# AMD ROCm™ v5.0 Release Notes

| | |
|---|---|
| Revision | **0209** |
| Issue Date: | **February 2022** |

# Table of Contents

# 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 this release.

## NOTICE FOR OPEN-SOURCE AND CLOSED-SOURCE ROCM REPOSITORIES IN FUTURE RELEASES

To make a distinction between open-source and closed-source components, all ROCm repositories will consist of sub-folders in future releases.

- All open-source components will be placed in the *base-url/<rocm-ver>/main* sub-folder
- All closed-source components will reside in the *base-url/<rocm-ver>/ proprietary* sub-folder

## LIST OF SUPPORTED OPERATING SYSTEMS

The AMD ROCm platform supports the following operating systems:

| OS-Version (64-bit) | Kernel Versions |
|---|---|
| CentOS 8.3 | 4.18.0-193.el8 |
| CentOS 7.9 | 3.10.0-1127 |
| RHEL 8.5 | 4.18.0-348.7.1.el8_5.x86_64 |
| RHEL 8.4 | 4.18.0-305.el8.x86_64 |
| RHEL 7.9 | 3.10.0-1160.6.1.el7 |
| SLES 15 SP3 | 5.3.18-59.16-default |
| Ubuntu 20.04.3 | 5.8.0 LTS / 5.11 HWE |
| Ubuntu 18.04.5 [5.4 HWE kernel] | 5.4.0-71-generic |

### Support for RHEL v8.5

This release extends support for RHEL v8.5.

## Supported GPUs

### RADEON PRO V620 AND W6800 WORKSTATION GPUS

This release extends ROCm support for Radeon Pro V620 and W6800 Workstation GPUs.

- SRIOV virtualization support for Radeon Pro V620
- KVM Hypervisor (1VF support only) on Ubuntu Host OS with Ubuntu, CentOs, and RHEL Guest
- Support for ROCm-SMI in an SRIOV environment. For more details, refer to the ROCm SMI API documentation.

**NOTE:** Radeon Pro v620 is not supported on SLES.

## ROCM INSTALLATION UPDATES FOR ROCM V5.0

This release has the following ROCm installation enhancements.

### Support for Kernel Mode Driver

In this release, users can install the kernel-mode driver using the Installer method. Some of the ROCm-specific use cases that the installer currently supports are:

- OpenCL (ROCr/KFD based) runtime
- HIP runtimes
- ROCm libraries and applications
- ROCm Compiler and device libraries
- ROCr runtime and thunk
- Kernel-mode driver

### Support for Multi-version ROCm Installation and Uninstallation

Users now can install multiple ROCm releases simultaneously on a system using the newly introduced installer script and package manager install mechanism.

Users can also uninstall multi-version ROCm releases using the *amdgpu-uninstall* script and package manager.

### Support for Updating Information on Local Repositories

In this release, the *amdgpu-install* script automates the process of updating local repository information before proceeding to ROCm installation.

### Support for Release Upgrades

Users can now upgrade the existing ROCm installation to specific or latest ROCm releases.

For more details, refer to the AMD ROCm Installation Guide v5.0.

# AMD ROCm V5.0 Documentation Updates

## NEW AMD ROCM DOCUMENTATION PORTAL – ROCM V4.5 AND ABOVE

Beginning ROCm release v5.0, AMD ROCm documentation has a new portal at *https://docs.amd.com*. This portal consists of ROCm documentation v4.5 and above.

For documentation prior to ROCm v4.5, you may continue to access *http://rocmdocs.amd.com*.

## DOCUMENTATION UPDATES FOR ROCM 5.0

### Deployment Tools

#### ROCM DATA CENTER TOOL DOCUMENTATION UPDATES

- ROCm Data Center Tool User Guide
- ROCm Data Center Tool API Guide

#### ROCM SYSTEM MANAGEMENT INTERFACE UPDATES

- System Management Interface Guide
- System Management Interface API Guide

#### ROCM COMMAND LINE INTERFACE UPDATES

- Command Line Interface Guide

### Machine Learning/AI Documentation Updates

- Deep Learning Guide
- MIGraphX API Guide
- MIOpen API Guide
- MIVisionX API Guide

## ROCm Libraries Documentation Updates

- hipSOLVER User Guide
- RCCL User Guide
- rocALUTION User Guide
- rocBLAS User Guide
- rocFFT User Guide
- rocRAND User Guide
- rocSOLVER User Guide
- rocSPARSE User Guide
- rocThrust User Guide

## Compilers and Tools

### ROCDEBUGGER DOCUMENTATION UPDATES

- ROCDebugger User Guide
- ROCDebugger API Guide

### ROCTRACER

- ROCTracer User Guide
- ROCTracer API Guide

### COMPILERS

- AMD Instinct High Performance Computing and Tuning Guide
- AMD Compiler Reference Guide

### HIPIFY DOCUMENTATION

- HIPify User Guide
- HIP Supported CUDA API Reference Guide

### ROCM DEBUG AGENT

- ROCm Debug Agent Guide
- System Level Debug Guide
- ROCm Validation Suite

## Programming Models Documentation

### HIP DOCUMENTATION

- HIP Programming Guide
- HIP API Guide
- HIP FAQ Guide

### OPENMP DOCUMENTATION

- OpenMP Support Guide

## ROCm Glossary

- ROCm Glossary – Terms and Definitions

## AMD ROCM LEGACY DOCUMENTATION LINKS – ROCM V4.3 AND PRIOR

- 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

## HIP ENHANCEMENTS

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

### 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.

### 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*

## NEW ENVIRONMENT VARIABLE

The following new environment variable is added in this release:

| Environment Variable | Value | Description |
|---|---|---|
| **HSA_COOP_CU_COUNT** | 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. |

## ROCM MATH AND COMMUNICATION LIBRARIES

In this release, ROCm Math and Communication Libraries consists of the following enhancements and fixes:

| Library | Changes |
|---|---|
| **rocBLAS** | **Added**<br>Added rocblas_get_version_string_size convenience function<br>Added rocblas_xtrmm_outofplace, an out-of-place version of rocblas_xtrmm<br>Added hpl and trig initialization for gemm_ex to rocblas-bench<br>Added source code gemm. It can be used as an alternative to Tensile for debugging and development<br>Added option ROCM_MATHLIBS_API_USE_HIP_COMPLEX to opt-in to use hipFloatComplex and hipDoubleComplex<br>**Optimizations**<br>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.<br>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.<br>**Changed**<br>Instantiate templated rocBLAS functions to reduce size of librocblas.so<br>Removed static library dependency on msgpack<br>Removed boost dependencies for clients<br>**Fixed**<br>Option to install script to build only rocBLAS clients with a pre-built rocBLAS library<br>Correctly set output of nrm2_batched_ex and nrm2_strided_batched_ex when given bad input<br>Fix for dgmm with side == rocblas_side_left and a negative incx<br>Fixed out-of-bounds read for small trsm<br>Fixed numerical checking for tbmv_strided_batched |

| Library | Changes |
|---|---|
| **hipBLAS** | **Added**<br>Added rocSOLVER functions to hipblas-bench<br>Added option ROCM_MATHLIBS_API_USE_HIP_COMPLEX to opt-in to use hipFloatComplex and hipDoubleComplex<br>Added compilation warning for future trmm changes<br>Added documentation to hipblas.h<br>Added option to forgo pivoting for getrf and getri when ipiv is nullptr<br>Added code coverage option<br>**Fixed**<br>Fixed use of incorrect 'HIP_PATH' when building from source.<br>Fixed windows packaging<br>Allowing negative increments in hipblas-bench<br>Removed boost dependency |
| **rocFFT** | **Changed**<br>Enabled runtime compilation of single FFT kernels > length 1024.<br>Re-aligned split device library into 4 roughly equal libraries.<br>Implemented the FuseShim framework to replace the original OptimizePlan<br>Implemented the generic buffer-assignment framework. The buffer assignment is no longer performed by each node. A generic algorithm is designed to test and pick the best assignment path. With the help of FuseShim, more kernel-fusions are achieved.<br>Do not read the imaginary part of the DC and Nyquist modes for even-length complex-to-real transforms.<br>**Optimizations**<br>Optimized twiddle-conjugation; complex-to-complex inverse transforms have similar performance to foward transforms now.<br>Improved performance of single-kernel small 2D transforms. |
| **hipFFT** | **Fixed**<br>Fixed incorrect reporting of rocFFT version.<br>**Changed**<br>Unconditionally enabled callback functionality. On the CUDA backend, |

| Library | Changes |
|---|---|
| | callbacks only run correctly when hipFFT is built as a static library, and is linked against the static cuFFT library. |
| **rocSPARSE** | **Added**<br>csrmv, coomv, ellmv, hybmv for (conjugate) transposed matrices<br>csrmv for symmetric matrices<br>**Changed**<br>spmm_ex is now deprecated and will be removed in the next major release<br>**Improved**<br>Optimization for gtsv |
| **hipSPARSE** | **Added**<br>Added (conjugate) transpose support for csrmv, hybmv and spmv routines |
| **rocALUTION** | **Changed**<br>Removed deprecated GlobalPairwiseAMG class, please use PairwiseAMG instead.<br>**Improved**<br>Improved documentation |
| **rocTHRUST** | **Updates**<br>Updated to match upstream Thrust 1.13.0<br>Updated to match upstream Thrust 1.14.0<br>Added async scan<br>**Changed**<br>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) |
| **rocSOLVER** | **Added**<br>Symmetric matrix factorizations:<br>LASYF<br>• SYTF2, SYTRF (with batched and strided_batched versions)<br>Added rocsolver_get_version_string_size to help with version string |

| Library | Changes |
|---|---|
| | queries<br>Added rocblas_layer_mode_ex and the ability to print kernel calls in the trace and profile logs<br>Expanded batched and strided_batched sample programs.<br>**Optimizations**<br>Improved general performance of LU factorization<br>Increased parallelism of specialized kernels when compiling from source, reducing build times on multi-core systems.<br>**Changed**<br>The rocsolver-test client now prints the rocSOLVER version used to run the tests, rather than the version used to build them<br>The rocsolver-bench client now prints the rocSOLVER version used in the benchmark<br>**Fixed**<br>Added missing stdint.h include to rocsolver.h |
| **hipSOLVER** | **Added**<br>Added functions<br>• sytrf<br>- hipsolverSsytrf_bufferSize, hipsolverDsytrf_bufferSize, hipsolverCsytrf_bufferSize, hipsolverZsytrf_bufferSize<br>- hipsolverSsytrf, hipsolverDsytrf, hipsolverCsytrf, hipsolverZsytrf<br>**Fixed**<br>Fixed use of incorrect HIP_PATH when building from source (#40). |
| **RCCL** | **Added**<br>Compatibility with NCCL 2.10.3<br>**Known issues**<br>Managed memory is not currently supported for clique-based kernels |

| Library | Changes |
|---|---|
| **hipCUB** | **Fixed**<br>Added missing includes to hipcub.hpp<br>**Added**<br>Bfloat16 support to test cases (device_reduce & device_radix_sort)<br>Device merge sort<br>Block merge sort<br>API update to CUB 1.14.0<br>**Changed**<br>The SetupNVCC.cmake automatic target selector select all of the capabalities of all available card for NVIDIA backend. |
| **rocPRIM** | **Fixed**<br>Enable bfloat16 tests and reduce threshold for bfloat16<br>Fix device scan limit_size feature<br>Non-optimized builds no longer trigger local memory limit errors<br>**Added**<br>Scan size limit feature<br>Reduce size limit feature<br>Transform size limit feature<br>Add block_load_striped and block_store_striped<br>Add gather_to_blocked to gather values from other threads into a blocked arrangement<br>The block sizes for device merge sorts initial block sort and its merge steps are now separate in its kernel config<br>Block sort step supports multiple items per thread<br>**Changed**<br>size_limit for scan, reduce and transform can now be set in the config struct instead of a parameter<br>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). |

| Library | Changes |
|---|---|
| | low-res input with high-res output (e.g. float input, double output)<br>Revert old Fiji workaround, because they solved the issue at compiler side<br>Update README cmake minimum version number<br>Block sort support multiple items per thread<br>Currently only powers of two block sizes, and items per threads are supported and only for full blocks<br>Bumped the minimum required version of CMake to 3.16<br>**Known issues**<br>Unit tests may soft hang on MI200 when running in hipMallocManaged mode.<br>device_segmented_radix_sort, device_scan unit tests failing for HIP on Windows<br>ReduceEmptyInput cause random faulire with bfloat16 |

## Clock Throttling for GPU Events

This feature lists GPU events as they occur in real-time and can be used with *kfdtest* to produce *vm_fault* events for testing.

The command can be called with either "**-e**" or "**--showevents**" like this:

**-e** [EVENT [EVENT ...]], **--showevents** [EVENT [EVENT ...]] Show event list

Where "EVENT" is any list combination of '**VM_FAULT**', '**THERMAL_THROTTLE**', or '**GPU_RESET**' and is NOT case sensitive.

**Note:** If no event arguments are passed, all events will be watched by default.

### CLI COMMANDS

```
./rocm-smi --showevents vm_fault thermal_throttle gpu_reset

========== ROCm System Management Interface ======================
========================= Show Events ============================
press 'q' or 'ctrl + c' to quit
DEVICE          TIME            TYPE            DESCRIPTION

======================== End of ROCm SMI Log =====================

*run kfdtest in another window to test for vm_fault events
```

**Note:** Unlike other rocm-smi CLI commands, this command does not quit unless specified by the user. Users may press either '**q**' or '**ctrl + c**' to quit.

## Display XGMI Bandwidth Between Nodes

The *rsmi_minmax_bandwidth_get* API reads the HW Topology file and displays bandwidth (min-max) between any two NUMA nodes in a matrix format.

The Command Line Interface (CLI) command can be called as follows:

```
./rocm-smi --shownodesbw
```

```
CLI ---shownodesbw

usage- We show maximum theoretical xgmi bandwidth between 2 numa nodes

sample output-

================= ROCm System Management Interface ================
================= Bandwidth ========================================
GPU0 GPU1 GPU2 GPU3 GPU4 GPU5 GPU6 GPU7
GPU0 N/A 50000-200000 50000-50000 0-0 0-0 0-0 50000-100000 0-0
GPU1 50000-200000 N/A 0-0 50000-50000 0-0 50000-50000 0-0 0-0
GPU2 50000-50000 0-0 N/A 50000-200000 50000-100000 0-0 0-0 0-0
GPU3 0-0 50000-50000 50000-200000 N/A 0-0 0-0 0-0 50000-50000
GPU4 0-0 0-0 50000-100000 0-0 N/A 50000-200000 50000-50000 0-0
GPU5 0-0 50000-50000 0-0 0-0 50000-200000 N/A 0-0 50000-50000
GPU6 50000-100000 0-0 0-0 0-0 50000-50000 0-0 N/A 50000-200000
GPU7 0-0 0-0 0-0 50000-50000 0-0 50000-50000 50000-200000 N/A
Format: min-max; Units: mps
```

**Note:**"0-0" min-max bandwidth indicates devices are not connected directly.

## P2P Connection Status

The *rsmi_is_p2p_accessible* API returns "True" if P2P can be implemented between two nodes, and returns "False" if P2P cannot be implemented between the two nodes.

The Command Line Interface command can be called as follows:

```
./rocm-smi –showtopoaccess
```

```
Sample Output:
./rocm-smi --showtopoaccess
====================== ROCm System Management Interface ======================
=================== Link accessibility between two GPUs ======================
GPU0 GPU1
GPU0 True True
GPU1 True True
============================== End of ROCm SMI Log ===========================
```

# Breaking Changes

## RUNTIME BREAKING CHANGE

Re-ordering of the enumerated type in hip_runtime_api.h to better match NV. See below for the difference in enumerated types.

ROCm software will be affected if any of the defined enums listed below are used in the code. Applications built with ROCm v5.0 enumerated types will work with a ROCm 4.5.2 driver. However, an undefined behavior error will occur with a ROCm v4.5.2 application that uses these enumerated types with a ROCm 5.0 runtime.

```
typedef enum hipDeviceAttribute_t {
-    hipDeviceAttributeMaxThreadsPerBlock,       ///< Maximum number of threads
per block.
-    hipDeviceAttributeMaxBlockDimX,             ///< Maximum x-dimension of a
block.
-    hipDeviceAttributeMaxBlockDimY,             ///< Maximum y-dimension of a
block.
-    hipDeviceAttributeMaxBlockDimZ,             ///< Maximum z-dimension of a
block.
-    hipDeviceAttributeMaxGridDimX,              ///< Maximum x-dimension of a
grid.
-    hipDeviceAttributeMaxGridDimY,              ///< Maximum y-dimension of a
grid.
-    hipDeviceAttributeMaxGridDimZ,              ///< Maximum z-dimension of a
grid.
-    hipDeviceAttributeMaxSharedMemoryPerBlock,  ///< Maximum shared memory
available per block in
-                                                ///< bytes.
-    hipDeviceAttributeTotalConstantMemory,      ///< Constant memory size in
bytes.
-    hipDeviceAttributeWarpSize,                 ///< Warp size in threads.
-    hipDeviceAttributeMaxRegistersPerBlock,  ///< Maximum number of 32-bit
registers available to a
-                                             ///< thread block. This number is
shared by all thread
-                                             ///< blocks simultaneously
resident on a
-                                             ///< multiprocessor.
```

```
-    hipDeviceAttributeClockRate,             ///< Peak clock frequency in
kilohertz.
-    hipDeviceAttributeMemoryClockRate,       ///< Peak memory clock frequency
in kilohertz.
-    hipDeviceAttributeMemoryBusWidth,        ///< Global memory bus width in
bits.
-    hipDeviceAttributeMultiprocessorCount,   ///< Number of multiprocessors on
the device.
-    hipDeviceAttributeComputeMode,           ///< Compute mode that device is
currently in.
-    hipDeviceAttributeL2CacheSize,  ///< Size of L2 cache in bytes. 0 if the
device doesn't have L2
-                                    ///< cache.
-    hipDeviceAttributeMaxThreadsPerMultiProcessor,  ///< Maximum resident
threads per
-                                                    ///< multiprocessor.
-    hipDeviceAttributeComputeCapabilityMajor,       ///< Major compute
capability version number.
-    hipDeviceAttributeComputeCapabilityMinor,       ///< Minor compute
capability version number.
-    hipDeviceAttributeConcurrentKernels,  ///< Device can possibly execute
multiple kernels
-                                          ///< concurrently.
-    hipDeviceAttributePciBusId,           ///< PCI Bus ID.
-    hipDeviceAttributePciDeviceId,        ///< PCI Device ID.
-    hipDeviceAttributeMaxSharedMemoryPerMultiprocessor,  ///< Maximum Shared
Memory Per
-                                                         ///< Multiprocessor.
-    hipDeviceAttributeIsMultiGpuBoard,                   ///< Multiple GPU
devices.
-    hipDeviceAttributeIntegrated,                        ///< iGPU
-    hipDeviceAttributeCooperativeLaunch,                 ///< Support
cooperative launch
-    hipDeviceAttributeCooperativeMultiDeviceLaunch,      ///< Support
cooperative launch on multiple devices
-    hipDeviceAttributeMaxTexture1DWidth,    ///< Maximum number of elements in
1D images
```

```
-    hipDeviceAttributeMaxTexture2DWidth,    ///< Maximum dimension width of 2D
images in image elements
-    hipDeviceAttributeMaxTexture2DHeight,   ///< Maximum dimension height of
2D images in image elements
-    hipDeviceAttributeMaxTexture3DWidth,    ///< Maximum dimension width of 3D
images in image elements
-    hipDeviceAttributeMaxTexture3DHeight,   ///< Maximum dimensions height of
3D images in image elements
-    hipDeviceAttributeMaxTexture3DDepth,    ///< Maximum dimensions depth of
3D images in image elements
+    hipDeviceAttributeCudaCompatibleBegin = 0,


-    hipDeviceAttributeHdpMemFlushCntl,      ///< Address of the
HDP_MEM_COHERENCY_FLUSH_CNTL register
-    hipDeviceAttributeHdpRegFlushCntl,      ///< Address of the
HDP_REG_COHERENCY_FLUSH_CNTL register
+    hipDeviceAttributeEccEnabled = hipDeviceAttributeCudaCompatibleBegin, ///<
Whether ECC support is enabled.
+    hipDeviceAttributeAccessPolicyMaxWindowSize,        ///< Cuda only. The
maximum size of the window policy in bytes.
+    hipDeviceAttributeAsyncEngineCount,                 ///< Cuda only.
Asynchronous engines number.
+    hipDeviceAttributeCanMapHostMemory,                 ///< Whether host
memory can be mapped into device address space
+    hipDeviceAttributeCanUseHostPointerForRegisteredMem,///< Cuda only. Device
can access host registered memory
+                                                        ///< at the same
virtual address as the CPU
+    hipDeviceAttributeClockRate,                        ///< Peak clock
frequency in kilohertz.
+    hipDeviceAttributeComputeMode,                      ///< Compute mode that
device is currently in.
+    hipDeviceAttributeComputePreemptionSupported,       ///< Cuda only. Device
supports Compute Preemption.
+    hipDeviceAttributeConcurrentKernels,                ///< Device can
possibly execute multiple kernels concurrently.
+    hipDeviceAttributeConcurrentManagedAccess,          ///< Device can
coherently access managed memory concurrently with the CPU
```

```
+    hipDeviceAttributeCooperativeLaunch,                ///< Support
cooperative launch
+    hipDeviceAttributeCooperativeMultiDeviceLaunch,     ///< Support
cooperative launch on multiple devices
+    hipDeviceAttributeDeviceOverlap,                    ///< Cuda only. Device
can concurrently copy memory and execute a kernel.
+                                                        ///< Deprecated. Use
instead asyncEngineCount.
+    hipDeviceAttributeDirectManagedMemAccessFromHost,   ///< Host can directly
access managed memory on
+                                                        ///< the device
without migration
+    hipDeviceAttributeGlobalL1CacheSupported,           ///< Cuda only. Device
supports caching globals in L1
+    hipDeviceAttributeHostNativeAtomicSupported,        ///< Cuda only. Link
between the device and the host supports native atomic operations
+    hipDeviceAttributeIntegrated,                       ///< Device is
integrated GPU
+    hipDeviceAttributeIsMultiGpuBoard,                  ///< Multiple GPU
devices.
+    hipDeviceAttributeKernelExecTimeout,                ///< Run time limit
for kernels executed on the device
+    hipDeviceAttributeL2CacheSize,                      ///< Size of L2 cache
in bytes. 0 if the device doesn't have L2 cache.
+    hipDeviceAttributeLocalL1CacheSupported,            ///< caching locals in
L1 is supported
+    hipDeviceAttributeLuid,                             ///< Cuda only. 8-byte
locally unique identifier in 8 bytes. Undefined on TCC and non-Windows
platforms
+    hipDeviceAttributeLuidDeviceNodeMask,               ///< Cuda only. Luid
device node mask. Undefined on TCC and non-Windows platforms
+    hipDeviceAttributeComputeCapabilityMajor,           ///< Major compute
capability version number.
+    hipDeviceAttributeManagedMemory,                    ///< Device supports
allocating managed memory on this system
+    hipDeviceAttributeMaxBlocksPerMultiProcessor,       ///< Cuda only. Max
block size per multiprocessor
+    hipDeviceAttributeMaxBlockDimX,                     ///< Max block size in
width.
```

```
+    hipDeviceAttributeMaxBlockDimY,                    ///< Max block size in
height.
+    hipDeviceAttributeMaxBlockDimZ,                    ///< Max block size in
depth.
+    hipDeviceAttributeMaxGridDimX,                     ///< Max grid size  in
width.
+    hipDeviceAttributeMaxGridDimY,                     ///< Max grid size  in
height.
+    hipDeviceAttributeMaxGridDimZ,                     ///< Max grid size  in
depth.
+    hipDeviceAttributeMaxSurface1D,                    ///< Maximum size of
1D surface.
+    hipDeviceAttributeMaxSurface1DLayered,             ///< Cuda only.
Maximum dimensions of 1D layered surface.
+    hipDeviceAttributeMaxSurface2D,                    ///< Maximum dimension
(width, height) of 2D surface.
+    hipDeviceAttributeMaxSurface2DLayered,             ///< Cuda only.
Maximum dimensions of 2D layered surface.
+    hipDeviceAttributeMaxSurface3D,                    ///< Maximum dimension
(width, height, depth) of 3D surface.
+    hipDeviceAttributeMaxSurfaceCubemap,               ///< Cuda only.
Maximum dimensions of Cubemap surface.
+    hipDeviceAttributeMaxSurfaceCubemapLayered,        ///< Cuda only.
Maximum dimension of Cubemap layered surface.
+    hipDeviceAttributeMaxTexture1DWidth,               ///< Maximum size of
1D texture.
+    hipDeviceAttributeMaxTexture1DLayered,             ///< Cuda only.
Maximum dimensions of 1D layered texture.
+    hipDeviceAttributeMaxTexture1DLinear,              ///< Maximum number of
elements allocatable in a 1D linear texture.
+                                                       ///< Use
cudaDeviceGetTexture1DLinearMaxWidth() instead on Cuda.
+    hipDeviceAttributeMaxTexture1DMipmap,              ///< Cuda only.
Maximum size of 1D mipmapped texture.
+    hipDeviceAttributeMaxTexture2DWidth,               ///< Maximum dimension
width of 2D texture.
+    hipDeviceAttributeMaxTexture2DHeight,              ///< Maximum dimension
hight of 2D texture.
+    hipDeviceAttributeMaxTexture2DGather,              ///< Cuda only.
Maximum dimensions of 2D texture if gather operations  performed.
```

```
+    hipDeviceAttributeMaxTexture2DLayered,             ///< Cuda only.
Maximum dimensions of 2D layered texture.
+    hipDeviceAttributeMaxTexture2DLinear,              ///< Cuda only.
Maximum dimensions (width, height, pitch) of 2D textures bound to pitched
memory.
+    hipDeviceAttributeMaxTexture2DMipmap,              ///< Cuda only.
Maximum dimensions of 2D mipmapped texture.
+    hipDeviceAttributeMaxTexture3DWidth,               ///< Maximum dimension
width of 3D texture.
+    hipDeviceAttributeMaxTexture3DHeight,              ///< Maximum dimension
height of 3D texture.
+    hipDeviceAttributeMaxTexture3DDepth,               ///< Maximum dimension
depth of 3D texture.
+    hipDeviceAttributeMaxTexture3DAlt,                 ///< Cuda only.
Maximum dimensions of alternate 3D texture.
+    hipDeviceAttributeMaxTextureCubemap,               ///< Cuda only.
Maximum dimensions of Cubemap texture
+    hipDeviceAttributeMaxTextureCubemapLayered,        ///< Cuda only.
Maximum dimensions of Cubemap layered texture.
+    hipDeviceAttributeMaxThreadsDim,                   ///< Maximum dimension
of a block
+    hipDeviceAttributeMaxThreadsPerBlock,              ///< Maximum number of
threads per block.
+    hipDeviceAttributeMaxThreadsPerMultiProcessor,     ///< Maximum resident
threads per multiprocessor.
+    hipDeviceAttributeMaxPitch,                        ///< Maximum pitch in
bytes allowed by memory copies
+    hipDeviceAttributeMemoryBusWidth,                  ///< Global memory bus
width in bits.
+    hipDeviceAttributeMemoryClockRate,                 ///< Peak memory clock
frequency in kilohertz.
+    hipDeviceAttributeComputeCapabilityMinor,          ///< Minor compute
capability version number.
+    hipDeviceAttributeMultiGpuBoardGroupID,            ///< Cuda only. Unique
ID of device group on the same multi-GPU board
+    hipDeviceAttributeMultiprocessorCount,             ///< Number of
multiprocessors on the device.
+    hipDeviceAttributeName,                            ///< Device name.
+    hipDeviceAttributePageableMemoryAccess,            ///< Device supports
coherently accessing pageable memory
```

```
+                                                           ///< without calling
hipHostRegister on it
+    hipDeviceAttributePageableMemoryAccessUsesHostPageTables, ///< Device
accesses pageable memory via the host's page tables
+    hipDeviceAttributePciBusId,                            ///< PCI Bus ID.
+    hipDeviceAttributePciDeviceId,                         ///< PCI Device ID.
+    hipDeviceAttributePciDomainID,                         ///< PCI Domain ID.
+    hipDeviceAttributePersistingL2CacheMaxSize,            ///< Cuda11 only.
Maximum l2 persisting lines capacity in bytes
+    hipDeviceAttributeMaxRegistersPerBlock,                ///< 32-bit registers
available to a thread block. This number is shared
+                                                           ///< by all thread
blocks simultaneously resident on a multiprocessor.
+    hipDeviceAttributeMaxRegistersPerMultiprocessor,       ///< 32-bit registers
available per block.
+    hipDeviceAttributeReservedSharedMemPerBlock,           ///< Cuda11 only.
Shared memory reserved by CUDA driver per block.
+    hipDeviceAttributeMaxSharedMemoryPerBlock,             ///< Maximum shared
memory available per block in bytes.
+    hipDeviceAttributeSharedMemPerBlockOptin,              ///< Cuda only.
Maximum shared memory per block usable by special opt in.
+    hipDeviceAttributeSharedMemPerMultiprocessor,          ///< Cuda only. Shared
memory available per multiprocessor.
+    hipDeviceAttributeSingleToDoublePrecisionPerfRatio, ///< Cuda only.
Performance ratio of single precision to double precision.
+    hipDeviceAttributeStreamPrioritiesSupported,           ///< Cuda only.
Whether to support stream priorities.
+    hipDeviceAttributeSurfaceAlignment,                    ///< Cuda only.
Alignment requirement for surfaces
+    hipDeviceAttributeTccDriver,                           ///< Cuda only.
Whether device is a Tesla device using TCC driver
+    hipDeviceAttributeTextureAlignment,                    ///< Alignment
requirement for textures
+    hipDeviceAttributeTexturePitchAlignment,               ///< Pitch alignment
requirement for 2D texture references bound to pitched memory;
+    hipDeviceAttributeTotalConstantMemory,                 ///< Constant memory
size in bytes.
+    hipDeviceAttributeTotalGlobalMem,                      ///< Global memory
available on devicice.
```

```
+    hipDeviceAttributeUnifiedAddressing,                ///< Cuda only. An
unified address space shared with the host.
+    hipDeviceAttributeUuid,                             ///< Cuda only. Unique
ID in 16 byte.
+    hipDeviceAttributeWarpSize,                         ///< Warp size in
threads.


-    hipDeviceAttributeMaxPitch,             ///< Maximum pitch in bytes
allowed by memory copies
-    hipDeviceAttributeTextureAlignment,     ///<Alignment requirement for
textures
-    hipDeviceAttributeTexturePitchAlignment, ///<Pitch alignment requirement
for 2D texture references bound to pitched memory;
-    hipDeviceAttributeKernelExecTimeout,    ///<Run time limit for kernels
executed on the device
-    hipDeviceAttributeCanMapHostMemory,     ///<Device can map host memory
into device address space
-    hipDeviceAttributeEccEnabled,           ///<Device has ECC support enabled
+    hipDeviceAttributeCudaCompatibleEnd = 9999,
+    hipDeviceAttributeAmdSpecificBegin = 10000,


-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedFunc,        ///<
Supports cooperative launch on multiple
-                                                                  ///devices
with unmatched functions
-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim,     ///<
Supports cooperative launch on multiple
-                                                                  ///devices
with unmatched grid dimensions
-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedBlockDim,    ///<
Supports cooperative launch on multiple
-                                                                  ///devices
with unmatched block dimensions
-    hipDeviceAttributeCooperativeMultiDeviceUnmatchedSharedMem,   ///<
Supports cooperative launch on multiple
-                                                                  ///devices
with unmatched shared memories
-    hipDeviceAttributeAsicRevision,         ///< Revision of the GPU in this
device
```

```
-    hipDeviceAttributeManagedMemory,        ///< Device supports allocating
managed memory on this system
-    hipDeviceAttributeDirectManagedMemAccessFromHost, ///< Host can directly
access managed memory on
-                                                       /// the device without
migration
-    hipDeviceAttributeConcurrentManagedAccess,  ///< Device can coherently
access managed memory
-                                                /// concurrently with the CPU
-    hipDeviceAttributePageableMemoryAccess,     ///< Device supports
coherently accessing pageable memory
-                                                /// without calling
hipHostRegister on it
-    hipDeviceAttributePageableMemoryAccessUsesHostPageTables, ///< Device
accesses pageable memory via
-                                                             /// the host's
page tables
-    hipDeviceAttributeCanUseStreamWaitValue ///< '1' if Device supports
hipStreamWaitValue32() and
-                                            ///< hipStreamWaitValue64() , '0'
otherwise.
+    hipDeviceAttributeClockInstructionRate =
hipDeviceAttributeAmdSpecificBegin,  ///< Frequency in khz of the timer used by
the device-side "clock*"
+    hipDeviceAttributeArch,                                    ///< Device
architecture
+    hipDeviceAttributeMaxSharedMemoryPerMultiprocessor,        ///< Maximum
Shared Memory PerMultiprocessor.
+    hipDeviceAttributeGcnArch,                                 ///< Device
gcn architecture
+    hipDeviceAttributeGcnArchName,                             ///< Device
gcnArch name in 256 bytes
+    hipDeviceAttributeHdpMemFlushCntl,                         ///< Address
of the HDP_MEM_COHERENCY_FLUSH_CNTL register
+    hipDeviceAttributeHdpRegFlushCntl,                         ///< Address
of the HDP_REG_COHERENCY_FLUSH_CNTL register
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedFunc,     ///< Supports
cooperative launch on multiple
+                                                               ///< devices
with unmatched functions
```

```
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim,   ///< Supports
cooperative launch on multiple
+                                                                ///< devices
with unmatched grid dimensions
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedBlockDim,  ///< Supports
cooperative launch on multiple
+                                                                ///< devices
with unmatched block dimensions
+    hipDeviceAttributeCooperativeMultiDeviceUnmatchedSharedMem, ///< Supports
cooperative launch on multiple
+                                                                ///< devices
with unmatched shared memories
+    hipDeviceAttributeIsLargeBar,                               ///< Whether
it is LargeBar
+    hipDeviceAttributeAsicRevision,                             ///< Revision
of the GPU in this device
+    hipDeviceAttributeCanUseStreamWaitValue,                    ///< '1' if
Device supports hipStreamWaitValue32() and
+                                                                ///<
hipStreamWaitValue64() , '0' otherwise.


+    hipDeviceAttributeAmdSpecificEnd = 19999,
+    hipDeviceAttributeVendorSpecificBegin = 20000,
+    // Extended attributes for vendors
 } hipDeviceAttribute_t;


 enum hipComputeMode {
```

# Known Issues in This Release

## 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.

## 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

**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.

**Current behavior**

BeginNS is lower than DispatchNS, which is incorrect.

**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.

**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.

## RADEON PRO V620 AND W6800 WORKSTATION GPUS

### 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.

## Deprecations and Warnings in This Release

### 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);
```

## HIP API DEPRECATIONS AND WARNINGS

### 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.

### 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, 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, 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.

### 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.

### WARNING - MIOPENTENSILE DEPRECATION

MIOpenTensile will be deprecated in a future release.