AMD ROCm™ Release Notes v3.9

Revision	**1028**
Issue Date:	**October 2020**

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.

Table of Contents

SUPPORTED OPERATING SYSTEMS

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

It also covers fixed defects and known issues in the AMD ROCm v3.9 release.

SUPPORTED OPERATING SYSTEMS

Support for Ubuntu 20.04.1

In this release, AMD ROCm extends support to Ubuntu 20.04.1, including v5.4 and v5.6-oem.

Support for SLES 15 SP2

This release extends support to SLES 15 SP2.

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 7.8 & RHEL 7.8 (Kernel 3.10.0-1127) (Using devtoolset-7 runtime support)
- CentOS 8.2 & RHEL 8.2 (Kernel 4.18.0) (devtoolset is not required)
- SLES 15 SP2

ROCm INSTALLATION UPDATES

FRESH INSTALLATION OF AMD ROCM V3.9 RECOMMENDED

A fresh and clean installation of AMD ROCm v3.9 is recommended. An upgrade from previous releases to AMD ROCm v3.9 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 v3.9 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 <= 3.9.0
    - command: echo <version> | sudo tee /opt/rocm-<version>/.info/version
    - example: echo 3.9.0 | sudo tee /opt/rocm-3.9.0/.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.

AMD ROCm V3.9 DOCUMENTATION UPDATES

AMD ROCM INSTALLATION GUIDE

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

- *Supported Environments*
- *Multi-Version Installation Instructions*

ROCM COMPILER DOCUMENTATION

The ROCm Compiler documentation updates include,

- OpenMP – Extras v12.9-0
- OpenMP-Extras Installation
- OpenMP-Extras Source Build
- AOMP-v11.9-0
- AOMP Source Build

For more information, see

https://rocmdocs.amd.com/en/latest/Programming_Guides/openmp_support.html

ROCM SYSTEM MANAGEMENT INFORMATION

ROCM-SMI version: 1.4.1 | Kernel version: 5.6.20

- ROCm SMI and Command Line Interface
- ROCm SMI APIs for Compute Unit Occupancy
    - Usage
    - Optional Arguments
    - Display Options
    - Topology
    - Pages Information
    - Hardware-related Information
    - Software-related/controlled information
    - Set Options
    - Reset Options
    - Auto-response Options
    - Output Options

For more information, refer to
https://rocmdocs.amd.com/en/latest/ROCm_System_Managment/ROCm-System-Managment.html#rocm-command-line-interface

AMD ROCM HIP DOCUMENTATION

- HIP Porting Guide – CU_Pointer_Attribute_Memory_Type

For more information, refer to

https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-porting-guide.html#hip-porting-guide

AMD ROCM GENERAL DOCUMENTATION LINKS

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

- For installation instructions on supped 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/Installation-Guide.html#software-stack-for-amd-gpu

- For AMD ROCm Release History, see
  https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation-Guide.html#amd-rocm-version-history

WHAT'S NEW IN THIS RELEASE

ROCM COMPILER ENHANCEMENTS

The ROCm compiler support in the *llvm-amdgpu-12.0.dev-amd64.deb* package is enhanced to include support for OpenMP. To utilize this support, the additional package *openmp-extras_12.9-0_amd64.deb* is required.

Note, by default, both packages are installed during the ROCm v3.9 installation.

For information about ROCm installation, refer to the *Installation Guide*.

AMD ROCm supports the following compilers:

- C++ compiler - Clang++
- C compiler - Clang
- Flang - FORTRAN compiler (FORTRAN 2003 standard)

NOTE: All of the above-mentioned compilers support:

- OpenMP standard 4.5 and an evolving subset of the OpenMP 5.0 standard
- OpenMP computational offloading to the AMD GPUs

Auxiliary Package Supporting OpenMP

The *openmp-extras_12.9-0_amd64.deb* auxiliary package supports OpenMP within the ROCm compiler. It contains OpenMP specific header files, which are installed in */opt/rocm/llvm/include* as well as runtime libraries, fortran runtime libraries, and device bitcode files in */opt/rocm/llvm/lib*. The auxiliary package also consists of examples in the */opt/rocm/llvm/examples* folder.

NOTE: The optional AOMP package resides in */opt/rocm//aomp/bin/clang* and the ROCm compiler, which supports OpenMP for AMDGPU, is located in */opt/rocm/llvm/bin/clang*.

AOMP Optional Package Deprecation

Before the AMD ROCm v3.9 release, the optional AOMP package provided support for OpenMP. While AOMP is available in this release, the optional package may be deprecated from ROCm in the future. It is recommended you transition to the ROCm compiler or AOMP standalone releases for OpenMP support.

Understanding ROCm Compiler OpenMP Support and AOMP OpenMP Support

The AOMP OpenMP support in ROCm v3.9 is based on the standalone AOMP v11.9-0, with LLVM v11 as the underlying system. However, the ROCm compiler's OpenMP support is based on LLVM v12 (upstream).

NOTE: Do not combine the object files from the two LLVM implementations. You must rebuild the application in its entirety using either the AOMP OpenMP or the ROCm OpenMP implementation.

Example – OpenMP Using the ROCm Compiler

```
$ cat helloworld.c

#include <stdio.h>

#include <omp.h>

 int main(void) {

  int isHost = 1;

#pragma omp target map(tofrom: isHost)

  {

    isHost = omp_is_initial_device();

    printf("Hello world. %d\n", 100);

    for (int i =0; i<5; i++) {

      printf("Hello world. iteration %d\n", i);

    }

  }

   printf("Target region executed on the %s\n", isHost ? "host" : "device");

  return isHost;

}

$ /opt/rocm/llvm/bin/clang  -O3 -target x86_64-pc-linux-gnu -fopenmp -fopenmp-
targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa -march=gfx900
helloworld.c -o helloworld
```

```
$ export LIBOMPTARGET_KERNEL_TRACE=1

$ ./helloworld

DEVID: 0 SGN:1 ConstWGSize:256  args: 1 teamsXthrds:(   1X 256) reqd:(   1X
0) n:__omp_offloading_34_af0aaa_main_l7

Hello world. 100

Hello world. iteration 0

Hello world. iteration 1

Hello world. iteration 2

Hello world. iteration 3

Hello world. iteration 4

Target region executed on the device
```

For more examples, see */opt/rocm/llvm/examples*

For more information about AMD ROCm compilers, see the Compiler Documentation section at,

https://rocmdocs.amd.com/en/latest/index.html

AMD ROCM SYSTEM MANAGEMENT INFORMATION

The AMD ROCm v3.9 release consists of the following ROCm System Management Information (SMI) enhancements:

- Shows the hardware topology
- The ROCm-SMI showpids option shows per-process Compute Unit (CU) Occupancy, VRAM usage, and SDMA usage
- Support for GPU Reset Event and Thermal Throttling Event in ROCm-SMI Library

ROCm-SMI Hardware Topology

The ROCm-SMI Command Line Interface (CLI) is enhanced to include new options to denote GPU inter-connect topology in the system along with the relative distance between each other and the closest NUMA (CPU) node for each GPU.

ROCm-SMI CLI Option	Description
--showtopo	hardware topology information
--showtopoweight	relative weight between GPUs
--showtopohops	number of hops between GPUs
--showtopotype	link type between GPUs
--showtoponuma	NUMA nodes

Compute Unit Occupancy

The AMD ROCm stack now supports a user process in querying Compute Unit (CU) occupancy at a particular moment. This service can be accessed to determine if a process P is using sufficient compute units.

A periodic collection is used to build the profile of a compute unit occupancy for a workload.

ROCm-SMI CLI Option	Description for Displaying
--showpids	current running Kernel Fusion Driver PIDs, including: • VRAM usage • SDMA usage • CU occupancy

ROCm supports this capability only on GFX9 devices. Users can access the functionality in two ways:

- indirectly from the SMI library
- directly via Sysfs

NOTE: On systems that have both GFX9 and non-GFX9 devices, users should interpret the compute unit (CU) occupancy value carefully as the service does not support non-GFX9 devices.

ACCESSING COMPUTE UNIT OCCUPANCY INDIRECTLY

The ROCm System Management Interface (SMI) library provides a convenient interface to determine the CU occupancy for a process. To get the CU occupancy of a process reported in percentage terms, invoke the SMI interface using *rsmi_compute_process_info_by_pid_get()*. The value is reported through the member field cu_occupancy of struct *rsmi_process_info_t*.

```
/**
 * @brief Encodes information about a process
 * @cu_occupancy Compute Unit usage in percent
 */
typedef struct {
    - - -,
    uint32_t cu_occupancy;
} rsmi_process_info_t;


/**
 * API to get information about a process
rsmi_status_t
    rsmi_compute_process_info_by_pid_get(uint32_t pid,
        rsmi_process_info_t *proc);
```

ACCESSING COMPUTE UNIT OCCUPANCY DIRECTLY USING SYSFS

Information provided by SMI library is built from sysfs. For every valid device, ROCm stack surfaces a file by the name cu_occupancy in Sysfs. Users can read this file to determine how that device is being used by a particular workload. The general structure of the file path is */proc/<pid>/stats_<gpuid>/cu_occupancy*

```
 /**
   * CU occupancy files for processes P1 and P2 on two devices with
   * ids: 1008 and 112326
   */
  /sys/devices/virtual/kfd/kfd/proc/<Pid_1>/stats_1008/cu_occupancy
  /sys/devices/virtual/kfd/kfd/proc/<Pid_1>/stats_2326/cu_occupancy
  /sys/devices/virtual/kfd/kfd/proc/<Pid_2>/stats_1008/cu_occupancy
  /sys/devices/virtual/kfd/kfd/proc/<Pid_2>/stats_2326/cu_occupancy


// To get CU occupancy for a process P<i>
  for each valid-device from device-list {
    path-1 = Build path for cu_occupancy file;
    path-2 = Build path for file Gpu-Properties;
    cu_in_use += Open and Read the file path-1;
    cu_total_cnt += Open and Read the file path-2;
  }
  cu_percent = ((cu_in_use * 100) / cu_total_cnt);
```

GPU Reset Event and Thermal Throttling Event

The ROCm-SMI library clients can now register for the following events:

Event	Trigger
RSMI_EVT_NOTIF_THERMAL_THROTTLE	if GPU is throttled
RSMI_EVT_NOTIF_GPU_PRE_RESET	before a GPU is reset
RSMI_EVT_NOTIF_GPU_POST_RESET	after a successful GPU reset

ROCM COMMUNICATION AND MATH LIBRARIES

'rocfft_execution_info_set_stream' API

rocFFT is a software library for computing Fast Fourier Transforms (FFT). It is part of AMD's software ecosystem based on ROCm. In addition to AMD GPU devices, the library can be compiled with the CUDA compiler using HIP tools for running on Nvidia GPU devices.

The 'rocfft_execution_info_set_stream' API is a function to specify optional and additional information to control execution. This API specifies the compute stream, which must be invoked before the call to *rocfft_execute*. Compute stream is the underlying device queue/stream where the library computations are inserted.

PREREQUISITES

Using the compute stream API makes the following assumptions:

- This stream already exists in the program and assigns work to the stream
- The stream must be of type hipStream_t. Note, it is an error to pass the address of a hipStream_t object

PARAMETERS

Input

info execution info handle
stream underlying compute stream

Improved GEMM Performance

Currently, *rocblas_gemm_ext2()* supports matrix multiplication *D <= alpha * A * B + beta * C*, where the A, B, C, and D matrices are single-precision float, column-major, and non-transposed, except that the row stride of C may equal 0. This means the first row of C is broadcast M times in C:

```
D00 D01 D02 D03              A00 A01 A02 A03      B00 B01 B02 B03              C00 C01 C02 C03
D10 D11 D12 D13  <=  alpha * A10 A11 A12 A13   *  B10 B11 B12 B13   +  beta * C00 C01 C02 C03   (first row is broadcast)
D20 D21 D22 D23              A20 A21 A22 A23      B20 B21 B22 B23              C00 C01 C02 C03
D30 D31 D32 D33              A30 A31 A32 A33      B30 B31 B32 B33              C00 C01 C02 C03
```

If an optimized kernel solution for a particular problem is not available, a slow fallback algorithm is used, and the first time a fallback algorithm is used, the following message is printed to standard error:

"Warning: Using slow on-host algorithm, because it is not implemented in Tensile yet."

NOTE: *ROCBLAS_LAYER* controls the logging of the calls. It is recommended to use logging with the rocblas_gemm_ext2() feature, to identify the precise parameters which are passed to it.

- Setting the *ROCBLAS_LAYER* environment variable to 2 will print the problem parameters as they are being executed.
- Setting the *ROCBLAS_LAYER* environment variable to 4 will collect all of the sizes, and print them out at the end of program execution.

For more logging information, refer to *https://rocblas.readthedocs.io/en/latest/logging.html*.

New Matrix Pruning Functions

In this release, the following new Matrix Pruning functions are introduced.

API	Description
rocsparse_prune_dense2csr_buffer_size()	This function computes the size of the user allocated temporary storage buffer used when converting and pruning a dense matrix to a CSR matrix.
rocsparse_prune_dense2csr_nnz()	This function computes the number of nonzero elements per row and the total number of nonzero elements in a dense matrix once elements less than the threshold are pruned from the matrix.
rocsparse_prune_dense2csr()	This function converts the matrix A in dense format into a sparse matrix in CSR format while pruning values that are less than the threshold. All the parameters are

	assumed to have been pre-allocated by the user.
rocsparse_prune_csr2csr_buffer_size()	Convert and prune sparse CSR matrix into a sparse CSR matrix.
rocsparse_prune_csr2csr_nnz()	Convert and prune sparse CSR matrix into a sparse CSR matrix.
rocsparse_prune_csr2csr()	Convert and prune sparse CSR matrix into a sparse CSR matrix.
rocsparse_prune_dense2csr_by_percentage_buffer_size()	This function computes the size of the user allocated temporary storage buffer used when converting and pruning by percentage a dense matrix to a CSR matrix.
rocsparse_prune_dense2csr_nnz_by_percentage()	This function computes the number of nonzero elements per row and the total number of nonzero elements in a dense matrix when converting and pruning by percentage a dense matrix to a CSR matrix.
rocsparse_prune_dense2csr_by_percentage()	This function converts the matrix A in dense format into a sparse matrix in CSR format while pruning values based on percentage.
rocsparse_prune_csr2csr_by_percentage_buffer_size()	Convert and prune by percentage a sparse CSR matrix into a sparse CSR matrix.
rocsparse_prune_csr2csr_nnz_by_percentage()	Convert and prune by percentage a sparse CSR matrix into a sparse CSR matrix.
rocsparse_prune_csr2csr_by_percentage()	Convert and prune by percentage a sparse CSR matrix into a sparse CSR matrix.

rocSOLVER General Matrix Singular Value Decomposition API

The rocSOLVER General Matrix Singular Value Decomposition (GESVD) API is now available in the AMD ROCm v3.9 release.

GESVD computes the Singular Values and, optionally, the Singular Vectors of a general m-by-n matrix A (Singular Value Decomposition).

The SVD of matrix A is given by:

```
A = U * S * V'
```

For more information, refer to

https://rocsolver.readthedocs.io/en/latest/userguide_api.html

AOMP ENHANCEMENTS

AOMP v11.9-0

The source code base for this release is the upstream LLVM 11 monorepo release/11.x sources as of August 18, 2020, with the hash value

1e6907f09030b636054b1c7b01de36f281a61fa2

The llvm-project branch used to build this release is aomp11. In addition to completing the source tarball, the artifacts of this release include the file llvm-project.patch. This file shows the delta from the llvm-project upstream release/11.x. The size of this patch XXXX lines in XXX files. These changes include support for flang driver, OMPD support, and the hsa libomptarget plugin. The goal is to reduce this with continued upstreaming activity.

The changes for this release of AOMP are:

- Fix compiler warnings for build_project.sh and build_openmp.sh.
- Fix: [flang] The AOMP 11.7-1 Fortran compiler claims to support the -isystem flag, but ignores it.
- Fix: [flang] producing internal compiler error when a character is used with KIND.
- Fix: [flang] openmp map clause on complex allocatable expressions !$omp target data map(chunk%tiles(1)%field%density0).
- DeviceRTL memory footprint has been reduced from ~2.3GB to ~770MB for AMDGCN target.
- Workaround for red_bug_51 failing on gfx908.
- Switch to python3 for ompd and rocgdb.
- Now require cmake 3.13.4 to compile from source.
- Fix aompcc to accept file type cxx.

AOMP v11.08-0

The source code base for this release is the upstream LLVM 11 monorepo release/11.x sources as of August 18, 2020 with the hash value

aabff0f7d564b22600b33731e0d78d2e70d060b4

The amd-llvm-project branch used to build this release is amd-stg-openmp. In addition to complete source tarball, the artifacts of this release includes the file llvm-project.patch. This file shows the delta from the llvm-project upstream release/11.x which is currently at 32715 lines in 240 files. These changes include support for flang driver, OMPD support and the hsa libomptarget plugin. Our goal is to reduce this with continued upstreaming activity.

The major changes for this release of AOMP are:

- Switch to the LLVM 11.x stable code base.
- OMPD updates for flang.
- To support debugging OpenMP, selected OpenMP runtime sources are included in lib-debug/src/openmp. The ROCgdb debugger will find these automatically.
- Threadsafe hsa plugin for libomptarget.
- Updates to support device libraries.
- Openmpi configure issue with real16 resolved.
- DeviceRTL memory use is now independent of number of openmp binaries.
- Startup latency on first kernel launch reduced by order of magnitude.

AOMP v11.07-1

The source code base for this release is the upstream LLVM 11 monorepo development sources as July 10, 2020 with hash valued 979c5023d3f0656cf51bd645936f52acd62b0333 The amd-llvm-project branch used to build this release is amd-stg-openmp. In addition to complete source tarball, the artifacts of this release includes the file llvm-project.patch. This file shows the delta from the llvm-project upstream trunk which is currently at 34121 lines in 277 files. Our goal is to reduce this with continued upstreaming activity.

- Inclusion of OMPD support which is not yet upstream
- Build of ROCgdb
- Host runtime optimisation. GPU image information is now mostly read on the host instead of from the GPU.
- Fixed the source build scripts so that building from the source tarball does not fail because of missing test directories. This fixes issue #116.

FIXED DEFECTS

The following defects are fixed in this release.

Defects	Resolution
Random Soft Hang Observed When Running ResNet-Based Models	Code fix
(AOMP) ‘Undefined Hidden Symbol’ Linker Error Causes Compilation Failure in HIP	Code fix
MIGraphx -> test_gpu_ops_test FAILED	Code fix
Unable to install RDC on CentOS/RHEL 7.8/8.2 & SLES	Code fix

KNOWN ISSUES

The following are the known issues in this release.

(AOMP) HIP EXAMPLE *DEVICE_LIB* FAILS TO COMPILE

The HIP example *device_lib* fails to compile and displays the following error:

lld: error: undefined hidden symbol: inc_arrayval

The recommended workaround is to use */opt/rocm/hip/bin/hipcc* to compile HIP applications.

HIPFORT INSTALLATION FAILURE

Hipfort fails to install during the ROCm installation.

As a workaround, you may force install hipfort using the following instructions:

Ubuntu

```
sudo apt-get -o Dpkg::Options::="--force-overwrite" install hipfort
```

SLES

Zypper gives you an option to continue with the overwrite during the installation.

CentOS

Download hipfort to a temporary location and force install with rpm:

```
yum install --downloadonly --downloaddir=/tmp/hipfort hipfort

rpm -i --replacefiles hipfort<package-version>
```

MEMORY FAULT ACCESS ERROR DURING MEMORY TEST OF ROCM VALIDATION SUITE

When the ROCm Validation Suite (RVS) is installed using the prebuilt Debian/rpm package and run for the first time, the memory module displays the following error message,

"Memory access fault by GPU node-<x> (Agent handle: 0xa55170) on address 0x7fc268c00000. Reason: Page not present or supervisor privilege.

Aborted (core dumped)"

As a workaround, run the test again. Subsequent runs appear to fix the error.

NOTE: The error may appear after a system reboot. Run the test again to fix the issue.

Note, reinstallation of ROCm Validation Suite is not required.

DEPRECATIONS

This section describes deprecations and removals in AMD ROCm.

WARNING: COMPILER-GENERATED CODE OBJECT VERSION 2 DEPRECATION

Compiler-generated code object version 2 is no longer supported and will be removed shortly. AMD ROCm users must plan for the code object version 2 deprecation immediately.

Support for loading code object version 2 is also being 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. In order 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.

ROCm officially supports AMD GPUs that use the following chips:

- GFX8 GPUs
  "Fiji" chips, such as on the AMD Radeon R9 Fury X and Radeon Instinct MI8
  "Polaris 10" chips, such as on the AMD Radeon RX 580 and Radeon Instinct MI6

- 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 or AMD Radeon VII

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

ROCm offers support for the following microprocessors from AMD's "gfx8" generation of 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 3.9.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 configurations parameters such as 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 configurations parameters such as 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	