, profiler, and docker containers
- New pseudorandom generators are available in rocRAND. Added support for half-precision transforms in hipFFT/rocFFT. Added LU refactorization and linear system solver for sparse matrices in rocSOLVER.

8.4.2 OS and GPU Support Changes

- SLES15 SP5 support was added this release. SLES15 SP3 support was dropped.
- AMD Instinct MI50, Radeon Pro VII, and Radeon VII products (collectively referred to as gfx906 GPUs) will be entering the maintenance mode starting Q3 2023. This will be aligned with ROCm 5.7 GA release date.
  - No new features and performance optimizations will be supported for the gfx906 GPUs beyond ROCm 5.7
  - Bug fixes / critical security patches will continue to be supported for the gfx906 GPUs till Q2 2024 (End of Maintenance [EOM])(will be aligned with the closest ROCm release)
  - Bug fixes during the maintenance will be made to the next ROCm point release
  - Bug fixes will not be back ported to older ROCm releases for this SKU
  - Distro / Operating system updates will continue as per the ROCm release cadence for gfx906 GPUs till EOM.

8.4.3 AMDSMI CLI 23.0.0.4

8.4.3.1 Added

- AMDSMI CLI tool enabled for Linux Bare Metal & Guest
- Package: amd-smi-lib

8.4.3.2 Known Issues

- not all Error Correction Code (ECC) fields are currently supported
- RHEL 8 & SLES 15 have extra install steps

8.4.4 Kernel Modules (DKMS)

8.4.4.1 Fixes

- Stability fix for multi GPU system reproducilble via ROCm_Bandwidth_Test as reported in Issue 2198.

8.4.5 HIP 5.6 (For ROCm 5.6)

8.4.5.1 Optimizations

- Consolidation of hipamd, rocclr and OpenCL projects in clr
- Optimized lock for graph global capture mode

8.4.5.2 Added

- Added hipRTC support for amd_hip_fp16
- Added hipStreamGetDevice implementation to get the device associated with the stream
- Added HIP_AD_FORMAT_SIGNED_INT16 in hipArray formats
- hipArrayGetInfo for getting information about the specified array
- hipArrayGetDescriptor for getting 1D or 2D array descriptor
- hipArray3DGetDescriptor to get 3D array descriptor

8.4.5.3 Changed

- hipMallocAsync to return success for zero size allocation to match hipMalloc
- Separation of hipcc perl binaries from HIP project to hipcc project. hip-devel package depends on newly added hipcc package
- Consolidation of hipamd, ROCclr, and OpenCL repositories into a single repository called clr. Instructions are updated to build HIP from sources in the HIP Installation guide
- Removed hipBusBandwidth and hipCommander samples from hip-tests

8.4.5.4 Fixed

- Fixed regression in hipMemCpyParam3D when offset is applied

8.4.5.5 Known Issues

- Limited testing on xnack+ configuration
  - Multiple HIP tests failures (gpuvm fault or hangs)
- hipSetDevice and hipSetDeviceFlags APIs return hipErrorInvalidDevice instead of hipErrorNoDevice, on a system without GPU
- Known memory leak when code object files are loaded/unloaded via hipModuleLoad/hipModuleUnload APIs. Issue will be fixed in a future ROCm release

8.4.5.6 Upcoming changes in future release

- Removal of gcnarch from hipDeviceProp_t structure
- Addition of new fields in hipDeviceProp_t structure
  - maxTexture1D
  - maxTexture2D
  - maxTexture1DLayered
  - maxTexture2DLayered
  - sharedMemPerMultiprocessor
  - deviceOverlap
  - asyncEngineCount

  - surfaceAlignment
  - unifiedAddressing
  - computePreemptionSupported
  - uuid
- Removal of deprecated code
  - hip-hcc codes from hip code tree
- Correct hipArray usage in HIP APIs such as hipMemcpyAtoH and hipMemcpyHtoA
- HIPMEMCPY_3D fields correction (unsigned int -> size_t)
- Renaming of 'memoryType' in hipPointerAttribute_t structure to 'type'

8.4.6 ROCgdb-13 (For ROCm 5.6.0)

8.4.6.1 Optimized

- Improved performances when handling the end of a process with a large number of threads.

Known Issues

- On certain configurations, ROCgdb can show the following warning message:

  warning: Probes-based dynamic linker interface failed. Reverting to original interface.

  This does not affect ROCgdb's functionalities.

8.4.7 ROCprofiler (For ROCm 5.6.0)

In ROCm 5.6 the rocprofilerv1 and rocprofilerv2 include and library files of ROCm 5.5 are split into separate files. The rocmtools files that were deprecated in ROCm 5.5 have been removed.

ROCm 5.6	rocprofilerv1	rocprofilerv2
Tool script	bin/rocprof	bin/rocprofv2
API include	include/rocprofiler/rocprofiler.h	include/rocprofiler/v2/rocprofiler.h
API library	lib/librocprofiler.so.1	lib/librocprofiler.so.2

The ROCm Profiler Tool that uses rocprofilerV1 can be invoked using the following command:

```
$ rocprof …
```

To write a custom tool based on the rocprofilerV1 API do the following:

```
main.c:
#include <rocprofiler/rocprofiler.h> // Use the rocprofilerV1 API
int main() {
  // Use the rocprofilerV1 API
  return 0;
}
```

This can be built in the following manner:

```
$ gcc main.c -I/opt/rocm-5.6.0/include -L/opt/rocm-5.6.0/lib -lrocprofiler64
```

The resulting a.out will depend on /opt/rocm-5.6.0/lib/librocprofiler64.so.1.

The ROCm Profiler that uses rocprofilerV2 API can be invoked using the following command:

```
$ rocprofv2 ...
```

To write a custom tool based on the rocprofilerV2 API do the following:

```
main.c:
#include <rocprofiler/v2/rocprofiler.h> // Use the rocprofilerV2 API
int main() {
  // Use the rocprofilerV2 API
  return 0;
}
```

This can be built in the following manner:

```
$ gcc main.c -I/opt/rocm-5.6.0/include -L/opt/rocm-5.6.0/lib -lrocprofiler64-v2
```

The resulting a.out will depend on /opt/rocm-5.6.0/lib/librocprofiler64.so.2.

8.4.7.1 Optimized

- Improved Test Suite

8.4.7.2 Added

- 'end_time' need to be disabled in roctx_trace.txt

8.4.7.3 Fixed

- rocprof in ROcm/5.4.0 gpu selector broken.
- rocprof in ROCm/5.4.1 fails to generate kernel info.
- rocprof clobbers LD_PRELOAD.

8.4.8 Library Changes in ROCM 5.6.0

Library	Version
hipBLAS	1.0.0
hipCUB	2.13.1
hipFFT	1.0.12
hipSOLVER	1.8.0
hipSPARSE	2.3.6
MIOpen	2.19.0
rccl	2.15.5
rocALUTION	2.1.9
rocBLAS	3.0.0
rocFFT	1.0.23
rocm-cmake	0.9.0
rocPRIM	2.13.0
rocRAND	2.10.17
rocSOLVER	3.22.0
rocSPARSE	2.5.2
rocThrust	2.18.0
rocWMMA	1.1.0
Tensile	4.37.0

8.4.8.1 hipBLAS 1.0.0

hipBLAS 1.0.0 for ROCm 5.6.0

8.4.8.1.1 Changed

- added const qualifier to hipBLAS functions (swap, sbmv, spmv, symv, trsm) where missing

8.4.8.1.2 Removed

- removed support for deprecated hipblasInt8Datatype_t enum
- removed support for deprecated hipblasSetInt8Datatype and hipblasGetInt8Datatype functions

8.4.8.1.3 Deprecated

- in-place trmm is deprecated. It will be replaced by trmm which includes both in-place and out-of-place functionality

8.4.8.2 hipCUB 2.13.1

hipCUB 2.13.1 for ROCm 5.6.0

8.4.8.2.1 Added

- Benchmarks for BlockShuffle, BlockLoad, and BlockStore.

8.4.8.2.2 Changed

- CUB backend references CUB and Thrust version 1.17.2.
- Improved benchmark coverage of BlockScan by adding ExclusiveScan, benchmark coverage of BlockRadixSort by adding SortBlockedToStriped, and benchmark coverage of WarpScan by adding Broadcast.
- Updated docs directory structure to match the standard of rocm-docs-core.

8.4.8.2.3 Known Issues

- BlockRadixRankMatch is currently broken under the rocPRIM backend.
- BlockRadixRankMatch with a warp size that does not exactly divide the block size is broken under the CUB backend.

8.4.8.3 hipFFT 1.0.12

hipFFT 1.0.12 for ROCm 5.6.0

8.4.8.3.1 Added

- Implemented the hipfftXtMakePlanMany, hipfftXtGetSizeMany, hipfftXtExec APIs, to allow requesting half-precision transforms.

8.4.8.3.2 Changed

- Added –precision argument to benchmark/test clients. –double is still accepted but is deprecated as a method to request a double-precision transform.

8.4.8.4 hipSOLVER 1.8.0

hipSOLVER 1.8.0 for ROCm 5.6.0

8.4.8.4.1 Added

- Added compatibility API with hipsolverRf prefix

8.4.8.5 hipSPARSE 2.3.6

hipSPARSE 2.3.6 for ROCm 5.6.0

8.4.8.5.1 Added

- Added SpGEMM algorithms

8.4.8.5.2 Changed

- For hipsparseXbsr2csr and hipsparseXcsr2bsr, blockDim == 0 now returns HIPSPARSE_STATUS_INVALID_SIZE

8.4.8.6 MIOpen 2.19.0

MIOpen 2.19.0 for ROCm 5.6.0

8.4.8.6.1 Added

- ROCm 5.5 support for gfx1101 (Navi32)

8.4.8.6.2 Changed

- Tuning results for MLIR on ROCm 5.5
- Bumping MLIR commit to 5.5.0 release tag

8.4.8.6.3 Fixed

- Fix 3d convolution Host API bug
- [HOTFIX][MI200][FP16] Disabled ConvHipImplicitGemmBwdXdlops when FP16_ALT is required.

8.4.8.7 rccl 2.15.5

RCCL 2.15.5 for ROCm 5.6.0

8.4.8.7.1 Changed

- Compatibility with NCCL 2.15.5
- Unit test executable renamed to rccl-UnitTests

8.4.8.7.2 Added

- HW-topology aware binary tree implementation
- Experimental support for MSCCL
- New unit tests for hipGraph support
- NPKit integration

8.4.8.7.3 Fixed

- rocm-smi ID conversion
- Support for HIP_VISIBLE_DEVICES for unit tests
- Support for p2p transfers to non (HIP) visible devices

8.4.8.7.4 Removed

- Removed TransferBench from tools. Exists in standalone repo: https://github.com/ROCmSoftwarePlatform/TransferBench

8.4.8.8 rocALUTION 2.1.9

rocALUTION 2.1.9 for ROCm 5.6.0

8.4.8.8.1 Improved

- Fixed synchronization issues in level 1 routines

8.4.8.9 rocBLAS 3.0.0

rocBLAS 3.0.0 for ROCm 5.6.0

8.4.8.9.1 Optimizations

- Improved performance of Level 2 rocBLAS GEMV on gfx90a GPU for non-transposed problems having small matrices and larger batch counts. Performance enhanced for problem sizes when m and n <= 32 and batch_count >= 256.
- Improved performance of rocBLAS syr2k for single, double, and double-complex precision, and her2k for double-complex precision. Slightly improved performance for general sizes on gfx90a.

8.4.8.9.2 Added

- Added bf16 inputs and f32 compute support to Level 1 rocBLAS Extension functions axpy_ex, scal_ex and nrm2_ex.

8.4.8.9.3 Deprecated

- trmm inplace is deprecated. It will be replaced by trmm that has both inplace and out-of-place functionality
- rocblas_query_int8_layout_flag() is deprecated and will be removed in a future release
- rocblas_gemm_flags_pack_int8x4 enum is deprecated and will be removed in a future release
- rocblas_set_device_memory_size() is deprecated and will be replaced by a future function rocblas_increase_device_memory_size()
- rocblas_is_user_managing_device_memory() is deprecated and will be removed in a future release

8.4.8.9.4 Removed

- is_complex helper was deprecated and now removed. Use rocblas_is_complex instead.
- The enum truncate_t and the value truncate was deprecated and now removed from. It was replaced by rocblas_truncate_t and rocblas_truncate, respectively.
- rocblas_set_int8_type_for_hipblas was deprecated and is now removed.
- rocblas_get_int8_type_for_hipblas was deprecated and is now removed.

8.4.8.9.5 Dependencies

- build only dependency on python joblib added as used by Tensile build
- fix for cmake install on some OS when performed by install.sh -d –cmake_install

8.4.8.9.6 Fixed

- make trsm offset calculations 64 bit safe

8.4.8.9.7 Changed

- refactor rotg test code

8.4.8.10 rocFFT 1.0.23

rocFFT 1.0.23 for ROCm 5.6.0

8.4.8.10.1 Added

- Implemented half-precision transforms, which can be requested by passing rocfft_precision_half to rocfft_plan_create.
- Implemented a hierarchical solution map which saves how to decompose a problem and the kernels to be used.
- Implemented a first version of offline-tuner to support tuning kernels for C2C/Z2Z problems.

8.4.8.10.2 Changed

- Replaced std::complex with hipComplex data types for data generator.
- FFT plan dimensions are now sorted to be row-major internally where possible, which produces better plans if the dimensions were accidentally specified in a different order (column-major, for example).
- Added –precision argument to benchmark/test clients. –double is still accepted but is deprecated as a method to request a double-precision transform.

8.4.8.10.3 Fixed

- Fixed over-allocation of LDS in some real-complex kernels, which was resulting in kernel launch failure.

8.4.8.11 rocm-cmake 0.9.0

rocm-cmake 0.9.0 for ROCm 5.6.0

8.4.8.11.1 Added

- Added the option ROCM_HEADER_WRAPPER_WERROR
  - Compile-time C macro in the wrapper headers causes errors to be emitted instead of warnings.
  - Configure-time CMake option sets the default for the C macro.

8.4.8.12 rocPRIM 2.13.0

rocPRIM 2.13.0 for ROCm 5.6.0

8.4.8.12.1 Added

- New block level radix_rank primitive.
- New block level radix_rank_match primitive.
- Added a stable block sorting implementation. This be used with block_sort by using the block_sort_algorithm::stable_merge_sort algorithm.

8.4.8.12.2 Changed

- Improved the performance of block_radix_sort and device_radix_sort.
- Improved the performance of device_merge_sort.
- Updated docs directory structure to match the standard of rocm-docs-core. Contributed by: v01dXYZ.

8.4.8.12.3 Known Issues

- Disabled GPU error messages relating to incorrect warp operation usage with Navi GPUs on Windows, due to GPU printf performance issues on Windows.
- When ROCPRIM_DISABLE_LOOKBACK_SCAN is set, device_scan fails for input sizes bigger than scan_config::size_limit, which defaults to std::numeric_limits<unsigned int>::max().

8.4.8.13 rocRAND 2.10.17

rocRAND 2.10.17 for ROCm 5.6.0

8.4.8.13.1 Added

- MT19937 pseudo random number generator based on M. Matsumoto and T. Nishimura, 1998, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator.
- New benchmark for the device API using Google Benchmark, benchmark_rocrand_device_api, replacing benchmark_rocrand_kernel. benchmark_rocrand_kernel is deprecated and will be removed in a future version. Likewise, benchmark_curand_host_api is added to replace benchmark_curand_generate and benchmark_curand_device_api is added to replace benchmark_curand_kernel.
- experimental HIP-CPU feature
- ThreeFry pseudorandom number generator based on Salmon et al., 2011, “Parallel random numbers: as easy as 1, 2, 3”.

8.4.8.13.2 Changed

- Python 2.7 is no longer officially supported.

8.4.8.14 rocSOLVER 3.22.0

rocSOLVER 3.22.0 for ROCm 5.6.0

8.4.8.14.1 Added

- LU refactorization for sparse matrices
  - CSRRF_ANALYSIS
  - CSRRF_SUMLU
  - CSRRF_SPLITLU
  - CSRRF_REFACTLU
- Linear system solver for sparse matrices
  - CSRRF_SOLVE
- Added type rocsolver_rfinfo for use with sparse matrix routines

8.4.8.14.2 Optimized

- Improved the performance of BDSQR and GESVD when singular vectors are requested

8.4.8.14.3 Fixed

- BDSQR and GESVD should no longer hang when the input contains NaN or Inf

8.4.8.15 rocSPARSE 2.5.2

rocSPARSE 2.5.2 for ROCm 5.6.0

8.4.8.15.1 Improved

- Fixed a memory leak in csritsv
- Fixed a bug in csrsm and bsrsm

8.4.8.16 rocThrust 2.18.0

rocThrust 2.18.0 for ROCm 5.6.0

8.4.8.16.1 Fixed

- lower_bound, upper_bound, and binary_search failed to compile for certain types.

8.4.8.16.2 Changed

- Updated docs directory structure to match the standard of rocm-docs-core.

8.4.8.17 rocWMMA 1.1.0

rocWMMA 1.1.0 for ROCm 5.6.0

8.4.8.17.1 Added

- Added cross-lane operation backends (Blend, Permute, Swizzle and Dpp)
- Added GPU kernels for rocWMMA unit test pre-process and post-process operations (fill, validation)
- Added performance gemm samples for half, single and double precision
- Added rocWMMA cmake versioning
- Added vectorized support in coordinate transforms
- Included ROCm smi for runtime clock rate detection
- Added fragment transforms for transpose and change data layout

8.4.8.17.2 Changed

- Default to GPU rocBLAS validation against rocWMMA
- Re-enabled int8 gemm tests on gfx9
- Upgraded to C++17
- Restructured unit test folder for consistency
- Consolidated rocWMMA samples common code

8.4.8.18 Tensile 4.37.0

Tensile 4.37.0 for ROCm 5.6.0

8.4.8.18.1 Added

- Added user driven tuning API
- Added decision tree fallback feature
- Added SingleBuffer + AtomicAdd option for GlobalSplitU
- DirectToVgpr support for fp16 and Int8 with TN orientation
- Added new test cases for various functions
- Added SingleBuffer algorithm for ZGEMM/CGEMM
- Added joblib for parallel map calls
- Added support for MFMA + LocalSplitU + DirectToVgprA+B
- Added asmcap check for MIArchVgpr
- Added support for MFMA + LocalSplitU
- Added frequency, power, and temperature data to the output

8.4.8.18.2 Optimizations

- Improved the performance of GlobalSplitU with SingleBuffer algorithm
- Reduced the running time of the extended and pre_checkin tests
- Optimized the Tailloop section of the assembly kernel
- Optimized complex GEMM (fixed vgpr allocation, unified CGEMM and ZGEMM code in MulMIoutAlphaToArch)
- Improved the performance of the second kernel of MultipleBuffer algorithm

8.4.8.18.3 Changed

- Updated custom kernels with 64-bit offsets
- Adapted 64-bit offset arguments for assembly kernels
- Improved temporary register re-use to reduce max sgpr usage
- Removed some restrictions on VectorWidth and DirectToVgpr
- Updated the dependency requirements for Tensile
- Changed the range of AssertSummationElementMultiple
- Modified the error messages for more clarity
- Changed DivideAndReminder to vectorStaticRemainder in case quotient is not used
- Removed dummy vgpr for vectorStaticRemainder
- Removed tmpVgpr parameter from vectorStaticRemainder/Divide/DivideAndReminder

- Removed qReg parameter from vectorStaticRemainder

8.4.8.18.4 Fixed

- Fixed tmp sgpr allocation to avoid over-writing values (alpha)
- 64-bit offset parameters for post kernels
- Fixed gfx908 CI test failures
- Fixed offset calculation to prevent overflow for large offsets
- Fixed issues when BufferLoad and BufferStore are equal to zero
- Fixed StoreCInUnroll + DirectToVgpr + no useInitAccVgprOpt mismatch
- Fixed DirectToVgpr + LocalSplitU + FractionalLoad mismatch
- Fixed the memory access error related to StaggerU + large stride
- Fixed ZGEMM 4x4 MatrixInst mismatch
- Fixed DGEMM 4x4 MatrixInst mismatch
- Fixed ASEM + GSU + NoTailLoop opt mismatch
- Fixed AssertSummationElementMultiple + GlobalSplitU issues
- Fixed ASEM + GSU + TailLoop inner unroll

8.5 ROCm 5.5.1

8.5.1 What's New in This Release

8.5.1.1 HIP SDK for Windows

AMD is pleased to announce the availability of the HIP SDK for Windows as part of the ROCm platform. The HIP SDK OS and GPU support page lists the versions of Windows and GPUs validated by AMD. HIP SDK features on Windows are described in detail in our What is ROCm? page and differs from the Linux feature set. Visit Quick Start page to get started. Known issues are tracked on GitHub.

8.5.1.2 HIP API Change

The following HIP API is updated in the ROCm 5.5.1 release:

8.5.1.2.1 hipDeviceSetCacheConfig

- The return value for hipDeviceSetCacheConfig is updated from hipErrorNotSupported to hipSuccess

8.5.2 Library Changes in ROCM 5.5.1

Library	Version
hipBLAS	0.54.0
hipCUB	2.13.1
hipFFT	1.0.11
hipSOLVER	1.7.0
hipSPARSE	2.3.5
MIOpen	2.19.0
rccl	2.15.5
rocALUTION	2.1.8
rocBLAS	2.47.0
rocFFT	1.0.22
rocm-cmake	0.8.1
rocPRIM	2.13.0
rocRAND	2.10.17
rocSOLVER	3.21.0
rocSPARSE	2.5.1
rocThrust	2.17.0
rocWMMA	1.0
Tensile	4.36.0

8.6 ROCm 5.5.0

8.6.1 What's New in This Release

8.6.1.1 HIP Enhancements

The ROCm v5.5 release consists of the following HIP enhancements:

8.6.1.1.1 Enhanced Stack Size Limit

In this release, the stack size limit is increased from 16k to 131056 bytes (or 128K - 16). Applications requiring to update the stack size can use hipDeviceSetLimit API.

8.6.1.1.2 hipcc Changes

The following hipcc changes are implemented in this release:

- hipcc will not implicitly link to libpthread and librt, as they are no longer a link time dependence for HIP programs. Applications that depend on these libraries must explicitly link to them.
- -use-staticlib and -use-sharedlib options are deprecated.

8.6.1.1.3 Future Changes

- Separation of hipcc binaries (Perl scripts) from HIP to hipcc project. Users will access separate hipcc package for installing hipcc binaries in future ROCm releases.
- In a future ROCm release, the following samples will be removed from the hip-tests project.
  - hipBusbandWidth at https://github.com/ROCm-Developer-Tools/hip-tests/tree/develop/samples/1_Utils/shipBusBandwidth
  - hipCommander at https://github.com/ROCm-Developer-Tools/hip-tests/tree/develop/samples/1_Utils/hipCommander

  Note that the samples will continue to be available in previous release branches.
- Removal of gcnarch from hipDeviceProp_t structure
- Addition of new fields in hipDeviceProp_t structure
  - maxTexture1D
  - maxTexture2D
  - maxTexture1DLayered
  - maxTexture2DLayered
  - sharedMemPerMultiprocessor
  - deviceOverlap
  - asyncEngineCount
  - surfaceAlignment
  - unifiedAddressing
  - computePreemptionSupported
  - hostRegisterSupported
  - uuid
- Removal of deprecated code
  - hip-hcc codes from hip code tree
- Correct hipArray usage in HIP APIs such as hipMemcpyAtoH and hipMemcpyHtoA
- HIPMEMCPY_3D fields correction to avoid truncation of “size_t” to “unsigned int” inside hipMemcpy3D()
- Renaming of ‘memoryType’ in hipPointerAttribute_t structure to ‘type’
- Correct hipGetLastError to return the last error instead of last API call’s return code
- Update hipExternalSemaphoreHandleDesc to add “unsigned int reserved[16]”

- Correct handling of flag values in hipIpcOpenMemHandle for hipIpcMemLazyEnablePeerAccess
- Remove hiparray* and make it opaque with hipArray_t

8.6.1.1.4 New HIP APIs in This Release

Note

This is a pre-official version (beta) release of the new APIs and may contain unresolved issues.

8.6.1.1.4.1 Memory Management HIP APIs

The new memory management HIP API is as follows:

- Sets information on the specified pointer [BETA].

```
hipError_t hipPointerSetAttribute(const void* value, hipPointer_attribute attribute, hipDeviceptr_t ptr);
```

8.6.1.1.4.2 Module Management HIP APIs

The new module management HIP APIs are as follows:

- Launches kernel f with launch parameters and shared memory on stream with arguments passed to kernelParams, where thread blocks can cooperate and synchronize as they execute.

```
hipError_t hipModuleLaunchCooperativeKernel(hipFunction_t f, unsigned int gridDimX, unsigned int␣
↪gridDimY, unsigned int gridDimZ, unsigned int blockDimX, unsigned int blockDimY, unsigned int␣
↪blockDimZ, unsigned int sharedMemBytes, hipStream_t stream, void** kernelParams);
```

- Launches kernels on multiple devices where thread blocks can cooperate and synchronize as they execute.

```
hipError_t hipModuleLaunchCooperativeKernelMultiDevice(hipFunctionLaunchParams* launchParamsList,
↪ unsigned int numDevices, unsigned int flags);
```

8.6.1.1.4.3 HIP Graph Management APIs

The new HIP Graph Management APIs are as follows:

- Creates a memory allocation node and adds it to a graph [BETA]

```
hipError_t hipGraphAddMemAllocNode(hipGraphNode_t* pGraphNode, hipGraph_t graph, const␣
↪hipGraphNode_t* pDependencies, size_t numDependencies, hipMemAllocNodeParams* pNodeParams);
```

- Return parameters for memory allocation node [BETA]

```
hipError_t hipGraphMemAllocNodeGetParams(hipGraphNode_t node, hipMemAllocNodeParams*␣
↪pNodeParams);
```

- Creates a memory free node and adds it to a graph [BETA]

```
hipError_t hipGraphAddMemFreeNode(hipGraphNode_t* pGraphNode, hipGraph_t graph, const␣
↪hipGraphNode_t* pDependencies, size_t numDependencies, void* dev_ptr);
```

- Returns parameters for memory free node [BETA].

```
hipError_t hipGraphMemFreeNodeGetParams(hipGraphNode_t node, void* dev_ptr);
```

- Write a DOT file describing graph structure [BETA].

```
hipError_t hipGraphDebugDotPrint(hipGraph_t graph, const char* path, unsigned int flags);
```

- Copies attributes from source node to destination node [BETA].

```
hipError_t hipGraphKernelNodeCopyAttributes(hipGraphNode_t hSrc, hipGraphNode_t hDst);
```

- Enables or disables the specified node in the given graphExec [BETA]

```
hipError_t hipGraphNodeSetEnabled(hipGraphExec_t hGraphExec, hipGraphNode_t hNode, unsigned int␣
↪isEnabled);
```

- Query whether a node in the given graphExec is enabled [BETA]

```
hipError_t hipGraphNodeGetEnabled(hipGraphExec_t hGraphExec, hipGraphNode_t hNode, unsigned␣
↪int* isEnabled);
```

8.6.1.1.5 OpenMP Enhancements

This release consists of the following OpenMP enhancements:

- Additional support for OMPT functions get_device_time and get_record_type.
- Add support for min/max fast fp atomics on AMD GPUs.
- Fix the use of the abs function in C device regions.

8.6.2 Deprecations and Warnings

8.6.2.1 HIP Deprecation

The hipcc and hipconfig Perl scripts are deprecated. In a future release, compiled binaries will be available as hipcc.bin and hipconfig.bin as replacements for the Perl scripts.

> Note
>
> There will be a transition period where the Perl scripts and compiled binaries are available before the scripts are removed. There will be no functional difference between the Perl scripts and their compiled binary counterpart. No user action is required. Once these are available, users can optionally switch to hipcc.bin and hipconfig.bin. The hipcc/hipconfig soft link will be assimilated to point from hipcc/hipconfig to the respective compiled binaries as the default option.

8.6.2.1.1 Linux Filesystem Hierarchy Standard for ROCm

ROCm packages have adopted the Linux foundation filesystem hierarchy standard in this release to ensure ROCm components follow open source conventions for Linux-based distributions. While moving to a new filesystem hierarchy, ROCm ensures backward compatibility with its 5.1 version or older filesystem hierarchy. See below for a detailed explanation of the new filesystem hierarchy and backward compatibility.

8.6.2.1.2 New Filesystem Hierarchy

The following is the new filesystem hierarchy:4

```
/opt/rocm-<ver>
   | --bin
     | --All externally exposed Binaries
   | --libexec
      | --<component>
         | -- Component specific private non-ISA executables (architecture independent)
   | --include
      | -- <component>
         | --<header files>
   | --lib
      | --lib<soname>.so -> lib<soname>.so.major -> lib<soname>.so.major.minor.patch
         (public libraries linked with application)
      | --<component> (component specific private library, executable data)
      | --<cmake>
         | --components
            | --<component>.config.cmake
   | --share
      | --html/<component>/*.html
      | --info/<component>/*.[pdf, md, txt]
      | --man
      | --doc
         | --<component>
            | --<licenses>
      | --<component>
         | --<misc files> (arch independent non-executable)
         | --samples
```

> Note
>
> ROCm will not support backward compatibility with the v5.1(old) file system hierarchy in its next major release.

For more information, refer to https://refspecs.linuxfoundation.org/fhs.shtml.

8.6.2.1.3 Backward Compatibility with Older Filesystems

ROCm has moved header files and libraries to its new location as indicated in the above structure and included symbolic-link and wrapper header files in its old location for backward compatibility.

> Note
>
> ROCm will continue supporting backward compatibility until the next major release.

8.6.2.1.4 Wrapper header files

Wrapper header files are placed in the old location (/opt/rocm-xxx/<component>/include) with a warning message to include files from the new location (/opt/rocm-xxx/include) as shown in the example below:

```
// Code snippet from hip_runtime.h
#pragma message “This file is deprecated. Use file from include path /opt/rocm-ver/include/ and prefix with hip”.
#include ”hip/hip_runtime.h”
```

The wrapper header files' backward compatibility deprecation is as follows:

- #pragma message announcing deprecation – ROCm v5.2 release
- #pragma message changed to #warning – Future release
- #warning changed to #error – Future release
- Backward compatibility wrappers removed – Future release

8.6.2.1.5 Library files

Library files are available in the /opt/rocm-xxx/lib folder. For backward compatibility, the old library location (/opt/rocm-xxx/<component>/lib) has a soft link to the library at the new location.

Example:

```
$ ls -l /opt/rocm/hip/lib/
total 4
drwxr-xr-x 4 root root 4096 May 12 10:45 cmake
lrwxrwxrwx 1 root root   24 May 10 23:32 libamdhip64.so -> ../../lib/libamdhip64.so
```

8.6.2.1.6 CMake Config files

All CMake configuration files are available in the /opt/rocm-xxx/lib/cmake/<component> folder. For backward compatibility, the old CMake locations (/opt/rocm-xxx/<component>/lib/cmake) consist of a soft link to the new CMake config.

Example:

```
$ ls -l /opt/rocm/hip/lib/cmake/hip/
total 0
lrwxrwxrwx 1 root root 42 May 10 23:32 hip-config.cmake -> ../../../../lib/cmake/hip/hip-config.cmake
```

8.6.2.2 ROCm Support For Code Object V3 Deprecated

Support for Code Object v3 is deprecated and will be removed in a future release.

8.6.2.3 Comgr V3.0 Changes

The following APIs and macros have been marked as deprecated. These are expected to be removed in a future ROCm release and coincides with the release of Comgr v3.0.

8.6.2.3.1 API Changes

- amd_comgr_action_info_set_options()
- amd_comgr_action_info_get_options()

8.6.2.3.2 Actions and Data Types

- AMD_COMGR_ACTION_ADD_DEVICE_LIBRARIES
- AMD_COMGR_ACTION_COMPILE_SOURCE_TO_FATBIN

For replacements, see the AMD_COMGR_ACTION_INFO_GET/SET_OPTION_LIST APIs, and the AMD_COMGR_ACTION_COMPILE_SOURCE_(WITH_DEVICE_LIBS)_TO_BC macros.

8.6.2.4 Deprecated Environment Variables

The following environment variables are removed in this ROCm release:

- GPU_MAX_COMMAND_QUEUES
- GPU_MAX_WORKGROUP_SIZE_2D_X
- GPU_MAX_WORKGROUP_SIZE_2D_Y
- GPU_MAX_WORKGROUP_SIZE_3D_X
- GPU_MAX_WORKGROUP_SIZE_3D_Y
- GPU_MAX_WORKGROUP_SIZE_3D_Z
- GPU_BLIT_ENGINE_TYPE
- GPU_USE_SYNC_OBJECTS
- AMD_OCL_SC_LIB
- AMD_OCL_ENABLE_MESSAGE_BOX
- GPU_FORCE_64BIT_PTR
- GPU_FORCE_OCL20_32BIT
- GPU_RAW_TIMESTAMP
- GPU_SELECT_COMPUTE_RINGS_ID
- GPU_USE_SINGLE_SCRATCH
- GPU_ENABLE_LARGE_ALLOCATION
- HSA_LOCAL_MEMORY_ENABLE
- HSA_ENABLE_COARSE_GRAIN_SVM
- GPU_IFH_MODE
- OCL_SYSMEM_REQUIREMENT

- OCL_CODE_CACHE_ENABLE
- OCL_CODE_CACHE_RESET

8.6.3 Known Issues In This Release

The following are the known issues in this release.

8.6.3.1 DISTRIBUTED/TEST_DISTRIBUTED_SPAWN Fails

When user applications call ncclCommAbort to destruct communicators and then create new communicators repeatedly, subsequent communicators may fail to initialize.

This issue is under investigation and will be resolved in a future release.

8.6.3.2 Failures In HIP Directed Tests

Multiple HIP directed tests fail.

8.6.4 Library Changes in ROCM 5.5.0

Library	Version
hipBLAS	0.53.0 0.54.0
hipCUB	2.13.0 2.13.1
hipFFT	1.0.10 1.0.11
hipSOLVER	1.6.0 1.7.0
hipSPARSE	2.3.3 2.3.5
MIOpen	2.19.0
rccl	2.13.4 2.15.5
rocALUTION	2.1.3 2.1.8
rocBLAS	2.46.0 2.47.0
rocFFT	1.0.21 1.0.22
rocm-cmake	0.8.0 0.8.1
rocPRIM	2.12.0 2.13.0
rocRAND	2.10.16 2.10.17
rocSOLVER	3.20.0 3.21.0
rocSPARSE	2.4.0 2.5.1
rocThrust	2.17.0
rocWMMA	0.9 1.0
Tensile	4.35.0 4.36.0

8.6.4.1 hipBLAS 0.54.0

hipBLAS 0.54.0 for ROCm 5.5.0

8.6.4.1.1 Added

- added option to opt-in to use ___half for hipblasHalf type in the API for c++ users who define HIPBLAS_USE_HIP_HALF
- added scripts to plot performance for multiple functions
- data driven hipblas-bench and hipblas-test execution via external yaml format data files
- client smoke test added for quick validation using command hipblas-test –yaml hipblas_smoke.yaml

8.6.4.1.2 Fixed

- fixed datatype conversion functions to support more rocBLAS/cuBLAS datatypes
- fixed geqrf to return successfully when nullptrs are passed in with n == 0 || m == 0
- fixed getrs to return successfully when given nullptrs with corresponding size = 0
- fixed getrs to give info = -1 when transpose is not an expected type
- fixed gels to return successfully when given nullptrs with corresponding size = 0
- fixed gels to give info = -1 when transpose is not in ('N', 'T') for real cases or not in ('N', 'C') for complex cases

8.6.4.1.3 Changed

- changed reference code for Windows to OpenBLAS
- hipblas client executables all now begin with hipblas- prefix

8.6.4.2 hipCUB 2.13.1

hipCUB 2.13.1 for ROCm 5.5.0

8.6.4.2.1 Added

- Benchmarks for BlockShuffle, BlockLoad, and BlockStore.

8.6.4.2.2 Changed

- CUB backend references CUB and Thrust version 1.17.2.
- Improved benchmark coverage of BlockScan by adding ExclusiveScan, benchmark coverage of BlockRadixSort by adding SortBlockedToStriped, and benchmark coverage of WarpScan by adding Broadcast.

8.6.4.2.3 Fixed

- Windows HIP SDK support

8.6.4.2.4 Known Issues

- BlockRadixRankMatch is currently broken under the rocPRIM backend.
- BlockRadixRankMatch with a warp size that does not exactly divide the block size is broken under the CUB backend.

8.6.4.3 hipFFT 1.0.11

hipFFT 1.0.11 for ROCm 5.5.0

8.6.4.3.1 Fixed

- Fixed old version rocm include/lib folders not removed on upgrade.

8.6.4.4 hipSOLVER 1.7.0

hipSOLVER 1.7.0 for ROCm 5.5.0

8.6.4.4.1 Added

- Added functions
  - gesvdj
    - hipsolverSgesvdj_bufferSize, hipsolverDgesvdj_bufferSize, hipsolverCgesvdj_bufferSize, hipsolverZgesvdj_bufferSize
    - hipsolverSgesvdj, hipsolverDgesvdj, hipsolverCgesvdj, hipsolverZgesvdj
  - gesvdjBatched
    - hipsolverSgesvdjBatched_bufferSize, hipsolverDgesvdjBatched_bufferSize, hipsolverCgesvdjBatched_bufferSize, hipsolverZgesvdjBatched_bufferSize
    - hipsolverSgesvdjBatched, hipsolverDgesvdjBatched, hipsolverCgesvdjBatched, hipsolverZgesvdjBatched

8.6.4.5 hipSPARSE 2.3.5

hipSPARSE 2.3.5 for ROCm 5.5.0

8.6.4.5.1 Improved

- Fixed an issue, where the rocm folder was not removed on upgrade of meta packages
- Fixed a compilation issue with cusparse backend
- Added more detailed messages on unit test failures due to missing input data
- Improved documentation
- Fixed a bug with deprecation messages when using gcc9 (Thanks @Maetveis)

8.6.4.6 MIOpen 2.19.0

MIOpen 2.19.0 for ROCm 5.5.0

8.6.4.6.1 Added

- ROCm 5.5 support for gfx1101 (Navi32)

8.6.4.6.2 Changed

- Tuning results for MLIR on ROCm 5.5
- Bumping MLIR commit to 5.5.0 release tag

8.6.4.6.3 Fixed

- Fix 3d convolution Host API bug
- [HOTFIX][MI200][FP16] Disabled ConvHipImplicitGemmBwdXdlops when FP16_ALT is required.

8.6.4.7 rccl 2.15.5

RCCL 2.15.5 for ROCm 5.5.0

8.6.4.7.1 Changed

- Compatibility with NCCL 2.15.5
- Unit test executable renamed to rccl-UnitTests

8.6.4.7.2 Added

- HW-topology aware binary tree implementation
- Experimental support for MSCCL
- New unit tests for hipGraph support
- NPKit integration

8.6.4.7.3 Fixed

- rocm-smi ID conversion
- Support for HIP_VISIBLE_DEVICES for unit tests
- Support for p2p transfers to non (HIP) visible devices

8.6.4.7.4 Removed

- Removed TransferBench from tools. Exists in standalone repo: https://github.com/ROCmSoftwarePlatform/TransferBench

8.6.4.8 rocALUTION 2.1.8

rocALUTION 2.1.8 for ROCm 5.5.0

8.6.4.8.1 Added

- Added build support for Navi32

8.6.4.8.2 Improved

- Fixed a typo in MPI backend
- Fixed a bug with the backend when HIP support is disabled
- Fixed a bug in SAAMG hierarchy building on HIP backend
- Improved SAAMG hierarchy build performance on HIP backend

8.6.4.8.3 Changed

- LocalVector::GetIndexValues(ValueType*) is deprecated, use LocalVector::GetIndexValues(const LocalVector&, LocalVector*) instead
- LocalVector::SetIndexValues(const ValueType*) is deprecated, use LocalVector::SetIndexValues(const LocalVector&, const LocalVector&) instead
- LocalMatrix::RSDirectInterpolation(const LocalVector&, const LocalVector&, LocalMatrix*, LocalMatrix*) is deprecated, use LocalMatrix::RSDirectInterpolation(const LocalVector&, const LocalVector&, LocalMatrix*) instead

- LocalMatrix::RSExtPIInterpolation(const LocalVector&, const LocalVector&, bool, float, LocalMatrix*, LocalMatrix*) is deprecated, use LocalMatrix::RSExtPIInterpolation(const LocalVector&, const LocalVector&, bool, LocalMatrix*) instead
- LocalMatrix::RugeStueben() is deprecated
- LocalMatrix::AMGSmoothedAggregation(ValueType, const LocalVector&, const LocalVector&, LocalMatrix*, LocalMatrix*, int) is deprecated, use LocalMatrix::AMGAggregation(ValueType, const LocalVector&, const LocalVector&, LocalMatrix*, int) instead
- LocalMatrix::AMGAggregation(const LocalVector&, LocalMatrix*, LocalMatrix*) is deprecated, use LocalMatrix::AMGAggregation(const LocalVector&, LocalMatrix*) instead

8.6.4.9 rocBLAS 2.47.0

rocBLAS 2.47.0 for ROCm 5.5.0

8.6.4.9.1 Added

- added functionality rocblas_geam_ex for matrix-matrix minimum operations
- added HIP Graph support as beta feature for rocBLAS Level 1, Level 2, and Level 3(pointer mode host) functions
- added beta features API. Exposed using compiler define ROCBLAS_BETA_FEATURES_API
- added support for vector initialization in the rocBLAS test framework with negative increments
- added windows build documentation for forthcoming support using ROCm HIP SDK
- added scripts to plot performance for multiple functions

8.6.4.9.2 Optimizations

- improved performance of Level 2 rocBLAS GEMV for float and double precision. Performance enhanced by 150-200% for certain problem sizes when (m==n) measured on a gfx90a GPU.
- improved performance of Level 2 rocBLAS GER for float, double and complex float precisions. Performance enhanced by 5-7% for certain problem sizes measured on a gfx90a GPU.
- improved performance of Level 2 rocBLAS SYMV for float and double precisions. Performance enhanced by 120-150% for certain problem sizes measured on both gfx908 and gfx90a GPUs.

8.6.4.9.3 Fixed

- fixed setting of executable mode on client script rocblas_gentest.py to avoid potential permission errors with clients rocblas-test and rocblas-bench
- fixed deprecated API compatibility with Visual Studio compiler
- fixed test framework memory exception handling for Level 2 functions when the host memory allocation exceeds the available memory

8.6.4.9.4 Changed

- install.sh internally runs rmake.py (also used on windows) and rmake.py may be used directly by developers on linux (use –help)
- rocblas client executables all now begin with rocblas- prefix

8.6.4.9.5 Removed

- install.sh removed options -o –cov as now Tensile will use the default COV format, set by cmake define Tensile_CODE_OBJECT_VERSION=default

8.6.4.10 rocFFT 1.0.22

rocFFT 1.0.22 for ROCm 5.5.0

8.6.4.10.1 Optimizations

- Improved performance of 1D lengths < 2048 that use Bluestein's algorithm.
- Reduced time for generating code during plan creation.
- Optimized 3D R2C/C2R lengths 32, 84, 128.
- Optimized batched small 1D R2C/C2R cases.

8.6.4.10.2 Added

- Added gfx1101 to default AMDGPU_TARGETS.

8.6.4.10.3 Changed

- Moved client programs to C++17.
- Moved planar kernels and infrequently used Stockham kernels to be runtime-compiled.
- Moved transpose, real-complex, Bluestein, and Stockham kernels to library kernel cache.

8.6.4.10.4 Fixed

- Removed zero-length twiddle table allocations, which fixes errors from hipMallocManaged.
- Fixed incorrect freeing of HIP stream handles during twiddle computation when multiple devices are present.

8.6.4.11 rocm-cmake 0.8.1

rocm-cmake 0.8.1 for ROCm 5.5.0

8.6.4.11.1 Fixed

- ROCMInstallTargets: Added compatibility symlinks for included cmake files in <ROCM>/lib/cmake/<PACKAGE>.

8.6.4.11.2 Changed

- ROCMHeaderWrapper: The wrapper header deprecation message is now a deprecation warning.

8.6.4.12 rocPRIM 2.13.0

rocPRIM 2.13.0 for ROCm 5.5.0

8.6.4.12.1 Added

- New block level radix_rank primitive.
- New block level radix_rank_match primitive.

8.6.4.12.2 Changed

- Improved the performance of block_radix_sort and device_radix_sort.

8.6.4.12.3 Known Issues

- Disabled GPU error messages relating to incorrect warp operation usage with Navi GPUs on Windows, due to GPU printf performance issues on Windows.

8.6.4.12.4 Fixed

- Fixed benchmark build on Windows

8.6.4.13 rocRAND 2.10.17

rocRAND 2.10.17 for ROCm 5.5.0

8.6.4.13.1 Added

- MT19937 pseudo random number generator based on M. Matsumoto and T. Nishimura, 1998, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator.
- New benchmark for the device API using Google Benchmark, benchmark_rocrand_device_api, replacing benchmark_rocrand_kernel. benchmark_rocrand_kernel is deprecated and will be removed in a future version. Likewise, benchmark_curand_host_api is added to replace benchmark_curand_generate and benchmark_curand_device_api is added to replace benchmark_curand_kernel.
- experimental HIP-CPU feature
- ThreeFry pseudorandom number generator based on Salmon et al., 2011, "Parallel random numbers: as easy as 1, 2, 3".

8.6.4.13.2 Changed

- Python 2.7 is no longer officially supported.

8.6.4.13.3 Fixed

- Windows HIP SDK support

8.6.4.14 rocSOLVER 3.21.0

rocSOLVER 3.21.0 for ROCm 5.5.0

8.6.4.14.1 Added

- SVD for general matrices using Jacobi algorithm:
  - GESVDJ (with batched and strided_batched versions)
- LU factorization without pivoting for block tridiagonal matrices:
  - GEBLTTRF_NPVT (with batched and strided_batched versions)
- Linear system solver without pivoting for block tridiagonal matrices:
  - GEBLTTRS_NPVT (with batched and strided_batched, versions)
- Product of triangular matrices
  - LAUUM
- Added experimental hipGraph support for rocSOLVER functions

8.6.4.14.2 Optimized

- Improved the performance of SYEVJ/HEEVJ.

8.6.4.14.3 Changed

- STEDC, SYEVD/HEEVD and SYGVD/HEGVD now use fully implemented Divide and Conquer approach.

8.6.4.14.4 Fixed

- SYEVJ/HEEVJ should now be invariant under matrix scaling.
- SYEVJ/HEEVJ should now properly output the eigenvalues when no sweeps are executed.
- Fixed GETF2_NPVT and GETRF_NPVT input data initialization in tests and benchmarks.
- Fixed rocblas missing from the dependency list of the rocsolver deb and rpm packages.

8.6.4.15 rocSPARSE 2.5.1

rocSPARSE 2.5.1 for ROCm 5.5.0

8.6.4.15.1 Added

- Added bsrgemm and spgemm for BSR format
- Added bsrgeam
- Added build support for Navi32
- Added experimental hipGraph support for some rocSPARSE routines
- Added csritsv, spitsv csr iterative triangular solve
- Added mixed precisions for SpMV
- Added batched SpMM for transpose A in COO format with atomic atomic algorithm

8.6.4.15.2 Improved

- Optimization to csr2bsr
- Optimization to csr2csr_compress
- Optimization to csr2coo
- Optimization to gebsr2csr
- Optimization to csr2gebsr
- Fixes to documentation
- Fixes a bug in COO SpMV gridsize
- Fixes a bug in SpMM gridsize when using very large matrices

8.6.4.15.3 Known Issues

- In csritlu0, the algorithm rocsparse_itilu0_alg_sync_split_fusion has some accuracy issues to investigate with XNACK enabled. The fallback is rocsparse_itilu0_alg_sync_split.

8.6.4.16 rocWMMA 1.0

rocWMMA 1.0 for ROCm 5.5.0

8.6.4.16.1 Added

- Added support for wave32 on gfx11+
- Added infrastructure changes to support hipRTC
- Added performance tracking system

8.6.4.16.2 Changed

- Modified the assignment of hardware information
- Modified the data access for unsigned datatypes
- Added library config to support multiple architectures

8.6.4.17 Tensile 4.36.0

Tensile 4.36.0 for ROCm 5.5.0

8.6.4.17.1 Added

- Add functions for user-driven tuning
- Add GFX11 support: HostLibraryTests yamls, rearragne FP32©/FP64© instruction order, archCaps for instruction renaming condition, adjust vgpr bank for A/B/C for optimize, separate vscnt and vmcnt, dual mac
- Add binary search for Grid-Based algorithm
- Add reject condition for (StoreCInUnroll + BufferStore=0) and (DirectToVgpr + ScheduleIterAlg<3 + PrefetchGlobalRead==2)
- Add support for (DirectToLds + hgemm + NN/NT/TT) and (DirectToLds + hgemm + GlobalLoadVectorWidth < 4)
- Add support for (DirectToLds + hgemm(TLU=True only) or sgemm + NumLoadsCoalesced > 1)
- Add GSU SingleBuffer algorithm for HSS/BSS
- Add gfx900:xnack-, gfx1032, gfx1034, gfx1035
- Enable gfx1031 support

8.6.4.17.2 Optimizations

- Use AssertSizeLessThan for BufferStoreOffsetLimitCheck if it is smaller than MT1
- Improve InitAccVgprOpt

8.6.4.17.3 Changed

- Use global_atomic for GSU instead of flat and global_store for debug code
- Replace flat_load/store with global_load/store
- Use global_load/store for BufferLoad/Store=0 and enable scheduling
- LocalSplitU support for HGEMM+HPA when MFMA disabled
- Update Code Object Version
- Type cast local memory to COMPUTE_DATA_TYPE in LDS to avoid precision loss
- Update asm cap cache arguments
- Unify SplitGlobalRead into ThreadSeparateGlobalRead and remove SplitGlobalRead
- Change checks, error messages, assembly syntax, and coverage for DirectToLds
- Remove unused cmake file
- Clean up the LLVM dependency code
- Update ThreadSeparateGlobalRead test cases for PrefetchGlobalRead=2
- Update sgemm/hgemm test cases for DirectToLds and ThreadSepareteGlobalRead

8.6.4.17.4 Fixed

- Add build-id to header of compiled source kernels
- Fix solution index collisions
- Fix h beta vectorwidth4 correctness issue for WMMA
- Fix an error with BufferStore=0
- Fix mismatch issue with (StoreCInUnroll + PrefetchGlobalRead=2)
- Fix MoveMIoutToArch bug
- Fix flat load correctness issue on I8 and flat store correctness issue
- Fix mismatch issue with BufferLoad=0 + TailLoop for large array sizes
- Fix code generation error with BufferStore=0 and StoreCInUnrollPostLoop
- Fix issues with DirectToVgpr + ScheduleIterAlg<3
- Fix mismatch issue with DGEMM TT + LocalReadVectorWidth=2
- Fix mismatch issue with PrefetchGlobalRead=2
- Fix mismatch issue with DirectToVgpr + PrefetchGlobalRead=2 + small tile size
- Fix an error with PersistentKernel=0 + PrefetchAcrossPersistent=1 + PrefetchAcrossPersistentMode=1

- Fix mismatch issue with DirectToVgpr + DirectToLds + only 1 iteration in unroll loop case
- Remove duplicate GSU kernels: for GSU = 1, GSUAlgorithm SingleBuffer and MultipleBuffer kernels are identical
- Fix for failing CI tests due to CpuThreads=0
- Fix mismatch issue with DirectToLds + PrefetchGlobalRead=2
- Remove the reject condition for ThreadSeparateGlobalRead and DirectToLds (HGEMM, SGEMM only)
- Modify reject condition for minimum lanes of ThreadSeparateGlobalRead (SGEMM or larger data type only)

8.7 ROCm 5.4.3

8.7.1 Deprecations and Warnings

8.7.1.1 HIP Perl Scripts Deprecation

The hipcc and hipconfig Perl scripts are deprecated. In a future release, compiled binaries will be available as hipcc.bin and hipconfig.bin as replacements for the Perl scripts.

> Note
>
> There will be a transition period where the Perl scripts and compiled binaries are available before the scripts are removed. There will be no functional difference between the Perl scripts and their compiled binary counterpart. No user action is required. Once these are available, users can optionally switch to hipcc.bin and hipconfig.bin. The hipcc/hipconfig soft link will be assimilated to point from hipcc/hipconfig to the respective compiled binaries as the default option.

8.7.1.1.1 Linux Filesystem Hierarchy Standard for ROCm

ROCm packages have adopted the Linux foundation filesystem hierarchy standard in this release to ensure ROCm components follow open source conventions for Linux-based distributions. While moving to a new filesystem hierarchy, ROCm ensures backward compatibility with its 5.1 version or older filesystem hierarchy. See below for a detailed explanation of the new filesystem hierarchy and backward compatibility.

8.7.1.1.2 New Filesystem Hierarchy

The following is the new filesystem hierarchy:4

```
/opt/rocm-<ver>
    | --bin
      | --All externally exposed Binaries
    | --libexec
       | --<component>
           | -- Component specific private non-ISA executables (architecture independent)
    | --include
       | -- <component>
```

(continues on next page)

(continued from previous page)

```
        | --<header files>
  | --lib
      | --lib<soname>.so -> lib<soname>.so.major -> lib<soname>.so.major.minor.patch
        (public libraries linked with application)
      | --<component> (component specific private library, executable data)
      | --<cmake>
        | --components
          | --<component>.config.cmake
  | --share
      | --html/<component>/*.html
      | --info/<component>/*.[pdf, md, txt]
      | --man
      | --doc
        | --<component>
          | --<licenses>
      | --<component>
        | --<misc files> (arch independent non-executable)
        | --samples
```

Note

ROCm will not support backward compatibility with the v5.1(old) file system hierarchy in its next major release.

For more information, refer to https://refspecs.linuxfoundation.org/fhs.shtml.

8.7.1.1.3 Backward Compatibility with Older Filesystems

ROCm has moved header files and libraries to its new location as indicated in the above structure and included symbolic-link and wrapper header files in its old location for backward compatibility.

Note

ROCm will continue supporting backward compatibility until the next major release.

8.7.1.1.4 Wrapper header files

Wrapper header files are placed in the old location (/opt/rocm-xxx/<component>/include) with a warning message to include files from the new location (/opt/rocm-xxx/include) as shown in the example below:

```
// Code snippet from hip_runtime.h
#pragma message “This file is deprecated. Use file from include path /opt/rocm-ver/include/ and prefix with hip”.
#include ”hip/hip_runtime.h”
```

The wrapper header files’ backward compatibility deprecation is as follows:

- #pragma message announcing deprecation – ROCm v5.2 release
- #pragma message changed to #warning – Future release
- #warning changed to #error – Future release
- Backward compatibility wrappers removed – Future release

8.7.1.1.5 Library files

Library files are available in the /opt/rocm-xxx/lib folder. For backward compatibility, the old library location (/opt/rocm-xxx/<component>/lib) has a soft link to the library at the new location.

Example:

```
$ ls -l /opt/rocm/hip/lib/
total 4
drwxr-xr-x 4 root root 4096 May 12 10:45 cmake
lrwxrwxrwx 1 root root   24 May 10 23:32 libamdhip64.so -> ../../lib/libamdhip64.so
```

8.7.1.1.6 CMake Config files

All CMake configuration files are available in the /opt/rocm-xxx/lib/cmake/<component> folder. For backward compatibility, the old CMake locations (/opt/rocm-xxx/<component>/lib/cmake) consist of a soft link to the new CMake config.

Example:

```
$ ls -l /opt/rocm/hip/lib/cmake/hip/
total 0
lrwxrwxrwx 1 root root 42 May 10 23:32 hip-config.cmake -> ../../../../lib/cmake/hip/hip-config.cmake
```

8.7.2 Fixed Defects

8.7.2.1 Compiler Improvements

In ROCm v5.4.3, improvements to the compiler address errors with the following signatures:

- “error: unhandled SGPR spill to memory”
- “cannot scavenge register without an emergency spill slot!”
- “error: ran out of registers during register allocation”

8.7.3 Known Issues

8.7.3.1 Compiler Option Error at Runtime

Some users may encounter a “Cannot find Symbol” error at runtime when using -save-temps. While most -save-temps use cases work correctly, this error may appear occasionally.

This issue is under investigation, and the known workaround is not to use -save-temps when the error appears.

8.7.4 Library Changes in ROCM 5.4.3

Library	Version
hipBLAS	0.53.0
hipCUB	2.13.0
hipFFT	1.0.10
hipSOLVER	1.6.0
hipSPARSE	2.3.3
rccl	2.13.4
rocALUTION	2.1.3
rocBLAS	2.46.0
rocFFT	1.0.20 1.0.21
rocm-cmake	0.8.0
rocPRIM	2.12.0
rocRAND	2.10.16
rocSOLVER	3.20.0
rocSPARSE	2.4.0
rocThrust	2.17.0
rocWMMA	0.9
Tensile	4.35.0

8.7.4.1 rocFFT 1.0.21

rocFFT 1.0.21 for ROCm 5.4.3

8.7.4.1.1 Fixed

- Removed source directory from rocm_install_targets call to prevent installation of rocfft.h in an unintended location.

8.8 ROCm 5.4.2

8.8.1 Deprecations and Warnings

8.8.1.1 HIP Perl Scripts Deprecation

The hipcc and hipconfig Perl scripts are deprecated. In a future release, compiled binaries will be available as hipcc.bin and hipconfig.bin as replacements for the Perl scripts.

> Note
>
> There will be a transition period where the Perl scripts and compiled binaries are available before the scripts are removed. There will be no functional difference between the Perl scripts and their compiled binary counterpart. No user action is required. Once these are available, users can optionally switch to hipcc.bin and hipconfig.bin. The hipcc/hipconfig soft link will be assimilated to point from hipcc/hipconfig to the respective compiled binaries as the default option.

8.8.1.2 hipcc Options Deprecation

The following hipcc options are being deprecated and will be removed in a future release:

- The --amdgpu-target option is being deprecated, and user must use the –offload-arch option to specify the GPU architecture.
- The --amdhsa-code-object-version option is being deprecated. Users can use the Clang/LLVM option -mllvm -mcode-object-version to debug issues related to code object versions.
- The --hipcc-func-supp/--hipcc-no-func-supp options are being deprecated, as the function calls are already supported in production on AMD GPUs.

8.8.2 Known Issues

Under certain circumstances typified by high register pressure, users may encounter a compiler abort with one of the following error messages:

- error: unhandled SGPR spill to memory
- cannot scavenge register without an emergency spill slot!
- error: ran out of registers during register allocation

This is a known issue and will be fixed in a future release.

8.8.3 Library Changes in ROCM 5.4.2

Library	Version
hipBLAS	0.53.0
hipCUB	2.13.0
hipFFT	1.0.10
hipSOLVER	1.6.0
hipSPARSE	2.3.3
rccl	2.13.4
rocALUTION	2.1.3
rocBLAS	2.46.0
rocFFT	1.0.20
rocm-cmake	0.8.0
rocPRIM	2.12.0
rocRAND	2.10.16
rocSOLVER	3.20.0
rocSPARSE	2.4.0
rocThrust	2.17.0
rocWMMA	0.9
Tensile	4.35.0

8.9 ROCm 5.4.1

8.9.1 What's New in This Release

8.9.1.1 HIP Enhancements

The ROCm v5.4.1 release consists of the following new HIP API:

8.9.1.1.1 New HIP API - hipLaunchHostFunc

The following new HIP API is introduced in the ROCm v5.4.1 release.

> Note
>
> This is a pre-official version (beta) release of the new APIs.

```
hipError_t hipLaunchHostFunc(hipStream_t stream, hipHostFn_t fn, void* userData);
```

This swaps the stream capture mode of a thread.

```
@param [in] mode - Pointer to mode value to swap with the current mode
```

This parameter returns #hipSuccess, #hipErrorInvalidValue.

For more information, refer to the HIP API documentation at /bundle/HIP_API_Guide/page/modules.html.

8.9.2 Deprecations and Warnings

8.9.2.1 HIP Perl Scripts Deprecation

The hipcc and hipconfig Perl scripts are deprecated. In a future release, compiled binaries will be available as hipcc.bin and hipconfig.bin as replacements for the Perl scripts.

> Note
>
> There will be a transition period where the Perl scripts and compiled binaries are available before the scripts are removed. There will be no functional difference between the Perl scripts and their compiled binary counterpart. No user action is required. Once these are available, users can optionally switch to hipcc.bin and hipconfig.bin. The hipcc/hipconfig soft link will be assimilated to point from hipcc/hipconfig to the respective compiled binaries as the default option.

8.9.3 IFWI Fixes

These defects were identified and documented as known issues in previous ROCm releases and are fixed in this release. AMD Instinct™ MI200 Firmware IFWI Maintenance Update #3

This IFWI release fixes the following issue in AMD Instinct™ MI210/MI250 Accelerators.

After prolonged periods of operation, certain MI200 Instinct™ Accelerators may perform in a degraded way resulting in application failures.

In this package, AMD delivers a new firmware version for MI200 GPU accelerators and a firmware installation tool – AMD FW FLASH 1.2.

GPU	Production Part Number	SKU	IFWI Name
MI210	113-D673XX	D67302	D6730200V.110
MI210	113-D673XX	D67301	D6730100V.073
MI250	113-D652XX	D65209	D6520900.073
MI250	113-D652XX	D65210	D6521000.073

Instructions on how to download and apply MI200 maintenance updates are available at:

https://www.amd.com/en/support/server-accelerators/amd-instinct/amd-instinct-mi-series/amd-instinct-mi210

8.9.3.1 AMD Instinct™ MI200 SRIOV Virtualization Support

Maintenance update #3, combined with ROCm 5.4.1, now provides SRIOV virtualization support for all AMD Instinct™ MI200 devices.

8.9.4 Library Changes in ROCM 5.4.1

Library	Version	
hipBLAS	0.53.0	
hipCUB	2.13.0	
hipFFT	1.0.10	
hipSOLVER	1.6.0	
hipSPARSE	2.3.3	
rccl	2.13.4	
rocALUTION	2.1.3	
rocBLAS	2.46.0	
rocFFT	1.0.19	1.0.20
rocm-cmake	0.8.0	
rocPRIM	2.12.0	
rocRAND	2.10.16	
rocSOLVER	3.20.0	
rocSPARSE	2.4.0	
rocThrust	2.17.0	
rocWMMA	0.9	
Tensile	4.35.0	

8.9.4.1 rocFFT 1.0.20

rocFFT 1.0.20 for ROCm 5.4.1

8.9.4.1.1 Fixed

- Fixed incorrect results on strided large 1D FFTs where batch size does not equal the stride.

8.10 ROCm 5.4.0

8.10.1 What's New in This Release

8.10.1.1 HIP Enhancements

The ROCm v5.4 release consists of the following HIP enhancements:

8.10.1.1.1 Support for Wall Clock64

A new timer function wall_clock64() is supported, which returns wall clock count at a constant frequency on the device.

```
long long int wall_clock64();
```

It returns wall clock count at a constant frequency on the device, which can be queried via HIP API with the hipDeviceAttributeWallClockRate attribute of the device in the HIP application code.

Example:

```
int wallClkRate = 0; //in kilohertz
+HIPCHECK(hipDeviceGetAttribute(&wallClkRate, hipDeviceAttributeWallClockRate, deviceId));
```

Where hipDeviceAttributeWallClockRate is a device attribute.

> Note
>
> The wall clock frequency is a per-device attribute.

8.10.1.1.2 New Registry Added for GPU_MAX_HW_QUEUES

The GPU_MAX_HW_QUEUES registry defines the maximum number of independent hardware queues allocated per process per device.

The environment variable controls how many independent hardware queues HIP runtime can create per process, per device. If the application allocates more HIP streams than this number, then the HIP runtime reuses the same hardware queues for the new streams in a round-robin manner.

> Note
>
> This maximum number does not apply to hardware queues created for CU-masked HIP streams or cooperative queues for HIP Cooperative Groups (there is only one queue per device).

For more details, refer to the HIP Programming Guide.

8.10.1.2 New HIP APIs in This Release

The following new HIP APIs are available in the ROCm v5.4 release.

> Note
>
> This is a pre-official version (beta) release of the new APIs.

8.10.1.2.1 Error Handling

```
hipError_t hipDrvGetErrorName(hipError_t hipError, const char** errorString);
```

This returns HIP errors in the text string format.

```
hipError_t hipDrvGetErrorString(hipError_t hipError, const char** errorString);
```

This returns text string messages with more details about the error.

For more information, refer to the HIP API Guide.

8.10.1.2.2 HIP Tests Source Separation

With ROCm v5.4, a separate GitHub project is created at

https://github.com/ROCm-Developer-Tools/hip-tests

This contains HIP catch2 tests and samples, and new tests will continue to develop.

In future ROCm releases, catch2 tests and samples will be removed from the HIP project.

8.10.2 OpenMP Enhancements

This release consists of the following OpenMP enhancements:

- Enable new device RTL in libomptarget as default.
- New flag -fopenmp-target-fast to imply -fopenmp-target-ignore-env-vars -fopenmp-assume-no-thread-state -fopenmp-assume-no-nested-parallelism.
- Support for the collapse clause and non-unit stride in cases where the No-Loop specialized kernel is generated.
- Initial implementation of optimized cross-team sum reduction for float and double type scalars.
- Pool-based optimization in the OpenMP runtime to reduce locking during data transfer.

8.10.3 Deprecations and Warnings

8.10.3.1 HIP Perl Scripts Deprecation

The hipcc and hipconfig Perl scripts are deprecated. In a future release, compiled binaries will be available as hipcc.bin and hipconfig.bin as replacements for the Perl scripts.

Note

There will be a transition period where the Perl scripts and compiled binaries are available before the scripts are removed. There will be no functional difference between the Perl scripts and their compiled binary counterpart. No user action is required. Once these are available, users can optionally switch to hipcc.bin and hipconfig.bin. The hipcc/hipconfig soft link will be assimilated to point from hipcc/hipconfig to the respective compiled binaries as the default option.

8.10.3.1.1 Linux Filesystem Hierarchy Standard for ROCm

ROCm packages have adopted the Linux foundation filesystem hierarchy standard in this release to ensure ROCm components follow open source conventions for Linux-based distributions. While moving to a new filesystem hierarchy, ROCm ensures backward compatibility with its 5.1 version or older filesystem hierarchy. See below for a detailed explanation of the new filesystem hierarchy and backward compatibility.

8.10.3.1.2 New Filesystem Hierarchy

The following is the new filesystem hierarchy:

```
/opt/rocm-<ver>
   | --bin
     | --All externally exposed Binaries
   | --libexec
      | --<component>
         | -- Component specific private non-ISA executables (architecture independent)
   | --include
      | -- <component>
         | --<header files>
   | --lib
      | --lib<soname>.so -> lib<soname>.so.major -> lib<soname>.so.major.minor.patch
         (public libraries linked with application)
      | --<component> (component specific private library, executable data)
      | --<cmake>
         | --components
            | --<component>.config.cmake
   | --share
      | --html/<component>/*.html
      | --info/<component>/*.[pdf, md, txt]
      | --man
      | --doc
         | --<component>
            | --<licenses>
      | --<component>
         | --<misc files> (arch independent non-executable)
         | --samples
```

Note

ROCm will not support backward compatibility with the v5.1(old) file system hierarchy in its next major release.

For more information, refer to https://refspecs.linuxfoundation.org/fhs.shtml.

8.10.3.1.3 Backward Compatibility with Older Filesystems

ROCm has moved header files and libraries to its new location as indicated in the above structure and included symbolic-link and wrapper header files in its old location for backward compatibility.

> Note
>
> ROCm will continue supporting backward compatibility until the next major release.

8.10.3.1.4 Wrapper header files

Wrapper header files are placed in the old location (/opt/rocm-xxx/<component>/include) with a warning message to include files from the new location (/opt/rocm-xxx/include) as shown in the example below:

```
// Code snippet from hip_runtime.h
#pragma message “This file is deprecated. Use file from include path /opt/rocm-ver/include/ and prefix with hip”.
#include ”hip/hip_runtime.h”
```

The wrapper header files' backward compatibility deprecation is as follows:

- #pragma message announcing deprecation – ROCm v5.2 release
- #pragma message changed to #warning – Future release
- #warning changed to #error – Future release
- Backward compatibility wrappers removed – Future release

8.10.3.1.5 Library files

Library files are available in the /opt/rocm-xxx/lib folder. For backward compatibility, the old library location (/opt/rocm-xxx/<component>/lib) has a soft link to the library at the new location.

Example:

```
$ ls -l /opt/rocm/hip/lib/
total 4
drwxr-xr-x 4 root root 4096 May 12 10:45 cmake
lrwxrwxrwx 1 root root   24 May 10 23:32 libamdhip64.so -> ../../lib/libamdhip64.so
```

8.10.3.1.6 CMake Config files

All CMake configuration files are available in the /opt/rocm-xxx/lib/cmake/<component> folder. For backward compatibility, the old CMake locations (/opt/rocm-xxx/<component>/lib/cmake) consist of a soft link to the new CMake config.

Example:

```
$ ls -l /opt/rocm/hip/lib/cmake/hip/
total 0
lrwxrwxrwx 1 root root 42 May 10 23:32 hip-config.cmake -> ../../../../lib/cmake/hip/hip-config.cmake
```

8.10.4 Fixed Defects

The following defects are fixed in this release.

These defects were identified and documented as known issues in previous ROCm releases and are fixed in this release.

8.10.4.1 Memory Allocated Using hipHostMalloc() with Flags Did Not Exhibit Fine-Grain Behavior

8.10.4.1.1 Issue

The test was incorrectly using the hipDeviceAttributePageableMemoryAccess device attribute to determine coherent support.

8.10.4.1.2 Fix

hipHostMalloc() allocates memory with fine-grained access by default when the environment variable HIP_HOST_COHERENT=1 is used.

For more information, refer to HIP Runtime API Reference.

8.10.4.2 SoftHang with hipStreamWithCUMask test on AMD Instinct™

8.10.4.2.1 Issue

On GFX10 GPUs, kernel execution hangs when it is launched on streams created using hipStreamWithCUMask.

8.10.4.2.2 Fix

On GFX10 GPUs, each workgroup processor encompasses two compute units, and the compute units must be enabled as a pair. The hipStreamWithCUMask API unit test cases are updated to set compute unit mask (cuMask) in pairs for GFX10 GPUs.

8.10.4.3 ROCm Tools GPU IDs

The HIP language device IDs are not the same as the GPU IDs reported by the tools. GPU IDs are globally unique and guaranteed to be consistent across APIs and processes.

GPU IDs reported by ROCTracer and ROCProfiler or ROCm Tools are HSA Driver Node ID of that GPU, as it is a unique ID for that device in that particular node.

8.10.5 Library Changes in ROCM 5.4.0

Library	Version
hipBLAS	0.52.0 → 0.53.0
hipCUB	2.12.0 → 2.13.0
hipFFT	1.0.9 → 1.0.10
hipSOLVER	1.5.0 → 1.6.0
hipSPARSE	2.3.1 → 2.3.3
rccl	2.12.10 → 2.13.4
rocALUTION	2.1.0 → 2.1.3
rocBLAS	2.45.0 → 2.46.0
rocFFT	1.0.18 → 1.0.19
rocm-cmake	→ 0.8.0
rocPRIM	2.11.0 → 2.12.0
rocRAND	2.10.15 → 2.10.16
rocSOLVER	3.19.0 → 3.20.0
rocSPARSE	2.2.0 → 2.4.0
rocThrust	2.16.0 → 2.17.0
rocWMMA	0.8 → 0.9
Tensile	4.34.0 → 4.35.0

8.10.5.1 hipBLAS 0.53.0

hipBLAS 0.53.0 for ROCm 5.4.0

8.10.5.1.1 Added

- Allow for selection of int8 datatype
- Added support for hipblasXgels and hipblasXgelsStridedBatched operations (with s,d,c,z precisions), only supported with rocBLAS backend
- Added support for hipblasXgelsBatched operations (with s,d,c,z precisions)

8.10.5.2 hipCUB 2.13.0

hipCUB 2.13.0 for ROCm 5.4.0

8.10.5.2.1 Added

- CMake functionality to improve build parallelism of the test suite that splits compilation units by function or by parameters.
- New overload for BlockAdjacentDifference::SubtractLeftPartialTile that takes a predecessor item.

8.10.5.2.2 Changed

- Improved build parallelism of the test suite by splitting up large compilation units for DeviceRadixSort, DeviceSegmentedRadixSort and DeviceSegmentedSort.
- CUB backend references CUB and thrust version 1.17.1.

8.10.5.3 hipFFT 1.0.10

hipFFT 1.0.10 for ROCm 5.4.0

8.10.5.3.1 Added

- Added hipfftExtPlanScaleFactor API to efficiently multiply each output element of a FFT by a given scaling factor. Result scaling must be supported in the backend FFT library.

8.10.5.3.2 Changed

- When hipFFT is built against the rocFFT backend, rocFFT 1.0.19 or higher is now required.

8.10.5.4 hipSOLVER 1.6.0

hipSOLVER 1.6.0 for ROCm 5.4.0

8.10.5.4.1 Added

- Added compatibility-only functions
  - gesvdaStridedBatched
    - hipsolverDnSgesvdaStridedBatched_bufferSize, hipsolverDnDgesvdaStridedBatched_bufferSize, hipsolverDnCgesvdaStridedBatched_bufferSize, hipsolverDnZgesvdaStridedBatched_bufferSize
    - hipsolverDnSgesvdaStridedBatched, hipsolverDnDgesvdaStridedBatched, hipsolverDnCgesvdaStridedBatched, hipsolverDnZgesvdaStridedBatched

8.10.5.5 hipSPARSE 2.3.3

hipSPARSE 2.3.3 for ROCm 5.4.0

8.10.5.5.1 Added

- Added hipsparseCsr2cscEx2_bufferSize and hipsparseCsr2cscEx2 routines

8.10.5.5.2 Changed

- HIPSPARSE_ORDER_COLUMN has been renamed to HIPSPARSE_ORDER_COL to match cusparse

8.10.5.6 rccl 2.13.4

RCCL 2.13.4 for ROCm 5.4.0

8.10.5.6.1 Changed

- Compatibility with NCCL 2.13.4
- Improvements to RCCL when running with hipGraphs
- RCCL_ENABLE_HIPGRAPH environment variable is no longer necessary to enable hipGraph support
- Minor latency improvements

8.10.5.6.2 Fixed

- Resolved potential memory access error due to asynchronous memset

8.10.5.7 rocALUTION 2.1.3

rocALUTION 2.1.3 for ROCm 5.4.0

8.10.5.7.1 Added

- Added build support for Navi31 and Navi33
- Added support for non-squared global matrices

8.10.5.7.2 Improved

- Fixed a memory leak in MatrixMult on HIP backend
- Global structures can now be used with a single process

8.10.5.7.3 Changed

- Switched GTest death test style to 'threadsafe'
- GlobalVector::GetGhostSize() is deprecated and will be removed
- ParallelManager::GetGlobalSize(), ParallelManager::GetLocalSize(), ParallelManager::SetGlobalSize() and ParallelManager::SetLocalSize() are deprecated and will be removed
- Vector::GetGhostSize() is deprecated and will be removed
- Multigrid::SetOperatorFormat(unsigned int) is deprecated and will be removed, use Multigrid::SetOperatorFormat(unsigned int, int) instead
- RugeStuebenAMG::SetCouplingStrength(ValueType) is deprecated and will be removed, use SetStrengthThreshold(float) instead

8.10.5.8 rocBLAS 2.46.0

rocBLAS 2.46.0 for ROCm 5.4.0

8.10.5.8.1 Added

- client smoke test dataset added for quick validation using command rocblas-test –yaml rocblas_smoke.yaml
- Added stream order device memory allocation as a non-default beta option.

8.10.5.8.2 Optimized

- Improved trsm performance for small sizes by using a substitution method technique
- Improved syr2k and her2k performance significantly by using a block-recursive algorithm

8.10.5.8.3 Changed

- Level 2, Level 1, and Extension functions: argument checking when the handle is set to rocblas_pointer_mode_host now returns the status of rocblas_status_invalid_pointer only for pointers that must be dereferenced based on the alpha and beta argument values. With handle mode rocblas_pointer_mode_device only pointers that are always dereferenced regardless of alpha and beta values are checked and so may lead to a return status of rocblas_status_invalid_pointer. This improves consistency with legacy BLAS behaviour.
- Add variable to turn on/off ieee16/ieee32 tests for mixed precision gemm
- Allow hipBLAS to select int8 datatype
- Disallow B == C && ldb != ldc in rocblas_xtrmm_outofplace

8.10.5.8.4 Fixed

- FORTRAN interfaces generalized for FORTRAN compilers other than gfortran
- fix for trsm_strided_batched rocblas-bench performance gathering
- Fix for rocm-smi path in commandrunner.py script to match ROCm 5.2 and above

8.10.5.9 rocFFT 1.0.19

rocFFT 1.0.19 for ROCm 5.4.0

8.10.5.9.1 Optimizations

- Optimized some strided large 1D plans.

8.10.5.9.2 Added

- Added rocfft_plan_description_set_scale_factor API to efficiently multiply each output element of a FFT by a given scaling factor.
- Created a rocfft_kernel_cache.db file next to the installed library. SBCC kernels are moved to this file when built with the library, and are runtime-compiled for new GPU architectures.
- Added gfx1100 and gfx1102 to default AMDGPU_TARGETS.

8.10.5.9.3 Changed

- Moved runtime compilation cache to in-memory by default. A default on-disk cache can encounter contention problems on multi-node clusters with a shared filesystem. rocFFT can still be told to use an on-disk cache by setting the ROCFFT_RTC_CACHE_PATH environment variable.

8.10.5.10 rocPRIM 2.12.0

rocPRIM 2.12.0 for ROCm 5.4.0

8.10.5.10.1 Changed

- device_partition, device_unique, and device_reduce_by_key now support problem sizes larger than 2^32 items.

8.10.5.10.2 Removed

- block_sort::sort() overload for keys and values with a dynamic size. This overload was documented but the implementation is missing. To avoid further confusion the documentation is removed until a decision is made on implementing the function.

8.10.5.10.3 Fixed

- Fixed the compilation failure in device_merge if the two key iterators don't match.

8.10.5.11 rocRAND 2.10.16

rocRAND 2.10.16 for ROCm 5.4.0

8.10.5.11.1 Added

- MRG31K3P pseudorandom number generator based on L'Ecuyer and Touzin, 2000, "Fast combined multiple recursive generators with multipliers of the form a = ±2q ±2r".
- LFSR113 pseudorandom number generator based on L'Ecuyer, 1999, "Tables of maximally equidistributed combined LFSR generators".
- SCRAMBLED_SOBOL32 and SCRAMBLED_SOBOL64 quasirandom number generators. The Scrambled Sobol sequences are generated by scrambling the output of a Sobol sequence.

8.10.5.11.2 Changed

- The mrg_<distribution>_distribution structures, which provided numbers based on MRG32K3A, are now replaced by mrg_engine_<distribution>_distribution, where < distribution> is log_normal, normal, poisson, or uniform. These structures provide numbers for MRG31K3P (with template type rocrand_state_mrg31k3p) and MRG32K3A (with template type rocrand_state_mrg32k3a).

8.10.5.11.3 Fixed

- Sobol64 now returns 64 bits random numbers, instead of 32 bits random numbers. As a result, the performance of this generator has regressed.
- Fixed a bug that prevented compiling code in C++ mode (with a host compiler) when it included the rocRAND headers on Windows.

8.10.5.12 rocSOLVER 3.20.0

rocSOLVER 3.20.0 for ROCm 5.4.0

8.10.5.12.1 Added

- Partial SVD for bidiagonal matrices:
  - BDSVDX
- Partial SVD for general matrices:
  - GESVDX (with batched and strided_batched versions)

8.10.5.12.2 Changed

- Changed ROCSOLVER_EMBED_FMT default to ON for users building directly with CMake. This matches the existing default when building with install.sh or rmake.py.

8.10.5.13 rocSPARSE 2.4.0

rocSPARSE 2.4.0 for ROCm 5.4.0

8.10.5.13.1 Added

- Added rocsparse_spmv_ex routine
- Added rocsparse_bsrmv_ex_analysis and rocsparse_bsrmv_ex routines
- Added csritilu0 routine
- Added build support for Navi31 and Navi 33

8.10.5.13.2 Improved

- Optimization to segmented algorithm for COO SpMV by performing analysis
- Improve performance when generating random matrices.
- Fixed bug in ellmv
- Optimized bsr2csr routine
- Fixed integer overflow bugs

8.10.5.14 rocThrust 2.17.0

rocThrust 2.17.0 for ROCm 5.4.0

8.10.5.14.1 Added

- Updated to match upstream Thrust 1.17.0

8.10.5.15 rocWMMA 0.9

rocWMMA 0.9 for ROCm 5.4.0

8.10.5.15.1 Added

- Added gemm driver APIs for flow control builtins
- Added benchmark logging systems
- Restructured tests to follow naming convention. Added macros for test generation

8.10.5.15.2 Changed

- Changed CMake to accomodate the modified test infrastructure
- Fine tuned the multi-block kernels with and without lds
- Adjusted Maximum Vector Width to dWordx4 Width
- Updated Efficiencies to display as whole number percentages
- Updated throughput from GFlops/s to TFlops/s
- Reset the ad-hoc tests to use smaller sizes
- Modified the output validation to use CPU-based implementation against rocWMMA
- Modified the extended vector test to return error codes for memory allocation failures

8.10.5.16 Tensile 4.35.0

Tensile 4.35.0 for ROCm 5.4.0

8.10.5.16.1 Added

- Async DMA support for Transpose Data Layout (ThreadSeparateGlobalReadA/B)
- Option to output library logic in dictionary format
- No solution found error message for benchmarking client
- Exact K check for StoreCInUnrollExact
- Support for CGEMM + MIArchVgpr
- client-path parameter for using prebuilt client

- CleanUpBuildFiles global parameter
- Debug flag for printing library logic index of winning solution
- NumWarmups global parameter for benchmarking
- Windows support for benchmarking client
- DirectToVgpr support for CGEMM
- TensileLibLogicToYaml for creating tuning configs from library logic solutions

8.10.5.16.2 Optimizations

- Put beta code and store separately if StoreCInUnroll = x4 store
- Improved performance for StoreCInUnroll + b128 store

8.10.5.16.3 Changed

- Re-enable HardwareMonitor for gfx90a
- Decision trees use MLFeatures instead of Properties

8.10.5.16.4 Fixed

- Reject DirectToVgpr + MatrixInstBM/BN > 1
- Fix benchmark timings when using warmups and/or validation
- Fix mismatch issue with DirectToVgprB + VectorWidth > 1
- Fix mismatch issue with DirectToLds + NumLoadsCoalesced > 1 + TailLoop
- Fix incorrect reject condition for DirectToVgpr
- Fix reject condition for DirectToVgpr + MIWaveTile < VectorWidth
- Fix incorrect instruction generation with StoreCInUnroll

8.11 ROCm 5.3.3

8.11.1 Fixed Defects

8.11.1.1 Issue with rocTHRUST and rocPRIM Libraries

There was a known issue with rocTHRUST and rocPRIM libraries supporting iterator and types in ROCm v5.3.x releases.

- thrust::merge no longer correctly supports different iterator types for keys_input1 and keys_input2.
- rocprim::device_merge no longer correctly supports using different types for keys_input1 and keys_input2.

This issue is resolved with the following fixes to compilation failures:

- rocPRIM: in device_merge if the two key iterators do not match.
- rocTHRUST: in thrust::merge if the two key iterators do not match.

8.11.2 Library Changes in ROCM 5.3.3

Library	Version
hipBLAS	0.52.0
hipCUB	2.12.0
hipFFT	1.0.9
hipSOLVER	1.5.0
hipSPARSE	2.3.1
rccl	2.12.10
rocALUTION	2.1.0
rocBLAS	2.45.0
rocFFT	1.0.18
rocm-cmake	0.8.0
rocPRIM	2.11.0
rocRAND	2.10.15
rocSOLVER	3.19.0
rocSPARSE	2.2.0
rocThrust	2.16.0
rocWMMA	0.8
Tensile	4.34.0

8.12 ROCm 5.3.2

8.12.1 Fixed Defects

The following known issues in ROCm v5.3.2 are fixed in this release.

8.12.1.1 Peer-to-Peer DMA Mapping Errors with SLES and RHEL

Peer-to-Peer Direct Memory Access (DMA) mapping errors on Dell systems (R7525 and R750XA) with SLES 15 SP3/SP4 and RHEL 9.0 are fixed in this release.

Previously, running rocminfo resulted in Peer-to-Peer DMA mapping errors.

8.12.1.2 RCCL Tuning Table

The RCCL tuning table is updated for supported platforms.

8.12.1.3 SGEMM (F32 GEMM) Routines in rocBLAS

Functional correctness failures in SGEMM (F32 GEMM) routines in rocBLAS for certain problem sizes and ranges are fixed in this release.

8.12.2 Known Issues

This section consists of known issues in this release.

8.12.2.1 AMD Instinct™ MI200 SRIOV Virtualization Issue

There is a known issue in this ROCm v5.3 release with all AMD Instinct™ MI200 devices running within a virtual function (VF) under SRIOV virtualization. This issue will likely impact the functionality of SRIOV-based workloads but does not impact Discrete Device Assignment (DDA) or bare metal.

Until a fix is provided, users should rely on ROCm v5.2.3 to support their SRIOV workloads.

8.12.2.2 AMD Instinct™ MI200 Firmware Updates

Customers cannot update the Integrated Firmware Image (IFWI) for AMD Instinct™ MI200 accelerators.

An updated firmware maintenance bundle consisting of an installation tool and images specific to AMD Instinct™ MI200 accelerators is under planning and will be available soon.

8.12.2.3 Known Issue with rocThrust and rocPRIM Libraries

There is a known known issue with rocThrust and rocPRIM libraries supporting iterator and types in ROCm v5.3.x releases.

- thrust::merge no longer correctly supports different iterator types for keys_input1 and keys_input2.
- rocprim::device_merge no longer correctly supports using different types for keys_input1 and keys_input2.

This issue is currently under investigation and will be resolved in a future release.

8.12.3 Library Changes in ROCM 5.3.2

Library	Version
hipBLAS	0.52.0
hipCUB	2.12.0
hipFFT	1.0.9
hipSOLVER	1.5.0
hipSPARSE	2.3.1
rccl	2.12.10
rocALUTION	2.1.0
rocBLAS	2.45.0
rocFFT	1.0.18
rocm-cmake	0.8.0
rocPRIM	2.11.0
rocRAND	2.10.15
rocSOLVER	3.19.0
rocSPARSE	2.2.0
rocThrust	2.16.0
rocWMMA	0.8
Tensile	4.34.0

8.13 ROCm 5.3.0

8.13.1 Deprecations and Warnings

8.13.1.1 HIP Perl Scripts Deprecation

The hipcc and hipconfig Perl scripts are deprecated. In a future release, compiled binaries will be available as hipcc.bin and hipconfig.bin as replacements for the Perl scripts.

> Note
>
> There will be a transition period where the Perl scripts and compiled binaries are available before the scripts are removed. There will be no functional difference between the Perl scripts and their compiled binary counterpart. No user action is required. Once these are available, users can optionally switch to hipcc.bin and hipconfig.bin. The hipcc/hipconfig soft link will be assimilated to point from hipcc/hipconfig to the respective compiled binaries as the default option.

8.13.1.2 Linux Filesystem Hierarchy Standard for ROCm

ROCm packages have adopted the Linux foundation filesystem hierarchy standard in this release to ensure ROCm components follow open source conventions for Linux-based distributions. While moving to a new filesystem hierarchy, ROCm ensures backward compatibility with its 5.1 version or older filesystem hierarchy. See below for a detailed explanation of the new filesystem hierarchy and backward compatibility.

8.13.1.2.1 New Filesystem Hierarchy

The following is the new filesystem hierarchy:

```
/opt/rocm-<ver>
   | --bin
     | --All externally exposed Binaries
   | --libexec
      | --<component>
         | -- Component specific private non-ISA executables (architecture independent)
   | --include
      | -- <component>
         | --<header files>
   | --lib
      | --lib<soname>.so -> lib<soname>.so.major -> lib<soname>.so.major.minor.patch
         (public libraries linked with application)
      | --<component> (component specific private library, executable data)
      | --<cmake>
         | --components
            | --<component>.config.cmake
   | --share
      | --html/<component>/*.html
      | --info/<component>/*.[pdf, md, txt]
      | --man
      | --doc
         | --<component>
            | --<licenses>
      | --<component>
         | --<misc files> (arch independent non-executable)
         | --samples
```

> Note
>
> ROCm will not support backward compatibility with the v5.1(old) file system hierarchy in its next major release.

For more information, refer to https://refspecs.linuxfoundation.org/fhs.shtml.

8.13.1.2.2 Backward Compatibility with Older Filesystems

ROCm has moved header files and libraries to its new location as indicated in the above structure and included symbolic-link and wrapper header files in its old location for backward compatibility.

> Note
>
> ROCm will continue supporting backward compatibility until the next major release.

8.13.1.2.3 Wrapper header files

Wrapper header files are placed in the old location (/opt/rocm-xxx/<component>/include) with a warning message to include files from the new location (/opt/rocm-xxx/include) as shown in the example below:

```
// Code snippet from hip_runtime.h
#pragma message “This file is deprecated. Use file from include path /opt/rocm-ver/include/ and prefix with hip”.
#include ”hip/hip_runtime.h”
```

The wrapper header files' backward compatibility deprecation is as follows:

- #pragma message announcing deprecation – ROCm v5.2 release
- #pragma message changed to #warning – Future release
- #warning changed to #error – Future release
- Backward compatibility wrappers removed – Future release

8.13.1.2.4 Library files

Library files are available in the /opt/rocm-xxx/lib folder. For backward compatibility, the old library location (/opt/rocm-xxx/<component>/lib) has a soft link to the library at the new location.

Example:

```
$ ls -l /opt/rocm/hip/lib/
total 4
drwxr-xr-x 4 root root 4096 May 12 10:45 cmake
lrwxrwxrwx 1 root root   24 May 10 23:32 libamdhip64.so -> ../../lib/libamdhip64.so
```

8.13.1.2.5 CMake Config files

All CMake configuration files are available in the /opt/rocm-xxx/lib/cmake/<component> folder. For backward compatibility, the old CMake locations (/opt/rocm-xxx/<component>/lib/cmake) consist of a soft link to the new CMake config.

Example:

```
$ ls -l /opt/rocm/hip/lib/cmake/hip/
total 0
lrwxrwxrwx 1 root root 42 May 10 23:32 hip-config.cmake -> ../../../../lib/cmake/hip/hip-config.cmake
```

8.13.2 Fixed Defects

The following defects are fixed in this release.

These defects were identified and documented as known issues in previous ROCm releases and are fixed in the ROCm v5.3 release.

8.13.2.1 Kernel produces incorrect results with ROCm 5.2

User code did not initialize certain data constructs, leading to a correctness issue. A strict reading of the C++ standard suggests that failing to initialize these data constructs is undefined behavior. However, a special case was added for a specific compiler builtin to handle the uninitialized data in a defined manner.

The compiler fix consists of the following patches:

- A new noundef attribute is added. This attribute denotes when a function call argument or return val may never contain uninitialized bits. For more information, see https://reviews.llvm.org/D81678
- The application of this attribute was refined such that it was not added to a specific compiler builtin where the compiler knows that inactive lanes do not impact program execution.

For more information, see https://github.com/RadeonOpenCompute/llvm-project/commit/accf36c58409268ca1f216cdf5ad812ba97ceccd.

8.13.3 Known Issues

This section consists of known issues in this release.

8.13.3.1 Issue with OpenMP-Extras Package Upgrade

The openmp-extras package has been split into runtime (openmp-extras-runtime) and dev (openmp-extras-devel) packages. This change has broken the upgrade support for the openmp-extras package in RHEL/SLES. An available workaround in RHEL is to use the following command for upgrades:

```
sudo yum upgrade rocm-language-runtime --allowerasing
```

An available workaround in SLES is to use the following command for upgrades:

```
zypper update --force-resolution <meta-package>
```

8.13.3.2 AMD Instinct™ MI200 SRIOV Virtualization Issue

There is a known issue in this ROCm v5.3 release with all AMD Instinct™ MI200 devices running within a virtual function (VF) under SRIOV virtualization. This issue will likely impact the functionality of SRIOV-based workloads, but does not impact Discrete Device Assignment (DDA) or Bare Metal.

Until a fix is provided, users should rely on ROCm v5.2.3 to support their SRIOV workloads.

8.13.3.3 System Crash when IMMOU is Enabled

If IOMMU is enabled in SBIOS and ROCm is installed, the system may report the following failure or errors when running workloads such as bandwidth test, clinfo, and HelloWord.cl and cause a system crash.

- IO PAGE FAULT
- IRQ remapping does not support X2APIC mode
- NMI error

Workaround: To avoid the system crash, add amd_iommu=on iommu=pt as the kernel bootparam, as indicated in the warning message.

8.13.4 Library Changes in ROCM 5.3.0

Library	Version
hipBLAS	0.51.0 0.52.0
hipCUB	2.11.1 2.12.0
hipFFT	1.0.8 1.0.9
hipSOLVER	1.4.0 1.5.0
hipSPARSE	2.2.0 2.3.1
rccl	2.12.10
rocALUTION	2.0.3 2.1.0
rocBLAS	2.44.0 2.45.0
rocFFT	1.0.17 1.0.18
rocm-cmake	0.8.0
rocPRIM	2.10.14 2.11.0
rocRAND	2.10.14 2.10.15
rocSOLVER	3.18.0 3.19.0
rocSPARSE	2.2.0
rocThrust	2.15.0 2.16.0
rocWMMA	0.7 0.8
Tensile	4.33.0 4.34.0

8.13.4.1 hipBLAS 0.52.0

hipBLAS 0.52.0 for ROCm 5.3.0

8.13.4.1.1 Added

- Added –cudapath option to install.sh to allow user to specify which cuda build they would like to use.
- Added –installcuda option to install.sh to install cuda via a package manager. Can be used with new –installcudaversion option to specify which version of cuda to install.

8.13.4.1.2 Fixed

- Fixed #includes to support a compiler version.
- Fixed client dependency support in install.sh

8.13.4.2 hipCUB 2.12.0

hipCUB 2.12.0 for ROCm 5.3.0

8.13.4.2.1 Added

- UniqueByKey device algorithm
- SubtractLeft, SubtractLeftPartialTile, SubtractRight, SubtractRightPartialTile overloads in BlockAdjacentDifference.
  - The old overloads (FlagHeads, FlagTails, FlagHeadsAndTails) are deprecated.
- DeviceAdjacentDifference algorithm.
- Extended benchmark suite of DeviceHistogram, DeviceScan, DevicePartition, DeviceReduce, DeviceSegmentedReduce, DeviceSegmentedRadixSort, DeviceRadixSort, DeviceSpmv, DeviceMergeSort, DeviceSegmentedSort

8.13.4.2.2 Changed

- Obsolated type traits defined in util_type.hpp. Use the standard library equivalents instead.
- CUB backend references CUB and thrust version 1.16.0.
- DeviceRadixSort's num_items parameter's type is now templated instead of being an int.
  - If an integral type with a size at most 4 bytes is passed (i.e. an int), the former logic applies.
  - Otherwise the algorithm uses a larger indexing type that makes it possible to sort input data over 2**32 elements.
- Improved build parallelism of the test suite by splitting up large compilation units

8.13.4.3 hipFFT 1.0.9

hipFFT 1.0.9 for ROCm 5.3.0

8.13.4.3.1 Changed

- Clean up build warnings.
- GNUInstall Dir enhancements.
- Requires gtest 1.11.

8.13.4.4 hipSOLVER 1.5.0

hipSOLVER 1.5.0 for ROCm 5.3.0

8.13.4.4.1 Added

- Added functions
  - syevj
    - hipsolverSsyevj_bufferSize, hipsolverDsyevj_bufferSize, hipsolverCheevj_bufferSize, hipsolverZheevj_bufferSize
    - hipsolverSsyevj, hipsolverDsyevj, hipsolverCheevj, hipsolverZheevj
  - syevjBatched
    - hipsolverSsyevjBatched_bufferSize, hipsolverDsyevjBatched_bufferSize, hipsolverCheevjBatched_bufferSize, hipsolverZheevjBatched_bufferSize
    - hipsolverSsyevjBatched, hipsolverDsyevjBatched, hipsolverCheevjBatched, hipsolverZheevjBatched
  - sygvj
    - hipsolverSsygvj_bufferSize, hipsolverDsygvj_bufferSize, hipsolverChegvj_bufferSize, hipsolverZhegvj_bufferSize
    - hipsolverSsygvj, hipsolverDsygvj, hipsolverChegvj, hipsolverZhegvj
- Added compatibility-only functions
  - syevdx/heevdx
    - hipsolverDnSsyevdx_bufferSize, hipsolverDnDsyevdx_bufferSize, hipsolverDnCheevdx_bufferSize, hipsolverDnZheevdx_bufferSize
    - hipsolverDnSsyevdx, hipsolverDnDsyevdx, hipsolverDnCheevdx, hipsolverDnZheevdx
  - sygvdx/hegvdx
    - hipsolverDnSsygvdx_bufferSize, hipsolverDnDsygvdx_bufferSize, hipsolverDnChegvdx_bufferSize, hipsolverDnZhegvdx_bufferSize
    - hipsolverDnSsygvdx, hipsolverDnDsygvdx, hipsolverDnChegvdx, hipsolverDnZhegvdx
- Added –mem_query option to hipsolver-bench, which will print the amount of device memory workspace required by the function.

8.13.4.4.2 Changed

- The rocSOLVER backend will now set info to zero if rocSOLVER does not reference info. (Applies to orgbr/ungbr, orgqr/ungqr, orgtr/ungtr, ormqr/unmqr, ormtr/unmtr, gebrd, geqrf, getrs, potrs, and sytrd/hetrd).
- gesvdj will no longer require extra workspace to transpose V when jobz is HIPSOLVER_EIG_MODE_VECTOR and econ is 1.

8.13.4.4.3 Fixed

- Fixed Fortran return value declarations within hipsolver_module.f90
- Fixed gesvdj_bufferSize returning HIPSOLVER_STATUS_INVALID_VALUE when jobz is HIPSOLVER_EIG_MODE_NOVECTOR and 1 <= ldv < n
- Fixed gesvdj returning HIPSOLVER_STATUS_INVALID_VALUE when jobz is HIPSOLVER_EIG_MODE_VECTOR, econ is 1, and m < n

8.13.4.5 hipSPARSE 2.3.1

hipSPARSE 2.3.1 for ROCm 5.3.0

8.13.4.5.1 Added

- Add SpMM and SpMM batched for CSC format

8.13.4.6 rocALUTION 2.1.0

rocALUTION 2.1.0 for ROCm 5.3.0

8.13.4.6.1 Added

- Benchmarking tool
- Ext+I Interpolation with sparsify strategies added for RS-AMG

8.13.4.6.2 Improved

- ParallelManager

8.13.4.7 rocBLAS 2.45.0

rocBLAS 2.45.0 for ROCm 5.3.0

8.13.4.7.1 Added

- install.sh option –upgrade_tensile_venv_pip to upgrade Pip in Tensile Virtual Environment. The corresponding CMake option is TENSILE_VENV_UPGRADE_PIP.
- install.sh option –relocatable or -r adds rpath and removes ldconf entry on rocBLAS build.
- install.sh option –lazy-library-loading to enable on-demand loading of tensile library files at runtime to speedup rocBLAS initialization.
- Support for RHEL9 and CS9.
- Added Numerical checking routine for symmetric, Hermitian, and triangular matrices, so that they could be checked for any numerical abnormalities such as NaN, Zero, infinity and denormal value.

8.13.4.7.2 Optimizations

- trmm_outofplace performance improvements for all sizes and data types using block-recursive algorithm.
- herkx performance improvements for all sizes and data types using block-recursive algorithm.
- syrk/herk performance improvements by utilising optimised syrkx/herkx code.
- symm/hemm performance improvements for all sizes and datatypes using block-recursive algorithm.

8.13.4.7.3 Changed

- Unifying library logic file names: affects HBH (->HHS_BH), BBH (->BBS_BH), 4xi8BH (->4xi8II_BH). All HPA types are using the new naming convention now.
- Level 3 function argument checking when the handle is set to rocblas_pointer_mode_host now returns the status of rocblas_status_invalid_pointer only for pointers that must be dereferenced based on the alpha and beta argument values. With handle mode rocblas_pointer_mode_device only pointers that are always dereferenced regardless of alpha and beta values are checked and so may lead to a return status of rocblas_status_invalid_pointer. This improves consistency with legacy BLAS behaviour.
- Level 1, 2, and 3 function argument checking for enums is now more rigorously matching legacy BLAS so returns rocblas_status_invalid_value if arguments do not match the accepted subset.
- Add quick-return for internal trmm and gemm template functions.
- Moved function block sizes to a shared header file.
- Level 1, 2, and 3 functions use rocblas_stride datatype for offset.
- Modified the matrix and vector memory allocation in our test infrastructure for all Level 1, 2, 3 and BLAS_EX functions.
- Added specific initialization for symmetric, Hermitian, and triangular matrix types in our test infrastructure.
- Added NaN tests to the test infrastructure for the rest of Level 3, BLAS_EX functions.

8.13.4.7.4 Fixed

- Improved logic to #include <filesystem> vs <experimental/filesystem>.
- install.sh -s option to build rocblas as a static library.
- dot function now sets the device results asynchronously for N <= 0

8.13.4.7.5 Deprecated

- is_complex helper is now deprecated. Use rocblas_is_complex instead.
- The enum truncate_t and the value truncate is now deprecated and will removed from the ROCm release 6.0. It is replaced by rocblas_truncate_t and rocblas_truncate, respectively. The new enum rocblas_truncate_t and the value rocblas_truncate could be used from this ROCm release for an easy transition.

8.13.4.7.6 Removed

- install.sh options –hip-clang , –no-hip-clang, –merge-files, –no-merge-files are removed.

8.13.4.8 rocFFT 1.0.18

rocFFT 1.0.18 for ROCm 5.3.0

8.13.4.8.1 Changed

- Runtime compilation cache now looks for environment variables XDG_CACHE_HOME (on Linux) and LOCALAPPDATA (on Windows) before falling back to HOME.

8.13.4.8.2 Optimizations

- Optimized 2D R2C/C2R to use 2-kernel plans where possible.
- Improved performance of the Bluestein algorithm.
- Optimized sbcc-168 and 100 by using half-lds.

8.13.4.8.3 Fixed

- Fixed occasional failures to parallelize runtime compilation of kernels. Failures would be retried serially and ultimately succeed, but this would take extra time.
- Fixed failures of some R2C 3D transforms that use the unsupported TILE_UNALGNED SBRC kernels. An example is 98^3 R2C out-of-place.
- Fixed bugs in SBRC_ERC type.

8.13.4.9 rocm-cmake 0.8.0

rocm-cmake 0.8.0 for ROCm 5.3.0

8.13.4.9.1 Fixed

- Fixed error in prerm scripts created by rocm_create_package that could break uninstall for packages using the PTH option.

8.13.4.9.2 Changed

- ROCM_USE_DEV_COMPONENT set to on by default for all platforms. This means that Windows will now generate runtime and devel packages by default
- ROCMInstallTargets now defaults CMAKE_INSTALL_LIBDIR to lib if not otherwise specified.
- Changed default Debian compression type to xz and enabled multi-threaded package compression.
- rocm_create_package will no longer warn upon failure to determine version of program rpmbuild.

8.13.4.10 rocPRIM 2.11.0

rocPRIM 2.11.0 for ROCm 5.3.0

8.13.4.10.1 Added

- New functions subtract_left and subtract_right in block_adjacent_difference to apply functions on pairs of adjacent items distributed between threads in a block.
- New device level adjacent_difference primitives.
- Added experimental tooling for automatic kernel configuration tuning for various architectures
- Benchmarks collect and output more detailed system information
- CMake functionality to improve build parallelism of the test suite that splits compilation units by function or by parameters.
- Reverse iterator.

8.13.4.11 rocRAND 2.10.15

rocRAND 2.10.15 for ROCm 5.3.0

8.13.4.11.1 Changed

- Increased number of warmup iterations for rocrand_benchmark_generate from 5 to 15 to eliminate corner cases that would generate artificially high benchmark scores.

8.13.4.12 rocSOLVER 3.19.0

rocSOLVER 3.19.0 for ROCm 5.3.0

8.13.4.12.1 Added

- Partial eigensolver routines for symmetric/hermitian matrices:
  - SYEVX (with batched and strided_batched versions)
  - HEEVX (with batched and strided_batched versions)
- Generalized symmetric- and hermitian-definite partial eigensolvers:
  - SYGVX (with batched and strided_batched versions)
  - HEGVX (with batched and strided_batched versions)
- Eigensolver routines for symmetric/hermitian matrices using Jacobi algorithm:
  - SYEVJ (with batched and strided_batched versions)
  - HEEVJ (with batched and strided_batched versions)
- Generalized symmetric- and hermitian-definite eigensolvers using Jacobi algorithm:
  - SYGVJ (with batched and strided_batched versions)
  - HEGVJ (with batched and strided_batched versions)
- Added –profile_kernels option to rocsolver-bench, which will include kernel calls in the profile log (if profile logging is enabled with –profile).

8.13.4.12.2 Changed

- Changed rocsolver-bench result labels cpu_time and gpu_time to cpu_time_us and gpu_time_us, respectively.

8.13.4.12.3 Removed

- Removed dependency on cblas from the rocsolver test and benchmark clients.

8.13.4.12.4 Fixed

- Fixed incorrect SYGS2/HEGS2, SYGST/HEGST, SYGV/HEGV, and SYGVD/HEGVD results for batch counts larger than 32.
- Fixed STEIN memory access fault when nev is 0.
- Fixed incorrect STEBZ results for close eigenvalues when range = index.
- Fixed git unsafe repository error when building with ./install.sh -cd as a non-root user.

8.13.4.13 rocThrust 2.16.0

rocThrust 2.16.0 for ROCm 5.3.0

8.13.4.13.1 Changed

- rocThrust functionality dependent on device malloc works is functional as ROCm 5.2 reneabled device malloc. Device launched thrust::sort and thrust::sort_by_key are available for use.

8.13.4.14 rocWMMA 0.8

rocWMMA 0.8 for ROCm 5.3.0

8.13.4.15 Tensile 4.34.0

Tensile 4.34.0 for ROCm 5.3.0

8.13.4.15.1 Added

- Lazy loading of solution libraries and code object files
- Support for dictionary style logic files
- Support for decision tree based logic files using dictionary format
- DecisionTreeLibrary for solution selection
- DirectToLDS support for HGEMM
- DirectToVgpr support for SGEMM
- Grid based distance metric for solution selection
- Support for gfx11xx
- Support for DirectToVgprA/B + TLU=False
- ForkParameters Groups as a way of specifying solution parameters
- Support for a new Tensile yaml config format
- TensileClientConfig for generating Tensile client config files
- Options for TensileCreateLibrary to build client and create client config file

8.13.4.15.2 Optimizations

- Solution generation is now cached and is not repeated if solution parameters are unchanged

8.13.4.15.3 Changed

- Default MACInstruction to FMA

8.13.4.15.4 Fixed

- Accept StaggerUStride=0 as valid
- Reject invalid data types for UnrollLoopEfficiencyEnable
- Fix invalid code generation issues related to DirectToVgpr
- Return hipErrorNotFound if no modules are loaded
- Fix performance drop for NN ZGEMM with 96x64 macro tile
- Fix memory violation for general batched kernels when alpha/beta/K = 0

8.14 ROCm 5.2.3

8.14.1 Changes in This Release

8.14.1.1 Ubuntu 18.04 End of Life Announcement

Support for Ubuntu 18.04 ends in this release. Future releases of ROCm will not provide prebuilt packages for Ubuntu 18.04. HIP and Other Runtimes

8.14.1.2 HIP Runtime

8.14.1.2.1 Fixes

- A bug was discovered in the HIP graph capture implementation in the ROCm v5.2.0 release. If the same kernel is called twice (with different argument values) in a graph capture, the implementation only kept the argument values for the second kernel call.
- A bug was introduced in the hiprtc implementation in the ROCm v5.2.0 release. This bug caused the hiprtcGetLoweredName call to fail for named expressions with whitespace in it.

Example:

The named expression my_sqrt<complex<double>> passed but my_sqrt<complex<double >> failed. ROCm Libraries

8.14.1.3 RCCL

8.14.1.3.1 Added

Compatibility with NCCL 2.12.10

- Packages for test and benchmark executables on all supported OSes using CPack
- Added custom signal handler - opt-in with RCCL_ENABLE_SIGNALHANDLER=1
  - Additional details provided if Binary File Descriptor library (BFD) is pre-installed.
- Added experimental support for using multiple ranks per device
  - Requires using a new interface to create communicator (ncclCommInitRankMulti), refer to the interface documentation for details.
  - To avoid potential deadlocks, user might have to set an environment variables increasing the number of hardware queues. For example,

```
export GPU_MAX_HW_QUEUES=16
```

- Added support for reusing ports in NET/IB channels
  - Opt-in with NCCL_IB_SOCK_CLIENT_PORT_REUSE=1 and NCCL_IB_SOCK_SERVER_PORT_REUSE=1
  - When "Call to bind failed: Address already in use" error happens in large-scale AlltoAll(for example, >=64 MI200 nodes), users are suggested to opt-in either one or both of the options to resolve the massive port usage issue
  - Avoid using NCCL_IB_SOCK_SERVER_PORT_REUSE when NCCL_NCHANNELS_PER_NET_PEER is tuned >1

8.14.1.3.2 Removed

- Removed experimental clique-based kernels

8.14.1.4 Development Tools

No notable changes in this release for development tools, including the compiler, profiler, and debugger Deployment and Management Tools

No notable changes in this release for deployment and management tools. Older ROCm Releases

For release information for older ROCm releases, refer to https://github.com/RadeonOpenCompute/ROCm/blob/master/CHANGELOG.md

8.14.2 Library Changes in ROCM 5.2.3

Library	Version
hipBLAS	0.51.0
hipCUB	2.11.1
hipFFT	1.0.8
hipSOLVER	1.4.0
hipSPARSE	2.2.0
rccl	2.11.4 2.12.10
rocALUTION	2.0.3
rocBLAS	2.44.0
rocFFT	1.0.17
rocPRIM	2.10.14
rocRAND	2.10.14
rocSOLVER	3.18.0
rocSPARSE	2.2.0
rocThrust	2.15.0
rocWMMA	0.7
Tensile	4.33.0

8.14.2.1 rccl 2.12.10

RCCL 2.12.10 for ROCm 5.2.3

8.14.2.1.1 Added

- Compatibility with NCCL 2.12.10
- Packages for test and benchmark executables on all supported OSes using CPack.
- Adding custom signal handler - opt-in with RCCL_ENABLE_SIGNALHANDLER=1
  - Additional details provided if Binary File Descriptor library (BFD) is pre-installed
- Adding support for reusing ports in NET/IB channels
  - Opt-in with NCCL_IB_SOCK_CLIENT_PORT_REUSE=1 and NCCL_IB_SOCK_SERVER_PORT_REUSE=1
  - When "Call to bind failed : Address already in use" error happens in large-scale AlltoAll (e.g., >=64 MI200 nodes), users are suggested to opt-in either one or both of the options to resolve the massive port usage issue
  - Avoid using NCCL_IB_SOCK_SERVER_PORT_REUSE when NCCL_NCHANNELS_PER_NET_PEER is tuned >1

8.14.2.1.2 Removed

- Removed experimental clique-based kernels

8.15 ROCm 5.2.1

8.15.1 Library Changes in ROCM 5.2.1

Library	Version
hipBLAS	0.51.0
hipCUB	2.11.1
hipFFT	1.0.8
hipSOLVER	1.4.0
hipSPARSE	2.2.0
rccl	2.11.4
rocALUTION	2.0.3
rocBLAS	2.44.0
rocFFT	1.0.17
rocPRIM	2.10.14
rocRAND	2.10.14
rocSOLVER	3.18.0
rocSPARSE	2.2.0
rocThrust	2.15.0
rocWMMA	0.7
Tensile	4.33.0

8.16 ROCm 5.2.0

8.16.1 What's New in This Release

8.16.1.1 HIP Enhancements

The ROCm v5.2 release consists of the following HIP enhancements:

8.16.1.1.1 HIP Installation Guide Updates

The HIP Installation Guide is updated to include building HIP tests from source on the AMD and NVIDIA platforms.

For more details, refer to the HIP Installation Guide v5.2.

8.16.1.1.2 Support for device-side malloc on HIP-Clang

HIP-Clang now supports device-side malloc. This implementation does not require the use of hipDeviceSetLimit(hipLimitMallocHeapSize,value) nor respect any setting. The heap is fully dynamic and can grow until the available free memory on the device is consumed.

The test codes at the following link show how to implement applications using malloc and free functions in device kernels:

https://github.com/ROCm-Developer-Tools/HIP/blob/develop/tests/src/deviceLib/hipDeviceMalloc.cpp

8.16.1.1.3 New HIP APIs in This Release

The following new HIP APIs are available in the ROCm v5.2 release. Note that this is a pre-official version (beta) release of the new APIs:

8.16.1.1.3.1 Device management HIP APIs

The new device management HIP APIs are as follows:

- Gets a UUID for the device. This API returns a UUID for the device.

```
hipError_t hipDeviceGetUuid(hipUUID* uuid, hipDevice_t device);
```

> Note
>
> This new API corresponds to the following CUDA API:
>
> ```
> CUresult cuDeviceGetUuid(CUuuid* uuid, CUdevice dev);
> ```

- Gets default memory pool of the specified device

```
hipError_t hipDeviceGetDefaultMemPool(hipMemPool_t* mem_pool, int device);
```

- Sets the current memory pool of a device

```
hipError_t hipDeviceSetMemPool(int device, hipMemPool_t mem_pool);
```

- Gets the current memory pool for the specified device

```
hipError_t hipDeviceGetMemPool(hipMemPool_t* mem_pool, int device);
```

8.16.1.1.3.2 New HIP Runtime APIs in Memory Management

The new Stream Ordered Memory Allocator functions of HIP runtime APIs in memory management are as follows:

- Allocates memory with stream ordered semantics

```
hipError_t hipMallocAsync(void** dev_ptr, size_t size, hipStream_t stream);
```

- Frees memory with stream ordered semantics

```
hipError_t hipFreeAsync(void* dev_ptr, hipStream_t stream);
```

- Releases freed memory back to the OS

```
hipError_t hipMemPoolTrimTo(hipMemPool_t mem_pool, size_t min_bytes_to_hold);
```

- Sets attributes of a memory pool

```
hipError_t hipMemPoolSetAttribute(hipMemPool_t mem_pool, hipMemPoolAttr attr, void* value);
```

- Gets attributes of a memory pool

```
hipError_t hipMemPoolGetAttribute(hipMemPool_t mem_pool, hipMemPoolAttr attr, void* value);
```

- Controls visibility of the specified pool between devices

```
hipError_t hipMemPoolSetAccess(hipMemPool_t mem_pool, const hipMemAccessDesc* desc_list, size_t␣
↪count);
```

- Returns the accessibility of a pool from a device

```
hipError_t hipMemPoolGetAccess(hipMemAccessFlags* flags, hipMemPool_t mem_pool,␣
↪hipMemLocation* location);
```

- Creates a memory pool

```
hipError_t hipMemPoolCreate(hipMemPool_t* mem_pool, const hipMemPoolProps* pool_props);
```

- Destroys the specified memory pool

```
hipError_t hipMemPoolDestroy(hipMemPool_t mem_pool);
```

- Allocates memory from a specified pool with stream ordered semantics

```
hipError_t hipMallocFromPoolAsync(void** dev_ptr, size_t size, hipMemPool_t mem_pool, hipStream_t␣
↪stream);
```

- Exports a memory pool to the requested handle type

```
hipError_t hipMemPoolExportToShareableHandle(
    void*                    shared_handle,
    hipMemPool_t             mem_pool,
    hipMemAllocationHandleType handle_type,
    unsigned int             flags);
```

- Imports a memory pool from a shared handle

```
hipError_t hipMemPoolImportFromShareableHandle(
    hipMemPool_t*            mem_pool,
    void*                    shared_handle,
    hipMemAllocationHandleType handle_type,
    unsigned int             flags);
```

- Exports data to share a memory pool allocation between processes

```
hipError_t hipMemPoolExportPointer(hipMemPoolPtrExportData* export_data, void* dev_ptr);
Import a memory pool allocation from another process.t
hipError_t hipMemPoolImportPointer(
    void**                  dev_ptr,
    hipMemPool_t            mem_pool,
    hipMemPoolPtrExportData* export_data);
```

8.16.1.1.3.3 HIP Graph Management APIs

The new HIP Graph Management APIs are as follows:

- Enqueues a host function call in a stream

```
hipError_t hipLaunchHostFunc(hipStream_t stream, hipHostFn_t fn, void* userData);
```

- Swaps the stream capture mode of a thread

```
hipError_t hipThreadExchangeStreamCaptureMode(hipStreamCaptureMode* mode);
```

- Sets a node attribute

```
hipError_t hipGraphKernelNodeSetAttribute(hipGraphNode_t hNode, hipKernelNodeAttrID attr, const␣
↪hipKernelNodeAttrValue* value);
```

- Gets a node attribute

```
hipError_t hipGraphKernelNodeGetAttribute(hipGraphNode_t hNode, hipKernelNodeAttrID attr,          ␣
↪                                hipKernelNodeAttrValue* value);
```

8.16.1.1.3.4 Support for Virtual Memory Management APIs

The new APIs for virtual memory management are as follows:

- Frees an address range reservation made via hipMemAddressReserve

```
hipError_t hipMemAddressFree(void* devPtr, size_t size);
```

- Reserves an address range

```
hipError_t hipMemAddressReserve(void** ptr, size_t size, size_t alignment, void* addr, unsigned long␣
↪long flags);
```

- Creates a memory allocation described by the properties and size

```
hipError_t hipMemCreate(hipMemGenericAllocationHandle_t* handle, size_t size, const␣
↪hipMemAllocationProp* prop, unsigned long long flags);
```

- Exports an allocation to a requested shareable handle type

```
hipError_t hipMemExportToShareableHandle(void* shareableHandle, hipMemGenericAllocationHandle_t␣
↪handle, hipMemAllocationHandleType handleType, unsigned long long flags);
```

- Gets the access flags set for the given location and ptr

```
hipError_t hipMemGetAccess(unsigned long long* flags, const hipMemLocation* location, void* ptr);
```

- Calculates either the minimal or recommended granularity

```
hipError_t hipMemGetAllocationGranularity(size_t* granularity, const hipMemAllocationProp* prop,␣
↪hipMemAllocationGranularity_flags option);
```

- Retrieves the property structure of the given handle

```
hipError_t hipMemGetAllocationPropertiesFromHandle(hipMemAllocationProp* prop,␣
↪hipMemGenericAllocationHandle_t handle);
```

- Imports an allocation from a requested shareable handle type

```
hipError_t hipMemImportFromShareableHandle(hipMemGenericAllocationHandle_t* handle, void*␣
↪osHandle, hipMemAllocationHandleType shHandleType);
```

- Maps an allocation handle to a reserved virtual address range

```
hipError_t hipMemMap(void* ptr, size_t size, size_t offset, hipMemGenericAllocationHandle_t handle,␣
↪unsigned long long flags);
```

- Maps or unmaps subregions of sparse HIP arrays and sparse HIP mipmapped arrays

```
hipError_t hipMemMapArrayAsync(hipArrayMapInfo* mapInfoList, unsigned int  count, hipStream_t␣
↪stream);
```

- Release a memory handle representing a memory allocation, that was previously allocated through hipMemCreate

```
hipError_t hipMemRelease(hipMemGenericAllocationHandle_t handle);
```

- Returns the allocation handle of the backing memory allocation given the address

```
hipError_t hipMemRetainAllocationHandle(hipMemGenericAllocationHandle_t* handle, void* addr);
```

- Sets the access flags for each location specified in desc for the given virtual address range

```
hipError_t hipMemSetAccess(void* ptr, size_t size, const hipMemAccessDesc* desc, size_t count);
```

- Unmaps memory allocation of a given address range

```
hipError_t hipMemUnmap(void* ptr, size_t size);
```

For more information, refer to the HIP API documentation at Modules.

8.16.1.1.4 Planned HIP Changes in Future Releases

Changes to hipDeviceProp_t, HIPMEMCPY_3D, and hipArray structures (and related HIP APIs) are planned in the next major release. These changes may impact backward compatibility.

Refer to the Release Notes document in subsequent releases for more information. ROCm Math and Communication Libraries

In this release, ROCm Math and Communication Libraries consist of the following enhancements and fixes: New rocWMMA for Matrix Multiplication and Accumulation Operations Acceleration

This release introduces a new ROCm C++ library for accelerating mixed precision matrix multiplication and accumulation (MFMA) operations leveraging specialized GPU matrix cores. rocWMMA provides a C++ API to facilitate breaking down matrix multiply accumulate problems into fragments and using them in block-wise operations that are distributed in parallel across GPU wavefronts. The API is a header library of GPU device code, meaning matrix core acceleration may be compiled directly into your kernel device code. This can benefit from compiler optimization in the generation of kernel assembly and does not incur additional overhead costs of linking to external runtime libraries or having to launch separate kernels.

rocWMMA is released as a header library and includes test and sample projects to validate and illustrate example usages of the C++ API. GEMM matrix multiplication is used as primary validation given the heavy precedent for the library. However, the usage portfolio is growing significantly and demonstrates different ways rocWMMA may be consumed.

For more information, refer to Communication Libraries.

8.16.1.2 OpenMP Enhancements in This Release

8.16.1.2.1 OMPT Target Support

The OpenMP runtime in ROCm implements a subset of the OMPT device APIs, as described in the OpenMP specification document. These are APIs that allow first-party tools to examine the profile and traces for kernels that execute on a device. A tool may register callbacks for data transfer and kernel dispatch entry points. A tool may use APIs to start and stop tracing for device-related activities such as data transfer and kernel dispatch timings and associated metadata. If device tracing is enabled, trace records for device activities are collected during program execution and returned to the tool using the APIs described in the specification.

Following is an example demonstrating how a tool would use the OMPT target APIs supported. The README in /opt/rocm/llvm/examples/tools/ompt outlines the steps to follow, and you can run the provided example as indicated below:

```
cd /opt/rocm/llvm/examples/tools/ompt/veccopy-ompt-target-tracing
make run
```

The file veccopy-ompt-target-tracing.c simulates how a tool would initiate device activity tracing. The file callbacks.h shows the callbacks that may be registered and implemented by the tool.

8.16.2 Deprecations and Warnings

8.16.2.1 Linux Filesystem Hierarchy Standard for ROCm

ROCm packages have adopted the Linux foundation filesystem hierarchy standard in this release to ensure ROCm components follow open source conventions for Linux-based distributions. While moving to a new filesystem hierarchy, ROCm ensures backward compatibility with its 5.1 version or older filesystem hierarchy. See below for a detailed explanation of the new filesystem hierarchy and backward compatibility.

8.16.2.1.1 New Filesystem Hierarchy

The following is the new filesystem hierarchy:

```
/opt/rocm-<ver>
   | --bin
     | --All externally exposed Binaries
   | --libexec
      | --<component>
         | -- Component specific private non-ISA executables (architecture independent)
   | --include
      | -- <component>
         | --<header files>
   | --lib
      | --lib<soname>.so -> lib<soname>.so.major -> lib<soname>.so.major.minor.patch
         (public libraries linked with application)
      | --<component> (component specific private library, executable data)
      | --<cmake>
         | --components
            | --<component>.config.cmake
   | --share
      | --html/<component>/*.html
      | --info/<component>/*.[pdf, md, txt]
      | --man
      | --doc
         | --<component>
            | --<licenses>
      | --<component>
         | --<misc files> (arch independent non-executable)
         | --samples
```

> Note
>
> ROCm will not support backward compatibility with the v5.1(old) file system hierarchy in its next major release.

For more information, refer to https://refspecs.linuxfoundation.org/fhs.shtml.

8.16.2.1.2 Backward Compatibility with Older Filesystems

ROCm has moved header files and libraries to its new location as indicated in the above structure and included symbolic-link and wrapper header files in its old location for backward compatibility.

> Note
>
> ROCm will continue supporting backward compatibility until the next major release.

8.16.2.1.3 Wrapper header files

Wrapper header files are placed in the old location (/opt/rocm-xxx/<component>/include) with a warning message to include files from the new location (/opt/rocm-xxx/include) as shown in the example below:

```
// Code snippet from hip_runtime.h
#pragma message “This file is deprecated. Use file from include path /opt/rocm-ver/include/ and prefix with hip”.
#include ”hip/hip_runtime.h”
```

The wrapper header files' backward compatibility deprecation is as follows:

- #pragma message announcing deprecation – ROCm v5.2 release
- #pragma message changed to #warning – Future release
- #warning changed to #error – Future release
- Backward compatibility wrappers removed – Future release

8.16.2.1.4 Library files

Library files are available in the /opt/rocm-xxx/lib folder. For backward compatibility, the old library location (/opt/rocm-xxx/<component>/lib) has a soft link to the library at the new location.

Example:

```
$ ls -l /opt/rocm/hip/lib/
total 4
drwxr-xr-x 4 root root 4096 May 12 10:45 cmake
lrwxrwxrwx 1 root root   24 May 10 23:32 libamdhip64.so -> ../../lib/libamdhip64.so
```

8.16.2.1.5 CMake Config files

All CMake configuration files are available in the /opt/rocm-xxx/lib/cmake/<component> folder. For backward compatibility, the old CMake locations (/opt/rocm-xxx/<component>/lib/cmake) consist of a soft link to the new CMake config.

Example:

```
$ ls -l /opt/rocm/hip/lib/cmake/hip/
total 0
lrwxrwxrwx 1 root root 42 May 10 23:32 hip-config.cmake -> ../../../../lib/cmake/hip/hip-config.cmake
```

8.16.2.2 Planned deprecation of hip-rocclr and hip-base packages

In the ROCm v5.2 release, hip-rocclr and hip-base packages (Debian and RPM) are planned for deprecation and will be removed in a future release. hip-runtime-amd and hip-dev(el) will replace these packages respectively. Users of hip-rocclr must install two packages, hip-runtime-amd and hip-dev, to get the same set of packages installed by hip-rocclr previously.

Currently, both package names hip-rocclr (or) hip-runtime-amd and hip-base (or) hip-dev(el) are supported. Deprecation of Integrated HIP Directed Tests

The integrated HIP directed tests, which are currently built by default, are deprecated in this release. The default building and execution support through CMake will be removed in future release.

8.16.3 Fixed Defects

Fixed Defect	Fix
ROCmInfo does not list gpus	Code fix
Hang observed while restoring cooperative group samples	Code fix
ROCM-SMI over SRIOV: Unsupported commands do not return proper error message	Code fix

8.16.4 Known Issues

This section consists of known issues in this release.

8.16.4.1 Compiler Error on gfx1030 When Compiling at -O0

8.16.4.1.1 Issue

A compiler error occurs when using -O0 flag to compile code for gfx1030 that calls atomicAddNoRet, which is defined in amd_hip_atomic.h. The compiler generates an illegal instruction for gfx1030.

8.16.4.1.2 Workaround

The workaround is not to use the -O0 flag for this case. For higher optimization levels, the compiler does not generate an invalid instruction.

8.16.4.2 System Freeze Observed During CUDA Memtest Checkpoint

8.16.4.2.1 Issue

Checkpoint/Restore in Userspace (CRIU) requires 20 MB of VRAM approximately to checkpoint and restore. The CRIU process may freeze if the maximum amount of available VRAM is allocated to checkpoint applications.

8.16.4.2.2 Workaround

To use CRIU to checkpoint and restore your application, limit the amount of VRAM the application uses to ensure at least 20 MB is available.

8.16.4.3 HPC test fails with the “HSA_STATUS_ERROR_MEMORY_FAULT” error

8.16.4.3.1 Issue

The compiler may incorrectly compile a program that uses the ___shfl_sync(mask, value, srcLane) function when the “value” parameter to the function is undefined along some path to the function. For most functions, uninitialized inputs cause undefined behavior, but the definition for ___shfl_sync should allow for undefined values.

8.16.4.3.2 Workaround

The workaround is to initialize the parameters to ___shfl_sync.

> Note
>
> When the -Wall compilation flag is used, the compiler generates a warning indicating the variable is initialized along some path.

Example:

```
double res = 0.0; // Initialize the input to ___shfl_sync.
if (lane == 0) {
  res = <some expression>
}
res = ___shfl_sync(mask, res, 0);
```

8.16.4.4 Kernel produces incorrect result

8.16.4.4.1 Issue

In recent changes to Clang, insertion of the noundef attribute to all the function arguments has been enabled by default.

In the HIP kernel, variable var in shfl_sync may not be initialized, so LLVM IR treats it as undef.

So, the function argument that is potentially undef (because it is not intialized) has always been assumed to be noundef by LLVM IR (since Clang has inserted noundef attribute). This leads to ambiguous kernel execution.

8.16.4.4.2 Workaround

- Skip adding noundef attribute to functions tagged with convergent attribute. Refer to https://reviews.llvm.org/D124158 for more information.
- Introduce shuffle attribute and add it to ___shfl like APIs at hip headers. Clang can skip adding noundef attribute, if it finds that argument is tagged with shuffle attribute. Refer to https://reviews.llvm.org/D125378 for more information.
- Introduce clang builtin for ___shfl to identify it and skip adding noundef attribute.
- Introduce ___builtin_freeze to use on the relevant arguments in library wrappers. The library/header need to insert freezes on the relevant inputs.

8.16.4.5 Issue with Applications Triggering Oversubscription

There is a known issue with applications that trigger oversubscription. A hardware hang occurs when ROCgdb is used on AMD Instinct™ MI50 and MI100 systems.

This issue is under investigation and will be fixed in a future release.

8.16.5 Library Changes in ROCM 5.2.0

Library	Version
hipBLAS	0.50.0 0.51.0
hipCUB	2.11.0 2.11.1
hipFFT	1.0.7 1.0.8
hipSOLVER	1.3.0 1.4.0
hipSPARSE	2.1.0 2.2.0
rccl	2.11.4
rocALUTION	2.0.2 2.0.3
rocBLAS	2.43.0 2.44.0
rocFFT	1.0.16 1.0.17
rocPRIM	2.10.13 2.10.14
rocRAND	2.10.13 2.10.14
rocSOLVER	3.17.0 3.18.0
rocSPARSE	2.1.0 2.2.0
rocThrust	2.14.0 2.15.0
rocWMMA	0.7
Tensile	4.32.0 4.33.0

8.16.5.1 hipBLAS 0.51.0

hipBLAS 0.51.0 for ROCm 5.2.0

8.16.5.1.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.
- Added File/Folder Reorg Changes with backward compatibility support enabled using ROCM-CMAKE wrapper functions
- Added user-specified initialization option to hipblas-bench

8.16.5.1.2 Fixed

- Fixed version gathering in performance measuring script

8.16.5.2 hipCUB 2.11.1

hipCUB 2.11.1 for ROCm 5.2.0

8.16.5.2.1 Added

- Packages for tests and benchmark executable on all supported OSes using CPack.

8.16.5.3 hipFFT 1.0.8

hipFFT 1.0.8 for ROCm 5.2.0

8.16.5.3.1 Added

- Added File/Folder Reorg Changes with backward compatibility support using ROCM-CMAKE wrapper functions.
- Packages for test and benchmark executables on all supported OSes using CPack.

8.16.5.4 hipSOLVER 1.4.0

hipSOLVER 1.4.0 for ROCm 5.2.0

8.16.5.4.1 Added

- Package generation for test and benchmark executables on all supported OSes using CPack.
- File/Folder Reorg
  - Added File/Folder Reorg Changes with backward compatibility support using ROCM-CMAKE wrapper functions.

8.16.5.4.2 Fixed

- Fixed the ReadTheDocs documentation generation.

8.16.5.5 hipSPARSE 2.2.0

hipSPARSE 2.2.0 for ROCm 5.2.0

8.16.5.5.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.

8.16.5.6 rocALUTION 2.0.3

rocALUTION 2.0.3 for ROCm 5.2.0

8.16.5.6.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.

8.16.5.7 rocBLAS 2.44.0

rocBLAS 2.44.0 for ROCm 5.2.0

8.16.5.7.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.
- Added Denormal number detection to the Numerical checking helper function to detect denormal/subnormal numbers in the input and the output vectors of rocBLAS level 1 and 2 functions.
- Added Denormal number detection to the Numerical checking helper function to detect denormal/subnormal numbers in the input and the output general matrices of rocBLAS level 2 and 3 functions.
- Added NaN initialization tests to the yaml files of Level 2 rocBLAS batched and strided-batched functions for testing purposes.
- Added memory allocation check to avoid disk swapping during rocblas-test runs by skipping tests.

8.16.5.7.2 Optimizations

- Improved performance of non-batched and batched her2 for all sizes and data types.
- Improved performance of non-batched and batched amin for all data types using shuffle reductions.
- Improved performance of non-batched and batched amax for all data types using shuffle reductions.
- Improved performance of trsv for all sizes and data types.

8.16.5.7.3 Changed

- Modifying gemm_ex for HBH (High-precision F16). The alpha/beta data type remains as F32 without narrowing to F16 and expanding back to F32 in the kernel. This change prevents rounding errors due to alpha/beta conversion in situations where alpha/beta are not exactly represented as an F16.
- Modified non-batched and batched asum, nrm2 functions to use shuffle instruction based reductions.
- For gemm, gemm_ex, gemm_ex2 internal API use rocblas_stride datatype for offset.
- For symm, hemm, syrk, herk, dgmm, geam internal API use rocblas_stride datatype for offset.
- AMD copyright year for all rocBLAS files.
- For gemv (transpose-case), typecasted the 'lda'(offset) datatype to size_t during offset calculation to avoid overflow and remove duplicate template functions.

8.16.5.7.4 Fixed

- For function her2 avoid overflow in offset calculation.
- For trsm when alpha == 0 and on host, allow A to be nullptr.
- Fixed memory access issue in trsv.
- Fixed git pre-commit script to update only AMD copyright year.
- Fixed dgmm, geam test functions to set correct stride values.
- For functions ssyr2k and dsyr2k allow trans == rocblas_operation_conjugate_transpose.
- Fixed compilation error for clients-only build.

8.16.5.7.5 Removed

- Remove Navi12 (gfx1011) from fat binary.

8.16.5.8 rocFFT 1.0.17

rocFFT 1.0.17 for ROCm 5.2.0

8.16.5.8.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.
- Added File/Folder Reorg Changes with backward compatibility support using ROCM-CMAKE wrapper functions.

8.16.5.8.2 Changed

- Improved reuse of twiddle memory between plans.
- Set a default load/store callback when only one callback type is set via the API for improved performance.

8.16.5.8.3 Optimizations

- Introduced a new access pattern of lds (non-linear) and applied it on sbcc kernels len 64 to get performance improvement.

8.16.5.8.4 Fixed

- Fixed plan creation failure in cases where SBCC kernels would need to write to non-unit-stride buffers.

8.16.5.9 rocPRIM 2.10.14

rocPRIM 2.10.14 for ROCm 5.2.0

8.16.5.9.1 Added

- Packages for tests and benchmark executable on all supported OSes using CPack.
- Added File/Folder Reorg Changes and Enabled Backward compatibility support using wrapper headers.

8.16.5.10 rocRAND 2.10.14

rocRAND 2.10.14 for ROCm 5.2.0

8.16.5.10.1 Added

- Backward compatibility for deprecated #include <rocrand.h> using wrapper header files.
- Packages for test and benchmark executables on all supported OSes using CPack.

8.16.5.11 rocSOLVER 3.18.0

rocSOLVER 3.18.0 for ROCm 5.2.0

8.16.5.11.1 Added

- Partial eigenvalue decomposition routines:
  - STEBZ
  - STEIN
- Package generation for test and benchmark executables on all supported OSes using CPack.
- Added tests for multi-level logging
- Added tests for rocsolver-bench client
- File/Folder Reorg
  - Added File/Folder Reorg Changes with backward compatibility support using ROCM-CMAKE wrapper functions.

8.16.5.11.2 Fixed

- Fixed compatibility with libfmt 8.1

8.16.5.12 rocSPARSE 2.2.0

rocSPARSE 2.2.0 for ROCm 5.2.0

8.16.5.12.1 Added

- batched SpMM for CSR, COO and Blocked ELL formats.
- Packages for test and benchmark executables on all supported OSes using CPack.
- Clients file importers and exporters.

8.16.5.12.2 Improved

- Clients code size reduction.
- Clients error handling.
- Clients benchmarking for performance tracking.

8.16.5.12.3 Changed

- Test adjustments due to roundoff errors.
- Fixing API calls compatiblity with rocPRIM.

8.16.5.12.4 Known Issues

- none

8.16.5.13 rocThrust 2.15.0

rocThrust 2.15.0 for ROCm 5.2.0

8.16.5.13.1 Added

- Packages for tests and benchmark executable on all supported OSes using CPack.

8.16.5.14 rocWMMA 0.7

rocWMMA 0.7 for ROCm 5.2.0

8.16.5.14.1 Added

- Added unit tests for DLRM kernels
- Added GEMM sample
- Added DLRM sample
- Added SGEMV sample
- Added unit tests for cooperative wmma load and stores
- Added unit tests for IOBarrier.h
- Added wmma load/ store tests for different matrix types (A, B and Accumulator)
- Added more block sizes 1, 2, 4, 8 to test MmaSyncMultiTest
- Added block sizes 4, 8 to test MmaSynMultiLdsTest
- Added support for wmma load / store layouts with block dimension greater than 64
- Added IOShape structure to define the attributes of mapping and layouts for all wmma matrix types
- Added CI testing for rocWMMA

8.16.5.14.2 Changed

- Renamed wmma to rocwmma in cmake, header files and documentation
- Renamed library files
- Modified Layout.h to use different matrix offset calculations (base offset, incremental offset and cumulative offset)
- Opaque load/store continue to use incrementatl offsets as they fill the entire block
- Cooperative load/store use cumulative offsets as they fill only small portions for the entire block
- Increased Max split counts to 64 for cooperative load/store
- Moved all the wmma definitions, API headers to rocwmma namespace
- Modified wmma fill unit tests to validate all matrix types (A, B, Accumulator)

8.16.5.15 Tensile 4.33.0

Tensile 4.33.0 for ROCm 5.2.0

8.16.5.15.1 Added

- TensileUpdateLibrary for updating old library logic files
- Support for TensileRetuneLibrary to use sizes from separate file
- ZGEMM DirectToVgpr/DirectToLds/StoreCInUnroll/MIArchVgpr support
- Tests for denorm correctness
- Option to write different architectures to different TensileLibrary files

8.16.5.15.2 Optimizations

- Optimize MessagePackLoadLibraryFile by switching to fread
- DGEMM tail loop optimization for PrefetchAcrossPersistentMode=1/DirectToVgpr

8.16.5.15.3 Changed

- Alpha/beta datatype remains as F32 for HPA HGEMM
- Force assembly kernels to not flush denorms
- Use hipDeviceAttributePhysicalMultiProcessorCount as multiProcessorCount

8.16.5.15.4 Fixed

- Fix segmentation fault when run i8 datatype with TENSILE_DB=0x80

8.17 ROCm 5.1.3

8.17.1 Library Changes in ROCM 5.1.3

Library	Version
hipBLAS	0.50.0
hipCUB	2.11.0
hipFFT	1.0.7
hipSOLVER	1.3.0
hipSPARSE	2.1.0
rccl	2.11.4
rocALUTION	2.0.2
rocBLAS	2.43.0
rocFFT	1.0.16
rocPRIM	2.10.13
rocRAND	2.10.13
rocSOLVER	3.17.0
rocSPARSE	2.1.0
rocThrust	2.14.0
Tensile	4.32.0

8.18 ROCm 5.1.1

8.18.1 Library Changes in ROCM 5.1.1

Library	Version
hipBLAS	0.50.0
hipCUB	2.11.0
hipFFT	1.0.7
hipSOLVER	1.3.0
hipSPARSE	2.1.0
rccl	2.11.4
rocALUTION	2.0.2
rocBLAS	2.43.0
rocFFT	1.0.16
rocPRIM	2.10.13
rocRAND	2.10.13
rocSOLVER	3.17.0
rocSPARSE	2.1.0
rocThrust	2.14.0
Tensile	4.32.0

8.19 ROCm 5.1.0

8.19.1 What's New in This Release

8.19.1.1 HIP Enhancements

The ROCm v5.1 release consists of the following HIP enhancements.

8.19.1.1.1 HIP Installation Guide Updates

The HIP Installation Guide is updated to include installation and building HIP from source on the AMD and NVIDIA platforms.

Refer to the HIP Installation Guide v5.1 for more details.

8.19.1.1.2 Support for HIP Graph

ROCm v5.1 extends support for HIP Graph.

8.19.1.1.3 Planned Changes for HIP in Future Releases

8.19.1.1.3.1 Separation of hiprtc (libhiprtc) library from hip runtime (amdhip64)

On ROCm/Linux, to maintain backward compatibility, the hipruntime library (amdhip64) will continue to include hiprtc symbols in future releases. The backward compatible support may be discontinued by removing hiprtc symbols from the hipruntime library (amdhip64) in the next major release.

8.19.1.1.3.2 hipDeviceProp_t Structure Enhancements

Changes to the hipDeviceProp_t structure in the next major release may result in backward incompatibility. More details on these changes will be provided in subsequent releases.

8.19.1.2 ROCDebugger Enhancements

8.19.1.2.1 Multi-language Source Level Debugger

The compiler now generates a source-level variable and function argument debug information.

The accuracy is guaranteed if the compiler options -g -O0 are used and apply only to HIP.

This enhancement enables ROCDebugger users to interact with the HIP source-level variables and function arguments.

> Note
>
> The newly-suggested compiler -g option must be used instead of the previously-suggested -ggdb option. Although the effect of these two options is currently equivalent, this is not guaranteed for the future and might get changed by the upstream LLVM community.

8.19.1.2.2 Machine Interface Lanes Support

ROCDebugger Machine Interface (MI) extends support to lanes. The following enhancements are made:

- Added a new -lane-info command, listing the current thread's lanes.
- The -thread-select command now supports a lane switch to switch to a specific lane of a thread:

```
-thread-select -l LANE THREAD
```

- The =thread-selected notification gained a lane-id attribute. This enables the frontend to know which lane of the thread was selected.
- The *stopped asynchronous record gained lane-id and hit-lanes attributes. The former indicates which lane is selected, and the latter indicates which lanes explain the stop.
- MI commands now accept a global –lane option, similar to the global –thread and –frame options.
- MI varobjs are now lane-aware.

For more information, refer to the ROC Debugger User Guide at ROCgdb.

8.19.1.2.3 Enhanced - clone-inferior Command

The clone-inferior command now ensures that the TTY, CMD, ARGS, and AMDGPU PRECISE-MEMORY settings are copied from the original inferior to the new one. All modifications to the environment variables done using the 'set environment' or 'unset environment' commands are also copied to the new inferior.

8.19.1.3 MIOpen Support for RDNA GPUs

This release includes support for AMD Radeon™ Pro W6800, in addition to other bug fixes and performance improvements as listed below:

- MIOpen now supports RDNA GPUs!! (via MIOpen PRs 973, 780, 764, 740, 739, 677, 660, 653, 493, 498)
- Fixed a correctness issue with ImplicitGemm algorithm
- Updated the performance data for new kernel versions
- Improved MIOpen build time by splitting large kernel header files
- Fixed an issue in reduction kernels for padded tensors
- Various other bug fixes and performance improvements

For more information, see Documentation.

8.19.1.4 Checkpoint Restore Support With CRIU

The new Checkpoint Restore in Userspace (CRIU) functionality is implemented to support AMD GPU and ROCm applications.

CRIU is a userspace tool to Checkpoint and Restore an application.

CRIU lacked the support for checkpoint restore applications that used device files such as a GPU. With this ROCm release, CRIU is enhanced with a new plugin to support AMD GPUs, which includes:

- Single and Multi GPU systems (Gfx9)
- Checkpoint / Restore on a different system
- Checkpoint / Restore inside a docker container
- PyTorch
- Tensorflow
- Using CRIU Image Streamer

For more information, refer to https://github.com/checkpoint-restore/criu/tree/criu-dev/plugins/amdgpu

> Note
>
> The CRIU plugin (amdgpu_plugin) is merged upstream with the CRIU repository. The KFD kernel patches are also available upstream with the amd-staging-drm-next branch (public) and the ROCm 5.1 release branch.
>
> Note
>
> This is a Beta release of the Checkpoint and Restore functionality, and some features are not available in this release.

For more information, refer to the following websites:

- https://github.com/RadeonOpenCompute/criu/blob/amdgpu_plugin-03252022/Documentation/amdgpu_plugin.txt
- https://criu.org/Main_Page

8.19.2 Fixed Defects

The following defects are fixed in this release.

8.19.2.1 Driver Fails To Load after Installation

The issue with the driver failing to load after ROCm installation is now fixed.

The driver installs successfully, and the server reboots with working rocminfo and clinfo.

8.19.2.2 ROCDebugger Fixed Defects

8.19.2.2.1 Breakpoints in GPU kernel code Before Kernel Is Loaded

Previously, setting a breakpoint in device code by line number before the device code was loaded into the program resulted in ROCgdb incorrectly moving the breakpoint to the first following line that contains host code.

Now, the breakpoint is left pending. When the GPU kernel gets loaded, the breakpoint resolves to a location in the kernel.

8.19.2.2.2 Registers Invalidated After Write

Previously, the stale just-written value was presented as a current value.

ROCgdb now invalidates the cached values of registers whose content might differ after being written. For example, registers with read-only bits.

ROCgdb also invalidates all volatile registers when a volatile register is written. For example, writing VCC invalidates the content of STATUS as STATUS.VCCZ may change.

8.19.2.2.3 Scheduler-locking and GPU Wavefronts

When scheduler-locking is in effect, new wavefronts created by a resumed thread, CPU, or GPU wavefront, are held in the halt state. For example, the “set scheduler-locking” command.

8.19.2.2.4 ROCDebugger Fails Before Completion of Kernel Execution

It was possible (although erroneous) for a debugger to load GPU code in memory, send it to the device, start executing a kernel on the device, and dispose of the original code before the kernel had finished execution. If a breakpoint was hit after this point, the debugger failed with an internal error while trying to access the debug information.

This issue is now fixed by ensuring that the debugger keeps a local copy of the original code and debug information.

8.19.3 Known Issues

8.19.3.1 Random Memory Access Fault Errors Observed While Running Math Libraries Unit Tests

Issue: Random memory access fault issues are observed while running Math libraries unit tests. This issue is encountered in ROCm v5.0, ROCm v5.0.1, and ROCm v5.0.2.

Note, the faults only occur in the SRIOV environment.

Workaround: Use SDMA to update the page table. The Guest set up steps are as follows:

```
sudo modprobe amdgpu vm_update_mode=0
```

To verify, use

Guest:

```
cat /sys/module/amdgpu/parameters/vm_update_mode 0
```

Where expectation is 0.

8.19.3.2 CU Masking Causes Application to Freeze

Using CU Masking results in an application freeze or runs exceptionally slowly. This issue is noticed only in the GFX10 suite of products. Note, this issue is observed only in GFX10 suite of products.

This issue is under active investigation at this time.

8.19.3.3 Failed Checkpoint in Docker Containers

A defect with Ubuntu images kernel-5.13-30-generic and kernel-5.13-35-generic with Overlay FS results in incorrect reporting of the mount ID.

This issue with Ubuntu causes CRIU checkpointing to fail in Docker containers.

As a workaround, use an older version of the kernel. For example, Ubuntu 5.11.0-46-generic.

8.19.3.4 Issue with Restoring Workloads Using Cooperative Groups Feature

Workloads that use the cooperative groups function to ensure all waves can be resident at the same time may fail to restore correctly. This issue is under investigation and will be fixed in a future release.

8.19.3.5 Radeon Pro V620 and W6800 Workstation GPUs

8.19.3.5.1 No Support for ROCDebugger on SRIOV

ROCDebugger is not supported in the SRIOV environment on any GPU.

This is a known issue and will be fixed in a future release.

8.19.3.6 Random Error Messages in ROCm SMI for SR-IOV

Random error messages are generated by unsupported functions or commands.

This is a known issue and will be fixed in a future release.

8.19.4 Library Changes in ROCM 5.1.0

Library	Version
hipBLAS	0.49.0 0.50.0
hipCUB	2.10.13 2.11.0
hipFFT	1.0.4 1.0.7
hipSOLVER	1.2.0 1.3.0
hipSPARSE	2.0.0 2.1.0
rccl	2.10.3 2.11.4
rocALUTION	2.0.1 2.0.2
rocBLAS	2.42.0 2.43.0
rocFFT	1.0.13 1.0.16
rocPRIM	2.10.12 2.10.13
rocRAND	2.10.12 2.10.13
rocSOLVER	3.16.0 3.17.0
rocSPARSE	2.0.0 2.1.0
rocThrust	2.13.0 2.14.0
Tensile	4.31.0 4.32.0

8.19.4.1 hipBLAS 0.50.0

hipBLAS 0.50.0 for ROCm 5.1.0

8.19.4.1.1 Added

- Added library version and device information to hipblas-test output
- Added –rocsolver-path command line option to choose path to pre-built rocSOLVER, as absolute or relative path
- Added –cmake_install command line option to update cmake to minimum version if required
- Added cmake-arg parameter to pass in cmake arguments while building
- Added infrastructure to support readthedocs hipBLAS documentation.

8.19.4.1.2 Fixed

- Added hipblasVersionMinor define. hipblaseVersionMinor remains defined for backwards compatibility.
- Doxygen warnings in hipblas.h header file.

8.19.4.1.3 Changed

- rocblas-path command line option can be specified as either absolute or relative path
- Help message improvements in install.sh and rmake.py
- Updated googletest dependency from 1.10.0 to 1.11.0

8.19.4.2 hipCUB 2.11.0

hipCUB 2.11.0 for ROCm 5.1.0

8.19.4.2.1 Added

- Device segmented sort
- Warp merge sort, WarpMask and thread sort from cub 1.15.0 supported in hipCUB
- Device three way partition

8.19.4.2.2 Changed

- Device_scan and device_segmented_scan: inclusive_scan now uses the input-type as accumulator-type, exclusive_scan uses initial-value-type.
  - This particularly changes behaviour of small-size input types with large-size output types (e.g. short input, int output).
  - And low-res input with high-res output (e.g. float input, double output)
  - Block merge sort no longer supports non power of two blocksizes

8.19.4.3 hipFFT 1.0.7

hipFFT 1.0.7 for ROCm 5.1.0

8.19.4.3.1 Changed

- Use fft_params struct for accuracy and benchmark clients.

8.19.4.4 hipSOLVER 1.3.0

hipSOLVER 1.3.0 for ROCm 5.1.0

8.19.4.4.1 Added

- Added functions
  - gels
    - hipsolverSSgels_bufferSize, hipsolverDDgels_bufferSize, hipsolverCCgels_bufferSize, hipsolverZZgels_bufferSize
    - hipsolverSSgels, hipsolverDDgels, hipsolverCCgels, hipsolverZZgels
- Added library version and device information to hipsolver-test output.
- Added compatibility API with hipsolverDn prefix.
- Added compatibility-only functions
  - gesvdj
    - hipsolverDnSgesvdj_bufferSize, hipsolverDnDgesvdj_bufferSize, hipsolverDnCgesvdj_bufferSize, hipsolverDnZgesvdj_bufferSize
    - hipsolverDnSgesvdj, hipsolverDnDgesvdj, hipsolverDnCgesvdj, hipsolverDnZgesvdj
  - gesvdjBatched
    - hipsolverDnSgesvdjBatched_bufferSize, hipsolverDnDgesvdjBatched_bufferSize, hipsolverDnCgesvdjBatched_bufferSize, hipsolverDnZgesvdjBatched_bufferSize
    - hipsolverDnSgesvdjBatched, hipsolverDnDgesvdjBatched, hipsolverDnCgesvdjBatched, hipsolverDnZgesvdjBatched
  - syevj
    - hipsolverDnSsyevj_bufferSize, hipsolverDnDsyevj_bufferSize, hipsolverDnCheevj_bufferSize, hipsolverDnZheevj_bufferSize
    - hipsolverDnSsyevj, hipsolverDnDsyevj, hipsolverDnCheevj, hipsolverDnZheevj
  - syevjBatched
    - hipsolverDnSsyevjBatched_bufferSize, hipsolverDnDsyevjBatched_bufferSize, hipsolverDnCheevjBatched_bufferSize, hipsolverDnZheevjBatched_bufferSize
    - hipsolverDnSsyevjBatched, hipsolverDnDsyevjBatched, hipsolverDnCheevjBatched, hipsolverDnZheevjBatched
  - sygvj
    - hipsolverDnSsygvj_bufferSize, hipsolverDnDsygvj_bufferSize, hipsolverDnChegvj_bufferSize, hipsolverDnZhegvj_bufferSize
    - hipsolverDnSsygvj, hipsolverDnDsygvj, hipsolverDnChegvj, hipsolverDnZhegvj

8.19.4.4.2 Changed

- The rocSOLVER backend now allows hipsolverXXgels and hipsolverXXgesv to be called in-place when B == X.
- The rocSOLVER backend now allows rwork to be passed as a null pointer to hipsolverXgesvd.

8.19.4.4.3 Fixed

- bufferSize functions will now return HIPSOLVER_STATUS_NOT_INITIALIZED instead of HIPSOLVER_STATUS_INVALID_VALUE when both handle and lwork are null.
- Fixed rare memory allocation failure in syevd/heevd and sygvd/hegvd caused by improper workspace array allocation outside of rocSOLVER.

8.19.4.5 hipSPARSE 2.1.0

hipSPARSE 2.1.0 for ROCm 5.1.0

8.19.4.5.1 Added

- Added gtsv_interleaved_batch and gpsv_interleaved_batch routines
- Add SpGEMM_reuse

8.19.4.5.2 Changed

- Changed BUILD_CUDA with USE_CUDA in install script and cmake files
- Update googletest to 11.1

8.19.4.5.3 Improved

- Fixed a bug in SpMM Alg versioning

8.19.4.5.4 Known Issues

- none

8.19.4.6 rccl 2.11.4

RCCL 2.11.4 for ROCm 5.1.0

8.19.4.6.1 Added

- Compatibility with NCCL 2.11.4

8.19.4.6.2 Known Issues

- Managed memory is not currently supported for clique-based kernels

8.19.4.7 rocALUTION 2.0.2

rocALUTION 2.0.2 for ROCm 5.1.0

8.19.4.7.1 Added

- Added out-of-place matrix transpose functionality
- Added LocalVector<bool>

8.19.4.8 rocBLAS 2.43.0

rocBLAS 2.43.0 for ROCm 5.1.0

8.19.4.8.1 Added

- Option to install script for number of jobs to use for rocBLAS and Tensile compilation (-j, –jobs)
- Option to install script to build clients without using any Fortran (–clients_no_fortran)
- rocblas_client_initialize function, to perform rocBLAS initialize for clients(benchmark/test) and report the execution time.
- Added tests for output of reduction functions when given bad input
- Added user specified initialization (rand_int/trig_float/hpl) for initializing matrices and vectors in rocblas-bench

8.19.4.8.2 Optimizations

- Improved performance of trsm with side == left and n == 1
- Improved perforamnce of trsm with side == left and m <= 32 along with side == right and n <= 32

8.19.4.8.3 Changed

- For syrkx and trmm internal API use rocblas_stride datatype for offset
- For non-batched and batched gemm_ex functions if the C matrix pointer equals the D matrix pointer (aliased) their respective type and leading dimension arguments must now match
- Test client dependencies updated to GTest 1.11
- non-global false positives reported by cppcheck from file based suppression to inline suppression. File based suppression will only be used for global false positives.
- Help menu messages in install.sh
- For ger function, typecast the 'lda'(offset) datatype to size_t during offset calculation to avoid overflow and remove duplicate template functions.
- Modified default initialization from rand_int to hpl for initializing matrices and vectors in rocblas-bench

8.19.4.8.4 Fixed

- For function trmv (non-transposed cases) avoid overflow in offset calculation
- Fixed cppcheck errors/warnings
- Fixed doxygen warnings

8.19.4.9 rocFFT 1.0.16

rocFFT 1.0.16 for ROCm 5.1.0

8.19.4.9.1 Changed

- Supported unaligned tile dimension for SBRC_2D kernels.
- Improved (more RAII) test and benchmark infrastructure.
- Enabled runtime compilation of length-2304 FFT kernel during plan creation.

8.19.4.9.2 Optimizations

- Optimized more large 1D cases by using L1D_CC plan.
- Optimized 3D 200^3 C2R case.
- Optimized 1D 2^30 double precision on MI200.

8.19.4.9.3 Fixed

- Fixed correctness of some R2C transforms with unusual strides.

8.19.4.9.4 Removed

- The hipFFT API (header) has been removed from after a long deprecation period. Please use the hipFFT package/repository to obtain the hipFFT API.

8.19.4.10 rocPRIM 2.10.13

rocPRIM 2.10.13 for ROCm 5.1.0

8.19.4.10.1 Fixed

- Fixed radix sort int64_t bug introduced in [2.10.11]

8.19.4.10.2 Added

- Future value
- Added device partition_three_way to partition input to three output iterators based on two predicates

8.19.4.10.3 Changed

- The reduce/scan algorithm precision issues in the tests has been resolved for half types.

8.19.4.10.4 Known Issues

- device_segmented_radix_sort unit test failing for HIP on Windows

8.19.4.11 rocRAND 2.10.13

rocRAND 2.10.13 for ROCm 5.1.0

8.19.4.11.1 Added

- Generating a random sequence different sizes now produces the same sequence without gaps indepent of how many values are generated per call.
  - Only in the case of XORWOW, MRG32K3A, PHILOX4X32_10, SOBOL32 and SOBOL64
  - This only holds true if the size in each call is a divisor of the distributions output_width due to performance
  - Similarly the output pointer has to be aligned to output_width * sizeof(output_type)

8.19.4.11.2 Changed

- hipRAND split into a separate package
- Header file installation location changed to match other libraries.
  - Using the rocrand.h header file should now use #include <rocrand/rocrand.h>, rather than #include <rocrand/rocrand.h>
- rocRAND still includes hipRAND using a submodule
  - The rocRAND package also sets the provides field with hipRAND, so projects which require hipRAND can begin to specify it.

8.19.4.11.3 Fixed

- Fix offset behaviour for XORWOW, MRG32K3A and PHILOX4X32_10 generator, setting offset now correctly generates the same sequence starting from the offset.
  - Only uniform int and float will work as these can be generated with a single call to the generator

8.19.4.11.4 Known Issues

- kernel_xorwow unit test is failing for certain GPU architectures.

8.19.4.12 rocSOLVER 3.17.0

rocSOLVER 3.17.0 for ROCm 5.1.0

8.19.4.12.1 Optimized

- Optimized non-pivoting and batch cases of the LU factorization

8.19.4.12.2 Fixed

- Fixed missing synchronization in SYTRF with rocblas_fill_lower that could potentially result in incorrect pivot values.
- Fixed multi-level logging output to file with the ROCSOLVER_LOG_PATH, ROCSOLVER_LOG_TRACE_PATH, ROCSOLVER_LOG_BENCH_PATH and ROCSOLVER_LOG_PROFILE_PATH environment variables.
- Fixed performance regression in the batched LU factorization of tiny matrices

8.19.4.13 rocSPARSE 2.1.0

rocSPARSE 2.1.0 for ROCm 5.1.0

8.19.4.13.1 Added

- gtsv_interleaved_batch
- gpsv_interleaved_batch
- SpGEMM_reuse
- Allow copying of mat info struct

8.19.4.13.2 Improved

- Optimization for SDDMM
- Allow unsorted matrices in csrgemm multipass algorithm

8.19.4.13.3 Known Issues

- none

8.19.4.14 rocThrust 2.14.0

rocThrust 2.14.0 for ROCm 5.1.0

8.19.4.14.1 Added

- Updated to match upstream Thrust 1.15.0

8.19.4.14.2 Known Issues

- async_copy, partition, and stable_sort_by_key unit tests are failing on HIP on Windows.

8.19.4.15 Tensile 4.32.0

Tensile 4.32.0 for ROCm 5.1.0

8.19.4.15.1 Added

- Better control of parallelism to control memory usage
- Support for multiprocessing on Windows for TensileCreateLibrary
- New JSD metric and metric selection functionality
- Initial changes to support two-tier solution selection

8.19.4.15.2 Optimized

- Optimized runtime of TensileCreateLibraries by reducing max RAM usage
- StoreCInUnroll additional optimizations plus adaptive K support
- DGEMM NN optimizations with PrefetchGlobalRead(PGR)=2 support

8.19.4.15.3 Changed

- Update Googletest to 1.11.0

8.19.4.15.4 Removed

- Remove no longer supported benchmarking steps

8.20 ROCm 5.0.2

8.20.1 Fixed Defects

The following defects are fixed in the ROCm v5.0.2 release.

8.20.1.1 Issue with hostcall Facility in HIP Runtime

In ROCm v5.0, when using the “assert()” call in a HIP kernel, the compiler may sometimes fail to emit kernel metadata related to the hostcall facility, which results in incomplete initialization of the hostcall facility in the HIP runtime. This can cause the HIP kernel to crash when it attempts to execute the “assert()” call.

The root cause was an incorrect check in the compiler to determine whether the hostcall facility is required by the kernel. This is fixed in the ROCm v5.0.2 release.

The resolution includes a compiler change, which emits the required metadata by default, unless the compiler can prove that the hostcall facility is not required by the kernel. This ensures that the “assert()” call never fails.

Note: This fix may lead to breakage in some OpenMP offload use cases, which use print inside a target region and result in an abort in device code. The issue will be fixed in a future release. Compatibility Matrix Updates to ROCm Deep Learning Guide

The compatibility matrix in the AMD Deep Learning Guide is updated for ROCm v5.0.2.

8.20.2 Library Changes in ROCM 5.0.2

Library	Version
hipBLAS	0.49.0
hipCUB	2.10.13
hipFFT	1.0.4
hipSOLVER	1.2.0
hipSPARSE	2.0.0
rccl	2.10.3
rocALUTION	2.0.1
rocBLAS	2.42.0
rocFFT	1.0.13
rocPRIM	2.10.12
rocRAND	2.10.12
rocSOLVER	3.16.0
rocSPARSE	2.0.0
rocThrust	2.13.0
Tensile	4.31.0

8.21 ROCm 5.0.1

8.21.1 Deprecations and Warnings

8.21.1.1 Refactor of HIPCC/HIPCONFIG

In prior ROCm releases, by default, the hipcc/hipconfig Perl scripts were used to identify and set target compiler options, target platform, compiler, and runtime appropriately.

In ROCm v5.0.1, hipcc.bin and hipconfig.bin have been added as the compiled binary implementations of the hipcc and hipconfig. These new binaries are currently a work-in-progress, considered, and marked as experimental. ROCm plans to fully transition to hipcc.bin and hipconfig.bin in the a future ROCm release. The existing hipcc and hipconfig Perl scripts are renamed to hipcc.pl and hipconfig.pl respectively. New top-level hipcc and hipconfig Perl scripts are created, which can switch between the Perl script or the compiled binary based on the environment variable HIPCC_USE_PERL_SCRIPT.

In ROCm 5.0.1, by default, this environment variable is set to use hipcc and hipconfig through the Perl scripts.

Subsequently, Perl scripts will no longer be available in ROCm in a future release.

8.21.2 Library Changes in ROCM 5.0.1

Library	Version
hipBLAS	0.49.0
hipCUB	2.10.13
hipFFT	1.0.4
hipSOLVER	1.2.0
hipSPARSE	2.0.0
rccl	2.10.3
rocALUTION	2.0.1
rocBLAS	2.42.0
rocFFT	1.0.13
rocPRIM	2.10.12
rocRAND	2.10.12
rocSOLVER	3.16.0
rocSPARSE	2.0.0
rocThrust	2.13.0
Tensile	4.31.0

8.22 ROCm 5.0.0

8.22.1 What's New in This Release

8.22.1.1 HIP Enhancements

The ROCm v5.0 release consists of the following HIP enhancements.

8.22.1.1.1 HIP Installation Guide Updates

The HIP Installation Guide is updated to include building HIP from source on the NVIDIA platform.

Refer to the HIP Installation Guide v5.0 for more details.

8.22.1.1.2 Managed Memory Allocation

Managed memory, including the __managed__ keyword, is now supported in the HIP combined host/device compilation. Through unified memory allocation, managed memory allows data to be shared and accessible to both the CPU and GPU using a single pointer. The allocation is managed by the AMD GPU driver using the Linux Heterogeneous Memory Management (HMM) mechanism. The user can call managed memory API hipMallocManaged to allocate a large chunk of HMM memory, execute kernels on a device, and fetch data between the host and device as needed.

> Note
>
> In a HIP application, it is recommended to do a capability check before calling the managed memory APIs. For example,

```
int managed_memory = 0;
HIPCHECK(hipDeviceGetAttribute(&managed_memory,
  hipDeviceAttributeManagedMemory,p_gpuDevice));
if (!managed_memory ) {
  printf ("info: managed memory access not supported on the device %d\n Skipped\n", p_gpuDevice);
}
else {
  HIPCHECK(hipSetDevice(p_gpuDevice));
  HIPCHECK(hipMallocManaged(&Hmm, N * sizeof(T)));
. . .
}
```

Note

The managed memory capability check may not be necessary; however, if HMM is not supported, managed malloc will fall back to using system memory. Other managed memory API calls will, then, have

Refer to the HIP API documentation for more details on managed memory APIs.

For the application, see

https://github.com/ROCm-Developer-Tools/HIP/blob/rocm-4.5.x/tests/src/runtimeApi/memory/hipMallocManaged.cpp

8.22.1.2 New Environment Variable

The following new environment variable is added in this release:

Environment Variable	Value	Description
HSA_	0 or 1 (default is 0)	Some processors support more CUs than can reliably be used in a cooperative dispatch. Setting the environment variable HSA_COOP_CU_COUNT to 1 will cause ROCr to return the correct CU count for cooperative groups through the HSA_AMD_AGENT_INFO_COOPERATIVE_COMPUTE_UNIT_COUNT attribute of hsa_agent_get_info(). Setting HSA_COOP_CU_COUNT to other values, or leaving it unset, will cause ROCr to return the same CU count for the attributes HSA_AMD_AGENT_INFO_COOPERATIVE_COMPUTE_UNIT_COUNT and HSA_AMD_AGENT_INFO_COMPUTE_UNIT_COUNT. Future ROCm releases will make HSA_COOP_CU_COUNT=1 the default.

8.22.2 Breaking Changes

8.22.2.1 Runtime Breaking Change

Re-ordering of the enumerated type in hip_runtime_api.h to better match NV. See below for the difference in enumerated types.

ROCm software will be affected if any of the defined enums listed below are used in the code. Applications built with ROCm v5.0 enumerated types will work with a ROCm 4.5.2 driver. However, an undefined behavior error will occur with a ROCm v4.5.2 application that uses these enumerated types with a ROCm 5.0 runtime.

```
typedef enum hipDeviceAttribute_t {
-    hipDeviceAttributeMaxThreadsPerBlock,       ///< Maximum number of threads per block.
-    hipDeviceAttributeMaxBlockDimX,             ///< Maximum x-dimension of a block.
-    hipDeviceAttributeMaxBlockDimY,             ///< Maximum y-dimension of a block.
-    hipDeviceAttributeMaxBlockDimZ,             ///< Maximum z-dimension of a block.
-    hipDeviceAttributeMaxGridDimX,              ///< Maximum x-dimension of a grid.
-    hipDeviceAttributeMaxGridDimY,              ///< Maximum y-dimension of a grid.
-    hipDeviceAttributeMaxGridDimZ,              ///< Maximum z-dimension of a grid.
-    hipDeviceAttributeMaxSharedMemoryPerBlock,  ///< Maximum shared memory available per block in
-                                                ///< bytes.
-    hipDeviceAttributeTotalConstantMemory,      ///< Constant memory size in bytes.
-    hipDeviceAttributeWarpSize,                 ///< Warp size in threads.
-    hipDeviceAttributeMaxRegistersPerBlock,  ///< Maximum number of 32-bit registers available to a
-                                             ///< thread block. This number is shared by all thread
-                                             ///< blocks simultaneously resident on a
-                                             ///< multiprocessor.
-    hipDeviceAttributeClockRate,             ///< Peak clock frequency in kilohertz.
-    hipDeviceAttributeMemoryClockRate,       ///< Peak memory clock frequency in kilohertz.
-    hipDeviceAttributeMemoryBusWidth,        ///< Global memory bus width in bits.
-    hipDeviceAttributeMultiprocessorCount,   ///< Number of multiprocessors on the device.
-    hipDeviceAttributeComputeMode,           ///< Compute mode that device is currently in.
-    hipDeviceAttributeL2CacheSize,  ///< Size of L2 cache in bytes. 0 if the device doesn't have L2
-                                    ///< cache.
-    hipDeviceAttributeMaxThreadsPerMultiProcessor,  ///< Maximum resident threads per
-                                                    ///< multiprocessor.
-    hipDeviceAttributeComputeCapabilityMajor,       ///< Major compute capability version number.
-    hipDeviceAttributeComputeCapabilityMinor,       ///< Minor compute capability version number.
-    hipDeviceAttributeConcurrentKernels,  ///< Device can possibly execute multiple kernels
-                                          ///< concurrently.
-    hipDeviceAttributePciBusId,           ///< PCI Bus ID.
-    hipDeviceAttributePciDeviceId,        ///< PCI Device ID.
-    hipDeviceAttributeMaxSharedMemoryPerMultiprocessor,  ///< Maximum Shared Memory Per
-                                                         ///< Multiprocessor.
-    hipDeviceAttributeIsMultiGpuBoard,                   ///< Multiple GPU devices.
-    hipDeviceAttributeIntegrated,                        ///< iGPU
-    hipDeviceAttributeCooperativeLaunch,                 ///< Support cooperative launch
-    hipDeviceAttributeCooperativeMultiDeviceLaunch,      ///< Support cooperative launch on multiple devices
-    hipDeviceAttributeMaxTexture1DWidth,    ///< Maximum number of elements in 1D images
-    hipDeviceAttributeMaxTexture2DWidth,    ///< Maximum dimension width of 2D images in image elements
-    hipDeviceAttributeMaxTexture2DHeight,   ///< Maximum dimension height of 2D images in image elements
-    hipDeviceAttributeMaxTexture3DWidth,    ///< Maximum dimension width of 3D images in image elements
-    hipDeviceAttributeMaxTexture3DHeight,   ///< Maximum dimensions height of 3D images in image elements
-    hipDeviceAttributeMaxTexture3DDepth,    ///< Maximum dimensions depth of 3D images in image elements
+    hipDeviceAttributeCudaCompatibleBegin = 0,
```

(continues on next page)

(continued from previous page)

```
-    hipDeviceAttributeHdpMemFlushCntl,      ///< Address of the HDP_MEM_COHERENCY_FLUSH_
↪CNTL register
-    hipDeviceAttributeHdpRegFlushCntl,      ///< Address of the HDP_REG_COHERENCY_FLUSH_CNTL␣
↪register
+    hipDeviceAttributeEccEnabled = hipDeviceAttributeCudaCompatibleBegin, ///< Whether ECC support is␣
↪enabled.
+    hipDeviceAttributeAccessPolicyMaxWindowSize,        ///< Cuda only. The maximum size of the window␣
↪policy in bytes.
+    hipDeviceAttributeAsyncEngineCount,                 ///< Cuda only. Asynchronous engines number.
+    hipDeviceAttributeCanMapHostMemory,                 ///< Whether host memory can be mapped into␣
↪device address space
+    hipDeviceAttributeCanUseHostPointerForRegisteredMem,///< Cuda only. Device can access host registered␣
↪memory
+                                                        ///< at the same virtual address as the CPU
+    hipDeviceAttributeClockRate,                        ///< Peak clock frequency in kilohertz.
+    hipDeviceAttributeComputeMode,                      ///< Compute mode that device is currently in.
+    hipDeviceAttributeComputePreemptionSupported,       ///< Cuda only. Device supports Compute␣
↪Preemption.
+    hipDeviceAttributeConcurrentKernels,                ///< Device can possibly execute multiple kernels␣
↪concurrently.
+    hipDeviceAttributeConcurrentManagedAccess,          ///< Device can coherently access managed memory␣
↪concurrently with the CPU
+    hipDeviceAttributeCooperativeLaunch,                ///< Support cooperative launch
+    hipDeviceAttributeCooperativeMultiDeviceLaunch,     ///< Support cooperative launch on multiple devices
+    hipDeviceAttributeDeviceOverlap,                    ///< Cuda only. Device can concurrently copy memory and␣
↪execute a kernel.
+                                                        ///< Deprecated. Use instead asyncEngineCount.
+    hipDeviceAttributeDirectManagedMemAccessFromHost,   ///< Host can directly access managed memory on
+                                                        ///< the device without migration
+    hipDeviceAttributeGlobalL1CacheSupported,           ///< Cuda only. Device supports caching globals in L1
+    hipDeviceAttributeHostNativeAtomicSupported,        ///< Cuda only. Link between the device and the host␣
↪supports native atomic operations
+    hipDeviceAttributeIntegrated,                       ///< Device is integrated GPU
+    hipDeviceAttributeIsMultiGpuBoard,                  ///< Multiple GPU devices.
+    hipDeviceAttributeKernelExecTimeout,                ///< Run time limit for kernels executed on the device
+    hipDeviceAttributeL2CacheSize,                      ///< Size of L2 cache in bytes. 0 if the device doesn't have␣
↪L2 cache.
+    hipDeviceAttributeLocalL1CacheSupported,            ///< caching locals in L1 is supported
+    hipDeviceAttributeLuid,                             ///< Cuda only. 8-byte locally unique identifier in 8 bytes.␣
↪Undefined on TCC and non-Windows platforms
+    hipDeviceAttributeLuidDeviceNodeMask,               ///< Cuda only. Luid device node mask. Undefined on␣
↪TCC and non-Windows platforms
+    hipDeviceAttributeComputeCapabilityMajor,           ///< Major compute capability version number.
+    hipDeviceAttributeManagedMemory,                    ///< Device supports allocating managed memory on␣
↪this system
+    hipDeviceAttributeMaxBlocksPerMultiProcessor,       ///< Cuda only. Max block size per multiprocessor
+    hipDeviceAttributeMaxBlockDimX,                     ///< Max block size in width.
+    hipDeviceAttributeMaxBlockDimY,                     ///< Max block size in height.
+    hipDeviceAttributeMaxBlockDimZ,                     ///< Max block size in depth.
+    hipDeviceAttributeMaxGridDimX,                      ///< Max grid size  in width.
+    hipDeviceAttributeMaxGridDimY,                      ///< Max grid size  in height.
+    hipDeviceAttributeMaxGridDimZ,                      ///< Max grid size  in depth.
+    hipDeviceAttributeMaxSurface1D,                     ///< Maximum size of 1D surface.
+    hipDeviceAttributeMaxSurface1DLayered,              ///< Cuda only. Maximum dimensions of 1D layered␣
↪surface.
+    hipDeviceAttributeMaxSurface2D,                     ///< Maximum dimension (width, height) of 2D surface.
```

(continues on next page)

(continued from previous page)

```
+    hipDeviceAttributeMaxSurface2DLayered,             ///< Cuda only. Maximum dimensions of 2D layered␣
↪surface.
+    hipDeviceAttributeMaxSurface3D,                   ///< Maximum dimension (width, height, depth) of 3D␣
↪surface.
+    hipDeviceAttributeMaxSurfaceCubemap,              ///< Cuda only. Maximum dimensions of Cubemap␣
↪surface.
+    hipDeviceAttributeMaxSurfaceCubemapLayered,       ///< Cuda only. Maximum dimension of Cubemap␣
↪layered surface.
+    hipDeviceAttributeMaxTexture1DWidth,              ///< Maximum size of 1D texture.
+    hipDeviceAttributeMaxTexture1DLayered,            ///< Cuda only. Maximum dimensions of 1D layered␣
↪texture.
+    hipDeviceAttributeMaxTexture1DLinear,             ///< Maximum number of elements allocatable in a 1D␣
↪linear texture.
+                                                      ///< Use cudaDeviceGetTexture1DLinearMaxWidth() instead on␣
↪Cuda.
+    hipDeviceAttributeMaxTexture1DMipmap,             ///< Cuda only. Maximum size of 1D mipmapped␣
↪texture.
+    hipDeviceAttributeMaxTexture2DWidth,              ///< Maximum dimension width of 2D texture.
+    hipDeviceAttributeMaxTexture2DHeight,             ///< Maximum dimension hight of 2D texture.
+    hipDeviceAttributeMaxTexture2DGather,             ///< Cuda only. Maximum dimensions of 2D texture if␣
↪gather operations  performed.
+    hipDeviceAttributeMaxTexture2DLayered,            ///< Cuda only. Maximum dimensions of 2D layered␣
↪texture.
+    hipDeviceAttributeMaxTexture2DLinear,             ///< Cuda only. Maximum dimensions (width, height,␣
↪pitch) of 2D textures bound to pitched memory.
+    hipDeviceAttributeMaxTexture2DMipmap,             ///< Cuda only. Maximum dimensions of 2D␣
↪mipmapped texture.
+    hipDeviceAttributeMaxTexture3DWidth,              ///< Maximum dimension width of 3D texture.
+    hipDeviceAttributeMaxTexture3DHeight,             ///< Maximum dimension height of 3D texture.
+    hipDeviceAttributeMaxTexture3DDepth,              ///< Maximum dimension depth of 3D texture.
+    hipDeviceAttributeMaxTexture3DAlt,                ///< Cuda only. Maximum dimensions of alternate 3D␣
↪texture.
+    hipDeviceAttributeMaxTextureCubemap,              ///< Cuda only. Maximum dimensions of Cubemap␣
↪texture
+    hipDeviceAttributeMaxTextureCubemapLayered,       ///< Cuda only. Maximum dimensions of Cubemap␣
↪layered texture.
+    hipDeviceAttributeMaxThreadsDim,                  ///< Maximum dimension of a block
+    hipDeviceAttributeMaxThreadsPerBlock,             ///< Maximum number of threads per block.
+    hipDeviceAttributeMaxThreadsPerMultiProcessor,    ///< Maximum resident threads per multiprocessor.
+    hipDeviceAttributeMaxPitch,                       ///< Maximum pitch in bytes allowed by memory copies
+    hipDeviceAttributeMemoryBusWidth,                 ///< Global memory bus width in bits.
+    hipDeviceAttributeMemoryClockRate,                ///< Peak memory clock frequency in kilohertz.
+    hipDeviceAttributeComputeCapabilityMinor,         ///< Minor compute capability version number.
+    hipDeviceAttributeMultiGpuBoardGroupID,           ///< Cuda only. Unique ID of device group on the␣
↪same multi-GPU board
+    hipDeviceAttributeMultiprocessorCount,            ///< Number of multiprocessors on the device.
+    hipDeviceAttributeName,                           ///< Device name.
+    hipDeviceAttributePageableMemoryAccess,           ///< Device supports coherently accessing pageable␣
↪memory
+                                                      ///< without calling hipHostRegister on it
+    hipDeviceAttributePageableMemoryAccessUsesHostPageTables, ///< Device accesses pageable memory via␣
↪the host's page tables
+    hipDeviceAttributePciBusId,                       ///< PCI Bus ID.
+    hipDeviceAttributePciDeviceId,                    ///< PCI Device ID.
+    hipDeviceAttributePciDomainID,                    ///< PCI Domain ID.
+    hipDeviceAttributePersistingL2CacheMaxSize,       ///< Cuda11 only. Maximum l2 persisting lines␣
```

(continues on next page)

(continued from previous page)

```
↪capacity in bytes
+    hipDeviceAttributeMaxRegistersPerBlock,             ///< 32-bit registers available to a thread block. This␣
↪number is shared
+                                                        ///< by all thread blocks simultaneously resident on a multiprocessor.
+    hipDeviceAttributeMaxRegistersPerMultiprocessor,    ///< 32-bit registers available per block.
+    hipDeviceAttributeReservedSharedMemPerBlock,        ///< Cuda11 only. Shared memory reserved by␣
↪CUDA driver per block.
+    hipDeviceAttributeMaxSharedMemoryPerBlock,          ///< Maximum shared memory available per block␣
↪in bytes.
+    hipDeviceAttributeSharedMemPerBlockOptin,           ///< Cuda only. Maximum shared memory per block␣
↪usable by special opt in.
+    hipDeviceAttributeSharedMemPerMultiprocessor,       ///< Cuda only. Shared memory available per␣
↪multiprocessor.
+    hipDeviceAttributeSingleToDoublePrecisionPerfRatio, ///< Cuda only. Performance ratio of single precision␣
↪to double precision.
+    hipDeviceAttributeStreamPrioritiesSupported,        ///< Cuda only. Whether to support stream priorities.
+    hipDeviceAttributeSurfaceAlignment,                 ///< Cuda only. Alignment requirement for surfaces
+    hipDeviceAttributeTccDriver,                        ///< Cuda only. Whether device is a Tesla device using TCC␣
↪driver
+    hipDeviceAttributeTextureAlignment,                 ///< Alignment requirement for textures
+    hipDeviceAttributeTexturePitchAlignment,            ///< Pitch alignment requirement for 2D texture␣
↪references bound to pitched memory;
+    hipDeviceAttributeTotalConstantMemory,              ///< Constant memory size in bytes.
+    hipDeviceAttributeTotalGlobalMem,                   ///< Global memory available on devicice.
+    hipDeviceAttributeUnifiedAddressing,                ///< Cuda only. An unified address space shared with the␣
↪host.
+    hipDeviceAttributeUuid,                             ///< Cuda only. Unique ID in 16 byte.
+    hipDeviceAttributeWarpSize,                         ///< Warp size in threads.

-    hipDeviceAttributeMaxPitch,             ///< Maximum pitch in bytes allowed by memory copies
-    hipDeviceAttributeTextureAlignment,     ///<Alignment requirement for textures
-    hipDeviceAttributeTexturePitchAlignment, ///<Pitch alignment requirement for 2D texture references bound␣
↪to pitched memory;
-    hipDeviceAttributeKernelExecTimeout,    ///<Run time limit for kernels executed on the device
-    hipDeviceAttributeCanMapHostMemory,     ///<Device can map host memory into device address space
-    hipDeviceAttributeEccEnabled,           ///<Device has ECC support enabled
+    hipDeviceAttributeCudaCompatibleEnd = 9999,
+    hipDeviceAttributeAmdSpecificBegin = 10000,

-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedFunc,        ///< Supports cooperative launch on␣
↪multiple
-                                                                  ///devices with unmatched functions
-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim,     ///< Supports cooperative launch on␣
↪multiple
-                                                                  ///devices with unmatched grid dimensions
-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedBlockDim,    ///< Supports cooperative launch on␣
↪multiple
-                                                                  ///devices with unmatched block dimensions
-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedSharedMem,   ///< Supports cooperative launch on␣
↪multiple
-                                                                  ///devices with unmatched shared memories
-    hipDeviceAttributeAsicRevision,         ///< Revision of the GPU in this device
-    hipDeviceAttributeManagedMemory,        ///< Device supports allocating managed memory on this system
-    hipDeviceAttributeDirectManagedMemAccessFromHost, ///< Host can directly access managed memory on
-                                                      /// the device without migration
-    hipDeviceAttributeConcurrentManagedAccess,  ///< Device can coherently access managed memory
```

(continues on next page)

(continued from previous page)

```
-                                        /// concurrently with the CPU
-    hipDeviceAttributePageableMemoryAccess,     ///< Device supports coherently accessing pageable memory
-                                        /// without calling hipHostRegister on it
-    hipDeviceAttributePageableMemoryAccessUsesHostPageTables, ///< Device accesses pageable memory via
-                                        /// the host's page tables
-    hipDeviceAttributeCanUseStreamWaitValue ///< '1' if Device supports hipStreamWaitValue32() and
-                                        ///< hipStreamWaitValue64() , '0' otherwise.
+    hipDeviceAttributeClockInstructionRate = hipDeviceAttributeAmdSpecificBegin,  ///< Frequency in khz of␣
↪the timer used by the device-side "clock*"
+    hipDeviceAttributeArch,                     ///< Device architecture
+    hipDeviceAttributeMaxSharedMemoryPerMultiprocessor,         ///< Maximum Shared Memory␣
↪PerMultiprocessor.
+    hipDeviceAttributeGcnArch,                  ///< Device gcn architecture
+    hipDeviceAttributeGcnArchName,                 ///< Device gcnArch name in 256 bytes
+    hipDeviceAttributeHdpMemFlushCntl,                  ///< Address of the HDP_MEM_COHERENCY_
↪FLUSH_CNTL register
+    hipDeviceAttributeHdpRegFlushCntl,                  ///< Address of the HDP_REG_COHERENCY_
↪FLUSH_CNTL register
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedFunc,      ///< Supports cooperative launch on␣
↪multiple
+                                                ///< devices with unmatched functions
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim,   ///< Supports cooperative launch on␣
↪multiple
+                                                ///< devices with unmatched grid dimensions
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedBlockDim,  ///< Supports cooperative launch on␣
↪multiple
+                                                ///< devices with unmatched block dimensions
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedSharedMem, ///< Supports cooperative launch on␣
↪multiple
+                                                ///< devices with unmatched shared memories
+    hipDeviceAttributeIsLargeBar,                   ///< Whether it is LargeBar
+    hipDeviceAttributeAsicRevision,                 ///< Revision of the GPU in this device
+    hipDeviceAttributeCanUseStreamWaitValue,            ///< '1' if Device supports␣
↪hipStreamWaitValue32() and
+                                                ///< hipStreamWaitValue64() , '0' otherwise.

+    hipDeviceAttributeAmdSpecificEnd = 19999,
+    hipDeviceAttributeVendorSpecificBegin = 20000,
+    // Extended attributes for vendors
 } hipDeviceAttribute_t;

 enum hipComputeMode {
```

8.22.3 Known Issues

8.22.3.1 Incorrect dGPU Behavior When Using AMDVBFlash Tool

The AMDVBFlash tool, used for flashing the VBIOS image to dGPU, does not communicate with the ROM Controller specifically when the driver is present. This is because the driver, as part of its runtime power management feature, puts the dGPU to a sleep state.

As a workaround, users can run amdgpu.runpm=0, which temporarily disables the runtime power management feature from the driver and dynamically changes some power control-related sysfs files.

8.22.3.2 Issue with START Timestamp in ROCProfiler

Users may encounter an issue with the enabled timestamp functionality for monitoring one or multiple counters. ROCProfiler outputs the following four timestamps for each kernel:

- Dispatch
- Start
- End
- Complete

8.22.3.2.1 Issue

This defect is related to the Start timestamp functionality, which incorrectly shows an earlier time than the Dispatch timestamp.

To reproduce the issue,

1. Enable timing using the –timestamp on flag.
2. Use the -i option with the input filename that contains the name of the counter(s) to monitor.
3. Run the program.
4. Check the output result file.

8.22.3.2.2 Current behavior

BeginNS is lower than DispatchNS, which is incorrect.

8.22.3.2.3 Expected behavior

The correct order is:

Dispatch < Start < End < Complete

Users cannot use ROCProfiler to measure the time spent on each kernel because of the incorrect timestamp with counter collection enabled.

8.22.3.2.4 Recommended Workaround

Users are recommended to collect kernel execution timestamps without monitoring counters, as follows:

1. Enable timing using the –timestamp on flag, and run the application.
2. Rerun the application using the -i option with the input filename that contains the name of the counter(s) to monitor, and save this to a different output file using the -o flag.
3. Check the output result file from step 1.
4. The order of timestamps correctly displays as: DispathNS < BeginNS < EndNS < CompleteNS
5. Users can find the values of the collected counters in the output file generated in step 2.

8.22.3.3 Radeon Pro V620 and W6800 Workstation GPUs

8.22.3.3.1 No Support for SMI and ROCDebugger on SRIOV

System Management Interface (SMI) and ROCDebugger are not supported in the SRIOV environment on any GPU. For more information, refer to the Systems Management Interface documentation.

8.22.4 Deprecations and Warnings

8.22.4.1 ROCm Libraries Changes – Deprecations and Deprecation Removal

- The hipFFT.h header is now provided only by the hipFFT package. Up to ROCm 5.0, users would get hipFFT.h in the rocFFT package too.
- The GlobalPairwiseAMG class is now entirely removed, users should use the PairwiseAMG class instead.
- The rocsparse_spmm signature in 5.0 was changed to match that of rocsparse_spmm_ex. In 5.0, rocsparse_spmm_ex is still present, but deprecated. Signature diff for rocsparse_spmm rocsparse_spmm in 5.0

```
rocsparse_status rocsparse_spmm(rocsparse_handle            handle,
                                rocsparse_operation         trans_A,
                                rocsparse_operation         trans_B,
                                const void*                 alpha,
                                const rocsparse_spmat_descr mat_A,
                                const rocsparse_dnmat_descr mat_B,
                                const void*                 beta,
                                const rocsparse_dnmat_descr mat_C,
                                rocsparse_datatype          compute_type,
                                rocsparse_spmm_alg          alg,
                                rocsparse_spmm_stage        stage,
                                size_t*                     buffer_size,
                                void*                       temp_buffer);
```

rocSPARSE_spmm in 4.0

```
rocsparse_status rocsparse_spmm(rocsparse_handle            handle,
                                rocsparse_operation         trans_A,
                                rocsparse_operation         trans_B,
                                const void*                 alpha,
                                const rocsparse_spmat_descr mat_A,
                                const rocsparse_dnmat_descr mat_B,
                                const void*                 beta,
                                const rocsparse_dnmat_descr mat_C,
                                rocsparse_datatype          compute_type,
                                rocsparse_spmm_alg          alg,
                                size_t*                     buffer_size,
                                void*                       temp_buffer);
```

8.22.4.2 HIP API Deprecations and Warnings

8.22.4.2.1 Warning - Arithmetic Operators of HIP Complex and Vector Types

In this release, arithmetic operators of HIP complex and vector types are deprecated.

- As alternatives to arithmetic operators of HIP complex types, users can use arithmetic operators of std::complex types.
- As alternatives to arithmetic operators of HIP vector types, users can use the operators of the native clang vector type associated with the data member of HIP vector types.

During the deprecation, two macros _HIP_ENABLE_COMPLEX_OPERATORS and _HIP_ENABLE_VECTOR_OPERATORS are provided to allow users to conditionally enable arithmetic operators of HIP complex or vector types.

Note, the two macros are mutually exclusive and, by default, set to Off.

The arithmetic operators of HIP complex and vector types will be removed in a future release.

Refer to the HIP API Guide for more information.

8.22.4.3 Warning - Compiler-Generated Code Object Version 4 Deprecation

Support for loading compiler-generated code object version 4 will be deprecated in a future release with no release announcement and replaced with code object 5 as the default version.

The current default is code object version 4.

8.22.4.4 Warning - MIOpenTensile Deprecation

MIOpenTensile will be deprecated in a future release.

8.22.5 Library Changes in ROCM 5.0.0

Library	Version
hipBLAS	0.49.0
hipCUB	2.10.13
hipFFT	1.0.4
hipSOLVER	1.2.0
hipSPARSE	2.0.0
rccl	2.10.3
rocALUTION	2.0.1
rocBLAS	2.42.0
rocFFT	1.0.13
rocPRIM	2.10.12
rocRAND	2.10.12
rocSOLVER	3.16.0
rocSPARSE	2.0.0
rocThrust	2.13.0
Tensile	4.31.0

8.22.5.1 hipBLAS 0.49.0

hipBLAS 0.49.0 for ROCm 5.0.0

8.22.5.1.1 Added

- Added rocSOLVER functions to hipblas-bench
- Added option ROCM_MATHLIBS_API_USE_HIP_COMPLEX to opt-in to use hipFloatComplex and hipDoubleComplex
- Added compilation warning for future trmm changes
- Added documentation to hipblas.h
- Added option to forgo pivoting for getrf and getri when ipiv is nullptr
- Added code coverage option

8.22.5.1.2 Fixed

- Fixed use of incorrect 'HIP_PATH' when building from source.
- Fixed windows packaging
- Allowing negative increments in hipblas-bench
- Removed boost dependency

8.22.5.2 hipCUB 2.10.13

hipCUB 2.10.13 for ROCm 5.0.0

8.22.5.2.1 Fixed

- Added missing includes to hipcub.hpp

8.22.5.2.2 Added

- Bfloat16 support to test cases (device_reduce & device_radix_sort)
- Device merge sort
- Block merge sort
- API update to CUB 1.14.0

8.22.5.2.3 Changed

- The SetupNVCC.cmake automatic target selector select all of the capabalities of all available card for NVIDIA backend.

8.22.5.3 hipFFT 1.0.4

hipFFT 1.0.4 for ROCm 5.0.0

8.22.5.3.1 Fixed

- Add calls to rocFFT setup/cleanup.
- Cmake fixes for clients and backend support.

8.22.5.3.2 Added

- Added support for Windows 10 as a build target.

8.22.5.4 hipSOLVER 1.2.0

hipSOLVER 1.2.0 for ROCm 5.0.0

8.22.5.4.1 Added

- Added functions
  - sytrf
    - hipsolverSsytrf_bufferSize, hipsolverDsytrf_bufferSize, hipsolverCsytrf_bufferSize, hipsolverZsytrf_bufferSize
    - hipsolverSsytrf, hipsolverDsytrf, hipsolverCsytrf, hipsolverZsytrf

8.22.5.4.2 Fixed

- Fixed use of incorrect HIP_PATH when building from source (#40). Thanks @jakub329homola!

8.22.5.5 hipSPARSE 2.0.0

hipSPARSE 2.0.0 for ROCm 5.0.0

8.22.5.5.1 Added

- Added (conjugate) transpose support for csrmv, hybmv and spmv routines

8.22.5.6 rccl 2.10.3

RCCL 2.10.3 for ROCm 5.0.0

8.22.5.6.1 Added

- Compatibility with NCCL 2.10.3

8.22.5.6.2 Known Issues

- Managed memory is not currently supported for clique-based kernels

8.22.5.7 rocALUTION 2.0.1

rocALUTION 2.0.1 for ROCm 5.0.0

8.22.5.7.1 Changed

- Removed deprecated GlobalPairwiseAMG class, please use PairwiseAMG instead.
- Changed to C++ 14 Standard

8.22.5.7.2 Improved

- Added sanitizer option
- Improved documentation

8.22.5.8 rocBLAS 2.42.0

rocBLAS 2.42.0 for ROCm 5.0.0

8.22.5.8.1 Added

- Added rocblas_get_version_string_size convenience function
- Added rocblas_xtrmm_outofplace, an out-of-place version of rocblas_xtrmm
- Added hpl and trig initialization for gemm_ex to rocblas-bench
- Added source code gemm. It can be used as an alternative to Tensile for debugging and development
- Added option ROCM_MATHLIBS_API_USE_HIP_COMPLEX to opt-in to use hipFloatComplex and hipDoubleComplex

8.22.5.8.2 Optimizations

- Improved performance of non-batched and batched single-precision GER for size m > 1024. Performance enhanced by 5-10% measured on a MI100 (gfx908) GPU.
- Improved performance of non-batched and batched HER for all sizes and data types. Performance enhanced by 2-17% measured on a MI100 (gfx908) GPU.

8.22.5.8.3 Changed

- Instantiate templated rocBLAS functions to reduce size of librocblas.so
- Removed static library dependency on msgpack
- Removed boost dependencies for clients

8.22.5.8.4 Fixed

- Option to install script to build only rocBLAS clients with a pre-built rocBLAS library
- Correctly set output of nrm2_batched_ex and nrm2_strided_batched_ex when given bad input
- Fix for dgmm with side == rocblas_side_left and a negative incx
- Fixed out-of-bounds read for small trsm
- Fixed numerical checking for tbmv_strided_batched

8.22.5.9 rocFFT 1.0.13

rocFFT 1.0.13 for ROCm 5.0.0

8.22.5.9.1 Optimizations

- Improved many plans by removing unnecessary transpose steps.
- Optimized scheme selection for 3D problems.
  - Imposed less restrictions on 3D_BLOCK_RC selection. More problems can use 3D_BLOCK_RC and have some performance gain.
  - Enabled 3D_RC. Some 3D problems with SBCC-supported z-dim can use less kernels and get benefit.
  - Force –length 336 336 56 (dp) use faster 3D_RC to avoid it from being skipped by conservative threshold test.
- Optimized some even-length R2C/C2R cases by doing more operations in-place and combining pre/post processing into Stockham kernels.
- Added radix-17.

8.22.5.9.2 Added

- Added new kernel generator for select fused-2D transforms.

8.22.5.9.3 Fixed

- Improved large 1D transform decompositions.

8.22.5.10 rocPRIM 2.10.12

rocPRIM 2.10.12 for ROCm 5.0.0

8.22.5.10.1 Fixed

- Enable bfloat16 tests and reduce threshold for bfloat16
- Fix device scan limit_size feature
- Non-optimized builds no longer trigger local memory limit errors

8.22.5.10.2 Added

- Added scan size limit feature
- Added reduce size limit feature
- Added transform size limit feature
- Add block_load_striped and block_store_striped
- Add gather_to_blocked to gather values from other threads into a blocked arrangement
- The block sizes for device merge sorts initial block sort and its merge steps are now separate in its kernel config
  - the block sort step supports multiple items per thread

8.22.5.10.3 Changed

- size_limit for scan, reduce and transform can now be set in the config struct instead of a parameter
- Device_scan and device_segmented_scan: inclusive_scan now uses the input-type as accumulator-type, exclusive_scan uses initial-value-type.
  - This particularly changes behaviour of small-size input types with large-size output types (e.g. short input, int output).
  - And low-res input with high-res output (e.g. float input, double output)
- Revert old Fiji workaround, because they solved the issue at compiler side
- Update README cmake minimum version number
- Block sort support multiple items per thread

  - currently only powers of two block sizes, and items per threads are supported and only for full blocks

- Bumped the minimum required version of CMake to 3.16

8.22.5.10.4 Known Issues

- Unit tests may soft hang on MI200 when running in hipMallocManaged mode.
- device_segmented_radix_sort, device_scan unit tests failing for HIP on Windows
- ReduceEmptyInput cause random faulire with bfloat16

8.22.5.11 rocRAND 2.10.12

rocRAND 2.10.12 for ROCm 5.0.0

8.22.5.11.1 Changed

- No updates or changes for ROCm 5.0.0.

8.22.5.12 rocSOLVER 3.16.0

rocSOLVER 3.16.0 for ROCm 5.0.0

8.22.5.12.1 Added

- Symmetric matrix factorizations:
  - LASYF
  - SYTF2, SYTRF (with batched and strided_batched versions)
- Added rocsolver_get_version_string_size to help with version string queries
- Added rocblas_layer_mode_ex and the ability to print kernel calls in the trace and profile logs
- Expanded batched and strided_batched sample programs.

8.22.5.12.2 Optimized

- Improved general performance of LU factorization
- Increased parallelism of specialized kernels when compiling from source, reducing build times on multi-core systems.

8.22.5.12.3 Changed

- The rocsolver-test client now prints the rocSOLVER version used to run the tests, rather than the version used to build them
- The rocsolver-bench client now prints the rocSOLVER version used in the benchmark

8.22.5.12.4 Fixed

- Added missing stdint.h include to rocsolver.h

8.22.5.13 rocSPARSE 2.0.0

rocSPARSE 2.0.0 for ROCm 5.0.0

8.22.5.13.1 Added

- csrmv, coomv, ellmv, hybmv for (conjugate) transposed matrices
- csrmv for symmetric matrices

8.22.5.13.2 Changed

- spmm_ex is now deprecated and will be removed in the next major release

8.22.5.13.3 Improved

- Optimization for gtsv

8.22.5.14 rocThrust 2.13.0

rocThrust 2.13.0 for ROCm 5.0.0

8.22.5.14.1 Added

- Updated to match upstream Thrust 1.13.0
- Updated to match upstream Thrust 1.14.0
- Added async scan

8.22.5.14.2 Changed

- Scan algorithms: inclusive_scan now uses the input-type as accumulator-type, exclusive_scan uses initial-value-type.
  - This particularly changes behaviour of small-size input types with large-size output types (e.g. short input, int output).
  - And low-res input with high-res output (e.g. float input, double output)

8.22.5.15 Tensile 4.31.0

Tensile 4.31.0 for ROCm 5.0.0

8.22.5.15.1 Added

- DirectToLds support (x2/x4)
- DirectToVgpr support for DGEMM
- Parameter to control number of files kernels are merged into to better parallelize kernel compilation
- FP16 alternate implementation for HPA HGEMM on aldebaran

8.22.5.15.2 Optimized

- Add DGEMM NN custom kernel for HPL on aldebaran

8.22.5.15.3 Changed

- Update tensile_client executable to std=c++14

8.22.5.15.4 Removed

- Remove unused old Tensile client code

8.22.5.15.5 Fixed

- Fix hipErrorInvalidHandle during benchmarks
- Fix addrVgpr for atomic GSU
- Fix for Python 3.8: add case for Constant nodeType
- Fix architecture mapping for gfx1011 and gfx1012
- Fix PrintSolutionRejectionReason verbiage in KernelWriter.py
- Fix vgpr alignment problem when enabling flat buffer load

CHAPTER

NINE

GPU SUPPORT AND OS COMPATIBILITY (LINUX)

9.1 Supported Linux Distributions

AMD ROCm™ Platform supports the following Linux distributions.

Supported

Distribution	Processor Architectures	Validated Kernel	Support
CentOS 7.9	x86-64	3.10	
RHEL 7.9	x86-64	3.10	
RHEL 8.7	x86-64	4.18	
RHEL 8.8	x86-64	4.18	
RHEL 9.1	x86-64	5.14	
RHEL 9.2	x86-64	5.14	
SLES 15 SP4	x86-64	5.14.21	
SLES 15 SP5	x86-64	5.14.21	
Ubuntu 20.04.5	x86-64	5.15	
Ubuntu 20.04.6	x86-64	5.15	
Ubuntu 22.04.2	x86-64	5.19	
Ubuntu 22.04.3	x86-64	6.2	

Added in version 5.7.0:

- Ubuntu 22.04.3 support was added.

Unsupported

Distribution	Processor Architectures	Validated Kernel	Support
RHEL 9.0	x86-64	5.14	
RHEL 8.6	x86-64	5.14	
SLES 15 SP3	x86-64	5.3	
Ubuntu 22.04.0	x86-64	5.15 LTS, 5.17 OEM	
Ubuntu 20.04.4	x86-64	5.13 HWE, 5.13 OEM	
Ubuntu 22.04.1	x86-64	5.15 LTS	

- : Supported - AMD performs full testing of all ROCm components on distro GA image.

- : Unsupported - AMD no longer performs builds and testing on these previously supported distro GA images.

9.2 Virtualization Support

ROCm supports virtualization for select GPUs only as shown below.

Hypervisor	Version	GPU	Validated Guest OS (validated kernel)
VMWare	ESXi 8	MI250	Ubuntu 20.04 (5.15.0-56-generic)
VMWare	ESXi 8	MI210	Ubuntu 20.04 (5.15.0-56-generic), SLES 15 SP4 (5.14.21-150400.24.18-default)
VMWare	ESXi 7	MI210	Ubuntu 20.04 (5.15.0-56-generic), SLES 15 SP4 (5.14.21-150400.24.18-default)

9.3 Linux Supported GPUs

The table below shows supported GPUs for Instinct™, Radeon Pro™ and Radeon™ GPUs. Please click the tabs below to switch between GPU product lines. If a GPU is not listed on this table, the GPU is not officially supported by AMD.

AMD Instinct™

Use Driver Shipped with ROCm

Product Name	Architecture	LLVM Target	Support
AMD Instinct™ MI250X	CDNA2	gfx90a	
AMD Instinct™ MI250	CDNA2	gfx90a	
AMD Instinct™ MI210	CDNA2	gfx90a	
AMD Instinct™ MI100	CDNA	gfx908	
AMD Instinct™ MI50	GCN5.1	gfx906	
AMD Instinct™ MI25	GCN5.0	gfx900	

Radeon Pro™

Use Radeon Pro Driver

Name	Architecture	LLVM Target	Support
AMD Radeon™ Pro W6800	RDNA2	gfx1030	
AMD Radeon™ Pro V620	RDNA2	gfx1030	
AMD Radeon™ Pro VII	GCN5.1	gfx906	

Radeon™

Use Radeon Pro Driver

Name	Architecture	LLVM Target	Support
AMD Radeon™ VII	GCN5.1	gfx906	

9.3.1 Support Status

- : Supported - AMD enables these GPUs in our software distributions for the corresponding ROCm product.
- : Deprecated - Support will be removed in a future release.
- : Unsupported - This configuration is not enabled in our software distributions.

9.4 CPU Support

ROCm requires CPUs that support PCIe™ Atomics. Modern CPUs after the release of 1st generation AMD Zen CPU and Intel™ Haswell support PCIe Atomics.

CHAPTER

TEN

GPU AND OS SUPPORT (WINDOWS)

10.1 Supported SKUs

AMD ROCm™ Platform supports the following Windows SKU.

Distribution	Processor Architectures	Validated update
Windows 10	x86-64	22H2 (GA)
Windows 11	x86-64	22H2 (GA)
Windows Server 2022	x86-64	

10.2 Windows Supported GPUs

The table below shows supported GPUs for Radeon Pro™ and Radeon™ GPUs. Please click the tabs below to switch between GPU product lines. If a GPU is not listed on this table, the GPU is not officially supported by AMD.

Radeon Pro™

Name	Architecture	LLVM Target	Runtime	HIP SDK
AMD Radeon Pro™ W7900	RDNA3	gfx1100		
AMD Radeon Pro™ W7800	RDNA3	gfx1100		
AMD Radeon Pro™ W6800	RDNA2	gfx1030		
AMD Radeon Pro™ W6600	RDNA2	gfx1032		
AMD Radeon Pro™ W5500	RDNA1	gfx1012		
AMD Radeon Pro™ VII	GCN5.1	gfx906		

Radeon™

Name	Architecture	LLVM Target	Runtime	HIP SDK
AMD Radeon™ RX 7900 XTX	RDNA3	gfx1100		
AMD Radeon™ RX 7900 XT	RDNA3	gfx1100		
AMD Radeon™ RX 7600	RDNA3	gfx1102		
AMD Radeon™ RX 6950 XT	RDNA2	gfx1030		
AMD Radeon™ RX 6900 XT	RDNA2	gfx1030		
AMD Radeon™ RX 6800 XT	RDNA2	gfx1030		
AMD Radeon™ RX 6800	RDNA2	gfx1030		
AMD Radeon™ RX 6750	RDNA2	gfx1032		
AMD Radeon™ RX 6700 XT	RDNA2	gfx1032		
AMD Radeon™ RX 6700	RDNA2	gfx1032		
AMD Radeon™ RX 6650 XT	RDNA2	gfx1032		
AMD Radeon™ RX 6600 XT	RDNA2	gfx1032		
AMD Radeon™ RX 6600	RDNA2	gfx1032		

10.2.1 Component Support

ROCm components are described in the reference page. Support on Windows is provided with two levels on enablement.

- Runtime: Runtime enables the use of the HIP/OpenCL runtimes only.
- HIP SDK: Runtime plus additional components refer to libraries found under Math Libraries and C++ Primitive Libraries. Some Math Libraries are Linux exclusive, please check the library details.

10.2.2 Support Status

- : Supported - AMD enables these GPUs in our software distributions for the corresponding ROCm product.
- : Deprecated - Support will be removed in a future release.
- : Unsupported - This configuration is not enabled in our software distributions.

10.3 CPU Support

ROCm requires CPUs that support PCIe™ Atomics. Modern CPUs after the release of 1st generation AMD Zen CPU and Intel™ Haswell support PCIe Atomics.

CHAPTER

ELEVEN

ROCM RELEASE HISTORY

Version	Release Date
5.7.0	Sep 15, 2023
5.6.1	Aug 29, 2023
5.6.0	Jun 28, 2023
5.5.1	May 24, 2023
5.5.0	May 1, 2023
5.4.3	Feb 7, 2023
5.4.2	Jan 13, 2023
5.4.1	Dec 15, 2022
5.4.0	Nov 30, 2022
5.3.3	Nov 17, 2022
5.3.2	Nov 9, 2022
5.3.0	Oct 4, 2022
5.2.3	Aug 18, 2022
5.2.1	Jul 21, 2022
5.2.0	Jun 28, 2022
5.1.3	May 20, 2022
5.1.1	Apr 8, 2022
5.1.0	Mar 30, 2022
5.0.2	Mar 4, 2022
5.0.1	Feb 16, 2022
5.0.0	Feb 9, 2022

CHAPTER

TWELVE

COMPATIBILITY

User space & Kernel Fusion Driver Forward and backward compatibility of ROCm user space components and the kernel space Kernel Fusion Driver (KFD).

- User/Kernel-Space Support Matrix

Docker Image Support ROCm releases several Docker container images.

- Docker Image Support Matrix

3rd Party Support Several 3rd party libraries ship with ROCm enablement as well as several ROCm components provide interfaces compatible with 3rd party solutions.

- 3rd Party Support Matrix

12.1 User/Kernel-Space Support Matrix

ROCm™ provides forward and backward compatibility between the Kernel Fusion Driver (KFD) and its user space software for +/- 2 releases. This table shows the compatibility combinations that are currently supported.

KFD	Tested user space versions
5.0.2	5.1.0, 5.2.0
5.1.0	5.0.2
5.1.3	5.2.0, 5.3.0
5.2.0	5.0.2, 5.1.3
5.2.3	5.3.0, 5.4.0
5.3.0	5.1.3, 5.2.3
5.3.3	5.4.0, 5.5.0
5.4.0	5.2.3, 5.3.3
5.4.3	5.5.0, 5.6.0
5.4.4	5.5.0
5.5.0	5.3.3, 5.4.3
5.5.1	5.6.0, 5.7.0
5.6.0	5.4.3, 5.5.1
5.6.1	5.7.0
5.7.0	5.5.0, 5.6.1

12.2 Docker Image Support Matrix

The software support matrices for ROCm container releases is listed.

12.2.1 ROCm 5.6

12.2.1.1 PyTorch

12.2.1.1.1 Ubuntu+ rocm5.6_internal_testing +169530b

- ROCm5.6
- Python 3.8
- Torch 2.0.0
- Apex 0.1
- Torchvision 0.15.1
- Tensorboard 2.12.0
- MAGMA
- UCX 1.10.0
- OMPI 4.0.3
- OFED 5.4.3

12.2.1.1.2 CentOS7+ rocm5.6_internal_testing +169530b

- ROCm5.6
- Python 3.8
- Torch 2.0.0
- Apex 0.1
- Torchvision 0.15.1
- Tensorboard 2.12.0
- MAGMA

12.2.1.1.3 1.13 +bfeb431

- ROCm5.6
- Python 3.8
- Torch 1.13.1
- Apex 0.1
- Torchvision 0.14.0
- Tensorboard 2.12.0

- MAGMA
- UCX 1.10.0
- OMPI 4.0.3
- OFED 5.4.3

12.2.1.1.4 1.12 +05d5d04

- ROCm5.6
- Python 3.8
- Torch 1.12.1
- Apex 0.1
- Torchvision 0.13.1
- Tensorboard 2.12.0
- MAGMA
- UCX 1.10.0
- OMPI 4.0.3
- OFED 5.4.3

12.2.1.2 TensorFlow

12.2.1.2.1 tensorflow_develop-upstream-QA-rocm56 +c88a9f4

- ROCm5.6
- Python 3.9
- tensorflow-rocm 2.13.0
- OFED 5.3
- OMPI 4.0.7
- Horovod 0.27.0
- Tensorboard 2.12.0

12.2.1.2.2 r2.11-rocm-enhanced +5be4141

- ROCm5.6
- Python 3.9
- tensorflow-rocm 2.11.0
- OFED 5.3
- OMPI 4.0.7
- Horovod 0.27.0
- Tensorboard 2.11.2

12.2.1.2.3 r2.10-rocm-enhanced +72789a3

- ROCm5.6
- Python 3.9
- tensorflow-rocm 2.10.1
- OFED 5.3
- OMPI 4.0.7
- Horovod 0.27.0
- Tensorboard 2.10.1

12.3 3rd Party Support Matrix

ROCm™ supports various 3rd party libraries and frameworks. Supported versions are tested and known to work. Non-supported versions of 3rd parties may also work, but aren't tested.

12.3.1 Deep Learning

ROCm releases support the most recent and two prior releases of PyTorch and TensorFlow.

ROCm	PyTorch	TensorFlow
5.0.2	1.8, 1.9, 1.10	2.6, 2.7, 2.8
5.1.3	1.9, 1.10, 1.11	2.7, 2.8, 2.9
5.2.x	1.10, 1.11, 1.12	2.8, 2.9, 2.9
5.3.x	1.10.1, 1.11, 1.12.1, 1.13	2.8, 2.9, 2.10
5.4.x	1.10.1, 1.11, 1.12.1, 1.13	2.8, 2.9, 2.10, 2.11
5.5.x	1.10.1, 1.11, 1.12.1, 1.13	2.10, 2.11, 2.13
5.6.x	1.12.1, 1.13, 2.0	2.12, 2.13
5.7.x	1.12.1, 1.13, 2.0	2.12, 2.13

12.3.2 Communication libraries

ROCm supports OpenUCX an "an open-source, production-grade communication framework for data-centric and high-performance applications".

UCX version	ROCm 5.4 and older	ROCm 5.5 and newer
-1.14.0	COMPATIBLE	INCOMPATIBLE
1.14.1+	COMPATIBLE	COMPATIBLE

The Unified Collective Communication Library UCC also has support for ROCm devices.

UCC version	ROCm 5.5 and older	ROCm 5.6 and newer
-1.1.0	COMPATIBLE	INCOMPATIBLE
1.2.0+	COMPATIBLE	COMPATIBLE

12.3.3 Algorithm libraries

ROCm releases provide algorithm libraries with interfaces compatible with contemporary CUDA / NVIDIA HPC SDK alternatives.

- Thrust → rocThrust
- CUB → hipCUB

ROCm	Thrust / CUB	HPC SDK
5.0.2	1.14	21.9
5.1.3	1.15	22.1
5.2.x	1.15	22.2, 22.3
5.3.x	1.16	22.7
5.4.x	1.16	22.9
5.5.x	1.17	22.9
5.6	1.17.2	22.9
5.7	1.17.2	22.9

For the latest documentation of these libraries, refer to the associated documentation.

CHAPTER

THIRTEEN

LICENSING TERMS

ROCm™ is released by Advanced Micro Devices, Inc. and is licensed per component separately. The following table is a list of ROCm components with links to their respective license terms. These components may include third party components subject to additional licenses. Please review individual repositories for more information. The table shows ROCm components, the name of license and link to the license terms.

Component	License
AMDMIGraphX	MIT
HIPCC	MIT
HIPIFY	MIT
HIP	MIT
MIOpenGEMM	MIT
MIOpen	MIT
MIVisionX	MIT
RCP	MIT
ROCK-Kernel-Driver	GPL 2.0 WITH Linux-syscall-note
ROCR-Runtime	The University of Illinois/NCSA
ROCT-Thunk-Interface	MIT
ROCclr	MIT
ROCdbgapi	MIT
ROCgdb	GNU General Public License v2.0
ROCm-CompilerSupport	The University of Illinois/NCSA
ROCm-Device-Libs	The University of Illinois/NCSA
ROCm-OpenCL-Runtime/api/opencl/khronos/icd	Apache 2.0
ROCm-OpenCL-Runtime	MIT
ROCmValidationSuite	MIT
Tensile	MIT
aomp-extras	MIT
aomp	Apache 2.0
atmi	MIT
clang-ocl	MIT
flang	Apache 2.0
half	MIT
hipBLAS	MIT
hipCUB	Custom
hipFFT	MIT
hipSOLVER	MIT
hipSPARSELt	MIT
hipSPARSE	MIT
hipTensor	MIT

continues on next page

Table 13.1 – continued from previous page

Component	License
hipamd	MIT
hipfort	MIT
llvm-project	Apache
rccl	Custom
rdc	MIT
rocALUTION	MIT
rocBLAS	MIT
rocFFT	MIT
rocPRIM	MIT
rocRAND	MIT
rocSOLVER	BSD-2-Clause
rocSPARSE	MIT
rocThrust	Apache 2.0
rocWMMA	MIT
rocm-cmake	MIT
rocm_bandwidth_test	The University of Illinois/NCSA
rocm_smi_lib	The University of Illinois/NCSA
rocminfo	The University of Illinois/NCSA
rocprofiler	MIT
rocr_debug_agent	The University of Illinois/NCSA
roctracer	MIT
rocm-llvm-alt	AMD Proprietary License

Open sourced ROCm components are released via public GitHub repositories, packages on https://repo.radeon.com and other distribution channels. Proprietary products are only available on https://repo.radeon.com. Currently, only one component of ROCm, rocm-llvm-alt is governed by a proprietary license. Proprietary components are organized in a proprietary subdirectory in the package repositories to distinguish from open sourced packages.

The additional terms and conditions below apply to your use of ROCm technical documentation.

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED “AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD Arrow logo, ROCm, and combinations thereof are trademarks of Advanced Micro Devices,

Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

13.1 Package Licensing

Attention: AQL Profiler and AOCC CPU optimization are both provided in binary form, each subject to the license agreement enclosed in the directory for the binary and is available here: /opt/rocm/share/doc/rocm-llvm-alt/EULA. By using, installing, copying or distributing AQL Profiler and/or AOCC CPU Optimizations, you agree to the terms and conditions of this license agreement. If you do not agree to the terms of this agreement, do not install, copy or use the AQL Profiler and/or the AOCC CPU Optimizations.

For the rest of the ROCm packages, you can find the licensing information at the following location: /opt/rocm/share/doc/<component-name>/

For example, you can fetch the licensing information of the _amd_comgr_ component (Code Object Manager) from the amd_comgr folder. A file named LICENSE.txt contains the license details at: /opt/rocm-5.4.3/share/doc/amd_comgr/LICENSE.txt

CHAPTER

FOURTEEN

ALL REFERENCE MATERIAL

14.1 ROCm Software Groups

HIP HIP is both AMD's GPU programming language extension and the GPU runtime.

- HIP
- HIP Examples
- HIPIFY

Math Libraries HIP Math Libraries support the following domains:

- Linear Algebra Libraries
- Fast Fourier Transforms
- Random Numbers

C++ Primitive Libraries ROCm template libraries for C++ primitives and algorithms are as follows:

- rocPRIM
- rocThrust
- hipCUB
- hipTensor

Communication Libraries Inter and intra-node communication is supported by the following projects:

- RCCL

AI Libraries Libraries related to AI.

- MIOpen
- Composable Kernel
- MIGraphX

Computer Vision Computer vision related projects.

- MIVisionX
- rocAL

OpenMP

- OpenMP Support Guide

Compilers and Tools

- ROCmCC
- ROCdbgapi
- ROCgdb
- ROCProfiler
- ROCTracer

Management Tools

- AMD SMI
- ROCm SMI
- ROCm Data Center Tool

Validation Tools

- ROCm Validation Suite
- TransferBench

GPU Architectures

- AMD Instinct MI200
- AMD Instinct MI100

CHAPTER

FIFTEEN

HIP

HIP is both AMD's GPU programming language extension and the GPU runtime. This page introduces the HIP runtime and other HIP libraries and tools.

15.1 HIP Runtime

HIP Runtime The HIP Runtime is used to enable GPU acceleration for all HIP language based products.

- Documentation
- GitHub
- Examples

15.2 Porting tools

HIPIFY HIPIFY assists with porting applications from based on CUDA to the HIP Runtime. Supported CUDA APIs are documented here as well.

- Documentation
- GitHub
- Changelog

CHAPTER

SIXTEEN

MATH LIBRARIES

AMD provides various math domain and support libraries as part of ROCm.

16.1 rocLIB vs. hipLIB

Several libraries are prefixed with either “roc” or “hip”. The rocLIB variants (such as rocRAND, rocBLAS) are tested and optimized for AMD hardware using supported toolchains. The hipLIB variants (such as hipRAND, hipBLAS) are compatibility layers that provide an interface akin to their cuLIB (such as cuRAND, cuBLAS) variants while performing static dispatching of API calls to the appropriate vendor libraries as their back-ends. Due to their static dispatch nature, support for either vendor is decided at compile-time of the hipLIB in question. For dynamic dispatch between vendor implementations, refer to the Orochi library.

Linear Algebra Libraries

- rocBLAS
- hipBLAS
- hipBLASLt
- rocALUTION
- rocWMMA
- rocSOLVER
- hipSOLVER
- rocSPARSE
- hipSPARSE
- hipSPARSELt

Fast Fourier Transforms

- rocFFT
- hipFFT

Random Numbers

- rocRAND
- hipRAND

16.2 Linear Algebra Libraries

ROCm libraries for linear algebra are as follows:

rocBLAS rocBLAS is an AMD GPU optimized library for BLAS (Basic Linear Algebra Subprograms).

- Documentation
- GitHub
- Changelog
- Examples

hipBLAS hipBLAS is a compatibility layer for GPU accelerated BLAS optimized for AMD GPUs via rocBLAS and rocSOLVER. hipBLAS allows for a common interface for other GPU BLAS libraries.

- Documentation
- GitHub
- Changelog

hipBLASLt hipBLASLt is a library that provides general matrix-matrix operations with a flexible API and extends functionalities beyond traditional BLAS library. hipBLASLt is exposed APIs in HIP programming language with an underlying optimized generator as a back-end kernel provider.

- Documentation
- GitHub
- Changelog

rocALUTION rocALUTION is a sparse linear algebra library with focus on exploring fine-grained parallelism on top of AMD's ROCm runtime and toolchains, targeting modern CPU and GPU platforms.

- Documentation
- GitHub
- Changelog

rocWMMA rocWMMA provides an API to break down mixed precision matrix multiply-accumulate (MMA) problems into fragments and distributes these over GPU wavefronts.

- Documentation
- GitHub
- Changelog

rocSOLVER rocSOLVER provides a subset of LAPACK (Linear Algebra Package) functionality on the ROCm platform.

- Documentation
- GitHub
- Changelog

hipSOLVER hipSOLVER is a LAPACK marshalling library supporting both rocSOLVER and cuSOLVER as backends whilst exporting a unified interface.

- Documentation
- GitHub
- Changelog

rocSPARSE rocSPARSE is a library to provide BLAS for sparse computations.

- Documentation
- GitHub
- Changelog

hipSPARSE hipSPARSE is a marshalling library to provide sparse BLAS functionality, supporting both rocSPARSE and cuSPARSE as backends.

- Documentation
- GitHub
- Changelog

hipSPARSELt hipSPARSE is a marshalling library to provide sparse BLAS functionality, supporting both rocSPARSELt and cuSPARSELt as backends.

- Documentation
- GitHub

16.3 Fast Fourier Transforms

ROCm libraries for FFT are as follows:

rocFFT rocFFT is an AMD GPU optimized library for FFT.

- Documentation
- GitHub
- Changelog

hipFFT hipFFT is a compatibility layer for GPU accelerated FFT optimized for AMD GPUs using rocFFT. hipFFT allows for a common interface for other non AMD GPU FFT libraries.

- Documentation
- GitHub
- Changelog

16.4 Random Numbers

rocRAND rocRAND is an AMD GPU optimized library for pseudo-random number generators (PRNG).

- Documentation
- GitHub
- Changelog
- Examples

hipRAND hipRAND is a compatibility layer for GPU accelerated pseudo-random number generation (PRNG) optimized for AMD GPUs using rocRAND. hipRAND allows for a common interface for other non AMD GPU PRNG libraries.

- Documentation

- GitHub
- Changelog

CHAPTER

SEVENTEEN

C++ PRIMITIVE LIBRARIES

ROCm template libraries for algorithms are as follows:

rocPRIM rocPRIM is an AMD GPU optimized template library of algorithm primitives, like transforms, reductions, scans, etc. It also serves as a common back-end for similar libraries found inside ROCm.

- Documentation
- GitHub
- Changelog
- Examples

rocThrust rocThrust is a template library of algorithm primitives with a Thrust-compatible interface. Their CPU back-ends are identical, while the GPU back-end calls into rocPRIM.

- Documentation
- GitHub
- Changelog
- Examples

hipCUB hipCUB is a template library of algorithm primitives with a CUB-compatible interface. It's back-end is rocPRIM.

- Documentation
- GitHub
- Changelog
- Examples

hipTensor hipTensor is AMD's C++ library for accelerating tensor primitives based on the composable kernel library, through general purpose kernel languages, like HIP C++.

- Documentation
- GitHub

CHAPTER

EIGHTEEN

COMMUNICATION LIBRARIES

RCCL RCCL (pronounced “Rickle”) is a stand-alone library of standard collective communication routines for GPUs, implementing all-reduce, all-gather, reduce, broadcast, reduce-scatter, gather, scatter, and all-to-all. The collective operations are implemented using ring and tree algorithms and have been optimized for throughput and latency.

- Documentation
- GitHub
- Changelog
- Examples

CHAPTER

NINETEEN

AI LIBRARIES

MIOpen AMD's library for high performance machine learning primitives.

- Documentation
- GitHub
- Changelog

Composable Kernel Composable Kernel: Performance Portable Programming Model for Machine Learning Tensor Operators

- Documentation
- GitHub
- Changelog

MIGraphX AMD MIGraphX is AMD's graph inference engine that accelerates machine learning model inference.

- Documentation
- GitHub
- Changelog

CHAPTER

TWENTY

COMPUTER VISION

MIVisionX MIVisionX toolkit is a set of comprehensive computer vision and machine intelligence libraries, utilities, and applications bundled into a single toolkit. AMD MIVisionX also delivers a highly optimized open-source implementation of the Khronos OpenVX™ and OpenVX™ Extensions.

- Documentation
- GitHub
- Changelog

rocAL The AMD ROCm Augmentation Library (rocAL) is designed to efficiently decode and process images and videos from a variety of storage formats and modify them through a processing graph programmable by the user. rocAL currently provides C API.

- Documentation

CHAPTER

TWENTYONE

OPENMP SUPPORT IN ROCM

21.1 Introduction

The ROCm™ installation includes an LLVM-based implementation that fully supports the OpenMP 4.5 standard and a subset of OpenMP 5.0, 5.1, and 5.2 standards. Fortran, C/C++ compilers, and corresponding runtime libraries are included. Along with host APIs, the OpenMP compilers support offloading code and data onto GPU devices. This document briefly describes the installation location of the OpenMP toolchain, example usage of device offloading, and usage of rocprof with OpenMP applications. The GPUs supported are the same as those supported by this ROCm release. See the list of supported GPUs in GPU Support and OS Compatibility (Linux).

The ROCm OpenMP compiler is implemented using LLVM compiler technology. openmp-toolchain illustrates the internal steps taken to translate a user's application into an executable that can offload computation to the AMDGPU. The compilation is a two-pass process. Pass 1 compiles the application to generate the CPU code and Pass 2 links the CPU code to the AMDGPU device code.

![OpenMP Toolchain](…/…/data/reference/openmp/openmp_toolchain.svg "OpenMP toolchain" =800x600)

21.1.1 Installation

The OpenMP toolchain is automatically installed as part of the standard ROCm installation and is available under /opt/rocm-{version}/llvm. The sub-directories are:

bin: Compilers (flang and clang) and other binaries.

- examples: The usage section below shows how to compile and run these programs.
- include: Header files.
- lib: Libraries including those required for target offload.
- lib-debug: Debug versions of the above libraries.

21.2 OpenMP: Usage

The example programs can be compiled and run by pointing the environment variable ROCM_PATH to the ROCm install directory.

Example:

```
export ROCM_PATH=/opt/rocm-{version}
cd $ROCM_PATH/share/openmp-extras/examples/openmp/veccopy
sudo make run
```

Note: sudo is required since we are building inside the /opt directory. Alternatively, copy the files to your home directory first.

The above invocation of Make compiles and runs the program. Note the options that are required for target offload from an OpenMP program:

```
-fopenmp --offload-arch=<gpu-arch>
```

Note: The compiler also accepts the alternative offloading notation:

```
-fopenmp -fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa -march=<gpu-arch>
```

Obtain the value of gpu-arch by running the following command:

```
% /opt/rocm-{version}/bin/rocminfo | grep gfx
```

See the complete list of compiler command-line references here.

21.2.1 Using rocprof with OpenMP

The following steps describe a typical workflow for using rocprof with OpenMP code compiled with AOMP:

1. Run rocprof with the program command line:

   ```
   % rocprof <application> <args>
   ```

   This produces a results.csv file in the user's current directory that shows basic stats such as kernel names, grid size, number of registers used, etc. The user can choose to specify the preferred output file name using the o option.

2. Add options for a detailed result:

   ```
   --stats: % rocprof --stats <application> <args>
   ```

   The stats option produces timestamps for the kernels. Look into the output CSV file for the field, DurationNs, which is useful in getting an understanding of the critical kernels in the code.

   Apart from --stats, the option --timestamp on produces a timestamp for the kernels.

3. After learning about the required kernels, the user can take a detailed look at each one of them. rocprof has support for hardware counters: a set of basic and a set of derived ones. See the complete list of counters using options –list-basic and –list-derived. rocprof accepts either a text or an XML file as an input.

For more details on rocprof, refer to the ROCProfilerV1 User Manual.

21.2.2 Using Tracing Options

Prerequisite: When using the --sys-trace option, compile the OpenMP program with:

```
-Wl,-rpath,/opt/rocm-{version}/lib -lamdhip64
```

The following tracing options are widely used to generate useful information:

- --hsa-trace: This option is used to get a JSON output file with the HSA API execution traces and a flat profile in a CSV file.
- --sys-trace: This allows programmers to trace both HIP and HSA calls. Since this option results in loading libamdhip64.so, follow the prerequisite as mentioned above.

A CSV and a JSON file are produced by the above trace options. The CSV file presents the data in a tabular format, and the JSON file can be visualized using Google Chrome at chrome://tracing/ or Perfetto. Navigate to Chrome or Perfetto and load the JSON file to see the timeline of the HSA calls.

For more details on tracing, refer to the ROCProfilerV1 User Manual.

21.2.3 Environment Variables

Environment Variable	Purpose
OMP_NUM_TE	To set the number of teams for kernel launch, which is otherwise chosen by the implementation by default. You can set this number (subject to implementation limits) for performance tuning.
LIBOMPTARGET_KERNEL_	To print useful statistics for device operations. Setting it to 1 and running the program emits the name of every kernel launched, the number of teams and threads used, and the corresponding register usage. Setting it to 2 additionally emits timing information for kernel launches and data transfer operations between the host and the device.
LIBOMPTARGET_INFO	To print informational messages from the device runtime as the program executes. Setting it to a value of 1 or higher, prints fine-grain information and setting it to -1 prints complete information.
LIBOMPTARGET_DEBUG	To get detailed debugging information about data transfer operations and kernel launch when using a debug version of the device library. Set this environment variable to 1 to get the detailed information from the library.
GPU_MAX_HW	To set the number of HSA queues in the OpenMP runtime. The HSA queues are created on demand up to the maximum value as supplied here. The queue creation starts with a single initialized queue to avoid unnecessary allocation of resources. The provided value is capped if it exceeds the recommended, device-specific value.
LIBOMPTARGET_AMDGPU_	To set the threshold size up to which data transfers are initiated asynchronously. The default threshold size is 110241024 bytes (1MB).
OMPX_FORCE_	To force the runtime to execute all operations synchronously, i.e., wait for an operation to complete immediately. This affects data transfers and kernel execution. While it is mainly designed for debugging, it may have a minor positive effect on performance in certain situations.

21.3 OpenMP: Features

The OpenMP programming model is greatly enhanced with the following new features implemented in the past releases.

21.3.1 Asynchronous Behavior in OpenMP Target Regions

- Controlling Asynchronous Behavior

The OpenMP offloading runtime executes in an asynchronous fashion by default, allowing multiple data transfers to start concurrently. However, if the data to be transferred becomes larger than the default threshold of 1MB, the runtime falls back to a synchronous data transfer. The buffers that have been locked already are always executed asynchronously. You can overrule this default behavior by setting LIBOMPTARGET_AMDGPU_MAX_ASYNC_COPY_BYTES and OMPX_FORCE_SYNC_REGIONS. See the Environment Variables table for details.

- Multithreaded Offloading on the Same Device

The libomptarget plugin for GPU offloading allows creation of separate configurable HSA queues per chiplet, which enables two or more threads to concurrently offload to the same device.

- Parallel Memory Copy Invocations

Implicit asynchronous execution of single target region enables parallel memory copy invocations.

21.3.2 Unified Shared Memory

Unified Shared Memory (USM) provides a pointer-based approach to memory management. To implement USM, fulfill the following system requirements along with Xnack capability.

21.3.2.1 Prerequisites

- Linux Kernel versions above 5.14
- Latest KFD driver packaged in ROCm stack
- Xnack, as USM support can only be tested with applications compiled with Xnack capability

21.3.2.2 Xnack Capability

When enabled, Xnack capability allows GPU threads to access CPU (system) memory, allocated with OS-allocators, such as malloc, new, and mmap. Xnack must be enabled both at compile- and run-time. To enable Xnack support at compile-time, use:

```
--offload-arch=gfx908:xnack+
```

Or use another functionally equivalent option Xnack-any:

```
--offload-arch=gfx908
```

To enable Xnack functionality at runtime on a per-application basis, use environment variable:

```
HSA_XNACK=1
```

When Xnack support is not needed:

- Build the applications to maximize resource utilization using:

```
--offload-arch=gfx908:xnack-
```

- At runtime, set the HSA_XNACK environment variable to 0.

21.3.2.3 Unified Shared Memory Pragma

This OpenMP pragma is available on MI200 through xnack+ support.

```
omp requires unified_shared_memory
```

As stated in the OpenMP specifications, this pragma makes the map clause on target constructs optional. By default, on MI200, all memory allocated on the host is fine grain. Using the map clause on a target clause is allowed, which transforms the access semantics of the associated memory to coarse grain.

A simple program demonstrating the use of this feature is:

```
$ cat parallel_for.cpp
#include <stdlib.h>
#include <stdio.h>

#define N 64
#pragma omp requires unified_shared_memory
int main() {
  int n = N;
  int *a = new int[n];
  int *b = new int[n];

  for(int i = 0; i < n; i++)
    b[i] = i;

  #pragma omp target parallel for map(to:b[:n])
  for(int i = 0; i < n; i++)
    a[i] = b[i];

  for(int i = 0; i < n; i++)
    if(a[i] != i)
      printf("error at %d: expected %d, got %d\n", i, i+1, a[i]);

  return 0;
}
$ clang++ -O2 -target x86_64-pc-linux-gnu -fopenmp --offload-arch=gfx90a:xnack+ parallel_for.cpp
$ HSA_XNACK=1 ./a.out
```

In the above code example, pointer “a” is not mapped in the target region, while pointer “b” is. Both are valid pointers on the GPU device and passed by-value to the kernel implementing the target region. This means the pointer values on the host and the device are the same.

The difference between the memory pages pointed to by these two variables is that the pages pointed by “a” are in fine-grain memory, while the pages pointed to by “b” are in coarse-grain memory during and after the execution of the target region. This is accomplished in the OpenMP runtime library with calls to the ROCr runtime to set the pages pointed by “b” as coarse grain.

21.3.3 OMPT Target Support

The OpenMP runtime in ROCm implements a subset of the OMPT device APIs, as described in the OpenMP specification document. These APIs allow first-party tools to examine the profile and kernel traces that execute on a device. A tool can register callbacks for data transfer and kernel dispatch entry points or use APIs to start and stop tracing for device-related activities such as data transfer and kernel dispatch timings and associated metadata. If device tracing is enabled, trace records for device activities are collected during program execution and returned to the tool using the APIs described in the specification.

The following example demonstrates how a tool uses the supported OMPT target APIs. The README in /opt/rocm/llvm/examples/tools/ompt outlines the steps to be followed, and the provided example can be run as shown below:

```
cd $ROCM_PATH/share/openmp-extras/examples/tools/ompt/veccopy-ompt-target-tracing
sudo make run
```

The file veccopy-ompt-target-tracing.c simulates how a tool initiates device activity tracing. The file callbacks.h shows the callbacks registered and implemented by the tool.

21.3.4 Floating Point Atomic Operations

The MI200-series GPUs support the generation of hardware floating-point atomics using the OpenMP atomic pragma. The support includes single- and double-precision floating-point atomic operations. The programmer must ensure that the memory subjected to the atomic operation is in coarse-grain memory by mapping it explicitly with the help of map clauses when not implicitly mapped by the compiler as per the OpenMP specifications. This makes these hardware floating-point atomic instructions "fast," as they are faster than using a default compare-and-swap loop scheme, but at the same time "unsafe," as they are not supported on fine-grain memory. The operation in unified_shared_memory mode also requires programmers to map the memory explicitly when not implicitly mapped by the compiler.

To request fast floating-point atomic instructions at the file level, use compiler flag -munsafe-fp-atomics or a hint clause on a specific pragma:

```
double a = 0.0;
#pragma omp atomic hint(AMD_fast_fp_atomics)
a = a + 1.0;
```

Note: AMD_unsafe_fp_atomics is an alias for AMD_fast_fp_atomics, and AMD_safe_fp_atomics is implemented with a compare-and-swap loop.

To disable the generation of fast floating-point atomic instructions at the file level, build using the option -msafe-fp-atomics or use a hint clause on a specific pragma:

```
double a = 0.0;
#pragma omp atomic hint(AMD_safe_fp_atomics)
a = a + 1.0;
```

The hint clause value always has a precedence over the compiler flag, which allows programmers to create atomic constructs with a different behavior than the rest of the file.

See the example below, where the user builds the program using -msafe-fp-atomics to select a file-wide "safe atomic" compilation. However, the fast atomics hint clause over variable "a" takes precedence and operates on "a" using a fast/unsafe floating-point atomic, while the variable "b" in the absence of a hint clause is operated upon using safe floating-point atomics as per the compiler flag.

```
double a = 0.0;.
#pragma omp atomic hint(AMD_fast_fp_atomics)
a = a + 1.0;

double b = 0.0;
#pragma omp atomic
b = b + 1.0;
```

21.3.5 Address Sanitizer (ASan) Tool

Address Sanitizer is a memory error detector tool utilized by applications to detect various errors ranging from spatial issues such as out-of-bound access to temporal issues such as use-after-free. The AOMP compiler supports ASan for AMD GPUs with applications written in both HIP and OpenMP.

Features Supported on Host Platform (Target x86_64):

- Use-after-free
- Buffer overflows
- Heap buffer overflow
- Stack buffer overflow
- Global buffer overflow
- Use-after-return
- Use-after-scope
- Initialization order bugs

Features Supported on AMDGPU Platform (amdgcn-amd-amdhsa):

- Heap buffer overflow
- Global buffer overflow

Software (Kernel/OS) Requirements: Unified Shared Memory support with Xnack capability. See the section on Unified Shared Memory for prerequisites and details on Xnack.

Example:

- Heap buffer overflow

```
void  main() {
.......  // Some program statements
.......  // Some program statements
#pragma omp target map(to : A[0:N], B[0:N]) map(from: C[0:N])
{
#pragma omp parallel for
    for(int i =0 ; i < N; i++){
    C[i+10] = A[i] + B[i];
  }   // end of for loop
}
.......   // Some program statements
}// end of main
```

See the complete sample code for heap buffer overflow here.

- Global buffer overflow

```
#pragma omp declare target
  int A[N],B[N],C[N];
#pragma omp end declare target
void main(){
.....  // some program statements
.....  // some program statements
#pragma omp target data map(to:A[0:N],B[0:N]) map(from: C[0:N])
{
#pragma omp target update to(A,B)
#pragma omp target parallel for
for(int i=0; i<N; i++){
   C[i]=A[i*100]+B[i+22];
} // end of for loop
#pragma omp target update from(C)
}
.....  // some program statements
} // end of main
```

See the complete sample code for global buffer overflow here.

21.3.6 Clang Compiler Option for Kernel Optimization

You can use the clang compiler option -fopenmp-target-fast for kernel optimization if certain constraints implied by its component options are satisfied. -fopenmp-target-fast enables the following options:

- -fopenmp-target-ignore-env-vars: It enables code generation of specialized kernels including No-loop and Cross-team reductions.
- -fopenmp-assume-no-thread-state: It enables the compiler to assume that no thread in a parallel region modifies an Internal Control Variable (ICV), thus potentially reducing the device runtime code execution.
- -fopenmp-assume-no-nested-parallelism: It enables the compiler to assume that no thread in a parallel region encounters a parallel region, thus potentially reducing the device runtime code execution.
- -O3 if no -O* is specified by the user.

21.3.7 Specialized Kernels

Clang will attempt to generate specialized kernels based on compiler options and OpenMP constructs. The following specialized kernels are supported:

- No-Loop
- Big-Jump-Loop
- Cross-Team (Xteam) Reductions

To enable the generation of specialized kernels, follow these guidelines:

- Do not specify teams, threads, and schedule-related environment variables. The num_teams clause in an OpenMP target construct acts as an override and prevents the generation of the No-Loop kernel. If the specification of num_teams clause is a user requirement then clang tries to generate the Big-Jump-Loop kernel instead of the No-Loop kernel.
- Assert the absence of the teams, threads, and schedule-related environment variables by adding the command-line option -fopenmp-target-ignore-env-vars.

- To automatically enable the specialized kernel generation, use -Ofast or -fopenmp-target-fast for compilation.
- To disable specialized kernel generation, use -fno-openmp-target-ignore-env-vars.

21.3.7.1 No-Loop Kernel Generation

The No-loop kernel generation feature optimizes the compiler performance by generating a specialized kernel for certain OpenMP target constructs such as target teams distribute parallel for. The specialized kernel generation feature assumes every thread executes a single iteration of the user loop, which leads the runtime to launch a total number of GPU threads equal to or greater than the iteration space size of the target region loop. This allows the compiler to generate code for the loop body without an enclosing loop, resulting in reduced control-flow complexity and potentially better performance.

21.3.7.2 Big-Jump-Loop Kernel Generation

A No-Loop kernel is not generated if the OpenMP teams construct uses a num_teams clause. Instead, the compiler attempts to generate a different specialized kernel called the Big-Jump-Loop kernel. The compiler launches the kernel with a grid size determined by the number of teams specified by the OpenMP num_teams clause and the blocksize chosen either by the compiler or specified by the corresponding OpenMP clause.

21.3.7.3 Xteam Optimized Reduction Kernel Generation

If the OpenMP construct has a reduction clause, the compiler attempts to generate optimized code by utilizing efficient Xteam communication. New APIs for Xteam reduction are implemented in the device runtime and are automatically generated by clang.

CHAPTER

TWENTYTWO

COMPILERS AND TOOLS

ROCdbgapi The AMD Debugger API is a library that provides all the support necessary for a debugger and other tools to perform low level control of the execution and inspection of execution state of AMD's commercially available GPU architectures.

- Documentation
- GitHub

ROCmCC ROCmCC is a Clang/LLVM-based compiler. It is optimized for high-performance computing on AMD GPUs and CPUs and supports various heterogeneous programming models such as HIP, OpenMP, and OpenCL.

- Documentation

ROCgdb This is ROCgdb, the ROCm source-level debugger for Linux, based on GDB, the GNU source-level debugger.

- Documentation
- GitHub

ROCProfiler ROC profiler library. Profiling with performance counters and derived metrics. Library supports GFX8/GFX9. Hardware specific low-level performance analysis interface for profiling of GPU compute applications. The profiling includes hardware performance counters with complex performance metrics.

- Documentation
- GitHub

ROCTracer Callback/Activity Library for Performance tracing AMD GPUs

- Documentation
- GitHub

22.1 See Also

- Compiler Disambiguation

22.2 Compiler Reference Guide

22.2.1 Introduction to Compiler Reference Guide

ROCmCC is a Clang/LLVM-based compiler. It is optimized for high-performance computing on AMD GPUs and CPUs and supports various heterogeneous programming models such as HIP, OpenMP, and OpenCL.

ROCmCC is made available via two packages: rocm-llvm and rocm-llvm-alt. The differences are listed in the table below.

Table 22.1: Differences between rocm-llvm and rocm-llvm-alt

rocm-llvm	rocm-llvm-alt
Installed by default when ROCm™ itself is installed	An optional package
Provides an open-source compiler	Provides an additional closed-source compiler for users interested in additional CPU optimizations not available in rocm-llvm

For more details, see:

- AMD GPU usage: llvm.org/docs/AMDGPUUsage.html
- Releases and source: https://github.com/RadeonOpenCompute/llvm-project

22.2.1.1 ROCm Compiler Interfaces

ROCm currently provides two compiler interfaces for compiling HIP programs:

- /opt/rocm/bin/hipcc
- /opt/rocm/bin/amdclang++

Both leverage the same LLVM compiler technology with the AMD GCN GPU support; however, they offer a slightly different user experience. The hipcc command-line interface aims to provide a more familiar user interface to users who are experienced in CUDA but relatively new to the ROCm/HIP development environment. On the other hand, amdclang++ provides a user interface identical to the clang++ compiler. It is more suitable for experienced developers who want to directly interact with the clang compiler and gain full control of their application's build process.

The major differences between hipcc and amdclang++ are listed below:

Table 22.2: Differences between hipcc and amdclang++

*	hipcc	amdclang++
Compiling HIP source files	Treats all source files as HIP language source files	Enables the HIP language support for files with the .hip extension or through the -x hip compiler option
Detecting GPU architecture	Auto-detects the GPUs available on the system and generates code for those devices when no GPU architecture is specified	Has AMD GCN gfx803 as the default GPU architecture. The --offload-arch compiler option may be used to target other GPU architectures
Finding a HIP installation	Finds the HIP installation based on its own location and its knowledge about the ROCm directory structure	First looks for HIP under the same parent directory as its own LLVM directory and then falls back on /opt/rocm. Users can use the --rocm-path option to instruct the compiler to use HIP from the specified ROCm installation.
Linking to the HIP runtime library	Is configured to automatically link to the HIP runtime from the detected HIP installation	Requires the --hip-link flag to be specified to link to the HIP runtime. Alternatively, users can use the -l<dir> -lamdhip64 option to link to a HIP runtime library.
Device function inlining	Inlines all GPU device functions, which provide greater performance and compatibility for codes that contain file scoped or device function scoped ___shared___ variables. However, it may increase compile time.	Relies on inlining heuristics to control inlining. Users experiencing performance or compilation issues with code using file scoped or device function scoped ___shared___ variables could try -mllvm -amdgpu-early-inline-all=true -mllvm -amdgpu-function-calls=false to work around the issue. There are plans to address these issues with future compiler improvements.
Source code location	https://github.com/ROCm-Developer-Tools/HIPCC	https://github.com/RadeonOpenCompute/llvm-project

22.2.2 Compiler Options and Features

This chapter discusses compiler options and features.

22.2.2.1 AMD GPU Compilation

This section outlines commonly used compiler flags for hipcc and amdclang++.

-x hip
: Compiles the source file as a HIP program.

-fopenmp
: Enables the OpenMP support.

-fopenmp-targets=<gpu>
: Enables the OpenMP target offload support of the specified GPU architecture.

  Gpu
  : The GPU architecture. E.g. gfx908.

--gpu-max-threads-per-block=<value>:
: Sets the default limit of threads per block. Also referred to as the launch bounds.

  Value
  : The default maximum amount of threads per block.

-munsafe-fp-atomics
: Enables unsafe floating point atomic instructions (AMDGPU only).

-ffast-math
: Allows aggressive, lossy floating-point optimizations.

-mwavefrontsize64, -mno-wavefrontsize64
: Sets wavefront size to be 64 or 32 on RDNA architectures.

-mcumode
: Switches between CU and WGP modes on RDNA architectures.

--offload-arch=<gpu>
: HIP offloading target ID. May be specified more than once.

  Gpu
  : The a device architecture followed by target ID features delimited by a colon. Each target ID feature is a predefined string followed by a plus or minus sign (e.g. gfx908:xnack+:sramecc-).

-g
: Generates source-level debug information.

-fgpu-rdc, -fno-gpu-rdc
: Generates relocatable device code, also known as separate compilation mode.

22.2.2.2 AMD Optimizations for Zen Architectures

The CPU compiler optimizations described in this chapter originate from the AMD Optimizing C/C++ Compiler (AOCC) compiler. They are available in ROCmCC if the optional rocm-llvm-alt package is installed. The user's interaction with the compiler does not change once rocm-llvm-alt is installed. The user should use the same compiler entry point, provided AMD provides high-performance compiler optimizations for Zen-based processors in AOCC.

For more information, refer to https://www.amd.com/en/developer/aocc.html.

22.2.2.2.1 -famd-opt

Enables a default set of AMD proprietary optimizations for the AMD Zen CPU architectures.

-fno-amd-opt disables the AMD proprietary optimizations.

The -famd-opt flag is useful when a user wants to build with the proprietary optimization compiler and not have to depend on setting any of the other proprietary optimization flags.

Note: -famd-opt can be used in addition to the other proprietary CPU optimization flags. The table of optimizations below implicitly enables the invocation of the AMD proprietary optimizations compiler, whereas the -famd-opt flag requires this to be handled explicitly.

22.2.2.2.2 -fstruct-layout=[1,2,3,4,5,6,7]

Analyzes the whole program to determine if the structures in the code can be peeled and the pointer or integer fields in the structure can be compressed. If feasible, this optimization transforms the code to enable these improvements. This transformation is likely to improve cache utilization and memory bandwidth. It is expected to improve the scalability of programs executed on multiple cores.

This is effective only under -flto, as the whole program analysis is required to perform this optimization. Users can choose different levels of aggressiveness with which this optimization can be applied to the application, with 1 being the least aggressive and 7 being the most aggressive level.

Table 22.3: -fstruct-layout Values and Their Effects

-fstru value	Struc ture peel- ing	Pointer size after selective com- pression of self- referential pointers in structures, wher- ever safe	Type of structure fields eligible for compression	Whether compression performed under safety check
1	En- abled	NA	NA	NA
2	En- abled	32-bit	NA	NA
3	En- abled	16-bit	NA	NA
4	En- abled	32-bit	Integer	Yes
5	En- abled	16-bit	Integer	Yes
6	En- abled	32-bit	64-bit signed int or unsigned int. Users must ensure that the values assigned to 64-bit signed int fields are in range -(2^31 - 1) to +(2^31 - 1) and 64-bit unsigned int fields are in the range 0 to +(2^31 - 1). Otherwise, you may obtain incorrect results.	No. Users must ensure the safety based on the program compiled.
7	En- abled	16-bit	64-bit signed int or unsigned int. Users must ensure that the values assigned to 64-bit signed int fields are in range -(2^31 - 1) to +(2^31 - 1) and 64-bit unsigned int fields are in the range 0 to +(2^31 - 1). Otherwise, you may obtain incorrect results.	No. Users must ensure the safety based on the program compiled.

22.2.2.2.3 -fitodcalls

Promotes indirect-to-direct calls by placing conditional calls. Application or benchmarks that have a small and deterministic set of target functions for function pointers passed as call parameters benefit from this optimization. Indirect-to-direct call promotion transforms the code to use all possible determined targets under runtime checks and falls back to the original code for all the other cases. Runtime checks are introduced by the compiler for each of these possible function pointer targets followed by direct calls to the targets.

This is a link time optimization, which is invoked as -flto -fitodcalls

22.2.2.2.4 -fitodcallsbyclone

Performs value specialization for functions with function pointers passed as an argument. It does this specialization by generating a clone of the function. The cloning of the function happens in the call chain as needed, to allow conversion of indirect function call to direct call.

This complements -fitodcalls optimization and is also a link time optimization, which is invoked as -flto -fitodcallsbyclone.

22.2.2.2.5 -fremap-arrays

Transforms the data layout of a single dimensional array to provide better cache locality. This optimization is effective only under -flto, as the whole program needs to be analyzed to perform this optimization, which can be invoked as -flto -fremap-arrays.

22.2.2.2.6 -finline-aggressive

Enables improved inlining capability through better heuristics. This optimization is more effective when used with -flto, as the whole program analysis is required to perform this optimization, which can be invoked as -flto -finline-aggressive.

22.2.2.2.7 -fnt-store (non-temporal store)

Generates a non-temporal store instruction for array accesses in a loop with a large trip count.

22.2.2.2.8 -fnt-store=aggressive

This is an experimental option to generate non-temporal store instruction for array accesses in a loop, whose iteration count cannot be determined at compile time. In this case, the compiler assumes the iteration count to be huge.

22.2.2.2.9 Optimizations Through Driver -mllvm <options>

The following optimization options must be invoked through driver -mllvm <options>:

22.2.2.2.9.1 -enable-partial-unswitch

Enables partial loop unswitching, which is an enhancement to the existing loop unswitching optimization in LLVM. Partial loop unswitching hoists a condition inside a loop from a path for which the execution condition remains invariant, whereas the original loop unswitching works for a condition that is completely loop invariant. The condition inside the loop gets hoisted out from the invariant path, and the original loop is retained for the path where the condition is variant.

22.2.2.2.9.2 -aggressive-loop-unswitch

Experimental option that enables aggressive loop unswitching heuristic (including -enable-partial-unswitch) based on the usage of the branch conditional values. Loop unswitching leads to code bloat. Code bloat can be minimized if the hoisted condition is executed more often. This heuristic prioritizes the conditions based on the number of times they are used within the loop. The heuristic can be controlled with the following options:

- -unswitch-identical-branches-min-count=<n>
  - Enables unswitching of a loop with respect to a branch conditional value (B), where B appears in at least <n> compares in the loop. This option is enabled with -aggressive-loop-unswitch. The default value is 3.

Usage: -mllvm -aggressive-loop-unswitch -mllvm -unswitch-identical-branches-min-count=<n>

Where, n is a positive integer and lower value of <n> facilitates more unswitching.

- -unswitch-identical-branches-max-count=<n>
  - Enables unswitching of a loop with respect to a branch conditional value (B), where B appears in at most <n> compares in the loop. This option is enabled with -aggressive-loop-unswitch. The default value is 6.

Usage: -mllvm -aggressive-loop-unswitch -mllvm -unswitch-identical-branches-max-count=<n>

Where, n is a positive integer and higher value of <n> facilitates more unswitching.

Note: These options may facilitate more unswitching under some workloads. Since loop-unswitching inherently leads to code bloat, facilitating more unswitching may significantly increase the code size. Hence, it may also lead to longer compilation times.

22.2.2.2.9.3 -enable-strided-vectorization

Enables strided memory vectorization as an enhancement to the interleaved vectorization framework present in LLVM. It enables the effective use of gather and scatter kind of instruction patterns. This flag must be used along with the interleave vectorization flag.

22.2.2.2.9.4 -enable-epilog-vectorization

Enables vectorization of epilog-iterations as an enhancement to existing vectorization framework. This enables generation of an additional epilog vector loop version for the remainder iterations of the original vector loop. The vector size or factor of the original loop should be large enough to allow an effective epilog vectorization of the remaining iterations. This optimization takes place only when the original vector loop is vectorized with a vector width or factor of 16. This vectorization width of 16 may be overwritten by -min-width-epilog-vectorization command-line option.

22.2.2.2.9.5 -enable-redundant-movs

Removes any redundant mov operations including redundant loads from memory and stores to memory. This can be invoked using -Wl,-plugin-opt=-enable-redundant-movs.

22.2.2.2.9.6 -merge-constant

Attempts to promote frequently occurring constants to registers. The aim is to reduce the size of the instruction encoding for instructions using constants and obtain a performance improvement.

22.2.2.2.9.7 -function-specialize

Optimizes the functions with compile time constant formal arguments.

22.2.2.2.9.8 -lv-function-specialization

Generates specialized function versions when the loops inside function are vectorizable and the arguments are not aliased with each other.

22.2.2.2.9.9 -enable-vectorize-compares

Enables vectorization on certain loops with conditional breaks assuming the memory accesses are safely bound within the page boundary.

22.2.2.2.9.10 -inline-recursion=[1,2,3,4]

Enables inlining for recursive functions based on heuristics where the aggressiveness of heuristics increases with the level (1-4). The default level is 2. Higher levels may lead to code bloat due to expansion of recursive functions at call sites.

Table 22.4: -inline-recursion Level and Their Effects

-inline-recursion value	Inline depth of heuristics used to enable inlining for recursive functions
1	1
2	1
3	1
4	10

This is more effective with -flto as the whole program needs to be analyzed to perform this optimization, which can be invoked as -flto -inline-recursion=[1,2,3,4].

22.2.2.2.9.11 -reduce-array-computations=[1,2,3]

Performs array data flow analysis and optimizes the unused array computations.

Table 22.5: -reduce-array-computations Values and Their Effects

-reduce-array-computations value	Array elements eligible for elimination of computations
1	Unused
2	Zero valued
3	Both unused and zero valued

This optimization is effective with -flto as the whole program needs to be analyzed to perform this optimization, which can be invoked as -flto -reduce-array-computations=[1,2,3].

22.2.2.2.9.12 -global-vectorize-slp={true,false}

Vectorizes the straight-line code inside a basic block with data reordering vector operations. This option is set to true by default.

22.2.2.2.9.13 -region-vectorize

Experimental flag for enabling vectorization on certain loops with complex control flow, which the normal vectorizer cannot handle.

This optimization is effective with -flto as the whole program needs to be analyzed to perform this optimization, which can be invoked as -flto -region-vectorize.

22.2.2.2.9.14 -enable-x86-prefetching

Enables the generation of x86 prefetch instruction for the memory references inside a loop or inside an innermost loop of a loop nest to prefetch the second dimension of multidimensional array/memory references in the innermost loop of a loop nest. This is an experimental pass; its profitability is being improved.

22.2.2.2.9.15 -suppress-fmas

Identifies the reduction patterns on FMA and suppresses the FMA generation, as it is not profitable on the reduction patterns.

22.2.2.2.9.16 -enable-icm-vrp

Enables estimation of the virtual register pressure before performing loop invariant code motion. This estimation is used to control the number of loop invariants that will be hoisted during the loop invariant code motion.

22.2.2.2.9.17 -loop-splitting

Enables splitting of loops into multiple loops to eliminate the branches, which compare the loop induction with an invariant or constant expression. This option is enabled under -O3 by default. To disable this optimization, use -loop-splitting=false.

22.2.2.2.9.18 -enable-ipo-loop-split

Enables splitting of loops into multiple loops to eliminate the branches, which compares the loop induction with a constant expression. This constant expression can be derived through inter-procedural analysis. This option is enabled under -O3 by default. To disable this optimization, use -enable-ipo-loop-split=false.

22.2.2.2.9.19 -compute-interchange-order

Enables heuristic for finding the best possible interchange order for a loop nest. To enable this option, use -enable-loopinterchange. This option is set to false by default.

Usage:

```
-mllvm -enable-loopinterchange -mllvm -compute-interchange-order
```

22.2.2.2.9.20 -convert-pow-exp-to-int={true,false}

Converts the call to floating point exponent version of pow to its integer exponent version if the floating-point exponent can be converted to integer. This option is set to true by default.

22.2.2.2.9.21 -do-lock-reordering={none,normal,aggressive}

Reorders the control predicates in increasing order of complexity from outer predicate to inner when it is safe. The normal mode reorders simple expressions, while the aggressive mode reorders predicates involving function calls if no side effects are determined. This option is set to normal by default.

22.2.2.2.9.22 -fuse-tile-inner-loop

Enables fusion of adjacent tiled loops as a part of loop tiling transformation. This option is set to false by default.

22.2.2.2.9.23 -Hz,1,0x1 [Fortran]

Helps to preserve array index information for array access expressions which get linearized in the compiler front end. The preserved information is used by the compiler optimization phase in performing optimizations such as loop transformations. It is recommended that any user who is using optimizations such as loop transformations and other optimizations requiring de-linearized index expressions should use the Hz option. This option has no impact on any other aspects of the Flang front end.

22.2.2.3 Inline ASM Statements

Inline assembly (ASM) statements allow a developer to include assembly instructions directly in either host or device code. While the ROCm compiler supports ASM statements, their use is not recommended for the following reasons:

- The compiler's ability to produce both correct code and to optimize surrounding code is impeded.
- The compiler does not parse the content of the ASM statements and so cannot "see" its contents.
- The compiler must make conservative assumptions in an effort to retain correctness.
- The conservative assumptions may yield code that, on the whole, is less performant compared to code without ASM statements. It is possible that a syntactically correct ASM statement may cause incorrect runtime behavior.
- ASM statements are often ASIC-specific; code containing them is less portable and adds a maintenance burden to the developer if different ASICs are targeted.

- Writing correct ASM statements is often difficult; we strongly recommend thorough testing of any use of ASM statements.

Note: For developers who choose to include ASM statements in the code, AMD is interested in understanding the use case and appreciates feedback at https://github.com/RadeonOpenCompute/ROCm/issues

---

22.2.2.4 Miscellaneous OpenMP Compiler Features

This section discusses features that have been added or enhanced in the OpenMP compiler.

22.2.2.4.1 Offload-arch Tool

An LLVM library and tool that is used to query the execution capability of the current system as well as to query requirements of a binary file. It is used by OpenMP device runtime to ensure compatibility of an image with the current system while loading it. It is compatible with target ID support and multi-image fat binary support.

Usage:

```
offload-arch [Options] [Optional lookup-value]
```

When used without an option, offload-arch prints the value of the first offload arch found in the underlying system. This can be used by various clang front ends. For example, to compile for OpenMP offloading on your current system, invoke clang with the following command:

```
clang -fopenmp -fopenmp-targets=`offload-arch` foo.c
```

If an optional lookup-value is specified, offload-arch will check if the value is either a valid offload-arch or a codename and look up requested additional information.

The following command provides all the information for offload-arch gfx906:

```
offload-arch gfx906 -v
```

The options are listed below:

-a
: Prints values for all devices. Do not stop at the first device found.

-m
: Prints device code name (often found in pci.ids file).

-n
: Prints numeric pci-id.

-t
: Prints clang offload triple to use for the offload arch.

-v
: Verbose. Implies: -a -m -n -t. For: all devices, prints codename, numeric value, and triple.

-f <file>
: Prints offload requirements including offload-arch for each compiled offload image built into an application binary file.

-c

Prints offload capabilities of the underlying system. This option is used by the language runtime to select an image when multiple images are available. A capability must exist for each requirement of the selected image.

There are symbolic link aliases amdgpu-offload-arch and nvidia-arch for offload-arch. These aliases return 1 if no AMD GCN GPU or CUDA GPU is found. These aliases are useful in determining whether architecture-specific tests should be run or to conditionally load architecture-specific software.

22.2.2.4.2 Command-Line Simplification Using offload-arch Flag

Legacy mechanism of specifying offloading target for OpenMP involves using three flags, -fopenmp-targets, -Xopenmp-target, and -march. The first two flags take a target triple (like amdgcn-amd-amdhsa or nvptx64-nvidia-cuda), while the last flag takes device name (like gfx908 or sm_70) as input. Alternatively, users of ROCmCC compiler can use the flag —offload-arch for a combined effect of the above three flags.

Example:

```
# Legacy mechanism
clang -fopenmp -target x86_64-linux-gnu \
-fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa \
-march=gfx906 helloworld.c -o helloworld
```

Example:

```
# Using offload-arch flag
clang -fopenmp -target x86_64-linux-gnu \
--offload-arch=gfx906 helloworld.c -o helloworld.
```

To ensure backward compatibility, both styles are supported. This option is compatible with target ID support and multi-image fat binaries.

22.2.2.4.3 Target ID Support for OpenMP

The ROCmCC compiler supports specification of target features along with the GPU name while specifying a target offload device in the command line, using -march or --offload-arch options. The compiled image in such cases is specialized for a given configuration of device and target features (target ID).

Example:

```
# compiling for a gfx908 device with XNACK paging support turned ON
clang -fopenmp -target x86_64-linux-gnu \
-fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa \
-march=gfx908:xnack+ helloworld.c -o helloworld
```

Example:

```
# compiling for a gfx908 device with SRAMECC support turned OFF
clang -fopenmp -target x86_64-linux-gnu \
-fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa \
-march=gfx908:sramecc- helloworld.c -o helloworld
```

Example:

```
# compiling for a gfx908 device with SRAMECC support turned ON and XNACK paging support turned OFF
clang -fopenmp -target x86_64-linux-gnu \
-fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa \
-march=gfx908:sramecc+:xnack- helloworld.c -o helloworld
```

The target ID specified on the command line is passed to the clang driver using target-feature flag, to the LLVM optimizer and back end using -mattr flag, and to linker using -plugin-opt=-mattr flag. This feature is compatible with offload-arch command-line option and multi-image binaries for multiple architectures.

22.2.2.4.4 Multi-image Fat Binary for OpenMP

The ROCmCC compiler is enhanced to generate binaries that can contain heterogenous images. This heterogeneity could be in terms of:

- Images of different architectures, like AMD GCN and NVPTX
- Images of same architectures but for different GPUs, like gfx906 and gfx908
- Images of same architecture and same GPU but for different target features, like gfx908:xnack+ and gfx908:xnack-

An appropriate image is selected by the OpenMP device runtime for execution depending on the capability of the current system. This feature is compatible with target ID support and offload-arch command-line options and uses offload-arch tool to determine capability of the current system.

Example:

```
clang -fopenmp -target x86_64-linux-gnu \
-fopenmp-targets=amdgcn-amd-amdhsa,amdgcn-amd-amdhsa \
-Xopenmp-target=amdgcn-amd-amdhsa -march=gfx906 \
-Xopenmp-target=amdgcn-amd-amdhsa -march=gfx908 \
helloworld.c -o helloworld
```

Example:

```
clang -fopenmp -target x86_64-linux-gnu \
--offload-arch=gfx906 \
--offload-arch=gfx908 \
helloworld.c -o helloworld
```

Example:

```
clang -fopenmp -target x86_64-linux-gnu \
-fopenmp-targets=amdgcn-amd-amdhsa,amdgcn-amd-amdhsa,amdgcn-amd-amdhsa,amdgcn-amd-amdhsa \
-Xopenmp-target=amdgcn-amd-amdhsa -march=gfx908:sramecc+:xnack+ \
-Xopenmp-target=amdgcn-amd-amdhsa -march=gfx908:sramecc-:xnack+ \
-Xopenmp-target=amdgcn-amd-amdhsa -march=gfx908:sramecc+:xnack- \
-Xopenmp-target=amdgcn-amd-amdhsa -march=gfx908:sramecc-:xnack- \
helloworld.c -o helloworld
```

The ROCmCC compiler creates an instance of toolchain for each unique combination of target triple and the target GPU (along with the associated target features). clang-offload-wrapper tool is modified to insert a new structure ___tgt_image_info along with each image in the binary. Device runtime is also modified to query this structure to identify a compatible image based on the capability of the current system.

22.2.2.4.5 Unified Shared Memory (USM)

The following OpenMP pragma is available on MI200, and it must be executed with xnack+ support.

```
omp requires unified_shared_memory
```

For more details on USM refer to the Asynchronous Behavior in OpenMP Target Regions section of the OpenMP Guide.

22.2.2.5 Support Status of Other Clang Options

The following table lists the other Clang options and their support status.

Option	Support Status	Description
-###	Supported	Prints (but does not r
--analyzer-output <value>	Supported	"Static analyzer repor
--analyze	Supported	Runs the static analyz
-arcmt-migrate-emit-errors	Unsupported	Emits ARC errors eve
-arcmt-migrate-report-output <value>	Unsupported	Output path for the p
-byteswapio	Supported	Swaps byte-order for u
-B <dir>	Supported	Adds <dir> to search
-CC	Supported	Includes comments fro
-cl-denorms-are-zero	Supported	OpenCL only. Allows
-cl-fast-relaxed-math	Supported	OpenCL only. Sets -cl
-cl-finite-math-only	Supported	OpenCL only. Allows
-cl-fp32-correctly-rounded-divide-sqrt	Supported	OpenCL only. Specifie
-cl-kernel-arg-info	Supported	OpenCL only. Genera
-cl-mad-enable	Supported	OpenCL only. Allows
-cl-no-signed-zeros	Supported	OpenCL only. Allows
-cl-opt-disable	Supported	OpenCL only. Disable
-cl-single-precision-constant	Supported	OpenCL only. Treats
-cl-std= <value>	Supported	OpenCL language sta
-cl-strict-aliasing	Supported	OpenCL only. This op
-cl-uniform-work-group-size	Supported	OpenCL only. Defines
-cl-unsafe-math-optimizations	Supported	OpenCL only. Allows
--config <value>	Supported	Specifies configuration
--cuda-compile-host-device	Supported	Compiles CUDA code
--cuda-device-only	Supported	Compiles CUDA code
--cuda-host-only	Supported	Compiles CUDA code
--cuda-include-ptx=<value>	Unsupported	Includes PTX for the
--cuda-noopt-device-debug	Unsupported	Enables device-side de
--cuda-path-ignore-env	Unsupported	Ignores environment v
--cuda-path=<value>	Unsupported	CUDA installation pa
-cxx-isystem <directory>	Supported	Adds a directory to th
-C	Supported	Includes comments in
-c	Supported	Runs only preprocess,
-dD	Supported	Prints macro definitio
-dependency-dot <value>	Supported	Writes DOT-formatte
-dependency-file <value>	Supported	Writes dependency ou
-dI	Supported	Prints include directiv

Option	Support Status	Description
-dM	Supported	Prints macro definitio
-dsym-dir <dir>	Unsupported	Outputs dSYMs (if ar
-D <macro>	Supported	=<value>. Defines <
-emit-ast	Supported	Emits Clang AST files
-emit-interface-stubs	Supported	Generates interface st
-emit-llvm	Supported	Uses the LLVM repres
-emit-merged-ifs	Supported	Generates interface st
--emit-static-lib	Supported	Enables linker job to c
-enable-trivial-auto-var-init-zero-knowing-it-will-be-removed-from-clang	Supported	Declares enabling triv
-E	Supported	Runs the preprocessor
-fAAPCSBitfieldLoad	Unsupported	Follows the AAPCS st
-faddrsig	Supported	Emits an address-sign
-faligned-allocation	Supported	Enables C++17 aligne
-fallow-editor-placeholders	Supported	Treats editor placehol
-fallow-fortran-gnu-ext	Supported	Allows Fortran GNU c
-fansi-escape-codes	Supported	Uses ANSI escape cod
-fapple-kext	Unsupported	Uses Apple's kernel ex
-fapple-link-rtlib	Unsupported	Forces linking of the c
-fapple-pragma-pack	Unsupported	Enables Apple gcc-cor
-fapplication-extension	Unsupported	Restricts code to thos
-fbackslash	Supported	Treats backslash as C-
-fbasic-block-sections= <value>	Supported	"Places each function"
-fblocks	Supported	Enables the 'blocks' la
-fborland-extensions	Unsupported	Accepts non-standard
-fbuild-session-file= <file>	Supported	Uses the last modifica
-fbuild-session-timestamp= <time since Epoch in seconds>	Supported	Specifies starting time
-fbuiltin-module-map	Unsupported	Loads the Clang built
-fcall-saved-x10	Unsupported	Makes the x10 registe
-fcall-saved-x11	Unsupported	Makes the x11 registe
-fcall-saved-x12	Unsupported	Makes the x12 registe
-fcall-saved-x13	Unsupported	Makes the x13 registe
-fcall-saved-x14	Unsupported	Makes the x14 registe
-fcall-saved-x15	Unsupported	Makes the x15 registe
-fcall-saved-x18	Unsupported	Makes the x18 registe
-fcall-saved-x8	Unsupported	Makes the x8 register
-fcall-saved-x9	Unsupported	Makes the x9 register
-fcf-protection= <value>	Unsupported	Specifies the instrume
-fcf-protection	Unsupported	Enables cf-protection
-fchar8_t	Supported	Enables C++ built-in
-fclang-abi-compat= <version>	Supported	Attempts to match th
-fcolor-diagnostics	Supported	Enables colors in diag
-fcomment-block-commands= <arg>	Supported	Treats each comma-se
-fcommon	Supported	Places uninitialized gl
-fcomplete-member-pointers	Supported	Requires member poin
-fconvergent-functions	Supported	Assumes functions to
-fcoroutines-ts	Supported	Enables support for th
-fcoverage-mapping	Unsupported	Generates coverage m
-fcs-profile-generate= <directory>	Unsupported	Generates instrumente
-fcs-profile-generate	Unsupported	Generates instrumente
-fcuda-approx-transcendentals	Unsupported	Uses approximate tran

Option	Support Status	Description
-fcuda-flush-denormals-to-zero	Supported	Flushes denormal floa
-fcuda-short-ptr	Unsupported	Uses 32-bit pointers fo
-fcxx-exceptions	Supported	Enables C++ exceptio
-fdata-sections	Supported	Places each data in its
-fdebug-compilation-dir <value>	Supported	Specifies the compilat
-fdebug-default-version= <value>	Supported	Specifies the default D
-fdebug-info-for-profiling	Supported	Emits extra debug inf
-fdebug-macro	Supported	Emits macro debug in
-fdebug-prefix-map= <value>	Supported	Remaps file source pa
-fdebug-ranges-base-address	Supported	Uses DWARF base ad
-fdebug-types-section	Supported	Places debug types in
-fdeclspec	Supported	Allows __declspec as
-fdelayed-template-parsing	Supported	Parses templated func
-fdelete-null-pointer-checks	Supported	Treats usage of null p
-fdiagnostics-absolute-paths	Supported	Prints absolute paths
-fdiagnostics-hotness-threshold= <number>	Unsupported	Prevents optimization
-fdiagnostics-parseable-fixits	Supported	Prints fix-its in machi
-fdiagnostics-print-source-range-info	Supported	Prints source range sp
-fdiagnostics-show-hotness	Unsupported	Enables profile hotnes
-fdiagnostics-show-note-include-stack	Supported	Displays include stack
-fdiagnostics-show-option	Supported	Prints option name wi
-fdiagnostics-show-template-tree	Supported	Prints a template com
-fdigraphs	Supported	Enables alternative to
-fdiscard-value-names	Supported	Discards value names
-fdollars-in-identifiers	Supported	Allows “$” in identifie
-fdouble-square-bracket-attributes	Supported	Enables ‘[[]]’ attribute
-fdwarf-exceptions	Unsupported	Uses DWARF style ex
-feliminate-unused-debug-types	Supported	Eliminates debug info
-fembed-bitcode-marker	Supported	Embeds placeholder L
-fembed-bitcode= <option>	Supported	Embeds LLVM bitcod
-fembed-bitcode	Supported	Embeds LLVM IR bit
-femit-all-decls	Supported	Emits all declarations
-femulated-tls	Supported	Uses emutls functions
-fenable-matrix	Supported	Enables matrix data t
-fexceptions	Supported	Enables support for ex
-fexperimental-new-constant-interpreter	Supported	Enables the experimen
-fexperimental-new-pass-manager	Supported	Enables an experimen
-fexperimental-relative-c++±abi-vtables	Supported	Uses the experimental
-fexperimental-strict-floating-point	Supported	Enables experimental
-ffast-math	Supported	Allows aggressive, loss
-ffile-prefix-map= <value>	Supported	Remaps file source pa
-ffine-grained-bitfield-accesses	Supported	Uses separate accesses
-ffixed-form	Supported	Enables fixed-form for
-ffixed-point	Supported	Enables fixed point ty
-ffixed-r19	Unsupported	Reserves the r19 regis
-ffixed-r9	Unsupported	Reserves the r9 registe
-ffixed-x10	Unsupported	Reserves the x10 regis
-ffixed-x11	Unsupported	Reserves the x11 regis
-ffixed-x12	Unsupported	Reserves the x12 regis
-ffixed-x13	Unsupported	Reserves the x13 regis

Option	Support Status	Description
-ffixed-x14	Unsupported	Reserves the x14 regis
-ffixed-x15	Unsupported	Reserves the x15 regis
-ffixed-x16	Unsupported	Reserves the x16 regis
-ffixed-x17	Unsupported	Reserves the x17 regis
-ffixed-x18	Unsupported	Reserves the x18 regis
-ffixed-x19	Unsupported	Reserves the x19 regis
-ffixed-x1	Unsupported	Reserves the x1 regist
-ffixed-x20	Unsupported	Reserves the x20 regis
-ffixed-x21	Unsupported	Reserves the x21 regis
-ffixed-x22	Unsupported	Reserves the x22 regis
-ffixed-x23	Unsupported	Reserves the x23 regis
-ffixed-x24	Unsupported	Reserves the x24 regis
-ffixed-x25	Unsupported	Reserves the x25 regis
-ffixed-x26	Unsupported	Reserves the x26 regis
-ffixed-x27	Unsupported	Reserves the x27 regis
-ffixed-x28	Unsupported	Reserves the x28 regis
-ffixed-x29	Unsupported	Reserves the x29 regis
-ffixed-x2	Unsupported	Reserves the x2 regist
-ffixed-x30	Unsupported	Reserves the x30 regis
-ffixed-x31	Unsupported	Reserves the x31 regis
-ffixed-x3	Unsupported	Reserves the x3 regist
-ffixed-x4	Unsupported	Reserves the x4 regist
-ffixed-x5	Unsupported	Reserves the x5 regist
-ffixed-x6	Unsupported	Reserves the x6 regist
-ffixed-x7	Unsupported	Reserves the x7 regist
-ffixed-x8	Unsupported	Reserves the x8 regist
-ffixed-x9	Unsupported	Reserves the x9 regist
-fforce-dwarf-frame	Supported	Mandatorily emits a d
-fforce-emit-vtables	Supported	Emits more virtual ta
-fforce-enable-int128	Supported	Enables support for in
-ffp-contract= <value>	Supported	Forms fused FP ops (e
-ffp-exception-behavior= <value>	Supported	Specifies the exception
-ffp-model= <value>	Supported	Controls the semantic
-ffree-form	Supported	Enables free-form form
-ffreestanding	Supported	Asserts the compilatio
-ffunc-args-alias	Supported	Allows the function ar
-ffunction-sections	Supported	Places each function i
-fglobal-isel	Supported	Enables the global ins
-fgnu-keywords	Supported	Allows GNU-extension
-fgnu-runtime	Unsupported	Generates output com
-fgnu89-inline	Unsupported	Uses the gnu89 inline
-fgnuc-version= <value>	Supported	Sets various macros to
-fgpu-allow-device-init	Supported	Allows device-side init
-fgpu-rdc	Supported	Generates relocatable
-fhip-new-launch-api	Supported	Uses new kernel launc
-fignore-exceptions	Supported	Enables support for ig
-fimplicit-module-maps	Unsupported	Implicitly searches the
-finline-functions	Supported	Inlines suitable functi
-finline-hint-functions	Supported	Inlines functions that
-finstrument-function-entry-bare	Unsupported	Allows instrument fun

Option	Support Status	Description
-finstrument-functions-after-inlining	Unsupported	Similar to -finstrumen
-finstrument-functions	Unsupported	Generates calls to inst
-fintegrated-as	Supported	Enables the integrate
-fintegrated-cc1	Supported	Runs cc1 in-process
-fjump-tables	Supported	Uses jump tables for l
-fkeep-static-consts	Supported	Keeps static const var
-flax-vector-conversions= <value>	Supported	Enables implicit vecto
-flto-jobs= <value>	Unsupported	Controls the backend
-flto= <value>	Unsupported	Sets LTO mode to eit
-flto	Unsupported	Enables LTO in “full”
-fmacro-prefix-map= <value>	Supported	Remaps file source pa
-fmath-errno	Supported	Requires math functio
-fmax-tokens= <value>	Supported	Specifies max total nu
-fmax-type-align= <value>	Supported	Specifies the maximur
-fmemory-profile	Supported	Enables heap memory
-fmerge-all-constants	Supported	Allows merging of con
-fmessage-length= <value>	Supported	Formats message diag
-fmodule-file=[<name>=] <file>	Unsupported	Specifies the mapping
-fmodule-map-file= <file>	Unsupported	Loads the specified m
-fmodule-name= <name>	Unsupported	Specifies the name of
-fmodules-cache-path= <directory>	Unsupported	Specifies the module c
-fmodules-decluse	Unsupported	Asserts declaration of
-fmodules-disable-diagnostic-validation	Unsupported	Disables validation of
-fmodules-ignore-macro= <value>	Unsupported	Ignores the definition
-fmodules-prune-after= <seconds>	Unsupported	Specifies the interval (
-fmodules-prune-interval= <seconds>	Unsupported	Specifies the interval (
-fmodules-search-all	Unsupported	Searches even non-imp
-fmodules-strict-decluse	Unsupported	Similar to -fmodules-c
-fmodules-ts	Unsupported	Enables support for th
-fmodules-user-build-path <directory>	Unsupported	Specifies the module u
-fmodules-validate-input-files-content	Supported	Validates PCM input
-fmodules-validate-once-per-build-session	Unsupported	Prohibits verification
-fmodules-validate-system-headers	Supported	Validates the system h
-fmodules	Unsupported	Enables the “modules
-fms-compatibility-version= <value>	Supported	Specifies the dot-separ
-fms-compatibility	Supported	Enables full Microsoft
-fms-extensions	Supported	Accepts some non-stan
-fmsc-version= <value>	Supported	Specifies the Microsof
-fnew-alignment= <align>	Supported	Specifies the largest al
-fno-addrsig	Supported	Prohibits emitting an
-fno-allow-fortran-gnu-ext	Supported	Allows Fortran GNU e
-fno-assume-sane-operator-new	Supported	Prohibits the assumpt
-fno-autolink	Supported	Disables generation of
-fno-backslash	Supported	Allows treatment of b
-fno-builtin- <value>	Supported	Disables implicit built
-fno-builtin	Supported	Disables implicit built
-fno-c++±static-destructors	Supported	Disables C++ static d
-fno-char8_t	Supported	Disables C++ built-in
-fno-color-diagnostics	Supported	Disables colors in diag
-fno-common	Supported	Compiles common glo

Option	Support Status	Description
-fno-complete-member-pointers	Supported	Eliminates the require
-fno-constant-cfstrings	Supported	Disables creation of C
-fno-coverage-mapping	Supported	Disables code coverag
-fno-crash-diagnostics	Supported	Disables auto-generati
-fno-cuda-approx-transcendentals	Unsupported	Eliminates the usage o
-fno-debug-macro	Supported	Prohibits emitting the
-fno-declspec	Unsupported	Disallows declspec as
-fno-delayed-template-parsing	Supported	Disables delayed temp
-fno-delete-null-pointer-checks	Supported	Prohibits the treatme
-fno-diagnostics-fixit-info	Supported	Prohibits including fix
-fno-digraphs	Supported	Disallows alternative t
-fno-discard-value-names	Supported	Prohibits discarding v
-fno-dollars-in-identifiers	Supported	Disallows '$' in identi
-fno-double-square-bracket-attributes	Supported	Disables '[[]]' attribute
-fno-elide-constructors	Supported	Disables C++ copy co
-fno-elide-type	Supported	Prohibits eliding types
-fno-eliminate-unused-debug-types	Supported	Emits debug info for c
-fno-exceptions	Supported	Disables support for e
-fno-experimental-new-pass-manager	Supported	Disables an experimen
-fno-experimental-relative-c++±abi-vtables	Supported	Prohibits using the ex
-fno-fine-grained-bitfield-accesses	Supported	Allows using large-int
-fno-fixed-form	Supported	Disables fixed-form fo
-fno-fixed-point	Supported	Disables fixed point ty
-fno-force-enable-int128	Supported	Disables support for i
-fno-fortran-main	Supported	Prohibits linking in Fo
-fno-free-form	Supported	Disables free-form for
-fno-func-args-alias	Supported	Allows the function ar
-fno-global-isel	Supported	Disables the global ins
-fno-gnu-inline-asm	Supported	Disables GNU style in
-fno-gpu-allow-device-init	Supported	Disallows device-side i
-fno-hip-new-launch-api	Supported	Disallows new kernel l
-fno-integrated-as	Supported	Disables the integrate
-fno-integrated-cc1	Supported	Spawns a separate pro
-fno-jump-tables	Supported	Disallows jump tables
-fno-keep-static-consts	Supported	Prohibits keeping stat
-fno-lto	Supported	Disables LTO mode (c
-fno-memory-profile	Supported	Disables heap memory
-fno-merge-all-constants	Supported	Disallows merging of c
-fno-no-access-control	Supported	Disables C++ access c
-fno-objc-infer-related-result-type	Supported	Prohibits inferring Ob
-fno-operator-names	Supported	Disallows treatment o
-fno-pch-codegen	Supported	Disallows code-genera
-fno-pch-debuginfo	Supported	Prohibits generation c
-fno-plt	Supported	Asserts usage of GOT
-fno-preserve-as-comments	Supported	Prohibits preserving c
-fno-profile-generate	Supported	Disables generation of
-fno-profile-instr-generate	Supported	Disables generation of
-fno-profile-instr-use	Supported	Disables usage of instr
-fno-register-global-dtors-with-atexit	Supported	Disallows usage of ate
-fno-rtlib-add-rpath	Supported	Prohibits adding -rpat

Option	Support Status	Description
-fno-rtti-data	Supported	Disables generation of
-fno-rtti	Supported	Disables generation of
-fno-sanitize-address-poison-custom-array-cookie	Supported on Host only	Disables poisoning of
-fno-sanitize-address-use-after-scope	Supported on Host only	Disables use-after-scop
-fno-sanitize-address-use-odr-indicator	Supported on Host only	Disables ODR indicat
-fno-sanitize-blacklist	Supported on Host only	Prohibits using blackl
-fno-sanitize-cfi-canonical-jump-tables	Supported on Host only	Prohibits making the
-fno-sanitize-cfi-cross-dso	Supported on Host only	Disables control flow i
-fno-sanitize-coverage= <value>	Supported on Host only	Disables specified feat
-fno-sanitize-memory-track-origins	Supported on Host only	Disables origins tracki
-fno-sanitize-memory-use-after-dtor	Supported on Host only	Disables use-after-dest
-fno-sanitize-recover= <value>	Supported on Host only	Disables recovery for s
-fno-sanitize-stats	Supported on Host only	Disables sanitizer stat
-fno-sanitize-thread-atomics	Supported on Host only	Disables atomic opera
-fno-sanitize-thread-func-entry-exit	Supported on Host only	Disables function entr
-fno-sanitize-thread-memory-access	Supported on Host only	Disables memory acce
-fno-sanitize-trap= <value>	Supported on Host only	Disables trapping for
-fno-sanitize-trap	Supported on Host only	Disables trapping for
-fno-short-wchar	Supported	Forces wchar_t to be
-fno-show-column	Supported	Prohibits including co
-fno-show-source-location	Supported	Prohibits including so
-fno-signed-char	Supported	char is unsigned
-fno-signed-zeros	Supported	Allows optimizations
-fno-spell-checking	Supported	Disables spell-check
-fno-split-machine-functions	Supported	Disables late function
-fno-stack-clash-protection	Supported	Disables stack clash p
-fno-stack-protector	Supported	Disables the use of sta
-fno-standalone-debug	Supported	Limits debug informat
-fno-strict-float-cast-overflow	Supported	Relaxes language rules
-fno-strict-return	Supported	Prohibits treating the
-fno-sycl	Unsupported	Disables SYCL kernel
-fno-temp-file	Supported	Asserts direct creation
-fno-threadsafe-statics	Supported	Prohibits emitting coo
-fno-trigraphs	Supported	Prohibits processing t
-fno-unique-section-names	Supported	Prohibits the usage of
-fno-unroll-loops	Supported	Turns off the loop unr
-fno-use-cxa-atexit	Supported	Prohibits the usage of
-fno-use-flang-math-libs	Supported	Asserts the usage of F
-fno-use-init-array	Supported	Asserts the usage of .c
-fno-visibility-inlines-hidden-static-local-var	Supported	Disables -fvisibility-in
-fno-xray-function-index	Unsupported	Allows omitting funct
-fno-zero-initialized-in-bss	Supported	Prohibits placing zero
-fobjc-arc-exceptions	Unsupported	Asserts using EH-safe
-fobjc-arc	Unsupported	Synthesizes retain and
-fobjc-exceptions	Unsupported	Enables Objective-C e
-fobjc-runtime= <value>	Unsupported	Specifies the target O
-fobjc-weak	Unsupported	Enables ARC-style we
-fopenmp-simd	Unsupported	Emits OpenMP code
-fopenmp-targets= <value>	Unsupported	Specifies a comma-sep
-fopenmp	Unsupported	Parses OpenMP pragm

Option	Support Status	Description
-foptimization-record-file= <file>	Supported	Specifies the output n
-foptimization-record-passes= <regex>	Supported	Exclusively allows the
-forder-file-instrumentation	Supported	Generates instrumente
-fpack-struct= <value>	Unsupported	Specifies the default n
-fpascal-strings	Supported	Recognizes and constr
-fpass-plugin= <dsopath>	Supported	Loads pass plugin fro
-fpatchable-function-entry= <N,M>	Supported	Generates M NOPs be
-fpcc-struct-return	Unsupported	Overrides the default
-fpch-codegen	Supported	Generates code for us
-fpch-debuginfo	Supported	Generates debug info
-fpch-instantiate-templates	Supported	Instantiates templates
-fpch-validate-input-files-content	Supported	Validates PCH input
-fplugin= <dsopath>	Supported	Loads the named plug
-fprebuilt-module-path= <directory>	Unsupported	Specifies the prebuilt
-fprofile-exclude-files= <value>	Unsupported	Exclusively instrumen
-fprofile-filter-files= <value>	Unsupported	Exclusively instrumen
-fprofile-generate= <directory>	Unsupported	Generates instrumente
-fprofile-generate	Unsupported	Generates instrumente
-fprofile-instr-generate= <file>	Unsupported	Generates instrumente
-fprofile-instr-generate	Unsupported	Generates instrumente
-fprofile-instr-use= <value>	Unsupported	Uses instrumentation
-fprofile-remapping-file= <file>	Unsupported	Uses the remappings c
-fprofile-sample-accurate	Unsupported	Specifies that the sam
-fprofile-sample-use= <value>	Unsupported	Enables sample-based
-fprofile-use= <pathname>	Unsupported	Uses instrumentation
-freciprocal-math	Supported	Allows division operat
-freg-struct-return	Unsupported	Overrides the default
-fregister-global-dtors-with-atexit	Supported	Uses atexit or ___cxa_
-frelaxed-template-template-args	Supported	Enables C++17 relaxe
-freroll-loops	Supported	Turns on loop reroller
-fropi	Unsupported	Generates read-only p
-frtlib-add-rpath	Supported	Adds -rpath with arch
-frwpi	Unsupported	Generates read-write p
-fsanitize-address-field-padding= <value>	Supported on Host only	Specifies the level of fi
-fsanitize-address-globals-dead-stripping	Supported on Host only	Enables linker dead st
-fsanitize-address-poison-custom-array-cookie	Supported on Host only	Enables poisoning of a
-fsanitize-address-use-after-scope	Supported on Host only	Enables use-after-scop
-fsanitize-address-use-odr-indicator	Supported on Host only	Enables ODR indicato
-fsanitize-blacklist= <value>	Supported on Host only	Specifies the path to b
-fsanitize-cfi-canonical-jump-tables	Supported on Host only	Makes the jump table
-fsanitize-cfi-cross-dso	Supported on Host only	Enables control flow in
-fsanitize-cfi-icall-generalize-pointers	Supported on Host only	Generalizes pointers in
-fsanitize-coverage-allowlist= <value>	Supported on Host only	Restricts sanitizer cov
-fsanitize-coverage-blacklist= <value>	Supported on Host only	Deprecated; use -fsani
-fsanitize-coverage-blocklist= <value>	Supported on Host only	Disables sanitizer cove
-fsanitize-coverage-whitelist= <value>	Supported on Host only	Deprecated; use -fsani
-fsanitize-coverage= <value>	Supported on Host only	Specifies the type of c
-fsanitize-hwaddress-abi= <value>	Supported on Host only	Selects the HWAddres
-fsanitize-memory-track-origins= <value>	Supported on Host only	Enables origins tracki
-fsanitize-memory-track-origins	Supported on Host only	Enables origins tracki

Option	Support Status	Description
-fsanitize-memory-use-after-dtor	Supported on Host only	Enables use-after-dest
-fsanitize-recover= <value>	Supported on Host only	Enables recovery for s
-fsanitize-stats	Supported on Host only	Enables sanitizer stati
-fsanitize-system-blacklist= <value>	Supported on Host only	Specifies the path to s
-fsanitize-thread-atomics	Supported on Host only	Enables atomic operat
-fsanitize-thread-func-entry-exit	Supported on Host only	Enables function entry
-fsanitize-thread-memory-access	Supported on Host only	Enables memory acces
-fsanitize-trap= <value>	Supported on Host only	Enables trapping for s
-fsanitize-trap	Supported on Host only	Enables trapping for a
-fsanitize-undefined-strip-path-components= <number>	Supported on Host only	Strips (or keeps only,
-fsanitize= <check>	Supported on Host only	Turns on runtime chec
-fsave-optimization-record= <format>	Supported	Generates an optimiza
-fsave-optimization-record	Supported	Generates a YAML op
-fseh-exceptions	Supported	Uses SEH style except
-fshort-enums	Supported	Allocates to an enum
-fshort-wchar	Unsupported	Forces wchar_t to be
-fshow-overloads= <value>	Supported	Specifies which overlo
-fsigned-char	Supported	Asserts that the char
-fsized-deallocation	Supported	Enables C++14 sized
-fsjlj-exceptions	Supported	Uses SjLj style except
-fslp-vectorize	Supported	Enables the superword
-fsplit-dwarf-inlining	Unsupported	Provides minimal deb
-fsplit-lto-unit	Unsupported	Enables splitting of th
-fsplit-machine-functions	Supported	Enables late function
-fstack-clash-protection	Supported	Enables stack clash pr
-fstack-protector-all	Unsupported	Enables stack protecto
-fstack-protector-strong	Unsupported	Enables stack protecto
-fstack-protector	Unsupported	Enables stack protecto
-fstack-size-section	Supported	Emits section containi
-fstandalone-debug	Supported	Emits full debug info
-fstrict-enums	Supported	Enables optimizations
-fstrict-float-cast-overflow	Supported	Assumes the overflowi
-fstrict-vtable-pointers	Supported	Enables optimizations
-fsycl	Unsupported	Enables SYCL kernels
-fsystem-module	u	Builds this module as
-fthin-link-bitcode= <value>	Supported	Writes minimized bitc
-fthinlto-index= <value>	Unsupported	Performs ThinLTO im
-ftime-trace-granularity= <value>	Supported	Specifies the minimum
-ftime-trace	Supported	Turns on time profiler
-ftrap-function= <value>	Unsupported	Issues call to specified
-ftrapv-handler= <function name>	Unsupported	Specifies the function
-ftrapv	Supported	Traps on integer overf
-ftrigraphs	Supported	Processes trigraph seq
-ftrivial-auto-var-init-stop-after= <value>	Supported	Stops initializing trivi
-ftrivial-auto-var-init= <value>	Supported	Initializes trivial auto
-funique-basic-block-section-names	Supported	Uses unique names for
-funique-internal-linkage-names	Supported	Makes the Internal Li
-funroll-loops	Supported	Turns on loop unroller
-fuse-flang-math-libs	Supported	Uses Flang internal ru
-fuse-line-directives	Supported	Uses #line in preproc

Option	Support Status	Description
-fvalidate-ast-input-files-content	Supported	Computes and stores
-fveclib= <value>	Unsupported	Uses the given vector
-fvectorize	Unsupported	Enables the loop vect
-fverbose-asm	Supported	Generates verbose ass
-fvirtual-function-elimination	Supported	Enables dead virtual f
-fvisibility-global-new-delete-hidden	Supported	Marks the visibility of
-fvisibility-inlines-hidden-static-local-var	Supported	Marks the visibility of
-fvisibility-inlines-hidden	Supported	Marks the visibility of
-fvisibility-ms-compat	Supported	Marks the visibility of
-fvisibility= <value>	Supported	Sets the default symb
-fwasm-exceptions	Unsupported	Uses WebAssembly st
-fwhole-program-vtables	Unsupported	Enables whole progra
-fwrapv	Supported	Treats signed integer
-fwritable-strings	Supported	Stores string literals a
-fxray-always-emit-customevents	Unsupported	Mandates emitting __
-fxray-always-emit-typedevents	Unsupported	Mandates emitting __
-fxray-always-instrument= <value>	Unsupported	Deprecated: Specifies
-fxray-attr-list= <value>	Unsupported	Specifies the filename
-fxray-ignore-loops	Unsupported	Prohibits instrumenti
-fxray-instruction-threshold= <value>	Unsupported	Sets the minimum fun
-fxray-instrumentation-bundle= <value>	Unsupported	Specifies which XRay
-fxray-instrument	Unsupported	Generates XRay instr
-fxray-link-deps	Unsupported	Informs Clang to add
-fxray-modes= <value>	Unsupported	Specifies the list of mo
-fxray-never-instrument= <value>	Unsupported	Deprecated: Specifies
-fzvector	Supported	Enables System z vect
-F <value>	Unsupported	Adds directory to the
–gcc-toolchain= <value>	Supported	Uses the gcc toolchain
-gcodeview-ghash	Supported	Emits type record has
-gcodeview	Supported	Generates code view d
-gdwarf-2	Supported	Generates source-level
-gdwarf-3	Supported	Generates source-level
-gdwarf-4	Supported	Generates source-level
-gdwarf-5	Supported	Generates source-level
-gdwarf	Supported	Generates source-level
-gembed-source	Supported	Embeds source text in
-gline-directives-only	Supported	Emits debug line info
-gline-tables-only	Supported	Emits debug line num
-gmodules	Supported	Generates debug info
-gno-embed-source	Supported	Restores the default b
-gno-inline-line-tables	Supported	Prohibits emitting inl
–gpu-max-threads-per-block= <value>	Supported	Specifies the default m
-gsplit-dwarf= <value>	Supported	Sets DWARF fission m
-gz= <value>	Supported	Specifies DWARF deb
-gz	Supported	Shows DWARF debug
-G <size>	Unsupported	Puts objects of maxim
-g	Supported	Generates source-level
–help-hidden	Supported	Displays help for hidd
-help	Supported	Displays available opt
–hip-device-lib= <value>	Supported	Specifies the HIP devi

Option	Support Status	Description
–hip-link	Supported	Links clang-offload-bu
–hip-version= <value>	Supported	Allows specification of
-H	Supported	Shows header “include
-I-	Supported	Restricts all prior -I fl
-ibuiltininc	Supported	Enables built-in #incl
-idirafter <value>	Supported	Adds the directory to
-iframeworkwithsysroot <directory>	Unsupported	Adds the directory to
-iframework <value>	Unsupported	Adds the directory to
-imacros <file>	Supported	Specifies the file conta
-include-pch <file>	Supported	Includes the specified
-include <file>	Supported	Includes the specified
-index-header-map	Supported	Makes the next includ
-iprefix <dir>	Supported	Sets the -iwithprefix/-
-iquote <directory>	Supported	Adds the directory to
-isysroot <dir>	Supported	Sets the system root d
-isystem-after <directory>	Supported	Adds the directory to
-isystem <directory>	Supported	Adds the directory to
-ivfsoverlay <value>	Supported	Overlays the virtual fi
-iwithprefixbefore <dir>	Supported	Sets the directory to i
-iwithprefix <dir>	Supported	Sets the directory to S
-iwithsysroot <directory>	Supported	Adds directory to SYS
-I <dir>	Supported	Adds directory to incl
–libomptarget-nvptx-path= <value>	Unsupported	Specifies path to libon
-L <dir>	Supported	Adds directory to libr
-mabicalls	Unsupported	Enables SVR4-style p
-maix-struct-return	Unsupported	Returns all structs in
-malign-branch-boundary= <value>	Supported	Specifies the boundary
-malign-branch= <value>	Supported	Specifies the types of
-malign-double	Supported	Aligns doubles to two
-Mallocatable= <value>	Unsupported	Provides semantics for
-mbackchain	Unsupported	Links stack frames thr
-mbranch-protection= <value>	Unsupported	Enforces targets of inc
-mbranches-within-32B-boundaries	Supported	Aligns selected branch
-mcmodel=medany	Unsupported	Equivalent to -mcmod
-mcmodel=medlow	Unsupported	Equivalent to -mcmod
-mcmse	Unsupported	Allows use of CMSE (
-mcode-object-v3	Supported	Legacy option to spec
-mcode-object-version= <version>	Supported	Specifies code object A
-mcrc	Unsupported	Allows use of CRC ins
-mcumode	Supported	Specifies CU (-mcumo
-mdouble= <value>	Supported	Forces double to be 32
-MD	Supported	Writes a depfile conta
-meabi <value>	Supported	Sets EABI type. Valu
-membedded-data	Unsupported	Places constants in th
-menable-experimental-extensions	Unsupported	Enables usage of expe
-mexec-model= <value>	Unsupported	Specifies the execution
-mexecute-only	Unsupported	Disallows generation o
-mextern-sdata	Unsupported	Assumes externally de
-mfentry	Unsupported	Inserts calls to fentry
-mfix-cortex-a53-835769	Unsupported	Workaround Cortex-A

Option	Support Status	Description
0	Unsupported	Asserts usage of 32-bi
0	Unsupported	Asserts usage of 64-bi
-MF <file>	Supported	Writes depfile output
-mgeneral-regs-only	Unsupported	Generates code that e
-mglobal-merge	Supported	Enables merging of gl
-mgpopt	Unsupported	Allows using GP relat
-MG	Supported	Adds missing headers
-mharden-sls= <value>	Unsupported	Sets straight-line spec
-mhvx-length= <value>	Unsupported	Sets Hexagon Vector l
-mhvx= <value>	Unsupported	Sets Hexagon Vector e
-mhvx	Unsupported	Enables Hexagon Vect
-miamcu	Unsupported	Allows using Intel MC
–migrate	Unsupported	Runs the migrator
-mincremental-linker-compatible	Supported	(integrated-as) Emits
-mindirect-jump= <value>	Unsupported	Changes indirect jump
-Minform= <value>	Supported	Sets error level of mes
-mios-version-min= <value>	Unsupported	Sets iOS deployment t
-MJ <value>	Unsupported	Writes a compilation d
-mllvm <value>	Supported	Specifies additional ar
-mlocal-sdata	Unsupported	Extends the -G behav
-mlong-calls	Supported	Generates branches w
-mlong-double-128	Supported on Host only	Forces long double to
-mlong-double-64	Supported	Forces long double to
-mlong-double-80	Supported on Host only	Forces long double to
-mlvi-cfi	Supported on Host only	Enables only control-f
-mlvi-hardening	Supported on Host only	Enables all mitigation
-mmacosx-version-min= <value>	Unsupported	Sets Mac OS X deploy
-mmadd4	Supported	Enables the generatio
-mmark-bti-property	Unsupported	Adds .note.gnu.proper
-MMD	Supported	Writes a depfile conta
-mmemops	Supported	Enables generation of
-mms-bitfields	Unsupported	Sets the default struct
-mmsa	Unsupported	Enables MSA ASE (M
-mmt	Unsupported	Enables MT ASE (MI
-MM	Supported	Similar to -MMD but
-mno-abicalls	Unsupported	Disables SVR4-style p
-mno-crc	Unsupported	Disallows use of CRC
-mno-embedded-data	Unsupported	Prohibits placing cons
-mno-execute-only	Unsupported	Allows generation of d
-mno-extern-sdata	Unsupported	Prohibits assuming th
-mno-fix-cortex-a53-835769	Unsupported	Disallows workaround
-mno-global-merge	Supported	Disables merging of gl
-mno-gpopt	Unsupported	Prohibits using GP re
-mno-hvx	Unsupported	Disables Hexagon Vec
-mno-implicit-float	Supported	Prohibits generating i
-mno-incremental-linker-compatible	Supported	(integrated-as) Emits
-mno-local-sdata	Unsupported	Prohibits extending th
-mno-long-calls	Supported	Restores the default b
-mno-lvi-cfi	Supported on Host only	Disables control-flow m
-mno-lvi-hardening	Supported on Host only	Disables mitigations fo

Option	Support Status	Description
-mno-madd4	Supported	Disables the generatio
-mno-memops	Supported	Disables the generatio
-mno-movt	Supported	Disallows usage of mo
-mno-ms-bitfields	Supported	Prohibits setting the c
-mno-msa	Unsupported	Disables MSA ASE (M
-mno-mt	Unsupported	Disables MT ASE (M
-mno-neg-immediates	Supported	Disallows converting i
-mno-nvj	Supported	Disables generation of
-mno-nvs	Supported	Disables generation of
-mno-outline	Unsupported	Disables function outl
-mno-packets	Supported	Disables generation of
-mno-relax	Supported	Disables linker relaxat
-mno-restrict-it	Unsupported	Allows generation of c
-mno-save-restore	Unsupported	Disables usage of libra
-mno-seses	Unsupported	Disables speculative e
-mno-stack-arg-probe	Supported	Disables stack probes
-mno-tls-direct-seg-refs	Supported	Disables direct TLS a
-mno-unaligned-access	Unsupported	Forces all memory acc
-mno-wavefrontsize64	Supported	Asserts wavefront size
-mnocrc	Unsupported	Disallows usage of CR
-mnop-mcount	Supported	Generates mcount/__
-mnvj	Supported	Enables generation of
-mnvs	Supported	Enables generation of
-module-dependency-dir <value>	Unsupported	Specifies directory for
-module-file-info	Unsupported	Provides information
-momit-leaf-frame-pointer	Supported	Omits frame pointer s
-moutline	Unsupported	Enables function outli
-mpacked-stack	Unsupported	Asserts the usage of p
-mpackets	Supported	Enables generation of
-mpad-max-prefix-size= <value>	Supported	Specifies maximum nu
-mpie-copy-relocations	Supported	Asserts the usage of c
-mprefer-vector-width= <value>	Unsupported	Specifies preferred vec
-MP	Supported	Creates phony target
-mqdsp6-compat	Unsupported	Enables hexagon-qdsp
-MQ <value>	Supported	Specifies the name of
-mrecord-mcount	Supported	Generates a ___mcoun
-mrelax-all	Supported	(integrated-as) Relaxe
-mrelax	Supported	Enables linker relaxat
-mrestrict-it	Unsupported	Disallows generation c
-mrtd	Unsupported	Makes StdCall calling
-msave-restore	Unsupported	Enables using library
-mseses	Unsupported	Enables speculative ex
-msign-return-address= <value>	Unsupported	Specifies the return ad
-msmall-data-limit= <value>	Supported	Puts global and static
-msoft-float	Supported	Uses software floating
-msram-ecc	Supported	Legacy option to spec
-mstack-alignment= <value>	Unsupported	Sets the stack alignme
-mstack-arg-probe	Unsupported	Enables stack probes
-mstack-probe-size= <value>	Unsupported	Sets the stack probe s
-mstackrealign	Unsupported	Forces realign the stac

Option	Support Status	Description
-msve-vector-bits= <value>	Unsupported	Specifies the size in bi
-msvr4-struct-return	Unsupported	Returns small structs
-mthread-model <value>	Supported	Specifies the thread m
-mtls-direct-seg-refs	Supported	Enables direct TLS ac
-mtls-size= <value>	Unsupported	Specifies the bit size o
-mtp= <value>	Unsupported	Specifies the thread p
-mtune= <value>	Supported on Host only	Supported on X86 onl
-MT <value>	Unsupported	Specifies the name of
-munaligned-access	Unsupported	Allows memory access
-MV	Supported	Uses NMake/Jom forr
-mwavefrontsize64	Supported	Asserts wavefront size
-mxnack	Supported	Legacy option to spec
-M	Supported	Similar to -MD but al
–no-cuda-include-ptx= <value>	Supported	Prohibits including PT
–no-cuda-version-check	Supported	Disallows erroring out
-no-flang-libs	Supported	Prohibits linking again
–no-offload-arch= <value>	Supported	Removes CUDA/HIP
–no-system-header-prefix= <prefix>	Supported	Assumes no system he
-nobuiltininc	Supported	Disables built-in #inc
-nogpuinc	Supported	Prohibits adding CUD
-nogpulib	Supported	Prohibits linking devi
-nostdinc++	Unsupported	Disables standard #in
-ObjC++	Unsupported	Treats source input fil
-objcmt-atomic-property	Unsupported	Enables migration to
-objcmt-migrate-all	Unsupported	Enables migration to
-objcmt-migrate-annotation	Unsupported	Enables migration to
-objcmt-migrate-designated-init	Unsupported	Enables migration to
-objcmt-migrate-instancetype	Unsupported	Enables migration to
-objcmt-migrate-literals	Unsupported	Enables migration to
-objcmt-migrate-ns-macros	Unsupported	Enables migration to
-objcmt-migrate-property-dot-syntax	Unsupported	Enables migration of s
-objcmt-migrate-property	Unsupported	Enables migration to
-objcmt-migrate-protocol-conformance	Unsupported	Enables migration to
-objcmt-migrate-readonly-property	Unsupported	Enables migration to
-objcmt-migrate-readwrite-property	Unsupported	Enables migration to
-objcmt-migrate-subscripting	Unsupported	Enables migration to
-objcmt-ns-nonatomic-iosonly	Unsupported	Enables migration to
-objcmt-returns-innerpointer-property	Unsupported	Enables migration to
-objcmt-whitelist-dir-path= <value>	Unsupported	Modifies exclusively th
-ObjC	Unsupported	Treats source input fil
–offload-arch= <value>	Supported	Specifies CUDA offloa
-o <file>	Supported	Writes output to the g
-parallel-jobs= <value>	Supported	Specifies the number o
-pg	Supported	Enables mcount instru
-pipe	Supported	Asserts using pipes be
–precompile	Supported	Only precompiles the
-print-effective-triple	Supported	Prints the effective tar
-print-file-name= <file>	Supported	Prints the full library
-print-ivar-layout	Unsupported	Enables Objective-C I
-print-libgcc-file-name	Supported	“Prints the library pa

Option	Support Status	Description
-print-prog-name= <name>	Supported	Prints the full prograr
-print-resource-dir	Supported	Prints the resource di
-print-search-dirs	Supported	Prints the paths used
-print-supported-cpus	Supported	Prints the supported
-print-target-triple	Supported	Prints the normalized
-print-targets	Supported	Prints the registered t
-pthread	Supported	Supports POSIX thre
–ptxas-path= <value>	Unsupported	Specifies the path to p
-P	Supported	Disables linemarker ou
-Qn	Supported	Prohibits emitting me
-Qunused-arguments	Supported	Prohibits emitting wa
-Qy	Supported	Emits metadata conta
-relocatable-pch	Supported	Allows to build a relo
-rewrite-legacy-objc	Unsupported	Rewrites Legacy Obje
-rewrite-objc	Unsupported	Rewrites Objective-C
–rocm-device-lib-path= <value>	Supported	Specifies ROCm devic
–rocm-path= <value>	Supported	Specifies ROCm insta
-Rpass-analysis= <value>	Supported	Reports transformatic
-Rpass-missed= <value>	Supported	Reports missed transf
-Rpass= <value>	Supported	Reports transformatic
-rtlib= <value>	Unsupported	Specifies the compiler
-R <remark>	Unsupported	Enables the specified
-save-stats= <value>	Supported	Saves llvm statistics
-save-stats	Supported	Saves llvm statistics
-save-temps= <value>	Supported	Saves intermediate co
-save-temps	Supported	Saves intermediate co
-serialize-diagnostics= <value>	Supported	Serializes compiler dia
-shared-libsan	Unsupported	Dynamically links the
-static-flang-libs	Supported	Asserts linking using s
-static-libsan	Unsupported	Statically links the sa
-static-openmp	Supported	Asserts using the stat
-std= <value>	Supported	Specifies the language
-stdlib+±isystem <directory>	Supported	Specifies the directory
-stdlib= <value>	Supported	Specifies the C++ sta
-sycl-std= <value>	Unsupported	Specifies the SYCL la
–system-header-prefix= <prefix>	Supported	Assumes all #include
-S	Supported	Runs only preprocess
–target= <value>	Supported	Generates code for th
-Tbss <addr>	Supported	Sets the starting addr
-Tdata <addr>	Supported	Sets the starting addr
-time	Supported	Times individual com
-traditional-cpp	Unsupported	Enables some traditio
-trigraphs	Supported	Processes trigraph seq
-Ttext <addr>	Supported	Sets starting address
-T \ <script\>	Unsupported	Specifies the given. \
-undef	Supported	undefs all system defi
-unwindlib= <value>	Supported	Specifies the unwind l
-U <macro>	Supported	Undefines the given <
–verify-debug-info	Supported	Verifies the binary rep
-verify-pch	Unsupported	Loads and verifies if a

Option	Support Status	Description
–version	Supported	Prints version informa
-v	Supported	Shows commands to b
-Wa, <arg>	Supported	Passes the comma-sep
-Wdeprecated	Supported	Enables warnings for
-Wl, <arg>	Supported	Passes comma-separat
-working-directory <value>	Supported	Resolves file paths rel
-Wp, <arg>	Supported	Passes comma-separat
-W <warning>	Supported	Enables the specified
-w	Supported	Suppresses all warning
-Xanalyzer <arg>	Supported	Passes <arg> to the s
-Xarch_device <arg>	Supported	Passes <arg> to the C
-Xarch_host <arg>	Supported	Passes <arg> to the C
-Xassembler <arg>	Supported	Passes <arg> to the a
-Xclang <arg>	Supported	Passes <arg> to the c
-Xcuda-fatbinary <arg>	Supported	Passes <arg> to fatbi
-Xcuda-ptxas <arg>	Supported	Passes <arg> to the p
-Xlinker <arg>	Supported	Passes <arg> to the l
-Xopenmp-target= <triple> <arg>	Supported	Passes <arg> to the t
-Xopenmp-target <arg>	Supported	Passes <arg> to the t
-Xpreprocessor <arg>	Supported	Passes <arg> to the p
-x <language>	Supported	Assumes subsequent i
-z <arg>	Supported	Passes -z <arg> to th

CHAPTER

TWENTYTHREE

MANAGEMENT TOOLS

AMD SMI The AMD System Management Interface Library, or AMD SMI library, is a C library for Linux that provides a user space interface for applications to monitor and control AMD devices.

- Documentation
- GitHub
- Examples

ROCm SMI LIB This tool acts as a command line interface for manipulating and monitoring the AMD GPU kernel, and is intended to replace and deprecate the existing rocm_smi.py CLI tool. It uses ctypes to call the rocm_smi_lib API.

- Documentation
- GitHub
- Examples

ROCm Data Center Tool The ROCm™ Data Center Tool simplifies the administration and addresses key infrastructure challenges in AMD GPUs in cluster and data center environments.

- GitHub
- Changelog
- Examples

CHAPTER

TWENTYFOUR

VALIDATION TOOLS

RVS The ROCm Validation Suite is a system administrator's and cluster manager's tool for detecting and troubleshooting common problems affecting AMD GPU(s) running in a high-performance computing environment, enabled using the ROCm software stack on a compatible platform.

- Documentation
- GitHub
- Changelog

TransferBench TransferBench is a simple utility capable of benchmarking simultaneous transfers between user-specified devices (CPUs/GPUs).

- Documentation
- GitHub
- Changelog
- Examples

CHAPTER

TWENTYFIVE

ALL EXPLANATION MATERIAL

Compiler Nomencalture ROCm ships multiple compilers of varying origins and purposes. This article disambiguates compiler naming used throughout the documentation.

Using CMake ROCm components ship with 1st party CMake support. This article details how that support works and how to use it.

Linux Folder Structure Reorganization ROCm™ packages have adopted the Linux foundation file system hierarchy standard to ensure ROCm components follow open source conventions for Linux-based distributions.

GPU Isolation Techniques Restricting the access of applications to a subset of GPUs, aka isolating GPUs allows users to hide GPU resources from programs.

GPU Architectures AMD documentation around architectural details from both the CDNA and RDNA product lines.

CHAPTER

TWENTYSIX

ROCM COMPILERS DISAMBIGUATION

ROCm ships multiple compilers of varying origins and purposes. This article disambiguates compiler naming used throughout the documentation.

26.1 Compiler Terms

Term	Description
amd-clang-	Clang/LLVM-based compiler that is part of rocm-llvm package. The source code is available at https://github.com/RadeonOpenCompute/llvm-project.
AOCC	Closed-source clang-based compiler that includes additional CPU optimizations. Offered as part of ROCm via the rocm-llvm-alt package. See for details, https://developer.amd.com/amd-aocc/.
HIP-Clang	Informal term for the amdclang++ compiler
HIP-IFY	Tools including hipify-clang and hipify-perl, used to automatically translate CUDA source code into portable HIP C++. The source code is available at https://github.com/ROCm-Developer-Tools/HIPIFY
hipcc	HIP compiler driver. A utility that invokes clang or nvcc depending on the target and passes the appropriate include and library options for the target compiler and HIP infrastructure. The source code is available at https://github.com/ROCm-Developer-Tools/HIPCC.
ROCm	Clang/LLVM-based compiler. ROCmCC in itself is not a binary but refers to the overall compiler.

CHAPTER

TWENTYSEVEN

USING CMAKE

Most components in ROCm support CMake. Projects depending on header-only or library components typically require CMake 3.5 or higher whereas those wanting to make use of the CMake HIP language support will require CMake 3.21 or higher.

27.1 Finding Dependencies

Note: For a complete reference on how to deal with dependencies in CMake, refer to the CMake docs on find_package and the Using Dependencies Guide to get an overview of CMake related facilities.

In short, CMake supports finding dependencies in two ways:

- In Module mode, it consults a file Find<PackageName>.cmake which tries to find the component in typical install locations and layouts. CMake ships a few dozen such scripts, but users and projects may ship them as well.
- In Config mode, it locates a file named <packagename>-config.cmake or <PackageName>Config.cmake which describes the installed component in all regards needed to consume it.

ROCm predominantly relies on Config mode, one notable exception being the Module driving the compilation of HIP programs on NVIDIA runtimes. As such, when dependencies are not found in standard system locations, one either has to instruct CMake to search for package config files in additional folders using the CMAKE_PREFIX_PATH variable (a semi-colon separated list of filesystem paths), or using <PackageName>_ROOT variable on a project-specific basis.

There are nearly a dozen ways to set these variables. One may be more convenient over the other depending on your workflow. Conceptually the simplest is adding it to your CMake configuration command on the command-line via -D CMAKE_PREFIX_PATH=.... . AMD packaged ROCm installs can typically be added to the config file search paths such as:

- Windows: -D CMAKE_PREFIX_PATH=${env:HIP_PATH}
- Linux: -D CMAKE_PREFIX_PATH=/opt/rocm

ROCm provides the respective config-file packages, and this enables find_package to be used directly. ROCm does not require any Find module as the config-file packages are shipped with the upstream projects, such as rocPRIM and other ROCm libraries.

For a complete guide on where and how ROCm may be installed on a system, refer to the installation guides in these docs (Linux).

27.2 Using HIP in CMake

ROCm components providing a C/C++ interface support being consumed using any C/C++ toolchain that CMake knows how to drive. ROCm also supports the CMake HIP language features, allowing users to program using the HIP single-source programming model. When a program (or translation-unit) uses the HIP API without compiling any GPU device code, HIP can be treated in CMake as a simple C/C++ library.

27.2.1 Using the HIP single-source programming model

Source code written in the HIP dialect of C++ typically uses the .hip extension. When the HIP CMake language is enabled, it will automatically associate such source files with the HIP toolchain being used.

```
cmake_minimum_required(VERSION 3.21) # HIP language support requires 3.21
cmake_policy(VERSION 3.21.3...3.27)
project(MyProj LANGUAGES HIP)
add_executable(MyApp Main.hip)
```

Should you have existing CUDA code that is from the source compatible subset of HIP, you can tell CMake that despite their .cu extension, they're HIP sources. Do note that this mostly facilitates compiling kernel code-only source files, as host-side CUDA API won't compile in this fashion.

```
add_library(MyLib MyLib.cu)
set_source_files_properties(MyLib.cu PROPERTIES LANGUAGE HIP)
```

CMake itself only hosts part of the HIP language support, such as defining HIP-specific properties, etc. while the other half ships with the HIP implementation, such as ROCm. CMake will search for a file hip-lang-config.cmake describing how the the properties defined by CMake translate to toolchain invocations. If one installs ROCm using non-standard methods or layouts and CMake can't locate this file or detect parts of the SDK, there's a catch-all, last resort variable consulted locating this file, -D CMAKE_HIP_COMPILER_ROCM_ROOT:PATH= which should be set the root of the ROCm installation.

If the user doesn't provide a semi-colon delimited list of device architectures via CMAKE_HIP_ARCHITECTURES, CMake will select some sensible default. It is advised though that if a user knows what devices they wish to target, then set this variable explicitly.

27.2.2 Consuming ROCm C/C++ Libraries

Libraries such as rocBLAS, rocFFT, MIOpen, etc. behave as C/C++ libraries. Illustrated in the example below is a C++ application using MIOpen from CMake. It calls find_package(miopen), which provides the MIOpen imported target. This can be linked with target_link_libraries

```
cmake_minimum_required(VERSION 3.5) # find_package(miopen) requires 3.5
cmake_policy(VERSION 3.5...3.27)
project(MyProj LANGUAGES CXX)
find_package(miopen)
add_library(MyLib ...)
target_link_libraries(MyLib PUBLIC MIOpen)
```

Note: Most libraries are designed as host-only API, so using a GPU device compiler is not necessary for downstream projects unless they use GPU device code.

27.2.3 Consuming the HIP API in C++ code

Use the HIP API without compiling the GPU device code. As there is no GPU code, any C or C++ compiler can be used. The find_package(hip) provides the hip::host imported target to use HIP in this context.

```
cmake_minimum_required(VERSION 3.5) # find_package(hip) requires 3.5
cmake_policy(VERSION 3.5...3.27)
project(MyProj LANGUAGES CXX)
find_package(hip REQUIRED)
add_executable(MyApp ...)
target_link_libraries(MyApp PRIVATE hip::host)
```

27.2.4 Compiling device code in C++ language mode

> Attention: The workflow detailed here is considered legacy and is shown for understanding's sake. It pre-dates the existence of HIP language support in CMake. If source code has HIP device code in it, it is a HIP source file and should be compiled as such. Only resort to the method below if your HIP-enabled CMake codepath can't mandate CMake version 3.21.

If code uses the HIP API and compiles GPU device code, it requires using a device compiler. The compiler for CMake can be set using either the CMAKE_C_COMPILER and CMAKE_CXX_COMPILER variable or using the CC and CXX environment variables. This can be set when configuring CMake or put into a CMake toolchain file. The device compiler must be set to a compiler that supports AMD GPU targets, which is usually Clang.

The find_package(hip) provides the hip::device imported target to add all the flags necessary for device compilation.

```
cmake_minimum_required(VERSION 3.8) # cxx_std_11 requires 3.8
cmake_policy(VERSION 3.8...3.27)
project(MyProj LANGUAGES CXX)
find_package(hip REQUIRED)
add_library(MyLib ...)
target_link_libraries(MyLib PRIVATE hip::device)
target_compile_features(MyLib PRIVATE cxx_std_11)
```

Note: Compiling for the GPU device requires at least C++11.

This project can then be configured with the following CMake commands:

- Windows: cmake -D CMAKE_CXX_COMPILER:PATH=${env:HIP_PATH}\bin\clang++.exe
- Linux: cmake -D CMAKE_CXX_COMPILER:PATH=/opt/rocm/bin/amdclang++

Which use the device compiler provided from the binary packages of ROCm HIP SDK and repo.radeon.com respectively.

When using the CXX language support to compile HIP device code, selecting the target GPU architectures is done via setting the GPU_TARGETS variable. CMAKE_HIP_ARCHITECTURES only exists when the HIP language is enabled. By default, this is set to some subset of the currently supported architectures of AMD ROCm. It can be set to the CMake option -D GPU_TARGETS="gfx1032;gfx1035".

27.2.5 ROCm CMake Packages

Component	Package	Targets
HIP	hip	hip::host, hip::device
rocPRIM	rocprim	roc::rocprim
rocThrust	rocthrust	roc::rocthrust
hipCUB	hipcub	hip::hipcub
rocRAND	rocrand	roc::rocrand
rocBLAS	rocblas	roc::rocblas
rocSOLVER	rocsolver	roc::rocsolver
hipBLAS	hipblas	roc::hipblas
rocFFT	rocfft	roc::rocfft
hipFFT	hipfft	hip::hipfft
rocSPARSE	rocsparse	roc::rocsparse
hipSPARSE	hipsparse	roc::hipsparse
rocALUTION	rocalution	roc::rocalution
RCCL	rccl	rccl
MIOpen	miopen	MIOpen
MIGraphX	migraphx	migraphx::migraphx, migraphx::migraphx_c, migraphx::migraphx_cpu, migraphx::migraphx_gpu, migraphx::migraphx_onnx, migraphx::migraphx_tf

27.3 Using CMake Presets

CMake command-lines depending on how specific users like to be when compiling code can grow to unwieldy lengths. This is the primary reason why projects tend to bake script snippets into their build definitions controlling compiler warning levels, changing CMake defaults (CMAKE_BUILD_TYPE or BUILD_SHARED_LIBS just to name a few) and all sorts anti-patterns, all in the name of convenience.

Load on the command-line interface (CLI) starts immediately by selecting a toolchain, the set of utilities used to compile programs. To ease some of the toolchain related pains, CMake does consult the CC and CXX environmental variables when setting a default CMAKE_C[XX]_COMPILER respectively, but that is just the tip of the iceberg. There's a fair number of variables related to just the toolchain itself (typically supplied using toolchain files), and then we still haven't talked about user preference or project-specific options.

IDEs supporting CMake (Visual Studio, Visual Studio Code, CLion, etc.) all came up with their own way to register command-line fragments of different purpose in a setup-and-forget fashion for quick assembly using graphical front-ends. This is all nice, but configurations aren't portable, nor can they be reused in Continuous Integration (CI) pipelines. CMake has condensed existing practice into a portable JSON format that works in all IDEs and can be invoked from any command-line. This is CMake Presets .

There are two types of preset files: one supplied by the project, called CMakePresets.json which is meant to be committed to version control, typically used to drive CI; and one meant for the user to provide, called CMakeUserPresets.json, typically used to house user preference and adapting the build to the user's

environment. These JSON files are allowed to include other JSON files and the user presets always implicitly includes the non-user variant.

27.3.1 Using HIP with presets

Following is an example CMakeUserPresets.json file which actually compiles the amd/rocm-examples suite of sample applications on a typical ROCm installation:

```
{
  "version": 3,
  "cmakeMinimumRequired": {
    "major": 3,
    "minor": 21,
    "patch": 0
  },
  "configurePresets": [
    {
      "name": "layout",
      "hidden": true,
      "binaryDir": "${sourceDir}/build/${presetName}",
      "installDir": "${sourceDir}/install/${presetName}"
    },
    {
      "name": "generator-ninja-multi-config",
      "hidden": true,
      "generator": "Ninja Multi-Config"
    },
    {
      "name": "toolchain-makefiles-c/c++-amdclang",
      "hidden": true,
      "cacheVariables": {
        "CMAKE_C_COMPILER": "/opt/rocm/bin/amdclang",
        "CMAKE_CXX_COMPILER": "/opt/rocm/bin/amdclang++",
        "CMAKE_HIP_COMPILER": "/opt/rocm/bin/amdclang++"
      }
    },
    {
      "name": "clang-strict-iso-high-warn",
      "hidden": true,
      "cacheVariables": {
        "CMAKE_C_FLAGS": "-Wall -Wextra -pedantic",
        "CMAKE_CXX_FLAGS": "-Wall -Wextra -pedantic",
        "CMAKE_HIP_FLAGS": "-Wall -Wextra -pedantic"
      }
    },
    {
      "name": "ninja-mc-rocm",
      "displayName": "Ninja Multi-Config ROCm",
      "inherits": [
        "layout",
        "generator-ninja-multi-config",
        "toolchain-makefiles-c/c++-amdclang",
        "clang-strict-iso-high-warn"
      ]
    }
  ],
  "buildPresets": [
```

(continues on next page)

(continued from previous page)

```
    {
      "name": "ninja-mc-rocm-debug",
      "displayName": "Debug",
      "configuration": "Debug",
      "configurePreset": "ninja-mc-rocm"
    },
    {
      "name": "ninja-mc-rocm-release",
      "displayName": "Release",
      "configuration": "Release",
      "configurePreset": "ninja-mc-rocm"
    },
    {
      "name": "ninja-mc-rocm-debug-verbose",
      "displayName": "Debug (verbose)",
      "configuration": "Debug",
      "configurePreset": "ninja-mc-rocm",
      "verbose": true
    },
    {
      "name": "ninja-mc-rocm-release-verbose",
      "displayName": "Release (verbose)",
      "configuration": "Release",
      "configurePreset": "ninja-mc-rocm",
      "verbose": true
    }
  ],
  "testPresets": [
    {
      "name": "ninja-mc-rocm-debug",
      "displayName": "Debug",
      "configuration": "Debug",
      "configurePreset": "ninja-mc-rocm",
      "execution": {
        "jobs": 0
      }
    },
    {
      "name": "ninja-mc-rocm-release",
      "displayName": "Release",
      "configuration": "Release",
      "configurePreset": "ninja-mc-rocm",
      "execution": {
        "jobs": 0
      }
    }
  ]
}
```

Note: Getting presets to work reliably on Windows requires some CMake improvements and/or support from compiler vendors. (Refer to Add support to the Visual Studio generators and Sourcing environment scripts .)

CHAPTER

TWENTYEIGHT

ROCM FHS REORGANIZATION

28.1 Introduction

The ROCm platform has adopted the Linux foundation Filesystem Hierarchy Standard (FHS) https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html in order to to ensure ROCm is consistent with standard open source conventions. The following sections specify how current and future releases of ROCm adhere to FHS, how the previous ROCm filesystem is supported, and how improved versioning specifications are applied to ROCm.

28.2 Adopting the Linux foundation Filesystem Hierarchy Standard (FHS)

In order to standardize ROCm directory structure and directory content layout ROCm has adopted the FHS, adhering to open source conventions for Linux-based distribution. FHS ensures internal consistency within the ROCm stack, as well as external consistency with other systems and distributions. The ROCm proposed file structure is outlined below:

```
/opt/rocm-<ver>
   | -- bin
        | -- all public binaries
   | -- lib
        | -- lib<soname>.so->lib<soname>.so.major->lib<soname>.so.major.minor.patch
             (public libaries to link with applications)
        | -- <component>
             | -- architecture dependent libraries and binaries used internally by components
        | -- cmake
             | -- <component>
                  | --<component>-config.cmake
   | -- libexec
        | -- <component>
             | -- non ISA/architecture independent executables used internally by components
   | -- include
        | -- <component>
             | -- public header files
   | -- share
        | -- html
             | -- <component>
                  | -- html documentation
        | -- info
             | -- <component>
                  | -- info files
        | -- man
```

(continues on next page)

(continued from previous page)

```
            | -- <component>
                | -- man pages
        | -- doc
            | -- <component>
                | -- license files
        | -- <component>
            | -- samples
            | -- architecture independent misc files
```

28.3 Changes From Earlier ROCm Versions

The following table provides a brief overview of the new ROCm FHS layout, compared to the layout of earlier ROCm versions. Note that /opt/ is used to denote the default rocm-installation-path and should be replaced in case of a non-standard installation location of the ROCm distribution.

```
_____________________________________________________________________________
|  New ROCm Layout              |  Previous ROCm Layout  |
|_______________________________________|_____________________________________|
| /opt/rocm-<ver>               | /opt/rocm-<ver>        |
|     | -- bin                  |     | -- bin           |
|     | -- lib                  |     | -- lib           |
|         | -- cmake            |     | -- include       |
|     | -- libexec              |     | -- <component_1> |
|     | -- include              |         | -- bin       |
|         | -- <component_1>    |         | -- cmake     |
|     | -- share                |         | -- doc       |
|         | -- html             |         | -- lib       |
|         | -- info             |         | -- include   |
|         | -- man              |         | -- samples   |
|         | -- doc              |     | -- <component_n> |
|         | -- <component_1>    |         | -- bin       |
|             | -- samples      |         | -- cmake     |
|             | -- ..           |         | -- doc       |
|         | -- <component_n>    |         | -- lib       |
|             | -- samples      |         | -- include   |
|             | -- ..           |         | -- samples   |
|_____________________________________________________________________________|
```

28.4 ROCm FHS Reorganization: Backward Compatibility

The FHS file organization for ROCm was first introduced in the release of ROCm 5.2 . Backward compatibility was implemented to make sure users could still run their ROCm applications while transitioning to the new FHS. ROCm has moved header files and libraries to their new locations as indicated in the above structure, and included symbolic-links and wrapper header files in their old location for backward compatibility. The following sections detail ROCm backward compatibility implementation for wrapper header files, executable files, library files and CMake config files.

28.4.1 Wrapper Header Files

Wrapper header files are placed in the old location (/opt/rocm-<ver>/<component>/include) with a warning message to include files from the new location (/opt/rocm-<ver>/include) as shown in the example below.

```
#pragma message "This file is deprecated. Use file from include path /opt/rocm-ver/include/ and prefix with hip."
#include <hip/hip_runtime.h>
```

- Starting at ROCm 5.2 release, the deprecation for backward compatibility wrapper header files is: #pragma message announcing #warning.
- Starting from ROCm 6.0 (tentatively) backward compatibility for wrapper header files will be removed, and the #pragma message will be announcing #error.

28.4.2 Executable Files

Executable files are available in the /opt/rocm-<ver>/bin folder. For backward compatibility, the old library location (/opt/rocm-<ver>/<component>/bin) has a soft link to the library at the new location. Soft links will be removed in a future release, tentatively ROCm v6.0.

```
$ ls -l /opt/rocm/hip/bin/
lrwxrwxrwx 1 root root   24 Jan 1 23:32 hipcc -> ../../bin/hipcc
```

28.4.3 Library Files

Library files are available in the /opt/rocm-<ver>/lib folder. For backward compatibility, the old library location (/opt/rocm-<ver>/<component>/lib) has a soft link to the library at the new location. Soft links will be removed in a future release, tentatively ROCm v6.0.

```
$ ls -l /opt/rocm/hip/lib/
drwxr-xr-x 4 root root 4096 Jan 1 10:45 cmake
lrwxrwxrwx 1 root root   24 Jan 1 23:32 libamdhip64.so -> ../../lib/libamdhip64.so
```

28.4.4 CMake Config Files

All CMake configuration files are available in the /opt/rocm-<ver>/lib/cmake/<component> folder. For backward compatibility, the old CMake locations (/opt/rocm-<ver>/<component>/lib/cmake) consist of a soft link to the new CMake config. Soft links will be removed in a future release, tentatively ROCm v6.0.

```
$ ls -l /opt/rocm/hip/lib/cmake/hip/
lrwxrwxrwx 1 root root 42 Jan 1 23:32 hip-config.cmake -> ../../../../lib/cmake/hip/hip-config.cmake
```

28.5 Changes Required in Applications Using ROCm

Applications using ROCm are advised to use the new file paths. As the old files will be deprecated in a future release. Applications have to make sure to include correct header file and use correct search paths.

1. #include<header_file.h> needs to be changed to #include <component/header_file.h>

   For example: #include <hip.h> needs to change to #include <hip/hip.h>

2. Any variable in CMake or Makefiles pointing to component folder needs to changed.

   For example: VAR1=/opt/rocm/hip needs to be changed to VAR1=/opt/rocm VAR2=/opt/rocm/hsa needs to be changed to VAR2=/opt/rocm

3. Any reference to /opt/rocm/<component>/bin or /opt/rocm/<component>/lib needs to be changed to /opt/rocm/bin and /opt/rocm/lib/, respectively.

28.6 Changes in Versioning Specifications

In order to better manage ROCm dependencies specification and allow smoother releases of ROCm while avoiding dependency conflicts, the ROCm platform shall adhere to the following scheme when numbering and incrementing ROCm files versions:

rocm-<ver>, where <ver> = <x.y.z>

x.y.z denote: MAJOR.MINOR.PATCH

z: PATCH - increment z when implementing backward compatible bug fixes.

y: MINOR - increment y when implementing minor changes that add functionality but are still backward compatible.

x: MAJOR - increment x when implementing major changes that are not backward compatible.

CHAPTER

TWENTYNINE

GPU ISOLATION TECHNIQUES

Restricting the access of applications to a subset of GPUs, aka isolating GPUs allows users to hide GPU resources from programs. The programs by default will only use the “exposed” GPUs ignoring other (hidden) GPUs in the system.

There are multiple ways to achieve isolation of GPUs in the ROCm software stack, differing in which applications they apply to and the security they provide. This page serves as an overview of the techniques.

29.1 Environment Variables

The runtimes in the ROCm software stack read these environment variables to select the exposed or default device to present to applications using them.

Environment variables shouldn’t be used for isolating untrusted applications, as an application can reset them before initializing the runtime.

29.1.1 ROCR_VISIBLE_DEVICES

A list of device indices or UUID (universally unique identifier)s that will be exposed to applications.

Runtime : ROCm Platform Runtime. Applies to all applications using the user mode ROCm software stack.

Listing 29.1: Example to expose the 1. device and a device based on UUID.

```
export ROCR_VISIBLE_DEVICES=”0,GPU-DEADBEEFDEADBEEF”
```

29.1.2 GPU_DEVICE_ORDINAL

Devices indices exposed to OpenCL and HIP applications.

Runtime : ROCm Common Language Runtime (ROCclr). Applies to applications and runtimes using the ROCclr abstraction layer including HIP and OpenCL applications.

Listing 29.2: Example to expose the 1. and 3. device in the system.

```
export GPU_DEVICE_ORDINAL=”0,2”
```

29.1.3 HIP_VISIBLE_DEVICES

Device indices exposed to HIP applications.

Runtime : HIP Runtime. Applies only to applications using HIP on the AMD platform.

Listing 29.3: Example to expose the 1. and 3. devices in the system.

```
export HIP_VISIBLE_DEVICES="0,2"
```

29.1.4 CUDA_VISIBLE_DEVICES

Provided for CUDA compatibility, has the same effect as HIP_VISIBLE_DEVICES on the AMD platform.

Runtime : HIP or CUDA Runtime. Applies to HIP applications on the AMD or NVIDIA platform and CUDA applications.

29.1.5 OMP_DEFAULT_DEVICE

Default device used for OpenMP target offloading.

Runtime : OpenMP Runtime. Applies only to applications using OpenMP offloading.

Listing 29.4: Example on setting the default device to the third device.

```
export OMP_DEFAULT_DEVICE="2"
```

29.2 Docker

Docker uses Linux kernel namespaces to provide isolated environments for applications. This isolation applies to most devices by default, including GPUs. To access them in containers explicit access must be granted, please see Accessing GPUs in containers for details. Specifically refer to Restricting a container to a subset of the GPUs on exposing just a subset of all GPUs.

Docker isolation is more secure than environment variables, and applies to all programs that use the amdgpu kernel module interfaces. Even programs that don't use the ROCm runtime, like graphics applications using OpenGL or Vulkan, can only access the GPUs exposed to the container.

29.3 GPU Passthrough to Virtual Machines

Virtual machines achieve the highest level of isolation, because even the kernel of the virtual machine is isolated from the host. Devices physically installed in the host system can be passed to the virtual machine using PCIe passthrough. This allows for using the GPU with a different operating systems like a Windows guest from a Linux host.

Setting up PCIe passthrough is specific to the hypervisor used. ROCm officially supports VMware ESXi for select GPUs.

CHAPTER

THIRTY

GPU ARCHITECTURES

30.1 Architecture Guides

AMD Instinct MI200 Review hardware aspects of the AMD Instinct™ MI250 accelerators and the CDNA™ 2 architecture that is the foundation of these GPUs.

- Instruction Set Architecture
- Whitepaper
- Guide

AMD Instinct MI100 Review hardware aspects of the AMD Instinct™ MI100 accelerators and the CDNA™ 1 architecture that is the foundation of these GPUs.

- Instruction Set Architecture
- Whitepaper
- Guide

30.2 ISA Documentation

- AMD Instinct MI200/CDNA2 Instruction Set Architecture
- AMD Instinct MI100/CDNA1 Instruction Set Architecture
- AMD Instinct MI50/Vega 7nm Instruction Set Architecture
- AMD Instinct MI25/Vega Instruction Set Architecture
- AMD RDNA3 Instruction Set Architecture
- AMD RDNA2 Instruction Set Architecture
- AMD RDNA Instruction Set Architecture
- AMD GCN3 Instruction Set Architecture

30.3 White Papers

- AMD CDNA™ 2 Architecture White Paper
- AMD CDNA Architecture White Paper
- AMD Vega Architecture White Paper
- AMD RDNA Architecture White Paper

30.4 AMD Instinct Hardware

This chapter briefly reviews hardware aspects of the AMD Instinct MI250 accelerators and the CDNA™ 2 architecture that is the foundation of these GPUs.

30.4.1 AMD CDNA 2 Micro-architecture

The micro-architecture of the AMD Instinct MI250 accelerators is based on the AMD CDNA 2 architecture that targets compute applications such as HPC, artificial intelligence (AI), and Machine Learning (ML) and that run on everything from individual servers to the world's largest exascale supercomputers. The overall system architecture is designed for extreme scalability and compute performance.

Fig. 30.1 shows the components of a single Graphics Compute Die (GCD) of the CDNA 2 architecture. On the top and the bottom are AMD Infinity Fabric™ interfaces and their physical links that are used to connect the GPU die to the other system-level components of the node (see also Section 2.2). Both interfaces can drive four AMD Infinity Fabric links. One of the AMD Infinity Fabric links of the controller at the bottom can be configured as a PCIe link. Each of the AMD Infinity Fabric links between GPUs can run at up to 25 GT/sec, which correlates to a peak transfer bandwidth of 50 GB/sec for a 16-wide link (two bytes per transaction). Section 2.2 has more details on the number of AMD Infinity Fabric links and the resulting transfer rates between the system-level components.

To the left and the right are memory controllers that attach the High Bandwidth Memory (HBM) modules to the GCD. AMD Instinct MI250 GPUs use HBM2e, which offers a peak memory bandwidth of 1.6 TB/sec per GCD.

The execution units of the GPU are depicted in Fig. 30.1 as Compute Units (CU). The MI250 GCD has 104 active CUs. Each compute unit is further subdivided into four SIMD units that process SIMD instructions of 16 data elements per instruction (for the FP64 data type). This enables the CU to process 64 work items (a so-called "wavefront") at a peak clock frequency of 1.7 GHz. Therefore, the theoretical maximum FP64 peak performance per GCD is 45.3 TFLOPS for vector instructions. The MI250 compute units also provide specialized execution units (also called matrix cores), which are geared toward executing matrix operations like matrix-matrix multiplications. For FP64, the peak performance of these units amounts to 90.5 TFLOPS.

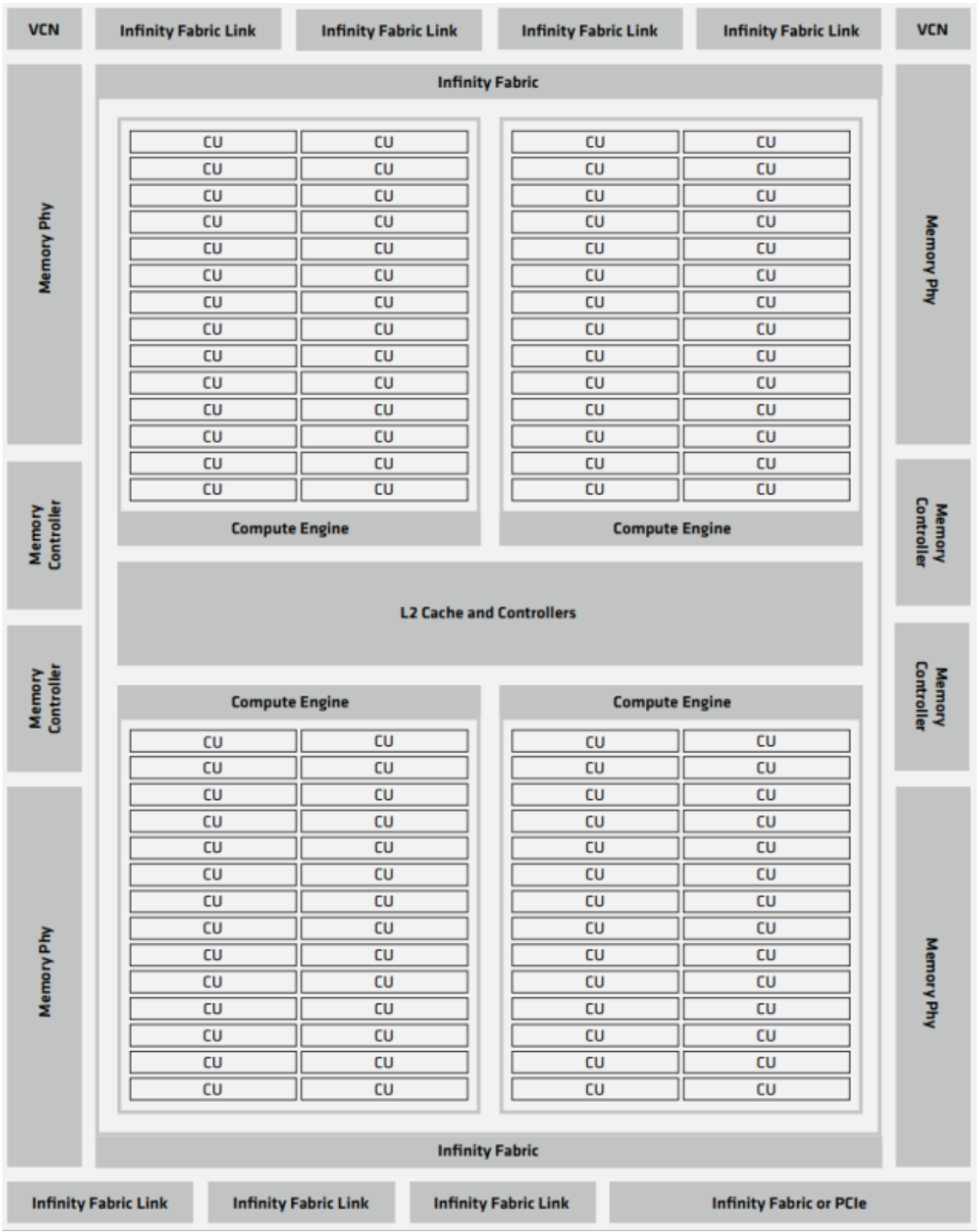


Fig. 30.1: Figure 1: Structure of a single GCD in the AMD Instinct MI250 accelerator.

Table 30.1: Peak-performance capabilities of the MI250 OAM for different data types.

Computation and Data Type	FLOPS/CLOCK/CU	Peak TFLOPS
Matrix FP64	256	90.5
Vector FP64	128	45.3
Matrix FP32	256	90.5
Packed FP32	256	90.5
Vector FP32	128	45.3
Matrix FP16	1024	362.1
Matrix BF16	1024	362.1
Matrix INT8	1024	362.1

Table 30.1 summarizes the aggregated peak performance of the AMD Instinct MI250 OCP Open Accelerator Modules (OAM, OCP is short for Open Compute Platform) and its two GCDs for different data types and execution units. The middle column lists the peak performance (number of data elements processed in a single instruction) of a single compute unit if a SIMD (or matrix) instruction is being retired in each clock cycle. The third column lists the theoretical peak performance of the OAM module. The theoretical aggregated peak memory bandwidth of the GPU is 3.2 TB/sec (1.6 TB/sec per GCD).

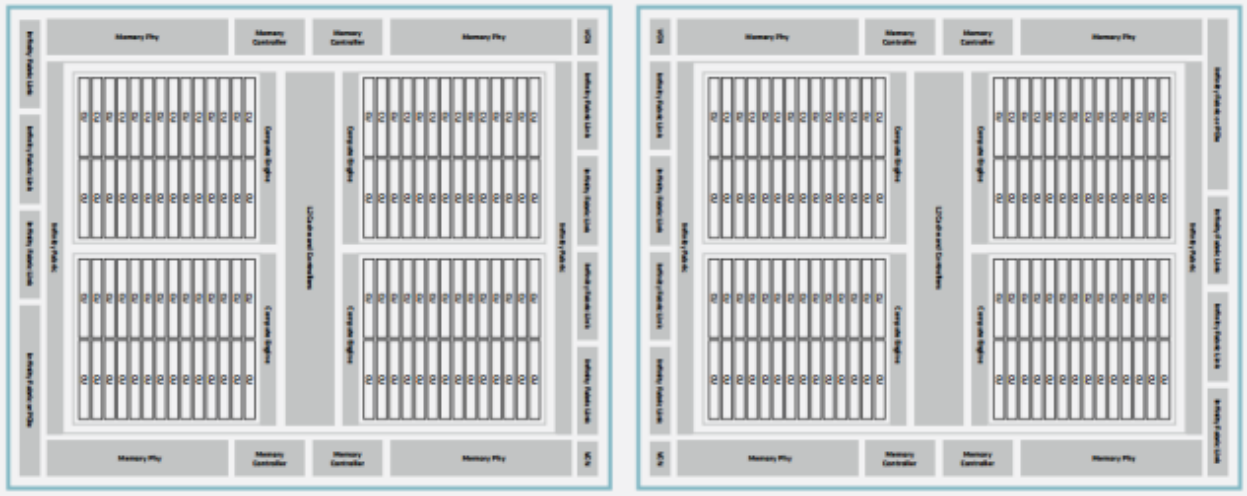


Fig. 30.2: Dual-GCD architecture of the AMD Instinct MI250 accelerators.

Fig. 30.2 shows the block diagram of an OAM package that consists of two GCDs, each of which constitutes one GPU device in the system. The two GCDs in the package are connected via four AMD Infinity Fabric links running at a theoretical peak rate of 25 GT/sec, giving 200 GB/sec peak transfer bandwidth between the two GCDs of an OAM, or a bidirectional peak transfer bandwidth of 400 GB/sec for the same.

30.4.2 Node-level Architecture

Fig. 30.3 shows the node-level architecture of a system that is based on the AMD Instinct MI250 accelerator. The MI250 OAMs attach to the host system via PCIe Gen 4 x16 links (yellow lines). Each GCD maintains its own PCIe x16 link to the host part of the system. Depending on the server platform, the GCD can attach to the AMD EPYC processor directly or via an optional PCIe switch . Note that some platforms may offer an x8 interface to the GCDs, which reduces the available host-to-GPU bandwidth.

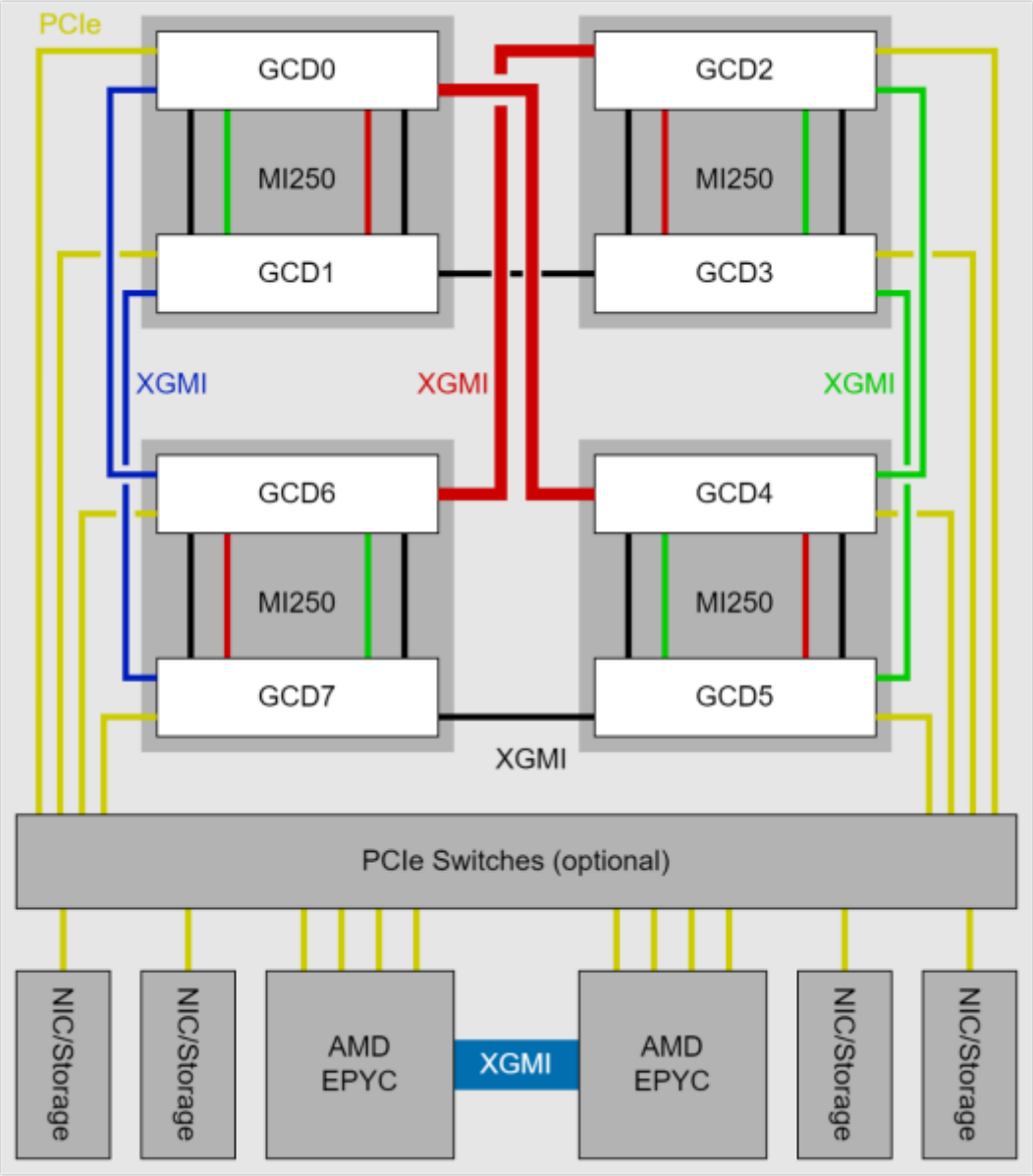


Fig. 30.3: Block diagram of AMD Instinct MI250 Accelerators with 3rd Generation AMD EPYC processor.

Fig. 30.3 shows the node-level architecture of a system with AMD EPYC processors in a dual-socket configuration and four AMD Instinct MI250 accelerators. The MI250 OAMs attach to the host processors system via PCIe Gen 4 x16 links (yellow lines). Depending on the system design, a PCIe switch may exist to make more PCIe lanes available for additional components like network interfaces and/or storage devices. Each GCD maintains its own PCIe x16 link to the host part of the system or to the PCIe switch. Please note, some

platforms may offer an x8 interface to the GCDs, which will reduce the available host-to-GPU bandwidth.

Between the OAMs and their respective GCDs, a peer-to-peer (P2P) network allows for direct data exchange between the GPU dies via AMD Infinity Fabric links (black, green, and red lines). Each of these 16-wide links connects to one of the two GPU dies in the MI250 OAM and operates at 25 GT/sec, which corresponds to a theoretical peak transfer rate of 50 GB/sec per link (or 100 GB/sec bidirectional peak transfer bandwidth). The GCD pairs 2 and 6 as well as GCDs 0 and 4 connect via two XGMI links, which is indicated by the thicker red line in Fig. 30.3.

30.5 MI200 Performance Counters and Metrics

This document lists and describes the hardware performance counters and the derived metrics available on the AMD Instinct™ MI200 GPU. All hardware performance monitors, and the derived performance metrics are accessible via AMD ROCm™ Profiler tool.

30.5.1 MI200 Performance Counters List

Note: Preliminary validation of all MI200 performance counters is in progress. Those with "[*]" appended to the names require further evaluation.

30.5.1.1 Graphics Register Bus Management (GRBM)

30.5.1.1.1 GRBM Counters

Hardware Counter	Unit	Definition
grbm_count	Cycles	Free-running GPU clock
grbm_gui_active	Cycles	GPU active cycles
grbm_cp_busy	Cycles	Any of the CP (CPC/CPF) blocks are busy.
grbm_spi_busy	Cycles	Any of the Shader Processor Input (SPI) are busy in the shader engine(s).
grbm_ta_busy	Cycles	Any of the Texture Addressing Unit (TA) are busy in the shader engine(s).
grbm_tc_busy	Cycles	Any of the Texture Cache Blocks (TCP/TCI/TCA/TCC) are busy.
grbm_cpc_busy	Cycles	The Command Processor - Compute (CPC) is busy.
grbm_cpf_busy	Cycles	The Command Processor - Fetcher (CPF) is busy.
grbm_utcl2_busy	Cycles	The Unified Translation Cache - Level 2 (UTCL2) block is busy.
grbm_ea_busy	Cycles	The Efficiency Arbiter (EA) block is busy.

30.5.1.2 Command Processor (CP)

The command processor counters are further classified into fetcher and compute.

30.5.1.2.1 Command Processor - Fetcher (CPF)

30.5.1.2.1.1 CPF Counters

Hardware Counter	Unit	Definition
cpf_cmp_utcl1_stall_on_translatic	Cycles	One of the Compute UTCL1s is stalled waiting on translation.
cpf_cpf_stat_idle[*]	Cycles	CPF idle
cpf_cpf_stat_stall	Cycles	CPF stall
cpf_cpf_tciu_busy	Cycles	CPF TCIU interface busy
cpf_cpf_tciu_idle	Cycles	CPF TCIU interface idle
cpf_cpf_tciu_stall[*]	Cycles	CPF TCIU interface is stalled waiting on free tags.

30.5.1.2.2 Command Processor - Compute (CPC)

30.5.1.2.2.1 CPC Counters

Hardware Counter	Unit	Definition
cpc_me1_busy_for_packet_decode	Cycles	CPC ME1 busy decoding packets
cpc_utcl1_stall_on_translation	Cycles	One of the UTCL1s is stalled waiting on translation
cpc_cpc_stat_busy	Cycles	CPC busy
cpc_cpc_stat_idle	Cycles	CPC idle
cpc_cpc_stat_stall	Cycles	CPC stalled
cpc_cpc_tciu_busy	Cycles	CPC TCIU interface busy
cpc_cpc_tciu_idle	Cycles	CPC TCIU interface idle
cpc_cpc_utcl2iu_busy	Cycles	CPC UTCL2 interface busy
cpc_cpc_utcl2iu_idle	Cycles	CPC UTCL2 interface idle
cpc_cpc_utcl2iu_stall[*]	Cycles	CPC UTCL2 interface stalled waiting
cpc_me1_dci0_spi_busy	Cycles	CPC ME1 Processor busy

30.5.1.3 Shader Processor Input (SPI)

30.5.1.3.1 SPI Counters

Hardware Counter	Unit	Definition
spi_csn_busy	Cycles	Number of clocks with outstanding waves
spi_csn_window_valid	Cycles	Clock count enabled by perfcounter_start event
spi_csn_num_threadgrou	Work-groups	Total number of dispatched workgroups
spi_csn_wave	Wave-fronts	Total number of dispatched wavefronts
spi_ra_req_no_alloc	Cycles	Arb cycles with requests but no allocation (need to multiply this value by 4)
spi_ra_req_no_alloc_csi	Cycles	Arb cycles with CSn req and no CSn alloc (need to multiply this value by 4)
spi_ra_res_stall_csn	Cycles	Arb cycles with CSn req and no CSn fits (need to multiply this value by 4)
spi_ra_tmp_stall_csn[*]	Cycles	Cycles where CSn wants to req but does not fit in temp space
spi_ra_wave_simd_full_	SIMD-cycles	Sum of SIMD where WAVE cannot take csn wave when not fits
spi_ra_vgpr_simd_full_	SIMD-cycles	Sum of SIMD where VGPR cannot take csn wave when not fits
spi_ra_sgpr_simd_full_(	SIMD-cycles	Sum of SIMD where SGPR cannot take csn wave when not fits
spi_ra_lds_cu_full_csn	CUs	Sum of CU where LDS cannot take csn wave when not fits
spi_ra_bar_cu_full_csn[	CUs	Sum of CU where BARRIER cannot take csn wave when not fits
spi_ra_bulky_cu_full_cs	CUs	Sum of CU where BULKY cannot take csn wave when not fits
spi_ra_tglim_cu_full_cs	Cycles	Cycles where csn wants to req but all CUs are at tg_limit
spi_ra_wvlim_cu_full_c	Cycles	Number of clocks csn is stalled due to WAVE LIMIT
spi_vwc_csc_wr	Cycles	Number of clocks to write CSC waves to VGPRs (need to multiply this value by 4)
spi_swc_csc_wr	Cycles	Number of clocks to write CSC waves to SGPRs (need to multiply this value by 4)

30.5.1.4 Compute Unit

The compute unit counters are further classified into instruction mix, MFMA operation counters, level counters, wavefront counters, wavefront cycle counters, local data share counters, and others.

30.5.1.4.1 Instruction Mix

Hardware Counter	Unit	Definition
sq_insts	Instr	Number of instructions issued
sq_insts_valu	Instr	Number of VALU instructions issued, including MFMA
sq_insts_valu_add_f16	Instr	Number of VALU F16 Add instructions issued
sq_insts_valu_mul_f16	Instr	Number of VALU F16 Multiply instructions issued
sq_insts_valu_fma_f16	Instr	Number of VALU F16 FMA instructions issued
sq_insts_valu_trans_f16	Instr	Number of VALU F16 Transcendental instructions issued

Table 30.2 – continued from previous page

Hardware Counter	Unit	Definition
sq_insts_valu_add_f32	Instr	Number of VALU F32 Add instructions issued
sq_insts_valu_mul_f32	Instr	Number of VALU F32 Multiply instructions issued
sq_insts_valu_fma_f32	Instr	Number of VALU F32 FMA instructions issued
sq_insts_valu_trans_f32	Instr	Number of VALU F32 Transcendental instructions issued
sq_insts_valu_add_f64	Instr	Number of VALU F64 Add instructions issued
sq_insts_valu_mul_f64	Instr	Number of VALU F64 Multiply instructions issued
sq_insts_valu_fma_f64	Instr	Number of VALU F64 FMA instructions issued
sq_insts_valu_trans_f64	Instr	Number of VALU F64 Transcendental instructions issued
sq_insts_valu_int32	Instr	Number of VALU 32-bit integer instructions issued (signed or unsigned)
sq_insts_valu_int64	Instr	Number of VALU 64-bit integer instructions issued (signed or unsigned)
sq_insts_valu_cvt	Instr	Number of VALU Conversion instructions issued
sq_insts_valu_mfma_i8	Instr	Number of 8-bit Integer MFMA instructions issued
sq_insts_valu_mfma_f16	Instr	Number of F16 MFMA instructions issued
sq_insts_valu_mfma_bf16	Instr	Number of BF16 MFMA instructions issued
sq_insts_valu_mfma_f32	Instr	Number of F32 MFMA instructions issued
sq_insts_valu_mfma_f64	Instr	Number of F64 MFMA instructions issued
sq_insts_mfma	Instr	Number of MFMA instructions issued
sq_insts_vmem_wr	Instr	Number of VMEM Write instructions issued
sq_insts_vmem_rd	Instr	Number of VMEM Read instructions issued
sq_insts_vmem	Instr	Number of VMEM instructions issued, including both FLAT and Buffer instructions
sq_insts_salu	Instr	Number of SALU instructions issued
sq_insts_smem	Instr	Number of SMEM instructions issued
sq_insts_smem_norm	Instr	Number of SMEM instructions issued to normalize to match smem_level. Used in m
sq_insts_flat	Instr	Number of FLAT instructions issued
sq_insts_flat_lds_only	Instr	Number of FLAT instructions issued that read/write only from/to LDS
sq_insts_lds	Instr	Number of LDS instructions issued
sq_insts_gds	Instr	Number of GDS instructions issued
sq_insts_exp_gds	Instr	Number of EXP and GDS instructions excluding skipped export instructions issued
sq_insts_branch	Instr	Number of Branch instructions issued
sq_insts_sendmsg	Instr	Number of SENDMSG instructions including s_endpgm issued
sq_insts_vskipped[∗]	Instr	Number of VSkipped instructions issued

30.5.1.4.2 MFMA Operation Counters

Hardware Counter	Unit	Definition
sq_insts_valu_mfma_mops_I8	IOP	Number of 8-bit integer MFMA ops in unit of 512
sq_insts_valu_mfma_mops_F16	FLOP	Number of F16 floating MFMA ops in unit of 512
sq_insts_valu_mfma_mops_BF16	FLOP	Number of BF16 floating MFMA ops in unit of 512
sq_insts_valu_mfma_mops_F32	FLOP	Number of F32 floating MFMA ops in unit of 512
sq_insts_valu_mfma_mops_F64	FLOP	Number of F64 floating MFMA ops in unit of 512

30.5.1.4.3 Level Counters

Hardware Counter	Unit	Definition
sq_accum_prev	Count	Accumulated counter sample value where accumulation takes place once every four cycles
sq_accum_prev_hi	Count	Accumulated counter sample value where accumulation takes place once every cycle
sq_level_waves	Waves	Number of inflight waves
sq_insts_level_vme	Instr	Number of inflight VMEM instructions
sq_insts_level_sme	Instr	Number of inflight SMEM instructions
sq_insts_level_lds	Instr	Number of inflight LDS instructions
sq_ifetch_level	Instr	Number of inflight instruction fetches

30.5.1.4.4 Wavefront Counters

Hardware Counter	Unit	Definition
sq_waves	Waves	Number of wavefronts dispatch to SQs, including both new and restored wavefronts
sq_waves_saved[*]	Waves	Number of context-saved wavefronts
sq_waves_restored[*	Waves	Number of context-restored wavefronts
sq_waves_eq_64	Waves	Number of wavefronts with exactly 64 active threads sent to SQs
sq_waves_lt_64	Waves	Number of wavefronts with less than 64 active threads sent to SQs
sq_waves_lt_48	Waves	Number of wavefronts with less than 48 active threads sent to SQs
sq_waves_lt_32	Waves	Number of wavefronts with less than 32 active threads sent to SQs
sq_waves_lt_16	Waves	Number of wavefronts with less than 16 active threads sent to SQs

30.5.1.4.5 Wavefront Cycle Counters

Hardware Counter	Unit	Definition
sq_cycles	Cycles	Free-running SQ clocks
sq_busy_cycles	Cycles	Number of cycles while SQ reports it to be busy
sq_busy_cu_cycles	Qcycles	Number of quad-cycles each CU is busy
sq_valu_mfma_bu	Cycles	Number of cycles the MFMA ALU is busy
sq_wave_cycles	Qcycles	Number of quad-cycles spent by waves in the CUs
sq_wait_any	Qcycles	Number of quad-cycles spent waiting for anything
sq_wait_inst_any	Qcycles	Number of quad-cycles spent waiting for an issued instruction
sq_active_inst_any	Qcycles	Number of quad-cycles spent by each wave to work on an instruction
sq_active_inst_vm	Qcycles	Number of quad-cycles spent by each wave to work on a non-FLAT VMEM instruction
sq_active_inst_lds	Qcycles	Number of quad-cycles spent by each wave to work on an LDS instruction
sq_active_inst_val	Qcycles	Number of quad-cycles spent by each wave to work on a VALU instruction
sq_active_inst_sca	Qcycles	Number of quad-cycles spent by each wave to work on an SCA instruction
sq_active_inst_exp	Qcycles	Number of quad-cycles spent by each wave to work on EXP or GDS instruction
sq_active_inst_mis	Qcycles	Number of quad-cycles spent by each wave to work on an MISC instruction, including branch and sendmsg
sq_active_inst_flat	Qcycles	Number of quad-cycles spent by each wave to work on a FLAT instruction
sq_inst_cycles_vm	Qcycles	Number of quad-cycles spent to send addr and cmd data for VMEM Write instructions, including both FLAT and Buffer
sq_inst_cycles_vm	Qcycles	Number of quad-cycles spent to send addr and cmd data for VMEM Read instructions, including both FLAT and Buffer
sq_inst_cycles_sm	Qcycles	Number of quad-cycles spent to execute scalar memory reads
sq_inst_cycles_sal	Cycles	Number of cycles spent to execute non-memory read scalar operations
sq_thread_cycles_	Cycles	Number of thread-cycles spent to execute VALU operations

30.5.1.4.6 Local Data Share

Hardware Counter	Unit	Definition
sq_lds_atomic_return	Cycles	Number of atomic return cycles in LDS
sq_lds_bank_conflict	Cycles	Number of cycles LDS is stalled by bank conflicts
sq_lds_addr_conflict[*]	Cycles	Number of cycles LDS is stalled by address conflicts
sq_lds_unaligned_stalls[*	Cycles	Number of cycles LDS is stalled processing flat unaligned load/store ops
sq_lds_mem_violations[*]	Count	Number of threads that have a memory violation in the LDS

30.5.1.4.7 Miscellaneous

30.5.1.4.7.1 Local Data Share

Hardware Counter	Unit	Definition
sq_ifetch	Count	Number of fetch requests from L1I cache, in 32-byte width
sq_items	Threads	Number of valid threads

30.5.1.5 L1I and sL1D Caches

30.5.1.5.1 L1I and sL1D Caches

Hardware Counter	Unit	Definition
sqc_icache_req	Req	Number of L1I cache requests
sqc_icache_hits	Coun	Number of L1I cache lookup-hits
sqc_icache_misses	Coun	Number of L1I cache non-duplicate lookup-misses
sqc_icache_misses_d	Coun	Number of d L1I cache duplicate lookup misses whose previous lookup miss on the same cache line is not fulfilled yet
sqc_dcache_req	Req	Number of sL1D cache requests
sqc_dcache_input_va	Cy-cles	Number of cycles while SQ input is valid but sL1D cache is not ready
sqc_dcache_hits	Coun	Number of sL1D cache lookup-hits
sqc_dcache_misses	Coun	Number of sL1D non-duplicate lookup-misses
sqc_dcache_misses_d	Coun	Number of sL1D duplicate lookup-misses
sqc_dcache_req_read	Req	Number of Read requests in a single 32-bit Data Word, DWORD (DW)
sqc_dcache_req_read	Req	Number of Read requests in 2 DW
sqc_dcache_req_read	Req	Number of Read requests in 4 DW
sqc_dcache_req_read	Req	Number of Read requests in 8 DW
sqc_dcache_req_read	Req	Number of Read requests in 16 DW
sqc_dcache_atomic[*]	Req	Number of Atomic requests
sqc_tc_req	Req	Number of L2 cache requests that were issued by instruction and constant caches
sqc_tc_inst_req	Req	Number of instruction cache line requests to L2 cache
sqc_tc_data_read_re	Req	Number of data Read requests to the L2 cache
sqc_tc_data_write_re	Req	Number of data Write requests to the L2 cache
sqc_tc_data_atomic_	Req	Number of data Atomic requests to the L2 cache
sqc_tc_stall[*]	Cy-cles	Number of cycles while the valid requests to L2 Cache are stalled

30.5.1.6 Vector L1 Cache Subsystem

The vector L1 cache subsystem counters are further classified into texture addressing unit, texture data unit, vector L1D cache, and texture cache arbiter.

30.5.1.6.1 Texture Addressing Unit

30.5.1.6.1.1 Texture Addressing Unit Counters

Hardware Counter	Unit	Definition
ta_ta_busy	Cycles	TA busy cycles
ta_total_wavefronts	Instr	Number of wavefront instructions
ta_buffer_wavefronts	Instr	Number of Buffer wavefront instructions
ta_buffer_read_wavefronts	Instr	Number of Buffer Read wavefront instructions
ta_buffer_write_wavefronts	Instr	Number of Buffer Write wavefront instructions
ta_buffer_atomic_wavefronts[*]	Instr	Number of Buffer Atomic wavefront instructions
ta_buffer_total_cycles	Cycles	Number of Buffer cycles, including Read and Write
ta_buffer_coalesced_read_cycles	Cycles	Number of coalesced Buffer read cycles
ta_buffer_coalesced_write_cycles	Cycles	Number of coalesced Buffer write cycles
ta_addr_stalled_by_tc	Cycles	Number of cycles TA address is stalled by TCP
ta_data_stalled_by_tc	Cycles	Number of cycles TA data is stalled by TCP
ta_addr_stalled_by_td_cycles[*]	Cycles	Number of cycles TA address is stalled by TD
ta_flat_wavefronts	Instr	Number of Flat wavefront instructions
ta_flat_read_wavefronts	Instr	Number of Flat Read wavefront instructions
ta_flat_write_wavefronts	Instr	Number of Flat Write wavefront instructions
ta_flat_atomic_wavefronts	Instr	Number of Flat Atomic wavefront instructions

30.5.1.6.2 Texture Data Unit

30.5.1.6.2.1 Texture Data Unit Counters

Hardware Counter	Unit	Definition
td_td_busy	Cycle	TD busy cycles
td_tc_stall	Cycle	Number of cycles TD is stalled by TCP
td_spi_stall[*]	Cycle	Number of cycles TD is stalled by SPI
td_load_wavefront	Instr	Number of wavefront instructions (Read/Write/Atomic)
td_store_wavefront	Instr	Number of Write wavefront instructions
td_atomic_wavefront	Instr	Number of Atomic wavefront instructions
td_coalescable_wavefront	Instr	Number of coalescable instructions

30.5.1.6.3 Vector L1D Cache

Hardware Counter	Unit	Definition
tcp_gate_en1	Cycles	Number of cycles/ vL1D interface clocks are turned on
tcp_gate_en2	Cycles	Number of cycles vL1D core clocks are turned on
tcp_td_tcp_stall_cycles	Cycles	Number of cycles TD stalls vL1D
tcp_tcr_tcp_stall_cycles	Cycles	Number of cycles TCR stalls vL1D
tcp_read_tagconflict_stall_cycles	Cycles	Number of cycles tag RAM conflict stalls on a Read
tcp_write_tagconflict_stall_cycles	Cycles	Number of cycles tag RAM conflict stalls on a Write
tcp_atomic_tagconflict_stall_cycles	Cycles	Number of cycles tag RAM conflict stalls on an Atomic

Table 30.3 – continued from previous page

Hardware Counter	Unit	Definition
tcp_pending_stall_cycles	Cycles	Number of cycles vL1D cache is stalled due to data pending from L2 Cac
tcp_ta_tcp_state_read	Req	Number of wavefront instruction requests to vL1D
tcp_volatile[*]	Req	Number of L1 volatile pixels/buffers from TA
tcp_total_accesses	Req	Number of vL1D accesses
tcp_total_read	Req	Number of vL1D Read accesses
tcp_total_write	Req	Number of vL1D Write accesses
tcp_total_atomic_with_ret	Req	Number of vL1D Atomic with return
tcp_total_atomic_without_ret	Req	Number of vL1D Atomic without return
tcp_total_writeback_invalidates	Count	Number of vL1D Writebacks and Invalidates
tcp_utcl1_request	Req	Number of address translation requests to UTCL1
tcp_utcl1_translation_hit	Req	Number of UTCL1 translation hits
tcp_utcl1_translation_miss	Req	Number of UTCL1 translation misses
tcp_utcl1_persmission_miss	Req	Number of UTCL1 permission misses
tcp_total_cache_accesses	Req	Number of vL1D cache accesses
tcp_tcp_latency	Cycles	Accumulated wave access latency to vL1D over all wavefronts
tcp_tcc_read_req_latency	Cycles	Accumulated vL1D-L2 request latency over all wavefronts for Reads and
tcp_tcc_write_req_latency	Cycles	Accumulated vL1D-L2 request latency over all wavefronts for Writes and
tcp_tcc_read_req	Req	Number of Read requests to L2 Cache
tcp_tcc_write_req	Req	Number of Write requests to L2 Cache
tcp_tcc_atomic_with_ret_req	Req	Number of Atomic requests to L2 Cache with return
tcp_tcc_atomic_without_ret_req	Req	Number of Atomic requests to L2 Cache without return
tcp_tcc_nc_read_req	Req	Number of NC Read requests to L2 Cache
tcp_tcc_uc_read_req	Req	Number of UC Read requests to L2 Cache
tcp_tcc_cc_read_req	Req	Number of CC Read requests to L2 Cache
tcp_tcc_rw_read_req	Req	Number of RW Read requests to L2 Cache
tcp_tcc_nc_write_req	Req	Number of NC Write requests to L2 Cache
tcp_tcc_uc_write_req	Req	Number of UC Write requests to L2 Cache
tcp_tcc_cc_write_req	Req	Number of CC Write requests to L2 Cache
tcp_tcc_rw_write_req	Req	Number of RW Write requests to L2 Cache
tcp_tcc_nc_atomic_req	Req	Number of NC Atomic requests to L2 Cache
tcp_tcc_uc_atomic_req	Req	Number of UC Atomic requests to L2 Cache
tcp_tcc_cc_atomic_req	Req	Number of CC Atomic requests to L2 Cache
tcp_tcc_rw_atomic_req	Req	Number of RW Atomic requests to L2 Cache

30.5.1.6.4 Texture Cache Arbiter (TCA)

Hardware Counter	Unit	Definition
tca_cycle	Cycles	TCA cycles
tca_busy	Cycles	Number of cycles TCA has a pending request

30.5.1.7 L2 Cache Access

30.5.1.7.1 L2 Cache Access Counters

Hardware Counter	Unit	Definition
tcc_cycle	Cycle	L2 Cache free-running clocks
tcc_busy	Cycle	L2 Cache busy cycles
tcc_req	Req	Number of L2 Cache requests
tcc_streaming_req[*]	Req	Number of L2 Cache Streaming requests
tcc_NC_req	Req	Number of NC requests
tcc_UC_req	Req	Number of UC requests
tcc_CC_req	Req	Number of CC requests
tcc_RW_req	Req	Number of RW requests
tcc_probe	Req	Number of L2 Cache probe requests
tcc_probe_all[*]	Req	Number of external probe requests with EA_TCC_preq_all== 1
tcc_read_req	Req	Number of L2 Cache Read requests
tcc_write_req	Req	Number of L2 Cache Write requests
tcc_atomic_req	Req	Number of L2 Cache Atomic requests
tcc_hit	Req	Number of L2 Cache lookup-hits
tcc_miss	Req	Number of L2 cache lookup-misses
tcc_writeback	Req	Number of lines written back to main memory, including writebacks of dirt
tcc_ea_wrreq	Req	Total number of 32-byte and 64-byte Write requests to EA
tcc_ea_wrreq_64B	Req	Total number of 64-byte Write requests to EA
tcc_ea_wr_uncached_32B	Req	Number of 32-byte Write/Atomic going over the TC_EA_wrreq interface
tcc_ea_wrreq_stall	Cycles	Number of cycles a Write request was stalled
tcc_ea_wrreq_io_credit_stall[*]	Cycles	Number of cycles an EA Write request runs out of IO credits
tcc_ea_wrreq_gmi_credit_stall[*]	Cycles	Number of cycles an EA Write request runs out of GMI credits
tcc_ea_wrreq_dram_credit_stall	Cycles	Number of cycles an EA Write request runs out of DRAM credits
tcc_too_many_ea_wrreqs_stall[*]	Cycles	Number of cycles the L2 Cache reaches maximum number of pending EA W
tcc_ea_wrreq_level	Req	Accumulated number of L2 Cache-EA Write requests in flight
tcc_ea_atomic	Req	Number of 32-byte and 64-byte Atomic requests to EA
tcc_ea_atomic_level	Req	Accumulated number of L2 Cache-EA Atomic requests in flight
tcc_ea_rdreq	Req	Total number of 32-byte and 64-byte Read requests to EA
tcc_ea_rdreq_32B	Req	Total number of 32-byte Read requests to EA
tcc_ea_rd_uncached_32B	Req	Number of 32-byte L2 Cache-EA Read due to uncached traffic. A 64-byte r
tcc_ea_rdreq_io_credit_stall[*]	Cycles	Number of cycles Read request interface runs out of IO credits
tcc_ea_rdreq_gmi_credit_stall[*]	Cycles	Number of cycles Read request interface runs out of GMI credits
tcc_ea_rdreq_dram_credit_stall	Cycles	Number of cycles Read request interface runs out of DRAM credits
tcc_ea_rdreq_level	Req	Accumulated number of L2 Cache-EA Read requests in flight
tcc_ea_rdreq_dram	Req	Number of 32-byte and 64-byte Read requests to HBM
tcc_ea_wrreq_dram	Req	Number of 32-byte and 64-byte Write requests to HBM
tcc_tag_stall	Cycles	Number of cycles the normal request pipeline in the tag was stalled for any
tcc_normal_writeback	Req	Number of L2 cache normal writeback
tcc_all_tc_op_wb_writeback[*]	Req	Number of instruction-triggered writeback requests
tcc_normal_evict	Req	Number of L2 cache normal evictions
tcc_all_tc_op_inv_evict[*]	Req	Number of instruction-triggered eviction requests

30.5.2 MI200 Derived Metrics List

30.5.2.1 Derived Metrics on MI200 GPUs

Derived Metric	Description
VFetchIn sts	The average number of vector fetch instructions from the video memory executed per work-item (affected by flow control). Excludes FLAT instructions that fetch from video memory
VWrite-Insts	The average number of vector write instructions to the video memory executed per work-item (affected by flow control). Excludes FLAT instructions that write to video memory
FlatVMe	The average number of FLAT instructions that read from or write to the video memory executed per work item (affected by flow control). Includes FLAT instructions that read from or write to scratch
LDSIn-sts	The average number of LDS read/write instructions executed per work item (affected by flow control). Excludes FLAT instructions that read from or write to LDS
FlatLDS sts	The average number of FLAT instructions that read or write to LDS executed per work item (affected by flow control)
VAL-UUti-liza-tion	The percentage of active vector ALU threads in a wave. A lower number can mean either more thread divergence in a wave or that the work-group size is not a multiple of 64. Value range: 0% (bad), 100% (ideal - no thread divergence)
VAL-UBusy	The percentage of GPU time vector ALU instructions are processed. Value range: 0% (bad) to 100% (optimal)
SALUBu	The percentage of GPU time scalar ALU instructions are processed. Value range: 0% (bad) to 100% (optimal)
MemWri	The total number of effective 32B write transactions to the memory
L2Cache	The percentage of fetch, write, atomic, and other instructions that hit the data in L2 cache. Value range: 0% (no hit) to 100% (optimal)
MemU-nit-Stalled	The percentage of GPU time the memory unit is stalled. Try reducing the number or size of fetches and writes if possible. Value range: 0% (optimal) to 100% (bad)
Write-Unit-Stalled	The percentage of GPU time the write unit is stalled. Value range: 0% to 100% (bad)
LDS-BankCon flict	The percentage of GPU time LDS is stalled by bank conflicts. Value range: 0% (optimal) to 100% (bad)

30.5.3 Abbreviations

30.5.3.1 MI200 Abbreviations

Abbreviation	Meaning
ALU	Arithmetic Logic Unit
Arb	Arbiter
BF16	Brain Floating Point – 16 bits
CC	Coherently Cached

Table 30.5 –

Abbreviation	Meaning
CP	Command Processor
CPC	Command Processor – Compute
CPF	Command Processor – Fetcher
CS	Compute Shader
CSC	Compute Shader Controller
CSn	Compute Shader, the n-th pipe
CU	Compute Unit
DW	32-bit Data Word, DWORD
EA	Efficiency Arbiter
F16	Half Precision Floating Point
FLAT	FLAT instructions allow read/write/atomic access to a generic memory address pointer, which can resolve
FMA	Fused Multiply Add
GDS	Global Data Share
GRBM	Graphics Register Bus Manager
HBM	High Bandwidth Memory
Instr	Instructions
IOP	Integer Operation
L2	Level-2 Cache
LDS	Local Data Share
ME1	Micro Engine, running packet processing firmware on CPC
MFMA	Matrix Fused Multiply Add
NC	Noncoherently Cached
RW	Coherently Cached with Write
SALU	Scalar ALU
SGPR	Scalar GPR
SIMD	Single Instruction Multiple Data
sL1D	Scalar Level-1 Data Cache
SMEM	Scalar Memory
SPI	Shader Processor Input
SQ	Sequencer
TA	Texture Addressing Unit
TC	Texture Cache
TCA	Texture Cache Arbiter
TCC	Texture Cache per Channel, known as L2 Cache
TCIU	Texture Cache Interface Unit, Command Processor (CP)'s interface to memory system
TCP	Texture Cache per Pipe, known as vector L1 Cache
TCR	Texture Cache Router
TD	Texture Data Unit
UC	Uncached
UTCL1	Unified Translation Cache – Level 1
UTCL2	Unified Translation Cache – Level 2
VALU	Vector ALU
VGPR	Vector GPR
vL1D	Vector Level -1 Data Cache
VMEM	Vector Memory

30.6 AMD Instinct™ MI100 Hardware

In this chapter, we are going to briefly review hardware aspects of the AMD Instinct™ MI100 accelerators and the CDNA architecture that is the foundation of these GPUs.

30.6.1 System Architecture

Fig. 30.4 shows the node-level architecture of a system that comprises two AMD EPYC™ processors and (up to) eight AMD Instinct™ accelerators. The two EPYC processors are connected to each other with the AMD Infinity™ fabric which provides a high-bandwidth (up to 18 GT/sec) and coherent links such that each processor can access the available node memory as a single shared-memory domain in a non-uniform memory architecture (NUMA) fashion. In a 2P, or dual-socket, configuration, three AMD Infinity™ fabric links are available to connect the processors plus one PCIe Gen 4 x16 link per processor can attach additional I/O devices such as the host adapters for the network fabric.

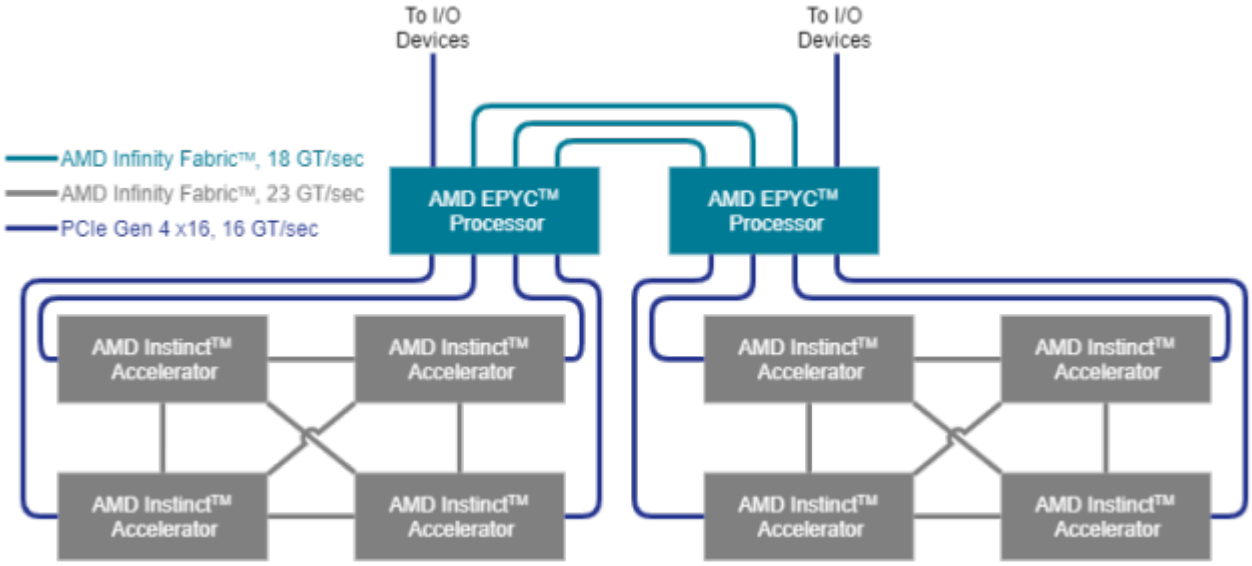


Fig. 30.4: Structure of a single GCD in the AMD Instinct MI100 accelerator.

In a typical node configuration, each processor can host up to four AMD Instinct™ accelerators that are attached using PCIe Gen 4 links at 16 GT/sec, which corresponds to a peak bidirectional link bandwidth of 32 GB/sec. Each hive of four accelerators can participate in a fully connected, coherent AMD Instinct™ fabric that connects the four accelerators using 23 GT/sec AMD Infinity fabric links that run at a higher frequency than the inter-processor links. This inter-GPU link can be established in certified server systems if the GPUs are mounted in neighboring PCIe slots by installing the AMD Infinity Fabric™ bridge for the AMD Instinct™ accelerators.

30.6.2 Micro-architecture

The micro-architecture of the AMD Instinct accelerators is based on the AMD CDNA architecture, which targets compute applications such as high-performance computing (HPC) and AI & machine learning (ML) that run on everything from individual servers to the world's largest exascale supercomputers. The overall system architecture is designed for extreme scalability and compute performance.

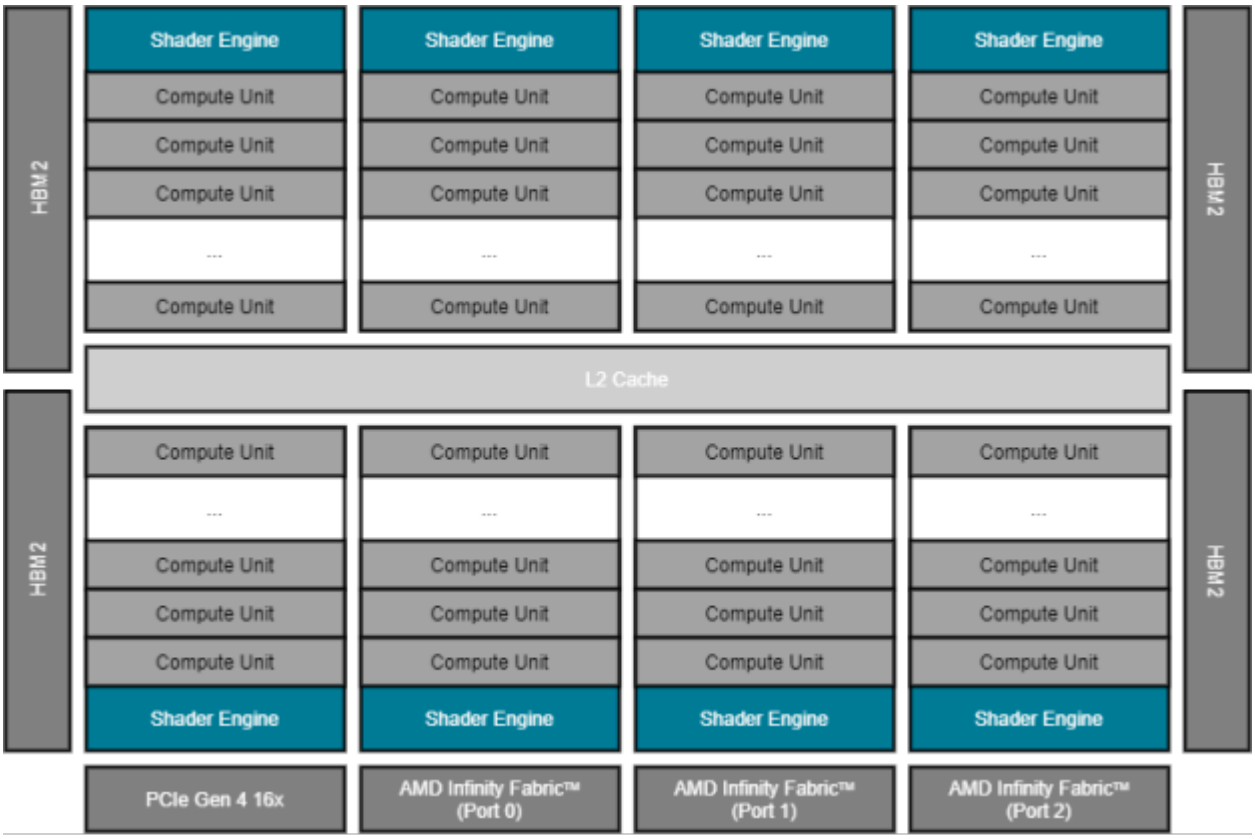


Fig. 30.5: Structure of the AMD Instinct accelerator (MI100 generation).

Fig. 30.5 shows the AMD Instinct accelerator with its PCIe Gen 4 x16 link (16 GT/sec, at the bottom) that connects the GPU to (one of) the host processor(s). It also shows the three AMD Infinity Fabric ports that provide high-speed links (23 GT/sec, also at the bottom) to the other GPUs of the local hive as shown in Fig. 30.4.

On the left and right of the floor plan, the High Bandwidth Memory (HBM) attaches via the GPU's memory controller. The MI100 generation of the AMD Instinct accelerator offers four stacks of HBM generation 2 (HBM2) for a total of 32GB with a 4,096bit-wide memory interface. The peak memory bandwidth of the attached HBM2 is 1.228 TB/sec at a memory clock frequency of 1.2 GHz.

The execution units of the GPU are depicted in Fig. 30.5 as Compute Units (CU). There are a total 120 compute units that are physically organized into eight Shader Engines (SE) with fifteen compute units per shader engine. Each compute unit is further sub-divided into four SIMD units that process SIMD instructions of 16 data elements per instruction. This enables the CU to process 64 data elements (a so-called 'wavefront') at a peak clock frequency of 1.5 GHz. Therefore, the theoretical maximum FP64 peak performance is 11.5 TFLOPS (4 [SIMD units] x 16 [elements per instruction] x 120 [CU] x 1.5 [GHz]).

Fig. 30.6 shows the block diagram of a single CU of an AMD Instinct™ MI100 accelerator and summarizes how instructions flow through the execution engines. The CU fetches the instructions via a 32KB instruction

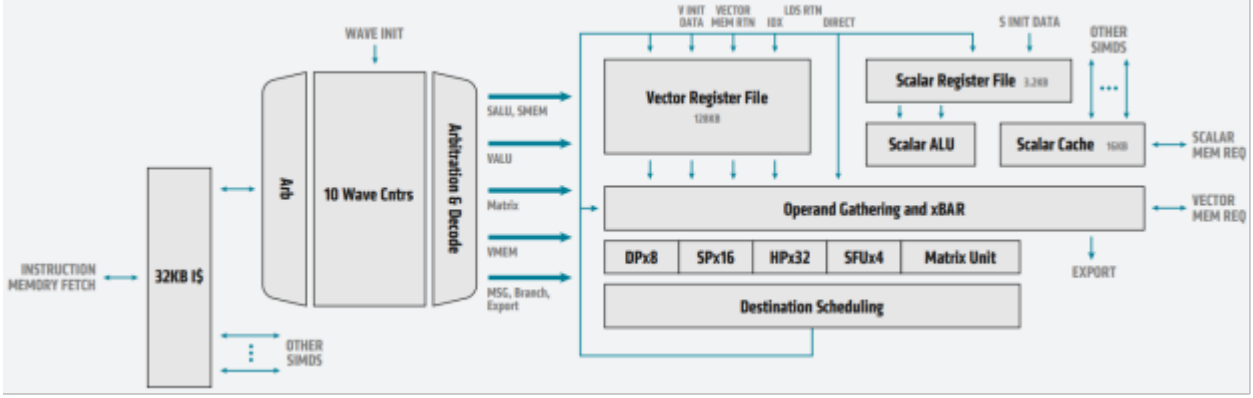


Fig. 30.6: Block diagram of an MI100 compute unit with detailed SIMD view of the AMD CDNA architecture

cache and moves them forward to execution via a dispatcher. The CU can handle up to ten wavefronts at a time and feed their instructions into the execution unit. The execution unit contains 256 vector general-purpose registers (VGPR) and 800 scalar general-purpose registers (SGPR). The VGPR and SGPR are dynamically allocated to the executing wavefronts. A wavefront can access a maximum of 102 scalar registers. Excess scalar-register usage will cause register spilling and thus may affect execution performance.

A wavefront can occupy any number of VGPRs from 0 to 256, directly affecting occupancy; that is, the number of concurrently active wavefronts in the CU. For instance, with 119 VGPRs used, only two wavefronts can be active in the CU at the same time. With the instruction latency of four cycles per SIMD instruction, the occupancy should be as high as possible such that the compute unit can improve execution efficiency by scheduling instructions from multiple wavefronts.

Table 30.6: Peak-performance capabilities of MI100 for different data types.

Computation and Data Type	FLOPS/CLOCK/CU	Peak TFLOPS
Vector FP64	64	11.5
Matrix FP32	256	46.1
Vector FP32	128	23.1
Matrix FP16	1024	184.6
Matrix BF16	512	92.3

CHAPTER

THIRTYONE

USING THE LLVM ADDRESS SANITIZER (ASAN) ON THE GPU

The LLVM Address Sanitizer provides a process that allows developers to detect runtime addressing errors in applications and libraries. The detection is achieved using a combination of compiler-added instrumentation and runtime techniques, including function interception and replacement.

Until now, the LLVM Address Sanitizer process was only available for traditional purely CPU applications. However, ROCm has extended this mechanism to additionally allow the detection of some addressing errors on the GPU in heterogeneous applications. Ideally, developers should treat heterogeneous HIP and OpenMP applications exactly like pure CPU applications. However, this simplicity has not been achieved yet.

This document provides documentation on using ROCm Address Sanitizer. For information about LLVM Address Sanitizer, see the LLVM documentation.

CHAPTER

THIRTYTWO

COMPILE FOR ADDRESS SANITIZER

The address sanitizer process begins by compiling the application of interest with the address sanitizer instrumentation.

Recommendations for doing this are:

- Compile as many application and dependent library sources as possible using an AMD-built clang-based compiler such as amdclang++.
- Add the following options to the existing compiler and linker options:
  - -fsanitize=address - enables instrumentation
  - -shared-libsan - use shared version of runtime
  - -g - add debug info for improved reporting
- Explicitly use xnack+ in the offload architecture option. For example, --offload-arch=gfx90a:xnack+ Other architectures are allowed, but their device code will not be instrumented and a warning will be emitted.

It is not an error to compile some files without address sanitizer instrumentation, but doing so reduces the ability of the process to detect addressing errors. However, if the main program "a.out" does not directly depend on the Address Sanitizer runtime (libclang_rt.asan-x86_64.so) after the build completes (check by running ldd (List Dynamic Dependencies) or readelf), the application will immediately report an error at runtime as described in the next section.

32.1 About Compilation Time

When -fsanitize=address is used, the LLVM compiler adds instrumentation code around every memory operation. This added code must be handled by all of the downstream components of the compiler toolchain and results in increased overall compilation time. This increase is especially evident in the AMDGPU device compiler and has in a few instances raised the compile time to an unacceptable level.

There are a few options if the compile time becomes unacceptable:

- Avoid instrumentation of the files which have the worst compile times. This will reduce the effectiveness of the address sanitizer process.
- Add the option -fsanitize-recover=address to the compiles with the worst compile times. This option simplifies the added instrumentation resulting in faster compilation. See below for more information.
- Disable instrumentation on a per-function basis by adding ___attribute___((no_sanitize("address"))) to functions found to be responsible for the large compile time. Again, this will reduce the effectiveness of the process.

CHAPTER

THIRTYTHREE

USE AMD SUPPLIED ADDRESS SANITIZER INSTRUMENTED LIBRARIES

ROCm releases provide optional packages containing address sanitizer instrumented builds of a subset of those ROCm libraries usually found in /opt/rocm-<version>/lib. These optional packages are typically named -asan. However, the instrumented libraries themselves have identical names as the regular uninstrumented libraries and are located in /opt/rocm-<version>/lib/asan. It is expected that the subset of address sanitizer instrumented ROCm libraries will be expanded in future releases. They are built using the amdclang++ and hipcc compilers, while some uninstrumented libraries are built with g++. The pre-existing build options are used, but, as described above, additional options are used: -fsanitize=address, -shared-libsan and -g.

These additional libraries avoid additional developer effort to locate repositories, identify the correct branch, check out the correct tags, and other efforts needed to build the libraries from the source. And they extend the ability of the process to detect addressing errors into the ROCm libraries themselves.

When adjusting an application build to add instrumentation, linking against these instrumented libraries is unnecessary. For example, any -L /opt/rocm-<version>/lib compiler options need not be changed. However, the instrumented libraries should be used when the application is run. It is particularly important that the instrumented language runtimes, like libamdhip64.so and librocm-core.so, are used; otherwise, device invalid access detections may not be reported.

CHAPTER

THIRTYFOUR

RUNNING ADDRESS SANITIZER INSTRUMENTED APPLICATIONS

34.1 Preparing to Run an Instrumented Application

Here are a few recommendations to consider before running an address sanitizer instrumented heterogeneous application.

- Ensure the Linux kernel running on the system has Heterogeneous Memory Management (HMM) support. A kernel version of 5.6 or higher should be sufficient.
- Ensure XNACK is enabled
  - For gfx90a (MI-2X0) or gfx940 (MI-3X0) use environment HSA_XNACK = 1.
  - For gfx906 (MI-50) or gfx908 (MI-100) use environment HSA_XNACK = 1 but also ensure the amdgpu kernel module is loaded with module argument noretry=0.
    This requirement is due to the fact that the XNACK setting for these GPUs is system-wide.
- Ensure that the application will use the instrumented libraries when it runs. The output from the shell command ldd <application name> can be used to see which libraries will be used. If the instrumented libraries are not listed by ldd, the environment variable LD_LIBRARY_PATH may need to be adjusted, or in some cases an RPATH compiled into the application may need to be changed and the application recompiled.
- Ensure that the application depends on the address sanitizer runtime. This can be checked by running the command readelf -d <application name> | grep NEEDED and verifying that shared library: libclang_rt.asan-x86_64.so appears in the output. If it does not appear, when executed the application will quickly output an address sanitizer error that looks like:

```
==3210==ASan runtime does not come first in initial library list; you should either link runtime to your␣
↪application or manually preload it with LD_PRELOAD.
```

- Ensure that the application llvm-symbolizer can be executed, and that it is located in /opt/rocm-<version>/llvm/bin. This executable is not strictly required, but if found is used to translate ("symbolize") a host-side instruction address into a more useful function name, file name, and line number (assuming the application has been built to include debug information).

There is an environment variable, ASAN_OPTIONS which can be used to adjust the runtime behavior of the ASAN runtime itself. There are more than a hundred "flags" that can be adjusted (see an old list at flags) but the default settings are correct and should be used in most cases. It must be noted that these options only affect the host ASAN runtime. The device runtime only currently supports the default settings for the few relevant options.

There are two ASAN_OPTION flags of particular note.

- halt_on_error=0/1 default 1.

This tells the ASAN runtime to halt the application immediately after detecting and reporting an addressing error. The default makes sense because the application has entered the realm of undefined behavior. If the developer wishes to have the application continue anyway, this option can be set to zero. However, the application and libraries should then be compiled with the additional option -fsanitize-recover=address. Note that the ROCm optional address sanitizer instrumented libraries are not compiled with this option and if an error is detected within one of them, but halt_on_error is set to 0, more undefined behavior will occur.

- detect_leaks=0/1 default 1. This option directs the address sanitizer runtime to enable the Leak Sanitizer (LSAN). Unfortunately, for heterogeneous applications, this default will result in significant output from the leak sanitizer when the application exits due to allocations made by the language runtime which are not considered to be to be leaks. This output can be avoided by adding detect_leaks=0 to the ASAN_OPTIONS, or alternatively by producing an LSAN suppression file (syntax described here) and activating it with environment variable LSAN_OPTIONS=suppressions=/path/to/suppression/file. When using a suppression file, a suppression report is printed by default. The suppression report can be disabled by using the LSAN_OPTIONS flag print_suppressions=0.

CHAPTER

THIRTYFIVE

RUNTIME OVERHEAD

Running an address sanitizer instrumented application incurs overheads which may result in unacceptably long runtimes or failure to run at all.

35.1 Higher Execution Time

Address sanitizer detection works by checking each address at runtime before the address is actually accessed by a load, store, or atomic instruction. This checking involves an additional load to "shadow" memory which records whether the address is "poisoned" or not, and additional logic that decides whether to produce an detection report or not.

This extra runtime work can cause the application to slow down by a factor of three or more, depending on how many memory accesses are executed. For heterogeneous applications, the shadow memory must be accessible by all devices and this can mean that shadow accesses from some devices may be more costly than non-shadow accesses.

35.2 Higher Memory Use

The address checking described above relies on the compiler to surround each program variable with a red zone and on address sanitizer runtime to surround each runtime memory allocation with a red zone and fill the shadow corresponding to each red zone with poison. The added memory for the red zones is additional overhead on top of the 13% overhead for the shadow memory itself.

Applications which consume most one or more available memory pools when run normally are likely to encounter allocation failures when run with instrumentation.

CHAPTER

THIRTYSIX

RUNTIME REPORTING

It is not the intention of this document to provide a detailed explanation of all of the types of reports that can be output by the address sanitizer runtime. Instead, the focus is on the differences between the standard reports for CPU issues, and reports for GPU issues.

An invalid address detection report for the CPU always starts with

```
==<PID>==ERROR: AddressSanitizer: <problem type> on address <memory address> at pc <pc> bp <bp>␣
↪sp <sp> <access> of size <N> at <memory address> thread T0
```

and continues with a stack trace for the access, a stack trace for the allocation and deallocation, if relevant, and a dump of the shadow near the .

In contrast, an invalid address detection report for the GPU always starts with

```
==<PID>==ERROR: AddressSanitizer: <problem type> on amdgpu device <device> at pc <pc> <access> of␣
↪size <n> in workgroup id (<X>,<Y>,<Z>)
```

Above, <device> is the integer device ID, and (<X>, <Y>, <Z>) is the ID of the workgroup or block where the invalid address was detected.

While the CPU report include a call stack for the thread attempting the invalid access, the GPU is currently to a call stack of size one, i.e. the (symbolized) of the invalid access, e.g.

```
#0 <pc> in <fuction signature> at /path/to/file.hip:<line>:<column>
```

This short call stack is followed by a GPU unique section that looks like

```
Thread ids and accessed addresses:
<lid0> <maddr 0> : <lid1> <maddr1> : ...
```

where each <lid j> <maddr j> indicates the lane ID and the invalid memory address held by lane j of the wavefront attempting the invalid access.

Additionally, reports for invalid GPU accesses to memory allocated by GPU code via malloc or new starting with, for example,

```
==1234==ERROR: AddressSanitizer: heap-buffer-overflow on amdgpu device 0 at pc 0x7fa9f5c92dcc
```

or

```
==5678==ERROR: AddressSanitizer: heap-use-after-free on amdgpu device 3 at pc 0x7f4c10062d74
```

currently may include one or two surprising CPU side tracebacks mentioning :hostcall". This is due to how malloc and free are implemented for GPU code and these call stacks can be ignored.

CHAPTER

THIRTYSEVEN

RUNNING WITH ROCGDB

rocgdb can be used to further investigate address sanitizer detected errors, with some preparation.

Currently, the address sanitizer runtime complains when starting rocgdb without preparation.

```
$ rocgdb my_app
==1122==ASan` runtime does not come first in initial library list; you should either link runtime to your␣
↪application or manually preload it with LD_PRELOAD.
```

This is solved by setting environment variable LD_PRELOAD to the path to the address sanitizer runtime, whose path can be obtained using the command

```
amdclang++ -print-file-name=libclang_rt.asan-x86_64.so
```

It is also recommended to set the environment variable HIP_ENABLE_DEFERRED_LOADING=0 before debugging HIP applications.

After starting rocgdb breakpoints can be set on the address sanitizer runtime error reporting entry points of interest. For example, if an address sanitizer error report includes

```
WRITE of size 4 in workgroup id (10,0,0)
```

the rocgdb command needed to stop the program before the report is printed is

```
(gdb) break ___asan_report_store4
```

Similarly, the appropriate command for a report including

```
READ of size <N> in workgroup ID (1,2,3)
```

is

```
(gdb) break ___asan_report_load<N>
```

It is possible to set breakpoints on all address sanitizer report functions using these commands:

```
$ rocgdb <path to application>
(gdb) start <commmand line arguments>
(gdb) rbreak ^___asan_report
(gdb) c
```

CHAPTER

THIRTYEIGHT

USING ADDRESS SANITIZER WITH A SHORT HIP APPLICATION (LINK NEEDED HERE)

CHAPTER

THIRTYNINE

KNOWN ISSUES WITH USING GPU SANITIZER

- Red zones must have limited size and it is possible for an invalid access to completely miss a red zone and not be detected.
- Lack of detection or false reports can be caused by the runtime not properly maintaining red zone shadows.
- Lack of detection on the GPU might also be due to the implementation not instrumenting accesses to all GPU specific address spaces. For example, in the current implementation accesses to “private” or “stack” variables on the GPU are not instrumented, and accesses to HIP shared variables (also known as “local data store” or “LDS”) are also not instrumented.
- It can also be the case that a memory fault is hit for an invalid address even with the instrumentation. This is usually caused by the invalid address being so wild that its shadow address is outside of any memory region, and the fault actually occurs on the access to the shadow address. It is also possible to hit a memory fault for the NULL pointer. While address 0 does have a shadow location, it is not poisoned by the runtime.

CHAPTER

FORTY

HOW ROCM USES PCIE ATOMICS

40.1 ROCm PCIe Feature and Overview BAR Memory

ROCm is an extension of HSA platform architecture, so it shares the queueing model, memory model, signaling and synchronization protocols. Platform atomics are integral to perform queuing and signaling memory operations where there may be multiple-writers across CPU and GPU agents.

The full list of HSA system architecture platform requirements are here: HSA Sys Arch Features.

The ROCm Platform uses the new PCI Express 3.0 (PCIe 3.0) features for Atomic Read-Modify-Write Transactions which extends inter-processor synchronization mechanisms to IO to support the defined set of HSA capabilities needed for queuing and signaling memory operations.

The new PCIe atomic operations operate as completers for CAS (Compare and Swap), FetchADD, SWAP atomics. The atomic operations are initiated by the I/O device which support 32-bit, 64-bit and 128-bit operand which target address have to be naturally aligned to operation sizes.

For ROCm the Platform atomics are used in ROCm in the following ways:

- Update HSA queue's read_dispatch_id: 64 bit atomic add used by the command processor on the GPU agent to update the packet ID it processed.
- Update HSA queue's write_dispatch_id: 64 bit atomic add used by the CPU and GPU agent to support multi-writer queue insertions.
- Update HSA Signals – 64bit atomic ops are used for CPU & GPU synchronization.

The PCIe 3.0 atomic operations feature allows atomic transactions to be requested by, routed through and completed by PCIe components. Routing and completion does not require software support. Component support for each is detectable via the DEVCAP2 register. Upstream bridges need to have atomic operations routing enabled or the Atomic Operations will fail even though PCIe endpoint and PCIe I/O Devices has the capability to Atomics Operations.

To do atomic operations routing capability between two or more Root Ports, each associated Root Port must indicate that capability via the atomic operations routing supported bit in the Device Capabilities 2 register.

If your system has a PCIe Express Switch it needs to support atomic operations routing. Atomic operations requests are permitted only if a component's DEVCTL2.ATOMICOP_REQUESTER_ENABLE field is set. These requests can only be serviced if the upstream components support atomic operations completion and/or routing to a component which does. Atomic operations routing support=1, routing is supported; Atomic operations routing support=0, routing is not supported.

Atomic Operation is a Non-Posted transaction supporting 32-bit and 64-bit address formats, there must be a response for Completion containing the result of the operation. Errors associated with the operation (uncorrectable error accessing the target location or carrying out the Atomic operation) are signaled to the

requester by setting the Completion Status field in the completion descriptor, they are set to to Completer Abort (CA) or Unsupported Request (UR).

To understand more about how PCIe Atomic operations work PCIe Atomics

Linux Kernel Patch to pci_enable_atomic_request

There are also a number of papers which talk about these new capabilities:

- Atomic Read Modify Write Primitives by Intel
- PCI express 3 Accelerator Whitepaper by Intel
- Intel PCIe Generation 3 Hotchips Paper
- PCIe Generation 4 Base Specification includes Atomics Operation

Other I/O devices with PCIe Atomics support

- Mellanox ConnectX-5 InfiniBand Card
- Cray Aries Interconnect
- Xilinx PCIe Ultrascale Whitepaper
- Xilinx 7 Series Devices

Future bus technology with richer I/O Atomics Operation Support

- GenZ

New PCIe Endpoints with support beyond AMD Ryzen and EPYC CPU; Intel Haswell or newer CPU's with PCIe Generation 3.0 support.

- Mellanox Bluefield SOC
- Cavium Thunder X2

In ROCm, we also take advantage of PCIe ID based ordering technology for P2P when the GPU originates two writes to two different targets:

1. write to another GPU memory,

2. then write to system memory to indicate transfer complete.

They are routed off to different ends of the computer but we want to make sure the write to system memory to indicate transfer complete occurs AFTER P2P write to GPU has complete.

40.1.1 BAR Memory Overview

On a Xeon E5 based system in the BIOS we can turn on above 4GB PCIe addressing, if so he need to set MMIO Base address (MMIOH Base) and Range (MMIO High Size) in the BIOS.

In Supermicro system in the system bios you need to see the following

- Advanced->PCIe/PCI/PnP configuration-> Above 4G Decoding = Enabled
- Advanced->PCIe/PCI/PnP Configuration->MMIOH Base = 512G
- Advanced->PCIe/PCI/PnP Configuration->MMIO High Size = 256G

When we support Large Bar Capability there is a Large Bar VBIOS which also disable the IO bar.

For GFX9 and Vega10 which have Physical Address up 44 bit and 48 bit Virtual address.

- BAR0-1 registers: 64bit, prefetchable, GPU memory. 8GB or 16GB depending on Vega10 SKU. Must be placed < 2^44 to support P2P access from other Vega10.
- BAR2-3 registers: 64bit, prefetchable, Doorbell. Must be placed < 2^44 to support P2P access from other Vega10.
- BAR4 register: Optional, not a boot device.
- BAR5 register: 32bit, non-prefetchable, MMIO. Must be placed < 4GB.

Here is how our BAR works on GFX 8 GPU's with 40 bit Physical Address Limit

```
11:00.0 Display controller: Advanced Micro Devices, Inc. [AMD/ATI] Fiji [Radeon R9 FURY / NANO Series] (rev
↪c1)

Subsystem: Advanced Micro Devices, Inc. [AMD/ATI] Device 0b35

Flags: bus master, fast devsel, latency 0, IRQ 119

Memory at bf40000000 (64-bit, prefetchable) [size=256M]

Memory at bf50000000 (64-bit, prefetchable) [size=2M]

I/O ports at 3000 [size=256]

Memory at c7400000 (32-bit, non-prefetchable) [size=256K]

Expansion ROM at c7440000 [disabled] [size=128K]
```

Legend:

1 : GPU Frame Buffer BAR – In this example it happens to be 256M, but typically this will be size of the GPU memory (typically 4GB+). This BAR has to be placed < 2^40 to allow peer-to-peer access from other GFX8 AMD GPUs. For GFX9 (Vega GPU) the BAR has to be placed < 2^44 to allow peer-to-peer access from other GFX9 AMD GPUs.

2 : Doorbell BAR – The size of the BAR is typically will be < 10MB (currently fixed at 2MB) for this generation GPUs. This BAR has to be placed < 2^40 to allow peer-to-peer access from other current generation AMD GPUs.

3 : IO BAR - This is for legacy VGA and boot device support, but since this the GPUs in this project are not VGA devices (headless), this is not a concern even if the SBIOS does not setup.

4 : MMIO BAR – This is required for the AMD Driver SW to access the configuration registers. Since the reminder of the BAR available is only 1 DWORD (32bit), this is placed < 4GB. This is fixed at 256KB.

5 : Expansion ROM – This is required for the AMD Driver SW to access the GPU's video-bios. This is currently fixed at 128KB.

For more information, you can review Overview of Changes to PCI Express 3.0.

CHAPTER

FORTYONE

ALL HOW-TO MATERIAL

Tuning Guides Use case-specific system setup and tuning guides.

Deep Learning Guide Installation of various Deep Learning frameworks and applications.

GPU-Enabled MPI This chapter exemplifies how to set up Open MPI with the ROCm platform.

System Debugging Guide Useful commands to debug misbehaving ROCm installations.

CHAPTER

FORTYTWO

TUNING GUIDES

Use case-specific system setup and tuning guides.

42.1 High Performance Computing

High Performance Computing (HPC) workloads have unique requirements. The default hardware and BIOS configurations for OEM platforms may not provide optimal performance for HPC workloads. To enable optimal HPC settings on a per-platform and per-workload level, this guide calls out:

- BIOS settings that can impact performance
- Hardware configuration best practices
- Supported versions of operating systems
- Workload-specific recommendations for optimal BIOS and operating system settings

There is also a discussion on the AMD Instinct™ software development environment, including information on how to install and run the DGEMM, STREAM, HPCG, and HPL benchmarks. This guidance provides a good starting point but is not exhaustively tested across all compilers.

Prerequisites to understanding this document and to performing tuning of HPC applications include:

- Experience in configuring servers
- Administrative access to the server's Management Interface (BMC)
- Administrative access to the operating system
- Familiarity with the OEM server's BMC (strongly recommended)
- Familiarity with the OS specific tools for configuration, monitoring, and troubleshooting (strongly recommended)

This document provides guidance on tuning systems with various AMD Instinct™ accelerators for HPC workloads. This document is not an all-inclusive guide, and some items referred to may have similar, but different, names in various OEM systems (for example, OEM-specific BIOS settings). This document also provides suggestions on items that should be the initial focus of additional, application-specific tuning.

This document is based on the AMD EPYC™ 7003-series processor family (former codename "Milan").

While this guide is a good starting point, developers are encouraged to perform their own performance testing for additional tuning.

AMD Instinct™ MI200 This chapter goes through how to configure your AMD Instinct™ MI200 accelerated compute nodes to get the best performance out of them.

- Instruction Set Architecture

- Whitepaper
- Guide

AMD Instinct™ MI100 This chapter briefly reviews hardware aspects of the AMD Instinct™ MI100 accelerators and the CDNA™ 1 architecture that is the foundation of these GPUs.

- Instruction Set Architecture
- Whitepaper
- Guide

42.2 Workstation

Workstation workloads, much like High Performance Computing have a unique set of requirements, a blend of both graphics and compute, certification, stability and the list continues.

The document covers specific software requirements and processes needed to use these GPUs for Single Root I/O Virtualization (SR-IOV) and Machine Learning (ML).

The main purpose of this document is to help users utilize the RDNA 2 GPUs to their full potential.

AMD Radeon™ PRO W6000 and V620 This chapter describes the AMD GPUs with RDNA™ 2 architecture, namely AMD Radeon PRO W6800 and AMD Radeon PRO V620

- AMD RDNA2 Instruction Set Architecture
- Whitepaper
- Guide

42.3 MI200 High Performance Computing and Tuning Guide

42.3.1 System Settings

This chapter reviews system settings that are required to configure the system for AMD Instinct MI250 accelerators and improve the performance of the GPUs. It is advised to configure the system for the best possible host configuration according to the "High Performance Computing (HPC) Tuning Guide for AMD EPYC 7003 Series Processors."

Configure the system BIOS settings as explained in System BIOS Settings and enact the below given settings via the command line as explained in Operating System Settings:

- Core C states
- IOMMU (if needed)

42.3.1.1 System BIOS Settings

For maximum MI250 GPU performance on systems with AMD EPYC™ 7003-series processors (codename "Milan") and AMI System BIOS, the following configuration of system BIOS settings has been validated. These settings must be used for the qualification process and should be set as default values for the system BIOS. Analogous settings for other non-AMI System BIOS providers could be set similarly. For systems with Intel processors, some settings may not apply or be available as listed in Table 42.1.

Table 42.1: Recom
GABYTE platform

BIOS Setting Location
Advanced / PCI Subsystem Settings
Advanced / PCI Subsystem Settings
AMD CBS / CPU Common Options
AMD CBS / CPU Common Options
AMD CBS / CPU Common Options / Performance
AMD CBS / DF Common Options / Memory Addressing
AMD CBS / DF Common Options / Memory Addressing
AMD CBS / DF Common Options / Link
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / UMC Common Options / DDR4 Common Options
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR
AMD CBS / UMC Common Options / DDR4 Common Options / DRAM Controller Configuration / DRAM Power Optio
AMD CBS / Security

42.3.1.1.1 NBIO Link Clock Frequency

The NBIOs (4x per AMD EPYC™ processor) are the serializers/deserializers (also known as "SerDes") that convert and prepare the I/O signals for the processor's 128 external I/O interface lanes (32 per NBIO).

LCLK (short for link clock frequency) controls the link speed of the internal bus that connects the NBIO silicon with the data fabric. All data between the processor and its PCIe lanes flow to the data fabric based on these LCLK frequency settings. The link clock frequency of the NBIO components need to be forced to the maximum frequency for optimal PCIe performance.

For AMD EPYC™ 7003 series processors, configuring all NBIOs to be in "Enhanced Preferred I/O" mode is sufficient to enable highest link clock frequency for the NBIO components.

42.3.1.1.2 Memory Configuration

For setting the memory addressing modes (see Table 42.1), especially the number of NUMA nodes per socket/processor (NPS), follow the guidance of the "High Performance Computing (HPC) Tuning Guide for AMD EPYC 7003 Series Processors" to provide the optimal configuration for host side computation. For most HPC workloads, NPS=4 is the recommended value.

42.3.1.2 Operating System Settings

42.3.1.2.1 CPU Core State - "C States"

There are several Core-States, or C-states that an AMD EPYC CPU can idle within:

- C0: active. This is the active state while running an application.
- C1: idle
- C2: idle and power gated. This is a deeper sleep state and will have a greater latency when moving back to the C0 state, compared to when the CPU is coming out of C1.

Disabling C2 is important for running with a high performance, low-latency network. To disable power-gating on all cores run the following on Linux systems:

```
cpupower idle-set -d 2
```

Note that the cpupower tool must be installed, as it is not part of the base packages of most Linux® distributions. The package needed varies with the respective Linux distribution.

Ubuntu

```
sudo apt install linux-tools-common
```

Red Hat Enterprise Linux

```
sudo yum install cpupowerutils
```

SUSE Linux Enterprise Server

```
sudo zypper install cpupower
```

42.3.1.2.2 AMD-IOPM-UTIL

This section applies to AMD EPYC™ 7002 processors to optimize advanced Dynamic Power Management (DPM) in the I/O logic (see NBIO description above) for performance. Certain I/O workloads may benefit from disabling this power management. This utility disables DPM for all PCI-e root complexes in the system and locks the logic into the highest performance operational mode.

Disabling I/O DPM will reduce the latency and/or improve the throughput of low-bandwidth messages for PCI-e InfiniBand NICs and GPUs. Other workloads with low-bandwidth bursty PCI-e I/O characteristics may benefit as well if multiple such PCI-e devices are installed in the system.

The actions of the utility do not persist across reboots. There is no need to change any existing firmware settings when using this utility. The "Preferred I/O" and "Enhanced Preferred I/O" settings should remain unchanged at enabled.

Tip: The recommended method to use the utility is either to create a system start-up script, for example, a one-shot systemd service unit, or run the utility when starting up a job scheduler on the system. The installer packages (see Power Management Utility) will create and enable a systemd service unit for you. This service unit is configured to run in one-shot mode. This means that even when the service unit runs as expected, the status of the service unit will show inactive. This is the expected behavior when the utility runs normally. If the service unit shows failed, the utility did not run as expected. The output in either case can be shown with the systemctl status command.

Stopping the service unit has no effect since the utility does not leave anything running. To undo the effects of the utility, disable the service unit with the systemctl disable command and reboot the system.

The utility does not have any command-line options, and it must be run with super-user permissions.

42.3.1.2.3 Systems with 256 CPU Threads - IOMMU Configuration

For systems that have 256 logical CPU cores or more (e.g., 64-core AMD EPYC™ 7763 in a dual-socket configuration and SMT enabled), setting the Input-Output Memory Management Unit (IOMMU) configuration to "disabled" can limit the number of available logical cores to 255. The reason is that the Linux® kernel disables X2APIC in this case and falls back to Advanced Programmable Interrupt Controller (APIC), which can only enumerate a maximum of 255 (logical) cores.

If SMT is enabled by setting "CCD/Core/Thread Enablement > SMT Control" to "enable", the following steps can be applied to the system to enable all (logical) cores of the system:

- In the server BIOS, set IOMMU to "Enabled".
- When configuring the Grub boot loader, add the following arguments for the Linux kernel: amd_iommu=on iommu=pt

- Update Grub to use the modified configuration:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

- Reboot the system.
- Verify IOMMU passthrough mode by inspecting the kernel log via dmesg:

```
[...]
[    0.000000] Kernel command line: [...] amd_iommu=on iommu=pt
   [...]
```

Once the system is properly configured, the AMD ROCm platform can be installed.

42.3.2 System Management

For a complete guide on how to install/manage/uninstall ROCm on Linux, refer to Deploy ROCm on Linux. For verifying that the installation was successful, refer to Verifying Kernel-mode Driver Installation and Validation Tools. Should verification fail, consult the System Debugging Guide.

42.3.2.1 Hardware Verification with ROCm

The AMD ROCm™ platform ships with tools to query the system structure. To query the GPU hardware, the rocm-smi command is available. It can show available GPUs in the system with their device ID and their respective firmware (or VBIOS) versions:

```
$ rocm-smi --showhw


======================= ROCm System Management Interface =======================
============================ Concise Hardware Info =============================
GPU  DID    GFX RAS  SDMA RAS  UMC RAS  VBIOS           BUS
0    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:29:00.0
1    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:2C:00.0
2    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:2F:00.0
3    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:32:00.0
4    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:AD:00.0
5    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:B0:00.0
6    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:B3:00.0
7    740c   ENABLED  ENABLED   ENABLED  113-D65209-063  0000:B6:00.0
================================================================================
============================ End of ROCm SMI Log ===============================
```

Fig. 42.1: rocm-smi --showhw output on an 8*MI200 system.

To see the system structure, the localization of the GPUs in the system, and the fabric connections between the system components, use:

- The first block of the output shows the distance between the GPUs similar to what the numactl command outputs for the NUMA domains of a system. The weight is a qualitative measure for the "distance" data must travel to reach one GPU from another one. While the values do not carry a special (physical) meaning, the higher the value the more hops are needed to reach the destination from the source GPU.
- The second block has a matrix named "Hops between two GPUs", where 1 means the two GPUs are directly connected with XGMI, 2 means both GPUs are linked to the same CPU socket and GPU

```
$ rocm-smi --showtopo


======================= ROCm System Management Interface =======================
=========================== Weight between two GPUs ============================
       GPU0         GPU1         GPU2         GPU3         GPU4         GPU5         GPU6         GPU7
GPU0   0            15           30           30           15           30           15           30
GPU1   15           0            30           15           30           45           30           15
GPU2   30           30           0            15           15           30           15           30
GPU3   30           15           15           0            30           15           30           45
GPU4   15           30           15           30           0            15           30           30
GPU5   30           45           30           15           15           0            30           15
GPU6   15           30           15           30           30           30           0            15
GPU7   30           15           30           45           30           15           15           0

============================= Hops between two GPUs ============================
       GPU0         GPU1         GPU2         GPU3         GPU4         GPU5         GPU6         GPU7
GPU0   0            1            1            1            1            1            1            1
GPU1   1            0            1            1            1            1            1            1
GPU2   1            1            0            1            1            1            1            1
GPU3   1            1            1            0            1            1            1            1
GPU4   1            1            1            1            0            1            1            1
GPU5   1            1            1            1            1            0            1            1
GPU6   1            1            1            1            1            1            0            1
GPU7   1            1            1            1            1            1            1            0

=========================== Link Type between two GPUs ===========================
       GPU0         GPU1         GPU2         GPU3         GPU4         GPU5         GPU6         GPU7
GPU0   0            XGMI         XGMI         XGMI         XGMI         XGMI         XGMI         XGMI
GPU1   XGMI         0            XGMI         XGMI         XGMI         XGMI         XGMI         XGMI
GPU2   XGMI         XGMI         0            XGMI         XGMI         XGMI         XGMI         XGMI
GPU3   XGMI         XGMI         XGMI         0            XGMI         XGMI         XGMI         XGMI
GPU4   XGMI         XGMI         XGMI         XGMI         0            XGMI         XGMI         XGMI
GPU5   XGMI         XGMI         XGMI         XGMI         XGMI         0            XGMI         XGMI
GPU6   XGMI         XGMI         XGMI         XGMI         XGMI         XGMI         0            XGMI
GPU7   XGMI         XGMI         XGMI         XGMI         XGMI         XGMI         XGMI         0

================================== Numa Nodes ==================================
GPU[0]          : (Topology) Numa Node: 0
GPU[0]          : (Topology) Numa Affinity: 0
GPU[1]          : (Topology) Numa Node: 0
GPU[1]          : (Topology) Numa Affinity: 0
GPU[2]          : (Topology) Numa Node: 0
GPU[2]          : (Topology) Numa Affinity: 0
GPU[3]          : (Topology) Numa Node: 0
GPU[3]          : (Topology) Numa Affinity: 0
GPU[4]          : (Topology) Numa Node: 1
GPU[4]          : (Topology) Numa Affinity: 1
GPU[5]          : (Topology) Numa Node: 1
GPU[5]          : (Topology) Numa Affinity: 1
GPU[6]          : (Topology) Numa Node: 1
GPU[6]          : (Topology) Numa Affinity: 1
GPU[7]          : (Topology) Numa Node: 1
GPU[7]          : (Topology) Numa Affinity: 1
============================ End of ROCm SMI Log ============================
```

Fig. 42.2: rocm-smi --showtopo output on an 8*MI200 system.

communications will go through the CPU, and 3 means both GPUs are linked to different CPU sockets so communications will go through both CPU sockets. This number is one for all GPUs in this case since they are all connected to each other through the Infinity Fabric links.

- The third block outputs the link types between the GPUs. This can either be "XGMI" for AMD Infinity Fabric links or "PCIE" for PCIe Gen4 links.
- The fourth block reveals the localization of a GPU with respect to the NUMA organization of the shared memory of the AMD EPYC processors.

To query the compute capabilities of the GPU devices, use rocminfo command. It lists specific details about the GPU devices, including but not limited to the number of compute units, width of the SIMD pipelines, memory information, and instruction set architecture:

For a complete list of architecture (LLVM target) names, refer to GPU OS Support.

42.3.2.2 Testing Inter-device Bandwidth

Section 42.4.2.1 showed the rocm-smi --showtopo command to show how the system structure and how the GPUs are located and connected in this structure. For more details, the rocm-bandwidth-test can run benchmarks to show the effective link bandwidth between the components of the system.

The ROCm Bandwidth Test program can be installed with the following package-manager commands:

Ubuntu

```
sudo apt install rocm-bandwidth-test
```

Red Hat Enterprise Linux

```
sudo yum install rocm-bandwidth-test
```

SUSE Linux Enterprise Server

```
sudo zypper install rocm-bandwidth-test
```

Alternatively, the source code can be downloaded and built from source.

The output will list the available compute devices (CPUs and GPUs), including their device ID and PCIe ID:

The output will also show a matrix that contains a "1" if a device can communicate to another device (CPU and GPU) of the system and it will show the NUMA distance (similar to rocm-smi):

The output also contains the measured bandwidth for unidirectional and bidirectional transfers between the devices (CPU and GPU):

```
$ rocminfo
ROCk module is loaded
=====================
HSA System Attributes
=====================
Runtime Version:         1.1
[...]
==========
HSA Agents
==========
*******
Agent 1
*******
  Name:                    AMD EPYC 7763 64-Core Processor
[...]
*******
Agent 10
*******
  Name:                    gfx90a
[...]
  Device Type:             GPU
  Cache Info:
    L1:                      16(0x10) KB
    L2:                      8192(0x2000) KB
  Chip ID:                 29708(0x740c)
  Cacheline Size:          64(0x40)
  Max Clock Freq. (MHz):   1700
  BDFID:                   46592
  Internal Node ID:        9
  Compute Unit:            104
  SIMDs per CU:            4
  Shader Engines:          8
  Shader Arrs. per Eng.:   1
[...]
  Pool Info:
    Pool 1
      Segment:                 GLOBAL; FLAGS: COARSE GRAINED
      Size:                    67092480(0x3ffc000) KB
      Allocatable:             TRUE
      Alloc Granule:           4KB
      Alloc Alignment:         4KB
      Accessible by all:       FALSE
[...]
  ISA Info:
    ISA 1
      Name:                    amdgcn-amd-amdhsa--gfx90a:sramecc+:xnack-
      Machine Models:          HSA_MACHINE_MODEL_LARGE
      Profiles:                HSA_PROFILE_BASE
[...]
*** Done ***
```

Fig. 42.3: rocminfo output fragment on an 8*MI200 system.

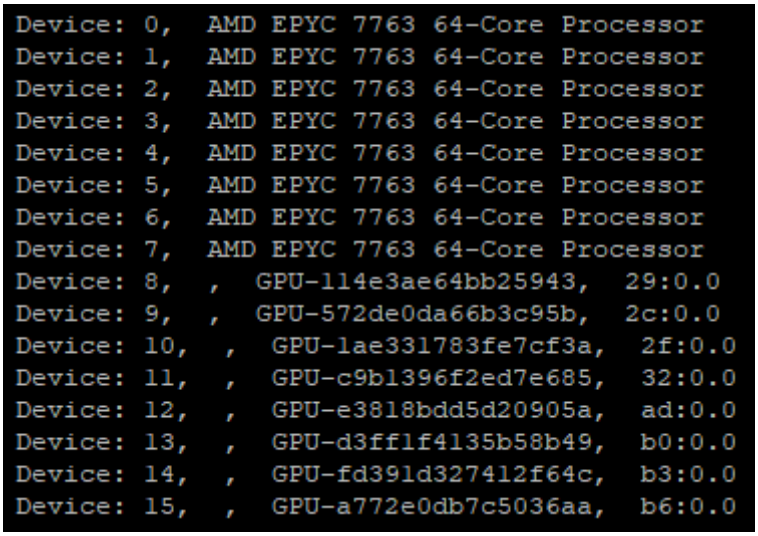


Fig. 42.4: rocm-bandwidth-test output fragment on an 8*MI200 system listing devices.

42.4 MI100 High Performance Computing and Tuning Guide

42.4.1 System Settings

This chapter reviews system settings that are required to configure the system for AMD Instinct™ MI100 accelerators and that can improve performance of the GPUs. It is advised to configure the system for best possible host configuration according to the "High Performance Computing (HPC) Tuning Guide for AMD EPYC™ 7002 Series Processors" or "High Performance Computing (HPC) Tuning Guide for AMD EPYC™ 7003 Series Processors" depending on the processor generation of the system.

In addition to the BIOS settings listed below the following settings (System BIOS Settings) will also have to be enacted via the command line (see Operating System Settings):

- Core C states
- AMD-PCI-UTIL (on AMD EPYC™ 7002 series processors)
- IOMMU (if needed)

42.4.1.1 System BIOS Settings

For maximum MI100 GPU performance on systems with AMD EPYC™ 7002 series processors (codename "Rome") and AMI System BIOS, the following configuration of System BIOS settings has been validated. These settings must be used for the qualification process and should be set as default values for the system BIOS. Analogous settings for other non-AMI System BIOS providers could be set similarly. For systems with Intel processors, some settings may not apply or be available as listed in Table 42.2.

Table 42.2: Recommended se
GABYTE platform.

BIOS Setting Location
Advanced / PCI Subsystem Settings
AMD CBS / CPU Common Options

Table 42.2 – cont

BIOS Setting Location
AMD CBS / CPU Common Options
AMD CBS / CPU Common Options / Performance
AMD CBS / DF Common Options / Memory Addressing
AMD CBS / DF Common Options / Memory Addressing
AMD CBS / DF Common Options / Link
AMD CBS / DF Common Options / Link
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / NBIO Common Options / SMU Common Options
AMD CBS / UMC Common Options / DDR4 Common Options
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR
AMD CBS / UMC Common Options / DDR4 Common Options / DRAM Controller Configuration / DRAM Power Optio
AMD CBS / Security

42.4.1.1.1 NBIO Link Clock Frequency

The NBIOs (4x per AMD EPYC™ processor) are the serializers/deserializers (also known as "SerDes") that convert and prepare the I/O signals for the processor's 128 external I/O interface lanes (32 per NBIO).

LCLK (short for link clock frequency) controls the link speed of the internal bus that connects the NBIO silicon with the data fabric. All data between the processor and its PCIe lanes flow to the data fabric based on these LCLK frequency settings. The link clock frequency of the NBIO components need to be forced to the maximum frequency for optimal PCIe performance.

For AMD EPYC™ 7002 series processors, this setting cannot be modified via configuration options in the server BIOS alone. Instead, the AMD-IOPM-UTIL (see Section 3.2.3) must be run at every server boot to disable Dynamic Power Management for all PCIe Root Complexes and NBIOs within the system and to lock the logic into the highest performance operational mode.

For AMD EPYC™ 7003 series processors, configuring all NBIOs to be in "Enhanced Preferred I/O" mode is sufficient to enable highest link clock frequency for the NBIO components.

Inter-Device Access

D/D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Inter-Device Numa Distance

D/D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	12	12	12	32	32	32	32	32	32	32	32	52	52	52	52
1	12	0	12	12	32	32	32	32	32	32	32	32	52	52	52	52
2	12	12	0	12	32	32	32	32	20	20	20	20	52	52	52	52
3	12	12	12	0	32	32	32	32	32	32	32	32	52	52	52	52
4	32	32	32	32	0	12	12	12	52	52	52	52	32	32	32	32
5	32	32	32	32	12	0	12	12	52	52	52	52	32	32	32	32
6	32	32	32	32	12	12	0	12	52	52	52	52	20	20	20	20
7	32	32	32	32	12	12	12	0	52	52	52	52	32	32	32	32
8	32	32	20	32	52	52	52	52	0	15	30	30	15	30	15	30
9	32	32	20	32	52	52	52	52	15	0	30	15	30	45	30	15
10	32	32	20	32	52	52	52	52	30	30	0	15	15	30	15	30
11	32	32	20	32	52	52	52	52	30	15	15	0	30	15	30	45
12	52	52	52	52	32	32	20	32	15	30	15	30	0	15	30	30
13	52	52	52	52	32	32	20	32	30	45	30	15	15	0	30	15
14	52	52	52	52	32	32	20	32	15	30	15	30	30	30	0	15
15	52	52	52	52	32	32	20	32	30	15	30	45	30	15	15	0

Fig. 42.5: rocm-bandwidth-test output fragment on an 8*MI200 system showing inter-device access matrix and NUMA distances.

Unidirectional copy peak bandwidth GB/s

D/D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.060	14.076	14.066	14.081	14.087	14.069	14.068	14.078
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.055	14.063	14.080	14.054	14.070	14.040	14.099	14.064
2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.063	14.064	14.068	14.062	14.069	14.069	14.089	14.075
3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.073	14.065	14.065	14.074	14.079	14.072	14.064	14.084
4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.067	14.077	14.075	14.072	14.056	14.078	14.080	14.071
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.074	14.070	14.069	14.058	14.077	14.090	14.064	14.043
6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.063	14.074	14.063	14.071	14.073	14.074	14.090	14.057
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.064	14.081	14.088	14.078	14.071	14.086	14.080	14.098
8	14.025	14.029	14.013	14.024	14.026	14.023	14.028	14.021	1052.523	152.244	37.503	36.058	78.092	71.090	39.258	37.466
9	14.025	14.016	14.023	14.032	14.029	14.017	14.023	14.025	152.078	1044.659	36.049	39.280	70.910	70.766	37.489	39.350
10	14.025	14.019	14.027	14.024	14.036	14.025	14.015	14.020	37.493	36.077	1047.267	152.133	39.225	37.496	78.281	70.994
11	14.018	14.028	14.020	14.028	14.025	14.033	14.023	14.022	36.065	39.298	152.409	1057.832	37.486	39.317	70.934	71.030
12	14.033	14.023	14.026	14.020	14.030	14.030	14.017	14.024	77.788	70.982	39.097	37.519	979.978	150.549	37.476	35.708
13	14.024	14.025	14.015	14.026	14.023	14.029	14.028	14.025	70.694	71.078	37.493	39.028	150.657	991.561	35.693	39.002
14	14.027	14.019	14.021	14.025	14.010	14.019	14.024	14.008	39.035	37.476	77.658	70.766	37.452	35.696	995.090	150.495
15	14.021	14.027	14.014	14.012	14.018	14.025	14.019	14.029	37.456	39.039	70.814	71.054	35.714	38.977	151.255	992.735

Bidirectional copy peak bandwidth GB/s

D/D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.011	23.711	25.001	24.234	24.830	24.284	24.494	24.970
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.571	24.201	24.991	24.956	24.207	25.029	24.216	24.523
2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.965	25.040	25.018	24.086	23.742	24.953	24.519	24.760
3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	25.036	24.086	24.916	24.968	24.433	24.968	24.352	24.024
4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.869	23.916	24.910	23.804	24.202	25.020	24.343	24.015
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.125	24.893	25.062	24.097	24.196	24.103	24.968	24.081
6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.997	23.728	25.004	24.926	24.314	23.874	24.936	24.119
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	24.446	25.004	24.394	23.909	24.369	23.877	24.958	23.817
8	24.011	24.571	24.965	25.036	24.869	24.125	24.997	24.446	N/A	285.619	62.601	64.016	145.434	125.147	73.372	62.859
9	23.711	24.201	25.040	24.086	23.916	24.893	23.728	25.004	285.619	N/A	64.001	74.350	124.682	124.688	64.255	73.960
10	25.001	24.991	25.018	24.916	24.910	25.062	25.004	24.394	62.601	64.001	N/A	287.577	73.148	62.926	145.567	125.035
11	24.234	24.956	24.086	24.968	23.804	24.097	24.926	23.909	64.016	74.350	287.577	N/A	64.290	74.000	124.917	124.591
12	24.830	24.207	23.742	24.433	24.202	24.196	24.314	24.369	145.434	124.682	73.148	64.290	N/A	282.540	64.239	62.789
13	24.284	25.029	24.953	24.968	25.020	24.103	23.874	23.877	125.147	124.688	62.926	74.000	282.540	N/A	62.756	73.067
14	24.494	24.216	24.519	24.352	24.343	24.968	24.936	24.958	73.372	64.255	145.567	124.917	64.239	62.756	N/A	282.070
15	24.970	24.523	24.760	24.024	24.015	24.081	24.119	23.817	62.859	73.960	125.035	124.591	62.789	73.067	282.070	N/A

Fig. 42.6: rocm-bandwidth-test output fragment on an 8*MI200 system showing uni- and bidirectional bandwidths.

42.4.1.1.2 Memory Configuration

For the memory addressing modes (see Table 42.2), especially the number of NUMA nodes per socket/processor (NPS), the recommended setting is to follow the guidance of the "High Performance Computing (HPC) Tuning Guide for AMD EPYC™ 7002 Series Processors" and "High Performance Computing (HPC) Tuning Guide for AMD EPYC™ 7003 Series Processors" to provide the optimal configuration for host side computation.

If the system is set to one NUMA domain per socket/processor (NPS1), bidirectional copy bandwidth between host memory and GPU memory may be slightly higher (up to about 16% more) than with four NUMA domains per socket processor (NPS4). For memory bandwidth sensitive applications using MPI, NPS4 is recommended. For applications that are not optimized for NUMA locality, NPS1 is the recommended setting.

42.4.1.2 Operating System Settings

42.4.1.2.1 CPU Core State - "C States"

There are several Core-States, or C-states that an AMD EPYC CPU can idle within:

- C0: active. This is the active state while running an application.
- C1: idle
- C2: idle and power gated. This is a deeper sleep state and will have a greater latency when moving back to the C0 state, compared to when the CPU is coming out of C1.

Disabling C2 is important for running with a high performance, low-latency network. To disable power-gating on all cores run the following on Linux systems:

```
cpupower idle-set -d 2
```

Note that the cpupower tool must be installed, as it is not part of the base packages of most Linux® distributions. The package needed varies with the respective Linux distribution.

Ubuntu

```
sudo apt install linux-tools-common
```

Red Hat Enterprise Linux

```
sudo yum install cpupowerutils
```

SUSE Linux Enterprise Server

```
sudo zypper install cpupower
```

42.4.1.2.2 AMD-IOPM-UTIL

This section applies to AMD EPYC™ 7002 processors to optimize advanced Dynamic Power Management (DPM) in the I/O logic (see NBIO description above) for performance. Certain I/O workloads may benefit from disabling this power management. This utility disables DPM for all PCI-e root complexes in the system and locks the logic into the highest performance operational mode.

Disabling I/O DPM will reduce the latency and/or improve the throughput of low-bandwidth messages for PCI-e InfiniBand NICs and GPUs. Other workloads with low-bandwidth bursty PCI-e I/O characteristics may benefit as well if multiple such PCI-e devices are installed in the system.

The actions of the utility do not persist across reboots. There is no need to change any existing firmware settings when using this utility. The "Preferred I/O" and "Enhanced Preferred I/O" settings should remain unchanged at enabled.

Tip: The recommended method to use the utility is either to create a system start-up script, for example, a one-shot systemd service unit, or run the utility when starting up a job scheduler on the system. The installer packages (see Power Management Utility) will create and enable a systemd service unit for you. This service unit is configured to run in one-shot mode. This means that even when the service unit runs as expected, the status of the service unit will show inactive. This is the expected behavior when the utility runs normally. If the service unit shows failed, the utility did not run as expected. The output in either case can be shown with the systemctl status command.

Stopping the service unit has no effect since the utility does not leave anything running. To undo the effects of the utility, disable the service unit with the systemctl disable command and reboot the system.

The utility does not have any command-line options, and it must be run with super-user permissions.

42.4.1.2.3 Systems with 256 CPU Threads - IOMMU Configuration

For systems that have 256 logical CPU cores or more (e.g., 64-core AMD EPYC™ 7763 in a dual-socket configuration and SMT enabled), setting the Input-Output Memory Management Unit (IOMMU) configuration to "disabled" can limit the number of available logical cores to 255. The reason is that the Linux® kernel disables X2APIC in this case and falls back to Advanced Programmable Interrupt Controller (APIC), which can only enumerate a maximum of 255 (logical) cores.

If SMT is enabled by setting "CCD/Core/Thread Enablement > SMT Control" to "enable", the following steps can be applied to the system to enable all (logical) cores of the system:

- In the server BIOS, set IOMMU to "Enabled".
- When configuring the Grub boot loader, add the following arguments for the Linux kernel: amd_iommu=on iommu=pt
- Update Grub to use the modified configuration:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

- Reboot the system.
- Verify IOMMU passthrough mode by inspecting the kernel log via dmesg:

```
[...]
[    0.000000] Kernel command line: [...] amd_iommu=on iommu=pt
   [...]
```

Once the system is properly configured, the AMD ROCm platform can be installed.

42.4.2 System Management

For a complete guide on how to install/manage/uninstall ROCm on Linux, refer to Deploy ROCm on Linux. For verifying that the installation was successful, refer to Verifying Kernel-mode Driver Installation and Validation Tools. Should verification fail, consult the System Debugging Guide.

42.4.2.1 Hardware Verification with ROCm

The AMD ROCm™ platform ships with tools to query the system structure. To query the GPU hardware, the rocm-smi command is available. It can show available GPUs in the system with their device ID and their respective firmware (or VBIOS) versions:

```
$ rocm-smi --showhw


======================== ROCm System Management Interface ========================
============================ Concise Hardware Info ===============================
GPU  DID   GFX RAS  SDMA RAS  UMC RAS  VBIOS             BUS
0    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:43:00.0
1    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:03:00.0
2    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:23:00.0
3    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:26:00.0
4    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:C3:00.0
5    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:C6:00.0
6    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:83:00.0
7    738c  ENABLED  ENABLED   ENABLED  113-D3431401-100  0000:A3:00.0
==============================================================================
============================ End of ROCm SMI Log ==============================
```

Fig. 42.7: rocm-smi --showhw output on an 8*MI100 system.

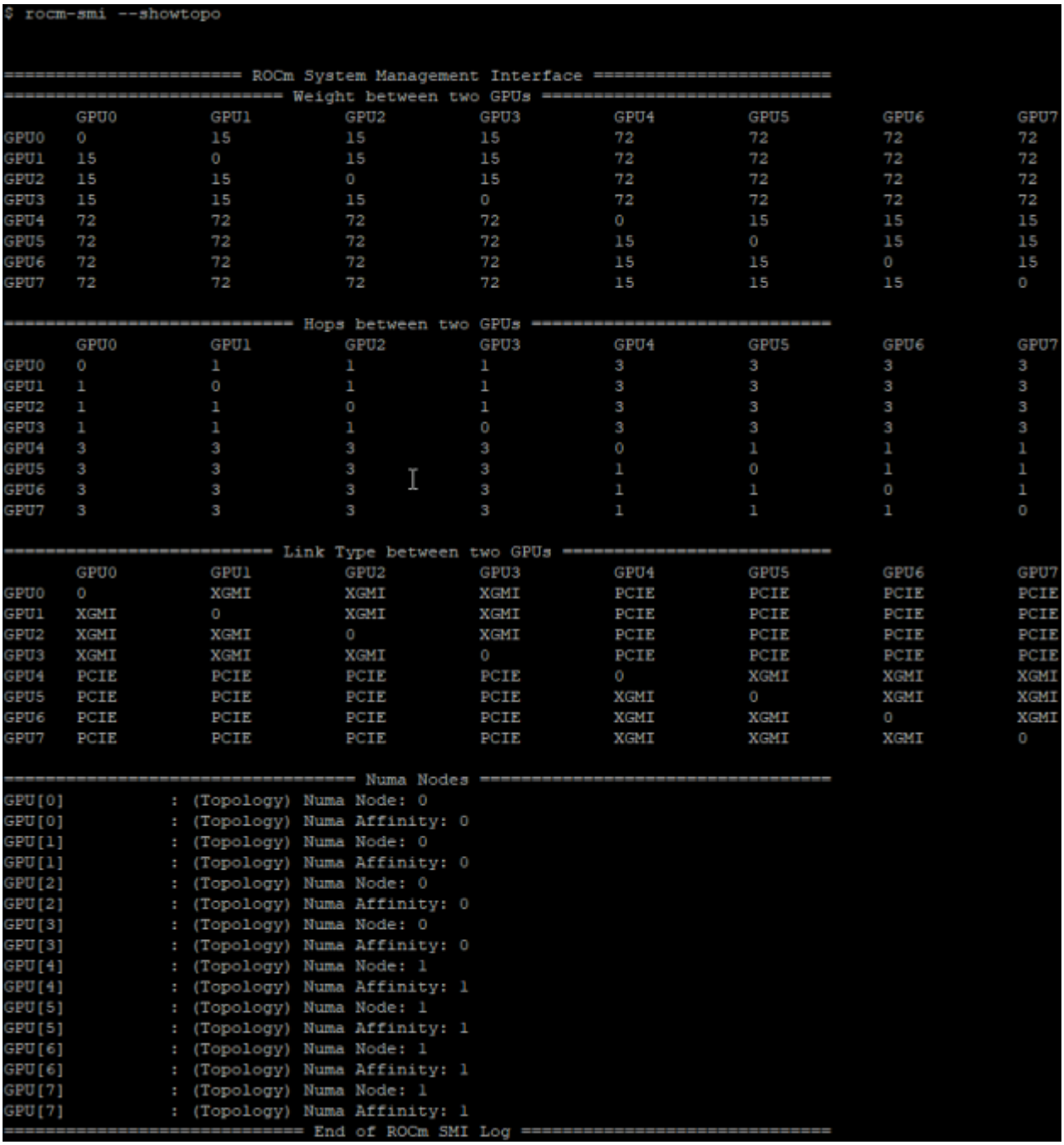
Another important query is to show the system structure, the localization of the GPUs in the system, and the fabric connections between the system components:

The previous command shows the system structure in four blocks:

- The first block of the output shows the distance between the GPUs similar to what the numactl command outputs for the NUMA domains of a system. The weight is a qualitative measure for the "distance" data must travel to reach one GPU from another one. While the values do not carry a special (physical) meaning, the higher the value the more hops are needed to reach the destination from the source GPU.
- The second block has a matrix for the number of hops required to send data from one GPU to another. For the GPUs in the local hive, this number is one, while for the others it is three (one hop to leave the hive, one hop across the processors, and one hop within the destination hive).
- The third block outputs the link types between the GPUs. This can either be "XGMI" for AMD Infinity Fabric™ links or "PCIE" for PCIe Gen4 links.
- The fourth block reveals the localization of a GPU with respect to the NUMA organization of the shared memory of the AMD EPYC™ processors.

To query the compute capabilities of the GPU devices, the rocminfo command is available with the AMD ROCm™ platform. It lists specific details about the GPU devices, including but not limited to the number of compute units, width of the SIMD pipelines, memory information, and instruction set architecture:

For a complete list of architecture (LLVM target) names, refer to GPU OS Support.



```
$ rocm-smi --showtopo


======================== ROCm System Management Interface ========================
============================ Weight between two GPUs =============================
       GPU0         GPU1         GPU2         GPU3         GPU4         GPU5         GPU6         GPU7
GPU0   0            15           15           15           72           72           72           72
GPU1   15           0            15           15           72           72           72           72
GPU2   15           15           0            15           72           72           72           72
GPU3   15           15           15           0            72           72           72           72
GPU4   72           72           72           72           0            15           15           15
GPU5   72           72           72           72           15           0            15           15
GPU6   72           72           72           72           15           15           0            15
GPU7   72           72           72           72           15           15           15           0

============================== Hops between two GPUs ==============================
       GPU0         GPU1         GPU2         GPU3         GPU4         GPU5         GPU6         GPU7
GPU0   0            1            1            1            3            3            3            3
GPU1   1            0            1            1            3            3            3            3
GPU2   1            1            0            1            3            3            3            3
GPU3   1            1            1            0            3            3            3            3
GPU4   3            3            3            3            0            1            1            1
GPU5   3            3            3            3            1            0            1            1
GPU6   3            3            3            3            1            1            0            1
GPU7   3            3            3            3            1            1            1            0

============================ Link Type between two GPUs ============================
       GPU0         GPU1         GPU2         GPU3         GPU4         GPU5         GPU6         GPU7
GPU0   0            XGMI         XGMI         XGMI         PCIE         PCIE         PCIE         PCIE
GPU1   XGMI         0            XGMI         XGMI         PCIE         PCIE         PCIE         PCIE
GPU2   XGMI         XGMI         0            XGMI         PCIE         PCIE         PCIE         PCIE
GPU3   XGMI         XGMI         XGMI         0            PCIE         PCIE         PCIE         PCIE
GPU4   PCIE         PCIE         PCIE         PCIE         0            XGMI         XGMI         XGMI
GPU5   PCIE         PCIE         PCIE         PCIE         XGMI         0            XGMI         XGMI
GPU6   PCIE         PCIE         PCIE         PCIE         XGMI         XGMI         0            XGMI
GPU7   PCIE         PCIE         PCIE         PCIE         XGMI         XGMI         XGMI         0

================================== Numa Nodes ==================================
GPU[0]          : (Topology) Numa Node: 0
GPU[0]          : (Topology) Numa Affinity: 0
GPU[1]          : (Topology) Numa Node: 0
GPU[1]          : (Topology) Numa Affinity: 0
GPU[2]          : (Topology) Numa Node: 0
GPU[2]          : (Topology) Numa Affinity: 0
GPU[3]          : (Topology) Numa Node: 0
GPU[3]          : (Topology) Numa Affinity: 0
GPU[4]          : (Topology) Numa Node: 1
GPU[4]          : (Topology) Numa Affinity: 1
GPU[5]          : (Topology) Numa Node: 1
GPU[5]          : (Topology) Numa Affinity: 1
GPU[6]          : (Topology) Numa Node: 1
GPU[6]          : (Topology) Numa Affinity: 1
GPU[7]          : (Topology) Numa Node: 1
GPU[7]          : (Topology) Numa Affinity: 1
============================ End of ROCm SMI Log ============================
```

Fig. 42.8: rocm-smi --showtopo output on an 8*MI100 system.

```
$ rocminfo
*******
Agent 10
*******
  Name:                    gfx908
[...]
  Device Type:             GPU
  Cache Info:
    L1:                      16(0x10) KB
  Chip ID:                 29580(0x738c)
  Cacheline Size:          64(0x40)
  Max Clock Freq. (MHz):   1502
  BDFID:                   41728
  Internal Node ID:        9
  Compute Unit:            120
  SIMDs per CU:            4
  Shader Engines:          8
  Shader Arrs. per Eng.:   1
[...]
  Pool Info:
    Pool 1
      Segment:                 GLOBAL; FLAGS: COARSE GRAINED
      Size:                    33538048(0x1ffc000) KB
      Allocatable:             TRUE
      Alloc Granule:           4KB
      Alloc Alignment:         4KB
      Accessible by all:       FALSE
[...]
  ISA Info:
    ISA 1
      Name:                    amdgcn-amd-amdhsa--gfx908:sramecc+:xnack-
      Machine Models:          HSA_MACHINE_MODEL_LARGE
      Profiles:                HSA_PROFILE_BASE
[...]
*** Done ***
```

Fig. 42.9: rocminfo output fragment on an 8*MI100 system.

42.4.2.2 Testing Inter-device Bandwidth

Section 42.4.2.1 showed the rocm-smi --showtopo command to show how the system structure and how the GPUs are located and connected in this structure. For more details, the rocm-bandwidth-test can run benchmarks to show the effective link bandwidth between the components of the system.

The ROCm Bandwidth Test program can be installed with the following package-manager commands:

Ubuntu

```
sudo apt install rocm-bandwidth-test
```

Red Hat Enterprise Linux

```
sudo yum install rocm-bandwidth-test
```

SUSE Linux Enterprise Server

```
sudo zypper install rocm-bandwidth-test
```

Alternatively, the source code can be downloaded and built from source.

The output will list the available compute devices (CPUs and GPUs):

```
Device: 0,  AMD EPYC 7742 64-Core Processor
Device: 1,  AMD EPYC 7742 64-Core Processor
Device: 2,  Device 738c,  03:0.0
Device: 3,  Device 738c,  23:0.0
Device: 4,  Device 738c,  26:0.0
Device: 5,  Device 738c,  53:0.0
Device: 6,  Device 738c,  83:0.0
Device: 7,  Device 738c,  a3:0.0
Device: 8,  Device 738c,  c3:0.0
Device: 9,  Device 738c,  c6:0.0
```

Fig. 42.10: rocm-bandwidth-test output fragment on an 8*MI100 system listing devices.

The output will also show a matrix that contains a “1” if a device can communicate to another device (CPU and GPU) of the system and it will show the NUMA distance (similar to rocm-smi):

The output also contains the measured bandwidth for unidirectional and bidirectional transfers between the devices (CPU and GPU):

Inter-Device Access

D/D	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1

Fig. 42.11: rocm-bandwidth-test output fragment on an 8*MI100 system showing inter-device access matrix.

Inter-Device Numa Distance

D/D	0	1	2	3	4	5	6	7	8	9
0	0	32	20	20	20	20	52	52	52	52
1	32	0	52	52	52	52	20	20	20	20
2	20	52	0	15	15	15	72	72	72	72
3	20	52	15	0	15	15	72	72	72	72
4	20	52	15	15	0	15	72	72	72	72
5	20	52	15	15	15	0	72	72	72	72
6	52	20	72	72	72	72	0	15	15	15
7	52	20	72	72	72	72	15	0	15	15
8	52	20	72	72	72	72	15	15	0	15
9	52	20	72	72	72	72	15	15	15	0

Fig. 42.12: rocm-bandwidth-test output fragment on an 8*MI100 system showing inter-device NUMA distance.

```
Device: 0,  AMD EPYC 7742 64-Core Processor
Device: 1,  AMD EPYC 7742 64-Core Processor
Device: 2,  Device 738c,  03:0.0
Device: 3,  Device 738c,  23:0.0
Device: 4,  Device 738c,  26:0.0
Device: 5,  Device 738c,  53:0.0
Device: 6,  Device 738c,  83:0.0
Device: 7,  Device 738c,  a3:0.0
Device: 8,  Device 738c,  c3:0.0
Device: 9,  Device 738c,  c6:0.0
```

Fig. 42.13: rocm-bandwidth-test output fragment on an 8*MI100 system showing uni- and bidirectional bandwidths.

42.5 RDNA2 Workstation Tuning Guide

42.5.1 System Settings

This chapter reviews system settings that are required to configure the system for ROCm virtualization on RDNA2-based AMD Radeon™ PRO GPUs. Installing ROCm on Bare Metal follows the routine ROCm installation procedure.

To enable ROCm virtualization on V620, one has to setup Single Root I/O Virtualization (SR-IOV) in the BIOS via setting found in the following (System BIOS Settings). A tested configuration can be followed in (Operating System Settings).

> Attention: SR-IOV is supported on V620 and unsupported on W6800.

42.5.1.1 System BIOS Settings

Table 42.3: Settings for the system BIOS in an ASrock platform.

Advanced / North Bridge Configuration	IOMMU	Enabled	Input-output Memory Management Unit
Advanced / North Bridge Configuration	ACS Enable	Enabled	Access Control Service
Advanced / PCIe/PCI/PnP Configuration	SR-IOV Support	Enabled	Single Root I/O Virtualization
Advanced / ACPI settings	PCI AER Support	Enabled	Advanced Error Reporting

To set up the host, update SBIOS to version 1.2a.

42.5.1.2 Operating System Settings

Table 42.4: System Configuration Prerequisites

Server	SMC 4124 [AS -4124GS-TNR]
Host OS	Ubuntu 20.04.3 LTS
Host Kernel	5.4.0-97-generic
CPU	AMD EPYC 7552 48-Core Processor
GPU	RDNA2 V620 (D603GLXE)
SBIOS	Version SMC_r_1.2a
VBIOS	113-D603GLXE-077
Guest OS 1	Ubuntu 20.04.5 LTS
Guest OS 2	RHEL 9.0
GIM Driver	gim-dkms_1.0.0.1234577_all
VM CPU Cores	32
VM RAM	64 GB

Install the following Kernel-based Virtual Machine (KVM) Hypervisor packages:

```
sudo apt-get -y install qemu-kvm qemu-utils  bridge-utils virt-manager  gir1.2-spiceclientgtk*  gir1.2-spice-client-
↪gtk* libvirt-daemon-system dnsmasq-base
sudo virsh net-start default /*to enable Virtual network by default
```

Enable IOMMU in GRUB settings by adding the following line to /etc/default/grub:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash amd_iommu=on" for AMD CPU
```

Update grub and reboot

```
sudo update=grub
sudo reboot
```

Install the GPU-IOV Module (GIM, where IOV is I/O Virtualization) driver and follow the steps below. To obtain the GIM driver, write to us here:

```
sudo dpkg -i <gim_driver>
sudo reboot
# Load Host Driver to Create 1VF
sudo modprobe gim vf_num=1
# Note: If GIM driver loaded successfully, we could see "gim info:(gim_init:213) *****Running GIM*****" in␣
↪dmesg
lspci -d 1002:
```

Which should output something like:

```
01:00.0 PCI bridge: Advanced Micro Devices, Inc. [AMD/ATI] Device 1478
02:00.0 PCI bridge: Advanced Micro Devices, Inc. [AMD/ATI] Device 1479
03:00.0 Display controller: Advanced Micro Devices, Inc. [AMD/ATI] Device 73a1
03:02.0 Display controller: Advanced Micro Devices, Inc. [AMD/ATI] Device 73ae → VF
```

42.5.1.3 Guest OS installation

First, assign GPU virtual function (VF) to VM using the following steps.

1. Shut down the VM.
2. Run virt-manager
3. In the Virtual Machine Manager GUI, select the VM and click Open.
4. In the VM GUI, go to Show Virtual Hardware Details > Add Hardware to configure hardware.
5. Go to Add Hardware > PCI Host Device > VF and click Finish.

Then start the VM.

Finally install ROCm on the virtual machine (VM). For detailed instructions, refer to the ROCm Installation Guide. For any issue encountered during installation, write to us here.

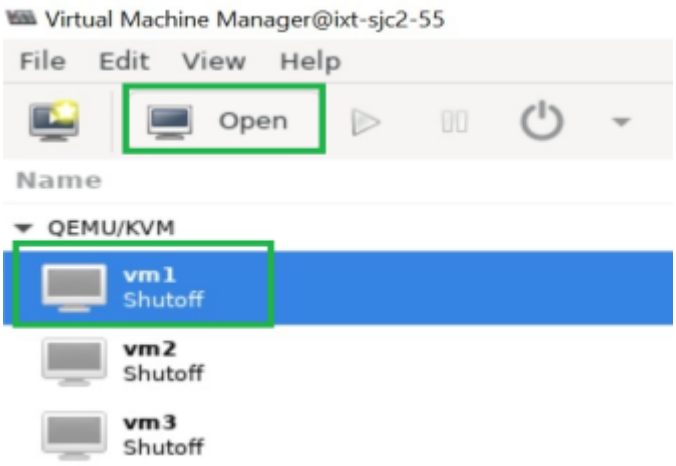


Fig. 42.14: Virtual Machine Manager

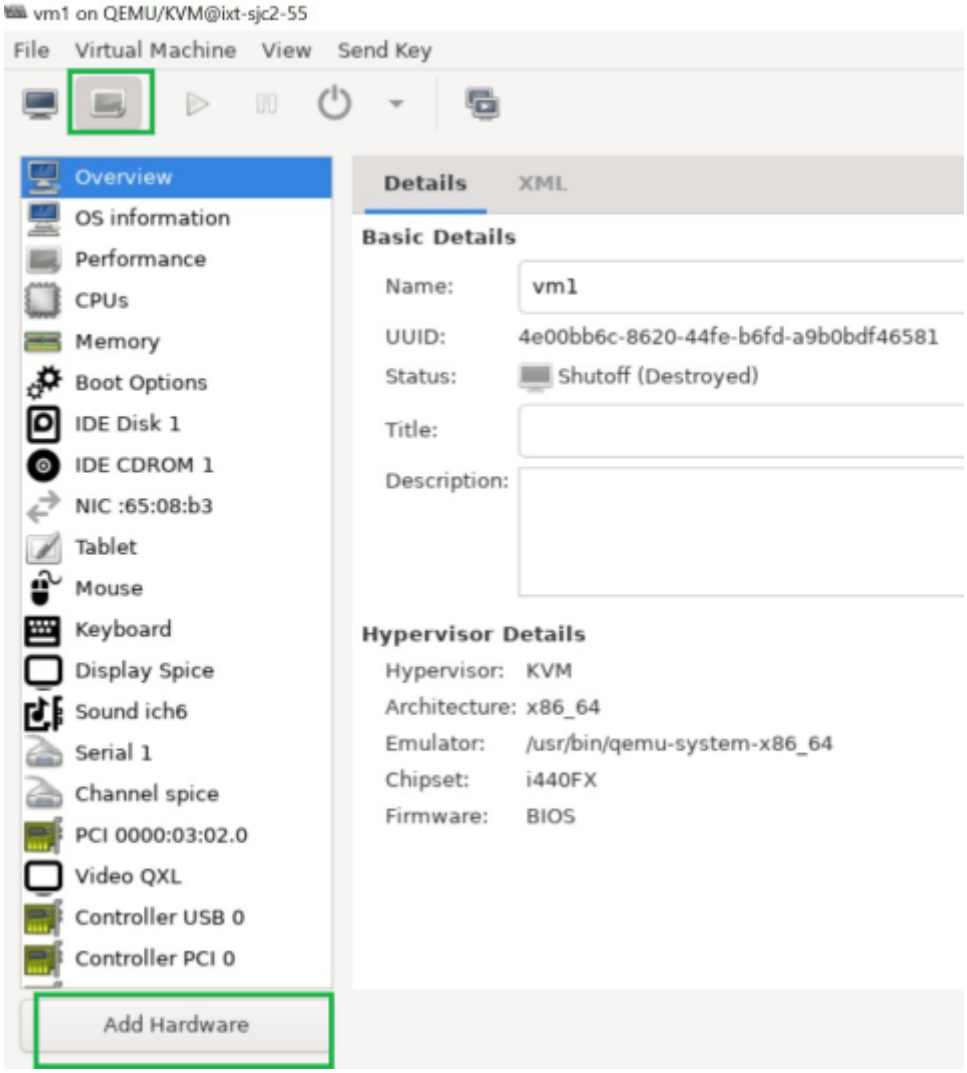


Fig. 42.15: Virtual Machine Manager

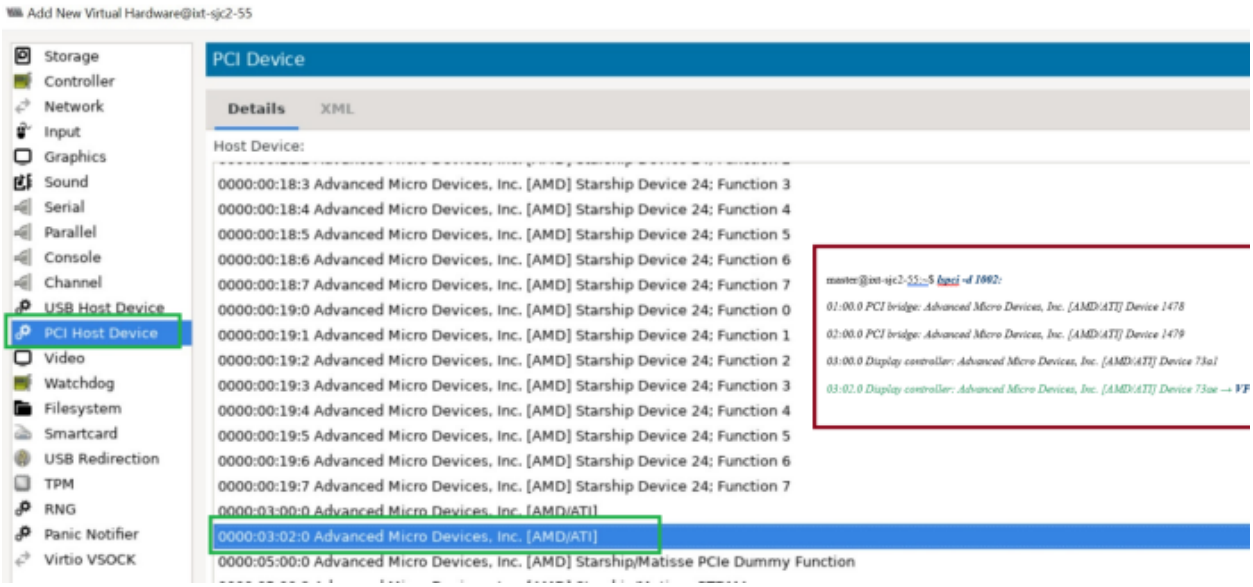


Fig. 42.16: VF Selection

CHAPTER

FORTYTHREE

DEEP LEARNING GUIDE

The following sections cover the different framework installations for ROCm and Deep Learning applications. Fig. 43.1 provides the sequential flow for the use of each framework. Refer to the ROCm Compatible Frameworks Release Notes for each framework's most current release notes at Deep Learning.

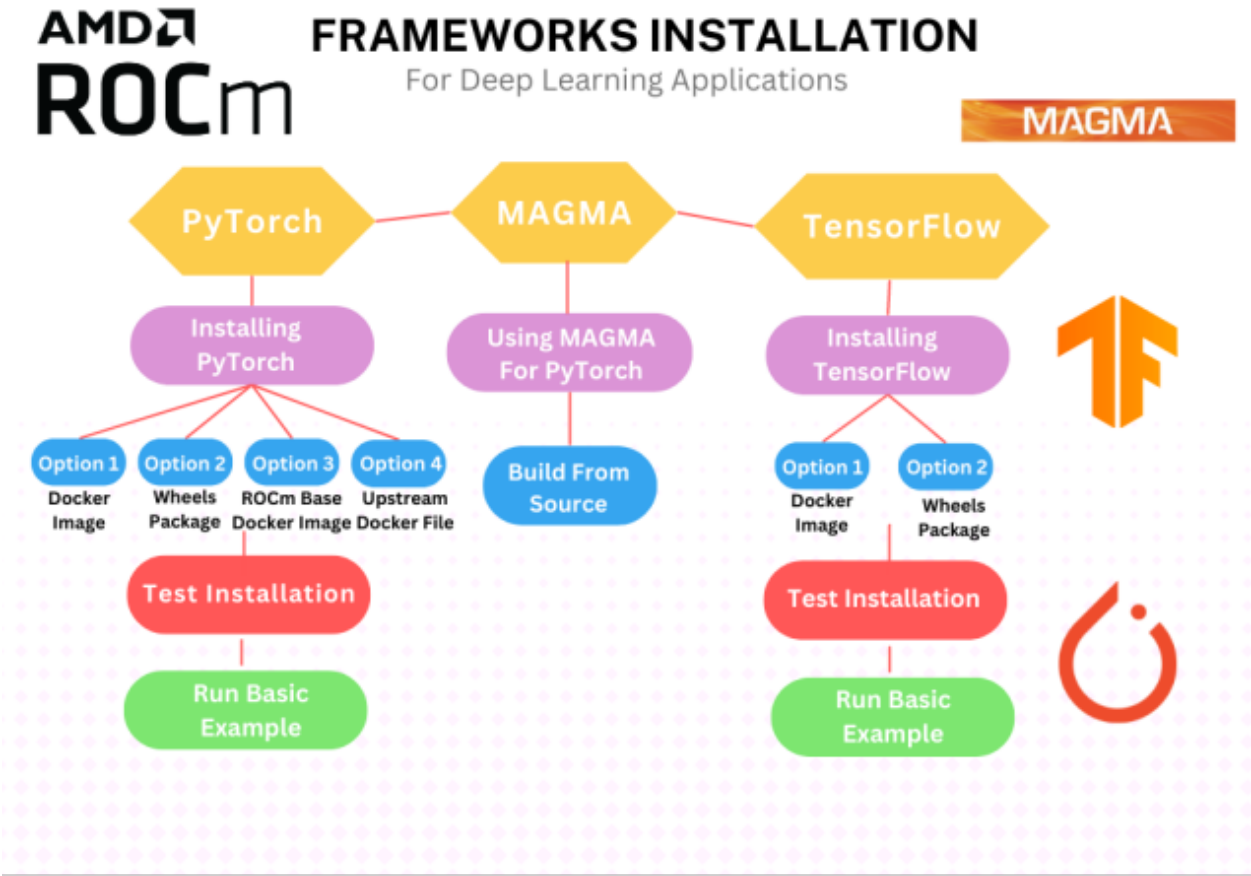


Fig. 43.1: ROCm Compatible Frameworks Flowchart

43.1 Frameworks Installation

- How to Install PyTorch?
- How to Install Tensorflow?
- How to Install Magma?

43.2 Magma Installation for ROCm

43.2.1 MAGMA for ROCm

Matrix Algebra on GPU and Multi-core Architectures, abbreviated as MAGMA, is a collection of next-generation dense linear algebra libraries that is designed for heterogeneous architectures, such as multiple GPUs and multi- or many-core CPUs.

MAGMA provides implementations for CUDA, HIP, Intel Xeon Phi, and OpenCL™. For more information, refer to https://icl.utk.edu/magma/index.html.

43.2.1.1 Using MAGMA for PyTorch

Tensor is fundamental to Deep Learning techniques because it provides extensive representational functionalities and math operations. This data structure is represented as a multidimensional matrix. MAGMA accelerates tensor operations with a variety of solutions including driver routines, computational routines, BLAS routines, auxiliary routines, and utility routines.

43.2.1.2 Build MAGMA from Source

To build MAGMA from the source, follow these steps:

1. In the event you want to compile only for your uarch, use:

   ```
   export PYTORCH_ROCM_ARCH=<uarch>
   ```

   <uarch> is the architecture reported by the rocminfo command.

2. Use the following:

   ```
   export PYTORCH_ROCM_ARCH=<uarch>

   # "install" hipMAGMA into /opt/rocm/magma by copying after build
   git clone https://bitbucket.org/icl/magma.git
   pushd magma
   # Fixes memory leaks of magma found while executing linalg UTs
   git checkout 5959b8783e45f1809812ed96ae762f38ee701972
   cp make.inc-examples/make.inc.hip-gcc-mkl make.inc
   echo 'LIBDIR += -L$(MKLROOT)/lib' >> make.inc
   echo 'LIB += -Wl,--enable-new-dtags -Wl,--rpath,/opt/rocm/lib -Wl,--rpath,$(MKLROOT)/lib -Wl,--rpath,
   ↪/opt/rocm/magma/lib' >> make.inc
   echo 'DEVCCFLAGS += --gpu-max-threads-per-block=256' >> make.inc
   export PATH="${PATH}:/opt/rocm/bin"
   if [[ -n "$PYTORCH_ROCM_ARCH" ]]; then
     amdgpu_targets=`echo $PYTORCH_ROCM_ARCH | sed 's/;/ /g'`
   ```

(continues on next page)

(continued from previous page)

```
else
  amdgpu_targets=`rocm_agent_enumerator | grep -v gfx000 | sort -u | xargs`
fi
for arch in $amdgpu_targets; do
  echo "DEVCCFLAGS += --amdgpu-target=$arch" >> make.inc
done
# hipcc with openmp flag may cause isnan() on __device__ not to be found; depending on context,␣
↪compiler may attempt to match with host definition
sed -i 's/^FOPENMP/#FOPENMP/g' make.inc
make -f make.gen.hipMAGMA -j $(nproc)
LANG=C.UTF-8 make lib/libmagma.so -j $(nproc) MKLROOT=/opt/conda
make testing/testing_dgemm -j $(nproc) MKLROOT=/opt/conda
popd
mv magma /opt/rocm
```

43.2.2 References

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," CoRR, p. abs/1512.00567, 2015

PyTorch, [Online]. Available: https://pytorch.org/vision/stable/index.html

PyTorch, [Online]. Available: https://pytorch.org/hub/pytorch_vision_inception_v3/

Stanford, [Online]. Available: http://cs231n.stanford.edu/

Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy

AMD, "ROCm issues," [Online]. Available: https://github.com/RadeonOpenCompute/ROCm/issues

PyTorch, [Online image]. https://pytorch.org/assets/brand-guidelines/PyTorch-Brand-Guidelines.pdf

TensorFlow, [Online image]. https://www.tensorflow.org/extras/tensorflow_brand_guidelines.pdf

MAGMA, [Online image]. https://bitbucket.org/icl/magma/src/master/docs/

Advanced Micro Devices, Inc., [Online]. Available: https://rocmsoftwareplatform.github.io/AMDMIGraphX/doc/html/

Advanced Micro Devices, Inc., [Online]. Available: https://github.com/ROCmSoftwarePlatform/AMDMIGraphX/wiki

Docker, [Online]. https://docs.docker.com/get-started/overview/

Torchvision, [Online]. Available https://pytorch.org/vision/master/index.html?highlight=torchvision#module-torchvision

43.3 PyTorch Installation for ROCm

43.3.1 PyTorch

PyTorch is an open source Machine Learning Python library, primarily differentiated by Tensor computing with GPU acceleration and a type-based automatic differentiation. Other advanced features include:

- Support for distributed training
- Native ONNX support

- C++ front-end
- The ability to deploy at scale using TorchServe
- A production-ready deployment mechanism through TorchScript

43.3.1.1 Installing PyTorch

To install ROCm on bare metal, refer to the sections GPU and OS Support (Linux) and Compatibility for hardware, software and 3rd-party framework compatibility between ROCm and PyTorch. The recommended option to get a PyTorch environment is through Docker. However, installing the PyTorch wheels package on bare metal is also supported.

43.3.1.1.1 Option 1 (Recommended): Use Docker Image with PyTorch Pre-Installed

Using Docker gives you portability and access to a prebuilt Docker container that has been rigorously tested within AMD. This might also save on the compilation time and should perform as it did when tested without facing potential installation issues.

Follow these steps:

1. Pull the latest public PyTorch Docker image.

   ```
   docker pull rocm/pytorch:latest
   ```

   Optionally, you may download a specific and supported configuration with different user-space ROCm versions, PyTorch versions, and supported operating systems. To download the PyTorch Docker image, refer to https://hub.docker.com/r/rocm/pytorch.

2. Start a Docker container using the downloaded image.

   ```
   docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined --device=/dev/kfd --device=/
   ↪dev/dri --group-add video --ipc=host --shm-size 8G rocm/pytorch:latest
   ```

   Note: This will automatically download the image if it does not exist on the host. You can also pass the -v argument to mount any data directories from the host onto the container.

43.3.1.1.2 Option 2: Install PyTorch Using Wheels Package

PyTorch supports the ROCm platform by providing tested wheels packages. To access this feature, refer to https://pytorch.org/get-started/locally/ and choose the "ROCm" compute platform. Fig. 43.2 is a matrix from https://pytorch.org/ that illustrates the installation compatibility between ROCm and the PyTorch build.

To install PyTorch using the wheels package, follow these installation steps:

1. Choose one of the following options: a. Obtain a base Docker image with the correct user-space ROCm version installed from https://hub.docker.com/repository/docker/rocm/dev-ubuntu-20.04.

   or

   b. Download a base OS Docker image and install ROCm following the installation directions in the section Installation. ROCm 5.2 is installed in this example, as supported by the installation matrix from https://pytorch.org/.

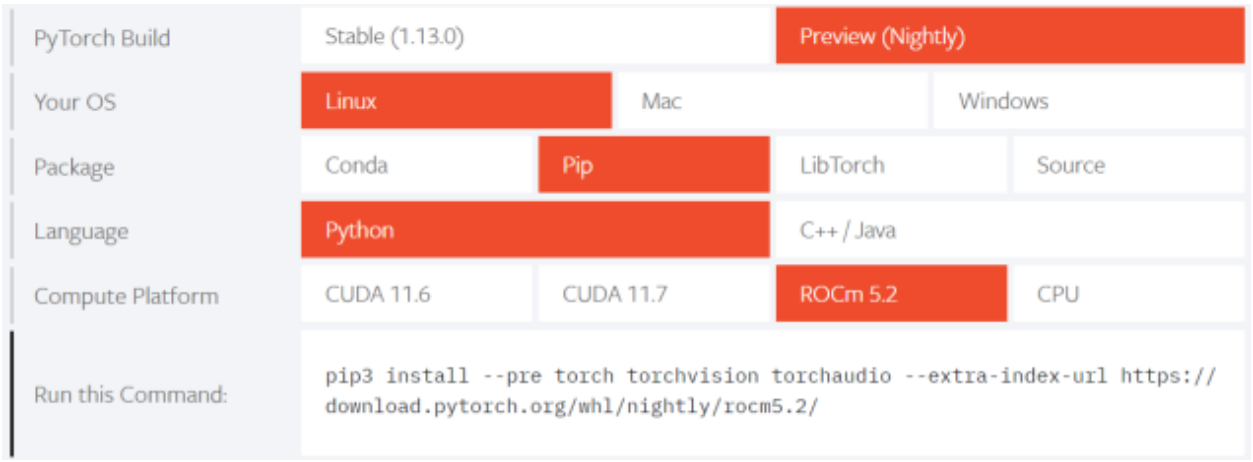


Fig. 43.2: Installation Matrix from Pytorch

or

c. Install on bare metal. Skip to Step 3.

```
docker run -it --device=/dev/kfd --device=/dev/dri --group-add video rocm/dev-ubuntu-20.04:latest
```

2. Start the Docker container, if not installing on bare metal.

```
docker run -it --device=/dev/kfd --device=/dev/dri --group-add video rocm/dev-ubuntu-20.04:latest
```

3. Install any dependencies needed for installing the wheels package.

```
sudo apt update
sudo apt install libjpeg-dev python3-dev
pip3 install wheel setuptools
```

4. Install torch, torchvision, and torchaudio as specified by the installation matrix.

Note: ROCm 5.2 PyTorch wheel in the command below is shown for reference.

```
pip3 install --pre torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/nightly/
↪rocm5.2/
```

43.3.1.1.3 Option 3: Install PyTorch Using PyTorch ROCm Base Docker Image

A prebuilt base Docker image is used to build PyTorch in this option. The base Docker has all dependencies installed, including:

- ROCm
- Torchvision
- Conda packages
- Compiler toolchain

Additionally, a particular environment flag (BUILD_ENVIRONMENT) is set, and the build scripts utilize that to determine the build environment configuration.

Follow these steps:

1. Obtain the Docker image.

```
docker pull rocm/pytorch:latest-base
```

   The above will download the base container, which does not contain PyTorch.

2. Start a Docker container using the image.

```
docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined --device=/dev/kfd --device=/
↪dev/dri --group-add video --ipc=host --shm-size 8G rocm/pytorch:latest-base
```

   You can also pass the -v argument to mount any data directories from the host onto the container.

3. Clone the PyTorch repository.

```
cd ~
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git submodule update --init --recursive
```

4. Build PyTorch for ROCm.

Note: By default in the rocm/pytorch:latest-base, PyTorch builds for these architectures simultaneously:

- gfx900
- gfx906
- gfx908
- gfx90a
- gfx1030

---

5. To determine your AMD uarch, run:

```
rocminfo | grep gfx
```

6. In the event you want to compile only for your uarch, use:

```
export PYTORCH_ROCM_ARCH=<uarch>
```

   <uarch> is the architecture reported by the rocminfo command.

7. Build PyTorch using the following command:

```
./.jenkins/pytorch/build.sh
```

   This will first convert PyTorch sources for HIP compatibility and build the PyTorch framework.

8. Alternatively, build PyTorch by issuing the following commands:

```
python3 tools/amd_build/build_amd.py
USE_ROCM=1 MAX_JOBS=4 python3 setup.py install --user
```

43.3.1.1.4 Option 4: Install Using PyTorch Upstream Docker File

Instead of using a prebuilt base Docker image, you can build a custom base Docker image using scripts from the PyTorch repository. This will utilize a standard Docker image from operating system maintainers and install all the dependencies required to build PyTorch, including

- ROCm
- Torchvision
- Conda packages
- Compiler toolchain

Follow these steps:

1. Clone the PyTorch repository on the host.

```
cd ~
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git submodule update --init --recursive
```

2. Build the PyTorch Docker image.

```
cd.circleci/docker
./build.sh pytorch-linux-bionic-rocm<version>-py3.7
# eg. ./build.sh pytorch-linux-bionic-rocm3.10-py3.7
```

   This should be complete with a message "Successfully build <image_id>."

3. Start a Docker container using the image:

```
docker run -it --cap-add=SYS_PTRACE --security-opt
seccomp=unconfined --device=/dev/kfd --device=/dev/dri --group-add
video --ipc=host --shm-size 8G <image_id>
```

   You can also pass -v argument to mount any data directories from the host onto the container.

4. Clone the PyTorch repository.

```
cd ~
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git submodule update --init --recursive
```

5. Build PyTorch for ROCm.

Note: By default in the rocm/pytorch:latest-base, PyTorch builds for these architectures simultaneously:

- gfx900
- gfx906
- gfx908
- gfx90a
- gfx1030

---

6. To determine your AMD uarch, run:

```
rocminfo | grep gfx
```

7. If you want to compile only for your uarch:

```
export PYTORCH_ROCM_ARCH=<uarch>
```

<uarch> is the architecture reported by the rocminfo command.

8. Build PyTorch using:

```
./.jenkins/pytorch/build.sh
```

This will first convert PyTorch sources to be HIP compatible and then build the PyTorch framework.

Alternatively, build PyTorch by issuing the following commands:

```
python3 tools/amd_build/build_amd.py
USE_ROCM=1 MAX_JOBS=4 python3 setup.py install --user
```

43.3.1.2 Test the PyTorch Installation

You can use PyTorch unit tests to validate a PyTorch installation. If using a prebuilt PyTorch Docker image from AMD ROCm Docker Hub or installing an official wheels package, these tests are already run on those configurations. Alternatively, you can manually run the unit tests to validate the PyTorch installation fully.

Follow these steps:

1. Test if PyTorch is installed and accessible by importing the torch package in Python.

Note: Do not run in the PyTorch git folder.

```
python3 -c 'import torch' 2> /dev/null && echo 'Success' || echo 'Failure'
```

2. Test if the GPU is accessible from PyTorch. In the PyTorch framework, torch.cuda is a generic mechanism to access the GPU; it will access an AMD GPU only if available.

```
python3 -c 'import torch; print(torch.cuda.is_available())'
```

3. Run the unit tests to validate the PyTorch installation fully. Run the following command from the PyTorch home directory:

```
BUILD_ENVIRONMENT=${BUILD_ENVIRONMENT:-rocm} ./.jenkins/pytorch/test.sh
```

This ensures that even for wheel installs in a non-controlled environment, the required environment variable will be set to skip certain unit tests for ROCm.

Note: Make sure the PyTorch source code is corresponding to the PyTorch wheel or installation in the Docker image. Incompatible PyTorch source code might give errors when running the unit tests.

This will first install some dependencies, such as a supported torchvision version for PyTorch. torchvision is used in some PyTorch tests for loading models. Next, this will run all the unit tests.

Note: Some tests may be skipped, as appropriate, based on your system configuration. All features of PyTorch are not supported on ROCm, and the tests that evaluate these features are skipped. In addition, depending on the host memory, or the number of available GPUs, other tests may be skipped. No test should fail if the compilation and installation are correct.

4. Run individual unit tests with the following command:

```
PYTORCH_TEST_WITH_ROCM=1 python3 test/test_nn.py --verbose
```

test_nn.py can be replaced with any other test set.

43.3.1.3 Run a Basic PyTorch Example

The PyTorch examples repository provides basic examples that exercise the functionality of the framework. MNIST (Modified National Institute of Standards and Technology) database is a collection of handwritten digits that may be used to train a Convolutional Neural Network for handwriting recognition. Alternatively, ImageNet is a database of images used to train a network for visual object recognition.

Follow these steps:

1. Clone the PyTorch examples repository.

```
git clone https://github.com/pytorch/examples.git
```

2. Run the MNIST example.

```
cd examples/mnist
```

3. Follow the instructions in the README file in this folder. In this case:

```
pip3 install -r requirements.txt
python3 main.py
```

4. Run the ImageNet example.

```
cd examples/imagenet
```

5. Follow the instructions in the README file in this folder. In this case:

```
pip3 install -r requirements.txt
python3 main.py
```

43.3.2 Using MIOpen kdb files with ROCm PyTorch wheels

PyTorch uses MIOpen for machine learning primitives. These primitives are compiled into kernels at runtime. Runtime compilation causes a small warm-up phase when starting PyTorch. MIOpen kdb files contain precompiled kernels that can speed up the warm-up phase of an application. More information is available in the MIOpeninstallation page.

MIOpen kdb files can be used with ROCm PyTorch wheels. However, the kdb files need to be placed in a specific location with respect to the PyTorch installation path. A helper script simplifies this task for the user. The script takes in the ROCm version and user's GPU architecture as inputs, and works for Ubuntu and CentOS.

Helper script: install_kdb_files_for_pytorch_wheels.sh

Usage:

After installing ROCm PyTorch wheels:

1. [Optional] export GFX_ARCH=gfx90a
2. [Optional] export ROCM_VERSION=5.5
3. ./install_kdb_files_for_pytorch_wheels.sh

43.3.3 References

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," CoRR, p. abs/1512.00567, 2015

PyTorch, [Online]. Available: https://pytorch.org/vision/stable/index.html

PyTorch, [Online]. Available: https://pytorch.org/hub/pytorch_vision_inception_v3/

Stanford, [Online]. Available: http://cs231n.stanford.edu/

Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy

AMD, "ROCm issues," [Online]. Available: https://github.com/RadeonOpenCompute/ROCm/issues

PyTorch, [Online image]. https://pytorch.org/assets/brand-guidelines/PyTorch-Brand-Guidelines.pdf

TensorFlow, [Online image]. https://www.tensorflow.org/extras/tensorflow_brand_guidelines.pdf

MAGMA, [Online image]. https://bitbucket.org/icl/magma/src/master/docs/

Advanced Micro Devices, Inc., [Online]. Available: https://rocmsoftwareplatform.github.io/AMDMIGraphX/doc/html/

Advanced Micro Devices, Inc., [Online]. Available: https://github.com/ROCmSoftwarePlatform/AMDMIGraphX/wiki

Docker, [Online]. https://docs.docker.com/get-started/overview/

Torchvision, [Online]. Available https://pytorch.org/vision/master/index.html?highlight=torchvision#module-torchvision

43.4 TensorFlow Installation for ROCm

43.4.1 TensorFlow

TensorFlow is an open source library for solving Machine Learning, Deep Learning, and Artificial Intelligence problems. It can be used to solve many problems across different sectors and industries but primarily focuses on training and inference in neural networks. It is one of the most popular and in-demand frameworks and is very active in open source contribution and development.

> Warning: ROCm 5.6 and 5.7 deviates from the standard practice of supporting the last three TensorFlow versions. This is due to incompatibilities between earlier TensorFlow versions and changes introduced in the ROCm 5.6 compiler. Refer to the following version support matrix:

ROCm	TensorFlow
5.6.x	2.12
5.7.0	2.12, 2.13
Post-5.7.0	Last three versions at ROCm release.

43.4.1.1 Installing TensorFlow

The following sections contain options for installing TensorFlow.

43.4.1.1.1 Option 1: Install TensorFlow Using Docker Image

To install ROCm on bare metal, follow the section Installation (Linux). The recommended option to get a TensorFlow environment is through Docker.

Using Docker provides portability and access to a prebuilt Docker container that has been rigorously tested within AMD. This might also save compilation time and should perform as tested without facing potential installation issues. Follow these steps:

1. Pull the latest public TensorFlow Docker image.

```
docker pull rocm/tensorflow:latest
```

2. Once you have pulled the image, run it by using the command below:

```
docker run -it --network=host --device=/dev/kfd --device=/dev/dri \
--ipc=host --shm-size 16G --group-add video --cap-add=SYS_PTRACE \
--security-opt seccomp=unconfined rocm/tensorflow:latest
```

43.4.1.1.2 Option 2: Install TensorFlow Using Wheels Package

To install TensorFlow using the wheels package, follow these steps:

1. Check the Python version.

```
python3 --version
```

If:	Then:
The Python version is less than 3.7	Upgrade Python.
The Python version is more than 3.7	Skip this step and go to Step 3.

Note: The supported Python versions are:

- 3.7
- 3.8
- 3.9
- 3.10

```
sudo apt-get install python3.7 # or python3.8 or python 3.9 or python 3.10
```

2. Set up multiple Python versions using update-alternatives.

```
update-alternatives --query python3
sudo update-alternatives --install
/usr/bin/python3 python3 /usr/bin/python[version] [priority]
```

Note: Follow the instruction in Step 2 for incompatible Python versions.

```
sudo update-alternatives --config python3
```

3. Follow the screen prompts, and select the Python version installed in Step 2.
4. Install or upgrade PIP.

```
sudo apt install python3-pip
```

To install PIP, use the following:

```
/usr/bin/python[version]  -m pip install --upgrade pip
```

Upgrade PIP for Python version installed in step 2:

```
sudo pip3 install --upgrade pip
```

5. Install TensorFlow for the Python version as indicated in Step 2.

```
/usr/bin/python[version] -m pip install --user tensorflow-rocm==[wheel-version] --upgrade
```

For a valid wheel version for a ROCm release, refer to the instruction below:

```
sudo apt install rocm-libs rccl
```

6. Update protobuf to 3.19 or lower.

```
/usr/bin/python3.7  -m pip install protobuf=3.19.0
sudo pip3 install tensorflow
```

7. Set the environment variable PYTHONPATH.

```
export PYTHONPATH="./.local/lib/python[version]/site-packages:$PYTHONPATH"  #Use same python␣
↪version as in step 2
```

8. Install libraries.

```
sudo apt install rocm-libs rccl
```

9. Test installation.

```
python3 -c 'import tensorflow' 2> /dev/null && echo 'Success' || echo 'Failure'
```

Note: For details on tensorflow-rocm wheels and ROCm version compatibility, see: https://github.com/ROCmSoftwarePlatform/tensorflow-upstream/blob/develop-upstream/rocm_

docs/tensorflow-rocm-release.md

43.4.1.2 Test the TensorFlow Installation

To test the installation of TensorFlow, run the container image as specified in the previous section Installing TensorFlow. Ensure you have access to the Python shell in the Docker container.

```
python3 -c 'import tensorflow' 2> /dev/null && echo 'Success' || echo 'Failure'
```

43.4.1.3 Run a Basic TensorFlow Example

The TensorFlow examples repository provides basic examples that exercise the framework's functionality. The MNIST database is a collection of handwritten digits that may be used to train a Convolutional Neural Network for handwriting recognition.

Follow these steps:

1. Clone the TensorFlow example repository.

   ```
   cd ~
   git clone https://github.com/tensorflow/models.git
   ```

2. Install the dependencies of the code, and run the code.

   ```
   #pip3 install requirement.txt
   #python mnist_tf.py
   ```

43.4.2 References

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," CoRR, p. abs/1512.00567, 2015

PyTorch, [Online]. Available: https://pytorch.org/vision/stable/index.html

PyTorch, [Online]. Available: https://pytorch.org/hub/pytorch_vision_inception_v3/

Stanford, [Online]. Available: http://cs231n.stanford.edu/

Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy

AMD, "ROCm issues," [Online]. Available: https://github.com/RadeonOpenCompute/ROCm/issues

PyTorch, [Online image]. https://pytorch.org/assets/brand-guidelines/PyTorch-Brand-Guidelines.pdf

TensorFlow, [Online image]. https://www.tensorflow.org/extras/tensorflow_brand_guidelines.pdf

MAGMA, [Online image]. https://bitbucket.org/icl/magma/src/master/docs/

Advanced Micro Devices, Inc., [Online]. Available: https://rocmsoftwareplatform.github.io/AMDMIGraphX/doc/html/

Advanced Micro Devices, Inc., [Online]. Available: https://github.com/ROCmSoftwarePlatform/AMDMIGraphX/wiki

Docker, [Online]. https://docs.docker.com/get-started/overview/

Torchvision, [Online]. Available https://pytorch.org/vision/master/index.html?highlight=torchvision#module-torchvision

CHAPTER

FORTYFOUR

GPU-ENABLED MPI

The Message Passing Interface (MPI) is a standard API for distributed and parallel application development that can scale to multi-node clusters. To facilitate the porting of applications to clusters with GPUs, ROCm enables various technologies. These technologies allow users to directly use GPU pointers in MPI calls and enable ROCm-aware MPI libraries to deliver optimal performance for both intra-node and inter-node GPU-to-GPU communication.

The AMD kernel driver exposes Remote Direct Memory Access (RDMA) through the PeerDirect interfaces to allow Host Channel Adapters (HCA, a type of Network Interface Card or NIC) to directly read and write to the GPU device memory with RDMA capabilities. These interfaces are currently registered as a peer_memory_client with Mellanox's OpenFabrics Enterprise Distribution (OFED) ib_core kernel module to allow high-speed DMA transfers between GPU and HCA. These interfaces are used to optimize inter-node MPI message communication.

This chapter exemplifies how to set up Open MPI with the ROCm platform. The Open MPI project is an open source implementation of the Message Passing Interface (MPI) that is developed and maintained by a consortium of academic, research, and industry partners.

Several MPI implementations can be made ROCm-aware by compiling them with Unified Communication Framework (UCX) support. One notable exception is MVAPICH2: It directly supports AMD GPUs without using UCX, and you can download it here. Use the latest version of the MVAPICH2-GDR package.

The Unified Communication Framework, is an open source cross-platform framework whose goal is to provide a common set of communication interfaces that targets a broad set of network programming models and interfaces. UCX is ROCm-aware, and ROCm technologies are used directly to implement various network operation primitives. For more details on the UCX design, refer to it's documentation.

44.1 Building UCX

The following section describes how to set up UCX so it can be used to compile Open MPI. The following environment variables are set, such that all software components will be installed in the same base directory (we assume to install them in your home directory; for other locations adjust the below environment variables accordingly, and make sure you have write permission for that location):

```
export INSTALL_DIR=$HOME/ompi_for_gpu
export BUILD_DIR=/tmp/ompi_for_gpu_build
mkdir -p $BUILD_DIR
```

The following sequences of build commands assume either the ROCmCC or the AOMP compiler is active in the environment, which will execute the commands.

44.2 Install UCX

The next step is to set up UCX by compiling its source code and install it:

```
export UCX_DIR=$INSTALL_DIR/ucx
cd $BUILD_DIR
git clone https://github.com/openucx/ucx.git -b v1.14.1
cd ucx
./autogen.sh
mkdir build
cd build
../configure -prefix=$UCX_DIR \
    --with-rocm=/opt/rocm
make -j $(nproc)
make -j $(nproc) install
```

The following table documents the compatibility of UCX versions with ROCm versions.

44.3 Install Open MPI

These are the steps to build Open MPI:

```
export OMPI_DIR=$INSTALL_DIR/ompi
cd $BUILD_DIR
git clone --recursive https://github.com/open-mpi/ompi.git \
    -b v5.0.x
cd ompi
./autogen.pl
mkdir build
cd build
../configure --prefix=$OMPI_DIR --with-ucx=$UCX_DIR \
    --with-rocm=/opt/rocm
make -j $(nproc)
make -j $(nproc) install
```

44.4 ROCm-enabled OSU

The OSU Micro Benchmarks v5.9 (OMB) can be used to evaluate the performance of various primitives with an AMD GPU device and ROCm support. This functionality is exposed when configured with --enable-rocm option. We can use the following steps to compile OMB:

```
export OSU_DIR=$INSTALL_DIR/osu
cd $BUILD_DIR
wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.9.tar.gz
tar xfz osu-micro-benchmarks-5.9.tar.gz
cd osu-micro-benchmarks-5.9
./configure --prefix=$INSTALL_DIR/osu --enable-rocm \
    --with-rocm=/opt/rocm \
    CC=$OMPI_DIR/bin/mpicc CXX=$OMPI_DIR/bin/mpicxx \
    LDFLAGS="-L$OMPI_DIR/lib/ -lmpi -L/opt/rocm/lib/ \
    $(hipconfig -C) -lamdhip64" CXXFLAGS="-std=c++11"
make -j $(nproc)
```

44.5 Intra-node Run

Before running an Open MPI job, it is essential to set some environment variables to ensure that the correct version of Open MPI and UCX is being used.

```
export LD_LIBRARY_PATH=$OMPI_DIR/lib:$UCX_DIR/lib:/opt/rocm/lib
export PATH=$OMPI_DIR/bin:$PATH
```

The following command runs the OSU bandwidth benchmark between the first two GPU devices (i.e., GPU 0 and GPU 1, same OAM) by default inside the same node. It measures the unidirectional bandwidth from the first device to the other.

```
$OMPI_DIR/bin/mpirun -np 2                \
  -x UCX_TLS=sm,self,rocm                 \
  --mca pml ucx mpi/pt2pt/osu_bw -d rocm D D
```

To select different devices, for example 2 and 3, use the following command:

```
export HIP_VISIBLE_DEVICES=2,3
export HSA_ENABLE_SDMA=0
```

The following output shows the effective transfer bandwidth measured for inter-die data transfer between GPU device 2 and 3 (same OAM). For messages larger than 67MB, an effective utilization of about 150GB/sec is achieved, which corresponds to 75% of the peak transfer bandwidth of 200GB/sec for that connection:

44.6 Collective Operations

Collective Operations on GPU buffers are best handled through the Unified Collective Communication Library (UCC) component in Open MPI. For this, the UCC library has to be configured and compiled with ROCm support.

Please note the compatibility table for UCC versions with the various ROCm versions.

An example for configuring UCC and Open MPI with ROCm support is shown below:

```
export UCC_DIR=$INSTALL_DIR/ucc
git clone https://github.com/openucx/ucc.git
cd ucc
./configure --with-rocm=/opt/rocm \
            --with-ucx=$UCX_DIR   \
            --prefix=$UCC_DIR
make -j && make install

# Configure and compile Open MPI with UCX, UCC, and ROCm support
cd ompi
./configure --with-rocm=/opt/rocm  \
            --with-ucx=$UCX_DIR    \
            --with-ucc=$UCC_DIR
            --prefix=$OMPI_DIR
```

To use the UCC component with an MPI application requires setting some additional parameters:

```
mpirun --mca pml ucx --mca osc ucx \
       --mca coll_ucc_enable 1     \
       --mca coll_ucc_priority 100 -np 64 ./my_mpi_app
```

```
# OSU MPI-ROCM Bandwidth Test v5.9
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size      Bandwidth (MB/s)
2                      1.01
4                      2.03
8                      4.04
16                     7.77
32                    10.43
64                    22.47
128                   44.26
256                   88.10
512                  114.57
1024                 211.02
2048                 290.50
4096                 335.14
8192                 389.36
16384               1826.78
32768               3484.68
65536               6773.21
131072             12619.03
262144             22524.74
524288             40163.84
1048576            60298.85
2097152            83916.57
4194304           118524.13
8388608           133420.64
16777216          143527.03
33554432          147779.60
67108864          151540.93
134217728         151317.96
268435456         150834.76
```

Fig. 44.1: Inter-GPU bandwidth with various payload sizes.

CHAPTER

FORTYFIVE

SYSTEM DEBUGGING GUIDE

45.1 ROCm Language and System Level Debug, Flags, and Environment Variables

Kernel options to avoid: the Ethernet port getting renamed every time you change graphics cards, net.ifnames=0 biosdevname=0

45.2 ROCr Error Code

- 2 Invalid Dimension
- 4 Invalid Group Memory
- 8 Invalid (or Null) Code
- 32 Invalid Format
- 64 Group is too large
- 128 Out of VGPRs
- 0x80000000 Debug Options

45.3 Command to Dump Firmware Version and Get Linux Kernel Version

sudo cat /sys/kernel/debug/dri/1/amdgpu_firmware_info

uname -a

45.4 Debug Flags

Debug messages when developing/debugging base ROCm driver. You could enable the printing from libhsakmt.so by setting an environment variable, HSAKMT_DEBUG_LEVEL. Available debug levels are 3-7. The higher level you set, the more messages will print.

- export HSAKMT_DEBUG_LEVEL=3 : Only pr_err() prints.
- export HSAKMT_DEBUG_LEVEL=4 : pr_err() and pr_warn() print.

- export HSAKMT_DEBUG_LEVEL=5 : We currently do not implement “notice”. Setting to 5 is same as setting to 4.
- export HSAKMT_DEBUG_LEVEL=6 : pr_err(), pr_warn(), and pr_info print.
- export HSAKMT_DEBUG_LEVEL=7 : Everything including pr_debug prints.

45.5 ROCr Level Environment Variables for Debug

HSA_ENABLE_SDMA=0

HSA_ENABLE_INTERRUPT=0

HSA_SVM_GUARD_PAGES=0

HSA_DISABLE_CACHE=1

45.6 Turn Off Page Retry on GFX9/Vega Devices

sudo -s

echo 1 > /sys/module/amdkfd/parameters/noretry

45.7 HIP Environment Variables 3.x

45.7.1 OpenCL Debug Flags

AMD_OCL_WAIT_COMMAND=1 (0 = OFF, 1 = On)

45.8 PCIe-Debug

For information on how to debug and profile HIP applications, see HIP Debugging

CHAPTER

FORTYSIX

MACHINE LEARNING, DEEP LEARNING, AND ARTIFICIAL INTELLIGENCE

Inception V3 with PyTorch A collection of detailed and guided examples for working with Inception V3 with PyTorch on ROCm.

Optimizing Inference with MIGraphX Walkthroughs of optimizing inference using MIGraphX.

46.1 Inception V3 with PyTorch

46.1.1 Deep Learning Training

Deep Learning models are designed to capture the complexity of the problem and the underlying data. These models are "deep," comprising multiple component layers. Training is finding the best parameters for each model layer to achieve a well-defined objective.

The training data consists of input features in supervised learning, similar to what the learned model is expected to see during the evaluation or inference phase. The target output is also included, which serves to teach the model. A loss metric is defined as part of training that evaluates the model's performance during the training process.

Training also includes the choice of an optimization algorithm that reduces the loss by adjusting the model's parameters. Training is an iterative process where training data is fed in, usually split into different batches, with the entirety of the training data passed during one training epoch. Training usually is run for multiple epochs.

46.1.2 Training Phases

Training occurs in multiple phases for every batch of training data. Table 46.1 provides an explanation of the types of training phases.

Table 46.1: Types of Training Phases

Types of Phases	
Forward Pass	The input features are fed into the model, whose parameters may be randomly initialized initially. Activations (outputs) of each layer are retained during this pass to help in the loss gradient computation during the backward pass.
Loss Computation	The output is compared against the target outputs, and the loss is computed.
Backward Pass	The loss is propagated backward, and the model's error gradients are computed and stored for each trainable parameter.
Optimization Pass	The optimization algorithm updates the model parameters using the stored error gradients.

Training is different from inference, particularly from the hardware perspective. Table 46.2 shows the contrast between training and inference.

Table 46.2: Training vs. Inference

Training	Inference
Training is measured in hours/days.	The inference is measured in minutes.
Training is generally run offline in a data center or cloud setting.	The inference is made on edge devices.
The memory requirements for training are higher than inference due to storing intermediate data, such as activations and error gradients.	The memory requirements are lower for inference than training.
Data for training is available on the disk before the training process and is generally significant. The training performance is measured by how fast the data batches can be processed.	Inference data usually arrive stochastically, which may be batched to improve performance. Inference performance is generally measured in throughput speed to process the batch of data and the delay in responding to the input (latency).

Different quantization data types are typically chosen between training (FP32, BF16) and inference (FP16, INT8). The computation hardware has different specializations from other datatypes, leading to improvement in performance if a faster datatype can be selected for the corresponding task.

46.1.3 Case Studies

The following sections contain case studies for the Inception v3 model.

46.1.3.1 Inception v3 with PyTorch

Convolution Neural Networks are forms of artificial neural networks commonly used for image processing. One of the core layers of such a network is the convolutional layer, which convolves the input with a weight tensor and passes the result to the next layer. Inception v3[1] is an architectural development over the ImageNet competition-winning entry, AlexNet, using more profound and broader networks while attempting to meet computational and memory budgets.

The implementation uses PyTorch as a framework. This case study utilizes torchvision[2], a repository of popular datasets and model architectures, for obtaining the model. torchvision also provides pre-trained weights as a starting point to develop new models or fine-tune the model for a new task.

46.1.3.1.1 Evaluating a Pre-Trained Model

The Inception v3 model introduces a simple image classification task with the pre-trained model. This does not involve training but utilizes an already pre-trained model from torchvision.

This example is adapted from the PyTorch research hub page on Inception v3[3].

Follow these steps:

1. Run the PyTorch ROCm-based Docker image or refer to the section Installing PyTorch for setting up a PyTorch environment on ROCm.

```
docker run -it -v $HOME:/data --cap-add=SYS_PTRACE --security-opt seccomp=unconfined --device=/
↪dev/kfd --device=/dev/dri --group-add video --ipc=host --shm-size 8G rocm/pytorch:latest
```

2. Run the Python shell and import packages and libraries for model creation.

```
import torch
import torchvision
```

3. Set the model in evaluation mode. Evaluation mode directs PyTorch not to store intermediate data, which would have been used in training.

```
model = torch.hub.load('pytorch/vision:v0.10.0', 'inception_v3', pretrained=True)
model.eval()
```

4. Download a sample image for inference.

```
import urllib
url, filename = ("https://github.com/pytorch/hub/raw/master/images/dog.jpg", "dog.jpg")
try: urllib.URLopener().retrieve(url, filename)
except: urllib.request.urlretrieve(url, filename)
```

5. Import torchvision and PIL.Image support libraries.

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," CoRR, p. abs/1512.00567, 2015

[2] PyTorch, [Online]. Available: https://pytorch.org/vision/stable/index.html

[3] PyTorch, [Online]. Available: https://pytorch.org/hub/pytorch_vision_inception_v3/

```
from PIL import Image
from torchvision import transforms
input_image = Image.open(filename)
```

6. Apply preprocessing and normalization.

```
preprocess = transforms.Compose([
    transforms.Resize(299),
    transforms.CenterCrop(299),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

7. Use input tensors and unsqueeze them later.

```
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0)
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')
```

8. Find out probabilities.

```
with torch.no_grad():
    output = model(input_batch)
print(output[0])
probabilities = torch.nn.functional.softmax(output[0], dim=0)
print(probabilities)
```

9. To understand the probabilities, download and examine the ImageNet labels.

```
wget https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt
```

10. Read the categories and show the top categories for the image.

```
with open("imagenet_classes.txt", "r") as f:
    categories = [s.strip() for s in f.readlines()]
top5_prob, top5_catid = torch.topk(probabilities, 5)
for i in range(top5_prob.size(0)):
    print(categories[top5_catid[i]], top5_prob[i].item())
```

46.1.3.1.2 Training Inception v3

The previous section focused on downloading and using the Inception v3 model for a simple image classification task. This section walks through training the model on a new dataset.

Follow these steps:

1. Run the PyTorch ROCm Docker image or refer to the section Installing PyTorch for setting up a PyTorch environment on ROCm.

```
docker pull rocm/pytorch:latest
docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined --device=/dev/kfd --device=/
↪dev/dri --group-add video --ipc=host --shm-size 8G rocm/pytorch:latest
```

2. Download an ImageNet database. For this example, the tiny-imagenet-200[4], a smaller ImageNet variant with 200 image classes and a training dataset with 100,000 images, was downsized to 64x64 color images.

```
wget http://cs231n.stanford.edu/tiny-imagenet-200.zip
```

3. Process the database to set the validation directory to the format expected by PyTorch's DataLoader.

4. Run the following script:

```
import io
import glob
import os
from shutil import move
from os.path import join
from os import listdir, rmdir
target_folder = './tiny-imagenet-200/val/'
val_dict = {}
with open('./tiny-imagenet-200/val/val_annotations.txt', 'r') as f:
    for line in f.readlines():
        split_line = line.split('\t')
        val_dict[split_line[0]] = split_line[1]

paths = glob.glob('./tiny-imagenet-200/val/images/*')
for path in paths:
    file = path.split('/')[-1]
    folder = val_dict[file]
    if not os.path.exists(target_folder + str(folder)):
        os.mkdir(target_folder + str(folder))
        os.mkdir(target_folder + str(folder) + '/images')

for path in paths:
    file = path.split('/')[-1]
    folder = val_dict[file]
    dest = target_folder + str(folder) + '/images/' + str(file)
    move(path, dest)

rmdir('./tiny-imagenet-200/val/images')
```

5. Open a Python shell.

6. Import dependencies, including torch, os, and torchvision.

```
import torch
import os
import torchvision
from torchvision import transforms
from torchvision.transforms.functional import InterpolationMode
```

7. Set parameters to guide the training process.

Note: The device is set to "cuda". In PyTorch, "cuda" is a generic keyword to denote a GPU.

```
device = "cuda"
```

[4] Stanford, [Online]. Available: http://cs231n.stanford.edu/

8. Set the data_path to the location of the training and validation data. In this case, the tiny-imagenet-200 is present as a subdirectory to the current directory.

```
data_path = "tiny-imagenet-200"
```

   The training image size is cropped for input into Inception v3.

```
train_crop_size = 299
```

9. To smooth the image, use bilinear interpolation, a resampling method that uses the distance weighted average of the four nearest pixel values to estimate a new pixel value.

```
interpolation = "bilinear"
```

   The next parameters control the size to which the validation image is cropped and resized.

```
val_crop_size = 299
val_resize_size = 342
```

   The pre-trained Inception v3 model is chosen to be downloaded from torchvision.

```
model_name = "inception_v3"
pretrained = True
```

   During each training step, a batch of images is processed to compute the loss gradient and perform the optimization. In the following setting, the size of the batch is determined.

```
batch_size = 32
```

   This refers to the number of CPU threads the data loader uses to perform efficient multi-process data loading.

```
num_workers = 16
```

   The torch.optim package provides methods to adjust the learning rate as the training progresses. This example uses the StepLR scheduler, which decays the learning rate by lr_gamma at every lr_step_size number of epochs.

```
learning_rate = 0.1
momentum = 0.9
weight_decay = 1e-4
lr_step_size = 30
lr_gamma = 0.1
```

   Note: One training epoch is when the neural network passes an entire dataset forward and backward.

```
epochs = 90
```

   The train and validation directories are determined.

```
train_dir = os.path.join(data_path, "train")
val_dir = os.path.join(data_path, "val")
```

10. Set up the training and testing data loaders.

```
interpolation = InterpolationMode(interpolation)

TRAIN_TRANSFORM_IMG = transforms.Compose([
Normalizaing and standardardizing the image
transforms.RandomResizedCrop(train_crop_size, interpolation=interpolation),
    transforms.PILToTensor(),
    transforms.ConvertImageDtype(torch.float),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225] )
    ])
dataset = torchvision.datasets.ImageFolder(
    train_dir,
    transform=TRAIN_TRANSFORM_IMG
)
TEST_TRANSFORM_IMG = transforms.Compose([
    transforms.Resize(val_resize_size, interpolation=interpolation),
    transforms.CenterCrop(val_crop_size),
    transforms.PILToTensor(),
    transforms.ConvertImageDtype(torch.float),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225] )
    ])

dataset_test = torchvision.datasets.ImageFolder(
    val_dir,
    transform=TEST_TRANSFORM_IMG
)

print("Creating data loaders")
train_sampler = torch.utils.data.RandomSampler(dataset)
test_sampler = torch.utils.data.SequentialSampler(dataset_test)

data_loader = torch.utils.data.DataLoader(
    dataset,
    batch_size=batch_size,
    sampler=train_sampler,
    num_workers=num_workers,
    pin_memory=True
)

data_loader_test = torch.utils.data.DataLoader(
    dataset_test, batch_size=batch_size, sampler=test_sampler, num_workers=num_workers, pin_
↪memory=True
)
```

Note: Use torchvision to obtain the Inception v3 model. Use the pre-trained model weights to speed up training.

```
print("Creating model")
print("Num classes = ", len(dataset.classes))
model = torchvision.models.__dict__[model_name](pretrained=pretrained)
```

11. Adapt Inception v3 for the current dataset. tiny-imagenet-200 contains only 200 classes, whereas Inception v3 is designed for 1,000-class output. The last layer of Inception v3 is replaced to match the output features required.

```
model.fc = torch.nn.Linear(model.fc.in_features, len(dataset.classes))
model.aux_logits = False
model.AuxLogits = None
```

12. Move the model to the GPU device.

```
model.to(device)
```

13. Set the loss criteria. For this example, Cross Entropy Loss[5] is used.

```
criterion = torch.nn.CrossEntropyLoss()
```

14. Set the optimizer to Stochastic Gradient Descent.

```
optimizer = torch.optim.SGD(
    model.parameters(),
    lr=learning_rate,
    momentum=momentum,
    weight_decay=weight_decay
)
```

15. Set the learning rate scheduler.

```
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=lr_step_size, gamma=lr_gamma)
```

16. Iterate over epochs. Each epoch is a complete pass through the training data.

```
print("Start training")
for epoch in range(epochs):
    model.train()
    epoch_loss = 0
    len_dataset = 0
```

17. Iterate over steps. The data is processed in batches, and each step passes through a full batch.

```
for step, (image, target) in enumerate(data_loader):
```

18. Pass the image and target to the GPU device.

```
image, target = image.to(device), target.to(device)
```

The following is the core training logic:

a. The image is fed into the model.

b. The output is compared with the target in the training data to obtain the loss.

c. This loss is back propagated to all parameters that require optimization.

d. The optimizer updates the parameters based on the selected optimization algorithm.

```
        output = model(image)
        loss = criterion(output, target)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

The epoch loss is updated, and the step loss prints.

[5] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy

```
        epoch_loss += output.shape[0] * loss.item()
        len_dataset += output.shape[0];
        if step % 10 == 0:
            print('Epoch: ', epoch, '| step : %d' % step, '| train loss : %0.4f' % loss.item() )
    epoch_loss = epoch_loss / len_dataset
    print('Epoch: ', epoch, '| train loss :  %0.4f' % epoch_loss )
```

The learning rate is updated at the end of each epoch.

```
lr_scheduler.step()
```

After training for the epoch, the model evaluates against the validation dataset.

```
model.eval()
    with torch.inference_mode():
        running_loss = 0
        for step, (image, target) in enumerate(data_loader_test):
            image, target = image.to(device), target.to(device)

            output = model(image)
            loss = criterion(output, target)

            running_loss += loss.item()
    running_loss = running_loss / len(data_loader_test)
    print('Epoch: ', epoch, '| test loss : %0.4f' % running_loss )
```

19. Save the model for use in inferencing tasks.

```
# save model
torch.save(model.state_dict(), "trained_inception_v3.pt")
```

Plotting the train and test loss shows both metrics reducing over training epochs. This is demonstrated in Fig. 46.1.

Fig. 46.1: Inception v3 Train and Loss Graph

46.1.3.2 Custom Model with CIFAR-10 on PyTorch

The CIFAR-10 (Canadian Institute for Advanced Research) dataset is a subset of the Tiny Images dataset (which contains 80 million images of 32x32 collected from the Internet) and consists of 60,000 32x32 color images. The images are labeled with one of 10 mutually exclusive classes: airplane, motor car, bird, cat, deer, dog, frog, cruise ship, stallion, and truck (but not pickup truck). There are 6,000 images per class, with 5,000 training and 1,000 testing images per class. Let us prepare a custom model for classifying these images using the PyTorch framework and go step-by-step as illustrated below.

Follow these steps:

1. Import dependencies, including torch, os, and torchvision.

```
import torch
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plot
import numpy as np
```

2. The output of torchvision datasets is PILImage images of range [0, 1]. Transform them to Tensors of normalized range [-1, 1].

```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

   During each training step, a batch of images is processed to compute the loss gradient and perform the optimization. In the following setting, the size of the batch is determined.

```
batch_size = 4
```

3. Download the dataset train and test datasets as follows. Specify the batch size, shuffle the dataset once, and specify the number of workers to the number of CPU threads used by the data loader to perform efficient multi-process data loading.

```
train_set = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,␣
↪transform=transform)
train_loader = torch.utils.data.DataLoader(train_set, batch_size=batch_size, shuffle=True, num_
↪workers=2)
```

4. Follow the same procedure for the testing set.

```
test_set = TorchVision.datasets.CIFAR10(root='./data', train=False, download=True,␣
↪transform=transform)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=batch_size, shuffle=False, num_workers=2)
print ("teast set and test loader")
```

5. Specify the defined classes of images belonging to this dataset.

```
classes = ('Aeroplane', 'motorcar', 'bird', 'cat', 'deer', 'puppy', 'frog', 'stallion', 'cruise', 'truck')
print("defined classes")
```

6. Denormalize the images and then iterate over them.

```
global image_number
image_number = 0
def show_image(img):
```

(continues on next page)

(continued from previous page)

```
    global image_number
    image_number = image_number + 1
    img = img / 2 + 0.5     # de-normalizing input image
    npimg = img.numpy()
    plot.imshow(np.transpose(npimg, (1, 2, 0)))
    plot.savefig("fig{}.jpg".format(image_number))
    print("fig{}.jpg".format(image_number))
    plot.show()
data_iter = iter(train_loader)
images, labels = data_iter.next()
show_image(torchvision.utils.make_grid(images))
print(' '.join('%5s' % classes[labels[j]] for j in range(batch_size)))
print("image created and saved ")
```

7. Import the torch.nn for constructing neural networks and torch.nn.functional to use the convolution functions.

```
import torch.nn as nn
import torch.nn.functional as F
```

8. Define the CNN (Convolution Neural Networks) and relevant activation functions.

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
self.pool = nn.MaxPool2d(2, 2)
self.conv3 = nn.Conv2d(3, 6, 5)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
net = Net()
print("created Net() ")
```

9. Set the optimizer to Stochastic Gradient Descent.

```
import torch.optim as optim
```

10. Set the loss criteria. For this example, Cross Entropy Loss[Page 420, 5] is used.

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

11. Iterate over epochs. Each epoch is a complete pass through the training data.

```
for epoch in range(2):  # loop over the dataset multiple times

    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999:    # print every 2000 mini-batches
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0
print('Finished Training')
```

```
PATH = './cifar_net.pth'
torch.save(net.state_dict(), PATH)
print("saved model to path :",PATH)
net = Net()
net.load_state_dict(torch.load(PATH))
print("loding back saved model")
outputs = net(images)
_, predicted = torch.max(outputs, 1)
print('Predicted: ', ' '.join('%5s' % classes[predicted[j]] for j in range(4)))
correct = 0
total = 0
```

As this is not training, calculating the gradients for outputs is not required.

```
# calculate outputs by running images through the network
with torch.no_grad():
    for data in test_loader:
        images, labels = data
        # calculate outputs by running images through the network
        outputs = net(images)
        # the class with the highest energy is what you can choose as prediction
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
print('Accuracy of the network on the 10000 test images: %d %%' % ( 100 * correct / total))
# prepare to count predictions for each class
correct_pred = {classname: 0 for classname in classes}
total_pred = {classname: 0 for classname in classes}
```

```
# again no gradients needed
with torch.no_grad():
    for data in test_loader:
        images, labels = data
        outputs = net(images)
```

(continues on next page)

(continued from previous page)

```
        _, predictions = torch.max(outputs, 1)
        # collect the correct predictions for each class
        for label, prediction in zip(labels, predictions):
            if label == prediction:
                correct_pred[classes[label]] += 1
            total_pred[classes[label]] += 1
# print accuracy for each class
for classname, correct_count in correct_pred.items():
    accuracy = 100 * float(correct_count) / total_pred[classname]
    print("Accuracy for class {:5s} is: {:.1f} %".format(classname,accuracy))
```

46.1.3.3 Case Study: TensorFlow with Fashion MNIST

Fashion MNIST is a dataset that contains 70,000 grayscale images in 10 categories.

Implement and train a neural network model using the TensorFlow framework to classify images of clothing, like sneakers and shirts.

The dataset has 60,000 images you will use to train the network and 10,000 to evaluate how accurately the network learned to classify images. The Fashion MNIST dataset can be accessed via TensorFlow internal libraries.

Access the source code from the following repository:

https://github.com/ROCmSoftwarePlatform/tensorflow_fashionmnist/blob/main/fashion_mnist.py

To understand the code step by step, follow these steps:

1. Import libraries like TensorFlow, NumPy, and Matplotlib to train the neural network and calculate and plot graphs.

   ```
   import tensorflow as tf
   import numpy as np
   import matplotlib.pyplot as plt
   ```

2. To verify that TensorFlow is installed, print the version of TensorFlow by using the below print statement:

   ```
   print(tf.__version__) r
   ```

3. Load the dataset from the available internal libraries to analyze and train a neural network upon the MNIST Fashion Dataset. Loading the dataset returns four NumPy arrays. The model uses the training set arrays, train_images and train_labels, to learn.

4. The model is tested against the test set, test_images, and test_labels arrays.

   ```
   fashion_mnist = tf.keras.datasets.fashion_mnist
   (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
   ```

   Since you have 10 types of images in the dataset, assign labels from zero to nine. Each image is assigned one label. The images are 28x28 NumPy arrays, with pixel values ranging from zero to 255.

5. Each image is mapped to a single label. Since the class names are not included with the dataset, store them, and later use them when plotting the images:

   ```
   class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat','Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle␣
   ↪boot']
   ```

6. Use this code to explore the dataset by knowing its dimensions:

```
train_images.shape
```

7. Use this code to print the size of this training set:

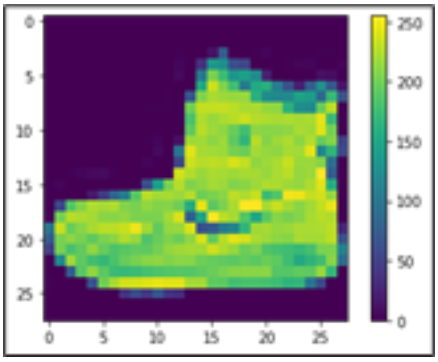
```
print(len(train_labels))
```

8. Use this code to print the labels of this training set:

```
print(train_labels)
```

9. Preprocess the data before training the network, and you can start inspecting the first image, as its pixels will fall in the range of zero to 255.

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```



10. From the above picture, you can see that values are from zero to 255. Before training this on the neural network, you must bring them in the range of zero to one. Hence, divide the values by 255.

```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

11. To ensure the data is in the correct format and ready to build and train the network, display the first 25 images from the training set and the class name below each image.

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

The basic building block of a neural network is the layer. Layers extract representations from the data fed into them. Deep Learning consists of chaining together simple layers. Most layers, such as tf.keras.layers.Dense, have parameters that are learned during training.


Ankle boot
T-shirt/top
T-shirt/top
Dress
T-shirt/top
Pullover
Sneaker
Pullover
Sandal
Sandal
T-shirt/top
Ankle boot
Sandal
Sandal
Sneaker
Ankle boot
Trouser
T-shirt/top
Shirt
Coat
Dress
Trouser
Coat
Bag
Coat

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

- The first layer in this network tf.keras.layers.Flatten transforms the format of the images from a two-dimensional array (of 28 x 28 pixels) to a one-dimensional array (of 28 * 28 = 784 pixels). Think of this layer as unstacking rows of pixels in the image and lining them up. This layer has no parameters to learn; it only reformats the data.
- After the pixels are flattened, the network consists of a sequence of two tf.keras.layers.Dense layers. These are densely connected or fully connected neural layers. The first Dense layer has 128 nodes (or neurons). The second (and last) layer returns a logits array with a length of 10. Each node contains a score that indicates the current image belongs to one of the 10 classes.

12. You must add the Loss function, Metrics, and Optimizer at the time of model compilation.

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

- Loss function —This measures how accurate the model is during training when you are looking to minimize this function to "steer" the model in the right direction.
- Optimizer —This is how the model is updated based on the data it sees and its loss function.
- Metrics —This is used to monitor the training and testing steps.

The following example uses accuracy, the fraction of the correctly classified images.

To train the neural network model, follow these steps:

1. Feed the training data to the model. The training data is in the train_images and train_labels arrays in this example. The model learns to associate images and labels.
2. Ask the model to make predictions about a test set—in this example, the test_images array.
3. Verify that the predictions match the labels from the test_labels array.
4. To start training, call the model.fit method because it "fits" the model to the training data.

```
model.fit(train_images, train_labels, epochs=10)
```

5. Compare how the model will perform on the test dataset.

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

6. With the model trained, you can use it to make predictions about some images: the model's linear outputs and logits. Attach a softmax layer to convert the logits to probabilities, making it easier to interpret.

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])

predictions = probability_model.predict(test_images)
```

7. The model has predicted the label for each image in the testing set. Look at the first prediction:

```
predictions[0]
```

A prediction is an array of 10 numbers. They represent the model's "confidence" that the image corresponds to each of the 10 different articles of clothing. You can see which label has the highest confidence value:

```
np.argmax(predictions[0])
```

8. Plot a graph to look at the complete set of 10 class predictions.

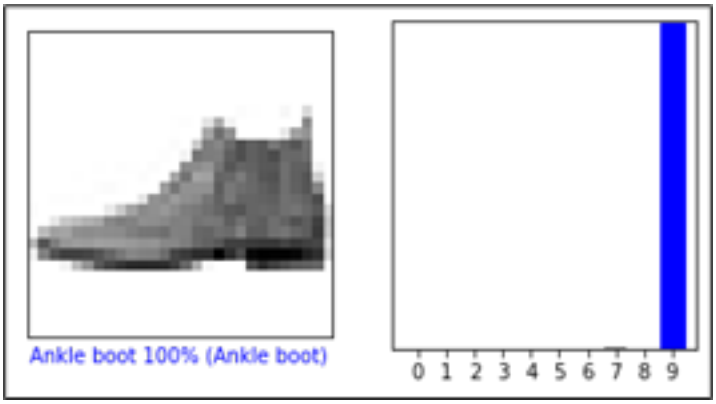
```
def plot_image(i, predictions_array, true_label, img):
true_label, img = true_label[i], img[i]
plt.grid(False)
plt.xticks([])
plt.yticks([])

plt.imshow(img, cmap=plt.cm.binary)

predicted_label = np.argmax(predictions_array)
if predicted_label == true_label:
    color = 'blue'
else:
    color = 'red'

plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
true_label = true_label[i]
plt.grid(False)
plt.xticks(range(10))
plt.yticks([])
thisplot = plt.bar(range(10), predictions_array, color="#777777")
plt.ylim([0, 1])
predicted_label = np.argmax(predictions_array)

thisplot[predicted_label].set_color('red')
thisplot[true_label].set_color('blue')
```

9. With the model trained, you can use it to make predictions about some images. Review the 0-th image predictions and the prediction array. Correct prediction labels are blue, and incorrect prediction labels are red. The number gives the percentage (out of 100) for the predicted label.

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```

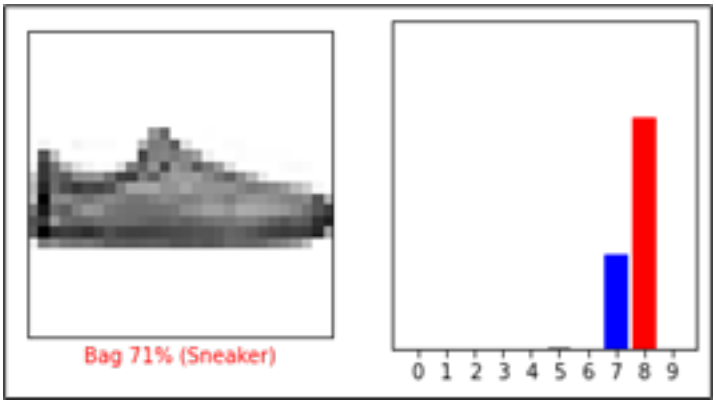
```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
```

(continues on next page)



(continued from previous page)

```
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



10. Use the trained model to predict a single image.

```
# Grab an image from the test dataset.
img = test_images[1]
print(img.shape)
```

11. tf.keras models are optimized to make predictions on a batch, or collection, of examples at once. Accordingly, even though you are using a single image, you must add it to a list.

```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

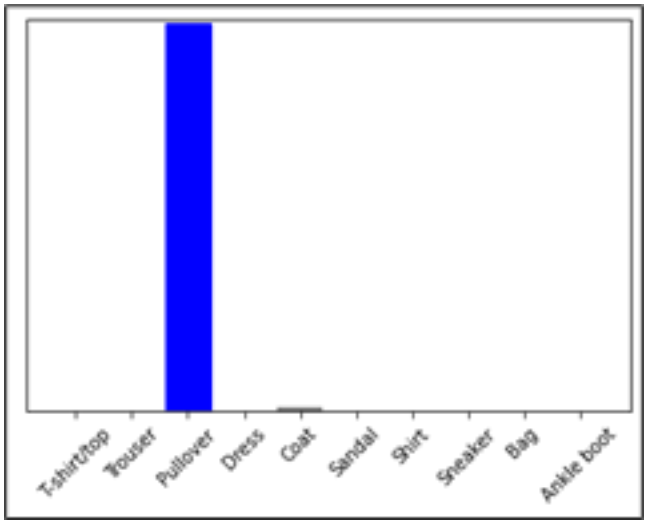
12. Predict the correct label for this image.

```
predictions_single = probability_model.predict(img)

print(predictions_single)

plot_value_array(1, predictions_single[0], test_labels)
```

(continues on next page)

(continued from previous page)

```
_ = plt.xticks(range(10), class_names, rotation=45)
plt.show()
```



13. tf.keras.Model.predict returns a list of lists—one for each image in the batch of data. Grab the predictions for our (only) image in the batch.

```
np.argmax(predictions_single[0])
```

46.1.3.4 Case Study: TensorFlow with Text Classification

This procedure demonstrates text classification starting from plain text files stored on disk. You will train a binary classifier to perform sentiment analysis on an IMDB dataset. At the end of the notebook, there is an exercise for you to try in which you will train a multi-class classifier to predict the tag for a programming question on Stack Overflow.

Follow these steps:

1. Import the necessary libraries.

```
import matplotlib.pyplot as plt
import os
import re
import shutil
import string
import tensorflow as tf

from tensorflow.keras import layers
from tensorflow.keras import losses
```

2. Get the data for the text classification, and extract the database from the given link of IMDB.

```
url = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"

dataset = tf.keras.utils.get_file("aclImdb_v1", url,
                                  untar=True, cache_dir='.',
                                  cache_subdir='')
```

```
Downloading data from https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
84131840/84125825 [==============================] - 1s 0us/step
84149932/84125825 [==============================] - 1s 0us/step
```

3. Fetch the data from the directory.

```
dataset_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')
print(os.listdir(dataset_dir))
```

4. Load the data for training purposes.

```
train_dir = os.path.join(dataset_dir, 'train')
os.listdir(train_dir)
```

```
['labeledBow.feat',
'urls_pos.txt',
'urls_unsup.txt',
'unsup',
'pos',
'unsupBow.feat',
'urls_neg.txt',
'neg']
```

5. The directories contain many text files, each of which is a single movie review. To look at one of them, use the following:

```
sample_file = os.path.join(train_dir, 'pos/1181_9.txt')
with open(sample_file) as f:
print(f.read())
```

6. As the IMDB dataset contains additional folders, remove them before using this utility.

```
remove_dir = os.path.join(train_dir, 'unsup')
shutil.rmtree(remove_dir)
batch_size = 32
seed = 42
```

7. The IMDB dataset has already been divided into train and test but lacks a validation set. Create a validation set using an 80:20 split of the training data by using the validation_split argument below:

```
raw_train_ds=tf.keras.utils.text_dataset_from_directory('aclImdb/train',batch_size=batch_size,␣
↪validation_split=0.2,subset='training', seed=seed)
```

8. As you will see in a moment, you can train a model by passing a dataset directly to model.fit. If you are new to tf.data, you can also iterate over the dataset and print a few examples as follows:

```
for text_batch, label_batch in raw_train_ds.take(1):
for i in range(3):
    print("Review", text_batch.numpy()[i])
    print("Label", label_batch.numpy()[i])
```

9. The labels are zero or one. To see which of these correspond to positive and negative movie reviews, check the class_names property on the dataset.

```
print("Label 0 corresponds to", raw_train_ds.class_names[0])
print("Label 1 corresponds to", raw_train_ds.class_names[1])
```

10. Next, create validation and test the dataset. Use the remaining 5,000 reviews from the training set for validation into two classes of 2,500 reviews each.

```
raw_val_ds = tf.keras.utils.text_dataset_from_directory('aclImdb/train',
batch_size=batch_size,validation_split=0.2,subset='validation', seed=seed)

raw_test_ds =
tf.keras.utils.text_dataset_from_directory(
    'aclImdb/test',
    batch_size=batch_size)
```

To prepare the data for training, follow these steps:

1. Standardize, tokenize, and vectorize the data using the helpful tf.keras.layers.TextVectorization layer.

```
def custom_standardization(input_data):
lowercase = tf.strings.lower(input_data)
stripped_html = tf.strings.regex_replace(lowercase, '<br/>', ' ')
return tf.strings.regex_replace(stripped_html,                    '[%s]' % re.escape(string.
↪punctuation),'')
```

2. Create a TextVectorization layer. Use this layer to standardize, tokenize, and vectorize our data. Set the output_mode to int to create unique integer indices for each token. Note that we are using the default split function and the custom standardization function you defined above. You will also define some constants for the model, like an explicit maximum sequence_length, which will cause the layer to pad or truncate sequences to exactly sequence_length values.

```
max_features = 10000
sequence_length = 250
vectorize_layer = layers.TextVectorization(
    standardize=custom_standardization,
    max_tokens=max_features,
    output_mode='int',
    output_sequence_length=sequence_length)
```

3. Call adapt to fit the state of the preprocessing layer to the dataset. This causes the model to build an index of strings to integers.

```
# Make a text-only dataset (without labels), then call adapt
train_text = raw_train_ds.map(lambda x, y: x)
vectorize_layer.adapt(train_text)
```

4. Create a function to see the result of using this layer to preprocess some data.

```
def vectorize_text(text, label):
text = tf.expand_dims(text, -1)
return vectorize_layer(text), label

text_batch, label_batch = next(iter(raw_train_ds))
first_review, first_label = text_batch[0], label_batch[0]
print("Review", first_review)
print("Label", raw_train_ds.class_names[first_label])
print("Vectorized review", vectorize_text(first_review, first_label))
```

5. As you can see above, each token has been replaced by an integer. Look up the token (string) that each integer corresponds to by calling get_vocabulary() on the layer.

```
Vectorized review (<tf.Tensor: shape=(1, 250), dtype=int64, numpy=
array([[9257,   15,   28,    5,    2,  241,   91,    5,   30,   58,    4,
           1, 1011,   36,  262,   38,    9, 3891,   35,    2,    1,   43,
         382, 5223,   16,    4, 1113,   12,   43, 5739,  300,    2,   83,
         225,   55, 3209, 3898,   20,  973,   18,   51,   55, 3209,  250,
         320,   34,   40, 4386,    3,  294,   58,   44,    6, 2911,    4,
        6757,   92,   22,  184,    2, 4916,  100, 1221,  336,  161, 1199,
        1484,    4,  808,  568,   31,    4,  839,  270,    3,  741,  741,
          50,  774,   31,   30, 2070,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0]])>, <tf.Tensor: shape=(), dtype=int32, num
py=0>)
1287 --->  silent
 313 --->  night
Vocabulary size: 10000
```

```
print("1287 ---> ",vectorize_layer.get_vocabulary()[1287])
print(" 313 ---> ",vectorize_layer.get_vocabulary()[313])
print('Vocabulary size: {}'.format(len(vectorize_layer.get_vocabulary())))
```

6. You are nearly ready to train your model. As a final preprocessing step, apply the TextVectorization layer we created earlier to train, validate, and test the dataset.

```
train_ds = raw_train_ds.map(vectorize_text)
val_ds = raw_val_ds.map(vectorize_text)
test_ds = raw_test_ds.map(vectorize_text)
```

The cache() function keeps data in memory after it is loaded off disk. This ensures the dataset does not become a bottleneck while training your model. If your dataset is too large to fit into memory, you can also use this method to create a performant on-disk cache, which is more efficient to read than many small files.

The prefetch() function overlaps data preprocessing and model execution while training.

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

7. Create your neural network.

```
embedding_dim = 16
model = tf.keras.Sequential([layers.Embedding(max_features + 1, embedding_dim),layers.Dropout(0.2),
↪layers.GlobalAveragePooling1D(),
layers.Dropout(0.2),layers.Dense(1)])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 16)          160016

 dropout (Dropout)           (None, None, 16)          0

 global_average_pooling1d (G  (None, 16)               0
 lobalAveragePooling1D)

 dropout_1 (Dropout)         (None, 16)                0

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 160,033
Trainable params: 160,033
Non-trainable params: 0
```

8. A model needs a loss function and an optimizer for training. Since this is a binary classification problem and the model outputs a probability (a single-unit layer with a sigmoid activation), use losses.BinaryCrossentropy loss function.

```
model.compile(loss=losses.BinaryCrossentropy(from_logits=True),
optimizer='adam',metrics=tf.metrics.BinaryAccuracy(threshold=0.0))
```

9. Train the model by passing the dataset object to the fit method.

```
epochs = 10
history = model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

```
Epoch 1/10
625/625 [==============================] - 5s 7ms/step - loss: 0.6660 - binary_accuracy: 0.6914 -
val_loss: 0.6174 - val_binary_accuracy: 0.7718
Epoch 2/10
625/625 [==============================] - 4s 7ms/step - loss: 0.5508 - binary_accuracy: 0.8005 -
val_loss: 0.5000 - val_binary_accuracy: 0.8226
Epoch 3/10
625/625 [==============================] - 4s 7ms/step - loss: 0.4450 - binary_accuracy: 0.8442 -
val_loss: 0.4210 - val_binary_accuracy: 0.8454
Epoch 4/10
625/625 [==============================] - 4s 7ms/step - loss: 0.3784 - binary_accuracy: 0.8666 -
val_loss: 0.3742 - val_binary_accuracy: 0.8608
Epoch 5/10
625/625 [==============================] - 4s 7ms/step - loss: 0.3361 - binary_accuracy: 0.8787 -
val_loss: 0.3454 - val_binary_accuracy: 0.8670
Epoch 6/10
625/625 [==============================] - 4s 7ms/step - loss: 0.3050 - binary_accuracy: 0.8874 -
val_loss: 0.3263 - val_binary_accuracy: 0.8714
Epoch 7/10
625/625 [==============================] - 4s 7ms/step - loss: 0.2822 - binary_accuracy: 0.8961 -
val_loss: 0.3130 - val_binary_accuracy: 0.8728
Epoch 8/10
625/625 [==============================] - 4s 7ms/step - loss: 0.2626 - binary_accuracy: 0.9043 -
val_loss: 0.3035 - val_binary_accuracy: 0.8764
Epoch 9/10
625/625 [==============================] - 4s 7ms/step - loss: 0.2464 - binary_accuracy: 0.9091 -
val_loss: 0.2969 - val_binary_accuracy: 0.8770
Epoch 10/10
625/625 [==============================] - 4s 7ms/step - loss: 0.2332 - binary_accuracy: 0.9160 -
val_loss: 0.2922 - val_binary_accuracy: 0.8790
```

10. See how the model performs. Two values are returned: loss (a number representing our error; lower values are better) and accuracy.

```
loss, accuracy = model.evaluate(test_ds)

print("Loss: ", loss)
print("Accuracy: ", accuracy)
```

Note: model.fit() returns a History object that contains a dictionary with everything that happened during training.

```
history_dict = history.history
history_dict.keys()
```

11. Four entries are for each monitored metric during training and validation. Use these to plot the training and validation loss for comparison, as well as the training and validation accuracy:

```
acc = history_dict['binary_accuracy']
val_acc = history_dict['val_binary_accuracy']
loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(acc) + 1)

# ”bo” is for ”blue dot”
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for ”solid blue line”
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Fig. 46.2 and Fig. 46.3 illustrate the training and validation loss and the training and validation accuracy.

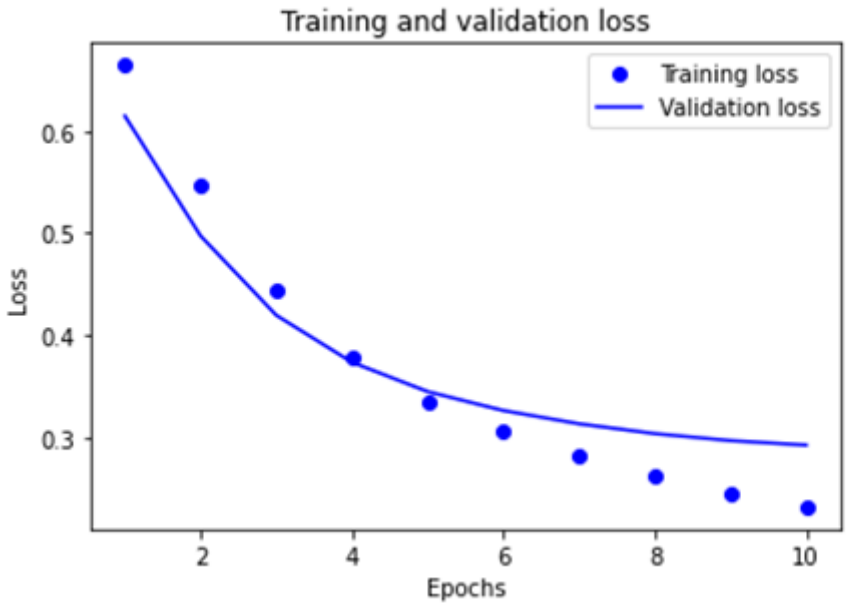


Fig. 46.2: Training and Validation Loss

12. Export the model.

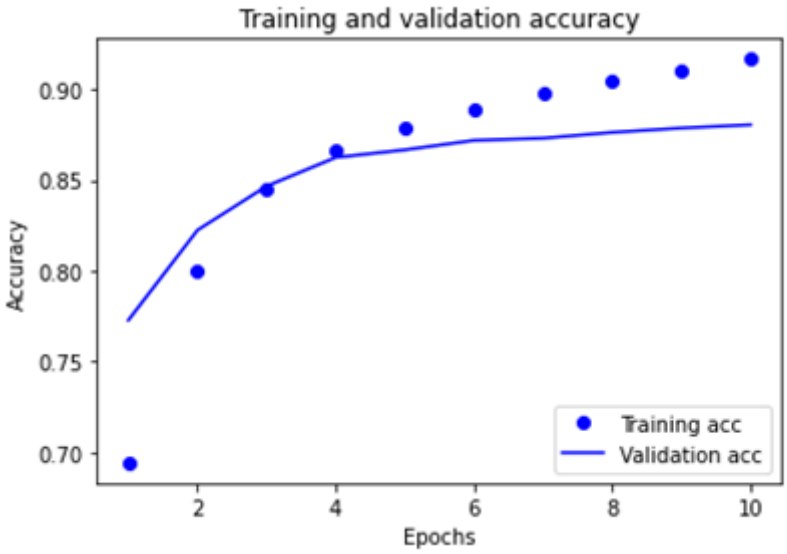


Fig. 46.3: Training and Validation Accuracy

```
export_model = tf.keras.Sequential([
vectorize_layer,
model,
layers.Activation('sigmoid')
])

export_model.compile(
    loss=losses.BinaryCrossentropy(from_logits=False), optimizer="adam", metrics=['accuracy']
)

# Test it with `raw_test_ds`, which yields raw strings
loss, accuracy = export_model.evaluate(raw_test_ds)
print(accuracy)
```

13. To get predictions for new examples, call model.predict().

```
examples = [
"The movie was great!",
"The movie was okay.",
"The movie was terrible..."
]

export_model.predict(examples)
```

46.1.4 References

46.2 Inference Optimization with MIGraphX

The following sections cover inferencing and introduces MIGraphX.

46.2.1 Inference

The inference is where capabilities learned during Deep Learning training are put to work. It refers to using a fully trained neural network to make conclusions (predictions) on unseen data that the model has never interacted with before. Deep Learning inferencing is achieved by feeding new data, such as new images, to the network, giving the Deep Neural Network a chance to classify the image.

Taking our previous example of MNIST, the DNN can be fed new images of handwritten digit images, allowing the neural network to classify digits. A fully trained DNN should make accurate predictions about what an image represents, and inference cannot happen without training.

46.2.2 MIGraphX Introduction

MIGraphX is a graph compiler focused on accelerating the Machine Learning inference that can target AMD GPUs and CPUs. MIGraphX accelerates the Machine Learning models by leveraging several graph-level transformations and optimizations. These optimizations include:

- Operator fusion
- Arithmetic simplifications
- Dead-code elimination
- Common subexpression elimination (CSE)
- Constant propagation

After doing all these transformations, MIGraphX emits code for the AMD GPU by calling to MIOpen or rocBLAS or creating HIP kernels for a particular operator. MIGraphX can also target CPUs using DNNL or ZenDNN libraries.

MIGraphX provides easy-to-use APIs in C++ and Python to import machine models in ONNX or TensorFlow. Users can compile, save, load, and run these models using the MIGraphX C++ and Python APIs. Internally, MIGraphX parses ONNX or TensorFlow models into internal graph representation where each operator in the model gets mapped to an operator within MIGraphX. Each of these operators defines various attributes such as:

- Number of arguments
- Type of arguments
- Shape of arguments

After optimization passes, all these operators get mapped to different kernels on GPUs or CPUs.

After importing a model into MIGraphX, the model is represented as migraphx::program. migraphx::program is made up of migraphx::module. The program can consist of several modules, but it always has one main_module. Modules are made up of migraphx::instruction_ref. Instructions contain the migraphx::op and arguments to the operator.

46.2.3 Installing MIGraphX

There are three options to get started with MIGraphX installation. MIGraphX depends on ROCm libraries; assume that the machine has ROCm installed.

46.2.3.1 Option 1: Installing Binaries

To install MIGraphX on Debian-based systems like Ubuntu, use the following command:

```
sudo apt update && sudo apt install -y migraphx
```

The header files and libraries are installed under /opt/rocm-\<version\>, where <version> is the ROCm version.

46.2.3.2 Option 2: Building from Source

There are two ways to build the MIGraphX sources.

- Use the ROCm build tool - This approach uses rbuild to install the prerequisites and build the libraries with just one command.

  or

- Use CMake - This approach uses a script to install the prerequisites, then uses CMake to build the source.

For detailed steps on building from source and installing dependencies, refer to the following README file:

https://github.com/ROCmSoftwarePlatform/AMDMIGraphX#building-from-source

46.2.3.3 Option 3: Use Docker

To use Docker, follow these steps:

1. The easiest way to set up the development environment is to use Docker. To build Docker from scratch, first clone the MIGraphX repository by running:

   ```
   git clone --recursive https://github.com/ROCmSoftwarePlatform/AMDMIGraphX
   ```

2. The repository contains a Dockerfile from which you can build a Docker image as:

   ```
   docker build -t migraphx .
   ```

3. Then to enter the development environment, use Docker run:

   ```
   docker run --device='/dev/kfd' --device='/dev/dri' -v=`pwd`:/code/AMDMIGraphX -w /code/
   ↪AMDMIGraphX --group-add video -it migraphx
   ```

The Docker image contains all the prerequisites required for the installation, so users can go to the folder /code/AMDMIGraphX and follow the steps mentioned in Option 2: Building from Source.

46.2.4 MIGraphX Example

MIGraphX provides both C++ and Python APIs. The following sections show examples of both using the Inception v3 model. To walk through the examples, fetch the Inception v3 ONNX model by running the following:

```
import torch
import torchvision.models as models
inception = models.inception_v3(pretrained=True)
torch.onnx.export(inception,torch.randn(1,3,299,299), "inceptioni1.onnx")
```

This will create inceptioni1.onnx, which can be imported in MIGraphX using C++ or Python API.

46.2.4.1 MIGraphX Python API

Follow these steps:

1. To import the MIGraphX module in Python script, set PYTHONPATH to the MIGraphX libraries installation. If binaries are installed using steps mentioned in Option 1: Installing Binaries, perform the following action:

   ```
   export PYTHONPATH=$PYTHONPATH:/opt/rocm/
   ```

2. The following script shows the usage of Python API to import the ONNX model, compile it, and run inference on it. Set LD_LIBRARY_PATH to /opt/rocm/ if required.

   ```
   # import migraphx and numpy
   import migraphx
   import numpy as np
   # import and parse inception model
   model = migraphx.parse_onnx("inceptioni1.onnx")
   # compile model for the GPU target
   model.compile(migraphx.get_target("gpu"))
   # optionally print compiled model
   model.print()
   # create random input image
   input_image = np.random.rand(1, 3, 299, 299).astype('float32')
   # feed image to model, 'x.1` is the input param name
   results = model.run({'x.1': input_image})
   # get the results back
   result_np = np.array(results[0])
   # print the inferred class of the input image
   print(np.argmax(result_np))
   ```

   Find additional examples of Python API in the /examples directory of the MIGraphX repository.

46.2.5 MIGraphX C++ API

Follow these steps:

1. The following is a minimalist example that shows the usage of MIGraphX C++ API to load ONNX file, compile it for the GPU, and run inference on it. To use MIGraphX C++ API, you only need to load the migraphx.hpp file. This example runs inference on the Inception v3 model.

```
#include <vector>
#include <string>
#include <algorithm>
#include <ctime>
#include <random>
#include <migraphx/migraphx.hpp>

int main(int argc, char** argv)
{
    migraphx::program prog;
    migraphx::onnx_options onnx_opts;
    // import and parse onnx file into migraphx::program
    prog = parse_onnx("inceptioni1.onnx", onnx_opts);
    // print imported model
    prog.print();
    migraphx::target targ = migraphx::target("gpu");
    migraphx::compile_options comp_opts;
    comp_opts.set_offload_copy();
    // compile for the GPU
    prog.compile(targ, comp_opts);
    // print the compiled program
    prog.print();
    // randomly generate input image
    // of shape (1, 3, 299, 299)
    std::srand(unsigned(std::time(nullptr)));
    std::vector<float> input_image(1*299*299*3);
    std::generate(input_image.begin(), input_image.end(), std::rand);
    // users need to provide data for the input
    // parameters in order to run inference
    // you can query into migraph program for the parameters
    migraphx::program_parameters prog_params;
    auto param_shapes = prog.get_parameter_shapes();
    auto input        = param_shapes.names().front();
    // create argument for the parameter
    prog_params.add(input, migraphx::argument(param_shapes[input], input_image.data()));
    // run inference
    auto outputs = prog.eval(prog_params);
    // read back the output
    float* results = reinterpret_cast<float*>(outputs[0].data());
    float* max     = std::max_element(results, results + 1000);
    int answer = max - results;
    std::cout << "answer: " << answer << std::endl;
}
```

2. To compile this program, you can use CMake and you only need to link the migraphx::c library to use the MIGraphX C++ API. The following is the CMakeLists.txt file that can build the earlier example:

```
cmake_minimum_required(VERSION 3.5)
project (CAI)
```

(continues on next page)

(continued from previous page)

```
set (CMAKE_CXX_STANDARD 14)
set (EXAMPLE inception_inference)

list (APPEND CMAKE_PREFIX_PATH /opt/rocm/hip /opt/rocm)
find_package (migraphx)

message("source file: " ${EXAMPLE}.cpp " ---> bin: " ${EXAMPLE})
add_executable(${EXAMPLE} ${EXAMPLE}.cpp)

target_link_libraries(${EXAMPLE} migraphx::c)
```

3. To build the executable file, run the following from the directory containing the inception_inference.cpp file:

```
mkdir build
cd build
cmake ..
make -j$(nproc)
./inception_inference
```

Note:

Set `LD_LIBRARY_PATH` to `/opt/rocm/lib` if required during the build. Additional examples can be found in the MIGraphX repository under the `/examples/` directory.

46.2.6 Tuning MIGraphX

MIGraphX uses MIOpen kernels to target AMD GPU. For the model compiled with MIGraphX, tune MIOpen to pick the best possible kernel implementation. The MIOpen tuning results in a significant performance boost. Tuning can be done by setting the environment variable MIOPEN_FIND_ENFORCE=3.

Note:

The tuning process can take a long time to finish.

Example: The average inference time of the inception model example shown previously over 100 iterations using untuned kernels is 0.01383ms. After tuning, it reduces to 0.00459ms, which is a 3x improvement. This result is from ROCm v4.5 on a MI100 GPU.

Note:

The results may vary depending on the system configurations.

For reference, the following code snippet shows inference runs for only the first 10 iterations for both tuned and untuned kernels:

```
### UNTUNED ###
iterator : 0
```

(continues on next page)

(continued from previous page)

```
Inference complete
Inference time: 0.063ms
iterator : 1
Inference complete
Inference time: 0.008ms
iterator : 2
Inference complete
Inference time: 0.007ms
iterator : 3
Inference complete
Inference time: 0.007ms
iterator : 4
Inference complete
Inference time: 0.007ms
iterator : 5
Inference complete
Inference time: 0.008ms
iterator : 6
Inference complete
Inference time: 0.007ms
iterator : 7
Inference complete
Inference time: 0.028ms
iterator : 8
Inference complete
Inference time: 0.029ms
iterator : 9
Inference complete
Inference time: 0.029ms

### TUNED ###
iterator : 0
Inference complete
Inference time: 0.063ms
iterator : 1
Inference complete
Inference time: 0.004ms
iterator : 2
Inference complete
Inference time: 0.004ms
iterator : 3
Inference complete
Inference time: 0.004ms
iterator : 4
Inference complete
Inference time: 0.004ms
iterator : 5
Inference complete
Inference time: 0.004ms
iterator : 6
Inference complete
Inference time: 0.004ms
iterator : 7
Inference complete
Inference time: 0.004ms
iterator : 8
```

(continues on next page)

(continued from previous page)

```
Inference complete
Inference time: 0.004ms
iterator : 9
Inference complete
Inference time: 0.004ms
```

46.2.6.1 YModel

The best inference performance through MIGraphX is conditioned upon having tuned kernel configurations stored in a /home local User Database (DB). If a user were to move their model to a different server or allow a different user to use it, they would have to run through the MIOpen tuning process again to populate the next User DB with the best kernel configurations and corresponding solvers.

Tuning is time consuming, and if the users have not performed tuning, they would see discrepancies between expected or claimed inference performance and actual inference performance. This has led to repetitive and time-consuming tuning tasks for each user.

MIGraphX introduces a feature, known as YModel, that stores the kernel config parameters found during tuning into a .mxr file. This ensures the same level of expected performance, even when a model is copied to a different user/system.

The YModel feature is available starting from ROCm 5.4.1 and UIF 1.1.

46.2.6.1.1 YModel Example

Through the migraphx-driver functionality, you can generate .mxr files with tuning information stored inside it by passing additional --binary --output model.mxr to migraphx-driver along with the rest of the necessary flags.

For example, to generate .mxr file from the ONNX model, use the following:

```
./path/to/migraphx-driver compile --onnx resnet50.onnx --enable-offload-copy --binary --output resnet50.mxr
```

To run generated .mxr files through migraphx-driver, use the following:

```
./path/to/migraphx-driver run --migraphx resnet50.mxr --enable-offload-copy
```

Alternatively, you can use the MIGraphX C++ or Python API to generate .mxr file. Refer to Fig. 46.4 for an example.

```
// parse onnx file, compile and do other processing

    auto p1                = migraphx::parse_onnx("resnet50.onnx");

// once model has been processed, it can be saved as mxr file as follows

    std::string filename = "resnet50.mxr";

    migraphx::save(p1, filename.c_str());

  // to load mxr file, use load().

    auto p2 = migraphx::load(filename.c_str());
```

Fig. 46.4: Generating a .mxr File

CHAPTER

FORTYSEVEN

ABOUT ROCM DOCUMENTATION

ROCm documentation is made available under open source licenses. Documentation is built using open source toolchains. Contributions to our documentation is encouraged and welcome. As a contributor, please familiarize yourself with our documentation toolchain.

47.1 rocm-docs-core

rocm-docs-core is an AMD-maintained project that applies customization for our documentation. This project is the tool most ROCm repositories use as part of the documentation build. It is also available as a pip package on PyPI.

See the user and developer guides for rocm-docs-core at rocm-docs-core documentation.

47.2 Sphinx

Sphinx is a documentation generator originally used for Python. It is now widely used in the Open Source community. Originally, Sphinx supported reStructuredText (RST) based documentation, but Markdown support is now available. ROCm documentation plans to default to Markdown for new projects. Existing projects using RST are under no obligation to convert to Markdown. New projects that believe Markdown is not suitable should contact the documentation team prior to selecting RST.

47.3 Read the Docs

Read the Docs is the service that builds and hosts the HTML documentation generated using Sphinx to our end users.

47.4 Doxygen

Doxygen is a documentation generator that extracts information from inline code. ROCm projects typically use Doxygen for public API documentation unless the upstream project uses a different tool.

47.4.1 Breathe

Breathe is a Sphinx plugin to integrate Doxygen content.

47.4.2 MyST

Markedly Structured Text (MyST) is an extended flavor of Markdown (CommonMark) influenced by re-StructuredText (RST) and Sphinx. It is integrated into ROCm documentation by the Sphinx extension myst-parser. A cheat sheet that showcases how to use the MyST syntax is available over at the Jupyter reference.

47.4.3 Sphinx External TOC

Sphinx External Table of Contents (TOC) is a Sphinx extension used for ROCm documentation navigation. This tool generates a navigation menu on the left based on a YAML file that specifies the table of contents. It was selected due to its flexibility that allows scripts to operate on the YAML file. Please transition to this file for the project's navigation. You can see the _toc.yml.in file in this repository in the docs/sphinx folder for an example.

47.4.4 Sphinx Book Theme

Sphinx Book Theme is a Sphinx theme that defines the base appearance for ROCm documentation. ROCm documentation applies some customization, such as a custom header and footer on top of the Sphinx Book Theme.

47.4.5 Sphinx Design

Sphinx Design is a Sphinx extension that adds design functionality. ROCm documentation uses Sphinx Design for grids, cards, and synchronized tabs.

CHAPTER

FORTYEIGHT

CONTRIBUTING TO ROCM DOCS

AMD values and encourages the ROCm community to contribute to our code and documentation. This repository is focused on ROCm documentation and this contribution guide describes the recommended method for creating and modifying our documentation.

While interacting with ROCm Documentation, we encourage you to be polite and respectful in your contributions, content or otherwise. Authors, maintainers of these docs act on good intentions and to the best of their knowledge. Keep that in mind while you engage. Should you have issues with contributing itself, refer to discussions on the GitHub repository.

For additional information on documentation functionalities, see the user and developer guides for rocm-docs-core at rocm-docs-core documentation.

48.1 Supported Formats

Our documentation includes both Markdown and RST files. Markdown is encouraged over RST due to the lower barrier to participation. GitHub-flavored Markdown is preferred for all submissions as it renders accurately on our GitHub repositories. For existing documentation, MyST Markdown is used to implement certain features unsupported in GitHub Markdown. This is not encouraged for new documentation. AMD will transition to stricter use of GitHub-flavored Markdown with a few caveats. ROCm documentation also uses Sphinx Design in our Markdown and RST files. We also use Breathe syntax for Doxygen documentation in our Markdown files. See GitHub's guide on writing and formatting on GitHub as a starting point.

ROCm documentation adds additional requirements to Markdown and RST based files as follows:

- Level one headers are only used for page titles. There must be only one level 1 header per file for both Markdown and Restructured Text.
- Pass markdownlint check via our automated GitHub action on a Pull Request (PR). See the rocm-docs-core linting user guide for more details.

48.2 Filenames and folder structure

Please use snake case (all lower case letters and underscores instead of spaces) for file names. For example, example_file_name.md. Our documentation follows Pitchfork for folder structure. All documentation is in /docs except for special files like the contributing guide in the / folder. All images used in the documentation are placed in the /docs/data folder.

48.3 Language and Style

Adopt Microsoft CPP-Docs guidelines for Voice and Tone.

ROCm documentation templates to be made public shortly. ROCm templates dictate the recommended structure and flow of the documentation. Guidelines on how to integrate figures, equations, and tables are all based off MyST.

Font size and selection, page layout, white space control, and other formatting details are controlled via rocm-docs-core. Raise issues in rocm-docs-core for any formatting concerns and changes requested.

48.4 More

For more topics, such as submitting feedback and ways to build documentation, see the Contributing Section at rocm.docs.amd.com

48.5 Building Documentation

While contributing, one may build the documentation locally on the command-line or rely on Continuous Integration for previewing the resulting HTML pages in a browser.

48.5.1 Pull Request documentation builds

When opening a PR to the develop branch on GitHub, the page corresponding to the PR (https://github.com/RadeonOpenCompute/ROCm/pull/<pr_number>) will have a summary at the bottom. This requires the user be logged in to GitHub.

- There, click Show all checks and Details of the Read the Docs pipeline. It will take you to a URL of the form https://readthedocs.com/projects/advanced-micro-devices-rocm/builds/<some_build_num>/
  - The list of commands shown are the exact ones used by CI to produce a render of the documentation.
- There, click on the small blue link View docs (which is not the same as the bigger button with the same text). It will take you to the built HTML site with a URL of the form https://advanced-micro-devices-demo--<pr_number>.com.readthedocs.build/projects/alpha/en/<pr_number>/.

48.5.2 Build documentation from the Command Line

Python versions known to build documentation:

- 3.8

To build the docs locally using Python Virtual Environment (venv), execute the following commands from the project root:

```
python3 -mvenv .venv
# Windows
.venv/Scripts/python -m pip install -r docs/sphinx/requirements.txt
.venv/Scripts/python -m sphinx -T -E -b html -d _build/doctrees -D language=en docs _build/html
```

(continues on next page)

(continued from previous page)

```
# Linux
.venv/bin/python     -m pip install -r docs/sphinx/requirements.txt
.venv/bin/python     -m sphinx -T -E -b html -d _build/doctrees -D language=en docs _build/html
```

Then open up _build/html/index.html in your favorite browser.

48.5.3 Build documentation using Visual Studio (VS) Code

One can put together a productive environment to author documentation and also test it locally using VS Code with only a handful of extensions. Even though the extension landscape of VS Code is ever changing, here is one example setup that proved useful at the time of writing. In it, one can change/add content, build a new version of the docs using a single VS Code Task (or hotkey), see all errors/ warnings emitted by Sphinx in the Problems pane and immediately see the resulting website show up on a locally-served web server.

48.5.3.1 Configuring VS Code

1. Install the following extensions:
   - Python (ms-python.python)
   - Live Server (ritwickdey.LiveServer)
2. Add the following entries in .vscode/settings.json

   ```
   {
     "liveServer.settings.root": "/.vscode/build/html",
     "liveServer.settings.wait": 1000,
     "python.terminal.activateEnvInCurrentTerminal": true
   }
   ```

   The settings above are used for the following reasons:
   - liveServer.settings.root: Sets the root of the output website for live previews. Must be changed alongside the tasks.json command.
   - liveServer.settings.wait: Tells live server to wait with the update to give time for Sphinx to regenerate site contents and not refresh before all is done. (Empirical value)
   - python.terminal.activateEnvInCurrentTerminal: Automatic virtual environment activation is a nice touch, should you want to build the site from the integrated terminal.
3. Add the following tasks in .vscode/tasks.json

   ```
   {
     "version": "2.0.0",
     "tasks": [
       {
         "label": "Build Docs",
         "type": "process",
         "windows": {
           "command": "${workspaceFolder}/.venv/Scripts/python.exe"
         },
         "command": "${workspaceFolder}/.venv/bin/python3",
         "args": [
           "-m",
   ```

(continues on next page)

(continued from previous page)

```
        "sphinx",
        "-j",
        "auto",
        "-T",
        "-b",
        "html",
        "-d",
        "${workspaceFolder}/.vscode/build/doctrees",
        "-D",
        "language=en",
        "${workspaceFolder}/docs",
        "${workspaceFolder}/.vscode/build/html"
      ],
      "problemMatcher": [
        {
          "owner": "sphinx",
          "fileLocation": "absolute",
          "pattern": {
            "regexp": "^(?:.*\\.{3}\\s+)?(\\/[^:]*|[a-zA-Z]:\\\\[^:]*):(\\d+):\\s+(WARNING|ERROR):\\s+(.
↪*)$",
            "file": 1,
            "line": 2,
            "severity": 3,
            "message": 4
          },
        },
        {
          "owner": "sphinx",
          "fileLocation": "absolute",
          "pattern": {
            "regexp": "^(?:.*\\.{3}\\s+)?(\\/[^:]*|[a-zA-Z]:\\\\[^:]*):{1,2}\\s+(WARNING|ERROR):\\s+(.*)
↪$",
            "file": 1,
            "severity": 2,
            "message": 3
          }
        }
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      }
    },
  ],
}
```

(Implementation detail: two problem matchers were needed to be defined, because VS Code doesn't tolerate some problem information being potentially absent. While a single regex could match all types of errors, if a capture group remains empty (the line number doesn't show up in all warning/error messages) but the pattern references said empty capture group, VS Code discards the message completely.)

4. Configure Python virtual environment (venv)

   - From the Command Palette, run Python: Create Environment
     - Select venv environment and the docs/sphinx/requirements.txt file. (Simply pressing enter while hovering over the file from the drop down is insufficient, one has to select the radio

button with the 'Space' key if using the keyboard.)

5. Build the docs
   - Launch the default build Task using either:
     - a hotkey (default is Ctrl+Shift+B) or
     - by issuing the Tasks: Run Build Task from the Command Palette.
6. Open the live preview
   - Navigate to the output of the site within VS Code, right-click on .vscode/build/html/index.html and select Open with Live Server. The contents should update on every rebuild without having to refresh the browser.

48.6 How to provide feedback for ROCm documentation

There are four standard ways to provide feedback for this repository.

48.6.1 Pull Request

All contributions to ROCm documentation should arrive via the GitHub Flow targeting the develop branch of the repository. If you are unable to contribute via the GitHub Flow, feel free to email us.

48.6.2 GitHub Discussions

To ask questions or view answers to frequently asked questions, refer to GitHub Discussions. On GitHub Discussions, in addition to asking and answering questions, members can share updates, have open-ended conversations, and follow along on via public announcements.

48.6.3 GitHub Issue

Issues on existing or absent docs can be filed as GitHub Issues.

48.6.4 Email

Send other feedback or questions to rocm-feedback@amd.com

CHAPTER

FORTYNINE

LICENSE

Note: This license applies to the ROCm repository that contains documentation primarily. For other licensing information, see the Licensing Terms page.

INDEX

Symbols

C

O