AMD ROCm™ Release Notes v4.1

Revision	**0323**
Issue Date:	**March 2021**

Specification Agreement

This Specification Agreement (this "Agreement") is a legal agreement between Advanced Micro Devices, Inc. ("AMD") and "You" as the recipient of the attached AMD Specification (the "Specification"). If you are accessing the Specification as part of your performance of work for another party, you acknowledge that you have authority to bind such party to the terms and conditions of this Agreement. If you accessed the Specification by any means or otherwise use or provide Feedback (defined below) on the Specification, You agree to the terms and conditions set forth in this Agreement. If You do not agree to the terms and conditions set forth in this Agreement, you are not licensed to use the Specification; do not use, access or provide Feedback about the Specification. In consideration of Your use or access of the Specification (in whole or in part), the receipt and sufficiency of which are acknowledged, You agree as follows:

1. You may review the Specification only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with an AMD product in compliance with the requirements as set forth in the Specification and (b) to provide Feedback about the information disclosed in the Specification to AMD.

2. Except as expressly set forth in Paragraph 1, all rights in and to the Specification are retained by AMD. This Agreement does not give You any rights under any AMD patents, copyrights, trademarks or other intellectual property rights. You may not (i) duplicate any part of the Specification; (ii) remove this Agreement or any notices from the Specification, or (iii) give any part of the Specification, or assign or otherwise provide Your rights under this Agreement, to anyone else.

3. The Specification may contain preliminary information, errors, or inaccuracies, or may not include certain necessary information. Additionally, AMD reserves the right to discontinue or make changes to the Specification and its products at any time without notice. The Specification is provided entirely "AS IS." AMD MAKES NO WARRANTY OF ANY KIND AND DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, TITLE OR THOSE WARRANTIES ARISING AS A COURSE OF DEALING OR CUSTOM OF TRADE. AMD SHALL NOT BE LIABLE FOR DIRECT, INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE OR EXEMPLARY DAMAGES OF ANY KIND (INCLUDING LOSS OF BUSINESS, LOSS OF INFORMATION OR DATA, LOST PROFITS, LOSS OF CAPITAL, LOSS OF GOODWILL) REGARDLESS OF THE FORM OF ACTION WHETHER IN CONTRACT, TORT (INCLUDING NEGLIGENCE) AND STRICT PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4. Furthermore, AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur.

5. You have no obligation to give AMD any suggestions, comments or feedback ("Feedback") relating to the Specification. However, any Feedback You voluntarily provide may be used by AMD without restriction, fee or obligation of confidentiality. Accordingly, if You do give AMD Feedback on any version of the Specification, You agree AMD may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any product, as well as has the right to sublicense third parties to do the same. Further, You will not give AMD any Feedback that You may have reason to believe is (i) subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any product or intellectual

property incorporating or derived from Feedback or any Product or other AMD intellectual property to be licensed to or otherwise provided to any third party.

6. You shall adhere to all applicable U.S. import/export laws and regulations, as well as the import/export control laws and regulations of other countries as applicable. You further agree to not export, re-export, or transfer, directly or indirectly, any product, technical data, software or source code received from AMD under this license, or the direct product of such technical data or software to any country for which the United States or any other applicable government requires an export license or other governmental approval without first obtaining such licenses or approvals; or in violation of any applicable laws or regulations of the United States or the country where the technical data or software was obtained. You acknowledge the technical data and software received will not, in the absence of authorization from U.S. or local law and regulations as applicable, be used by or exported, re-exported or transferred to: (i) any sanctioned or embargoed country, or to nationals or residents of such countries; (ii) any restricted end-user as identified on any applicable government end-user list; or (iii) any party where the end-use involves nuclear, chemical/biological weapons, rocket systems, or unmanned air vehicles. For the most current Country Group listings, or for additional information about the EAR or Your obligations under those regulations, please refer to the U.S. Bureau of Industry and Security's website at *http://www.bis.doc.gov/*.

7. The Software and related documentation are "commercial items", as that term is defined at 48 C.F.R. §2.101, consisting of "commercial computer software" and "commercial computer software documentation", as such terms are used in 48 C.F.R. §12.212 and 48 C.F.R. §227.7202, respectively. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §227.7202-1 through 227.7202-4, as applicable, the commercial computer software and commercial computer software documentation are being licensed to U.S. Government end users (a) only as commercial items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions set forth in this Agreement. Unpublished rights are reserved under the copyright laws of the United States.

8. This Agreement is governed by the laws of the State of California without regard to its choice of law principles. Any dispute involving it must be brought in a court having jurisdiction of such dispute in Santa Clara County, California, and You waive any defenses and rights allowing the dispute to be litigated elsewhere. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. The failure of AMD to enforce any rights granted hereunder or to take action against You in the event of any breach hereunder shall not be deemed a waiver by AMD as to subsequent enforcement of rights or subsequent actions in the event of future breaches. This Agreement is the entire agreement between You and AMD concerning the Specification; it may be changed only by a written document signed by both You and an authorized representative of AMD.

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. In addition, any stated support is planned and is also subject to change. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

[This page is left blank intentionally]

Table of Contents

Table of Contents **5**

ROCm Installation Updates **7**

List of Supported Operating Systems 7

Fresh Installation of AMD ROCm v4.1 Recommended 7

ROCm Multi-Version Installation Update 8

Driver Compatibility Issue in ROCm v4.1 8

Bare Metal 9

Containers 9

AMD ROCm V4.1 Documentation Updates **10**

AMD ROCm Installation Guide 10

HIP Documentation Updates 10

ROCm Data Center Tool User and API Guide 10

ROCm SMI API Guide 11

ROC Debugger User and API Guide 11

AMD ROCm General Documentation Links 11

What's New in This Release **12**

TargetID For Multiple Configurations 12

New Code Object Format Version for TargetID 12

New Code Object Tools 13

ROCm Data Center Tool 14

Grafana Integration 14

ROCm Math and Communication Libraries 14

HIP Enhancements 15

Support for hipEventDisableTiming Flag 15

Cooperative Group Functions 15

Support for Extern Shared Declarations 17
OpenMP Enhancements and Fixes 17
Usability Enhancements 17
Fixes to Internal Clang Math Headers 18
OpenMP Defect Fixes 18
MIOpen-Tensile Integration 19
Known Issues in This Release 19
Upgrade to AMD ROCm v4.1 Not Supported 19
Performance Impact for Kernel Launch Bound Attribute 19
Issue with Passing a Subset of GPUs in a Multi-GPU System 20
Performance Impact for LDS-bound Kernels 20
Deprecations in This Release 20
Compiler-Generated Code Object Version 2 Deprecation 20
Hardware and Software Support 21
Hardware Support 21
Supported Graphics Processing Units 21
Supported CPUs 23
Not supported or limited support under ROCm 24
ROCm Support in Upstream Linux Kernels 26

ROCm Installation Updates

This document describes the features, fixed issues, and information about downloading and installing the AMD ROCm™ software.

It also covers known issues and deprecations in the AMD ROCm v4.1 release.

LIST OF SUPPORTED OPERATING SYSTEMS

The AMD ROCm platform is designed to support the following operating systems:

- Ubuntu 20.04.1 (5.4 and 5.6-oem) and 18.04.5 (Kernel 5.4)
- CentOS 7.9 (3.10.0-1127) & RHEL 7.9 (3.10.0-1160.6.1.el7) (Using devtoolset-7 runtime support)
- CentOS 8.3 (4.18.0-193.el8) and RHEL 8.3 (4.18.0-193.1.1.el8) (devtoolset is not required)
- SLES 15 SP2

FRESH INSTALLATION OF AMD ROCM V4.1 RECOMMENDED

A complete uninstallation of previous ROCm versions is required before installing a new version of ROCm. An upgrade from previous releases to AMD ROCm v4.1 is not supported. For more information, refer to the *AMD ROCm Installation Guide.*

Note: AMD ROCm release v3.3 or prior releases are not fully compatible with AMD ROCm v3.5 and higher versions. You must perform a fresh ROCm installation if you want to upgrade from AMD ROCm v3.3 or older to 3.5 or higher versions and vice-versa.

Note: *render group* is required only for Ubuntu v20.04. For all other ROCm supported operating systems, continue to use *video group*.

- For ROCm v3.5 and releases thereafter, the *clinfo* path is changed to */opt/rocm/opencl/bin/clinfo*.
- For ROCm v3.3 and older releases, the *clinfo* path remains */opt/rocm/opencl/bin/x86_64/clinfo*.

ROCM MULTI-VERSION INSTALLATION UPDATE

With the AMD ROCm v4.1 release, the following ROCm multi-version installation changes apply:

The meta packages *rocm-dkms<version>* are now deprecated for multi-version ROCm installs. For example, *rocm-dkms3.7.0, rocm-dkms3.8.0*.

- Multi-version installation of ROCm should be performed by installing *rocm-dev<version>* using each of the desired ROCm versions. For example, *rocm-dev3.7.0, rocm-dev3.8.0, rocm-dev3.9.0.*
- Version files must be created for each multi-version rocm <= 4.1.0
    - **Command:** *echo <version> | sudo tee /opt/rocm-<version>/.info/version*
    - **Example:** *echo 4.1.0 | sudo tee /opt/rocm-4.1.0/.info/version*
- The rock-dkms loadable kernel modules should be installed using a single *rock-dkms* package.
- ROCm v3.9 and above will not set any *ldconfig* entries for ROCm libraries for multi-version installation. Users must set *LD_LIBRARY_PATH* to load the ROCm library version of choice.

NOTE: The single version installation of the ROCm stack remains the same. The *rocm-dkms* package can be used for single version installs and is not deprecated at this time.

DRIVER COMPATIBILITY ISSUE IN ROCM V4.1

In certain scenarios, the ROCm 4.1 run-time and userspace environment are not compatible with ROCm v4.0 and older driver implementations for 7nm-based (Vega 20) hardware (MI50 and MI60).

To mitigate issues, the ROCm v4.1 or newer userspace prevents running older drivers for these GPUs.

Users are notified in the following scenarios:

- Bare Metal
- Containers

Bare Metal

In the bare-metal environment, the following error message displays in the console:

"HSA Error: Incompatible kernel and userspace, Vega 20 disabled. Upgrade amdgpu."

To test the compatibility, run the ROCm v4.1 version of rocminfo using the following instruction:

/opt/rocm-4.1.0/bin/rocminfo 2>&1 | less

Containers

A container (built with error detection for this issue) using a ROCm v4.1 or newer run-time is initiated to execute on an older kernel. The container fails to start and the following warning appears:

Error: Incompatible ROCm environment. The Docker container requires the latest kernel driver to operate correctly.

Upgrade the ROCm kernel to v4.1 or newer, or use a container tagged for v4.0.1 or older.

To inspect the version of the installed kernel driver, run either:

- *dpkg --status rock-dkms* [Debian-based]

or

- *rpm -ql rock-dkms* [RHEL, SUSE, and others]

To install or update the driver, follow the installation instructions at:

https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation-Guide.html

AMD ROCm V4.1 Documentation Updates

AMD ROCM INSTALLATION GUIDE

The AMD ROCm Installation Guide in this release includes the following updates:

- *Supported Environments*
- *Installation Instructions*
- *HIP Installation Instructions*

HIP DOCUMENTATION UPDATES

- HIP Programming Guide v4.1
  https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Programming_Guide_v4.1.pdf

- HIP API Guide v4.1
  https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_API_Guide_v4.1.pdf

- HIP-Supported CUDA API Reference Guide v4.1
  https://github.com/RadeonOpenCompute/ROCm/blob/master/HIP_Supported_CUDA_API_Reference_Guide_v4.1.pdf

- HIP FAQ
  https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-FAQ.html#hip-faq

ROCM DATA CENTER TOOL USER AND API GUIDE

- ROCm Data Center Tool User Guide

    - Grafana Plugin Integration

  For more information, refer to the ROCm Data Center User Guide at,
  https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_ROCm_DataCenter_Tool_User_Guide_v4.1.pdf

- ROCm Data Center Tool API Guide
  https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm_Data_Center_Tool_API_Manual_4.1pdf

ROCM SMI API GUIDE

- ROCm SMI API Guide
  https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm_SMI_API_GUIDE_v4.1.pdf

ROC DEBUGGER USER AND API GUIDE

- ROC Debugger User Guide
  https://github.com/RadeonOpenCompute/ROCm/blob/master/Debugging%20with%20ROCGDB%20User%20Guide%20v4.1.pdf

- Debugger API Guide
  https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD-Debugger%20API%20Guide%20v4.1.pdf

AMD ROCM GENERAL DOCUMENTATION LINKS

- For AMD ROCm documentation, see
  https://rocmdocs.amd.com/en/latest/

- For installation instructions on supported platforms, see
  https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation-Guide.html

- For AMD ROCm binary structure, see
  https://rocmdocs.amd.com/en/latest/Installation_Guide/Software-Stack-for-AMD-GPU.html

- For AMD ROCm release history, see
  https://rocmdocs.amd.com/en/latest/Current_Release_Notes/ROCm-Version-History.html

What's New in This Release

TARGETID FOR MULTIPLE CONFIGURATIONS

The new TargetID functionality allows compilations to specify various configurations of the supported hardware.

Previously, ROCm supported only a single configuration per target.

With the TargetID enhancement, ROCm supports configurations for Linux, PAL and associated configurations such as XNACK. This feature addresses configurations for the same target in different modes and allows applications to build executables that specify the supported configurations, including the option to be agnostic for the desired setting.

New Code Object Format Version for TargetID

AMD ROCm v4.1 introduces and defaults to a new code object format version 4. The following new options are now available:

- A new clang option -mcode-object-version can be used to request the legacy code object version 3 or code object version 2. For more information, refer to *https://llvm.org/docs/AMDGPUUsage.html#elf-code-object*
- A new clang *--offload-arch=* option is introduced to specify the offload target architecture(s) for the HIP language.
- The clang *--offload-arch=* and *-mcpu* options accept a new Target ID syntax. This allows both the processor and target feature settings to be specified. For more details, refer to *https://llvm.org/docs/AMDGPUUsage.html#amdgpu-target-id*
    - If a target feature is not specified, it defaults to a new concept of "any". The compiler, then, produces code, which executes on a target configured for either value of the setting impacting the overall performance. It is recommended to explicitly specify the setting for more efficient performance.
    - In particular, the setting for XNACK now defaults to produce less performant code than previous ROCm releases.
    - The legacy clang *-mxnack, -mno-xnack, -msram-ecc*, and *-mno-sram-ecc* options are deprecated. They are still supported, however, they will be removed in a future release.
    - The new Target ID syntax renames the SRAM ECC feature from *sram-ecc* to *sramecc*.
- The clang offload bundler uses the new offload *hipv4* for HIP code object version 4. For more information, see *https://clang.llvm.org/docs/ClangOffloadBundler.html*

- ROCm v4.1 corrects code object loading to enforce target feature settings of the code object to match the setting of the agent. It also corrects the recording of target feature settings in the code object. As a consequence, the legacy code objects may no longer load due to mismatches.
- gfx802, gfx803, and gfx805 do not support the XNACK target feature in the ROCm v4.1 release.

New Code Object Tools

AMD ROCm v4.1 provides new code object tools *roc-obj-ls* and *roc-obj-extract*. These tools allow for the listing and extraction of AMD GPU ROCm code objects that are embedded in HIP executables and shared objects. Each tool supports a *--help* option that provides more information.

Refer to the HIP Programming Guide v4.1 for additional information and examples.

https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Programming_Guide_v4.1.pdf

NOTE

The *extractkernel* tool in previous AMD ROCm releases has been removed from the AMD ROCm v4.1 release and will no longer be supported.

NOTE

The *roc-obj-ls and roc-obj-extract* tools may generate an error about the following missing Perl modules:

- *File::Which*
- *File::BaseDir*
- *File::Copy*
- *URI::Encode*

This error is due to the missing dependencies in the hip-base installer package. As a workaround, you may use the following instructions to install the Perl modules:

Ubuntu

```
apt-get install libfile-which-perl libfile-basedir-perl libfile-copy-recursive-
perl liburi-encode-perl
```

CentOS

```
yum install " perl(File::Which) perl(File::BaseDir) perl(File::Copy)
perl(URI::Encode)
```

ROCM DATA CENTER TOOL

Grafana Integration

The ROCm Data Center (RDC) Tool is enhanced with the Grafana plugin. Grafana is a common monitoring stack used for storing and visualizing time series data. Prometheus acts as the storage backend, and Grafana is used as the interface for analysis and visualization. Grafana has a plethora of visualization options and can be integrated with Prometheus for the ROCm Data Center (RDC) dashboard.

For more information about Grafana integration and installation, refer to the ROCm Data Center Tool User guide at:

https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_ROCm_DataCenter_Tool_User_Guide_v4.1.pdf

ROCM MATH AND COMMUNICATION LIBRARIES

The following enhancements are made in this release for ROCm Math and Communication Libraries:

Library	Changes	Link for More Information
rocSPARSE	Support for: • gebsrmm • gebsrmv • gebsrsv • coo2dense and dense2coo • generic API including axpby, gather, scatter, rot, spvv, spmv, spgemm, sparsetodense, densetosparse • mixed indexing types in matrix formats	*https://rocsparse.readthedocs.io/en/latest/*
rocSOLVER	Support for: • Eigensolver routines for symmetric/hermitian matrices: - STERF, STEQR • Linear solvers for general non-square systems: - GELS (API added with batched and strided_batched versions. Only the overdetermined non-transpose case is implemented in this release. Other cases will return rocblas_status_not_implemented status for now.) • Extended test coverage for functions returning information • Changelog file	*https://rocsolver.readthedocs.io/en/latest/*

Library	Changes	Link for More Information
	• Tridiagonalization routines for symmetric and hermitian matrices: - LATRD - SYTD2, SYTRD (with batched and strided_batched versions) - HETD2, HETRD (with batched and strided_batched versions) • Sample code and unit test for unified memory model/Heterogeneous Memory Management (HMM)	
hipCUB	New Iterator The new iterator *DiscardOutputIterator* in hipCUB represents a special kind of pointer that ignores values written to it upon dereference. It is useful for ignoring the output of certain algorithms without wasting memory capacity or bandwidth. *DiscardOutputIterator* may also be used to count the size of an algorithm's output, which was not known previously.	*https://hipcub.readthedocs.io/en/latest/*

HIP ENHANCEMENTS

AMD ROCm v4.1 consists of the following HIP enhancements:

Support for *hipEventDisableTiming* Flag

HIP now supports the *hipEventDisableTiming* flag for *hipEventCreateWithFlags*. Note, events created with this flag do not record profiling data and provide optimal performance when used for synchronization.

Cooperative Group Functions

Cooperative Groups defines, synchronizes, and communicates between groups of threads and blocks for efficiency and ease of management. HIP now supports the following kernel language Cooperative Groups types and functions:

Function	HIP	CUDA
void thread_group.sync() ;	✓	✓
unsigned thread_group.size();	✓	✓
unsigned thread_group.thread_rank() ;	✓	✓

Function	HIP	CUDA
bool thread_group.is_valid();	✓	✓
grid_group this_grid();	✓	✓
void grid_group.sync() ;	✓	✓
unsigned grid_group.size() ;	✓	✓
unsigned grid_group.thread_rank() ;	✓	✓
bool grid_group.is_valid();	✓	✓
multi_grid_group this_multi_grid() ;	✓	✓
void multi_grid_group.sync();	✓	✓
unsigned multi_grid_group.size() ;	✓	✓
unsigned multi_grid_group.thread_rank() ;	✓	✓
bool multi_grid_group.is_valid() ;	✓	✓
unsigned multi_grid_group.num_grids() ;	✓	✓
unsigned multi_grid_group.grid_rank();	✓	✓
thread_block this_thread_block() ;	✓	✓
multi_grid_group this_multi_grid() ;	✓	✓
void multi_grid_group.sync();	✓	✓
void thread_block.sync() ;	✓	✓
unsigned thread_block.size() ;	✓	✓
unsigned thread_block.thread_rank() ;	✓	✓
bool thread_block.is_valid() ;	✓	✓
dim3 thread_block.group_index() ;	✓	✓

Function	HIP	CUDA
dim3 thread_block.thread_index()	✓	✓

Support for Extern Shared Declarations

Previously, it was required to declare dynamic shared memory using the HIP_DYNAMIC_SHARED macro for accuracy as using static shared memory in the same kernel could result in overlapping memory ranges and data-races.

Now, the HIP-Clang compiler provides support for extern shared declarations, and the HIP_DYNAMIC_SHARED option is no longer required. You may use the standard extern definition:

```
extern __shared__ type var[];
```

OPENMP ENHANCEMENTS AND FIXES

This release includes the following OpenMP changes:

- Usability Enhancements
- Fixes to Internal Clang Math Headers
- OpenMP Defect Fixes

Usability Enhancements

- OMPD updates for flang
- To support OpenMP debugging, the selected OpenMP runtime sources are included in *lib-debug/src/openmp*. The ROCgdb debugger will find these automatically.
- Threadsafe hsa plugin for *libomptarget*
- Support multiple devices with malloc and hostrpc
- Improve hostrpc version check
- Add max reduction offload feature to flang
- Integration of changes to support HPC Toolkit
- Support for *fprintf*
- Initial support for GPU malloc and Free. The internal (device rtl) is required for GPU malloc and Free for nested parallelism. GPU malloc and Free are now replaced, which improves the device memory footprint.
- Increase detail of debug printing controlled by LIBOMPTARGET_KERNEL_TRACE environment variable
- Add support for *-gpubnames* in Flang Driver

Fixes to Internal Clang Math Headers

This release includes a set of changes applied to Clang internal headers to support OpenMP C, C++, FORTRAN, and HIP C. This establishes consistency between NVPTX and AMDGCN offloading, and OpenMP, HIP, and CUDA. OpenMP uses function variants and header overlays to define device versions of functions. This causes Clang LLVM IR codegen to mangle names of variants in both the definition and callsites of functions defined in the internal Clang headers. The changes apply to headers found in the installation subdirectory *lib/clang/11.0.0/include.*

The changes also temporarily eliminate the use of the libm bitcode libraries for C and C++. Although math functions are now defined with internal clang headers, a bitcode library of the C functions defined in the headers is still built for the FORTRAN toolchain linking. This is because FORTRAN cannot use C math headers. This bitcode library is installed in *lib/libdevice/libm-.bc*. The source build of the bitcode library is implemented with the aomp-extras repository and the component-built script *build_extras.sh*.

OpenMP Defect Fixes

The following OpenMP defects are fixed in this release:

- Openmpi configuration issue with real16.
- [flang] The AOMP 11.7-1 Fortran compiler claims to support the -isystem flag, but ignores it.
- [flang] producing internal compiler error when the character is used with KIND.
- [flang] openmp map clause on complex allocatable expressions !$omp target data map(chunk%tiles(1)%field%density0).
- Add a fatal error if missing -Xopenmp-target or -march options when -fopenmp-targets is specified. However, this requirement is not applicable for offloading to the host when there is only a single target and that target is the host.
- Openmp error message output for no_rocm_device_lib was asserting.
- Linkage on constant per-kernel symbols from external to weaklinkageonly to prevent duplicate symbols when building kokkos.
- Add environment variables ROCM_LLD_ARGS ROCM_LINK_ARGS ROCM_SELECT_ARGS to test driver options without compiler rebuild.
- Fix problems with device math functions being ambiguous, especially the pow function.ix aompcc to accept file type cxx.
- Fix a latent race between host runtime and devicertl.

MIOPEN-TENSILE INTEGRATION

MIOpenTensile provides host-callable interfaces to the Tensile library and supports the HIP programming model. You may use the Tensile feature in the HIP backend by setting the building environment variable value to ON.

```
MIOPEN_USE_MIOPENTENSILE=ON
```

MIOpenTensile is an open-source collaboration tool where external entities can submit source pull requests (PRs) for updates. MIOpenTensile maintainers review and approve the PRs using standard open-source practices.

For more information about the sources and the build system, see

https://github.com/ROCmSoftwarePlatform/MIOpenTensile

Known Issues in This Release

The following are the known issues in this release.

UPGRADE TO AMD ROCM V4.1 NOT SUPPORTED

An upgrade from previous releases to AMD ROCm v4.1 is not supported. A complete uninstallation of previous ROCm versions is required before installing a new version of ROCm.

PERFORMANCE IMPACT FOR KERNEL LAUNCH BOUND ATTRIBUTE

Kernels without the *__launch_bounds__* attribute assume the default maximum threads per block value. In the previous ROCm release, this value was 256. In the ROCm v4.1 release, it is changed to 1024. The objective of this change ensures the actual threads per block value used to launch a kernel, by default, are always within the launch bounds, thus, establishing the correctness of HIP programs.

NOTE: Using the above-mentioned approach may incur performance degradation in certain cases. Users must add a minimum launch bound to each kernel, which covers all possible threads per block values used to launch that kernel for correctness and performance.

The recommended workaround to recover the performance is to add *--gpu-max-threads-per-block=256* to the compilation options for HIP programs.

ISSUE WITH PASSING A SUBSET OF GPUS IN A MULTI-GPU SYSTEM

ROCm support for passing individual GPUs via the *docker --device* flag in a Docker run command has a known issue when passing a subset of GPUs in a multi-GPU system. The command runs without any warning or error notification. However, all GPU executable run outputs are randomly corrupted.

Using GPU targeting via the Docker command is not recommended for users of ROCm 4.1. There is no workaround for this issue currently.

PERFORMANCE IMPACT FOR LDS-BOUND KERNELS

The compiler in ROCm v4.1 generates LDS load and stores instructions that incorrectly assume equal performance between aligned and misaligned accesses. While this does not impact code correctness, it may result in sub-optimal performance.

This issue is under investigation, and there is no known workaround at this time.

Deprecations in This Release

This section describes deprecations and removals in AMD ROCm.

COMPILER-GENERATED CODE OBJECT VERSION 2 DEPRECATION

Compiler-generated code object version 2 is no longer supported and has been completely removed.

Support for loading code object version 2 is also deprecated with no announced removal release.

Hardware and Software Support

HARDWARE SUPPORT

ROCm is focused on using AMD GPUs to accelerate computational tasks such as machine learning, engineering workloads, and scientific computing. To focus our development efforts on these domains of interest, ROCm supports the following targeted set of hardware configurations.

Supported Graphics Processing Units

As the AMD ROCm platform has a focus on specific computational domains, AMD offers official support for a selection of GPUs that are designed to offer good performance and price in these domains.

Note: The integrated GPUs of Ryzen are not officially supported targets for ROCm.

ROCm officially supports AMD GPUs that use the following chips:

- GFX9 GPUs
  - "Vega 10" chips, such as on the AMD Radeon RX Vega 64 and Radeon Instinct MI25
  - "Vega 7nm" chips, such as on the Radeon Instinct MI50, Radeon Instinct MI60, AMD Radeon VII, Radeon Pro VII
- CDNA GPUs
  - MI100 chips such as on the AMD Instinct™ MI100

ROCm is a collection of software ranging from drivers and runtimes to libraries and developer tools. Some of this software may work with more GPUs than the "officially supported" list above, though AMD does not make any official claims of support for these devices on the ROCm software platform.

The following list of GPUs is enabled in the ROCm software. However, full support is not guaranteed:

- GFX8 GPUs
  - "Polaris 11" chips, such as on the AMD Radeon RX 570 and Radeon Pro WX 4100
  - "Polaris 12" chips, such as on the AMD Radeon RX 550 and Radeon RX 540

- GFX7 GPUs
  - "Hawaii" chips, such as the AMD Radeon R9 390X and FirePro W9100

As described in the next section, GFX8 GPUs require PCI Express 3.0 (PCIe 3.0) with support for PCIe atomics. This requires both CPU and motherboard support. GFX9 GPUs require PCIe 3.0 with support for PCIe atomics by default, but they can operate in most cases without this capability.

The integrated GPUs in AMD APUs are not officially supported targets for ROCm. As described below, "Carrizo", "Bristol Ridge", and "Raven Ridge" APUs are enabled in AMD upstream drivers and the ROCm OpenCL runtime. However, they are not enabled in the HIP runtime, and may not work due to motherboard or OEM hardware limitations. Note, they are not yet officially supported targets for ROCm.

GFX8 GPUS

Note: The GPUs require a host CPU and platform with PCIe 3.0 with support for PCIe atomics.

GFX8 GPUs			
Fiji (AMD)	**Polaris 10 (AMD)**	**Polaris 11 (AMD)**	**Polaris 12 (Lexa) (AMD)**
• Radeon R9 Fury • Radeon R9 Nano • Radeon R9 Fury X • Radeon Pro Duo (Fiji) • FirePro S9300 X2 • Radeon Instinct MI8	• Radeon RX 470 • Radeon RX 480 • Radeon RX 570 • Radeon RX 580 • Radeon Pro Duo (Polaris) • Radeon Pro WX 5100 • Radeon Pro WX 7100 • Radeon Instinct MI6	• Radeon RX 460 • Radeon RX 560 • Radeon Pro WX 4100	• Radeon RX 540 • Radeon RX 550 • Radeon Pro WX 2100 • Radeon Pro WX 3100

GFX9 GPUS

ROCm offers support for two chips from AMD's most recent "gfx9" generation of GPUs.

GFX9 GPUs	
Vega 10 (AMD)	**Vega 7nm (AMD)**
• Radeon RX Vega 56 • Radeon RX Vega 64 • Radeon Vega Frontier Edition • Radeon Pro WX 8200 • Radeon Pro WX 9100 • Radeon Pro V340 • Radeon Pro V340 MxGPU • Radeon Instinct MI25 **Note:** ROCm does not support Radeon Pro SSG.	• Radeon VII • Radeon Instinct MI50 • Radeon Instinct MI60

SUPPORTED CPUS

As described above, GFX8 GPUs require PCIe 3.0 with PCIe atomics to run ROCm. In particular, the CPU and every active PCIe point between the CPU and GPU require support for PCIe 3.0 and PCIe atomics. The CPU root must indicate PCIe AtomicOp Completion capabilities and any intermediate switch must indicate PCIe AtomicOp Routing capabilities.

The current CPUs which support PCIe Gen3 + PCIe Atomics are:

- AMD Ryzen CPUs
- CPUs in AMD Ryzen APUs
- AMD Ryzen Threadripper CPUs
- AMD EPYC CPUs
- Intel Xeon E7 v3 or newer CPUs
- Intel Xeon E5 v3 or newer CPUs
- Intel Xeon E3 v3 or newer CPUs
- Intel Core i7 v4, Core i5 v4, Core i3 v4 or newer CPUs (i.e. Haswell family or newer)
- Some Ivy Bridge-E systems

Beginning with ROCm 1.8, GFX9 GPUs (such as Vega 10) no longer require PCIe atomics. We have similarly made more options available for many PCIe lanes. GFX9 GPUs can now be run on CPUs without PCIe atomics and on older PCIe generations, such as PCIe 2.0. This is not supported on GPUs below GFX9, e.g. GFX8 cards in the Fiji and Polaris families.

If you are using any PCIe switches in your system, please note that PCIe Atomics are only supported on some switches, such as Broadcom PLX. When you install your GPUs, make sure you install them in a PCIe 3.0 x16, x8, x4, or x1 slot attached either directly to the CPU's Root I/O controller or via a PCIe switch directly attached to the CPU's Root I/O controller.

In our experience, many issues stem from trying to use consumer motherboards which provide physical x16 connectors that are electrically connected as e.g. PCIe 2.0 x4, PCIe slots connected via the Southbridge PCIe I/O controller, or PCIe slots connected through a PCIe switch that does not support PCIe atomics.

If you attempt to run ROCm on a system without proper PCIe atomic support, you may see an error in the kernel log (`dmesg`):

kfd: skipped device 1002:7300, PCI rejects atomics

Experimental support for our Hawaii (GFX7) GPUs (Radeon R9 290, R9 390, FirePro W9100, S9150, S9170) does not require or take advantage of PCIe Atomics. However, AMD recommends that you use a CPU from the list provided above for compatibility purposes.

NOT SUPPORTED OR LIMITED SUPPORT UNDER ROCM

LIMITED SUPPORT

- ROCm 4.x should support PCIe 2.0 enabled CPUs such as the AMD Opteron, Phenom, Phenom II, Athlon, Athlon X2, Athlon II and older Intel Xeon and Intel Core Architecture and Pentium CPUs. However, we have done very limited testing on these configurations, since our test farm has been catering to CPUs listed above. This is where we need community support.
  Please report these issues.
- Thunderbolt 1, 2, and 3 enabled breakout boxes should now be able to work with ROCm. Thunderbolt 1 and 2 are PCIe 2.0 based, and thus are only supported with GPUs that do not require PCIe 3.0 atomics (e.g. Vega 10). However, we have done no testing on this configuration and would need community support due to limited access to this type of equipment.

- AMD "Carrizo" and "Bristol Ridge" APUs are enabled to run OpenCL, but do not yet support HIP or our libraries built on top of these compilers and runtimes.
    - As of ROCm 2.1, "Carrizo" and "Bristol Ridge" require the use of upstream kernel drivers.
    - In addition, various "Carrizo" and "Bristol Ridge" platforms may not work due to OEM and ODM choices when it comes to key configuration parameters such as the inclusion of the required CRAT tables and IOMMU configuration parameters in the system BIOS.
    - Before purchasing such a system for ROCm, please verify that the BIOS provides an option for enabling IOMMUv2 and that the system BIOS properly exposes the correct CRAT table. Inquire with your vendor about the latter.

- AMD "Raven Ridge" APUs are enabled to run OpenCL, but do not yet support HIP or our libraries built on top of these compilers and runtimes.
    - As of ROCm 2.1, "Raven Ridge" requires the use of upstream kernel drivers.
    - In addition, various "Raven Ridge" platforms may not work due to OEM and ODM choices when it comes to key configuration parameters such as the inclusion of the required CRAT tables and IOMMU configuration parameters in the system BIOS.
    - Before purchasing such a system for ROCm, please verify that the BIOS provides an option for enabling IOMMUv2 and that the system BIOS properly exposes the correct CRAT table. Inquire with your vendor about the latter.

NOT SUPPORTED

- "Tonga", "Iceland", "Vega M", and "Vega 12" GPUs are not supported.
- AMD does not support GFX8-class GPUs (Fiji, Polaris, etc.) on CPUs that do not have PCIe3.0 with PCIe atomics.
    - AMD Carrizo and Kaveri APUs as hosts for such GPUs are not supported
    - Thunderbolt 1 and 2 enabled GPUs are not supported by GFX8 GPUs on ROCm. Thunderbolt 1 & 2 are based on PCIe 2.0.

In the default ROCm configuration, GFX8 and GFX9 GPUs require PCI Express 3.0 with PCIe atomics. The ROCm platform leverages these advanced capabilities to allow features such as user-level submission of work from the host to the GPU. This includes PCIe atomic Fetch and Add, Compare and Swap, Unconditional Swap, and AtomicOp Completion.

Current CPUs which support PCIe 3.0 + PCIe Atomics:

AMD	INTEL
Ryzen CPUs (Family 17h Model 01h-0Fh) • Ryzen 3 1300X • Ryzen 3 2300X • Ryzen 5 1600X • Ryzen 5 2600X • Ryzen 7 1800X • Ryzen 7 2700X	Intel Core i3, i5, and i7 CPUs from Haswell and beyond. This includes: • Haswell CPUs such as the Core i7 4790K • Broadwell CPUs such as the Core i7 5775C • Skylake CPUs such as the Core i7 6700K • Kaby Lake CPUs such as the Core i7 7740X • Coffee Lake CPUs such as the Core i7 8700K • Xeon CPUs from "v3" and newer • Some models of "Ivy Bridge-E" processors
Ryzen APUs (Family 17h Model 10h-1Fh – previously code-named Raven Ridge) such as: • Athlon 200GE • Ryzen 5 2400G **Note:** The integrated GPU in these devices is not guaranteed to work with ROCm.	
Ryzen Threadripper Workstation CPUs (Family 17h Model 01h-0Fh) such as: • Ryzen Threadripper 1950X • Ryzen Threadripper 2990WX	
EPYC Server CPUs (Family 17h Model 01h-0Fh) such as: • Epyc 7551P • Epyc 7601	

ROCM SUPPORT IN UPSTREAM LINUX KERNELS

As of ROCm 1.9.0, the ROCm user-level software is compatible with the AMD drivers in certain upstream Linux kernels.

As such, users have the option of either using the ROCK kernel driver that are part of AMD's ROCm repositories or using the upstream driver and only installing ROCm user-level utilities from AMD's ROCm repositories.

These releases of the upstream Linux kernel support the following GPUs in ROCm:

- 4.17: Fiji, Polaris 10, Polaris 11
- 4.18: Fiji, Polaris 10, Polaris 11, Vega10
- 4.20: Fiji, Polaris 10, Polaris 11, Vega10, Vega 7nm

The upstream driver may be useful for running ROCm software on systems that are not compatible with the kernel driver available in AMD's repositories.

For users that have the option of using either AMD's or the upstreamed driver, there are various tradeoffs to take into consideration:

	Using AMD's rock-dkms package	Using the upstream kernel driver
Pros	More GPU features and are enabled earlier	Includes the latest Linux kernel features
	Tested by AMD on supported distributions	May work on other distributions and with custom kernels
	Supported GPUs enabled regardless of kernel version	
	Includes the latest GPU firmware	
Cons	May not work on all Linux distributions or versions	Features and hardware support varies depending on the kernel version
	Not currently supported on kernels newer than 5.4	Limits GPU's usage of system memory to 3/8 of system memory (before 5.6). For 5.6 and beyond, both DKMS and upstream kernels allow the use of 15/16 of system memory.
		IPC and RDMA capabilities are not yet enabled
		Not tested by AMD to the same level as rock-dkms package
		Does not include the most up-to-date firmware