ROCm Documentation

Release 5.1.0

Advanced Micro Devices, Inc.

May 02, 2024

CONTENTS

What is ROCm?

ROCm is an open-source stack for GPU computation. ROCm is primarily Open-Source Software (OSS) that allows developers the freedom to customize and tailor their GPU software for their own needs while collaborating with a community of other developers, and helping each other find solutions in an agile, flexible, rapid and secure manner. *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*

CHAPTER

ONE

WHAT IS ROCM?

ROCm is an open-source stack for GPU computation. ROCm is primarily Open-Source Software (OSS) that allows developers the freedom to customize and tailor their GPU software for their own needs while collaborating with a community of other developers, and helping each other find solutions in an agile, flexible, rapid and secure manner.

ROCm is a collection of drivers, development tools and APIs enabling GPU programming from the low-level kernel to end-user applications. 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 the necessary OSS compilers, debuggers and libraries. ROCm is fully integrated into ML frameworks such as PyTorch and TensorFlow. ROCm can be deployed in many ways, including through the use of containers such as Docker, Spack, and your own build from source.

ROCm's goal is to allow our users to maximize their GPU hardware investment. ROCm is designed to help develop, test and deploy GPU accelerated HPC, AI, scientific computing, CAD, and other applications in a free, open-source, integrated and secure software ecosystem.

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/
↪latest/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/
↪debian 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/
↪latest/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/
↪debian jammy main
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' | sudo tee /etc/
↪apt/preferences.d/rocm-pin-600
EOF
```

3. Update the list of packages

```
sudo apt update
```

Red Hat Enterprise Linux

1. Add the repositories

RHEL 8.6

```
# Add the amdgpu module repository for RHEL 8.6
sudo tee /etc/yum.repos.d/amdgpu.repo <<'EOF'
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/latest/rhel/8.6/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/latest/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

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/latest/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/latest/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/latest/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/latest/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 SP4

```
# 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/latest/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
```

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 and OS Support (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:

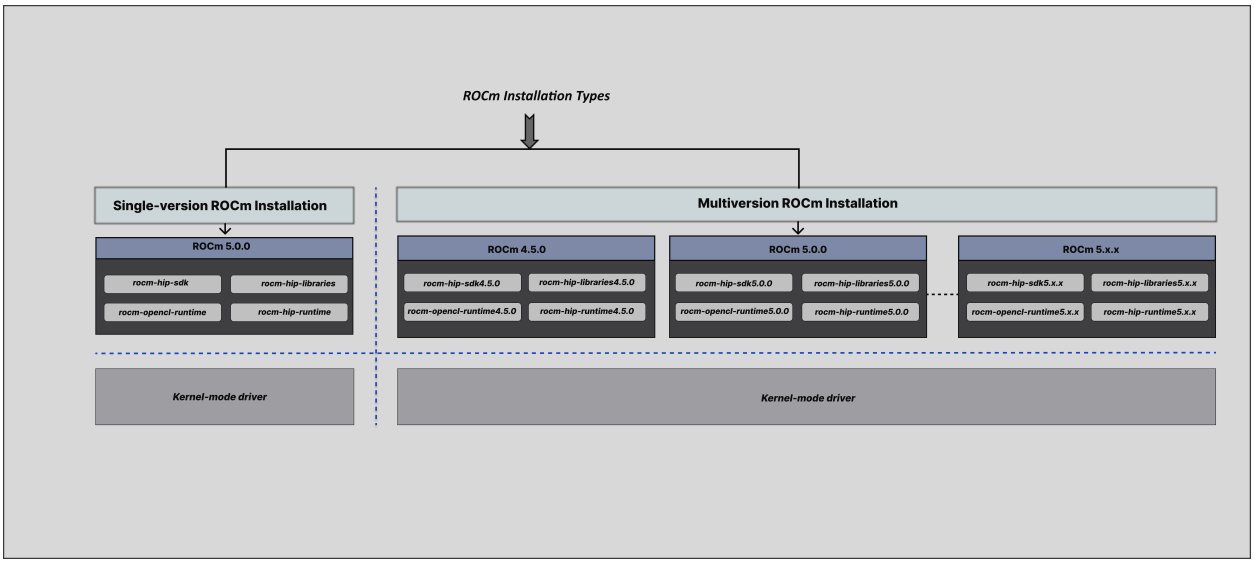


Fig. 3.1: ROCm Installation Types

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 Distributions*.

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

```
x86_64
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=20.04
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 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 15

Add the perl languages repository.

```
zypper addrepo https://download.opensuse.org/repositories/devel:languages:perl/SLE_15_
↪SP4/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 and OS Support (Linux)*

3.6.2 Installation (Linux)

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 18.04

```
# amdgpu repository for bionic
echo 'deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/
↪amdgpu/22.20.1/ubuntu bionic main' \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

Ubuntu 20.04

```
# amdgpu repository for focal
echo 'deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/
↪amdgpu/22.20.1/ubuntu focal main' \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

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 18.04

```
# ROCm repositories for bionic
for ver in 5.0.2 5.2.1; do
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/
↪apt/$ver bionic 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 20.04

```
# ROCm repositories for focal
for ver in 5.0.2 5.2.1; 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
```

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-sdk5.0.2 rocm-hip-sdk5.2.1
```

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 7.9

```
sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/rhel/7.9/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8.4

```
sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/rhel/8.4/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8.5

```
sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/rhel/8.5/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 5.0.2 5.2.1; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/yum/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

RHEL 8

```
for ver in 5.0.2 5.2.1; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/yum/$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-sdk5.0.2 rocm-hip-sdk5.2.1
```

SUSE Linux Enterprise Server 15

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.

```
sudo tee /etc/zypp/repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/sle/15/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 5.0.2 5.2.1; 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-sdk5.0.2 rocm-hip-sdk5.2.1
```

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-5.2.1/bin:/opt/rocm-5.2.1/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 18.04

```
# amdgpu repository for bionic
echo 'deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/
↪amdgpu/22.20.1/ubuntu bionic main' \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

Ubuntu 20.04

```
# amdgpu repository for focal
echo 'deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/
↪amdgpu/22.20.1/ubuntu focal main' \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list
sudo apt update
```

Red Hat Enterprise Linux

RHEL 7.9

```
sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/rhel/7.9/main/x86_64/
enabled=1
```

(continues on next page)

(continued from previous page)

```
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8.4

```
sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/rhel/8.4/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8.5

```
sudo tee /etc/yum.repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/rhel/8.5/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 15

```
sudo tee /etc/zypp/repos.d/amdgpu.repo <<EOF
[amdgpu]
name=amdgpu
baseurl=https://repo.radeon.com/amdgpu/22.20.1/sle/15/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 15

```
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 18.04

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/
↪apt/5.2.1 bionic 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 20.04

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/
↪apt/5.2.1 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
```

Red Hat Enterprise Linux

RHEL 7

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-5.2.1]
name=ROCm5.2.1
baseurl=https://repo.radeon.com/rocm/yum/5.2.1/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo yum clean all
```

RHEL 8

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-5.2.1]
name=ROCm5.2.1
baseurl=https://repo.radeon.com/rocm/yum/5.2.1/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 15

```
sudo tee /etc/zypp/repos.d/rocm.repo <<EOF
[ROCm-5.2.1]
name=ROCm5.2.1
name=rocm
baseurl=https://repo.radeon.com/rocm/zyp/5.2.1/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 15

```
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 15

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.

Example: Running HIP applications

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: ROCm Release Package Naming

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.

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.

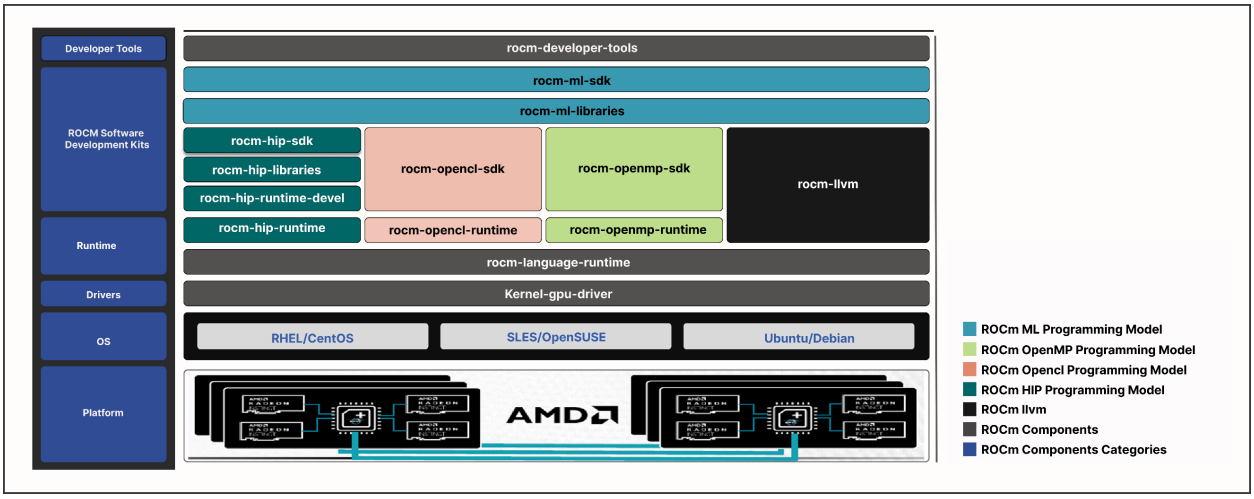


Fig. 3.3: ROCm Meta Packages

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

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.

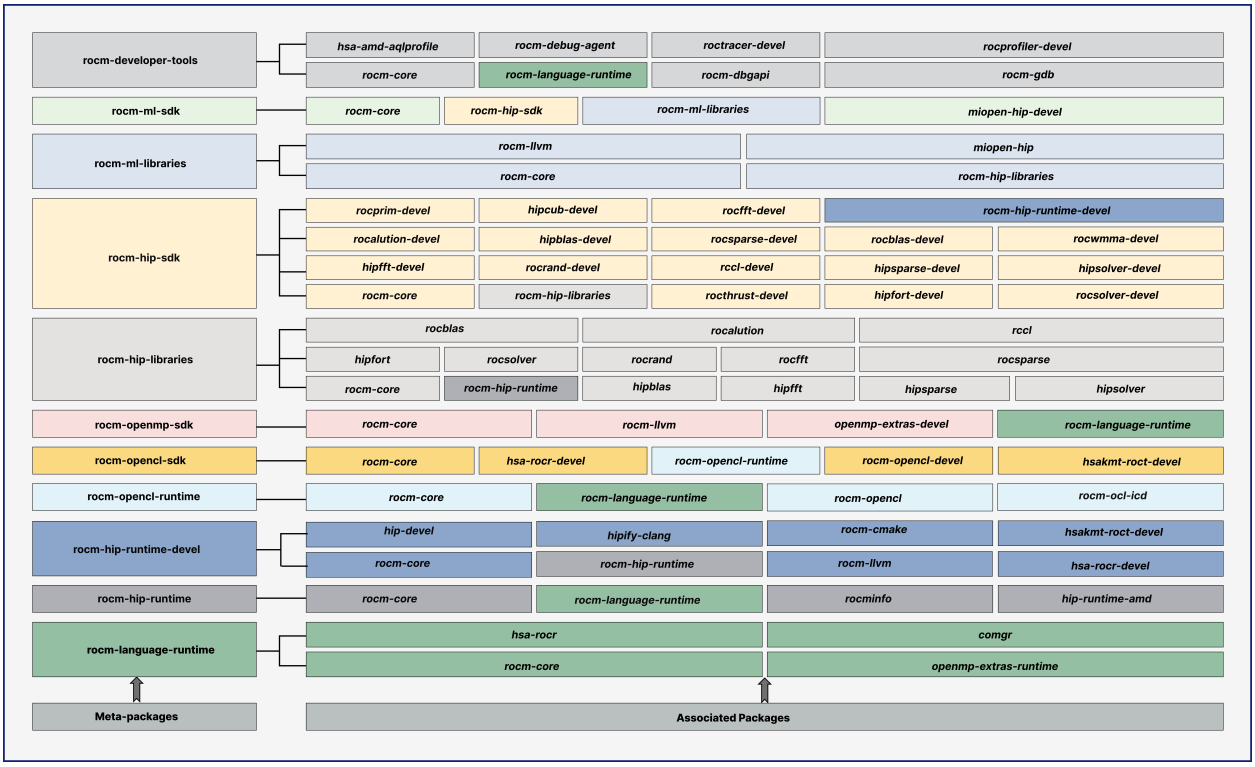


Fig. 3.4: Associated Packages

- 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.

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 and OS Support (Linux)*

3.7.2 Installation with install script

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

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 18.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/22.20.1/ubuntu/bionic/amdgpu-install_22.20.1.
↪50201-1_all.deb
sudo apt install ./amdgpu-install_22.20.1.50201-1_all.deb
```

Ubuntu 20.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/22.20.1/ubuntu/focal/amdgpu-install_22.20.1.
↪50201-1_all.deb
sudo apt install ./amdgpu-install_22.20.1.50201-1_all.deb
```

Red Hat Enterprise Linux

RHEL 7.9

```
sudo yum install https://repo.radeon.com/amdgpu-install/22.20.1/rhel/7.9/amdgpu-install-
↪22.20.1.50201-1.el7.noarch.rpm
```

RHEL 8.4

```
sudo yum install https://repo.radeon.com/amdgpu-install/22.20.1/rhel/8.4/amdgpu-install-
↪22.20.1.50201-1.el8.noarch.rpm
```

RHEL 8.5

```
sudo yum install https://repo.radeon.com/amdgpu-install/22.20.1/rhel/8.5/amdgpu-install-
↪22.20.1.50201-1.el8.noarch.rpm
```

SUSE Linux Enterprise Server 15

Service Pack 4

```
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/22.20.1/sle/
↪15/amdgpu-install-22.20.1.50201-1.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
```

(continues on next page)

(continued from previous page)

```
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
```

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 5.0.2 and 5.2.1 simultaneously, you are required to download the installer from the latest ROCm release v5.2.1.

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 18.04

```
for ver in 5.0.2 5.2.1; do
echo "deb [arch=amd64 signed-by=/etc/apt/trusted.gpg.d/rocm-keyring.gpg] https://repo.
↪radeon.com/rocm/apt/$ver bionic 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 20.04

```
for ver in 5.0.2 5.2.1; 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
```

Red Hat Enterprise Linux

RHEL 7

```
for ver in 5.0.2 5.2.1; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/yum/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

RHEL 8

```
for ver in 5.0.2 5.2.1; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/yum/$ver/main
enabled=1
priority=50
```

(continues on next page)

(continued from previous page)

```
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo yum clean all
```

SUSE Linux Enterprise Server 15

```
for ver in 5.0.2 5.2.1; 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
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 v5.2.1, will be installed as its latest release in the list.

```
sudo amdgpu-install --usecase=rocm --rocmrelease=5.0.2
sudo amdgpu-install --usecase=rocm --rocmrelease=5.2.1
```

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

DEPLOY ROCM DOCKER CONTAINERS

4.1 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.

4.2 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.

4.2.1 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>
```

4.2.2 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 ...
```

4.3 Docker images in the ROCm ecosystem

4.3.1 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.

4.3.2 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

FIVE

RELEASE NOTES

The release notes for the ROCm platform.

5.1 ROCm 5.2.1

5.1.1 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

CHAPTER

SIX

RELEASE NOTES

The release notes for the ROCm platform.

6.1 ROCm 5.2.1

6.1.1 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

6.2 ROCm 5.2.0

6.2.1 What's New in This Release

6.2.1.1 HIP Enhancements

The ROCm v5.2 release consists of the following HIP enhancements:

6.2.1.1.1 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.

6.2.1.1.2 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

6.2.1.1.3 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:

6.2.1.1.3.1 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);
```

6.2.1.1.3.2 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);
```

6.2.1.1.3.3 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);
```

6.2.1.1.3.4 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 https://docs.amd.com/bundle/HIP_API_Guide/page/modules.html

6.2.1.1.4 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 https://docs.amd.com/category/libraries.

6.2.1.2 OpenMP Enhancements in This Release

6.2.1.2.1 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.

6.2.2 Deprecations and Warnings

6.2.2.1 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.

6.2.2.1.1 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.

6.2.2.1.2 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.

6.2.2.1.3 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

6.2.2.1.4 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
```

6.2.2.1.5 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
```

6.2.2.2 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.

6.2.3 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

6.2.4 Known Issues

This section consists of known issues in this release.

6.2.4.1 Compiler Error on gfx1030 When Compiling at -O0

6.2.4.1.1 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.

6.2.4.1.2 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.

6.2.4.2 System Freeze Observed During CUDA Memtest Checkpoint

6.2.4.2.1 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.

6.2.4.2.2 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.

6.2.4.3 HPC test fails with the "HSA_STATUS_ERROR_MEMORY_FAULT" error

6.2.4.3.1 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.

6.2.4.3.2 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);
```

6.2.4.4 Kernel produces incorrect result

6.2.4.4.1 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.

6.2.4.4.2 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.

6.2.4.5 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.

6.2.5 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

6.2.5.1 hipBLAS 0.51.0

hipBLAS 0.51.0 for ROCm 5.2.0

6.2.5.1.1 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

6.2.5.1.2 Fixed

- Fixed version gathering in performance measuring script

6.2.5.2 hipCUB 2.11.1

hipCUB 2.11.1 for ROCm 5.2.0

6.2.5.2.1 Added

- Packages for tests and benchmark executable on all supported OSes using CPack.

6.2.5.3 hipFFT 1.0.8

hipFFT 1.0.8 for ROCm 5.2.0

6.2.5.3.1 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.

6.2.5.4 hipSOLVER 1.4.0

hipSOLVER 1.4.0 for ROCm 5.2.0

6.2.5.4.1 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.

6.2.5.4.2 Fixed

- Fixed the ReadTheDocs documentation generation.

6.2.5.5 hipSPARSE 2.2.0

hipSPARSE 2.2.0 for ROCm 5.2.0

6.2.5.5.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.

6.2.5.6 rocALUTION 2.0.3

rocALUTION 2.0.3 for ROCm 5.2.0

6.2.5.6.1 Added

- Packages for test and benchmark executables on all supported OSes using CPack.

6.2.5.7 rocBLAS 2.44.0

rocBLAS 2.44.0 for ROCm 5.2.0

6.2.5.7.1 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.

6.2.5.7.2 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.

6.2.5.7.3 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.

6.2.5.7.4 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.

6.2.5.7.5 Removed

- Remove Navi12 (gfx1011) from fat binary.

6.2.5.8 rocFFT 1.0.17

rocFFT 1.0.17 for ROCm 5.2.0

6.2.5.8.1 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.

6.2.5.8.2 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.

6.2.5.8.3 Optimizations

- Introduced a new access pattern of lds (non-linear) and applied it on sbcc kernels len 64 to get performance improvement.

6.2.5.8.4 Fixed

- Fixed plan creation failure in cases where SBCC kernels would need to write to non-unit-stride buffers.

6.2.5.9 rocPRIM 2.10.14

rocPRIM 2.10.14 for ROCm 5.2.0

6.2.5.9.1 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.

6.2.5.10 rocRAND 2.10.14

rocRAND 2.10.14 for ROCm 5.2.0

6.2.5.10.1 Added

- Backward compatibility for deprecated `#include <rocrand.h>` using wrapper header files.
- Packages for test and benchmark executables on all supported OSes using CPack.

6.2.5.11 rocSOLVER 3.18.0

rocSOLVER 3.18.0 for ROCm 5.2.0

6.2.5.11.1 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.

6.2.5.11.2 Fixed

- Fixed compatibility with libfmt 8.1

6.2.5.12 rocSPARSE 2.2.0

rocSPARSE 2.2.0 for ROCm 5.2.0

6.2.5.12.1 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.

6.2.5.12.2 Improved

- Clients code size reduction.
- Clients error handling.
- Clients benchmarking for performance tracking.

6.2.5.12.3 Changed

- Test adjustments due to roundoff errors.
- Fixing API calls compatiblity with rocPRIM.

6.2.5.12.4 Known Issues

- none

6.2.5.13 rocThrust 2.15.0

rocThrust 2.15.0 for ROCm 5.2.0

6.2.5.13.1 Added

- Packages for tests and benchmark executable on all supported OSes using CPack.

6.2.5.14 rocWMMA 0.7

rocWMMA 0.7 for ROCm 5.2.0

6.2.5.14.1 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

6.2.5.14.2 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)

6.2.5.15 Tensile 4.33.0

Tensile 4.33.0 for ROCm 5.2.0

6.2.5.15.1 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

6.2.5.15.2 Optimizations

- Optimize MessagePackLoadLibraryFile by switching to fread
- DGEMM tail loop optimization for PrefetchAcrossPersistentMode=1/DirectToVgpr

6.2.5.15.3 Changed

- Alpha/beta datatype remains as F32 for HPA HGEMM
- Force assembly kernels to not flush denorms
- Use hipDeviceAttributePhysicalMultiProcessorCount as multiProcessorCount

6.2.5.15.4 Fixed

- Fix segmentation fault when run i8 datatype with TENSILE_DB=0x80

6.3 ROCm 5.1.3

6.3.1 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

6.4 ROCm 5.1.1

6.4.1 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

6.5 ROCm 5.1.0

6.5.1 What's New in This Release

6.5.1.1 HIP Enhancements

The ROCm v5.1 release consists of the following HIP enhancements.

6.5.1.1.1 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.

6.5.1.1.2 Support for HIP Graph

ROCm v5.1 extends support for HIP Graph.

6.5.1.1.3 Planned Changes for HIP in Future Releases

6.5.1.1.3.1 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.

6.5.1.1.3.2 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.

6.5.1.2 ROCDebugger Enhancements

6.5.1.2.1 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.

6.5.1.2.2 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 https://docs.amd.com.

6.5.1.2.3 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.

6.5.1.3 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 https://docs.amd.com/bundle/MIOpen_gh-pages/page/releasenotes.html

6.5.1.4 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

6.5.2 Fixed Defects

The following defects are fixed in this release.

6.5.2.1 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.

6.5.2.2 ROCDebugger Fixed Defects

6.5.2.2.1 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.

6.5.2.2.2 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.

6.5.2.2.3 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.

6.5.2.2.4 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.

6.5.3 Known Issues

6.5.3.1 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.

6.5.3.2 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.

6.5.3.3 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.

6.5.3.4 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.

6.5.3.5 Radeon Pro V620 and W6800 Workstation GPUs

6.5.3.5.1 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.

6.5.3.6 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.

6.5.4 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

6.5.4.1 hipBLAS 0.50.0

hipBLAS 0.50.0 for ROCm 5.1.0

6.5.4.1.1 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.

6.5.4.1.2 Fixed

- Added hipblasVersionMinor define. hipblaseVersionMinor remains defined for backwards compatibility.
- Doxygen warnings in hipblas.h header file.

6.5.4.1.3 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

6.5.4.2 hipCUB 2.11.0

hipCUB 2.11.0 for ROCm 5.1.0

6.5.4.2.1 Added

- Device segmented sort
- Warp merge sort, WarpMask and thread sort from cub 1.15.0 supported in hipCUB
- Device three way partition

6.5.4.2.2 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

6.5.4.3 hipFFT 1.0.7

hipFFT 1.0.7 for ROCm 5.1.0

6.5.4.3.1 Changed

- Use fft_params struct for accuracy and benchmark clients.

6.5.4.4 hipSOLVER 1.3.0

hipSOLVER 1.3.0 for ROCm 5.1.0

6.5.4.4.1 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

6.5.4.4.2 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.

6.5.4.4.3 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.

6.5.4.5 hipSPARSE 2.1.0

hipSPARSE 2.1.0 for ROCm 5.1.0

6.5.4.5.1 Added

- Added gtsv_interleaved_batch and gpsv_interleaved_batch routines
- Add SpGEMM_reuse

6.5.4.5.2 Changed

- Changed BUILD_CUDA with USE_CUDA in install script and cmake files
- Update googletest to 11.1

6.5.4.5.3 Improved

- Fixed a bug in SpMM Alg versioning

6.5.4.5.4 Known Issues

- none

6.5.4.6 rccl 2.11.4

RCCL 2.11.4 for ROCm 5.1.0

6.5.4.6.1 Added

- Compatibility with NCCL 2.11.4

6.5.4.6.2 Known Issues

- Managed memory is not currently supported for clique-based kernels

6.5.4.7 rocALUTION 2.0.2

rocALUTION 2.0.2 for ROCm 5.1.0

6.5.4.7.1 Added

- Added out-of-place matrix transpose functionality
- Added LocalVector<bool>

6.5.4.8 rocBLAS 2.43.0

rocBLAS 2.43.0 for ROCm 5.1.0

6.5.4.8.1 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

6.5.4.8.2 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

6.5.4.8.3 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

6.5.4.8.4 Fixed

- For function trmv (non-transposed cases) avoid overflow in offset calculation
- Fixed cppcheck errors/warnings
- Fixed doxygen warnings

6.5.4.9 rocFFT 1.0.16

rocFFT 1.0.16 for ROCm 5.1.0

6.5.4.9.1 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.

6.5.4.9.2 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.

6.5.4.9.3 Fixed

- Fixed correctness of some R2C transforms with unusual strides.

6.5.4.9.4 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.

6.5.4.10 rocPRIM 2.10.13

rocPRIM 2.10.13 for ROCm 5.1.0

6.5.4.10.1 Fixed

- Fixed radix sort int64_t bug introduced in [2.10.11]

6.5.4.10.2 Added

- Future value
- Added device partition_three_way to partition input to three output iterators based on two predicates

6.5.4.10.3 Changed

- The reduce/scan algorithm precision issues in the tests has been resolved for half types.

6.5.4.10.4 Known Issues

- device_segmented_radix_sort unit test failing for HIP on Windows

6.5.4.11 rocRAND 2.10.13

rocRAND 2.10.13 for ROCm 5.1.0

6.5.4.11.1 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)`

6.5.4.11.2 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 &lt;rocrand/rocrand.h&gt;`, rather than `#include &lt;rocrand/rocrand.h&gt;`
- 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.

6.5.4.11.3 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

6.5.4.11.4 Known Issues

- kernel_xorwow unit test is failing for certain GPU architectures.

6.5.4.12 rocSOLVER 3.17.0

rocSOLVER 3.17.0 for ROCm 5.1.0

6.5.4.12.1 Optimized

- Optimized non-pivoting and batch cases of the LU factorization

6.5.4.12.2 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

6.5.4.13 rocSPARSE 2.1.0

rocSPARSE 2.1.0 for ROCm 5.1.0

6.5.4.13.1 Added

- gtsv_interleaved_batch
- gpsv_interleaved_batch
- SpGEMM_reuse
- Allow copying of mat info struct

6.5.4.13.2 Improved

- Optimization for SDDMM
- Allow unsorted matrices in csrgemm multipass algorithm

6.5.4.13.3 Known Issues

- none

6.5.4.14 rocThrust 2.14.0

rocThrust 2.14.0 for ROCm 5.1.0

6.5.4.14.1 Added

- Updated to match upstream Thrust 1.15.0

6.5.4.14.2 Known Issues

- async_copy, partition, and stable_sort_by_key unit tests are failing on HIP on Windows.

6.5.4.15 Tensile 4.32.0

Tensile 4.32.0 for ROCm 5.1.0

6.5.4.15.1 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

6.5.4.15.2 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

6.5.4.15.3 Changed

- Update Googletest to 1.11.0

6.5.4.15.4 Removed

- Remove no longer supported benchmarking steps

6.6 ROCm 5.0.2

6.6.1 Fixed Defects

The following defects are fixed in the ROCm v5.0.2 release.

6.6.1.1 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.

6.6.2 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

6.7 ROCm 5.0.1

6.7.1 Deprecations and Warnings

6.7.1.1 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.

6.7.2 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

6.8 ROCm 5.0.0

6.8.1 What's New in This Release

6.8.1.1 HIP Enhancements

The ROCm v5.0 release consists of the following HIP enhancements.

6.8.1.1.1 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.

6.8.1.1.2 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

6.8.1.2 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.

6.8.2 Breaking Changes

6.8.2.1 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
```

(continues on next page)

(continued from previous page)

```
-                                                       ///< 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,

-    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.
```

(continues on next page)

(continued from previous page)

```
+    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.
+    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.
```

(continues on next page)

(continued from previous page)

```
+    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.
```

(continues on next page)

(continued from previous page)

```
+    hipDeviceAttributePciDeviceId,                          ///< PCI Device ID.
+    hipDeviceAttributePciDomainID,                          ///< PCI Domain ID.
+    hipDeviceAttributePersistingL2CacheMaxSize,             ///< Cuda11 only. Maximum l2␣
↪persisting lines 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,
```

(continues on next page)

(continued from previous page)

```
-    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
-                                                /// 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␣
```

(continues on next page)

(continued from previous page)

```
↪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 {
```

6.8.3 Known Issues

6.8.3.1 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.

6.8.3.2 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

6.8.3.2.1 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.

6.8.3.2.2 Current behavior

BeginNS is lower than DispatchNS, which is incorrect.

6.8.3.2.3 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.

6.8.3.2.4 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.

6.8.3.3 Radeon Pro V620 and W6800 Workstation GPUs

6.8.3.3.1 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.

6.8.4 Deprecations and Warnings

6.8.4.1 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);
```

6.8.4.2 HIP API Deprecations and Warnings

6.8.4.2.1 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.

6.8.4.3 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.

6.8.4.4 Warning - MIOpenTensile Deprecation

MIOpenTensile will be deprecated in a future release.

6.8.5 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

6.8.5.1 hipBLAS 0.49.0

hipBLAS 0.49.0 for ROCm 5.0.0

6.8.5.1.1 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

6.8.5.1.2 Fixed

- Fixed use of incorrect 'HIP_PATH' when building from source.
- Fixed windows packaging
- Allowing negative increments in hipblas-bench
- Removed boost dependency

6.8.5.2 hipCUB 2.10.13

hipCUB 2.10.13 for ROCm 5.0.0

6.8.5.2.1 Fixed

- Added missing includes to hipcub.hpp

6.8.5.2.2 Added

- Bfloat16 support to test cases (device_reduce & device_radix_sort)
- Device merge sort
- Block merge sort
- API update to CUB 1.14.0

6.8.5.2.3 Changed

- The SetupNVCC.cmake automatic target selector select all of the capabalities of all available card for NVIDIA backend.

6.8.5.3 hipFFT 1.0.4

hipFFT 1.0.4 for ROCm 5.0.0

6.8.5.3.1 Fixed

- Add calls to rocFFT setup/cleanup.
- Cmake fixes for clients and backend support.

6.8.5.3.2 Added

- Added support for Windows 10 as a build target.

6.8.5.4 hipSOLVER 1.2.0

hipSOLVER 1.2.0 for ROCm 5.0.0

6.8.5.4.1 Added

- Added functions
  - sytrf
    - hipsolverSsytrf_bufferSize, hipsolverDsytrf_bufferSize, hipsolverCsytrf_bufferSize, hipsolverZsytrf_bufferSize
    - hipsolverSsytrf, hipsolverDsytrf, hipsolverCsytrf, hipsolverZsytrf

6.8.5.4.2 Fixed

- Fixed use of incorrect `HIP_PATH` when building from source (#40). Thanks @jakub329homola!

6.8.5.5 hipSPARSE 2.0.0

hipSPARSE 2.0.0 for ROCm 5.0.0

6.8.5.5.1 Added

- Added (conjugate) transpose support for csrmv, hybmv and spmv routines

6.8.5.6 rccl 2.10.3

RCCL 2.10.3 for ROCm 5.0.0

6.8.5.6.1 Added

- Compatibility with NCCL 2.10.3

6.8.5.6.2 Known Issues

- Managed memory is not currently supported for clique-based kernels

6.8.5.7 rocALUTION 2.0.1

rocALUTION 2.0.1 for ROCm 5.0.0

6.8.5.7.1 Changed

- Removed deprecated GlobalPairwiseAMG class, please use PairwiseAMG instead.
- Changed to C++ 14 Standard

6.8.5.7.2 Improved

- Added sanitizer option
- Improved documentation

6.8.5.8 rocBLAS 2.42.0

rocBLAS 2.42.0 for ROCm 5.0.0

6.8.5.8.1 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

6.8.5.8.2 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.

6.8.5.8.3 Changed

- Instantiate templated rocBLAS functions to reduce size of librocblas.so
- Removed static library dependency on msgpack
- Removed boost dependencies for clients

6.8.5.8.4 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

6.8.5.9 rocFFT 1.0.13

rocFFT 1.0.13 for ROCm 5.0.0

6.8.5.9.1 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.

6.8.5.9.2 Added

- Added new kernel generator for select fused-2D transforms.

6.8.5.9.3 Fixed

- Improved large 1D transform decompositions.

6.8.5.10 rocPRIM 2.10.12

rocPRIM 2.10.12 for ROCm 5.0.0

6.8.5.10.1 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

6.8.5.10.2 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

6.8.5.10.3 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

6.8.5.10.4 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

6.8.5.11 rocRAND 2.10.12

rocRAND 2.10.12 for ROCm 5.0.0

6.8.5.11.1 Changed

- No updates or changes for ROCm 5.0.0.

6.8.5.12 rocSOLVER 3.16.0

rocSOLVER 3.16.0 for ROCm 5.0.0

6.8.5.12.1 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.

6.8.5.12.2 Optimized

- Improved general performance of LU factorization
- Increased parallelism of specialized kernels when compiling from source, reducing build times on multi-core systems.

6.8.5.12.3 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

6.8.5.12.4 Fixed

- Added missing stdint.h include to rocsolver.h

6.8.5.13 rocSPARSE 2.0.0

rocSPARSE 2.0.0 for ROCm 5.0.0

6.8.5.13.1 Added

- csrmv, coomv, ellmv, hybmv for (conjugate) transposed matrices
- csrmv for symmetric matrices

6.8.5.13.2 Changed

- spmm_ex is now deprecated and will be removed in the next major release

6.8.5.13.3 Improved

- Optimization for gtsv

6.8.5.14 rocThrust 2.13.0

rocThrust 2.13.0 for ROCm 5.0.0

6.8.5.14.1 Added

- Updated to match upstream Thrust 1.13.0
- Updated to match upstream Thrust 1.14.0
- Added async scan

6.8.5.14.2 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)

6.8.5.15 Tensile 4.31.0

Tensile 4.31.0 for ROCm 5.0.0

6.8.5.15.1 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

6.8.5.15.2 Optimized

- Add DGEMM NN custom kernel for HPL on aldebaran

6.8.5.15.3 Changed

- Update tensile_client executable to std=c++14

6.8.5.15.4 Removed

- Remove unused old Tensile client code

6.8.5.15.5 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

SEVEN

GPU AND OS SUPPORT (LINUX)

7.1 Supported Distributions

AMD ROCm™ Platform supports the following Linux distributions.

Distribution	Processor Architectures	Validated Kernel
RHEL 9.1	x86-64	5.14
RHEL 8.6 to 8.7	x86-64	4.18
SLES 15 SP4	x86-64	
Ubuntu 20.04.5 LTS	x86-64	5.15
Ubuntu 22.04.1 LTS	x86-64	5.15, OEM 5.17

7.2 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`)

7.3 GPU Support Table

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	

7.3.1 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.

7.4 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

EIGHT

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*

8.1 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

8.2 Docker Image Support Matrix

The software support matrices for ROCm container releases is listed.

8.2.1 ROCm 5.6

8.2.1.1 PyTorch

8.2.1.1.1 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

8.2.1.1.2 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

8.2.1.1.3 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

8.2.1.1.4 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

8.2.1.2 TensorFlow

8.2.1.2.1 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

8.2.1.2.2 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

8.2.1.2.3 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

8.3 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.

8.3.1 Deep Learning

ROCm releases support the most recent and two prior releases of PyTorch and TensorFlow

ROCm	PyTorch	TensorFlow	MAGMA
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	2.5.4

8.3.2 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

8.3.3 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

For the latest documentation of these libraries, refer to the *associated documentation*.

CHAPTER

NINE

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. The table is ordered to follow ROCm's manifest file.

Component	License
ROCK-Kernel-Driver	GPL 2.0 WITH Linux-syscall-note
ROCT-Thunk-Interface	MIT
ROCR-Runtime	The University of Illinois/NCSA
rocm_smi_lib	The University of Illinois/NCSA
rocm-cmake	MIT
rocminfo	The University of Illinois/NCSA
rocprofiler	MIT
roctracer	MIT
ROCm-OpenCL-Runtime	MIT
ROCm-OpenCL-Runtime/api/opencl/khronos/icd	Apache 2.0
clang-ocl	MIT
HIP	MIT
hipamd	MIT
ROCclr	MIT
HIPIFY	MIT
HIPCC	MIT
llvm-project	Apache
rocm-llvm-alt	AMD Proprietary License
ROCm-Device-Libs	The University of Illinois/NCSA
atmi	MIT
ROCm-CompilerSupport	The University of Illinois/NCSA
rocr_debug_agent	The University of Illinois/NCSA
rocm_bandwidth_test	The University of Illinois/NCSA
half	MIT
RCP	MIT
ROCgdb	GNU General Public License v2.0
ROCdbgapi	MIT
rdc	MIT
rocBLAS	MIT
Tensile	MIT
hipBLAS	MIT
rocFFT	MIT

continues on next page

Table 9.1 – continued from previous page

Component	License
hipFFT	MIT
rocRAND	MIT
rocSPARSE	MIT
rocSOLVER	BSD-2-Clause
hipSOLVER	MIT
hipSPARSE	MIT
rocALUTION	MIT
MIOpenGEMM	MIT
MIOpen	MIT
rccl	Custom
MIVisionX	MIT
rocThrust	Apache 2.0
hipCUB	Custom
rocPRIM	MIT
rocWMMA	MIT
hipfort	MIT
AMDMIGraphX	MIT
ROCmValidationSuite	MIT
aomp	Apache 2.0
aomp-extras	MIT
flang	Apache 2.0

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.

9.1 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-5.2.3/share/doc/amd_comgr/LICENSE.txt`

CHAPTER

TEN

ALL REFERENCE MATERIAL

10.1 ROCm Software Groups

HIP HIP is both AMD's GPU programming language extension and the GPU runtime.

- hip:.doxygen/docBin/html/index
- Examples

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

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*
- ROCgdb
- ROCProfiler

- ROCTracer

Management Tools

- AMD SMI
- ROCm SMI
- ROCm Datacenter Tool

Validation Tools

- ROCm Validation Suite
- TransferBench

GPU Architectures

- *AMD Instinct MI200*
- *AMD Instinct MI100*

CHAPTER

ELEVEN

COMPILERS AND TOOLS

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.

ROCgdb This is ROCgdb, the ROCm source-level debugger for Linux, based on GDB, the GNU source-level debugger.

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.

ROCTracer Callback/Activity Library for Performance tracing AMD GPU's

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.

11.1 See Also

- *Compiler Disambiguation*

11.2 Compiler Reference Guide

11.2.1 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 11.1: 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

11.2.1.1 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 11.2: 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

11.2.2 Compiler Options and Features

This chapter discusses compiler options and features.

11.2.2.1 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.

11.2.2.2 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.

11.2.2.2.1 `-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.

11.2.2.2.2 `-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 11.3: -fstruct-layout Values and Their Effects

-fstruct-layout value	Structure peeling	Pointer size after selective compression of self-referential pointers in structures, wherever safe	Type of structure fields eligible for compression	Whether compression performed under safety check
1	Enabled	NA	NA	NA
2	Enabled	32-bit	NA	NA
3	Enabled	16-bit	NA	NA
4	Enabled	32-bit	Integer	Yes
5	Enabled	16-bit	Integer	Yes
6	Enabled	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	Enabled	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.

11.2.2.2.3 -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`

11.2.2.2.4 -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`.

11.2.2.2.5 -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`.

11.2.2.2.6 -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`.

11.2.2.2.7 -fnt-store (non-temporal store)

Generates a non-temporal store instruction for array accesses in a loop with a large trip count.

11.2.2.2.8 -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.

11.2.2.2.9 Optimizations Through Driver -mllvm <options>

The following optimization options must be invoked through driver `-mllvm <options>`:

11.2.2.2.9.1 -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.

11.2.2.2.9.2 -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.

11.2.2.2.9.3 `-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.

11.2.2.2.9.4 `-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.

11.2.2.2.9.5 `-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`.

11.2.2.2.9.6 `-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.

11.2.2.2.9.7 `-function-specialize`

Optimizes the functions with compile time constant formal arguments.

11.2.2.2.9.8 -lv-function-specialization

Generates specialized function versions when the loops inside function are vectorizable and the arguments are not aliased with each other.

11.2.2.2.9.9 -enable-vectorize-compares

Enables vectorization on certain loops with conditional breaks assuming the memory accesses are safely bound within the page boundary.

11.2.2.2.9.10 -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 11.4: -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]`.

11.2.2.2.9.11 -reduce-array-computations=[1,2,3]

Performs array data flow analysis and optimizes the unused array computations.

Table 11.5: -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]`.

11.2.2.2.9.12 -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.

11.2.2.2.9.13 -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`.

11.2.2.2.9.14 -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.

11.2.2.2.9.15 -suppress-fmas

Identifies the reduction patterns on FMA and suppresses the FMA generation, as it is not profitable on the reduction patterns.

11.2.2.2.9.16 -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.

11.2.2.2.9.17 -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`.

11.2.2.2.9.18 -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`.

11.2.2.2.9.19 -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
```

11.2.2.2.9.20 -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.

11.2.2.2.9.21 -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.

11.2.2.2.9.22 -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.

11.2.2.2.9.23 -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.

11.2.2.3 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

11.2.2.4 Miscellaneous OpenMP Compiler Features

This section discusses features that have been added or enhanced in the OpenMP compiler.

11.2.2.4.1 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.

11.2.2.4.2 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.

11.2.2.4.3 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.

11.2.2.4.4 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.

11.2.2.4.5 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 *Unified Shared Memory* section of the OpenMP Guide.

11.2.2.5 Support Status of Other Clang Options

The following table lists the other Clang options and their support status.

Option	Support Status	Descript
-###	Supported	Prints (bu
--analyzer-output <value>	Supported	"Static an
--analyze	Supported	Runs the s
-arcmt-migrate-emit-errors	Unsupported	Emits AR
-arcmt-migrate-report-output <value>	Unsupported	Output pa
-byteswapio	Supported	Swaps by
-B <dir>	Supported	Adds <di
-CC	Supported	Includes c
-cl-denorms-are-zero	Supported	OpenCL c
-cl-fast-relaxed-math	Supported	OpenCL c
-cl-finite-math-only	Supported	OpenCL c
-cl-fp32-correctly-rounded-divide-sqrt	Supported	OpenCL c
-cl-kernel-arg-info	Supported	OpenCL c
-cl-mad-enable	Supported	OpenCL c
-cl-no-signed-zeros	Supported	OpenCL c
-cl-opt-disable	Supported	OpenCL c
-cl-single-precision-constant	Supported	OpenCL c
-cl-std= <value>	Supported	OpenCL l
-cl-strict-aliasing	Supported	OpenCL c
-cl-uniform-work-group-size	Supported	OpenCL c
-cl-unsafe-math-optimizations	Supported	OpenCL c
--config <value>	Supported	Specifies
--cuda-compile-host-device	Supported	Compiles
--cuda-device-only	Supported	Compiles
--cuda-host-only	Supported	Compiles
--cuda-include-ptx=<value>	Unsupported	Includes P
--cuda-noopt-device-debug	Unsupported	Enables d
--cuda-path-ignore-env	Unsupported	Ignores en
--cuda-path=<value>	Unsupported	CUDA ins
-cxx-isystem <directory>	Supported	Adds a di
-C	Supported	Includes c
-c	Supported	Runs only
-dD	Supported	Prints ma
-dependency-dot <value>	Supported	Writes DO
-dependency-file <value>	Supported	Writes de
-dI	Supported	Prints inc
-dM	Supported	Prints ma

Option	Support Status	Descript
-dsym-dir <dir>	Unsupported	Outputs d
-D <macro>	Supported	=<value>
-emit-ast	Supported	Emits Cla
-emit-interface-stubs	Supported	Generates
-emit-llvm	Supported	Uses the I
-emit-merged-ifs	Supported	Generates
--emit-static-lib	Supported	Enables li
-enable-trivial-auto-var-init-zero-knowing-it-will-be-removed-from-clang	Supported	Declares e
-E	Supported	Runs the p
-fAAPCSBitfieldLoad	Unsupported	Follows th
-faddrsig	Supported	Emits an a
-faligned-allocation	Supported	Enables C
-fallow-editor-placeholders	Supported	Treats edi
-fallow-fortran-gnu-ext	Supported	Allows Fo
-fansi-escape-codes	Supported	Uses ANS
-fapple-kext	Unsupported	Uses App
-fapple-link-rtlib	Unsupported	Forces lin
-fapple-pragma-pack	Unsupported	Enables A
-fapplication-extension	Unsupported	Restricts c
-fbackslash	Supported	Treats bac
-fbasic-block-sections= <value>	Supported	“Places ea
-fblocks	Supported	Enables th
-fborland-extensions	Unsupported	Accepts n
-fbuild-session-file= <file>	Supported	Uses the l
-fbuild-session-timestamp= <time since Epoch in seconds>	Supported	Specifies
-fbuiltin-module-map	Unsupported	Loads the
-fcall-saved-x10	Unsupported	Makes the
-fcall-saved-x11	Unsupported	Makes the
-fcall-saved-x12	Unsupported	Makes the
-fcall-saved-x13	Unsupported	Makes the
-fcall-saved-x14	Unsupported	Makes the
-fcall-saved-x15	Unsupported	Makes the
-fcall-saved-x18	Unsupported	Makes the
-fcall-saved-x8	Unsupported	Makes the
-fcall-saved-x9	Unsupported	Makes the
-fcf-protection= <value>	Unsupported	Specifies
-fcf-protection	Unsupported	Enables c
-fchar8_t	Supported	Enables C
-fclang-abi-compat= <version>	Supported	Attempts
-fcolor-diagnostics	Supported	Enables c
-fcomment-block-commands= <arg>	Supported	Treats eac
-fcommon	Supported	Places uni
-fcomplete-member-pointers	Supported	Requires m
-fconvergent-functions	Supported	Assumes
-fcoroutines-ts	Supported	Enables su
-fcoverage-mapping	Unsupported	Generates
-fcs-profile-generate= <directory>	Unsupported	Generates
-fcs-profile-generate	Unsupported	Generates
-fcuda-approx-transcendentals	Unsupported	Uses appr
-fcuda-flush-denormals-to-zero	Supported	Flushes de

Option	Support Status	Descript
-fcuda-short-ptr	Unsupported	Uses 32-b
-fcxx-exceptions	Supported	Enables C
-fdata-sections	Supported	Places eac
-fdebug-compilation-dir <value>	Supported	Specifies
-fdebug-default-version= <value>	Supported	Specifies
-fdebug-info-for-profiling	Supported	Emits extr
-fdebug-macro	Supported	Emits mac
-fdebug-prefix-map= <value>	Supported	Remaps fi
-fdebug-ranges-base-address	Supported	Uses DWA
-fdebug-types-section	Supported	Places deb
-fdeclspec	Supported	Allows __
-fdelayed-template-parsing	Supported	Parses ten
-fdelete-null-pointer-checks	Supported	Treats usa
-fdiagnostics-absolute-paths	Supported	Prints abs
-fdiagnostics-hotness-threshold= <number>	Unsupported	Prevents c
-fdiagnostics-parseable-fixits	Supported	Prints fix-
-fdiagnostics-print-source-range-info	Supported	Prints sou
-fdiagnostics-show-hotness	Unsupported	Enables p
-fdiagnostics-show-note-include-stack	Supported	Displays i
-fdiagnostics-show-option	Supported	Prints opt
-fdiagnostics-show-template-tree	Supported	Prints a te
-fdigraphs	Supported	Enables a
-fdiscard-value-names	Supported	Discards v
-fdollars-in-identifiers	Supported	Allows “$
-fdouble-square-bracket-attributes	Supported	Enables ‘[
-fdwarf-exceptions	Unsupported	Uses DWA
-feliminate-unused-debug-types	Supported	Eliminate
-fembed-bitcode-marker	Supported	Embeds p
-fembed-bitcode= <option>	Supported	Embeds L
-fembed-bitcode	Supported	Embeds L
-femit-all-decls	Supported	Emits all
-femulated-tls	Supported	Uses emu
-fenable-matrix	Supported	Enables m
-fexceptions	Supported	Enables su
-fexperimental-new-constant-interpreter	Supported	Enables th
-fexperimental-new-pass-manager	Supported	Enables a
-fexperimental-relative-c++±abi-vtables	Supported	Uses the e
-fexperimental-strict-floating-point	Supported	Enables ex
-ffast-math	Supported	Allows ag
-ffile-prefix-map= <value>	Supported	Remaps fi
-ffine-grained-bitfield-accesses	Supported	Uses sepa
-ffixed-form	Supported	Enables fi
-ffixed-point	Supported	Enables fi
-ffixed-r19	Unsupported	Reserves t
-ffixed-r9	Unsupported	Reserves t
-ffixed-x10	Unsupported	Reserves t
-ffixed-x11	Unsupported	Reserves t
-ffixed-x12	Unsupported	Reserves t
-ffixed-x13	Unsupported	Reserves t
-ffixed-x14	Unsupported	Reserves t

Option	Support Status	Descript
-ffixed-x15	Unsupported	Reserves
-ffixed-x16	Unsupported	Reserves
-ffixed-x17	Unsupported	Reserves
-ffixed-x18	Unsupported	Reserves
-ffixed-x19	Unsupported	Reserves
-ffixed-x1	Unsupported	Reserves
-ffixed-x20	Unsupported	Reserves
-ffixed-x21	Unsupported	Reserves
-ffixed-x22	Unsupported	Reserves
-ffixed-x23	Unsupported	Reserves
-ffixed-x24	Unsupported	Reserves
-ffixed-x25	Unsupported	Reserves
-ffixed-x26	Unsupported	Reserves
-ffixed-x27	Unsupported	Reserves
-ffixed-x28	Unsupported	Reserves
-ffixed-x29	Unsupported	Reserves
-ffixed-x2	Unsupported	Reserves
-ffixed-x30	Unsupported	Reserves
-ffixed-x31	Unsupported	Reserves
-ffixed-x3	Unsupported	Reserves
-ffixed-x4	Unsupported	Reserves
-ffixed-x5	Unsupported	Reserves
-ffixed-x6	Unsupported	Reserves
-ffixed-x7	Unsupported	Reserves
-ffixed-x8	Unsupported	Reserves
-ffixed-x9	Unsupported	Reserves
-fforce-dwarf-frame	Supported	Mandator
-fforce-emit-vtables	Supported	Emits mo
-fforce-enable-int128	Supported	Enables s
-ffp-contract= <value>	Supported	Forms fus
-ffp-exception-behavior= <value>	Supported	Specifies
-ffp-model= <value>	Supported	Controls t
-ffree-form	Supported	Enables f
-ffreestanding	Supported	Asserts th
-ffunc-args-alias	Supported	Allows th
-ffunction-sections	Supported	Places ea
-fglobal-isel	Supported	Enables th
-fgnu-keywords	Supported	Allows G
-fgnu-runtime	Unsupported	Generates
-fgnu89-inline	Unsupported	Uses the g
-fgnuc-version= <value>	Supported	Sets vario
-fgpu-allow-device-init	Supported	Allows de
-fgpu-rdc	Supported	Generates
-fhip-new-launch-api	Supported	Uses new
-fignore-exceptions	Supported	Enables s
-fimplicit-module-maps	Unsupported	Implicitly
-finline-functions	Supported	Inlines su
-finline-hint-functions	Supported	Inlines fu
-finstrument-function-entry-bare	Unsupported	Allows in
-finstrument-functions-after-inlining	Unsupported	Similar to

Option	Support Status	Descript
-finstrument-functions	Unsupported	Generates
-fintegrated-as	Supported	Enables th
-fintegrated-cc1	Supported	Runs cc1
-fjump-tables	Supported	Uses jump
-fkeep-static-consts	Supported	Keeps sta
-flax-vector-conversions= <value>	Supported	Enables in
-flto-jobs= <value>	Unsupported	Controls t
-flto= <value>	Unsupported	Sets LTO
-flto	Unsupported	Enables L
-fmacro-prefix-map= <value>	Supported	Remaps fi
-fmath-errno	Supported	Requires r
-fmax-tokens= <value>	Supported	Specifies
-fmax-type-align= <value>	Supported	Specifies
-fmemory-profile	Supported	Enables h
-fmerge-all-constants	Supported	Allows m
-fmessage-length= <value>	Supported	Formats m
-fmodule-file=[<name>=] <file>	Unsupported	Specifies
-fmodule-map-file= <file>	Unsupported	Loads the
-fmodule-name= <name>	Unsupported	Specifies
-fmodules-cache-path= <directory>	Unsupported	Specifies
-fmodules-decluse	Unsupported	Asserts de
-fmodules-disable-diagnostic-validation	Unsupported	Disables v
-fmodules-ignore-macro= <value>	Unsupported	Ignores th
-fmodules-prune-after= <seconds>	Unsupported	Specifies
-fmodules-prune-interval= <seconds>	Unsupported	Specifies
-fmodules-search-all	Unsupported	Searches e
-fmodules-strict-decluse	Unsupported	Similar to
-fmodules-ts	Unsupported	Enables su
-fmodules-user-build-path <directory>	Unsupported	Specifies
-fmodules-validate-input-files-content	Supported	Validates
-fmodules-validate-once-per-build-session	Unsupported	Prohibits
-fmodules-validate-system-headers	Supported	Validates
-fmodules	Unsupported	Enables th
-fms-compatibility-version= <value>	Supported	Specifies
-fms-compatibility	Supported	Enables fu
-fms-extensions	Supported	Accepts so
-fmsc-version= <value>	Supported	Specifies
-fnew-alignment= <align>	Supported	Specifies
-fno-addrsig	Supported	Prohibits
-fno-allow-fortran-gnu-ext	Supported	Allows Fo
-fno-assume-sane-operator-new	Supported	Prohibits
-fno-autolink	Supported	Disables g
-fno-backslash	Supported	Allows tre
-fno-builtin- <value>	Supported	Disables i
-fno-builtin	Supported	Disables i
-fno-c+±static-destructors	Supported	Disables C
-fno-char8_t	Supported	Disables C
-fno-color-diagnostics	Supported	Disables c
-fno-common	Supported	Compiles
-fno-complete-member-pointers	Supported	Eliminate

Option	Support Status	Descript
-fno-constant-cfstrings	Supported	Disables c
-fno-coverage-mapping	Supported	Disables c
-fno-crash-diagnostics	Supported	Disables a
-fno-cuda-approx-transcendentals	Unsupported	Eliminate
-fno-debug-macro	Supported	Prohibits
-fno-declspec	Unsupported	Disallows
-fno-delayed-template-parsing	Supported	Disables c
-fno-delete-null-pointer-checks	Supported	Prohibits
-fno-diagnostics-fixit-info	Supported	Prohibits
-fno-digraphs	Supported	Disallows
-fno-discard-value-names	Supported	Prohibits
-fno-dollars-in-identifiers	Supported	Disallows
-fno-double-square-bracket-attributes	Supported	Disables '
-fno-elide-constructors	Supported	Disables C
-fno-elide-type	Supported	Prohibits
-fno-eliminate-unused-debug-types	Supported	Emits deb
-fno-exceptions	Supported	Disables s
-fno-experimental-new-pass-manager	Supported	Disables a
-fno-experimental-relative-c+±abi-vtables	Supported	Prohibits
-fno-fine-grained-bitfield-accesses	Supported	Allows us
-fno-fixed-form	Supported	Disables f
-fno-fixed-point	Supported	Disables f
-fno-force-enable-int128	Supported	Disables s
-fno-fortran-main	Supported	Prohibits
-fno-free-form	Supported	Disables f
-fno-func-args-alias	Supported	Allows th
-fno-global-isel	Supported	Disables t
-fno-gnu-inline-asm	Supported	Disables C
-fno-gpu-allow-device-init	Supported	Disallows
-fno-hip-new-launch-api	Supported	Disallows
-fno-integrated-as	Supported	Disables t
-fno-integrated-cc1	Supported	Spawns a
-fno-jump-tables	Supported	Disallows
-fno-keep-static-consts	Supported	Prohibits
-fno-lto	Supported	Disables l
-fno-memory-profile	Supported	Disables l
-fno-merge-all-constants	Supported	Disallows
-fno-no-access-control	Supported	Disables C
-fno-objc-infer-related-result-type	Supported	Prohibits
-fno-operator-names	Supported	Disallows
-fno-pch-codegen	Supported	Disallows
-fno-pch-debuginfo	Supported	Prohibits
-fno-plt	Supported	Asserts us
-fno-preserve-as-comments	Supported	Prohibits
-fno-profile-generate	Supported	Disables g
-fno-profile-instr-generate	Supported	Disables g
-fno-profile-instr-use	Supported	Disables u
-fno-register-global-dtors-with-atexit	Supported	Disallows
-fno-rtlib-add-rpath	Supported	Prohibits
-fno-rtti-data	Supported	Disables g

Option	Support Status	Descript
-fno-rtti	Supported	Disables g
-fno-sanitize-address-poison-custom-array-cookie	Supported on Host only	Disables p
-fno-sanitize-address-use-after-scope	Supported on Host only	Disables u
-fno-sanitize-address-use-odr-indicator	Supported on Host only	Disables C
-fno-sanitize-blacklist	Supported on Host only	Prohibits
-fno-sanitize-cfi-canonical-jump-tables	Supported on Host only	Prohibits
-fno-sanitize-cfi-cross-dso	Supported on Host only	Disables c
-fno-sanitize-coverage= <value>	Supported on Host only	Disables s
-fno-sanitize-memory-track-origins	Supported on Host only	Disables c
-fno-sanitize-memory-use-after-dtor	Supported on Host only	Disables u
-fno-sanitize-recover= <value>	Supported on Host only	Disables r
-fno-sanitize-stats	Supported on Host only	Disables s
-fno-sanitize-thread-atomics	Supported on Host only	Disables a
-fno-sanitize-thread-func-entry-exit	Supported on Host only	Disables f
-fno-sanitize-thread-memory-access	Supported on Host only	Disables r
-fno-sanitize-trap= <value>	Supported on Host only	Disables t
-fno-sanitize-trap	Supported on Host only	Disables t
-fno-short-wchar	Supported	Forces wc
-fno-show-column	Supported	Prohibits
-fno-show-source-location	Supported	Prohibits
-fno-signed-char	Supported	char is un
-fno-signed-zeros	Supported	Allows op
-fno-spell-checking	Supported	Disables s
-fno-split-machine-functions	Supported	Disables l
-fno-stack-clash-protection	Supported	Disables s
-fno-stack-protector	Supported	Disables t
-fno-standalone-debug	Supported	Limits deb
-fno-strict-float-cast-overflow	Supported	Relaxes la
-fno-strict-return	Supported	Prohibits
-fno-sycl	Unsupported	Disables S
-fno-temp-file	Supported	Asserts di
-fno-threadsafe-statics	Supported	Prohibits
-fno-trigraphs	Supported	Prohibits
-fno-unique-section-names	Supported	Prohibits
-fno-unroll-loops	Supported	Turns off
-fno-use-cxa-atexit	Supported	Prohibits
-fno-use-flang-math-libs	Supported	Asserts th
-fno-use-init-array	Supported	Asserts th
-fno-visibility-inlines-hidden-static-local-var	Supported	Disables -
-fno-xray-function-index	Unsupported	Allows or
-fno-zero-initialized-in-bss	Supported	Prohibits
-fobjc-arc-exceptions	Unsupported	Asserts us
-fobjc-arc	Unsupported	Synthesiz
-fobjc-exceptions	Unsupported	Enables C
-fobjc-runtime= <value>	Unsupported	Specifies
-fobjc-weak	Unsupported	Enables A
-fopenmp-simd	Unsupported	Emits Op
-fopenmp-targets= <value>	Unsupported	Specifies
-fopenmp	Unsupported	Parses Op
-foptimization-record-file= <file>	Supported	Specifies

Option	Support Status	Descript
-foptimization-record-passes= <regex>	Supported	Exclusive
-forder-file-instrumentation	Supported	Generates
-fpack-struct= <value>	Unsupported	Specifies
-fpascal-strings	Supported	Recogniz
-fpass-plugin= <dsopath>	Supported	Loads pas
-fpatchable-function-entry= <N,M>	Supported	Generates
-fpcc-struct-return	Unsupported	Overrides
-fpch-codegen	Supported	Generates
-fpch-debuginfo	Supported	Generates
-fpch-instantiate-templates	Supported	Instantiat
-fpch-validate-input-files-content	Supported	Validates
-fplugin= <dsopath>	Supported	Loads the
-fprebuilt-module-path= <directory>	Unsupported	Specifies
-fprofile-exclude-files= <value>	Unsupported	Exclusive
-fprofile-filter-files= <value>	Unsupported	Exclusive
-fprofile-generate= <directory>	Unsupported	Generates
-fprofile-generate	Unsupported	Generates
-fprofile-instr-generate= <file>	Unsupported	Generates
-fprofile-instr-generate	Unsupported	Generates
-fprofile-instr-use= <value>	Unsupported	Uses instr
-fprofile-remapping-file= <file>	Unsupported	Uses the r
-fprofile-sample-accurate	Unsupported	Specifies
-fprofile-sample-use= <value>	Unsupported	Enables s
-fprofile-use= <pathname>	Unsupported	Uses instr
-freciprocal-math	Supported	Allows di
-freg-struct-return	Unsupported	Overrides
-fregister-global-dtors-with-atexit	Supported	Uses atex
-frelaxed-template-template-args	Supported	Enables C
-freroll-loops	Supported	Turns on l
-fropi	Unsupported	Generates
-frtlib-add-rpath	Supported	Adds -rpa
-frwpi	Unsupported	Generates
-fsanitize-address-field-padding= <value>	Supported on Host only	Specifies
-fsanitize-address-globals-dead-stripping	Supported on Host only	Enables li
-fsanitize-address-poison-custom-array-cookie	Supported on Host only	Enables p
-fsanitize-address-use-after-scope	Supported on Host only	Enables u
-fsanitize-address-use-odr-indicator	Supported on Host only	Enables C
-fsanitize-blacklist= <value>	Supported on Host only	Specifies
-fsanitize-cfi-canonical-jump-tables	Supported on Host only	Makes the
-fsanitize-cfi-cross-dso	Supported on Host only	Enables c
-fsanitize-cfi-icall-generalize-pointers	Supported on Host only	Generaliz
-fsanitize-coverage-allowlist= <value>	Supported on Host only	Restricts s
-fsanitize-coverage-blacklist= <value>	Supported on Host only	Deprecate
-fsanitize-coverage-blocklist= <value>	Supported on Host only	Disables s
-fsanitize-coverage-whitelist= <value>	Supported on Host only	Deprecate
-fsanitize-coverage= <value>	Supported on Host only	Specifies
-fsanitize-hwaddress-abi= <value>	Supported on Host only	Selects th
-fsanitize-memory-track-origins= <value>	Supported on Host only	Enables o
-fsanitize-memory-track-origins	Supported on Host only	Enables o
-fsanitize-memory-use-after-dtor	Supported on Host only	Enables u

Option	Support Status	Descript
-fsanitize-recover= <value>	Supported on Host only	Enables re
-fsanitize-stats	Supported on Host only	Enables sa
-fsanitize-system-blacklist= <value>	Supported on Host only	Specifies
-fsanitize-thread-atomics	Supported on Host only	Enables a
-fsanitize-thread-func-entry-exit	Supported on Host only	Enables fu
-fsanitize-thread-memory-access	Supported on Host only	Enables m
-fsanitize-trap= <value>	Supported on Host only	Enables tr
-fsanitize-trap	Supported on Host only	Enables tr
-fsanitize-undefined-strip-path-components= <number>	Supported on Host only	Strips (or
-fsanitize= <check>	Supported on Host only	Turns on r
-fsave-optimization-record= <format>	Supported	Generates
-fsave-optimization-record	Supported	Generates
-fseh-exceptions	Supported	Uses SEH
-fshort-enums	Supported	Allocates
-fshort-wchar	Unsupported	Forces wc
-fshow-overloads= <value>	Supported	Specifies
-fsigned-char	Supported	Asserts th
-fsized-deallocation	Supported	Enables C
-fsjlj-exceptions	Supported	Uses SjLj
-fslp-vectorize	Supported	Enables th
-fsplit-dwarf-inlining	Unsupported	Provides m
-fsplit-lto-unit	Unsupported	Enables sp
-fsplit-machine-functions	Supported	Enables la
-fstack-clash-protection	Supported	Enables st
-fstack-protector-all	Unsupported	Enables st
-fstack-protector-strong	Unsupported	Enables st
-fstack-protector	Unsupported	Enables st
-fstack-size-section	Supported	Emits sec
-fstandalone-debug	Supported	Emits full
-fstrict-enums	Supported	Enables o
-fstrict-float-cast-overflow	Supported	Assumes
-fstrict-vtable-pointers	Supported	Enables o
-fsycl	Unsupported	Enables S
-fsystem-module	u	Builds thi
-fthin-link-bitcode= <value>	Supported	Writes mi
-fthinlto-index= <value>	Unsupported	Performs
-ftime-trace-granularity= <value>	Supported	Specifies
-ftime-trace	Supported	Turns on t
-ftrap-function= <value>	Unsupported	Issues cal
-ftrapv-handler= <function name>	Unsupported	Specifies
-ftrapv	Unsupported	Traps on i
-ftrigraphs	Supported	Processes
-ftrivial-auto-var-init-stop-after= <value>	Supported	Stops initi
-ftrivial-auto-var-init= <value>	Supported	Initializes
-funique-basic-block-section-names	Supported	Uses uniq
-funique-internal-linkage-names	Supported	Makes the
-funroll-loops	Supported	Turns on l
-fuse-flang-math-libs	Supported	Uses Flan
-fuse-line-directives	Supported	Uses #line
-fvalidate-ast-input-files-content	Supported	Computes

Option	Support Status	Descript
-fveclib= <value>	Unsupported	Uses the g
-fvectorize	Unsupported	Enables th
-fverbose-asm	Supported	Generates
-fvirtual-function-elimination	Supported	Enables d
-fvisibility-global-new-delete-hidden	Supported	Marks the
-fvisibility-inlines-hidden-static-local-var	Supported	Marks the
-fvisibility-inlines-hidden	Supported	Marks the
-fvisibility-ms-compat	Supported	Marks the
-fvisibility= <value>	Supported	Sets the d
-fwasm-exceptions	Unsupported	Uses Web
-fwhole-program-vtables	Unsupported	Enables w
-fwrapv	Supported	Treats sig
-fwritable-strings	Supported	Stores stri
-fxray-always-emit-customevents	Unsupported	Mandates
-fxray-always-emit-typedevents	Unsupported	Mandates
-fxray-always-instrument= <value>	Unsupported	Deprecate
-fxray-attr-list= <value>	Unsupported	Specifies
-fxray-ignore-loops	Unsupported	Prohibits
-fxray-instruction-threshold= <value>	Unsupported	Sets the m
-fxray-instrumentation-bundle= <value>	Unsupported	Specifies
-fxray-instrument	Unsupported	Generates
-fxray-link-deps	Unsupported	Informs C
-fxray-modes= <value>	Unsupported	Specifies
-fxray-never-instrument= <value>	Unsupported	Deprecate
-fzvector	Supported	Enables S
-F <value>	Unsupported	Adds dire
–gcc-toolchain= <value>	Supported	Uses the g
-gcodeview-ghash	Supported	Emits typ
-gcodeview	Supported	Generates
-gdwarf-2	Supported	Generates
-gdwarf-3	Supported	Generates
-gdwarf-4	Supported	Generates
-gdwarf-5	Supported	Generates
-gdwarf	Supported	Generates
-gembed-source	Supported	Embeds s
-gline-directives-only	Supported	Emits deb
-gline-tables-only	Supported	Emits deb
-gmodules	Supported	Generates
-gno-embed-source	Supported	Restores t
-gno-inline-line-tables	Supported	Prohibits
–gpu-max-threads-per-block= <value>	Supported	Specifies
-gsplit-dwarf= <value>	Supported	Sets DWA
-gz= <value>	Supported	Specifies
-gz	Supported	Shows DW
-G <size>	Unsupported	Puts objec
-g	Supported	Generates
–help-hidden	Supported	Displays h
-help	Supported	Displays a
–hip-device-lib= <value>	Supported	Specifies
–hip-link	Supported	Links clan

Option	Support Status	Descript
–hip-version= <value>	Supported	Allows sp
-H	Supported	Shows he
-I-	Supported	Restricts
-ibuiltininc	Supported	Enables b
-idirafter <value>	Supported	Adds the
-iframeworkwithsysroot <directory>	Unsupported	Adds the
-iframework <value>	Unsupported	Adds the
-imacros <file>	Supported	Specifies
-include-pch <file>	Supported	Includes t
-include <file>	Supported	Includes t
-index-header-map	Supported	Makes the
-iprefix <dir>	Supported	Sets the -i
-iquote <directory>	Supported	Adds the
-isysroot <dir>	Supported	Sets the s
-isystem-after <directory>	Supported	Adds the
-isystem <directory>	Supported	Adds the
-ivfsoverlay <value>	Supported	Overlays t
-iwithprefixbefore <dir>	Supported	Sets the d
-iwithprefix <dir>	Supported	Sets the d
-iwithsysroot <directory>	Supported	Adds dire
-I <dir>	Supported	Adds dire
–libomptarget-nvptx-path= <value>	Unsupported	Specifies
-L <dir>	Supported	Adds dire
-mabicalls	Unsupported	Enables S
-maix-struct-return	Unsupported	Returns al
-malign-branch-boundary= <value>	Supported	Specifies
-malign-branch= <value>	Supported	Specifies
-malign-double	Supported	Aligns do
-Mallocatable= <value>	Unsupported	Provides s
-mbackchain	Unsupported	Links stac
-mbranch-protection= <value>	Unsupported	Enforces t
-mbranches-within-32B-boundaries	Supported	Aligns sel
-mcmodel=medany	Unsupported	Equivalen
-mcmodel=medlow	Unsupported	Equivalen
-mcmse	Unsupported	Allows us
-mcode-object-v3	Supported	Legacy op
-mcode-object-version= <version>	Supported	Specifies
-mcrc	Unsupported	Allows us
-mcumode	Supported	Specifies
-mdouble= <value>	Supported	Forces do
-MD	Supported	Writes a d
-meabi <value>	Supported	Sets EAB
-membedded-data	Unsupported	Places co
-menable-experimental-extensions	Unsupported	Enables u
-mexec-model= <value>	Unsupported	Specifies
-mexecute-only	Unsupported	Disallows
-mextern-sdata	Unsupported	Assumes
-mfentry	Unsupported	Inserts ca
-mfix-cortex-a53-835769	Unsupported	Workarou
0	Unsupported	Asserts us

Option	Support Status	Descript
0	Unsupported	Asserts us
-MF <file>	Supported	Writes de
-mgeneral-regs-only	Unsupported	Generates
-mglobal-merge	Supported	Enables m
-mgpopt	Unsupported	Allows us
-MG	Supported	Adds miss
-mharden-sls= <value>	Unsupported	Sets straig
-mhvx-length= <value>	Unsupported	Sets Hexa
-mhvx= <value>	Unsupported	Sets Hexa
-mhvx	Unsupported	Enables H
-miamcu	Unsupported	Allows us
–migrate	Unsupported	Runs the m
-mincremental-linker-compatible	Supported	(integrate
-mindirect-jump= <value>	Unsupported	Changes i
-Minform= <value>	Supported	Sets error
-mios-version-min= <value>	Unsupported	Sets iOS d
-MJ <value>	Unsupported	Writes a c
-mllvm <value>	Supported	Specifies
-mlocal-sdata	Unsupported	Extends th
-mlong-calls	Supported	Generates
-mlong-double-128	Supported on Host only	Forces lon
-mlong-double-64	Supported	Forces lon
-mlong-double-80	Supported on Host only	Forces lon
-mlvi-cfi	Supported on Host only	Enables o
-mlvi-hardening	Supported on Host only	Enables a
-mmacosx-version-min= <value>	Unsupported	Sets Mac
-mmadd4	Supported	Enables th
-mmark-bti-property	Unsupported	Adds .not
-MMD	Supported	Writes a d
-mmemops	Supported	Enables g
-mms-bitfields	Unsupported	Sets the d
-mmsa	Unsupported	Enables M
-mmt	Unsupported	Enables M
-MM	Supported	Similar to
-mno-abicalls	Unsupported	Disables S
-mno-crc	Unsupported	Disallows
-mno-embedded-data	Unsupported	Prohibits
-mno-execute-only	Unsupported	Allows ge
-mno-extern-sdata	Unsupported	Prohibits
-mno-fix-cortex-a53-835769	Unsupported	Disallows
-mno-global-merge	Supported	Disables m
-mno-gpopt	Unsupported	Prohibits
-mno-hvx	Unsupported	Disables H
-mno-implicit-float	Supported	Prohibits
-mno-incremental-linker-compatible	Supported	(integrate
-mno-local-sdata	Unsupported	Prohibits
-mno-long-calls	Supported	Restores t
-mno-lvi-cfi	Supported on Host only	Disables c
-mno-lvi-hardening	Supported on Host only	Disables m
-mno-madd4	Supported	Disables t

Option	Support Status	Descript
-mno-memops	Supported	Disables t
-mno-movt	Supported	Disallows
-mno-ms-bitfields	Supported	Prohibits
-mno-msa	Unsupported	Disables M
-mno-mt	Unsupported	Disables M
-mno-neg-immediates	Supported	Disallows
-mno-nvj	Supported	Disables g
-mno-nvs	Supported	Disables g
-mno-outline	Unsupported	Disables f
-mno-packets	Supported	Disables g
-mno-relax	Supported	Disables l
-mno-restrict-it	Unsupported	Allows ge
-mno-save-restore	Unsupported	Disables u
-mno-seses	Unsupported	Disables s
-mno-stack-arg-probe	Supported	Disables s
-mno-tls-direct-seg-refs	Supported	Disables d
-mno-unaligned-access	Unsupported	Forces all
-mno-wavefrontsize64	Supported	Asserts w
-mnocrc	Unsupported	Disallows
-mnop-mcount	Supported	Generates
-mnvj	Supported	Enables g
-mnvs	Supported	Enables g
-module-dependency-dir <value>	Unsupported	Specifies
-module-file-info	Unsupported	Provides i
-momit-leaf-frame-pointer	Supported	Omits fra
-moutline	Unsupported	Enables fu
-mpacked-stack	Unsupported	Asserts th
-mpackets	Supported	Enables g
-mpad-max-prefix-size= <value>	Supported	Specifies
-mpie-copy-relocations	Supported	Asserts th
-mprefer-vector-width= <value>	Unsupported	Specifies
-MP	Supported	Creates ph
-mqdsp6-compat	Unsupported	Enables h
-MQ <value>	Supported	Specifies
-mrecord-mcount	Supported	Generates
-mrelax-all	Supported	(integrate
-mrelax	Supported	Enables li
-mrestrict-it	Unsupported	Disallows
-mrtd	Unsupported	Makes Sto
-msave-restore	Unsupported	Enables u
-mseses	Unsupported	Enables sp
-msign-return-address= <value>	Unsupported	Specifies
-msmall-data-limit= <value>	Supported	Puts globa
-msoft-float	Supported	Uses softw
-msram-ecc	Supported	Legacy op
-mstack-alignment= <value>	Unsupported	Sets the st
-mstack-arg-probe	Unsupported	Enables st
-mstack-probe-size= <value>	Unsupported	Sets the st
-mstackrealign	Unsupported	Forces rea
-msve-vector-bits= <value>	Unsupported	Specifies

Option	Support Status	Descript
-msvr4-struct-return	Unsupported	Returns s
-mthread-model <value>	Supported	Specifies
-mtls-direct-seg-refs	Supported	Enables d
-mtls-size= <value>	Unsupported	Specifies
-mtp= <value>	Unsupported	Specifies
-mtune= <value>	Supported on Host only	Supported
-MT <value>	Unsupported	Specifies
-munaligned-access	Unsupported	Allows m
-MV	Supported	Uses NM
-mwavefrontsize64	Supported	Asserts w
-mxnack	Supported	Legacy op
-M	Supported	Similar to
–no-cuda-include-ptx= <value>	Supported	Prohibits
–no-cuda-version-check	Supported	Disallows
-no-flang-libs	Supported	Prohibits
–no-offload-arch= <value>	Supported	Removes
–no-system-header-prefix= <prefix>	Supported	Assumes
-nobuiltininc	Supported	Disables b
-nogpuinc	Supported	Prohibits
-nogpulib	Supported	Prohibits
-nostdinc++	Unsupported	Disables s
-ObjC++	Unsupported	Treats sou
-objcmt-atomic-property	Unsupported	Enables m
-objcmt-migrate-all	Unsupported	Enables m
-objcmt-migrate-annotation	Unsupported	Enables m
-objcmt-migrate-designated-init	Unsupported	Enables m
-objcmt-migrate-instancetype	Unsupported	Enables m
-objcmt-migrate-literals	Unsupported	Enables m
-objcmt-migrate-ns-macros	Unsupported	Enables m
-objcmt-migrate-property-dot-syntax	Unsupported	Enables m
-objcmt-migrate-property	Unsupported	Enables m
-objcmt-migrate-protocol-conformance	Unsupported	Enables m
-objcmt-migrate-readonly-property	Unsupported	Enables m
-objcmt-migrate-readwrite-property	Unsupported	Enables m
-objcmt-migrate-subscripting	Unsupported	Enables m
-objcmt-ns-nonatomic-iosonly	Unsupported	Enables m
-objcmt-returns-innerpointer-property	Unsupported	Enables m
-objcmt-whitelist-dir-path= <value>	Unsupported	Modifies
-ObjC	Unsupported	Treats sou
–offload-arch= <value>	Supported	Specifies
-o <file>	Supported	Writes ou
-parallel-jobs= <value>	Supported	Specifies
-pg	Supported	Enables m
-pipe	Supported	Asserts us
–precompile	Supported	Only prec
-print-effective-triple	Supported	Prints the
-print-file-name= <file>	Supported	Prints the
-print-ivar-layout	Unsupported	Enables O
-print-libgcc-file-name	Supported	“Prints th
-print-prog-name= <name>	Supported	Prints the

Option	Support Status	Descript
-print-resource-dir	Supported	Prints the
-print-search-dirs	Supported	Prints the
-print-supported-cpus	Supported	Prints the
-print-target-triple	Supported	Prints the
-print-targets	Supported	Prints the
-pthread	Supported	Supports
–ptxas-path= <value>	Unsupported	Specifies
-P	Supported	Disables l
-Qn	Supported	Prohibits
-Qunused-arguments	Supported	Prohibits
-Qy	Supported	Emits me
-relocatable-pch	Supported	Allows to
-rewrite-legacy-objc	Unsupported	Rewrites
-rewrite-objc	Unsupported	Rewrites
–rocm-device-lib-path= <value>	Supported	Specifies
–rocm-path= <value>	Supported	Specifies
-Rpass-analysis= <value>	Supported	Reports tr
-Rpass-missed= <value>	Supported	Reports m
-Rpass= <value>	Supported	Reports tr
-rtlib= <value>	Unsupported	Specifies
-R <remark>	Unsupported	Enables th
-save-stats= <value>	Supported	Saves llvm
-save-stats	Supported	Saves llvm
-save-temps= <value>	Supported	Saves inte
-save-temps	Supported	Saves inte
-serialize-diagnostics= <value>	Supported	Serializes
-shared-libsan	Unsupported	Dynamica
-static-flang-libs	Supported	Asserts li
-static-libsan	Unsupported	Statically
-static-openmp	Supported	Asserts us
-std= <value>	Supported	Specifies
-stdlib+±isystem <directory>	Supported	Specifies
-stdlib= <value>	Supported	Specifies
-sycl-std= <value>	Unsupported	Specifies
–system-header-prefix= <prefix>	Supported	Assumes
-S	Supported	Runs only
–target= <value>	Supported	Generates
-Tbss <addr>	Supported	Sets the st
-Tdata <addr>	Supported	Sets the st
-time	Supported	Times ind
-traditional-cpp	Unsupported	Enables s
-trigraphs	Supported	Processes
-Ttext <addr>	Supported	Sets starti
-T \ <script\>	Unsupported	Specifies
-undef	Supported	undefs all
-unwindlib= <value>	Supported	Specifies
-U <macro>	Supported	Undefines
–verify-debug-info	Supported	Verifies th
-verify-pch	Unsupported	Loads and
–version	Supported	Prints ver

Option	Support Status	Descript
-v	Supported	Shows co
-Wa, <arg>	Supported	Passes the
-Wdeprecated	Supported	Enables w
-Wl, <arg>	Supported	Passes co
-working-directory <value>	Supported	Resolves
-Wp, <arg>	Supported	Passes co
-W <warning>	Supported	Enables th
-w	Supported	Suppresse
-Xanalyzer <arg>	Supported	Passes <a
-Xarch_device <arg>	Supported	Passes <a
-Xarch_host <arg>	Supported	Passes <a
-Xassembler <arg>	Supported	Passes <a
-Xclang <arg>	Supported	Passes <a
-Xcuda-fatbinary <arg>	Supported	Passes <a
-Xcuda-ptxas <arg>	Supported	Passes <a
-Xlinker <arg>	Supported	Passes <a
-Xopenmp-target= <triple> <arg>	Supported	Passes <a
-Xopenmp-target <arg>	Supported	Passes <a
-Xpreprocessor <arg>	Supported	Passes <a
-x <language>	Supported	Assumes
-z <arg>	Supported	Passes -z

CHAPTER

TWELVE

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.

12.1 HIP Runtime

HIP Runtime The HIP Runtime is used to enable GPU acceleration for all HIP language based products.

- hip:.doxygen/docBin/html/index
- Examples

12.2 Porting tools

HIPIFY HIPIFY assists with porting applications from based on CUDA to the HIP Runtime. Supported CUDA APIs are documented here as well.

- Reference Manual

CHAPTER

THIRTEEN

OPENMP SUPPORT IN ROCM

13.1 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 and OS Support (Linux)*.

13.1.1 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.

13.2 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 Makefile in the example above uses a more classical and verbose set of flags which can also be used:

```
-fopenmp -fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa
```

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.

13.2.1 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 ROCm Profiling Tools document on rocprof.

13.2.2 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 ROCm Profiling Tools document on rocprof.

13.2.3 Environment Variables

Environment Variable	Description
OMP_NUM_	The implementation chooses the number of teams for kernel launch. The user can change this number for performance tuning using this environment variable, subject to implementation limits.
LIBOMPTA	This environment variable is used 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.
LIBOMPTA	This environment variable is used to print informational messages from the device runtime as the program executes. Users can request fine-grain information by setting it to the value of 1 or higher and can set the value of -1 for complete information.
LIBOMPTA	If a debug version of the device library is present, setting this environment variable to 1 and using that library emits further detailed debugging information about data transfer operations and kernel launch.
GPU_MAX_	This environment variable is used to set the number of HSA queues in the OpenMP runtime.

13.3 OpenMP: Features

The OpenMP programming model is greatly enhanced with the following new features implemented in the past releases.

13.3.1 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.

13.3.1.1 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

13.3.1.2 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, the programmer should use

```
--offload-arch=gfx908:xnack+
```

Or, equivalently

```
--offload-arch=gfx908
```

Note: The second case is called Xnack-any and it is functionally equivalent to the first case.

At runtime, programmers enable Xnack functionality on a per-application basis using an environment variable:

```
HSA_XNACK=1
```

When Xnack support is not needed, then applications can be built to maximize resource utilization using:

```
--offload-arch=gfx908:xnack-
```

At runtime, the `HSA_XNACK` environment variable can be set to 0, as Xnack functionality is not needed.

13.3.1.3 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];
```

(continues on next page)

(continued from previous page)

```
  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.

13.3.2 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.

13.3.3 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;
```

13.3.4 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.

13.3.5 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 assumes that every thread executes a single iteration of the user loop, which implies that the runtime launches 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.

To enable the generation of the specialized kernel, follow these guidelines:

- Do not specify teams, threads, and schedule-related environment variables. The `num_teams` or a `thread_limit` clause in an OpenMP target construct acts as an override and prevents the generation of the specialized kernel. As the user is unable to specify the number of teams and threads used within target regions in the absence of the above-mentioned environment variables, the runtime will select the best values for the launch configuration based on runtime knowledge of the program.
- Assert the absence of the above-mentioned environment variables by adding the command-line option `-fopenmp-target-ignore-env-vars`. This option also allows programmers to enable the No-loop functionality at lower optimization levels.
- Also, the No-loop functionality is automatically enabled when `-O3` or `-Ofast` is used for compilation. To disable this feature, use `-fno-openmp-target-ignore-env-vars`.

Note The compiler might not generate the No-loop kernel in certain scenarios where the performance improvement is not substantial.

13.3.6 Cross-Team Optimized Reductions

In scenarios where a No-loop kernel is generated but the OpenMP construct has a reduction clause, the compiler may generate optimized code utilizing efficient Cross-Team (Xteam) communication. No separate user option is required, and there is a significant performance improvement with Xteam reduction. New APIs for Xteam reduction are implemented in the device runtime, and clang generates these APIs automatically.

CHAPTER

FOURTEEN

MATH LIBRARIES

AMD provides various math domain and support libraries as part of the ROCm.

14.1 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

Fast Fourier Transforms

- rocFFT
- hipFFT

Random Numbers

- rocRAND
- hipRAND

14.2 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
- 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
- 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
- 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
- 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
- Changelog

rocSOLVER `rocSOLVER` provides a subset of LAPACK (Linear Algebra Package) functionality on the ROCm platform.

- Documentation
- Changelog

hipSOLVER `hipSOLVER` is a LAPACK marshalling library supporting both `rocSOLVER` and `cuSOLVER` as backends whilst exporting a unified interface.

- Documentation
- Changelog

rocSPARSE `rocSPARSE` is a library to provide BLAS for sparse computations.

- Documentation
- Changelog

hipSPARSE `hipSPARSE` is a marshalling library to provide sparse BLAS functionality, supporting both `rocSPARSE` and `cuSPARSE` as backends.

- Documentation
- Changelog

14.3 Fast Fourier Transforms

ROCm libraries for FFT are as follows:

rocFFT rocFFT is an AMD GPU optimized library for FFT.

- Documentation
- 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
- Changelog

14.4 Random Numbers

rocRAND rocRAND is an AMD GPU optimized library for pseudo-random number generators (PRNG).

- Documentation
- 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
- Changelog

CHAPTER

FIFTEEN

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
- 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
- Changelog
- Examples

hipCUB hipCUB is a template library of algorithm primitives with a CUB-compatible interface. It's back-end is rocPRIM.

- Documentation
- Changelog
- Examples

CHAPTER

SIXTEEN

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
- Changelog
- Examples

CHAPTER

SEVENTEEN

AI LIBRARIES

MIOpen AMD's library for high performance machine learning primitives.

- Documentation

Composable Kernel Composable Kernel: Performance Portable Programming Model for Machine Learning Tensor Operators

- Documentation

MIGraphX AMD MIGraphX is AMD's graph inference engine that accelerates machine learning model inference.

- Documentation

CHAPTER

EIGHTEEN

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

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

NINETEEN

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.

- GitHub
- Examples

ROCm SMI 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 Datacenter Tool The ROCm™ Data Center Tool simplifies the administration and addresses key infrastructure challenges in AMD GPUs in cluster and data center environments.

- Documentation
- GitHub
- Examples

CHAPTER

TWENTY

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

TransferBench TransferBench is a simple utility capable of benchmarking simultaneous transfers between user-specified devices (CPUs/GPUs).

- Documentation
- Changelog
- transferbench:examples/index

CHAPTER

TWENTYONE

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

TWENTYTWO

ROCM COMPILERS DISAMBIGUATION

ROCm ships multiple compilers of varying origins and purposes. This article disambiguates compiler naming used throughout the documentation.

22.1 Compiler Terms

Term	Description
`amdcl`	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

TWENTYTHREE

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 CMake's HIP language support will require CMake 3.21 or higher.

23.1 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's 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).

23.2 Using HIP in CMake

ROCm componenents providing a C/C++ interface support being consumed using any C/C++ toolchain that CMake knows how to drive. ROCm also supports CMake's 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.

23.2.1 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.

23.2.2 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.

23.2.3 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)
```

23.2.4 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 for eg.

- 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 eg. `-D GPU_TARGETS="gfx1032;gfx1035"`.

23.2.5 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`

23.3 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'n'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 Intergration (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.

23.3.1 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"
      ]
    }
  ],
```

(continues on next page)

(continued from previous page)

```
  "buildPresets": [
    {
      "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

TWENTYFOUR

LINUX FOLDER STRUCTURE REORGANIZATION

24.1 Introduction

ROCm™ packages have adopted the Linux foundation file system hierarchy standard to ensure ROCm components follow open source conventions for Linux-based distributions. Following is the ROCm proposed file structure.

```
/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
              | -- <component>
                   | -- man pages
         | -- doc
              | -- <component>
                   | -- license files
         | -- <component>
              | -- samples
              | -- architecture independent misc files
```

24.2 Changes from earlier ROCm versions

ROCm with the file reorganization is going to have a lean structure. Following table gives the comparison with new and old folder structure.

```
 ____________________________________________________
|  New File Structure          |  Old File Structure    |
|______________________________|________________________|
| /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   |
|_____________________________________________________|
```

24.3 ROCm File reorganization transition plan

New file organization for ROCm was first introduced ROCm v5.2 release. Backward compatibility was in place to make sure users had a chance to change their applications using ROCm. 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.

24.3.1 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.

```
#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 deprecation plan for backward compatibility wrapper header files is as follows

- `#pragma` message announcing deprecation – ROCm v5.2 release.
- `#pragma` message changed to `#warning` – Future release, tentatively ROCm v5.5.
- `#warning` changed to `#error` – Future release, tentatively ROCm v5.6.
- Backward compatibility wrappers removed – Future release, tentatively ROCm v6.0.

24.3.2 Executable files

Executable files are available in the /opt/rocm-xxx/bin folder. For backward compatibility, the old library location (/opt/rocm-xxx/<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
```

24.3.3 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. 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
```

24.3.4 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. 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
```

24.4 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. Application 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.

24.5 References

ROCm deprecation warning

Linux File System Standard

CHAPTER

TWENTYFIVE

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.

25.1 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.

25.1.1 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 25.1: Example to expose the 1. device and a device based on UUID.

```
export ROCR_VISIBLE_DEVICES="0,GPU-DEADBEEFDEADBEEF"
```

25.1.2 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 25.2: Example to expose the 1. and 3. device in the system.

```
export GPU_DEVICE_ORDINAL="0,2"
```

25.1.3 HIP_VISIBLE_DEVICES

Device indices exposed to HIP applications.

Runtime : HIP Runtime. Applies only to applications using HIP on the AMD platform.

Listing 25.3: Example to expose the 1. and 3. devices in the system.

```
export HIP_VISIBLE_DEVICES="0,2"
```

25.1.4 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.

25.1.5 OMP_DEFAULT_DEVICE

Default device used for OpenMP target offloading.

Runtime : OpenMP Runtime. Applies only to applications using OpenMP offloading.

Listing 25.4: Example on setting the default device to the third device.

```
export OMP_DEFAULT_DEVICE="2"
```

25.2 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.

25.3 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

TWENTYSIX

GPU ARCHITECTURES

26.1 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*

26.2 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

26.3 White Papers

- AMD CDNA™ 2 Architecture White Paper
- AMD CDNA Architecture White Paper
- AMD Vega Architecture White Paper
- AMD RDNA Architecture White Paper

26.4 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.

26.4.1 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. 26.1 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. 26.1 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.

Table 26.1: 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

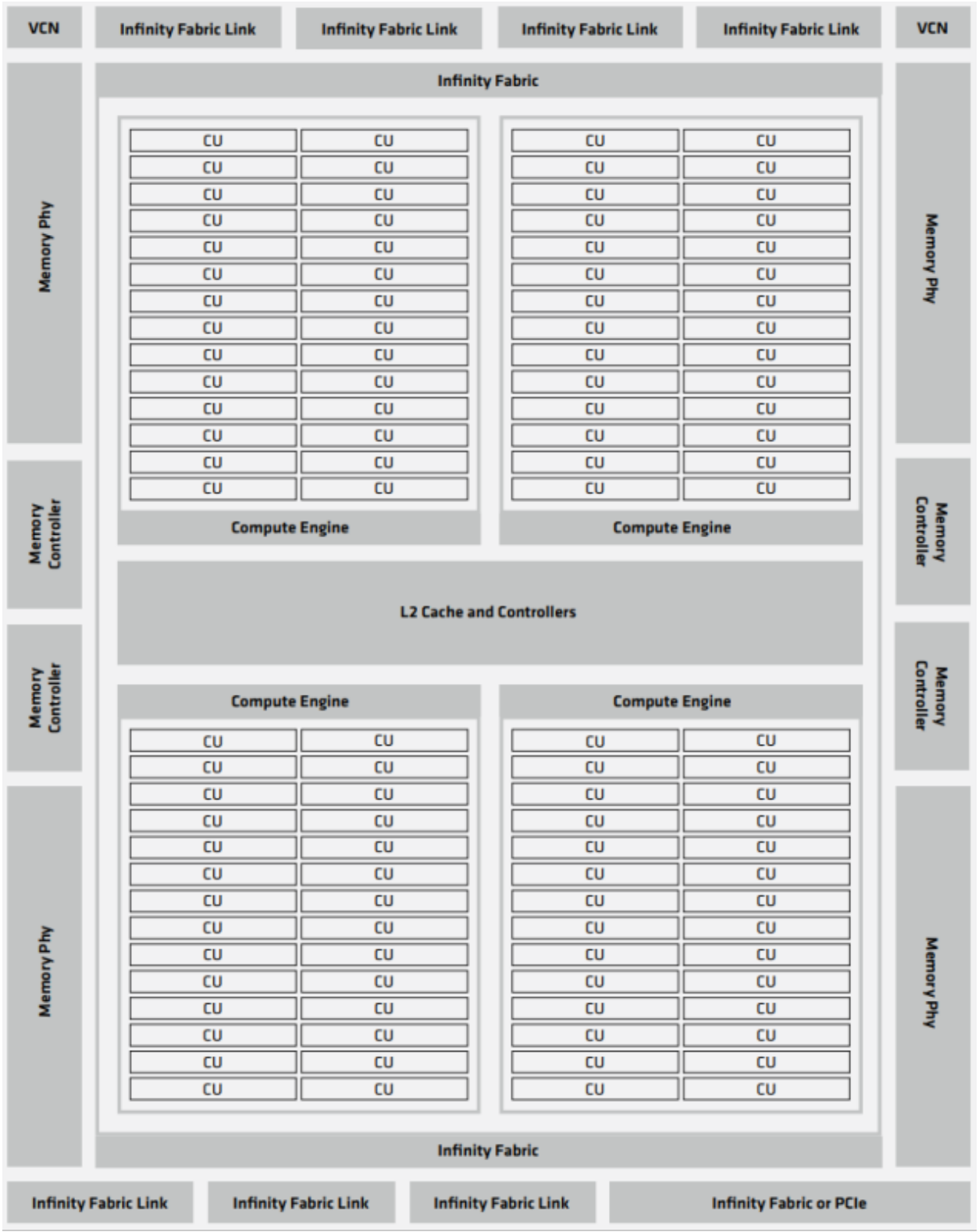


Fig. 26.1: Figure 1: Structure of a single GCD in the AMD Instinct MI250 accelerator.

Table 26.1 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).

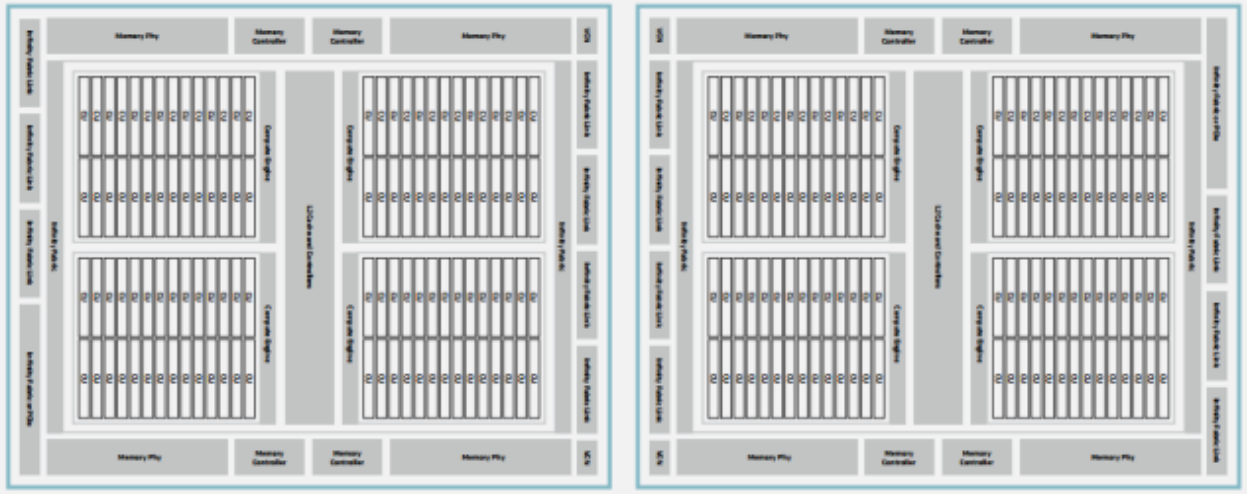


Fig. 26.2: Dual-GCD architecture of the AMD Instinct MI250 accelerators.

Fig. 26.2 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.

26.4.2 Node-level Architecture

Fig. 26.3 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. 26.3 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. 26.3.

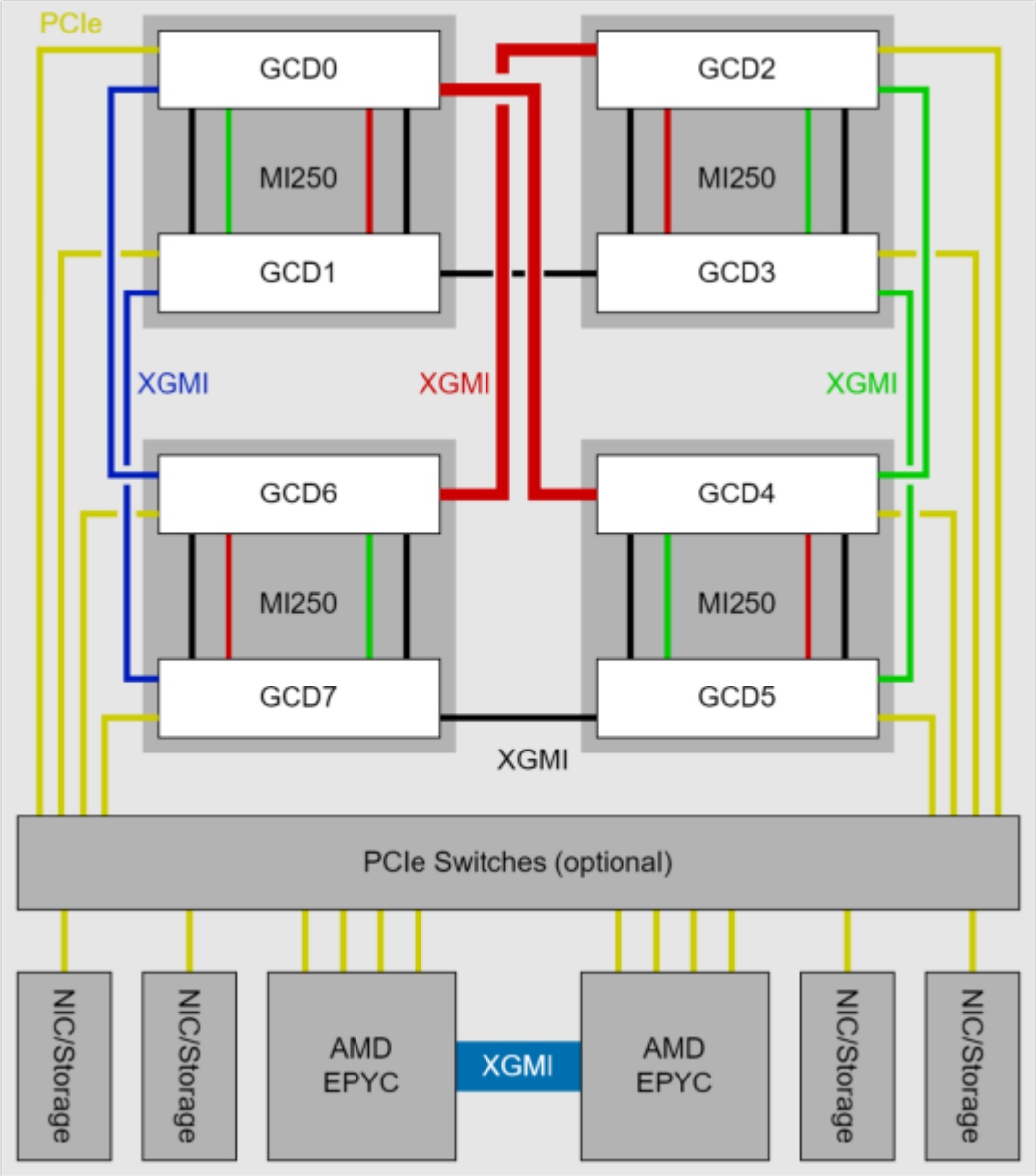


Fig. 26.3: Block diagram of AMD Instinct MI250 Accelerators with 3rd Generation AMD EPYC processor.

26.5 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.

26.5.1 System Architecture

Fig. 26.4 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.

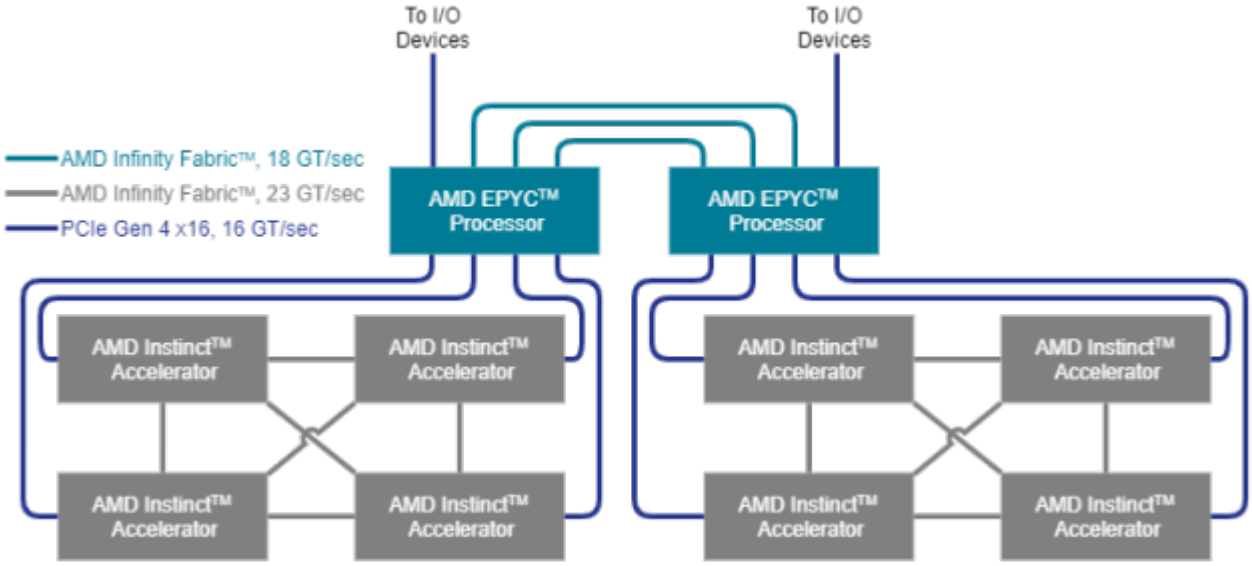


Fig. 26.4: Structure of a single GCD in the AMD Instinct MI250 accelerator.

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.

26.5.2 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. 26.5 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. 26.4.

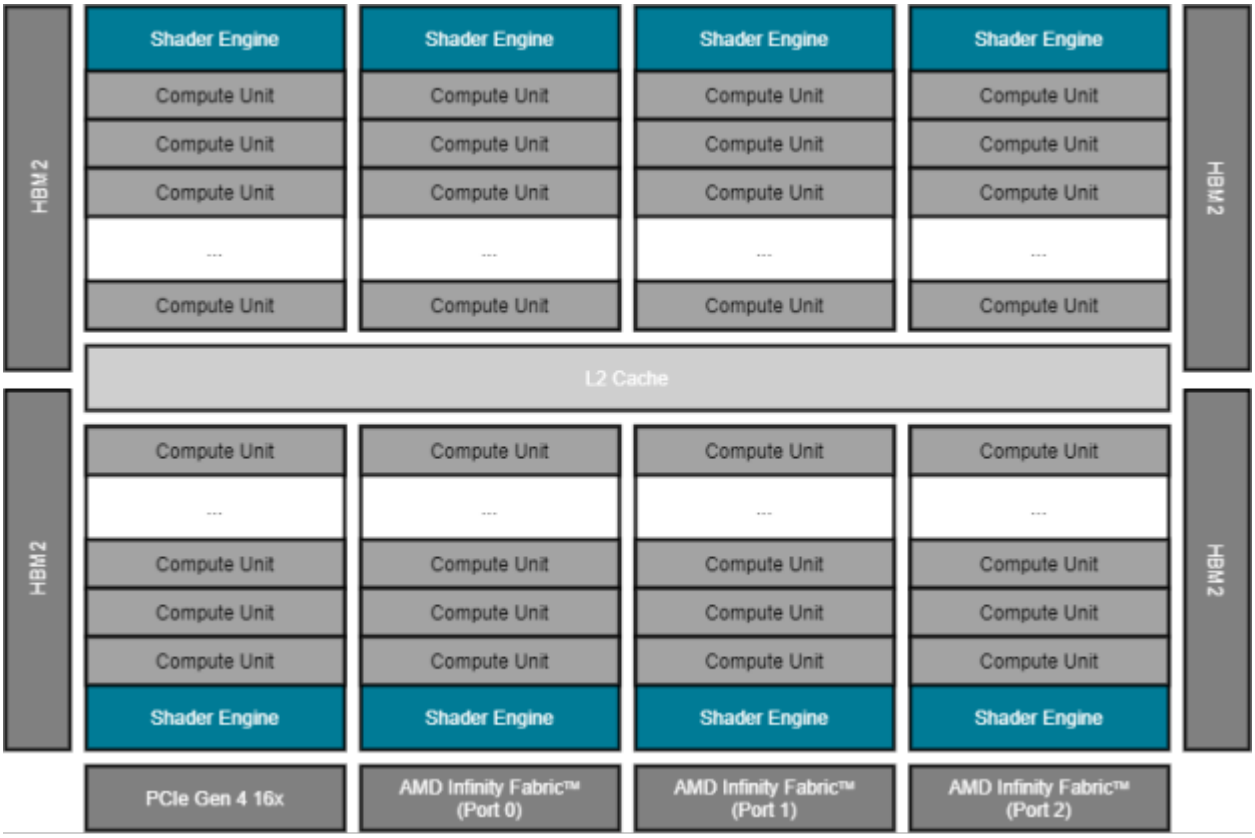


Fig. 26.5: Structure of the AMD Instinct accelerator (MI100 generation).

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. 26.5 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]`).

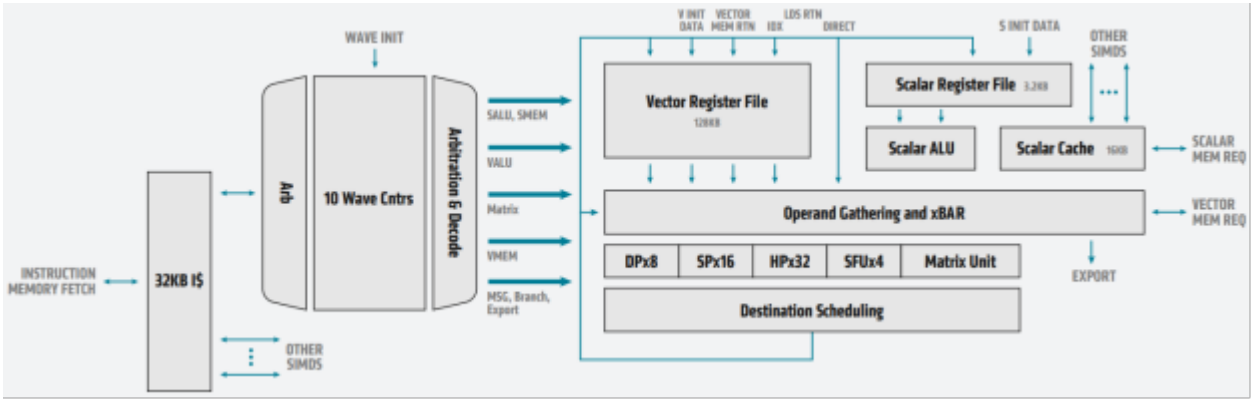


Fig. 26.6: Block diagram of an MI100 compute unit with detailed SIMD view of the AMD CDNA architecture

Fig. 26.6 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 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 26.2: 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

TWENTYSEVEN

HOW ROCM USES PCIE ATOMICS

27.1 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 AtomicOps operate as completers for `CAS` (Compare and Swap), `FetchADD`, `SWAP` atomics. The AtomicsOps 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 AtomicOp 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 AtomicOp 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 AtomicOp routing capability between two or more Root Ports, each associated Root Port must indicate that capability via the AtomicOp Routing Supported bit in the Device Capabilities 2 register.

If your system has a PCIe Express Switch it needs to support AtomicsOp routing. Again AtomicOp 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 AtomicOp completion and/or routing to a component which does. AtomicOp Routing Support=1 Routing is supported, AtomicOp 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.

Good Paper on Understanding PCIe Generation 3 Throughput

27.1.1 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.

27.2 Excepts form Overview of Changes to PCI Express 3.0

27.2.1 By Mike Jackson, Senior Staff Architect, MindShare, Inc.

27.2.2 Atomic Operations – Goal:

Support SMP-type operations across a PCIe network to allow for things like offloading tasks between CPU cores and accelerators like a GPU. The spec says this enables advanced synchronization mechanisms that are particularly useful with multiple producers or consumers that need to be synchronized in a non-blocking fashion. Three new atomic non-posted requests were added, plus the corresponding completion (the address must be naturally aligned with the operand size or the TLP is malformed):

- Fetch and Add – uses one operand as the "add" value. Reads the target location, adds the operand, and then writes the result back to the original location.
- Unconditional Swap – uses one operand as the "swap" value. Reads the target location and then writes the swap value to it.
- Compare and Swap – uses 2 operands: first data is compare value, second is swap value. Reads the target location, checks it against the compare value and, if equal, writes the swap value to the target location.
- AtomicOpCompletion – new completion to give the result so far atomic request and indicate that the atomicity of the transaction has been maintained.

Since AtomicOps are not locked they don't have the performance downsides of the PCI locked protocol. Compared to locked cycles, they provide "lower latency, higher scalability, advanced synchronization algorithms, and dramatically lower impact on other PCIe traffic." The lock mechanism can still be used across a bridge to PCI or PCI-X to achieve the desired operation.

AtomicOps can go from device to device, device to host, or host to device. Each completer indicates whether it supports this capability and guarantees atomic access if it does. The ability to route AtomicOps is also indicated in the registers for a given port.

27.2.3 ID-based Ordering – Goal:

Improve performance by avoiding stalls caused by ordering rules. For example, posted writes are never normally allowed to pass each other in a queue, but if they are requested by different functions, we can have some confidence that the requests are not dependent on each other. The previously reserved Attribute bit [2] is now combined with the RO bit to indicate ID ordering with or without relaxed ordering.

This only has meaning for memory requests, and is reserved for Configuration or IO requests. Completers are not required to copy this bit into a completion, and only use the bit if their enable bit is set for this operation.

To read more on PCIe Gen 3 new options https://www.mindshare.com/files/resources/PCIe%203-0.pdf

CHAPTER

TWENTYEIGHT

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

TWENTYNINE

TUNING GUIDES

Use case-specific system setup and tuning guides.

29.1 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*

29.2 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*

29.3 MI200 High Performance Computing and Tuning Guide

29.3.1 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)

29.3.1.1 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 29.1.

Table 29.1: Recommended settings
platform.

BIOS Setting Location	Param
Advanced / PCI Subsystem Settings	Above
Advanced / PCI Subsystem Settings	SR-IOV
AMD CBS / CPU Common Options	Global
AMD CBS / CPU Common Options	CCD/C
AMD CBS / CPU Common Options / Performance	SMT C
AMD CBS / DF Common Options / Memory Addressing	NUMA
AMD CBS / DF Common Options / Memory Addressing	Memor
AMD CBS / DF Common Options / Link	4-link x
AMD CBS / NBIO Common Options	IOMM
AMD CBS / NBIO Common Options	PCIe T
AMD CBS / NBIO Common Options	Preferr
AMD CBS / NBIO Common Options	Preferr
AMD CBS / NBIO Common Options	Enhanc
AMD CBS / NBIO Common Options / SMU Common Options	Determ
AMD CBS / NBIO Common Options / SMU Common Options	Determ
AMD CBS / NBIO Common Options / SMU Common Options	cTDP C
AMD CBS / NBIO Common Options / SMU Common Options	cTDP
AMD CBS / NBIO Common Options / SMU Common Options	Packag
AMD CBS / NBIO Common Options / SMU Common Options	Packag
AMD CBS / NBIO Common Options / SMU Common Options	xGMI l
AMD CBS / NBIO Common Options / SMU Common Options	xGMI l
AMD CBS / NBIO Common Options / SMU Common Options	xGMI l
AMD CBS / NBIO Common Options / SMU Common Options	APBDI
AMD CBS / NBIO Common Options / SMU Common Options	DF C-s
AMD CBS / NBIO Common Options / SMU Common Options	Fixed S
AMD CBS / UMC Common Options / DDR4 Common Options	Enforce
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR	Overclo
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR	Memor
AMD CBS / UMC Common Options / DDR4 Common Options / DRAM Controller Configuration / DRAM Power Options	Power l
AMD CBS / Security	TSME

29.3.1.1.1 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.

29.3.1.1.2 Memory Configuration

For setting the memory addressing modes (see Table 29.1), 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.

29.3.1.2 Operating System Settings

29.3.1.2.1 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 15

```
sudo zypper install cpupower
```

29.3.1.2.2 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.

29.3.1.2.3 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.

29.3.2 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*.

29.3.2.1 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. 29.1: `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 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*.

```
$ 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. 29.2: `rocm-smi --showtopo` output on an 8*MI200 system.

```
$ 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. 29.3: `rocminfo` output fragment on an 8*MI200 system.

29.3.2.2 Testing Inter-device Bandwidth

Section 29.4.2.1 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 15

```
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:

```
Device: 0,  AMD EPYC 7763 64-Core Processor
Device: 1,  AMD EPYC 7763 64-Core Processor
Device: 2,  AMD EPYC 7763 64-Core Processor
Device: 3,  AMD EPYC 7763 64-Core Processor
Device: 4,  AMD EPYC 7763 64-Core Processor
Device: 5,  AMD EPYC 7763 64-Core Processor
Device: 6,  AMD EPYC 7763 64-Core Processor
Device: 7,  AMD EPYC 7763 64-Core Processor
Device: 8,  ,  GPU-114e3ae64bb25943,  29:0.0
Device: 9,  ,  GPU-572de0da66b3c95b,  2c:0.0
Device: 10,  ,  GPU-1ae331783fe7cf3a,  2f:0.0
Device: 11,  ,  GPU-c9b1396f2ed7e685,  32:0.0
Device: 12,  ,  GPU-e3818bdd5d20905a,  ad:0.0
Device: 13,  ,  GPU-d3ff1f4135b58b49,  b0:0.0
Device: 14,  ,  GPU-fd391d327412f64c,  b3:0.0
Device: 15,  ,  GPU-a772e0db7c5036aa,  b6:0.0
```

Fig. 29.4: `rocm-bandwidth-test` output fragment on an 8*MI200 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	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. 29.5: `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. 29.6: `rocm-bandwidth-test` output fragment on an 8*MI200 system showing uni- and bidirectional bandwidths.

29.4 MI100 High Performance Computing and Tuning Guide

29.4.1 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)

29.4.1.1 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 29.2.

Table 29.2: Recommended settings for the sy
platform.

BIOS Setting Location	Param
Advanced / PCI Subsystem Settings	Above
AMD CBS / CPU Common Options	Global
AMD CBS / CPU Common Options	CCD/C
AMD CBS / CPU Common Options / Performance	SMT C
AMD CBS / DF Common Options / Memory Addressing	NUMA
AMD CBS / DF Common Options / Memory Addressing	Memor
AMD CBS / DF Common Options / Link	4-link x
AMD CBS / DF Common Options / Link	3-link x
AMD CBS / NBIO Common Options	IOMM
AMD CBS / NBIO Common Options	PCIe T
AMD CBS / NBIO Common Options	Preferr
AMD CBS / NBIO Common Options	Preferr
AMD CBS / NBIO Common Options	Enhanc
AMD CBS / NBIO Common Options / SMU Common Options	Determ
AMD CBS / NBIO Common Options / SMU Common Options	Determ
AMD CBS / NBIO Common Options / SMU Common Options	cTDP C
AMD CBS / NBIO Common Options / SMU Common Options	cTDP
AMD CBS / NBIO Common Options / SMU Common Options	Packag
AMD CBS / NBIO Common Options / SMU Common Options	Packag
AMD CBS / NBIO Common Options / SMU Common Options	xGMI l
AMD CBS / NBIO Common Options / SMU Common Options	xGMI l
AMD CBS / NBIO Common Options / SMU Common Options	xGMI l

Table 29.2 – continued from

BIOS Setting Location	Param
AMD CBS / NBIO Common Options / SMU Common Options	APBDI
AMD CBS / NBIO Common Options / SMU Common Options	DF C-s
AMD CBS / NBIO Common Options / SMU Common Options	Fixed S
AMD CBS / UMC Common Options / DDR4 Common Options	Enforce
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR	Overclo
AMD CBS / UMC Common Options / DDR4 Common Options / Enforce POR	Memor
AMD CBS / UMC Common Options / DDR4 Common Options / DRAM Controller Configuration / DRAM Power Options	Power
AMD CBS / Security	TSME

29.4.1.1.1 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.

29.4.1.1.2 Memory Configuration

For the memory addressing modes (see Table 29.2), 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.

29.4.1.2 Operating System Settings

29.4.1.2.1 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 15

```
sudo zypper install cpupower
```

29.4.1.2.2 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.

29.4.1.2.3 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.

29.4.2 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*.

29.4.2.1 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:

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.

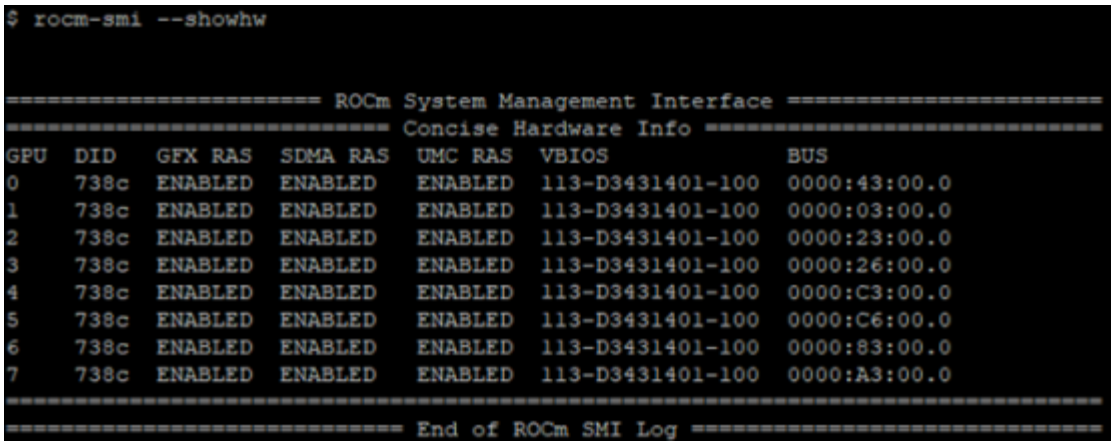


Fig. 29.7: `rocm-smi --showhw` output on an 8*MI100 system.

- 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*.

29.4.2.2 Testing Inter-device Bandwidth

Section 29.4.2.1 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.

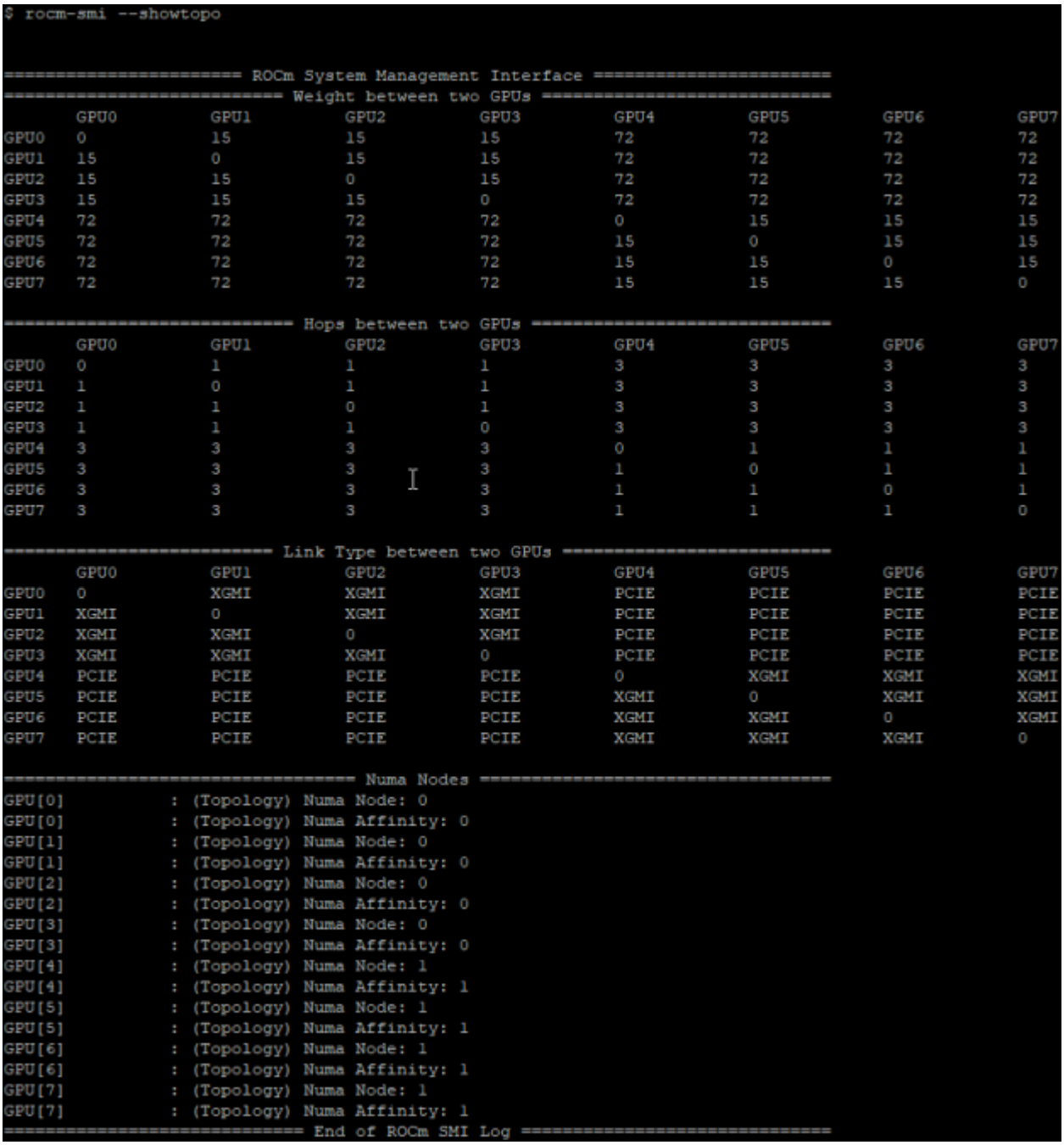
The ROCm Bandwidth Test program can be installed with the following package-manager commands:

Ubuntu

```
sudo apt install rocm-bandwidth-test
```

Red Hat Enterprise Linux

```
sudo yum install rocm-bandwidth-test
```



```
$ 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. 29.8: `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. 29.9: `rocminfo` output fragment on an 8*MI100 system.

SUSE Linux Enterprise Server 15

```
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. 29.10: `rocm-bandwidth-test` output fragment on an 8*MI100 system listing devices.

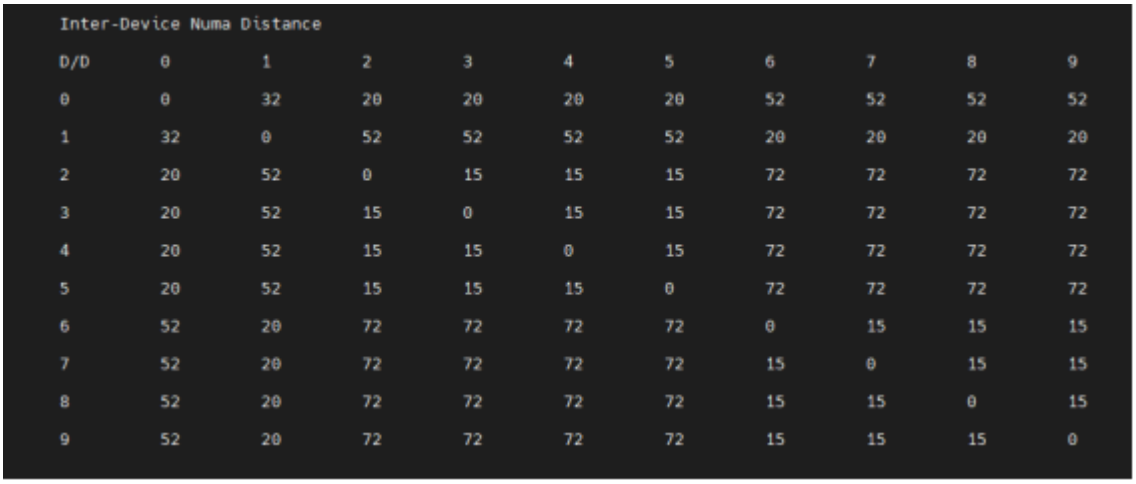
The output will also show a matrix that contains a "1" if a device can communicate to another device (CPU and GPU) of the system and it will show the NUMA distance (similar to `rocm-smi`):

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. 29.11: `rocm-bandwidth-test` output fragment on an 8*MI100 system showing inter-device access matrix.

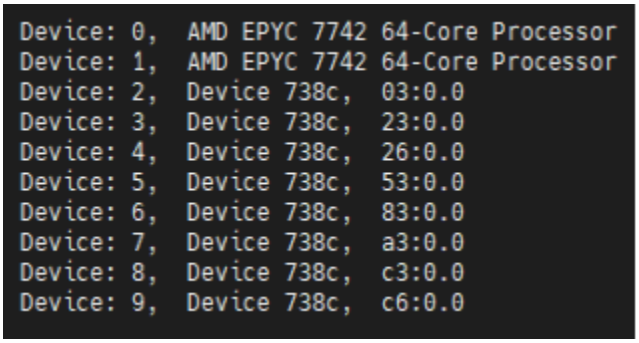
The output also contains the measured bandwidth for unidirectional and bidirectional transfers between the devices (CPU and GPU):



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. 29.12: `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. 29.13: `rocm-bandwidth-test` output fragment on an 8*MI100 system showing uni- and bidirectional bandwidths.

29.5 RDNA2 Workstation Tuning Guide

29.5.1 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.

29.5.1.1 System BIOS Settings

Table 29.3: 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.

29.5.1.2 Operating System Settings

Table 29.4: 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
```

29.5.1.3 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.

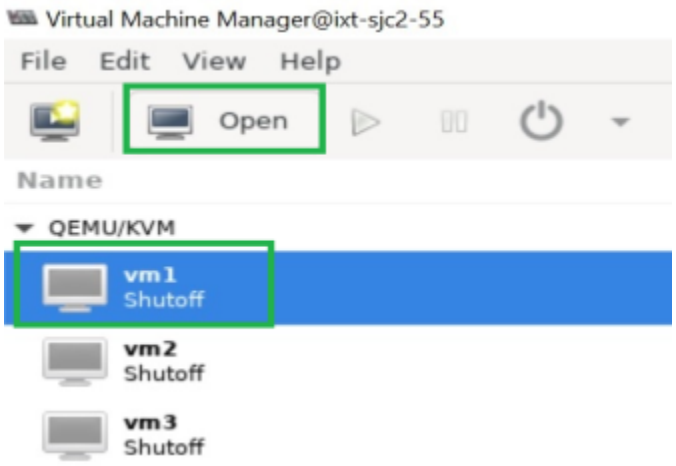


Fig. 29.14: Virtual Machine Manager

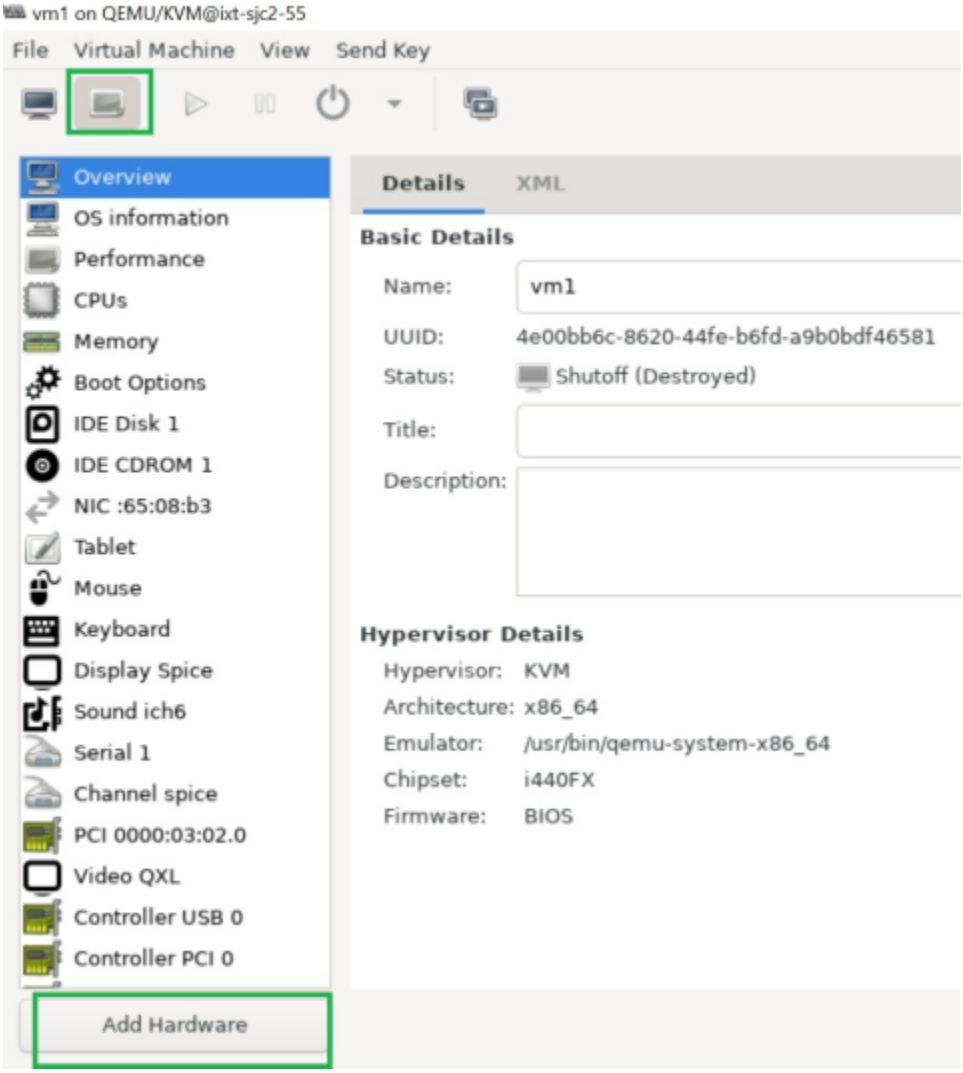


Fig. 29.15: Virtual Machine Manager

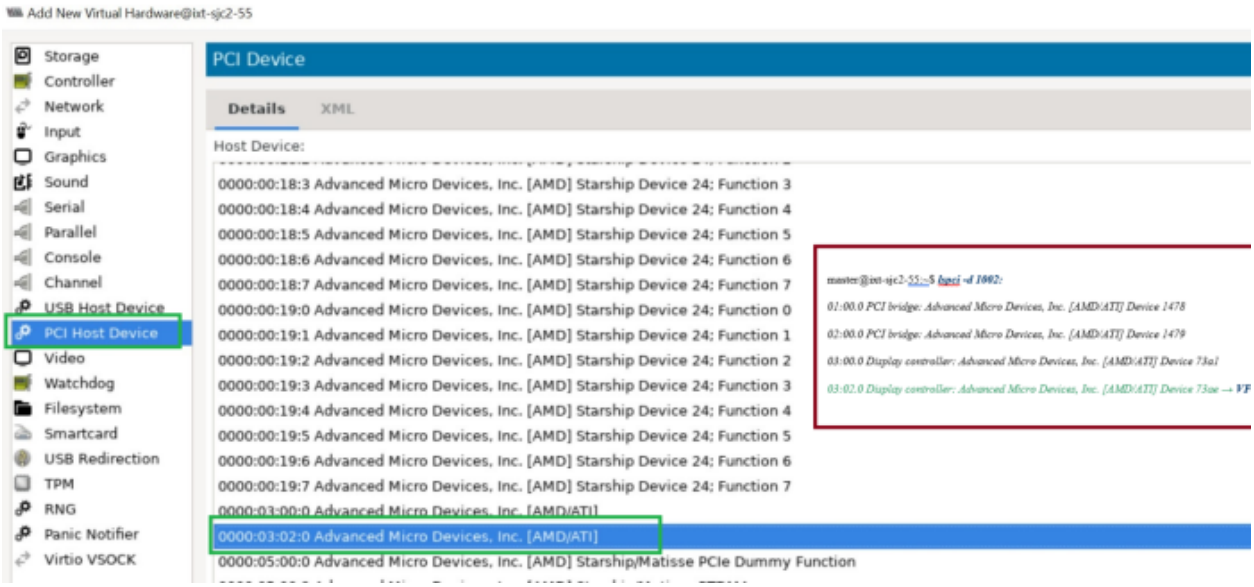


Fig. 29.16: VF Selection

CHAPTER

THIRTY

DEEP LEARNING GUIDE

The following sections cover the different framework installations for ROCm and Deep Learning applications. Fig. 30.1 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*.

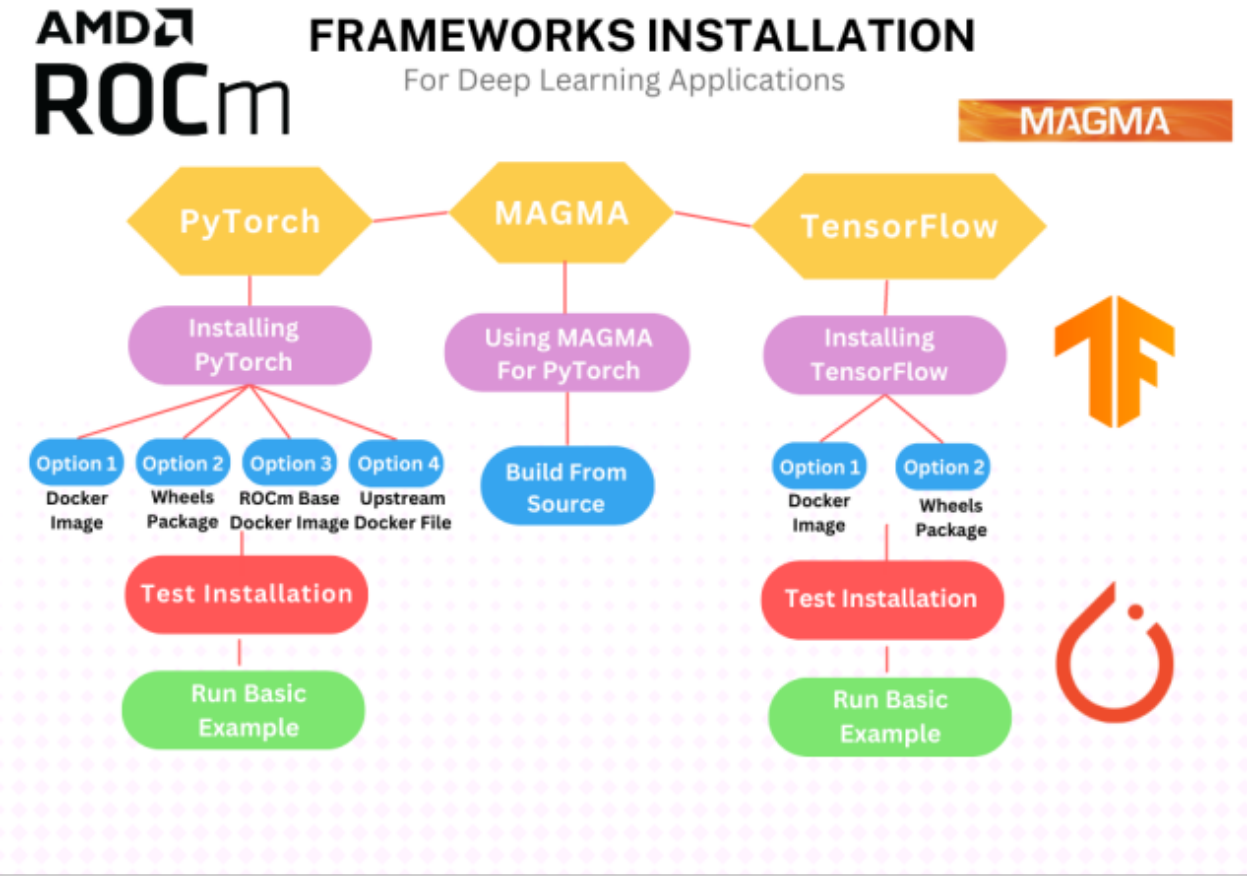


Fig. 30.1: ROCm Compatible Frameworks Flowchart

30.1 Frameworks Installation

- *How to Install PyTorch?*
- *How to Install Tensorflow?*
- *How to Install Magma?*

30.2 Magma Installation for ROCm

30.2.1 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.

30.2.1.1 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.

30.2.1.2 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
```

(continues on next page)

(continued from previous page)

```
  amdgpu_targets=`echo $PYTORCH_ROCM_ARCH | sed 's/;/ /g'`
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
```

30.2.2 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

30.3 PyTorch Installation for ROCm

30.3.1 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

30.3.1.1 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.

30.3.1.1.1 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.

30.3.1.1.2 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. 30.2 is a matrix from http://pytorch.org/ that illustrates the installation compatibility between ROCm and the PyTorch build.

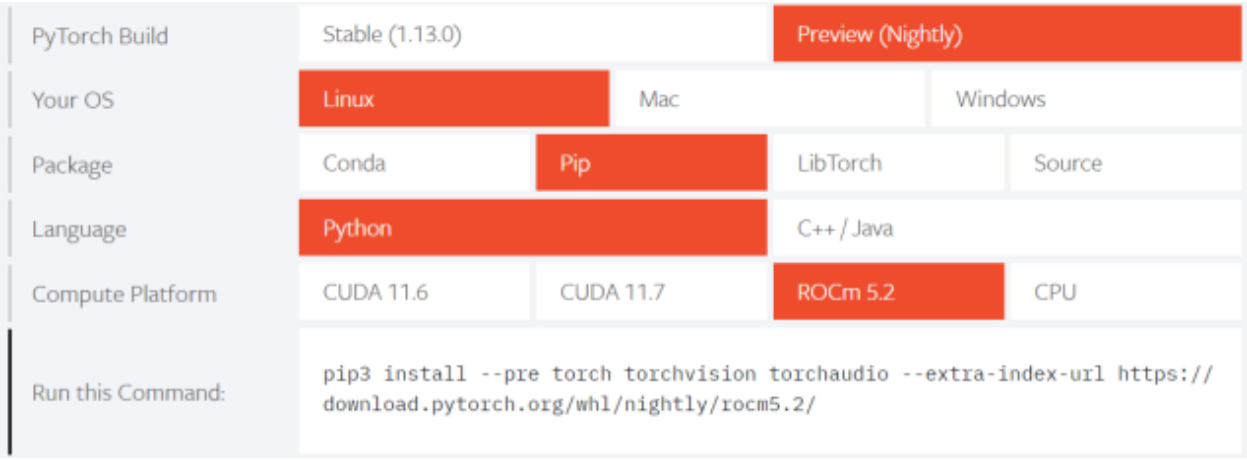


Fig. 30.2: Installation Matrix from Pytorch

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 http://pytorch.org/.

   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/
```

30.3.1.1.3 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
```

30.3.1.1.4 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
```

30.3.1.2 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 DockerHub 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.

30.3.1.3 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
```

30.3.2 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

30.4 TensorFlow Installation for ROCm

30.4.1 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.

30.4.1.1 Installing TensorFlow

The following sections contain options for installing TensorFlow.

30.4.1.1.1 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
```

30.4.1.1.2 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

30.4.1.2 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'
```

30.4.1.3 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
```

30.4.2 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

THIRTYONE

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.

31.1 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.
```

31.2 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 $BUILD_DIR
git clone https://github.com/openucx/ucx.git -b v1.14.1
cd ucx
./autogen.sh
mkdir build
cd build
../contrib/configure-release -prefix=$UCX_DIR \
    --with-rocm=/opt/rocm \
    --without-cuda -enable-optimizations -disable-logging \
    --disable-debug -disable-assertions \
    --disable-params-check -without-java
make -j $(nproc)
make -j $(nproc) install
```

The following *table* documents the compatibility of UCX versions with ROCm versions.

31.3 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 \
    --enable-mca-no-build=btl-uct --enable-mpi1-compatibility \
    CC=clang CXX=clang++ FC=flang
make -j $(nproc)
make -j $(nproc) install
```

31.4 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
```

(continues on next page)

(continued from previous page)

```
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)
```

31.5 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:

31.6 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. 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
```

(continues on next page)

```
# 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. 31.1: Inter-GPU bandwidth with various payload sizes.

(continued from previous page)

```
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
```

CHAPTER

THIRTYTWO

SYSTEM DEBUGGING GUIDE

32.1 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`

32.2 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

32.3 Command to Dump Firmware Version and Get Linux Kernel Version

```
sudo cat /sys/kernel/debug/dri/1/amdgpu_firmware_info
```

```
uname -a
```

32.4 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.

32.5 ROCr Level Environment Variables for Debug

```
HSA_ENABLE_SDMA=0

HSA_ENABLE_INTERRUPT=0

HSA_SVM_GUARD_PAGES=0

HSA_DISABLE_CACHE=1
```

32.6 Turn Off Page Retry on GFX9/Vega Devices

```
sudo -s

echo 1 > /sys/module/amdkfd/parameters/noretry
```

32.7 HIP Environment Variables 3.x

32.7.1 OpenCL Debug Flags

```
AMD_OCL_WAIT_COMMAND=1 (0 = OFF, 1 = On)
```

32.8 PCIe-Debug

Refer to ROCm PCIe Debug, https://rocmdocs.amd.com/en/latest/Other_Solutions/PCIe-Debug.html#pcie-debug. For information on how to debug and profile HIP applications, see HIP Debugging

CHAPTER

THIRTYTHREE

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.

33.1 Inception V3 with PyTorch

33.1.1 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.

33.1.2 Training Phases

Training occurs in multiple phases for every batch of training data. Table 33.1 provides an explanation of the types of training phases.

Table 33.1: 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 33.2 shows the contrast between training and inference.

Table 33.2: 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.

33.1.3 Case Studies

The following sections contain case studies for the Inception v3 model.

33.1.3.1 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.

33.1.3.1.1 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())
```

33.1.3.1.2 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/
```

(continues on next page)

(continued from previous page)

```
↪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.

[4] Stanford, [Online]. Available: http://cs231n.stanford.edu/

Note: The device is set to "cuda". In PyTorch, "cuda" is a generic keyword to denote a GPU.

```
device = "cuda"
```

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.

[5] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy

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.

```
        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. 33.1.

Fig. 33.1: Inception v3 Train and Loss Graph

33.1.3.2 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):
    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)
```

(continues on next page)

(continued from previous page)

```
        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 254, 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'
```

(continues on next page)

(continued from previous page)

```
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)
        _, 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))
```

33.1.3.3 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:

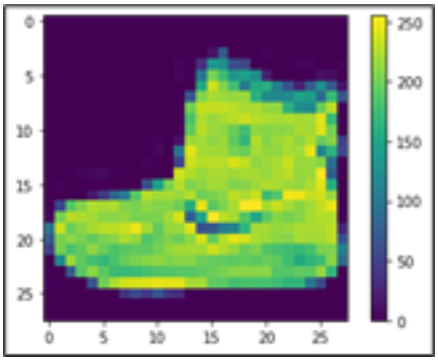
```
print(len(train_labels))
```

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.

```
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.


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

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]
```

(continues on next page)

(continued from previous page)

```
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.

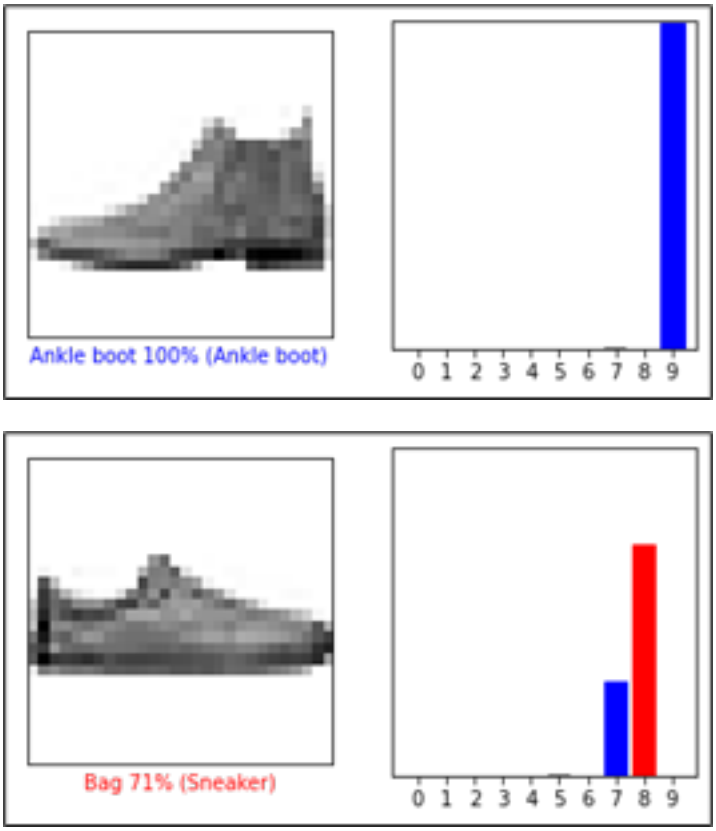
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```

```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
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.
```

(continues on next page)



(continued from previous page)

```
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.

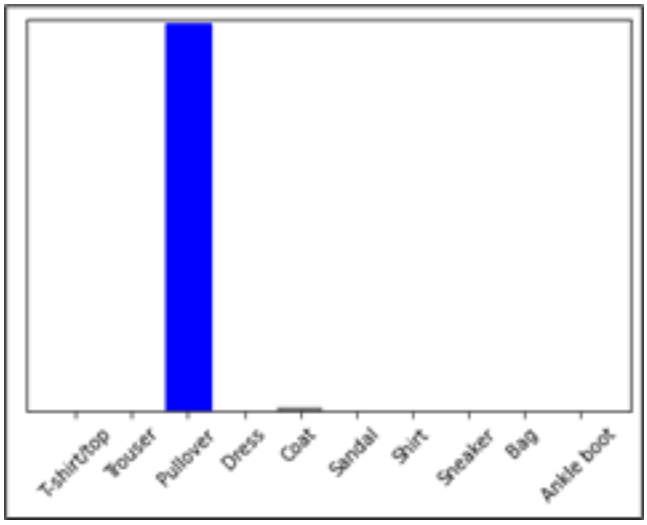
```
predictions_single = probability_model.predict(img)

print(predictions_single)

plot_value_array(1, predictions_single[0], test_labels)
_ = 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])
```



33.1.3.4 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)
```

(continues on next page)

(continued from previous page)

```
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.

```
print("1287 ---> ",vectorize_layer.get_vocabulary()[1287])
print(" 313 ---> ",vectorize_layer.get_vocabulary()[313])
print('Vocabulary size: {}'.format(len(vectorize_layer.get_vocabulary())))
```

```
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
```

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()
```

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: "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
```

```
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. 33.2 and Fig. 33.3 illustrate the training and validation loss and the training and validation accuracy.

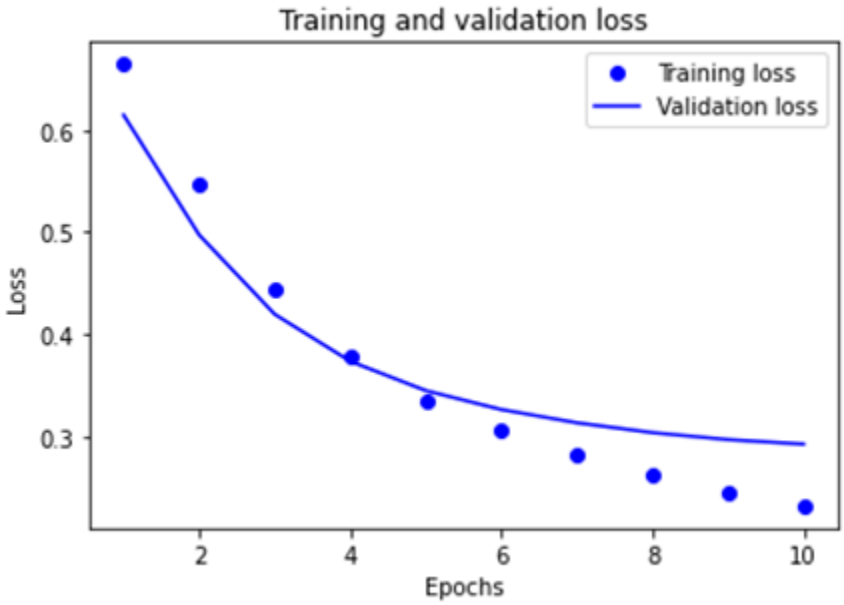


Fig. 33.2: Training and Validation Loss

12. Export the model.

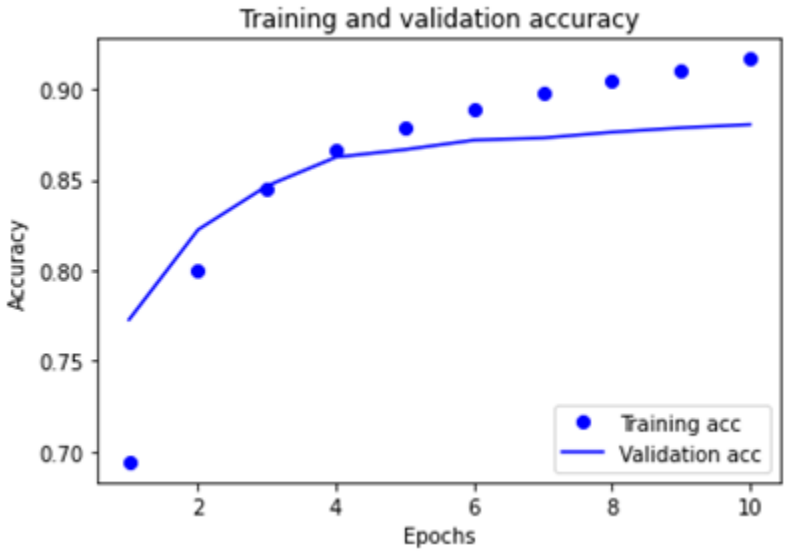


Fig. 33.3: Training and Validation Accuracy

```
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)
```

33.1.4 References

33.2 Inference Optimization with MIGraphX

The following sections cover inferencing and introduces MIGraphX.

33.2.1 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.

33.2.2 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 MIGraphX's 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.

33.2.3 Installing MIGraphX

There are three options to get started with MIGraphX installation. MIGraphX depends on ROCm libraries; assume that the machine has ROCm installed.

33.2.3.1 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.

33.2.3.2 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

33.2.3.3 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*.

33.2.4 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.

33.2.4.1 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.

33.2.5 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 MIGraphX's C++ API. The following is the `CMakeLists.txt` file that can build the earlier example:

```
cmake_minimum_required(VERSION 3.5)
project (CAI)

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.
```

33.2.6 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
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
```

(continues on next page)

(continued from previous page)

```
Inference time: 0.004ms
iterator : 7
Inference complete
Inference time: 0.004ms
iterator : 8
Inference complete
Inference time: 0.004ms
iterator : 9
Inference complete
Inference time: 0.004ms
```

33.2.6.1 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.

33.2.6.1.1 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 MIGraphX's C++ or Python API to generate `.mxr` file. Refer to Fig. 33.4 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. 33.4: Generating a `.mxr` File

CHAPTER

THIRTYFOUR

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.

34.1 ReadTheDocs

ReadTheDocs is our front end for the our documentation. By front end, this is the tool that serves our HTML based documentation to our end users.

34.2 Doxygen

Doxygen is the most common inline code documentation standard. ROCm projects are use Doxygen for public API documentation (unless the upstream project is using a different tool).

34.3 Sphinx

Sphinx is a documentation generator originally used for python. It is now widely used in the Open Source community. Originally, sphinx supported RST based documentation. 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.

34.3.1 MyST

Markedly Structured Text (MyST) is an extended flavor of Markdown (CommonMark) influenced by reStructuredText (RST) and Sphinx. It is integrated via `myst-parser`. A cheat sheet that showcases how to use the MyST syntax is available over at the Jupyter reference.

34.3.2 Sphinx Theme

ROCm is using the Sphinx Book Theme. This theme is used by Jupyter books. ROCm documentation applies some customization include a header and footer on top of the Sphinx Book Theme. A future custom ROCm theme will be part of our documentation goals.

34.3.3 Sphinx Design

Sphinx Design is an extension for sphinx based websites that add design functionality. Please see the documentation here. ROCm documentation uses sphinx design for grids, cards, and synchronized tabs. Other features may be used in the future.

34.3.4 Sphinx External TOC

ROCm uses the sphinx-external-toc for our navigation. This tool allows a YAML file based left navigation menu. This tool 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.

34.3.5 Breathe

Sphinx uses Breathe to integrate Doxygen content.

34.4 `rocm-docs-core` pip package

rocm-docs-core is an AMD maintained project that applies customization for our documentation. This project is the tool most ROCm repositories will use as part of the documentation build.

CHAPTER

THIRTYFIVE

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 recommend 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.

35.1 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 will render 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 will use breathe syntax for doxygen documentation in our markdown files. Other design elements for effective HTML rendering of the documents may be added to our markdown files. Please 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).

35.2 Filenames and folder structure

Please use snake case for file names. 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 place in the /docs/data folder.

35.3 How to provide feedback for for ROCm documentation

There are three standard ways to provide feedback for this repository.

35.3.1 Pull Request

All contributions to ROCm documentation should arrive via the GitHub Flow targetting the develop branch of the repository. If you are unable to contribute via the GitHub Flow, feel free to email us. TODO, confirm email address.

35.3.2 GitHub Issue

Issues on existing or absent docs can be filed as GitHub issues .

35.3.3 Email Feedback

35.4 Language and Style

Adopting 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, sphinx extention. Please raise issues in rocm-docs-core for any formatting concerns and changes requested.

35.5 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.

35.5.1 Command line documentation builds

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
# 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.

35.5.2 Pull Requests 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 `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>/`.

35.5.3 Build the docs using 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 serving web server.

35.5.3.1 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 in order are set for the following reasons:
   - Sets the root of the output website for live previews. Must be changed alongside the `tasks.json` command.
   - Tells live server to wait with the update to give time for Sphinx to regenerate site contents and not refresh before all is don. (Empirical value)
   - Automatic virtual env 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",
```

(continues on next page)

(continued from previous page)

```
      "type": "process",
      "windows": {
        "command": "${workspaceFolder}/.venv/Scripts/python.exe"
      },
      "command": "${workspaceFolder}/.venv/bin/python3",
      "args": [
        "-m",
        "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 dropdown 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.

INDEX

Symbols

C

O