

---

# **ROCm Documentation**

***Release 4.5.0***

**Advanced Micro Devices, Inc.**

**Nov 02, 2021**



# RELEASE DOCUMENTATION

<b>1</b>	<b>The AMD ROCm Programming-Language Run-Time</b>	<b>3</b>
<b>2</b>	<b>Solid Compilation Foundation and Language Support</b>	<b>5</b>
<b>3</b>	<b>ROCm Learning Center</b>	<b>7</b>
3.1	AMD ROCm™ Release Notes v4.5 . . . . .	7
3.1.1	List of Supported Operating Systems . . . . .	7
3.1.2	Enhanced Installation Process for ROCm v4.5 . . . . .	8
3.1.3	AMD ROCm v4.5 Documentation Updates . . . . .	8
3.1.3.1	AMD ROCm Installation Guide . . . . .	8
3.1.3.2	AMD Instinct™ High Performance Computing and Tuning . . . . .	8
3.1.3.3	HIP Documentation Updates . . . . .	9
3.1.3.4	System Interface Management . . . . .	9
3.1.3.5	AMD ROCm Data Center Tool . . . . .	9
3.1.3.6	ROCm SMI API Guide . . . . .	9
3.1.3.7	ROC Debugger User and API Guide . . . . .	10
3.1.3.8	OpenMP Documentation . . . . .	10
3.1.3.9	AMD ROCm General Documentation Links . . . . .	10
3.1.4	What's New in This Release . . . . .	10
3.1.4.1	HIP Enhancements . . . . .	10
3.1.4.1.1	HIP Direct Dispatch . . . . .	10
3.1.4.1.2	Support for HIP Graph . . . . .	11
3.1.4.1.3	Enhanced <i>launch_bounds</i> Check Error Log Message . . . . .	11
3.1.4.1.4	HIP Runtime Compilation . . . . .	11
3.1.4.1.5	New Flag for Backwards Compatibility on float/double atomicAdd Function . . . . .	11
3.1.4.1.6	Updated HIP Version Definition . . . . .	12
3.1.4.1.7	Planned HIP Enhancements and Fixes . . . . .	12
3.1.4.1.7.1	Changes to hiprtc implementation to match nvrtc behavior . . . . .	12
3.1.4.1.7.2	HIP device attribute enumeration . . . . .	12
3.1.4.1.7.3	Changes to behavior of hipGetLastError() and hipPeekAtLastError() to match CUDA behavior available . . . . .	12
3.1.4.2	Unified Memory Support in ROCm . . . . .	12
3.1.4.2.1	Supported Operating Systems and Versions . . . . .	13
3.1.4.2.2	Unified Memory Support and XNACK . . . . .	13
3.1.4.3	System Management Interface . . . . .	14
3.1.4.3.1	Enhanced ROCm SMI <i>setpoweroverdrive</i> Functionality . . . . .	14
3.1.4.4	OpenMP Enhancements . . . . .	15
3.1.5	ROCm Math and Communication Libraries . . . . .	15
3.1.6	Known Issues in This Release . . . . .	17
3.1.6.1	Cache Issues with ROCProfiler . . . . .	17

3.1.6.2	Compiler Support for Function Pointers and Virtual Functions . . . . .	17
3.1.6.3	Debugger Process Exit May Cause ROCgdb Internal Error . . . . .	17
3.1.6.4	clinfo and rocminfo Do Not Display Marketing Name . . . . .	17
3.1.6.5	Stability Issue on LAMMPS-KOKKOS Applications . . . . .	18
3.1.7	Deprecations . . . . .	18
3.1.7.1	AMD Instinct MI25 End of Life . . . . .	18
3.1.7.2	Planned Deprecation for Code Object Versions 2 AND 3 . . . . .	18
3.1.8	DISCLAIMER . . . . .	18
3.2	Deprecations . . . . .	19
3.2.1	ROCm Release v4.5 . . . . .	19
3.2.1.1	AMD Instinct MI25 End of Life . . . . .	19
3.2.1.2	Planned Deprecation for Code Object Versions 2 AND 3 . . . . .	19
3.2.2	ROCm Release v4.1 . . . . .	19
3.2.2.1	COMPILER-GENERATED CODE OBJECT VERSION 2 DEPRECATION . . . . .	19
3.2.2.2	Changed HIP Environment Variables in ROCm v4.1 Release . . . . .	19
3.2.3	ROCm Release v4.0 . . . . .	20
3.2.3.1	ROCr Runtime Deprecations . . . . .	20
3.2.3.2	Deprecated ROCr Runtime Enumerations . . . . .	20
3.2.3.3	Deprecated ROCr Runtime Structs . . . . .	21
3.2.3.4	AOMP DEPRECATION . . . . .	21
3.2.4	ROCm Release v3.5 . . . . .	21
3.2.4.1	Heterogeneous Compute Compiler . . . . .	21
3.3	AMD ROCm Version History . . . . .	21
3.3.1	New features and enhancements in ROCm v4.3 . . . . .	22
3.3.2	New features and enhancements in ROCm v4.2 . . . . .	22
3.3.3	New features and enhancements in ROCm v4.1 . . . . .	22
3.3.4	New features and enhancements in ROCm v4.0 . . . . .	23
3.3.5	New features and enhancements in ROCm v3.10 . . . . .	23
3.3.6	New features and enhancements in ROCm v3.9 . . . . .	23
3.3.7	New features and enhancements in ROCm v3.8 . . . . .	24
3.3.8	New features and enhancements in ROCm v3.7 . . . . .	24
3.3.9	Patch Release - ROCm v3.5.1 . . . . .	24
3.3.10	New features and enhancements in ROCm v3.5 . . . . .	24
3.3.11	New features and enhancements in ROCm v3.3 . . . . .	25
3.3.12	New features and enhancements in ROCm v3.2 . . . . .	25
3.3.13	New features and enhancements in ROCm v3.1 . . . . .	26
3.3.14	New features and enhancements in ROCm v3.0 . . . . .	26
3.3.15	New features and enhancements in ROCm v2.10 . . . . .	27
3.3.16	New features and enhancements in ROCm 2.9 . . . . .	27
3.3.17	New features and enhancements in ROCm 2.8 . . . . .	28
3.3.18	New features and enhancements in ROCm 2.7.2 . . . . .	28
3.3.19	Issues fixed in ROCm 2.7.2 . . . . .	28
3.3.20	Upgrading from ROCm 2.7 to 2.7.2 . . . . .	28
3.3.21	New features and enhancements in ROCm 2.6 . . . . .	29
3.3.22	New features and enhancements in ROCm 2.5 . . . . .	30
3.3.23	New features and enhancements in ROCm 2.4 . . . . .	31
3.3.24	New features and enhancements in ROCm 2.3 . . . . .	31
3.3.25	New features and enhancements in ROCm 2.2 . . . . .	32
3.3.26	New features and enhancements in ROCm 2.1 . . . . .	32
3.3.27	New features and enhancements in ROCm 2.0 . . . . .	32
3.3.28	New features and enhancements in ROCm 1.9.2 . . . . .	33
3.3.29	New features and enhancements in ROCm 1.9.1 . . . . .	33
3.3.30	New features and enhancements in ROCm 1.9.0 . . . . .	33
3.3.31	New features as of ROCm 1.8.3 . . . . .	34

3.3.32	New features as of ROCm 1.8	34
3.3.33	New Features as of ROCm 1.7	35
3.3.34	New Features as of ROCm 1.5	35
3.4	ROCm™ Learning Center and Knowledge Base - NEW!!	35
3.4.1	ROCm Knowledge Base	35
3.4.2	ROCm Learning Center	36
3.4.2.1	Getting Started	36
3.4.2.2	Fundamentals of HIP Programming	36
3.4.2.3	From CUDA to HIP	36
3.4.2.4	Deep Learning on ROCm	36
3.4.2.5	Multi-GPU Programming	36
3.5	DISCLAIMER	36
3.6	ROCm Installation Guide v4.5	37
3.6.1	Overview of ROCm Installation Methods	39
3.6.1.1	About This Document	40
3.6.1.2	System Requirements	40
3.6.2	Prerequisite Actions	40
3.6.2.1	Confirm You Have a Supported Linux Distribution Version	41
3.6.2.1.1	How to Check Linux Distribution and Kernel Versions on Your System	41
3.6.2.1.1.1	Linux Distribution Information	41
3.6.2.1.1.2	Kernel Information	41
3.6.2.1.1.3	OS and Kernel Version Match	41
3.6.2.2	Confirm You Have a ROCm-Capable GPU	41
3.6.2.2.1	How to Verify Your System Has a ROCm-Capable GPU	42
3.6.2.3	Confirm the System Has the Required Tools and Packages Installed	42
3.6.2.3.1	How to Install and Configure Devtoolset-7	42
3.6.2.3.2	Required packages	42
3.6.2.3.3	Setting Permissions for Groups	43
3.6.3	Meta-packages in ROCm Programming Models	43
3.6.3.1	ROCm Package Naming Conventions	44
3.6.3.2	Components of ROCm Programming Models	44
3.6.3.3	Packages in ROCm Programming Models	45
3.6.4	Installation Methods	46
3.6.4.1	Installer Script Method	46
3.6.4.1.1	Downloading and Installing the Installer Script on Ubuntu	47
3.6.4.1.1.1	Ubuntu 18.04	47
3.6.4.1.1.2	Ubuntu 20.04	47
3.6.4.1.2	Downloading and Installing the Installer Script on RHEL/CentOS	47
3.6.4.1.2.1	RHEL/CentOS 7.9	47
3.6.4.1.2.2	RHEL 8.4/CentOS 8.3	47
3.6.4.1.3	Downloading and Installing the Installer Script on SLES 15	48
3.6.4.1.3.1	SLES 15 Service Pack 3	48
3.6.4.1.4	Using the Installer Script on Linux Distributions	48
3.6.4.2	Package Manager Method	49
3.6.4.2.1	Installing ROCm on Linux Distributions	49
3.6.4.2.2	Understanding AMDGPU and ROCm Stack Repositories on Linux Dis- tributions	50
3.6.4.2.2.1	Repositories with Latest Packages	50
3.6.4.2.2.2	Repositories for Specific Releases	50
3.6.4.2.3	Using Package Manager on Ubuntu	50
3.6.4.2.3.1	Installation Of Kernel Headers and Development Packages on Ubuntu	50
3.6.4.2.3.2	Base URLs For AMDGPU and ROCm Stack Repositories	51
3.6.4.2.3.3	Adding AMDGPU Stack Repository	51

	3.6.4.2.3.4	Install the Kernel Mode Driver and Reboot System . . . . .	52
	3.6.4.2.3.5	Add the ROCm Stack Repository . . . . .	52
	3.6.4.2.3.6	Install ROCm Meta-packages . . . . .	52
	3.6.4.2.4	Using Package Manager on RHEL/CentOS . . . . .	53
	3.6.4.2.4.1	Installation Of Kernel Headers and Development Packages on RHEL/CentOS . . . . .	53
	3.6.4.2.4.2	Base URLs For AMDGPU and ROCm Stack Repositories . . . . .	54
	3.6.4.2.4.3	Adding the AMDGPU Stack Repository . . . . .	54
	3.6.4.2.4.4	Install the Kernel Mode Driver and Reboot System . . . . .	55
	3.6.4.2.4.5	Add the ROCm Stack Repository . . . . .	55
	3.6.4.2.4.6	Install ROCm Meta-Packages . . . . .	55
	3.6.4.2.5	Using Package Manager on SLES/OpenSUSE . . . . .	56
	3.6.4.2.5.1	Installation of Kernel Headers and Development Packages . . . . .	56
	3.6.4.2.5.2	Base URLs For AMDGPU And ROCm Stack Repositories . . . . .	56
	3.6.4.2.5.3	Adding AMDGPU Stack Repository . . . . .	56
	3.6.4.2.5.4	Install the Kernel Mode Driver and Reboot System . . . . .	57
	3.6.4.2.5.5	Add the ROCm Stack Repository . . . . .	57
	3.6.4.2.5.6	Install ROCm Meta-Packages . . . . .	58
	3.6.4.3	Verification Process . . . . .	58
	3.6.4.3.1	Verifying ROCm Installation . . . . .	58
	3.6.4.3.2	Verifying Package Installation . . . . .	58
	3.6.5	ROCm Stack Uninstallation . . . . .	58
	3.6.5.1	Uninstalling ROCm Stack . . . . .	59
	3.6.5.1.1	Removing ROCm Toolkit and Driver . . . . .	59
	3.6.5.1.2	Choosing an Uninstallation Method . . . . .	59
	3.6.5.1.2.1	Uninstallation Using Uninstall Script . . . . .	59
	3.6.5.1.2.2	Uninstallation Using Package Manager . . . . .	59
	3.6.6	Troubleshooting . . . . .	61
	3.6.7	Frequently Asked Questions . . . . .	62
3.7	HIP Installation v4.5 . . . . .		62
	3.7.1	HIP Prerequisites . . . . .	63
	3.7.2	AMD Platform . . . . .	63
	3.7.3	NVIDIA Platform . . . . .	63
	3.7.4	Building HIP from Source . . . . .	64
	3.7.4.1	Get HIP source code . . . . .	64
	3.7.4.2	Set the environment variables . . . . .	64
	3.7.4.3	Build HIP . . . . .	64
	3.7.4.4	Default paths and environment variables . . . . .	64
	3.7.4.5	Verify your installation . . . . .	65
3.8	ROCm Installation v4.3 . . . . .		65
	3.8.1	Deploying ROCm . . . . .	65
	3.8.1.1	ROCm Repositories . . . . .	66
	3.8.1.2	Base Operating System Kernel Upgrade . . . . .	66
	3.8.2	Prerequisites . . . . .	66
	3.8.2.1	Perl Modules for HIP-Base Package . . . . .	67
	3.8.2.2	Complete Reinstallation OF AMD ROCm V4.3 Recommended . . . . .	67
	3.8.2.3	Multi-version Installation Updates . . . . .	67
	3.8.3	Setting Permissions for Groups . . . . .	68
	3.8.4	Supported Operating Systems . . . . .	68
	3.8.4.1	Ubuntu . . . . .	68
	3.8.4.1.1	Installing a ROCm Package from a Debian Repository . . . . .	68
	3.8.4.1.2	Uninstalling ROCm Packages from Ubuntu . . . . .	70
	3.8.4.1.3	Using Debian-based ROCm with Upstream Kernel Drivers . . . . .	70
	3.8.4.2	CentOS RHEL . . . . .	70

3.8.4.2.1	Preparing RHEL for Installation	70
3.8.4.2.1.1	Installing CentOS for DKMS	71
3.8.4.2.2	Installing ROCm	71
3.8.4.2.3	Testing the ROCm Installation	72
3.8.4.2.4	Compiling Applications Using HCC, HIP, and Other ROCm Software	72
3.8.4.2.5	Uninstalling ROCm from CentOS/RHEL	72
3.8.4.2.6	Using ROCm on CentOS/RHEL with Upstream Kernel Drivers	72
3.8.4.2.7	Installing Development Packages for Cross Compilation	72
3.8.4.3	SLES 15 Service Pack 2	73
3.8.4.3.1	Performing an OpenCL-only Installation of ROCm	74
3.8.5	ROCm Installation Known Issues and Workarounds	74
3.8.6	Getting the ROCm Source Code	74
3.8.7	Downloading the ROCm Source Code	75
3.9	Multi Version Installation	75
3.9.1	<b>Prerequisites</b>	75
3.9.2	Before You Begin	76
3.10	Using CMake with AMD ROCm	78
3.10.1	Finding Dependencies	78
3.10.2	Using HIP in CMake	78
3.10.3	Using AMD ROCm Libraries	79
3.10.4	ROCm CMake Packages	80
3.11	Mesa Multimedia Installation	80
3.11.1	Prerequisites	80
3.11.1.1	System Prerequisites	81
3.11.1.2	Installation Prerequisites	81
3.11.2	Installation Instructions	81
3.11.3	Check Installation	82
3.12	Tools Installation	84
3.12.1	ROCTracer	84
3.12.1.1	ROC-TX library: code annotation events API	84
3.12.1.2	Usage	84
3.12.1.2.1	rocTracer API	84
3.12.1.2.2	rocTX API	84
3.12.1.2.3	Library source tree	85
3.12.1.2.4	API Description	85
3.12.1.2.5	Code examples	85
3.12.1.2.6	Build and run test	85
3.13	Software Stack for AMD GPU	86
3.13.1	Machine Learning and High Performance Computing Software Stack for AMD GPU v4.1	86
3.13.1.1	ROCm Binary Package Structure	86
3.13.1.1.1	ROCm Core Components	86
3.13.1.1.2	ROCm Support Software	87
3.13.1.1.3	ROCm Compilers	87
3.13.1.1.4	ROCm Device Libraries	87
3.13.1.1.5	ROCm Development ToolChain	87
3.13.1.1.6	ROCm Libraries	87
3.13.1.1.6.1	ROCm Platform Packages	89
3.13.1.1.7	Drivers, ToolChains, Libraries, and Source Code	89
3.13.1.1.7.1	List of ROCm Packages for Supported Operating Systems	90
3.13.1.1.7.2	ROCm-Library Meta Packages	90
3.13.1.1.7.3	Meta Packages	91
3.14	Hardware and Software Support Information	91
3.15	AMD Instinct™ High Performance Computing and Tuning Guide	91
3.16	HIP Programming Guide v4.5	92

3.16.1	Programming Guide (PDF)	93
3.16.2	Related Topics	93
3.16.2.1	HIP API Guide	93
3.16.2.2	HIP_Supported_CUDA_API_Reference_Guide	93
3.16.2.3	AMD ROCm Compiler Reference Guide	93
3.16.2.4	HIP Installation Instructions	93
3.16.2.5	HIP FAQ	93
3.17	HIP API Documentation v4.5	93
3.18	HIP-Supported CUDA API Reference Guide v4.5	94
3.19	AMD ROCm Compiler Reference Guide v4.5	94
3.19.1	Supported CUDA APIs	94
3.19.2	Deprecated HIP APIs	94
3.19.2.1	HIP Context Management APIs	94
3.20	OpenCL Programming Guide	95
3.21	OpenMP Support	95
3.21.1	Overview	95
3.21.2	Installation	95
3.21.3	Usage	96
3.21.4	Helpful Tips	96
3.22	ROCm Libraries	96
3.22.1	Deprecated Libraries	97
3.22.1.1	hipeigen	97
3.23	Deep Learning	97
3.23.1	MIOpen API	97
3.23.2	TensorFlow	97
3.23.2.1	AMD ROCm Tensorflow v1.15 Release	97
3.23.2.2	AMD ROCm Tensorflow v2.2.0-beta1 Release	97
3.23.2.3	Tensorflow Installation	97
3.23.2.3.1	Tensorflow ROCm port: Basic installation on RHEL	98
3.23.2.3.1.1	Install ROCm	98
3.23.2.4	Tensorflow benchmarking	99
3.23.2.5	Tensorflow Installation with Docker	100
3.23.2.6	Tensorflow More Resources	100
3.23.3	MIOpen	100
3.23.3.1	ROCm MIOpen v2.0.1 Release	100
3.23.3.2	Porting from cuDNN to MIOpen	101
3.23.3.3	The ROCm 3.3 has prebuilt packages for MIOpen	101
3.23.4	PyTorch	101
3.23.4.1	Building PyTorch for ROCm	101
3.23.4.2	Recommended: Install using published PyTorch ROCm docker image:	102
3.23.4.3	Option 2: Install using PyTorch upstream docker file	102
3.23.4.4	Option 3: Install using minimal ROCm docker file	103
3.23.4.5	PyTorch examples	104
3.23.4.6	Building Caffe2 for ROCm	105
3.23.4.7	Option 1: Docker image with Caffe2 installed:	105
3.23.4.8	Option 2: Install using Caffe2 ROCm docker image:	105
3.23.4.9	Test the Caffe2 Installation	106
3.23.4.10	Run benchmarks	106
3.23.4.11	Running example scripts	106
3.23.4.12	Building own docker images	106
3.24	MIVisionX	106
3.25	AMD ROCm Profiler	117
3.25.1	Overview	117
3.25.2	Profiling Modes	117

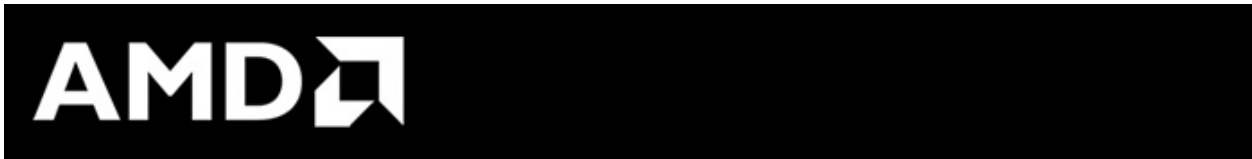


3.25.2.1	GPU profiling	118
3.25.2.1.1	Counters and metrics	118
3.25.2.1.1.1	Metrics query	119
3.25.2.1.1.2	Metrics collecting	120
3.25.2.1.1.3	Blocks instancing	120
3.25.2.1.1.4	HW limitations	120
3.25.2.2	Application tracing	121
3.25.2.2.1	HIP runtime trace	121
3.25.2.2.2	ROC <sub>r</sub> runtime trace	121
3.25.2.2.3	KFD driver trace	121
3.25.2.2.4	Code annotation	121
3.25.2.2.4.1	Start/stop API	121
3.25.2.2.4.2	rocTX basic markers API	122
3.25.2.3	Multiple GPUs profiling	122
3.25.3	Profiling control	122
3.25.3.1	Profiling scope	122
3.25.3.2	Tracing control	122
3.25.3.2.1	Filtering Traced APIs	122
3.25.3.2.2	Tracing period	123
3.25.3.3	Concurrent kernels	123
3.25.3.4	Multi-processes profiling	123
3.25.3.5	Errors logging	123
3.25.4	3rd party visualization tools	123
3.25.5	Runtime Environment Setup	123
3.25.6	Command line options	124
3.25.7	Publicly available counters and metrics	126
3.26	AMD ROCProfiler API	128
3.27	AMD ROCTracer API	130
3.28	AMD ROCm Debugger	131
3.29	AMD Debugger API	131
3.29.1	Introduction	131
3.29.2	Build the AMD Debugger API Library	131
3.29.3	Build the AMD Debugger API Specification Documentation	132
3.29.4	Known Limitations and Restrictions	133
3.29.5	Disclaimer	133
3.30	ROC <sub>m</sub> <sup>TM</sup> Data Center Tool	134
3.30.1	Objective	134
3.30.2	Target Audience	134
3.30.3	Download AMD ROC <sub>m</sub> Data Center Tool User Guide	135
3.30.4	Download AMD ROC <sub>m</sub> Data Center Tool API Guide	135
3.31	AMD ROC <sub>m</sub> Debug Agent Library	135
3.31.1	Introduction	135
3.31.2	Usage	135
3.31.3	Options	139
3.31.4	Build the ROCdebug-agent library	139
3.31.5	Test the ROCdebug-agent library	140
3.31.6	Known Limitations and Restrictions	141
3.31.7	Disclaimer	141
3.32	System Level Debug	141
3.32.1	ROC <sub>m</sub> Language & System Level Debug, Flags, and Environment Variables	141
3.32.1.1	ROC <sub>r</sub> Error Code	142
3.32.1.2	Command to dump firmware version and get Linux Kernel version	142
3.32.1.3	Debug Flags	142
3.32.1.4	ROC <sub>r</sub> level env variable for debug	142

	3.32.1.5	Turn Off Page Retry on GFX9/Vega devices	143
	3.32.1.6	HIP Environment Variables	143
	3.32.1.7	OpenCL Debug Flags	143
	3.32.1.8	PCIe-Debug	143
3.33		ROCmValidationSuite	143
	3.33.1	ROCmValidationSuite Modules	143
	3.33.2	Prerequisites	144
	3.33.3	Install ROCm stack, rocblas and rocm_smi64	145
	3.33.4	Building from Source	145
	3.33.5	Regression	146
3.34		System Management Interface	147
	3.34.1	ROCm SMI library	147
	3.34.2	ROCm System Management Interface (ROCm SMI) Library	147
	3.34.2.1	Important note about Versioning and Backward Compatibility	147
	3.34.3	Building ROCm SMI	147
	3.34.3.1	Additional Required software for building	147
	3.34.3.2	Building Documentation	148
	3.34.3.3	Building Tests	148
	3.34.4	Usage Basics	148
	3.34.4.1	Device Indices	148
	3.34.4.2	Hello ROCm SMI	149
	3.34.5	SYSFS Interface	149
	3.34.5.1	Naming and data format standards for sysfs files	149
	3.34.6	Global Attributes	151
	3.34.7	Voltages	153
	3.34.8	Fans	156
	3.34.9	Pulse with Modulation	159
	3.34.10	Temperatures	162
	3.34.11	Currents	165
	3.34.12	Power	168
	3.34.13	Energy	169
	3.34.14	Humidity	169
	3.34.15	Alarms	169
	3.34.16	Intrusion detection	172
	3.34.17	Average Sample Configuration	172
	3.34.18	sysfs attribute writes interpretation	173
	3.34.19	Performance	173
	3.34.20	KFD Topology	174
	3.34.21	HSA Agent Information	174
	3.34.22	Node Information	174
	3.34.23	Memory	174
	3.34.24	Cache	174
	3.34.25	IO-LINKS	175
	3.34.26	How to use topology information	175
	3.34.27	SMI Event Interface and Library	177
	3.34.28	ROCR_VISIBLE_DEVICES	177
	3.34.28.1	Interaction between ROCR_VISIBLE_DEVICES and CUDA_VISIBLE_DEVICES	177
	3.34.29	Device cgroup	178
3.35		ROCm Command Line Interface	178
	3.35.1	Clock and Temperature Management	178
	3.35.2	SDMA Usage Per-process	185
	3.35.3	Hardware Topology	185
3.36		GCN ISA Manuals	186
	3.36.1	GCN 1.1	186

3.36.2	GCN 2.0 . . . . .	186
3.36.3	Vega . . . . .	186
3.36.4	Inline GCN ISA Assembly Guide . . . . .	186
3.36.4.1	The Art of AMDGCN Assembly: How to Bend the Machine to Your Will . . . . .	186
3.36.4.2	DS Permute Instructions . . . . .	186
3.36.4.3	Passing Parameters to a Kernel . . . . .	187
3.36.4.4	The GPR Counting . . . . .	189
3.36.4.5	Compiling GCN ASM Kernel Into Hsaco . . . . .	190
3.37	Remote Device Programming . . . . .	191
3.37.1	ROCmRDMA . . . . .	191
3.37.1.1	Restrictions and limitations . . . . .	191
3.37.1.2	ROCmRDMA interface specification . . . . .	191
3.37.1.3	API versions . . . . .	191
3.37.1.4	Data structures . . . . .	192
3.37.1.5	The function to query ROCmRDMA interface . . . . .	192
3.37.1.6	ROCmRDMA interface functions description . . . . .	193
3.37.2	UCX . . . . .	194
3.37.3	OpenMPI . . . . .	194
3.37.4	IPC API . . . . .	195
3.37.4.1	New Datatypes . . . . .	195
3.37.5	MPICH . . . . .	198
3.37.5.1	Building and Installing MPICH . . . . .	198
3.38	v4.1 ROCm Installation . . . . .	198
3.38.1	Deploying ROCm . . . . .	199
3.38.1.1	ROCm Repositories . . . . .	199
3.38.1.2	Base Operating System Kernel Upgrade . . . . .	199
3.38.2	Prerequisites . . . . .	199
3.38.2.1	Perl Modules for HIP-Base Package . . . . .	200
3.38.2.2	Complete Reinstallation OF AMD ROCm V4.1 Recommended . . . . .	200
3.38.2.3	Multi-version Installation Updates . . . . .	200
3.38.3	Supported Operating Systems . . . . .	201
3.38.3.1	Ubuntu . . . . .	201
3.38.3.1.1	Installing a ROCm Package from a Debian Repository . . . . .	201
3.38.3.1.2	Uninstalling ROCm Packages from Ubuntu . . . . .	202
3.38.3.1.3	Using Debian-based ROCm with Upstream Kernel Drivers . . . . .	202
3.38.3.2	CentOS RHEL . . . . .	203
3.38.3.2.1	Preparing RHEL for Installation . . . . .	203
3.38.3.2.1.1	Installing CentOS for DKMS . . . . .	203
3.38.3.2.2	Installing ROCm . . . . .	204
3.38.3.2.3	Testing the ROCm Installation . . . . .	204
3.38.3.2.4	Compiling Applications Using HCC, HIP, and Other ROCm Software . . . . .	205
3.38.3.2.5	Uninstalling ROCm from CentOS/RHEL . . . . .	205
3.38.3.2.6	Using ROCm on CentOS/RHEL with Upstream Kernel Drivers . . . . .	205
3.38.3.2.7	Installing Development Packages for Cross Compilation . . . . .	205
3.38.3.3	SLES 15 Service Pack 2 . . . . .	205
3.38.3.3.1	Performing an OpenCL-only Installation of ROCm . . . . .	207
3.38.4	ROCm Installation Known Issues and Workarounds . . . . .	207
3.38.5	Getting the ROCm Source Code . . . . .	207
3.38.6	Downloading the ROCm Source Code . . . . .	207

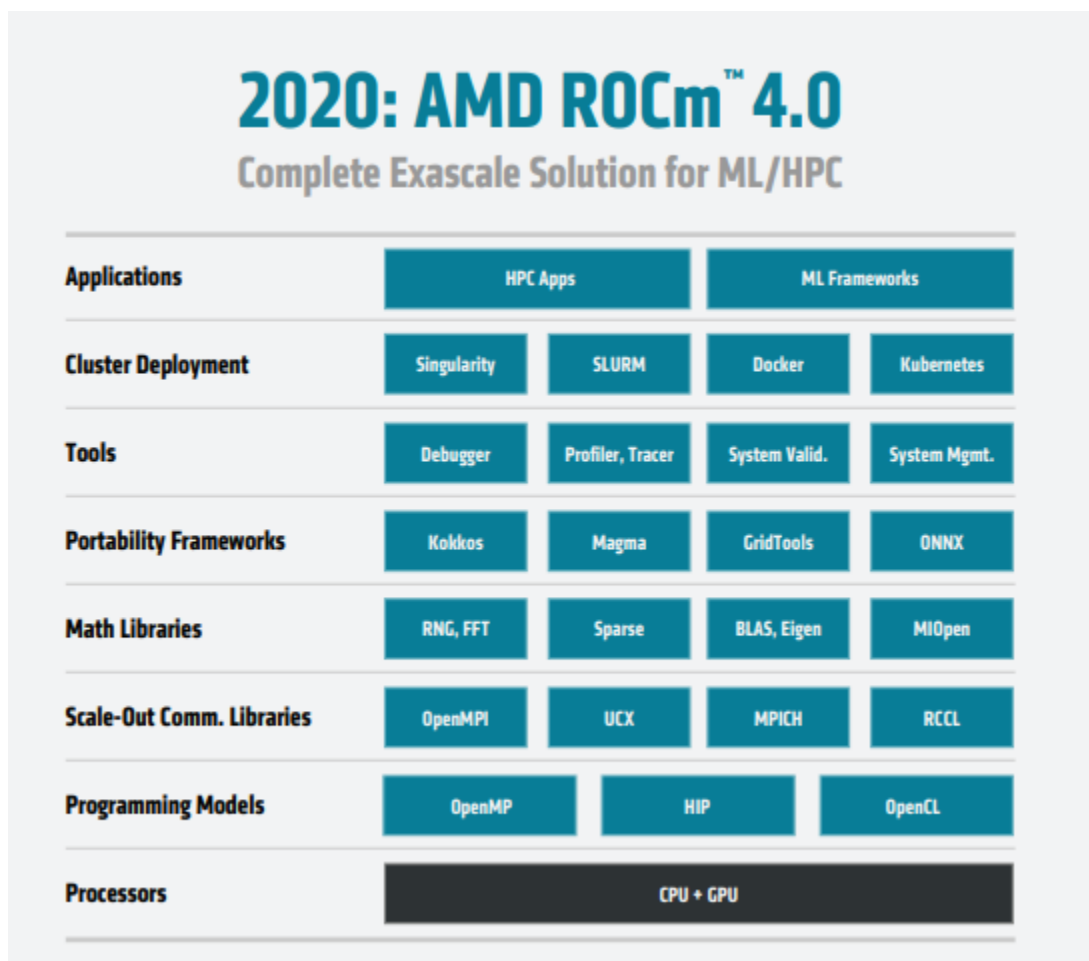




AMD ROCm is the first open-source software development platform for HPC/Hyperscale-class GPU computing. AMD ROCm brings the UNIX philosophy of choice, minimalism and modular software development to GPU computing.

Since the ROCm ecosystem is comprised of open technologies: frameworks (Tensorflow / PyTorch), libraries (MIOpen / Blas / RCCL), programming model (HIP), inter-connect (OCD) and up streamed Linux® Kernel support – the platform is continually optimized for performance and extensibility. Tools, guidance and insights are shared freely across the ROCm GitHub community and forums.

**Note:** The AMD ROCm™ open software platform is a compute stack for headless system deployments. GUI-based software applications are currently not supported.

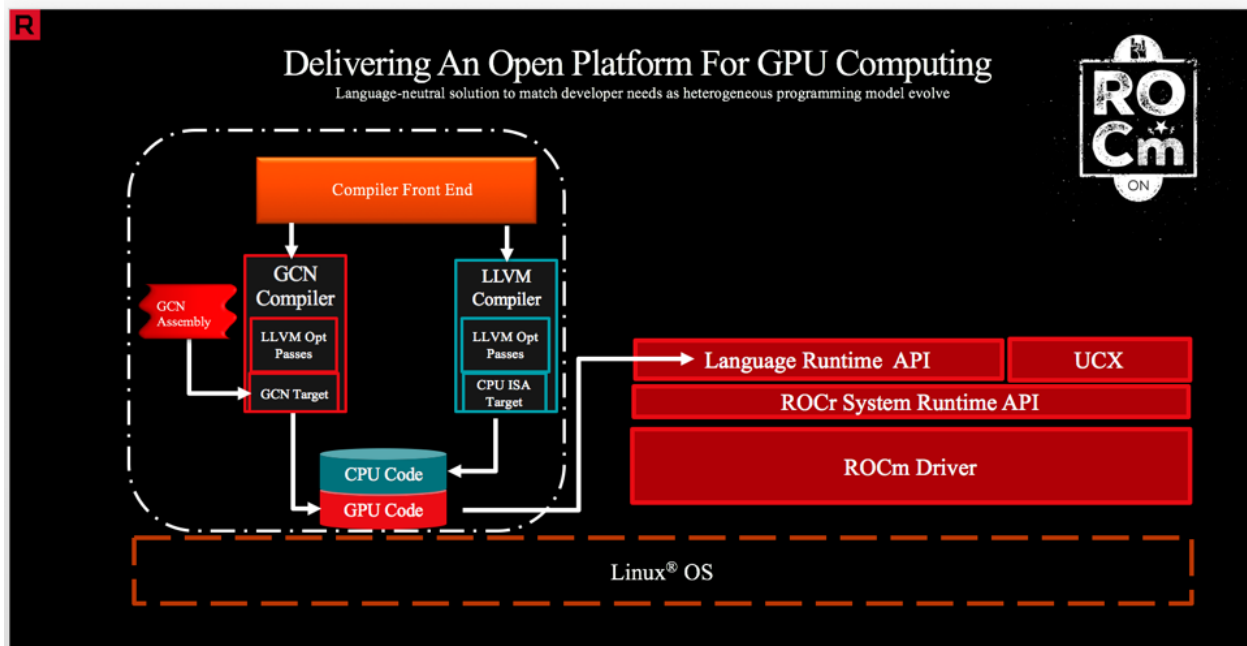


AMD ROCm is built for scale; it supports multi-GPU computing in and out of server-node communication through RDMA. AMD ROCm also simplifies the stack when the driver directly incorporates RDMA peer-sync support.



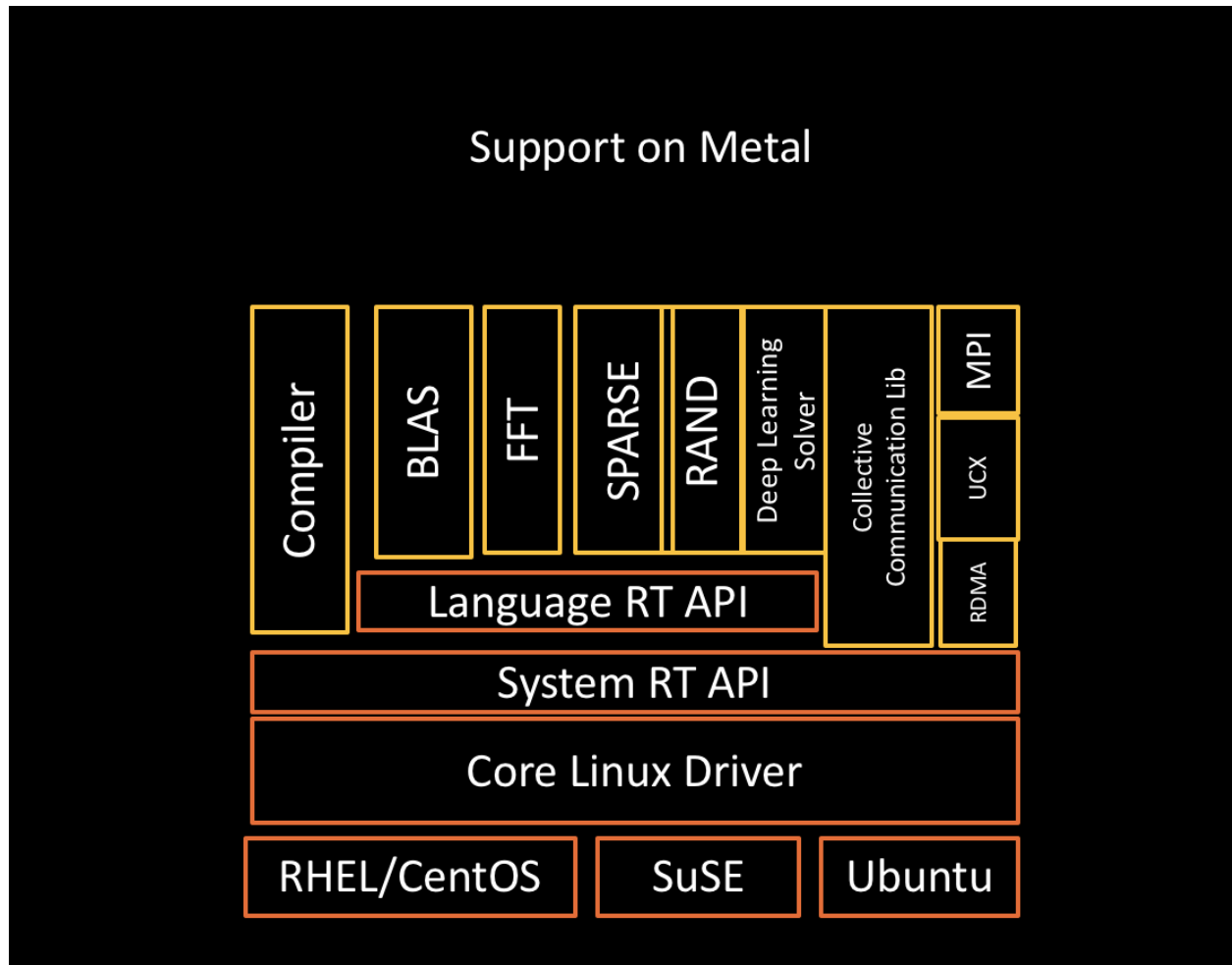
## THE AMD ROCM PROGRAMMING-LANGUAGE RUN-TIME

The AMD ROCr System Runtime is language independent and makes heavy use of the Heterogeneous System Architecture (HSA) Runtime API. This approach provides a rich foundation to execute programming languages, such as HIP and OpenMP.



Important features include the following:

- Multi-GPU coarse-grain shared virtual memory
- Process concurrency and preemption
- Large memory allocations
- HSA signals and atomics
- User-mode queues and DMA
- Standardized loader and code-object format
- Dynamic and offline-compilation support
- Peer-to-peer multi-GPU operation with RDMA support
- Profiler trace and event-collection API
- Systems-management API and tools





## SOLID COMPILATION FOUNDATION AND LANGUAGE SUPPORT

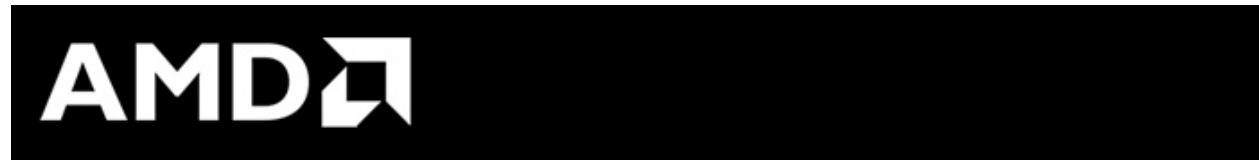
- LLVM compiler foundation
- HIP for application portability
- GCN assembler and disassembler

AMD ROCm gives developers the flexibility of choice for hardware and aids in the development of compute-intensive applications.



## ROCm LEARNING CENTER

<https://developer.amd.com/resources/rocm-resources/rocm-learning-center/>



### 3.1 AMD ROCm™ Release Notes v4.5

October, 2021

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.

#### 3.1.1 List of Supported Operating Systems

The AMD ROCm platform supports the following operating systems:

OS	Kernel
SLES15 SP3	5.3.18-24.49
RHEL 7.9	3.10.0-1160.6.1.el7
CentOS 7.9	3.10.0-1127
RHEL 8.4	4.18.0-193.1.1.el8
CentOS 8.3	4.18.0-193.el8
Ubuntu 18.04.5	5.4.0-71-generic
Ubuntu 20.04.3HWE	5.8.0-48-generic
Host OS	Azure RS1.86
Guest OS	Ubuntu 20.04

### 3.1.2 Enhanced Installation Process for ROCm v4.5

In addition to the installation method using the native Package Manager, AMD ROCm v4.5 introduces added methods to install ROCm. With this release, the ROCm installation uses the *amdgpu-install* and *amdgpu-uninstall* scripts.

The *amdgpu-install* script streamlines the installation process by:

- Abstracting the distribution-specific package installation logic
- Performing the repository set-up
- Allowing user to specify the use case and automating the installation of all the required packages,
- Performing post-install checks to verify whether the installation was performed successfully
- Installing the uninstallation script

The *amdgpu-uninstall* script allows the removal of the entire ROCm stack by using a single command.

Some of the ROCm-specific use cases that the installer currently supports are:

- OpenCL (ROC<sub>r</sub>/KFD based) runtime
- HIP runtimes
- ROCm libraries and applications
- ROCm Compiler and device libraries
- ROC<sub>r</sub> runtime and thunk

For more information, refer to the *Installation Methods* section in this guide.

**Note:** Graphics use cases are not supported in this release.

For more details, refer to the AMD ROCm Installation Guide v4.5 at,

[https://rocmdocs.amd.com/en/latest/Installation\\_Guide/Installation\\_new.html](https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation_new.html)

### 3.1.3 AMD ROCm v4.5 Documentation Updates

#### 3.1.3.1 AMD ROCm Installation Guide

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

- New - Installation Guide for ROCm v4.5

[https://rocmdocs.amd.com/en/latest/Installation\\_Guide/Installation\\_new.html](https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation_new.html)

#### 3.1.3.2 AMD Instinct™ High Performance Computing and Tuning

- New - AMD Instinct™ High Performance Computing and Tuning Guide  
see [AMD Instinct™ High Performance Computing and Tuning Guide](#)

### 3.1.3.3 HIP Documentation Updates

- HIP installation instructions  
[https://rocmdocs.amd.com/en/latest/Installation\\_Guide/HIP-Installation.html](https://rocmdocs.amd.com/en/latest/Installation_Guide/HIP-Installation.html)
- HIP Programming Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Programming\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Programming_Guide.pdf)
- HIP API Guide  
<https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD-HIP-API-4.5.pdf>
- HIP-Supported CUDA API Reference Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Supported\\_CUDA\\_API\\_Reference\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Supported_CUDA_API_Reference_Guide.pdf)
- AMD ROCm Compiler Reference Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_Compiler\\_Reference\\_Guide\\_v4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_Compiler_Reference_Guide_v4.5.pdf)
- HIP FAQ  
[https://rocmdocs.amd.com/en/latest/Programming\\_Guides/HIP-FAQ.html#hip-faq](https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-FAQ.html#hip-faq)

### 3.1.3.4 System Interface Management

- System Interface Management (SMI)  
[https://rocmdocs.amd.com/en/latest/ROCm\\_System\\_Managment/ROCm-System-Management.html](https://rocmdocs.amd.com/en/latest/ROCm_System_Managment/ROCm-System-Management.html)

### 3.1.3.5 AMD ROCm Data Center Tool

- AMD ROCm Data Center Tool API Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/RDC\\_API\\_Manual\\_4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/RDC_API_Manual_4.5.pdf)
- AMD ROCm Data Center Tool User Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_ROCm\\_DataCenter\\_Tool\\_User\\_Guide\\_v4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_ROCm_DataCenter_Tool_User_Guide_v4.5.pdf)

### 3.1.3.6 ROCm SMI API Guide

- ROCm SMI API Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm\\_SMI\\_Manual\\_4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm_SMI_Manual_4.5.pdf)

### 3.1.3.7 ROC Debugger User and API Guide

- ROCDebugger User Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger\\_User\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger_User_Guide.pdf)
- Debugger API Guide  
[https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger\\_API\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger_API_Guide.pdf)

### 3.1.3.8 OpenMP Documentation

- Updated OpenMP documentation  
[https://rocm-docs.amd.com/en/latest/Programming\\_Guides/openmp\\_support.html](https://rocm-docs.amd.com/en/latest/Programming_Guides/openmp_support.html)

### 3.1.3.9 AMD ROCm General Documentation Links

- For AMD ROCm documentation, see  
<https://rocm-docs.amd.com/en/latest/>
- For installation instructions on supported platforms, see  
[https://rocm-docs.amd.com/en/latest/Installation\\_Guide/Installation-Guide.html](https://rocm-docs.amd.com/en/latest/Installation_Guide/Installation-Guide.html)
- For AMD ROCm binary structure, see  
[https://rocm-docs.amd.com/en/latest/Installation\\_Guide/Software-Stack-for-AMD-GPU.html](https://rocm-docs.amd.com/en/latest/Installation_Guide/Software-Stack-for-AMD-GPU.html)
- For AMD ROCm release history, see  
[https://rocm-docs.amd.com/en/latest/Current\\_Release\\_Notes/ROCm-Version-History.html](https://rocm-docs.amd.com/en/latest/Current_Release_Notes/ROCm-Version-History.html)

## 3.1.4 What's New in This Release

### 3.1.4.1 HIP Enhancements

The ROCm v4.5 release consists of the following HIP enhancements:

#### 3.1.4.1.1 HIP Direct Dispatch

The conventional producer-consumer model where the host thread(producer) enqueues commands to a command queue (per stream), which is then processed by a separate, per-stream worker thread (consumer) created by the runtime, is no longer applicable.

In this release, for Direct Dispatch, the runtime directly queues a packet to the AQL queue (user mode queue to GPU) in Dispatch and some of the synchronization. This new functionality indicates the total latency of the HIP Dispatch API and the latency to launch the first wave on the GPU.

In addition, eliminating the threads in runtime has reduced the variance in the dispatch numbers as the thread scheduling delays and atomics/locks synchronization latencies are reduced.

This feature can be disabled by setting the following environment variable,

`AMD_DIRECT_DISPATCH=0`

### 3.1.4.1.2 Support for HIP Graph

ROCm v4.5 extends support for HIP Graph. For details, refer to the HIP API Guide at, <https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD-HIP-API-4.5.pdf>

### 3.1.4.1.3 Enhanced *launch\_bounds* Check Error Log Message

When a kernel is launched with HIP APIs, for example, `hipModuleLaunchKernel()`, HIP validates to check that input kernel dimension size is not larger than specified `launch_bounds`.

If exceeded, HIP returns launch failure if `AMD_LOG_LEVEL` is set with the proper value. Users can find more information in the error log message, including launch parameters of kernel dim size, launch bounds, and the name of the faulting kernel. It is helpful to figure out the faulting kernel. Besides, the kernel dim size and launch bounds values will also assist in debugging such failures.

For more details, refer to the HIP Programming Guide at

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Programming\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Programming_Guide.pdf)

### 3.1.4.1.4 HIP Runtime Compilation

HIP now supports runtime compilation (`hipRTC`), the usage of which will provide the possibility of optimizations and performance improvement compared with other APIs via regular offline static compilation.

`hipRTC` APIs accept HIP source files in character string format as input parameters and create handles of programs by compiling the HIP source files without spawning separate processes.

For more details on `hipRTC` APIs, refer to the HIP API Guide at

<https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD-HIP-API-4.5.pdf>

### 3.1.4.1.5 New Flag for Backwards Compatibility on float/double `atomicAdd` Function

In the ROCm4.5 release, a new compilation flag is introduced as an option in the CMAKE file. This flag ensures backwards compatibility in float/double `atomicAdd` functions.

```
\__HIP_USE_CMPXCHG_FOR_FP_ATOMICS
```

This compilation flag is not set(“0”) by default, so the HIP runtime uses the current float/double `atomicAdd` functions.

If this compilation flag is set to “1” with the CMAKE option, the existing float/double `atomicAdd` functions is used for compatibility with compilers that do not support floating point atomics.

```
D__HIP_USE_CMPXCHG_FOR_FP_ATOMICS=1
```

For details on how to build the HIP runtime, refer to the HIP Programming Guide at

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Programming\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Programming_Guide.pdf)

### 3.1.4.1.6 Updated HIP Version Definition

The HIP version definition is updated as follows:

```
HIP_VERSION=HIP_VERSION_MAJOR \* 10000000 + HIP_VERSION_MINOR \* 100000  
+ HIP_VERSION_PATCH)
```

The HIP version can be queried from the following HIP API call,

```
hipRuntimeGetVersion(&runtimeVersion);
```

The version returned is always greater than the versions in the previous ROCm releases.

**Note:** The version definition of the HIP runtime is different from that of CUDA. The function returns the HIP runtime version on the AMD platform, while on the NVIDIA platform, it returns the CUDA runtime version. There is no mapping or a correlation between the HIP and CUDA versions.

### 3.1.4.1.7 Planned HIP Enhancements and Fixes

#### 3.1.4.1.7.1 Changes to *hiprtc* implementation to match *nVRTC* behavior

In this release, there are changes to the *hiprtc* implementation to match the *nVRTC* behavior.

**Impact:** Applications can no longer explicitly include HIP runtime header files. Minor code changes are required to remove the HIP runtime header files.

#### 3.1.4.1.7.2 HIP device attribute enumeration

In a future release, there will be a breaking change in the HIP device attribute enumeration. Enum values are being rearranged to accommodate future enhancements and additions.

**Impact:** This will require users to rebuild their applications. No code changes are required.

#### 3.1.4.1.7.3 Changes to behavior of *hipGetLastError()* and *hipPeekAtLastError()* to match CUDA behavior available

In a later release, changes to behavior of *hipGetLastError()* and *hipPeekAtLastError()* to match CUDA behavior will be available.

**Impact:** Applications relying on the previous behavior will be impacted and may require some code changes.

### 3.1.4.2 Unified Memory Support in ROCm

Unified memory allows applications to map and migrate data between CPU and GPU seamlessly without explicitly copying it between different allocations. This enables a more complete implementation of *hipMallocManaged*, *hipMemAdvise*, *hipMemPrefetchAsync* and related APIs. Without unified memory, these APIs only support system memory. With unified memory, the driver can automatically migrate such memory to GPU memory for faster access.



### 3.1.4.2.1 Supported Operating Systems and Versions

This feature is only supported on recent Linux kernels. Currently, it works on Ubuntu versions with 5.6 or newer kernels and the DKMS driver from ROCm. Current releases of RHEL and SLES do not support this feature yet. Future releases of those distributions will add support for this. The unified memory feature is also supported in the KFD driver included with upstream kernels starting from Linux 5.14.

Unified memory only works on GFXv9 and later GPUs, including Vega10 and MI100. Fiji, Polaris and older GPUs are not supported. To check whether unified memory is enabled, look in the kernel log for this message:

```
$ dmesg | grep "HMM registered"
```

If unified memory is enabled, there should be a “message like registered xyzMB device memory”. If unified memory is not supported on your GPU or kernel version, this message is missing.

### 3.1.4.2.2 Unified Memory Support and XNACK

Unified memory support comes in two flavours, XNACK-enabled and XNACK-disabled. XNACK refers to the ability of the GPU to handle page faults gracefully and retry a memory access. In XNACK-enabled mode, the GPU can handle retry after page-faults, which enables mapping and migrating data on demand, as well as memory overcommitment. In XNACK-disabled mode, all memory must be resident and mapped in the GPU page tables when the GPU is executing application code. Any migrations involve temporary preemption of the GPU queues by the driver. Both page fault handling and preemptions, happen automatically and are transparent to the applications.

XNACK-enabled mode only has experimental support. XNACK-enabled mode requires compiling shader code differently. By default, the ROCm compiler builds code that works in both modes. Code can be optimized for one specific mode with compiler options:

OpenCL:

```
clang ... -mcpu=gfx908:**xnack+**sramecc- ... // xnack on, sramecc
off
clang ... -mcpu=gfx908:**xnack-**sramecc+ ... // xnack off, sramecc
on
```

HIP:

```
clang ... --cuda-gpu-arch=gfx906:xnack+ ... // xnack on
clang ... --cuda-gpu-arch=gfx906:xnack- ... // xnack off
```

Not all the math libraries included in ROCm support XNACK-enabled mode on current hardware. Applications will fail to run if their shaders are compiled in the incorrect mode.

On the current hardware, the XNACK mode can be chosen at boot-time by a module parameter `amdgpu.noretry`. The default is XNACK-disabled (`amdgpu.noretry=1`).

### 3.1.4.3 System Management Interface

#### 3.1.4.3.1 Enhanced ROCm SMI *setpoweroverdrive* Functionality

The ROCm System Management Interface (SMI) *setpoweroverdrive* functionality is used to lower the power cap on a device without needing to enable the OverDrive functionality in the driver. Similarly, even with the OverDrive driver functionality enabled, it is possible to request a lower power cap than the card's default.

Currently, any use of the “*setpoweroverdrive\**” functionality in *rocm-smi* prints an out-of-spec warning to the screen and requires the user to agree that using this functionality potentially voids their warranty. However, this warning should only be printed when users are trying to set the power cap to higher than the card's default, which requires the OverDrive driver functionality to be enabled.

For example:

The default power cap is 225.0W before any changes.

```
[atitest@rhel85 smi]$ ./rocm_smi.py --resetpoweroverdrive

===== ROCm System Management Interface
=====

===== Reset GPU Power OverDrive
=====

GPU[0] : Successfully reset Power OverDrive to: 225W

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

Now, after using --setpoweroverdrive to lower the power cap to 123 watts:

[atitest@rhel85 smi]$ ./rocm_smi.py --setpoweroverdrive 123

.. _rocm-system-management-interface-1:

===== ROCm System Management Interface
=====

===== Set GPU Power OverDrive
=====

GPU[0] : Successfully set power to: 123W

.. _end-of-rocm-smi-log-1:

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

Setting a power cap lower than the default of 225.0W (in this case,
123W) does not give a warning.

To verify that the power is set to the correct value:

[atitest@rhel85 smi]$ ./rocm_smi.py --showmaxpower

.. _rocm-system-management-interface-2:
```

(continues on next page)

(continued from previous page)

```
===== ROCm System Management Interface
=====

===== Power Cap =====

GPU[0] : Max Graphics Package Power (W): 123.0

.. _end-of-rocm-smi-log-2:

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

#### 3.1.4.4 OpenMP Enhancements

The ROCm installation includes an LLVM-based implementation, which fully supports OpenMP 4.5 standard and a subset of the OpenMP 5.0 standard. Fortran and C/C++ compilers and corresponding runtime libraries are included. Along with host APIs, the OpenMP compilers support offloading code and data onto GPU devices.

For more information, refer to

[https://rocmdocs.amd.com/en/latest/Programming\\_Guides/openmp\\_support.html](https://rocmdocs.amd.com/en/latest/Programming_Guides/openmp_support.html)

#### 3.1.5 ROCm Math and Communication Libraries

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

Library	Changes
rocBLAS	<p><b>Optimizations</b></p> <ul style="list-style-type: none"> <li>• Improved performance of non-batched and batched syr for all sizes and data types</li> <li>• Improved performance of non-batched and batched hemv for all sizes and data types</li> <li>• Improved performance of non-batched and batched symv for all sizes and data types</li> <li>• Improved memory utilization in rocbblas-bench, rocbblas-test gemm functions, increasing possible runtime sizes.</li> </ul> <p><b>Changes</b></p> <ul style="list-style-type: none"> <li>• Update from C++14 to C++17.</li> <li>• Packaging split into a runtime package (called rocbblas) and a development package (called rocbblas-dev for .deb packages, and rocbblas-devel for .rpm packages). The development package depends on runtime. The runtime package suggests the development package for all supported OSes except CentOS 7 to aid in the transition. The ‘suggests’ feature in packaging is a transitional feature and will be removed in a future ROCm release.</li> </ul> <p><b>Fixed</b></p> <ul style="list-style-type: none"> <li>• For function geam avoid overflow in offset calculation.</li> <li>• For function syr avoid overflow in offset calculation.</li> <li>• For function gemv (Transpose-case) avoid overflow in offset calculation.</li> <li>• For functions ssyrk and dsyrk, allow conjugate-transpose case to match legacy BLAS. Behavior is the same as the transpose case.</li> </ul>
hipBLAS	<p><b>Added</b></p> <ul style="list-style-type: none"> <li>• More support for hipblas-bench</li> </ul> <p><b>Fixed</b></p> <ul style="list-style-type: none"> <li>• Avoid large offset overflow for gemv and hemv in hipblas-test</li> </ul> <p><b>Changed</b></p> <ul style="list-style-type: none"> <li>• Packaging split into a runtime package called hipblas and a development package called hipblas-devel. The development package depends on runtime. The runtime package suggests the development package for all supported OSes except CentOS 7 to aid in the transition. The ‘suggests’ feature in packaging is a transitional feature and will be removed in a future rocm release.</li> </ul>
rocFFT	<p><b>Optimizations</b></p> <ul style="list-style-type: none"> <li>• Optimized SBCC kernels of length 52, 60, 72, 80, 84, 96, 104, 108, 112, 160, 168, 208, 216, 224, 240 with new kernel generator.</li> </ul> <p><b>Added</b></p> <ul style="list-style-type: none"> <li>• Split 2D device code into separate libraries.</li> </ul>
16	<p><b>Changed</b>      <b>Chapter 3. ROCm Learning Center</b></p> <ul style="list-style-type: none"> <li>• Packaging split into a runtime package called rocfft and a development package called rocfft-devel. The development package depends on runtime. The runtime package suggests the development package for all supported OSes except CentOS 7 to aid in the transition. The ‘suggests’ feature in packaging is a transitional feature and will be removed in a future rocm release.</li> </ul>

For more information about ROCm Libraries, refer to the documentation at [https://rocmdocs.amd.com/en/latest/ROCm\\_Libraries/ROCm\\_Libraries.html](https://rocmdocs.amd.com/en/latest/ROCm_Libraries/ROCm_Libraries.html)

### 3.1.6 Known Issues in This Release

The following are the known issues in this release.

#### 3.1.6.1 Cache Issues with ROCProfiler

When the same kernel is launched back-to-back multiple times on a GPU, a cache flush is executed each time the kernel finishes when profiler data is collected. The cache flush is inserted by ROCprofiler for each kernel. This prevents kernel from being cached, instead it is being read each time it is launched. As a result the cache hit rate from rocprofiler is reported as 0% or very low.

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

#### 3.1.6.2 Compiler Support for Function Pointers and Virtual Functions

A known issue in the compiler support for function pointers and virtual functions on the GPU may cause undefined behavior due to register corruption.

A temporary workaround is to compile the affected application with

```
-mllvm -amdgpu-fixed-function-abi=1* option
```

**Note:** This is an internal compiler flag and may be removed without notice once the issue is addressed in a future release.

#### 3.1.6.3 Debugger Process Exit May Cause ROCgdb Internal Error

If the debugger process exits during debugging, ROCgdb may report internal errors. This issue occurs as it attempts to access the AMD GPU state for the exited process. To recover, users must restart ROCgdb.

As a workaround, users can set breakpoints to prevent the debugged process from exiting. For example, users can set breakpoints at the last statement of the main function and in the abort() and exit() functions. This temporary solution allows the application to be re-run without restarting ROCgdb.

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

For more information, refer to the ROCgdb User Guide at,

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_ROCDebugger\\_User\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_ROCDebugger_User_Guide.pdf)

#### 3.1.6.4 clinfo and rocminfo Do Not Display Marketing Name

clinfo and rocminfo display a blank field for Marketing Name.

This is due to a missing package that is not yet available from ROCm. This package will be distributed in future ROCm releases.

### **3.1.6.5 Stability Issue on LAMMPS-KOKKOS Applications**

On mGPU machines, lammmps-kokkos applications experience a stability issue (AMD Instinct MI100™).

As a workaround, perform a Translation LookAside Buffer (TLB) flush.

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

## **3.1.7 Deprecations**

### **3.1.7.1 AMD Instinct MI25 End of Life**

ROCm release v4.5 is the final release to support AMD Instinct MI25. AMD Instinct MI25 has reached End of Life (EOL). ROCm 4.5 represents the last certified release for software and driver support. AMD will continue to provide technical support and issue resolution for AMD Instinct MI25 on ROCm v4.5 for a period of 12 months from the software GA date.

### **3.1.7.2 Planned Deprecation for Code Object Versions 2 AND 3**

With the ROCm v4.5 release, the generation of code object versions 2 and 3 is being deprecated and may be removed in a future release. This deprecation notice does not impact support for the execution of AMD GPU code object versions.

The `-mcode-object-version` Clang option can be used to instruct the compiler to generate a specific AMD GPU code object version. In ROCm v4.5, the compiler can generate AMD GPU code object version 2, 3, and 4, with version 4 being the default if not specified.

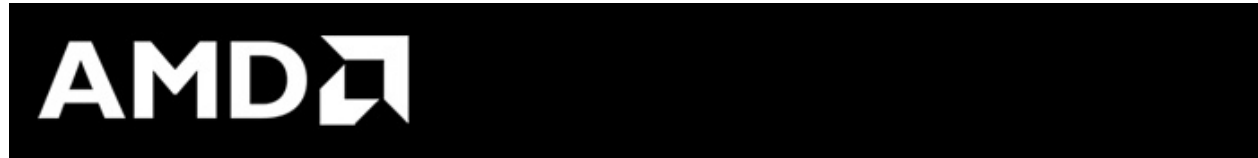
## **3.1.8 DISCLAIMER**

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes. THIS INFORMATION IS PROVIDED ‘AS IS.’ AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. AMD, the AMD Arrow logo, [insert all other AMD trademarks used in the material here per AMD Trademarks] and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies. [Insert any third party trademark attribution here per AMD’s Third Party Trademark List.] © [Insert year written\*] Advanced Micro Devices, Inc. All rights reserved.

Third-party Disclaimer

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED “AS IS” WITHOUT A WARRANTY OF ANY

KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.



## 3.2 Deprecations

### 3.2.1 ROCm Release v4.5

#### 3.2.1.1 AMD Instinct MI25 End of Life

ROCm release v4.5 is the final release to support AMD Instinct MI25. AMD Instinct MI25 has reached End of Life (EOL). ROCm 4.5 represents the last certified release for software and driver support. AMD will continue to provide technical support and issue resolution for AMD Instinct MI25 on ROCm v4.5 for a period of 12 months from the software GA date.

#### 3.2.1.2 Planned Deprecation for Code Object Versions 2 AND 3

With the ROCm v4.5 release, the generation of code object versions 2 and 3 is being deprecated and may be removed in a future release. This deprecation notice does not impact support for the execution of AMD GPU code object versions.

The `-mcode-object-version` Clang option can be used to instruct the compiler to generate a specific AMD GPU code object version. In ROCm v4.5, the compiler can generate AMD GPU code object version 2, 3, and 4, with version 4 being the default if not specified.

### 3.2.2 ROCm Release v4.1

#### 3.2.2.1 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.

#### 3.2.2.2 Changed HIP Environment Variables in ROCm v4.1 Release

In the ROCm v3.5 release, the Heterogeneous Compute Compiler (HCC) compiler was deprecated, and the HIP-Clang compiler was introduced for compiling Heterogeneous-Compute Interface for Portability (HIP) programs. Also, the HIP runtime API was implemented on top of the Radeon Open Compute Common Language runtime (ROCclr). ROCclr is an abstraction layer that provides the ability to interact with different runtime backends such as ROCr.

While the `HIP_PLATFORM=hcc` environment variable was functional in subsequent releases after ROCm v3.5, in the ROCm v4.1 release, changes to the following environment variables were implemented:

- `HIP_PLATFORM=hcc` was changed to `HIP_PLATFORM=amd`

- *HIP\_PLATFORM=nvcc was changed to HIP\_PLATFORM=nvidia*

Therefore, any applications continuing to use the HIP\_PLATFORM=hcc environment variable will fail.

**Workaround:** Update the environment variables to reflect the changes mentioned above.

### 3.2.3 ROCm Release v4.0

#### 3.2.3.1 ROCr Runtime Deprecations

The following ROCr Runtime enumerations, functions, and structs are deprecated in the AMD ROCm v4.0 release.

Deprecated ROCr Runtime Functions

- hsa\_isa\_get\_info
- hsa\_isa\_compatible
- hsa\_executable\_create
- hsa\_executable\_get\_symbol
- hsa\_executable\_iterate\_symbols
- hsa\_code\_object\_serialize
- hsa\_code\_object\_deserialize
- hsa\_code\_object\_destroy
- hsa\_code\_object\_get\_info
- hsa\_executable\_load\_code\_object
- hsa\_code\_object\_get\_symbol
- hsa\_code\_object\_get\_symbol\_from\_name
- hsa\_code\_symbol\_get\_info
- hsa\_code\_object\_iterate\_symbols

#### 3.2.3.2 Deprecated ROCr Runtime Enumerations

- HSA\_ISA\_INFO\_CALL\_CONVENTION\_COUNT
- HSA\_ISA\_INFO\_CALL\_CONVENTION\_INFO\_WAVEFRONT\_SIZE
- HSA\_ISA\_INFO\_CALL\_CONVENTION\_INFO\_WAVEFRONTS\_PER\_COMPUTE\_UNIT
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_MODULE\_NAME\_LENGTH
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_MODULE\_NAME
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_AGENT
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_VARIABLE\_ALLOCATION
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_VARIABLE\_SEGMENT
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_VARIABLE\_ALIGNMENT
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_VARIABLE\_SIZE
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_VARIABLE\_IS\_CONST



- HSA\_EXECUTABLE\_SYMBOL\_INFO\_KERNEL\_CALL\_CONVENTION
- HSA\_EXECUTABLE\_SYMBOL\_INFO\_INDIRECT\_FUNCTION\_CALL\_CONVENTION
  - hsa\_code\_object\_type\_t
  - hsa\_code\_object\_info\_t
  - hsa\_code\_symbol\_info\_t

### 3.2.3.3 Deprecated ROCr Runtime Structs

- hsa\_code\_object\_t
- hsa\_callback\_data\_t
- hsa\_code\_symbol\_t

### 3.2.3.4 AOMP DEPRECATION

As of AMD ROCm v4.0, AOMP (aomp-amdgpu) is deprecated. OpenMP support has moved to the openmp-extras auxiliary package, which leverages the ROCm compiler on LLVM 12.

For more information, refer to

[https://rocmdocs.amd.com/en/latest/Programming\\_Guides/openmp\\_support.html](https://rocmdocs.amd.com/en/latest/Programming_Guides/openmp_support.html)

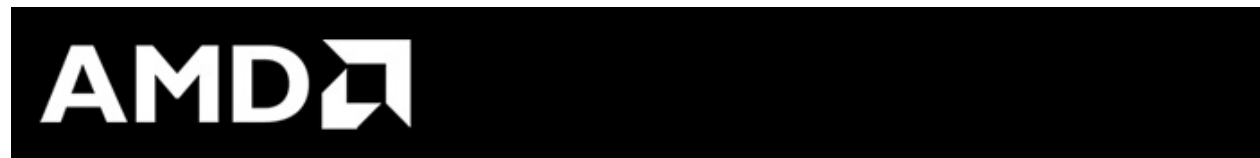
## 3.2.4 ROCm Release v3.5

### 3.2.4.1 Heterogeneous Compute Compiler

In the ROCm v3.5 release, the Heterogeneous Compute Compiler (HCC) compiler was deprecated and the HIP-Clang compiler was introduced for compiling Heterogeneous-Compute Interface for Portability (HIP) programs.

For more information, download the HIP Programming Guide at:

<https://github.com/RadeonOpenCompute/ROCm>



## 3.3 AMD ROCm Version History

This file contains historical information for ROCm releases.

### 3.3.1 New features and enhancements in ROCm v4.3

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-4.3.x>

- HIP Versioning Update
- Kernel Enqueue Serialization
- NUMA-aware Host Memory Allocation
- New Atomic System Scope Atomic Operations
- Indirect Function Call and C++ Virtual Functions
- Prometheus (Grafana) Integration with Automatic Node Detection
- Coarse Grain Utilization
- Add 64-bit Energy Accumulator In-band
- Support for Continuous Clocks Values
- Memory Utilization Counters
- Performance Determinism
- HBM Temperature Metric Per Stack
- Tracing Multiple MPI Ranks
- ROCm Math and Communication Libraries Enhancements and Fixes

### 3.3.2 New features and enhancements in ROCm v4.2

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-4.2.x>

- HIP Target Platform Macro
- Updated HIP 'Include' Directories
- HIP Stream Memory Operations
- HIP Events in Kernel Dispatch
- Changed Environment Variables for HIP
- ROCm Data Center Tool - RAS Integration
- ROCm Math and Communication Libraries Enhancements and Fixes

### 3.3.3 New features and enhancements in ROCm v4.1

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-4.1.x>

- TargetID for Multiple Configurations
- Grafana Integration in ROCm Data Center Tool
- ROCm Math and Communication Libraries Enhancements and Fixes
- HIP Enhancements
- OpenMP Enhancements and Fixes
- MIOpen Tensile Integration

### 3.3.4 New features and enhancements in ROCm v4.0

Release notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-4.0.x>

- Introducing AMD Instinct™ MI100 accelerator
- Important features of the AMD Instinct™ MI100 accelerator
- Matrix Core Engines and GFX908 Considerations
- RAS (Reliability, Availability, and Accessibility) features
- Using CMake with AMD ROCm
- AMD ROCm and MESA Multimedia
- Support for Printing PCIe Information on AMD Instinct™100
- New API for xGMI

### 3.3.5 New features and enhancements in ROCm v3.10

Release notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.10.x>

- Prometheus Plugin for ROCm Data Center Tool
- Python Binding
- System DMA (SDMA) Utilization
- ROCm-SMI Command Line Interface
- Enhanced ROCm SMI Library for Events
- ROCm SMI – Command Line Interface Hardware Topology
- New rocSOLVER APIs
- RCCL Alltoallv Support in PyTorch
- AOMP Release 11.11-0

### 3.3.6 New features and enhancements in ROCm v3.9

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/blob/roc-3.9.x/README.md>

- Compiler support for OpenMP
- ROCm-SMI Hardware Topology
- Compute Unit Occupancy
- Accessing Compute Unit Occupancy Directly Using SYSFS
- ‘rocfft\_execution\_info\_set\_stream’ API
- Improved GEMM Performance
- New Matrix Pruning Functions
- AOMP v11.9-0
- AOMP v11.08-0

### 3.3.7 New features and enhancements in ROCm v3.8

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.8.x>

- Hipfort-Interface for GPU Kernel Libraries
- ROCm Data Center Tool
- Error-Correcting Code Fields in ROCm Data Center Tool
- Static Linking Libraries

### 3.3.8 New features and enhancements in ROCm v3.7

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.7.x>

- AOMP Enhancements
- Compatibility with NVIDIA Communications Collective Library v2.7 API
- Singular Value Decomposition of Bi-diagonal Matrices
- rocSPARSE\_gemmi() Operations for Sparse Matrices

### 3.3.9 Patch Release - ROCm v3.5.1

AMD ROCm released a maintenance patch release v3.5.1. For more information about the release see,

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.5.1>

### 3.3.10 New features and enhancements in ROCm v3.5

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.5.0>

**rocProf Command Line Tool Python Requirement** SQLite3 is a required Python module for the rocprof command-line tool. You can install the SQLite3 Python module using the pip utility and set env var ROC\_PYTHON\_VERSION to the Python version, which includes the SQLite3 module.

**Heterogeneous-Compute Interface for Portability** In this release, the Heterogeneous Compute Compiler (HCC) compiler is deprecated and the HIP-Clang compiler is introduced for compiling Heterogeneous-Compute Interface for Portability (HIP) programs.

**Radeon Open Compute Common Language Runtime** In this release, the HIP runtime API is implemented on top of Radeon Open Compute Common Language Runtime (ROCclr). ROCclr is an abstraction layer that provides the ability to interact with different runtime backends such as ROCr.

**OpenCL Runtime** The following OpenCL runtime changes are made in this release:

-AMD ROCm OpenCL Runtime extends support to OpenCL2.2 -The developer branch is changed from master to master-next

**AMD ROCm GNU Debugger (ROCgdb)** The AMD ROCm Debugger (ROCgdb) is the AMD ROCm source-level debugger for Linux based on the GNU Debugger (GDB). It enables heterogeneous debugging on the AMD ROCm platform of an x86-based host architecture along with AMD GPU architectures and supported by the AMD Debugger API Library (ROCdbgapi).

**AMD ROCm Debugger API Library** The AMD ROCm Debugger API Library (ROCdbgapi) implements an AMD GPU debugger application programming interface (API) that provides the support necessary for a client of the library to control the execution and inspect the state of AMD GPU devices.

**rocProfiler Dispatch Callbacks Start Stop API** In this release, a new rocprofiler start/stop API is added to enable/disable GPU kernel HSA dispatch callbacks. The callback can be registered with the ‘rocprofiler\_set\_hsa\_callbacks’ API. The API helps you eliminate some profiling performance impact by invoking the profiler only for kernel dispatches of interest. This optimization will result in significant performance gains.

**ROCm Communications Collective Library** The ROCm Communications Collective Library (RCCL) consists of the following enhancements:

- Re-enable target 0x803
- Build time improvements for the HIP-Clang compiler

**NVIDIA Communications Collective Library Version Compatibility** AMD RCCL is now compatible with NVIDIA Communications Collective Library (NCCL) v2.6.4 and provides the following features:

Network interface improvements with API v3  
Network topology detection  
Improved CPU type detection  
Infiniband adaptive routing support

**MIOpen Optional Kernel Package Installation** MIOpen provides an optional pre-compiled kernel package to reduce startup latency.

**New SMI Event Interface and Library** An SMI event interface is added to the kernel and ROCm SMI lib for system administrators to get notified when specific events occur. On the kernel side, AMDKFD\_IOC\_SMI\_EVENTS input/output control is enhanced to allow notifications propagation to user mode through the event channel.

**API for CPU Affinity** A new API is introduced for aiding applications to select the appropriate memory node for a given accelerator(GPU).

**Radeon Performance Primitives Library** The new Radeon Performance Primitives (RPP) library is a comprehensive high-performance computer vision library for AMD (CPU and GPU) with the HIP and OpenCL backend. The target operating system is Linux.

### 3.3.11 New features and enhancements in ROCm v3.3

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.3.0>

**Multi-Version Installation** Users can install and access multiple versions of the ROCm toolkit simultaneously. Previously, users could install only a single version of the ROCm toolkit.

**GPU Process Information** A new functionality to display process information for GPUs is available in this release. For example, you can view the process details to determine if the GPU(s) must be reset.

**Support for 3D Pooling Layers** AMD ROCm is enhanced to include support for 3D pooling layers. The implementation of 3D pooling layers now allows users to run 3D convolutional networks, such as ResNext3D, on AMD Radeon Instinct GPUs.

**ONNX Enhancements** Open Neural Network eXchange (ONNX) is a widely-used neural net exchange format. The AMD model compiler & optimizer support the pre-trained models in ONNX, NNEF, & Caffe formats. Currently, ONNX versions 1.3 and below are supported.

### 3.3.12 New features and enhancements in ROCm v3.2

This release was not productized.

### 3.3.13 New features and enhancements in ROCm v3.1

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.1.0>

#### Change in ROCm Installation Directory Structure

A fresh installation of the ROCm toolkit installs the packages in the `/opt/rocm-<version>` folder. Previously, ROCm toolkit packages were installed in the `/opt/rocm` folder.

#### Reliability, Accessibility, and Serviceability Support for Vega 7nm

The Reliability, Accessibility, and Serviceability (RAS) support for Vega7nm is now available.

#### SLURM Support for AMD GPU

SLURM (Simple Linux Utility for Resource Management) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.

### 3.3.14 New features and enhancements in ROCm v3.0

Release Notes: <https://github.com/RadeonOpenCompute/ROCm/tree/roc-3.0.0>

- Support for CentOS RHEL v7.7
- Support is extended for CentOS/RHEL v7.7 in the ROCm v3.0 release. For more information about the CentOS/RHEL v7.7 release, see:
- CentOS/RHEL
- Initial distribution of AOMP 0.7-5 in ROCm v3.0

The code base for this release of AOMP is the Clang/LLVM 9.0 sources as of October 8th, 2019. The LLVM-project branch used to build this release is AOMP-191008. It is now locked. With this release, an artifact tarball of the entire source tree is created. This tree includes a Makefile in the root directory used to build AOMP from the release tarball. You can use Spack to build AOMP from this source tarball or build manually without Spack.

- Fast Fourier Transform Updates

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform. Fast Fourier transforms are used in signal processing, image processing, and many other areas. The following real FFT performance change is made in the ROCm v3.0 release:

- Implement efficient real/complex 2D transforms for even lengths.

Other improvements:

- More 2D test coverage sizes.
- Fix buffer allocation error for large 1D transforms.
- C++ compatibility improvements.

MemCopy Enhancement for rocProf In the v3.0 release, the rocProf tool is enhanced with an additional capability to dump asynchronous GPU memcpy information into a .csv file. You can use the `-hsa-trace` option to create the `results_mcopy.csv` file. Future enhancements will include column labels.

### 3.3.15 New features and enhancements in ROCm v2.10

#### rocBLAS Support for Complex GEMM

The rocBLAS library is a gpu-accelerated implementation of the standard Basic Linear Algebra Subroutines (BLAS). rocBLAS is designed to enable you to develop algorithms, including high performance computing, image analysis, and machine learning.

In the AMD ROCm release v2.10, support is extended to the General Matrix Multiply (GEMM) routine for multiple small matrices processed simultaneously for rocBLAS in AMD Radeon Instinct MI50. Both single and double precision, CGEMM and ZGEMM, are now supported in rocBLAS.

#### Support for SLES 15 SP1

In the AMD ROCm v2.10 release, support is added for SUSE Linux® Enterprise Server (SLES) 15 SP1. SLES is a modular operating system for both multimodal and traditional IT.

#### Code Marker Support for rocProfiler and rocTracer Libraries

Code markers provide the external correlation ID for the calling thread. This function indicates that the calling thread is entering and leaving an external API region.

### 3.3.16 New features and enhancements in ROCm 2.9

#### Initial release for Radeon Augmentation Library(RALI)

The AMD Radeon Augmentation Library (RALI) is designed to efficiently decode and process images from a variety of storage formats and modify them through a processing graph programmable by the user. RALI currently provides C API.

#### Quantization in MIGraphX v0.4

MIGraphX 0.4 introduces support for fp16 and int8 quantization. For additional details, as well as other new MIGraphX features, see MIGraphX documentation.

#### rocSparse csrgermm

csrgermm enables the user to perform matrix-matrix multiplication with two sparse matrices in CSR format.

#### Singularity Support

ROCm 2.9 adds support for Singularity container version 2.5.2.

#### Initial release of rocTX

ROCm 2.9 introduces rocTX, which provides a C API for code markup for performance profiling. This initial release of rocTX supports annotation of code ranges and ASCII markers.

- Added support for Ubuntu 18.04.3
- Ubuntu 18.04.3 is now supported in ROCm 2.9.

### 3.3.17 New features and enhancements in ROCm 2.8

Support for NCCL2.4.8 API

Implements `ncclCommAbort()` and `ncclCommGetAsyncError()` to match the NCCL 2.4.x API

### 3.3.18 New features and enhancements in ROCm 2.7.2

This release is a hotfix for ROCm release 2.7.

### 3.3.19 Issues fixed in ROCm 2.7.2

- A defect in upgrades from older ROCm releases has been fixed.
- `rocprofiler -hiptrace` and `-hsatrace` fails to load `roctracer` library
- In ROCm 2.7.2, `rocprofiler -hiptrace` and `-hsatrace` fails to load `roctracer` library defect has been fixed.
- To generate traces, please provide directory path also using the parameter: `-d <directoryPath>` for example:

`/opt/rocm/bin/rocprof -hsa-trace -d $PWD/traces /opt/rocm/hip/samples/0_Intro/bit_extract/bit_extract` All traces and results will be saved under `$PWD/traces` path

### 3.3.20 Upgrading from ROCm 2.7 to 2.7.2

To upgrade, please remove 2.7 completely as specified for ubuntu or for centos/rhel, and install 2.7.2 as per instructions install instructions

Other notes To use `rocprofiler` features, the following steps need to be completed before using `rocprofiler`:

Step-1: Install `roctracer` Ubuntu 16.04 or Ubuntu 18.04: `sudo apt install roctracer-dev` CentOS/RHEL 7.6: `sudo yum install roctracer-dev`

Step-2: Add `/opt/rocm/roctracer/lib` to `LD_LIBRARY_PATH` New features and enhancements in ROCm 2.7 [rocFFT] Real FFT Functional Improved real/complex 1D even-length transforms of unit stride. Performance improvements of up to 4.5x are observed. Large problem sizes should see approximately 2x.

rocRand Enhancements and Optimizations

Added support for new datatypes: `uchar`, `ushort`, `half`.

Improved performance on “Vega 7nm” chips, such as on the Radeon Instinct MI50

`mtgp32` uniform double performance changes due generation algorithm standardization. Better quality random numbers now generated with 30% decrease in performance

Up to 5% performance improvements for other algorithms

RAS

Added support for RAS on Radeon Instinct MI50, including:

- Memory error detection
- Memory error detection counter
- ROCm-SMI enhancements
- Added ROCm-SMI CLI and LIB support for FW version, compute running processes, utilization rates, utilization counter, link error counter, and unique ID.



### 3.3.21 New features and enhancements in ROCm 2.6

#### ROCmInfo enhancements

ROCmInfo was extended to do the following: For ROCr API call errors including initialization determine if the error could be explained by:

ROCK (driver) is not loaded / available User does not have membership in appropriate group - “video” If not above print the error string that is mapped to the returned error code If no error string is available, print the error code in hex Thrust - Functional Support on Vega20

ROCm2.6 contains the first official release of rocThrust and hipCUB. rocThrust is a port of thrust, a parallel algorithm library. hipCUB is a port of CUB, a reusable software component library. Thrust/CUB has been ported to the HIP/ROCm platform to use the rocPRIM library. The HIP ported library works on HIP/ROCm platforms.

Note: rocThrust and hipCUB library replaces <https://github.com/ROCmSoftwarePlatform/thrust> (hip-thrust), i.e. hip-thrust has been separated into two libraries, rocThrust and hipCUB. Existing hip-thrust users are encouraged to port their code to rocThrust and/or hipCUB. Hip-thrust will be removed from official distribution later this year.

#### MIGraphX v0.3

MIGraphX optimizer adds support to read models frozen from Tensorflow framework. Further details and an example usage at <https://github.com/ROCmSoftwarePlatform/AMDMIGraphX/wiki/Getting-started:-using-the-new-features-of-MIGraphX-0.3>

#### MIOpen 2.0

This release contains several new features including an immediate mode for selecting convolutions, bfloat16 support, new layers, modes, and algorithms.

MIOpenDriver, a tool for benchmarking and developing kernels is now shipped with MIOpen. BFloat16 now supported in HIP requires an updated rocBLAS as a GEMM backend.

Immediate mode API now provides the ability to quickly obtain a convolution kernel.

MIOpen now contains HIP source kernels and implements the ImplicitGEMM kernels. This is a new feature and is currently disabled by default. Use the environmental variable “MIOpen\_DEBUG\_CONV\_IMPLICIT\_GEMM=1” to activation this feature. ImplicitGEMM requires an up to date HIP version of at least 1.5.9211.

A new “loss” category of layers has been added, of which, CTC loss is the first. See the API reference for more details. 2.0 is the last release of active support for gfx803 architectures. In future releases, MIOpen will not actively debug and develop new features specifically for gfx803.

System Find-Db in memory cache is disabled by default. Please see build instructions to enable this feature. Additional documentation can be found here: <https://rocmsoftwareplatform.github.io/MIOpen/doc/html/>

#### Bfloat16 software support in rocBLAS/Tensile

Added mixed precision bfloat16/IEEE f32 to gemm\_ex. The input and output matrices are bfloat16. All arithmetic is in IEEE f32.

#### AMD Infinity Fabric™ Link enablement

The ability to connect four Radeon Instinct MI60 or Radeon Instinct MI50 boards in two hives or two Radeon Instinct MI60 or Radeon Instinct MI50 boards in four hives via AMD Infinity Fabric™ Link GPU interconnect technology has been added.

#### ROCm-smi features and bug fixes

mGPU & Vendor check

Fix clock printout if DPM is disabled

Fix finding marketing info on CentOS

Clarify some error messages

ROCm-smi-lib enhancements

Documentation updates

Improvements to `*name_get` functions

RCCL2 Enablement

RCCL2 supports collectives intranode communication using PCIe, Infinity Fabric™, and pinned host memory, as well as internode communication using Ethernet (TCP/IP sockets) and Infiniband/RoCE (Infiniband Verbs). Note: For Infiniband/RoCE, RDMA is not currently supported.

rocFFT enhancements

Added: Debian package with FFT test, benchmark, and sample programs Improved: hipFFT interfaces Improved: rocFFT CPU reference code, plan generation code and logging code

### 3.3.22 New features and enhancements in ROCm 2.5

UCX 1.6 support

Support for UCX version 1.6 has been added.

BFloat16 GEMM in rocBLAS/Tensile

Software support for BFloat16 on Radeon Instinct MI50, MI60 has been added. This includes:

Mixed precision GEMM with BFloat16 input and output matrices, and all arithmetic in IEEE32 bit

Input matrix values are converted from BFloat16 to IEEE32 bit, all arithmetic and accumulation is IEEE32 bit. Output values are rounded from IEEE32 bit to BFloat16

Accuracy should be correct to 0.5 ULP

ROCm-SMI enhancements

CLI support for querying the memory size, driver version, and firmware version has been added to ROCm-smi.

[PyTorch] multi-GPU functional support (CPU aggregation/Data Parallel)

Multi-GPU support is enabled in PyTorch using Dataparallel path for versions of PyTorch built using the 06c8aa7a3bbd91cda2fd6255ec82aad21fa1c0d5 commit or later.

rocSparse optimization on Radeon Instinct MI50 and MI60

This release includes performance optimizations for csrsv routines in the rocSparse library.

[Thrust] Preview

Preview release for early adopters. rocThrust is a port of thrust, a parallel algorithm library. Thrust has been ported to the HIP/ROCm platform to use the rocPRIM library. The HIP ported library works on HIP/ROCm platforms.

Note: This library will replace <https://github.com/ROCmSoftwarePlatform/thrust> in a future release. The package for rocThrust (this library) currently conflicts with version 2.5 package of thrust. They should not be installed together.

Support overlapping kernel execution in same HIP stream

HIP API has been enhanced to allow independent kernels to run in parallel on the same stream.

AMD Infinity Fabric™ Link enablement

The ability to connect four Radeon Instinct MI60 or Radeon Instinct MI50 boards in one hive via AMD Infinity Fabric™ Link GPU interconnect technology has been added.

### 3.3.23 New features and enhancements in ROCm 2.4

TensorFlow 2.0 support

ROCm 2.4 includes the enhanced compilation toolchain and a set of bug fixes to support TensorFlow 2.0 features natively

AMD Infinity Fabric™ Link enablement

ROCm 2.4 adds support to connect two Radeon Instinct MI60 or Radeon Instinct MI50 boards via AMD Infinity Fabric™ Link GPU interconnect technology.

### 3.3.24 New features and enhancements in ROCm 2.3

Mem usage per GPU

Per GPU memory usage is added to rocm-smi. Display information regarding used/total bytes for VRAM, visible VRAM and GTT, via the `--showmeminfo` flag

MIVisionX, v1.1 - ONNX

ONNX parser changes to adjust to new file formats

MIGraphX, v0.2

MIGraphX 0.2 supports the following new features:

New Python API

- Support for additional ONNX operators and fixes that now enable a large set of Imagenet models
- Support for RNN Operators
- Support for multi-stream Execution
- [Experimental] Support for Tensorflow frozen protobuf files

See: [Getting-started:-using-the-new-features-of-MIGraphX-0.2](#) for more details

MIOpen, v1.8 - 3d convolutions and int8

This release contains full 3-D convolution support and int8 support for inference. Additionally, there are major updates in the performance database for major models including those found in Torchvision. See: [MIOpen releases](#)

Caffe2 - mGPU support

Multi-gpu support is enabled for Caffe2.

rocTracer library, ROCm tracing API for collecting runtimes API and asynchronous GPU activity traces HIP/HCC domains support is introduced in rocTracer library.

BLAS - Int8 GEMM performance, Int8 functional and performance Introduces support and performance optimizations for Int8 GEMM, implements TRSV support, and includes improvements and optimizations with Tensile.

Prioritized L1/L2/L3 BLAS (functional) Functional implementation of BLAS L1/L2/L3 functions

BLAS - tensile optimization Improvements and optimizations with tensile

MIOpen Int8 support Support for int8

### 3.3.25 New features and enhancements in ROCm 2.2

rocSparse Optimization on Vega20 Cache usage optimizations for csrsv (sparse triangular solve), coomv (SpMV in COO format) and ellmv (SpMV in ELL format) are available.

DGEMM and DTRSM Optimization Improved DGEMM performance for reduced matrix sizes (k=384, k=256)

Caffe2 Added support for multi-GPU training

### 3.3.26 New features and enhancements in ROCm 2.1

RocTracer v1.0 preview release – ‘rocprof’ HSA runtime tracing and statistics support - Supports HSA API tracing and HSA asynchronous GPU activity including kernels execution and memory copy

Improvements to ROCM-SMI tool - Added support to show real-time PCIe bandwidth usage via the -b/-showbw flag

DGEMM Optimizations - Improved DGEMM performance for large square and reduced matrix sizes (k=384, k=256)

### 3.3.27 New features and enhancements in ROCm 2.0

Adds support for RHEL 7.6 / CentOS 7.6 and Ubuntu 18.04.1

Adds support for Vega 7nm, Polaris 12 GPUs

Introduces MIVisionX A comprehensive computer vision and machine intelligence libraries, utilities and applications bundled into a single toolkit. Improvements to ROCm Libraries rocSPARSE & hipSPARSE rocBLAS with improved DGEMM efficiency on Vega 7nm

MIOpen This release contains general bug fixes and an updated performance database Group convolutions backwards weights performance has been improved

RNNs now support fp16 Tensorflow multi-gpu and Tensorflow FP16 support for Vega 7nm TensorFlow v1.12 is enabled with fp16 support PyTorch/Caffe2 with Vega 7nm Support

fp16 support is enabled

Several bug fixes and performance enhancements

Known Issue: breaking changes are introduced in ROCm 2.0 which are not addressed upstream yet. Meanwhile, please continue to use ROCm fork at <https://github.com/ROCmSoftwarePlatform/pytorch>

Improvements to ROCProfiler tool

Support for Vega 7nm

Support for hipStreamCreateWithPriority

Creates a stream with the specified priority. It creates a stream on which enqueued kernels have a different priority for execution compared to kernels enqueued on normal priority streams. The priority could be higher or lower than normal priority streams.

OpenCL 2.0 support

ROCm 2.0 introduces full support for kernels written in the OpenCL 2.0 C language on certain devices and systems. Applications can detect this support by calling the “clGetDeviceInfo” query function with “param\_name” argument set to “CL\_DEVICE\_OPENCL\_C\_VERSION”.

In order to make use of OpenCL 2.0 C language features, the application must include the option “-cl-std=CL2.0” in options passed to the runtime API calls responsible for compiling or building device programs. The complete specification for the OpenCL 2.0 C language can be obtained using the following link: <https://www.khronos.org/registry/OpenCL/specs/opencl-2.0-opencl.pdf>

Improved Virtual Addressing (48 bit VA) management for Vega 10 and later GPUs

Fixes Clang AddressSanitizer and potentially other 3rd-party memory debugging tools with ROCm

Small performance improvement on workloads that do a lot of memory management

Removes virtual address space limitations on systems with more VRAM than system memory Kubernetes support

### **3.3.28 New features and enhancements in ROCm 1.9.2**

RDMA(MPI) support on Vega 7nm

Support ROCnRDMA based on Mellanox InfiniBand

Improvements to HCC

Improved link time optimization

Improvements to ROCProfiler tool

General bug fixes and implemented versioning APIs

New features and enhancements in ROCm 1.9.2

RDMA(MPI) support on Vega 7nm

Support ROCnRDMA based on Mellanox InfiniBand

Improvements to HCC

Improved link time optimization

Improvements to ROCProfiler tool

General bug fixes and implemented versioning APIs

Critical bug fixes

### **3.3.29 New features and enhancements in ROCm 1.9.1**

Added DPM support to Vega 7nm

Dynamic Power Management feature is enabled on Vega 7nm.

Fix for ‘ROCm profiling’ that used to fail with a “Version mismatch between HSA runtime and libhsa-runtime-tools64.so.1” error

### **3.3.30 New features and enhancements in ROCm 1.9.0**

Preview for Vega 7nm Enables developer preview support for Vega 7nm

System Management Interface Adds support for the ROCm SMI (System Management Interface) library, which provides monitoring and management capabilities for AMD GPUs.

Improvements to HIP/HCC Support for gfx906

Added deprecation warning for C++AMP. This will be the last version of HCC supporting C++AMP.

Improved optimization for global address space pointers passing into a GPU kernel

Fixed several race conditions in the HCC runtime

Performance tuning to the unpinned copy engine

Several codegen enhancement fixes in the compiler backend

Preview for rocpf Profiling Tool

Developer preview (alpha) of profiling tool rocProfiler. It includes a command-line front-end, `rpl_run.sh`, which enables:

Cmd-line tool for dumping public per kernel perf-counters/metrics and kernel timestamps

Input file with counters list and kernels selecting parameters

Multiple counters groups and app runs supported

Output results in CSV format

The tool can be installed from the `rocprofiler-dev` package. It will be installed into: `/opt/rocm/bin/rpl_run.sh`

Preview for rocr Debug Agent `rocr_debug_agent`

The ROCr Debug Agent is a library that can be loaded by ROCm Platform Runtime to provide the following functionality:

Print the state for wavefronts that report memory violation or upon executing a “`s_trap 2`” instruction. Allows SIGINT (ctrl c) or SIGTERM (kill -15) to print wavefront state of aborted GPU dispatches. It is enabled on Vega10 GPUs on ROCm1.9. The ROCm1.9 release will install the ROCr Debug Agent library at `/opt/rocm/lib/librocr_debug_agent64.so`

New distribution support Binary package support for Ubuntu 18.04 ROCm 1.9 is ABI compatible with KFD in upstream Linux kernels. Upstream Linux kernels support the following GPUs in these releases: 4.17: Fiji, Polaris 10, Polaris 11 4.18: Fiji, Polaris 10, Polaris 11, Vega10

Some ROCm features are not available in the upstream KFD:

More system memory available to ROCm applications Interoperability between graphics and compute RDMA IPC To try ROCm with an upstream kernel, install ROCm as normal, but do not install the `rock-dkms` package. Also add a udev rule to control `/dev/kfd` permissions:

```
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video"' | sudo tee
/etc/udev/rules.d/70-kfd.rules
```

### 3.3.31 New features as of ROCm 1.8.3

ROCm 1.8.3 is a minor update meant to fix compatibility issues on Ubuntu releases running kernel 4.15.0-33

### 3.3.32 New features as of ROCm 1.8

DKMS driver installation

Debian packages are provided for DKMS on Ubuntu

RPM packages are provided for CentOS/RHEL 7.4 and 7.5

See the ROCT-Thunk-Interface and ROCK-Kernel-Driver for additional documentation on driver setup

New distribution support

Binary package support for Ubuntu 16.04 and 18.04

Binary package support for CentOS 7.4 and 7.5

Binary package support for RHEL 7.4 and 7.5

Improved OpenMPI via UCX support

UCX support for OpenMPI

ROCm RDMA

### 3.3.33 New Features as of ROCm 1.7

DKMS driver installation

New driver installation uses Dynamic Kernel Module Support (DKMS)

Only amdkfd and amdgpu kernel modules are installed to support AMD hardware

Currently only Debian packages are provided for DKMS (no Fedora support available)

See the ROCT-Thunk-Interface and ROCK-Kernel-Driver for additional documentation on driver setup

### 3.3.34 New Features as of ROCm 1.5

Developer preview of the new OpenCL 1.2 compatible language runtime and compiler

OpenCL 2.0 compatible kernel language support with OpenCL 1.2 compatible runtime

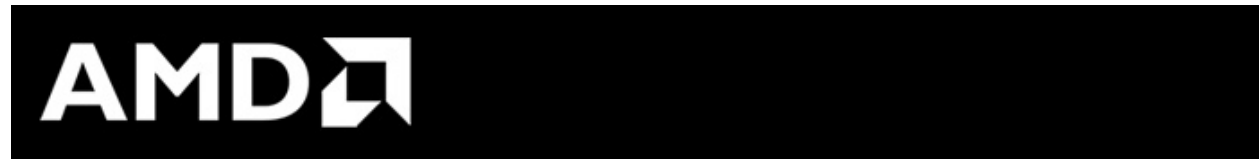
Supports offline ahead of time compilation today; during the Beta phase we will add in-process/in-memory compilation.

Binary Package support for Ubuntu 16.04

Binary Package support for Fedora 24 is not currently available

Dropping binary package support for Ubuntu 14.04, Fedora 23

IPC support



## 3.4 ROCm™ Learning Center and Knowledge Base - NEW!!

### 3.4.1 ROCm Knowledge Base

You can access the ROCm Community website and Knowledge Base at:

<https://community.amd.com/t5/knowledge-base/tkb-p/amd-rocm-tkb>

### 3.4.2 ROCm Learning Center

When it comes to solving the world's most profound computational challenges, scientists and researchers need the most powerful and accessible tools at their fingertips. With the ROCm™ open software platform built for GPU computing, HPC and ML developers can now gain access to an array of different open compute languages, compilers, libraries and tools that are both open and portable.

ROCm™ Learning Center offers resources to developers looking to tap the power of accelerated computing. No matter where they are in their journey, from those just getting started to experts in GPU programming, a broad range of technical resources below are designed to meet developers where they are at.

Happy learning!!

#### 3.4.2.1 Getting Started

<https://developer.amd.com/resources/rocm-resources/rocm-learning-center/>

#### 3.4.2.2 Fundamentals of HIP Programming

<https://developer.amd.com/resources/rocm-resources/rocm-learning-center/>

#### 3.4.2.3 From CUDA to HIP

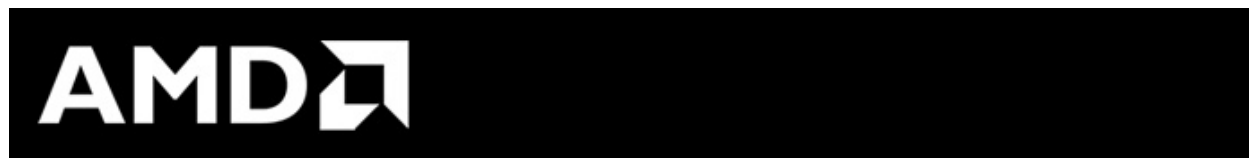
<https://developer.amd.com/resources/rocm-resources/rocm-learning-center/>

#### 3.4.2.4 Deep Learning on ROCm

<https://developer.amd.com/resources/rocm-resources/rocm-learning-center/>

#### 3.4.2.5 Multi-GPU Programming

<https://developer.amd.com/resources/rocm-resources/rocm-learning-center/>



## 3.5 DISCLAIMER

### Disclaimer

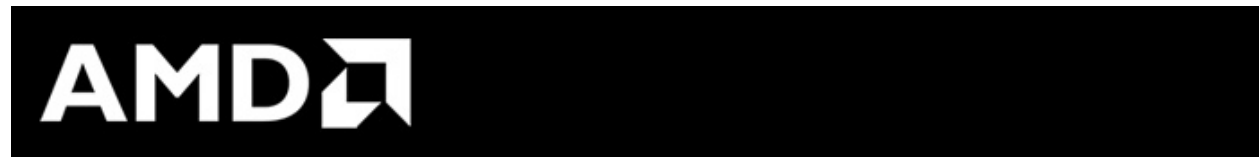
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time.



to the content hereof without obligation of AMD to notify any person of such revisions or changes. THIS INFORMATION IS PROVIDED ‘AS IS.’ AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. AMD, the AMD Arrow logo, [insert all other AMD trademarks used in the material here per AMD Trademarks] and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies. [Insert any third party trademark attribution here per AMD’s Third Party Trademark List.] © [Insert year written\*] Advanced Micro Devices, Inc. All rights reserved.

#### Third-party Disclaimer

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED “AS IS” WITHOUT A WARRANTY OF ANY KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.



## 3.6 ROCm Installation Guide v4.5

### Contents

- *ROCm Installation Guide v4.5*
  - *Overview of ROCm Installation Methods*
    - \* *About This Document*
    - \* *System Requirements*
  - *Prerequisite Actions*
    - \* *Confirm You Have a Supported Linux Distribution Version*
      - *How to Check Linux Distribution and Kernel Versions on Your System*
      - *Linux Distribution Information*
      - *Kernel Information*
      - *OS and Kernel Version Match*
    - \* *Confirm You Have a ROCm-Capable GPU*

- *How to Verify Your System Has a ROCm-Capable GPU*
- \* *Confirm the System Has the Required Tools and Packages Installed*
  - *How to Install and Configure Devtoolset-7*
  - *Required packages*
  - *Setting Permissions for Groups*
- *Meta-packages in ROCm Programming Models*
  - \* *ROCm Package Naming Conventions*
  - \* *Components of ROCm Programming Models*
  - \* *Packages in ROCm Programming Models*
- *Installation Methods*
  - \* *Installer Script Method*
    - *Downloading and Installing the Installer Script on Ubuntu*
    - *Ubuntu 18.04*
    - *Ubuntu 20.04*
    - *Downloading and Installing the Installer Script on RHEL/CentOS*
    - *RHEL/CentOS 7.9*
    - *RHEL 8.4/CentOS 8.3*
    - *Downloading and Installing the Installer Script on SLES 15*
    - *SLES 15 Service Pack 3*
    - *Using the Installer Script on Linux Distributions*
  - \* *Package Manager Method*
    - *Installing ROCm on Linux Distributions*
    - *Understanding AMDGPU and ROCm Stack Repositories on Linux Distributions*
    - *Repositories with Latest Packages*
    - *Repositories for Specific Releases*
    - *Using Package Manager on Ubuntu*
    - *Installation Of Kernel Headers and Development Packages on Ubuntu*
    - *Base URLs For AMDGPU and ROCm Stack Repositories*
    - *Adding AMDGPU Stack Repository*
    - *Install the Kernel Mode Driver and Reboot System*
    - *Add the ROCm Stack Repository*
    - *Install ROCm Meta-packages*
    - *Using Package Manager on RHEL/CentOS*
    - *Installation Of Kernel Headers and Development Packages on RHEL/CentOS*
    - *Base URLs For AMDGPU and ROCm Stack Repositories*

- *Adding the AMDGPU Stack Repository*
- *Install the Kernel Mode Driver and Reboot System*
- *Add the ROCm Stack Repository*
- *Install ROCm Meta-Packages*
- *Using Package Manager on SLES/OpenSUSE*
- *Installation of Kernel Headers and Development Packages*
- *Base URLs For AMDGPU And ROCm Stack Repositories*
- *Adding AMDGPU Stack Repository*
- *Install the Kernel Mode Driver and Reboot System*
- *Add the ROCm Stack Repository*
- *Install ROCm Meta-Packages*
- \* *Verification Process*
  - *Verifying ROCm Installation*
  - *Verifying Package Installation*
- *ROCm Stack Uninstallation*
  - \* *Uninstalling ROCm Stack*
    - *Removing ROCm Toolkit and Driver*
    - *Choosing an Uninstallation Method*
    - *Uninstallation Using Uninstall Script*
    - *Uninstallation Using Package Manager*
- *Troubleshooting*
- *Frequently Asked Questions*

### 3.6.1 Overview of ROCm Installation Methods

In addition to the installation method using the native Package Manager, AMD ROCm v4.5 introduces new methods to install ROCm. With this release, the ROCm installation uses the amdgpu-install and amdgpu-uninstall scripts.

The amdgpu-install script streamlines the installation process by:

- Abstracting the distribution-specific package installation logic
- Performing the repository set-up
- Allowing a user to specify the use case and automating the installation of all the required packages
- Performing post-install checks to verify whether the installation was completed successfully
- Installing the uninstallation script

The amdgpu-uninstall script allows the removal of the entire ROCm stack by using a single command.

Some of the ROCm-specific use cases that the installer currently supports are:

- OpenCL (ROCm/KFD based) runtime

- HIP runtimes
- ROCm libraries and applications
- ROCm Compiler and device libraries
- ROCr runtime and thunk

For more information, refer to the Installation Methods section in this guide.

### 3.6.1.1 About This Document

This document is intended for users familiar with the Linux environments and discusses the installation/uninstallation of ROCm programming models on the various flavors of Linux.

This document also refers to Radeon™ Software for Linux® as AMDGPU stack, including the kernel-mode driver `amdgpu-dkms`.

The guide provides the installation instructions for the following:

- ROCm Installation
- Heterogeneous-Computing Interface for Portability (HIP) SDK
- OPENCL™ SDK
- Kernel Mode Driver

### 3.6.1.2 System Requirements

The system requirements for the ROCm v4.5 installation are as follows:

OS
SLES15 SP3
RHEL 7.9
CentOS 7.9
RHEL 8.4
CentOS 8.3
Ubuntu 18.04.5 [5.11 HWE kernel]
Ubuntu 20.04.3 LTS [5.11 HWE kernel]

**NOTE:** Installing ROCm on Linux will require superuser privileges. For systems that have enabled sudo packages, ensure you use the sudo prefix for all required commands.

## 3.6.2 Prerequisite Actions

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

- Confirm the system has a supported Linux distribution version
- Confirm the system has a ROCm-capable GPU
- Confirm the System Has the Required Tools and Packages Installed

### 3.6.2.1 Confirm You Have a Supported Linux Distribution Version

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

**NOTE:** The ROCm installation is not supported on 32-bit operating systems.

#### 3.6.2.1.1 How to Check Linux Distribution and Kernel Versions on Your System

##### 3.6.2.1.1.1 Linux Distribution Information

Ensure you obtain the distribution information of the system by using the following command on your system from the Command Line Interface (CLI),

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

For example, running the command above on an Ubuntu system results in the following output:

```
x86_64
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"
```

##### 3.6.2.1.1.2 Kernel Information

Type the following command to check the kernel version of your Linux system.

```
$ uname -srmv
```

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

```
Linux 5.4.0-77-generic #86~18.04.5-Ubuntu SMP Fri Jun 18 01:23:22 UTC 2021 x86_64
```

##### 3.6.2.1.1.3 OS and Kernel Version Match

Confirm that the obtained Linux distribution and kernel versions match with System Requirements.

### 3.6.2.2 Confirm You Have a ROCm-Capable GPU

The ROCm platform is designed to support the following list of GPUs:

*GPU support for ROCm programming models*

Classification	GPU Name	GPU Architecture
GFX9 GPUs	AMD Instinct™ MI25	Vega 10
	AMD Radeon Instinct™ MI50	Vega 20
	AMD Radeon Instinct™ MI60	
	AMD Radeon Pro™ VII	
RDNA GPUs	AMD Radeon Pro™ W6800	AMD RDNA™
CDNA GPUs	AMD Instinct™ MI100	CDNA™

### 3.6.2.2.1 How to Verify Your System Has a ROCm-Capable GPU

To verify that your system has a ROCm-capable GPU, enter the following command from the Command Line Interface (CLI):

```
$ sudo lshw -class display
The command displays the details of detected GPUs on the system in the following
↳format:
*-display
description: VGA compatible controller
product: Vega 20
vendor: Advanced Micro Devices, Inc. [AMD/ATI]
physical id: 0
bus info: pci@0000:43:00.
version: c1
width: 64 bits
      clock: 33MHz
      capabilities: vga_controller bus_master cap_list rom
      configuration: driver=amdgpu latency=0
      resources: irq:66 memory:80000000-8fffffff memory:90000000-901fffff
↳ioport:2000(size=256) memory:9f600000-9f67ffff memory:c0000-dffff
```

**NOTE:** Verify from the output that the product field value matches the supported GPU Architecture in the table above.

### 3.6.2.3 Confirm the System Has the Required Tools and Packages Installed

You must install and configure Devtoolset-7 to use RHEL/CentOS 7.9

#### 3.6.2.3.1 How to Install and Configure Devtoolset-7

Refer to the RHEL/CentOS Installation section for more information on the steps necessary for installing and setting up Devtoolset-7.

#### 3.6.2.3.2 Required packages

Verify if the wget package for downloading files from server, is installed on your system using command below:

##### UBUNTU/DEBIAN

```
$ sudo apt list --installed | grep wget gnupg2
```

##### RHEL/CentOS

```
$ sudo yum list installed | grep wget
```

##### SLES/OPENSUSE

```
$ sudo zypper search --installed-only | grep wget
```

If the wget package not installed , execute the following command to install it:

##### UBUNTU/DEBIAN

```
$ sudo apt-get update
$ sudo apt-get install wget gnupg2
```

### RHEL/CentOS

```
$ sudo yum clean all
$ sudo yum install wget
```

### SLES/OPENSUSE

```
$ zypper install wget
```

#### 3.6.2.3.3 Setting Permissions for Groups

This section provides steps to add any current user to a video group to access GPU resources.

1. Issue the following command to check the groups in your system:

```
$ groups
```

2. Add yourself to the video group using the following instruction:

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

For all ROCm supported operating systems, continue to use the video group. By default, you can add any future users to the video and render groups.

**NOTE:** *render* group is required only for Ubuntu v20.04.

To add future users to the video and render groups, run the following command:

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

### 3.6.3 Meta-packages in ROCm Programming Models

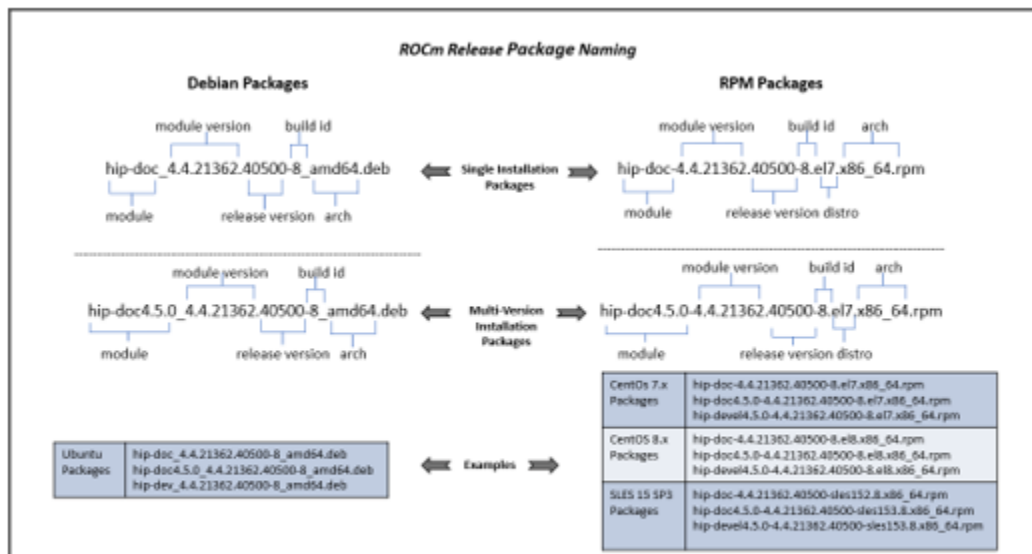
This section provides information about the required meta-packages for the following AMD ROCm™ programming models:

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

### 3.6.3.1 ROCm Package Naming Conventions

A meta-package is a grouping of related packages and dependencies used to support a specific use-case, for example, running HIP applications. All meta-packages exist in both versioned and non-versioned forms.

- Non-versioned packages – For a single installation of the latest version of ROCm
- Versioned packages – For multiple installations of ROCm



**ROCm Release Package Naming**

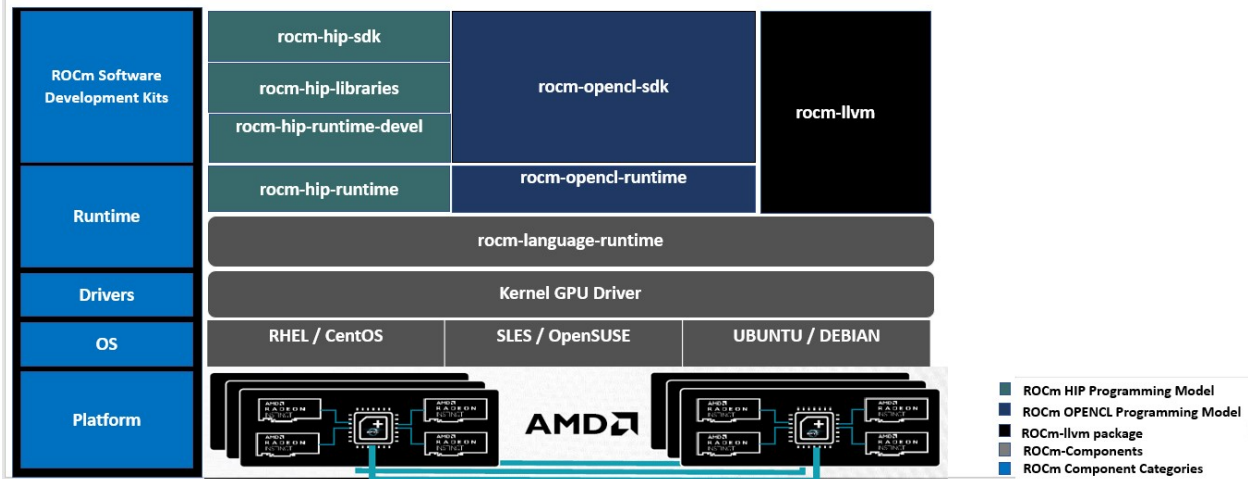
The image above demonstrates the single and multi-version ROCm packages' naming structure, including examples for various Linux distributions.

### 3.6.3.2 Components of ROCm Programming Models

The following image demonstrates the high-level layered architecture of ROCm programming models and their meta-packages. All meta-packages are a combination of required packages and libraries. For example,

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





NOTE: `rocm-llvm` is a single package that installs the required ROCm compiler files.

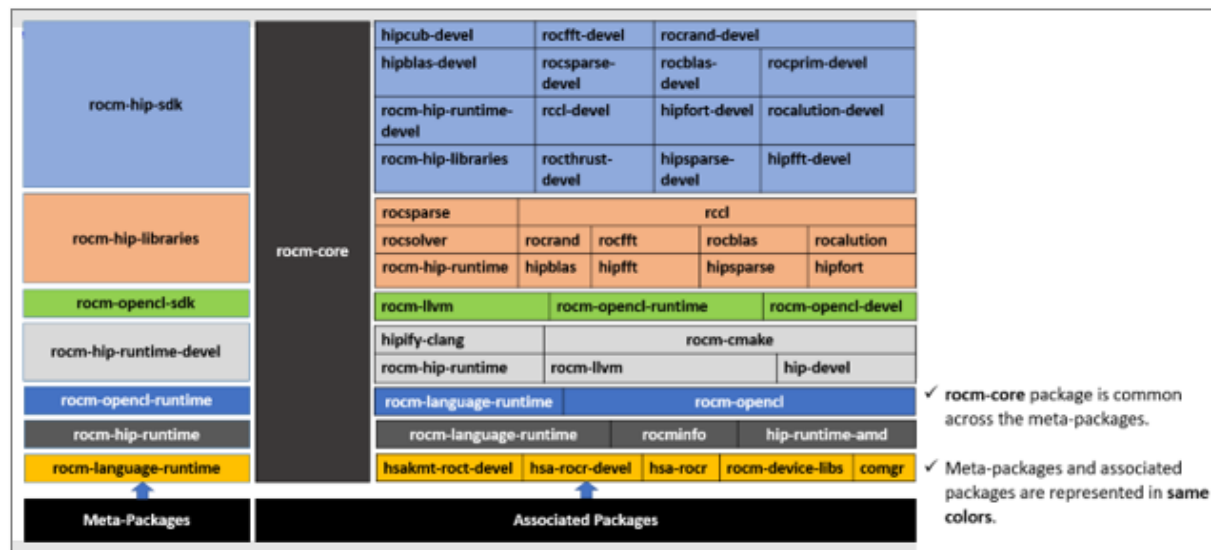
#### Meta-packages and Their Descriptions

Meta-packages	Description
<b>rocm-lang-runtime</b>	<i>rocm-language-runtime</i> meta-package is intended to install ROCr, also known as ROCm runtime.
<b>rocm-hip-runtime</b>	<i>rocm-hip-runtime</i> is intended to install packages necessary to run an application written in HIP for the AMD platform.
<b>rocm-opengl-runtime</b>	<i>rocm-opengl-runtime</i> installs packages required to run OpenCL based applications on the AMD platform.
<b>rocm-hip-runtime-devel</b>	<i>rocm-hip-runtime-devel</i> meta-package contains packages to develop an application on HIP or port it from CUDA.
<b>rocm-opengl-sdk</b>	<i>rocm-opengl-sdk</i> installs packages required to develop applications in OpenCL for the AMD platform.
<b>rocm-hip-libraries</b>	<i>rocm-hip-libraries</i> installs HIP libraries optimized for AMD platforms.
<b>rocm-hip-sdk</b>	<i>rocm-hip-sdk</i> installs packages necessary to develop/port applications using HIP and libraries for AMD platforms.

### 3.6.3.3 Packages in ROCm Programming Models

This section discusses the available meta-packages and their packages. In a ROCm programming model, packages refer to a collection of scripts, libraries, text files, a manifest, license, and other associated files that enable you to install a meta-package.

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



### Associated Packages

**NOTE:** The image above is for informational purposes only as the individual packages in a meta-package are subject to change. Users should install meta-packages, and not individual packages, to avoid conflicts.

## 3.6.4 Installation Methods

You may use the following installation methods to install ROCm:

- Installer Script Method
- Package Manager Method

### 3.6.4.1 Installer Script Method

The Installer script method automates the installation process for the AMDGPU and ROCm stack. The Installer script handles the complete installation process for ROCm, including setting up the repository, cleaning the system, updating and installing the desired drivers and meta-packages. With this approach, the system has more control over the ROCm installation process. Thus, users who are less familiar with the Linux standard commands can choose this method for ROCm installation.

For a fresh AMDGPU and ROCm installation using the Installer script method on Linux distribution, you must:

- Meet Prerequisites - Ensure the Prerequisite Actions are met before downloading and installing the installer using the Installer Script method.
- Download and Install the Installer – Ensure you download and install the installer script from the recommended URL. Note, the installer package is updated periodically to resolve known issues and add new features. The links for each Linux distribution always point to the latest available build.
- Use the Installer Script on Linux Distributions – Ensure you execute the script for installing use cases.

### 3.6.4.1.1 Downloading and Installing the Installer Script on Ubuntu

#### 3.6.4.1.1.1 Ubuntu 18.04

Download and install the installer using the following command:

```
$ sudo apt-get update

$ wget https://repo.radeon.com/amdgpu-install/21.40/ubuntu/bionic/amdgpu-install-21.40.40500-1_all.deb

$ sudo apt-get install ./amdgpu-install-21.40.40500-1_all.deb

$ sudo apt-get update
```

#### 3.6.4.1.1.2 Ubuntu 20.04

Download and install the installer.

```
$ sudo apt-get update

$ wget https://repo.radeon.com/amdgpu-install/21.40/ubuntu/focal/amdgpu-install-21.40.40500-1_all.deb

$ sudo apt-get install ./amdgpu-install-21.40.40500-1_all.deb

$ sudo apt-get update
```

### 3.6.4.1.2 Downloading and Installing the Installer Script on RHEL/CentOS

#### 3.6.4.1.2.1 RHEL/CentOS 7.9

Use the following command to download and install the installer on RHEL/CentOS 7.9.

```
$ sudo yum install https://repo.radeon.com/amdgpu-install/21.40/rhel/7.9/amdgpu-install-21.40.40500-1.noarch.rpm
```

#### 3.6.4.1.2.2 RHEL 8.4/CentOS 8.3

Use the following command to download and install the installer on RHEL 8.4/CentOS 8.3.

```
$ sudo yum install https://repo.radeon.com/amdgpu-install/21.40/rhel/8.4/amdgpu-install-21.40.40500-1.noarch.rpm
```

### 3.6.4.1.3 Downloading and Installing the Installer Script on SLES 15

#### 3.6.4.1.3.1 SLES 15 Service Pack 3

Use the following command to download and install the installer on SLES

```
$ sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/21.40/  
↪sle/15/amdgpu-install-21.40.40500-1.noarch.rpm
```

#### 3.6.4.1.4 Using the Installer Script on Linux Distributions

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

```
# To install a single use case  
$ sudo amdgpu-install --usecase=rocm
```

```
# To install multiple use-cases  
$ sudo amdgpu-install --usecase=hiplibsdk,rocm
```

```
# To display a list of available use cases. Note, the list in this section represents  
↪only a sample of available use cases for ROCm.
```

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

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

Available use cases:

rocm(for users and developers requiring full ROCm stack)

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

lrt(for users of applications requiring ROCm runtime)

- ROCm Compiler and device libraries
- ROCr runtime and thunk

opencl(for users of applications requiring OpenCL on Vega or later products)

- ROCr based OpenCL
- ROCm Language runtime

openclsdk (for application developers requiring ROCr based OpenCL)

- ROCr based OpenCL
- ROCm Language runtime
- development and SDK files for ROCr based OpenCL

hip(for users of HIP runtime on AMD products)

- HIP runtimes
- hiplibsdk (for application developers requiring HIP on AMD products)
- HIP runtimes
- ROCm math libraries
- HIP development libraries

**NOTE:** Adding `-y` as a parameter to `amdgpu-install` will skip user prompts (for automation). For example,

```
amdgpu-install -y --usecase=rocm
```

### 3.6.4.2 Package Manager Method

The Package Manager method involves a manual set up of the repository, which includes cleaning up the system, updating and installing/uninstalling meta-packages using standard commands such as yum, apt, and others respective to the Linux distribution.

**NOTE:** Users must enter the desired meta-package as the <package-name> in the command. To utilize the newly installed packages, users must install the relevant drivers and restart the system after the installation.

The typical functions of a package manager installation system include:

- Working with file archivers to extract package archives.
- Ensuring the integrity and authenticity of the package by verifying them checksums and digital certificates, respectively.
- Looking up, downloading, installing, or updating existing packages from an online repository.
- Grouping packages by function to reduce user confusion.
- Managing dependencies to ensure a package is installed with all packages it requires, thus avoiding dependency.

**NOTE:** Users may consult the documentation for their package manager for more details.

#### 3.6.4.2.1 Installing ROCm on Linux Distributions

For a fresh ROCm installation using the Package Manager method on a Linux distribution, follow the steps below:

1. Meet prerequisites - Ensure the Prerequisite Actions are met before the ROCm installation
2. Install kernel headers and development packages - Ensure kernel headers and development packages are installed on the system
3. Select the base URLs for AMDGPU and ROCm stack repository – Ensure the base URLs for AMDGPU, and ROCm stack repositories are selected
4. Add AMDGPU stack repository – Ensure AMDGPU stack repository is added
5. Install the kernel-mode driver and reboot the system – Ensure the kernel-mode driver is installed and the system is rebooted
6. Add ROCm stack repository – Ensure the ROCm stack repository is added
7. Install ROCm meta-packages – Users may install the desired meta-packages
8. Verify installation for the applicable distributions – Verify if the installation is successful.

**NOTE:** Refer to the sections below for specific commands to install each Linux distribution's ROCm and AMDGPU stack.

### 3.6.4.2.2 Understanding AMDGPU and ROCm Stack Repositories on Linux Distributions

The AMDGPU and ROCm stack repositories are divided into two categories:

- Repositories with latest release packages
- Repositories for specific releases

#### 3.6.4.2.2.1 Repositories with Latest Packages

These repositories contain the latest AMDGPU and ROCm packages available at the time. Based on the operating system's configuration, choosing this repository updates the packages automatically.

#### 3.6.4.2.2.2 Repositories for Specific Releases

The release-specific repositories consist of packages from a specific release of the AMDGPU stack and ROCm stack. The repositories are not updated for the latest packages with subsequent releases. When a new ROCm release is available, the new repository, specific to that release, is added. Users can select a specific release to install, update the previously installed single version to the later available release, or add the latest version of ROCm and currently installed by using the multi-version ROCm packages.

### 3.6.4.2.3 Using Package Manager on Ubuntu

#### 3.6.4.2.3.1 Installation Of Kernel Headers and Development Packages on Ubuntu

The following instructions to install kernel headers and development packages apply to all versions and kernels of Ubuntu.

The ROCm installation requires the linux-headers and linux-modules-extra package to be installed with the correct version corresponding to the kernel's version. For example, if the system is running the Linux kernel version 4.0-77, the identical versions of linux-headers and development packages must be installed. You may refer to the Kernel Information section to check the kernel version of the system.

For the Ubuntu/Debian environment, execute the following command to verify the kernel headers and development packages are installed with the respective versions.

```
$ sudo dpkg -l | grep linux-headers
```

The command indicates if there are Linux headers installed as shown below:

```
linux-headers-5.4.0-77-generic 5.4.0-77.86~18.04.1 amd64 Linux kernel headers_
↳for version 5.4.0 on 64 bit x86 SMP
```

Execute the following command to check whether the development packages are installed,

```
$ sudo dpkg -l | grep linux-modules-extra
```

When run, the command mentioned above lists the installed linux-modules-extra packages like the output below:

```
linux-modules-extra-5.4.0-77-generic 5.4.0-77.86~18.04.1 amd64 Linux kernel extra_
↳modules for version 5.4.0 on 64-bit x86 SMP
```

If the supported version installation of Linux headers and development packages are not installed on the system, execute the following command to install the packages:

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

### 3.6.4.2.3.2 Base URLs For AMDGPU and ROCm Stack Repositories

#### Ubuntu 18.04

Repositories with Latest Packages

- amdgpu baseurl: <https://repo.radeon.com/amdgpu/latest/ubuntu>
- rocm baseurl: <https://repo.radeon.com/rocm/apt/debian/>

Repositories for Specific Releases

- amdgpu baseurl: <https://repo.radeon.com/amdgpu/21.40/ubuntu>
- rocm base url: <https://repo.radeon.com/rocm/apt/4.5>

#### Ubuntu 20.04

Repositories with Latest Packages

- amdgpu baseurl: <https://repo.radeon.com/amdgpu/latest/ubuntu>
- rocm baseurl: <https://repo.radeon.com/rocm/apt/debian/>

Repositories for Specific Release

- amdgpu baseurl: <https://repo.radeon.com/amdgpu/21.40/ubuntu>
- rocm base url: <https://repo.radeon.com/rocm/apt/4.5>

### 3.6.4.2.3.3 Adding AMDGPU Stack Repository

#### Add GPG Key for AMDGPU and ROCm Stack

Add the gpg key for AMDGPU and ROCm repositories. For Debian-based systems like Ubuntu, configure the Debian ROCm repository as follows:

```
$ wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | sudo apt-key add -
```

**NOTE:** The gpg key may change. Ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm apt repository as mentioned above. The current rocm.gpg.key is not available in a standard key ring distribution. However, it has the following sha1sum hash:

```
777947b2579611bf4d377687b5013c69642c5762 rocm.gpg.key
```

#### Add the AMDGPU Stack Repository

You may skip this section if you have a version of the kernel-mode driver installed. If you do not have a version of the kernel-mode driver installed, follow the commands below to add the AMDGPU stack repository.

For <amdgpu baseurl> in the command below, refer to the AMDGPU base URLs as documented in Base URLs for AMDGPU and ROCm Stack Repositories

#### Ubuntu 18.04

```
$ echo 'deb [arch=amd64] <amdgpu baseurl> bionic main' | sudo tee /etc/apt/sources.
↳ list.d/amdgpu.list
```

### Ubuntu 20.04

```
$ echo 'deb [arch=amd64] <amdgpu baseurl> focal main' | sudo tee /etc/apt/sources.  
↪list.d/amdgpu.list
```

Execute the command below to update the package list

```
$ sudo apt-get update
```

#### 3.6.4.2.3.4 Install the Kernel Mode Driver and Reboot System

You may skip this section if you have the kernel-mode driver installed on your system. If you do not have the kernel-mode driver on your system, follow the instructions below. Ensure the system is rebooted after the kernel-mode driver is installed.

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

#### 3.6.4.2.3.5 Add the ROCm Stack Repository

Add the ROCm repository.

For <rocm baseurl> in the command below, refer to the ROCm base URLs as documented in Base URLs for AMDGPU and ROCm Stack Repositories

```
$ echo 'deb [arch=amd64] <rocm baseurl> ubuntu main' | sudo tee /etc/apt/sources.list.  
↪d/rocm.list  
  
$ sudo apt-get update
```

#### 3.6.4.2.3.6 Install ROCm Meta-packages

Install ROCm meta-packages. Specify the name of the meta-package you want to install as <package-name>, as shown below:

```
$ sudo apt install <package-name>
```

For example:

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



### 3.6.4.2.4 Using Package Manager on RHEL/CentOS

#### 3.6.4.2.4.1 Installation Of Kernel Headers and Development Packages on RHEL/CentOS

The ROCm installation requires the `linux-headers` and `linux-modules-extra` package to be installed with the correct version corresponding to the kernel's version. For example, if the system is running Linux kernel version 4.0-77, the identical versions of `linux-headers` and development packages must be installed.

Refer to the Kernel Information section to check the kernel version on your system.

To verify you have the supported version of the installed `linux-headers` and `linux-modules-extra` package, type the following on the command line:

```
$ sudo yum list installed | grep linux-headers
```

The command mentioned above displays the list of linux headers versions currently present on your system. Verify if the listed linux headers have the same versions as the kernel.

The following command lists the development packages on your system. Verify if the listed development package's version number matches the kernel version number.

```
$ sudo yum list installed | grep linux-modules-extra
```

If the supported version installation of linux headers and development packages does not exist on the system, execute the commands below to install:

```
$ sudo yum install kernel-headers-`uname -r` kernel-devel-`uname -r`
```

#### Preparing RHEL 7.9 for Installation

You must enable the external repositories to install on the `devtoolset-7` environment and the support files.

**NOTE:** Devtoolset is not required for CentOS 8.3/RHEL v8.4.

**NOTE:** The subscription for RHEL must be enabled and attached to a pool ID. See the Obtaining an RHEL image and license page for instructions on registering your system with the RHEL subscription server and linking to a pool id.

Enable the following repositories for RHEL v7.9:

```
$ sudo subscription-manager repos --enable rhel-server-rhsc1-7-rpms
$ sudo subscription-manager repos --enable rhel-7-server-optional-rpms
$ sudo subscription-manager repos --enable rhel-7-server-extras-rpms
$ sudo subscription-manager repos --enable=rhel-7-server-devtools-rpms
```

#### Preparing CentOS for Installation

The following steps help users prepare the CentOS system for the ROCm installation.

Extra Packages for Enterprise Linux (EPEL) provides additional packages for CENTOS that are not available in their standard repositories. Install the EPEL repository configuration package using the following command.

```
$ sudo yum install epel-release

$ sudo yum install -y centos-release-scl #Only for CentOS 7.9
```

#### Installing Devtoolset-7 for RHEL 7.9/CentOS 7.9

Use the following command to install Devtoolset-7:

```
$ sudo yum install devtoolset-7
$ source scl_source enable devtoolset-7
```

### 3.6.4.2.4.2 Base URLs For AMDGPU and ROCm Stack Repositories

#### CentOS/RHEL 7.9

Repositories with Latest Packages

- amdgpu baseurl=https://repo.radeon.com/amdgpu/latest/rhel/7.9/main/x86\_64
- rocm base url=https://repo.radeon.com/rocm/yum/rpm

Repositories for Specific Releases

- amdgpu baseurl=https://repo.radeon.com/amdgpu/21.40/rhel/7.9/main/x86\_64
- rocm baseurl=https://repo.radeon.com/rocm/yum/4.5

#### CentOS 8.3/RHEL 8.4

Repositories with Latest Packages

- amdgpu baseurl=https://repo.radeon.com/amdgpu/latest/rhel/8.4/main/x86\_64
- rocm base url=https://repo.radeon.com/rocm/centos8/rpm

Repositories for Specific Releases

- amdgpu baseurl=https://repo.radeon.com/amdgpu/21.40/rhel/8.4/main/x86\_64
- rocm baseurl=https://repo.radeon.com/rocm/centos8/4.5/

### 3.6.4.2.4.3 Adding the AMDGPU Stack Repository

You may skip this section if you have a version of the kernel-mode driver installed. If you do not have a version of the kernel-mode driver installed, follow the commands below to add the AMDGPU stack repository.

#### Add the AMDGPU Stack Repository

Create a `/etc/yum.repos.d/amdgpu.repo` file with the following contents with amdgpu base URL.

For `<amdgpu baseurl>` in the command below, refer to the AMDGPU base URLs as documented in Base URLs for AMDGPU and ROCm Stack Repositories

```
[amdgpu]
name=amdgpu
baseurl=<amdgpu baseurl>
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

**NOTE:** The gpg key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm to the yum repository as mentioned above. The current rocm.gpg.key is not available in a standard key ring distribution but has the following sha1sum hash:

```
777947b2579611bf4d377687b5013c69642c5762 rocm.gpg.key
```

Execute the command below to clean the cached files from enabled repositories:

```
$ sudo yum clean all
```

#### 3.6.4.2.4.4 Install the Kernel Mode Driver and Reboot System

You may skip this section if the kernel-mode driver is already installed on your system. If you do not have a version of the kernel-mode driver installed, follow the commands below to install the kernel-mode driver:

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

Reboot the system after the completion of driver installation.

```
$ sudo reboot
```

#### 3.6.4.2.4.5 Add the ROCm Stack Repository

Create a `/etc/yum.repos.d/rocm.repo` file with the following content.

For `<rocm baseurl>` in the command below, refer to the ROCm base URLs documented in Base URLs for AMDGPU and ROCm Stack Repositories.

```
[rocm]
name=rocm
baseurl=<rocm baseurl>
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

**NOTE:** The gpg key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm yum repository as mentioned above. The current `rocm.gpg.key` is not available in a standard key ring distribution, but has the following sha1sum hash:

```
777947b2579611bf4d377687b5013c69642c5762 rocm.gpg.key
```

Execute the command below to clean the cached files from enabled repositories:

```
$ sudo yum clean all
```

#### 3.6.4.2.4.6 Install ROCm Meta-Packages

Use the following command to install the ROCm packages.

```
$ sudo yum install <package-name>
```

Specify the meta-package name as `<package-name>`, which you want to install, in the command given above.

For example,

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

### 3.6.4.2.5 Using Package Manager on SLES/OpenSUSE

This section introduces the ROCm installation process on SLES/OpenSUSE.

#### 3.6.4.2.5.1 Installation of Kernel Headers and Development Packages

ROCm installation requires linux-headers and linux-modules-extra package to be installed with the correct version corresponding to the kernel's version. For example, if the system is running the Linux kernel version 4.0-77, the same versions of linux-headers and development packages must be installed.

Refer to the Kernel Information section to check the kernel version on your system.

Ensure that the correct version of the latest kernel-default-devel and kernel-default packages are installed. The following command lists the installed kernel-default-devel and kernel-default package.

```
$ sudo zypper info kernel-default-devel or kernel-default
```

**NOTE:** This next step is only required if you find from the above command that the “kernel-default-devel” and “kernel-default” versions of the package, corresponding to the kernel release version, do not exist on your system.

If the required version of packages does not exist on the system, install with the command below:

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

#### 3.6.4.2.5.2 Base URLs For AMDGPU And ROCm Stack Repositories

Repositories with Latest Packages

- amdgpu baseurl=[https://repo.radeon.com/amdgpu/latest/sle/15/main/x86\\_64](https://repo.radeon.com/amdgpu/latest/sle/15/main/x86_64)
- rocm baseurl=<https://repo.radeon.com/rocm/zypper>

Repositories for Specific Releases

- amdgpu baseurl=[https://repo.radeon.com/amdgpu/21.40/sle/15/main/x86\\_64](https://repo.radeon.com/amdgpu/21.40/sle/15/main/x86_64)
- rocm baseurl=<https://repo.radeon.com/rocm/zyp/4.5/>

#### 3.6.4.2.5.3 Adding AMDGPU Stack Repository

You may skip this section if you have a version of the kernel-mode driver installed. If you do not have a version of the kernel-mode driver installed, follow the commands below to add the AMDGPU stack repository.

##### Add the AMDGPU Stack Repository

Create a `/etc/zypp/repos.d/amdgpu.repo` file with the following content.

For `<amdgpu baseurl>` in the command below, refer to the AMDGPU base URLs as documented in Base URLs for AMDGPU and ROCm Stack Repositories.

```
[amdgpu]
name=amdgpu
baseurl=<amdgpu_baseurl>
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

**NOTE:** The gpg key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm zypp repository as mentioned above. The current rocm.gpg.key is not available in a standard key ring distribution but has the following sha1sum hash:

```
777947b2579611bf4d377687b5013c69642c5762 rocm.gpg.key
```

Use the following commands to update the added repository, and add the Perl repository:

```
$ sudo zypper ref
$ sudo zypper clean --all
$ sudo zypper addrepo https://download.opensuse.org/repositories/devel:languages:perl/
→SLE_15/devel:languages:perl.repo
$ sudo SUSEConnect -p sle-module-desktop-applications/15.3/x86_64
$ sudo SUSEConnect --product sle-module-development-tools/15.3/x86_64
$ sudo SUSEConnect--product PackageHub/15.3/x86_64
$ sudo zypper ref
```

#### 3.6.4.2.5.4 Install the Kernel Mode Driver and Reboot System

Install the kernel-mode driver. If you already have a version of the kernel-mode driver installed, you may skip this section. If you do not have a version of the kernel-mode driver installed, follow the commands below to install and reboot the system.

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

#### 3.6.4.2.5.5 Add the ROCm Stack Repository

Add the ROCm repository by executing the following commands,

Create a `/etc/zypp/repos.d/rocm.repo` file with the following content.

For `<rocm baseurl>` in the command below, refer to the ROCm base URLs documented in Base URLs for AMDGPU and ROCm Stack Repositories.

```
[rocm]
name=rocm
baseurl=<rocm_baseurl>
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

**NOTE:** The gpg key may change. Ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm zypp repository as mentioned above. The current rocm.gpg.key is not available in a standard key ring distribution but has the following sha1sum hash:

```
777947b2579611bf4d377687b5013c69642c5762 rocm.gpg.key
```

Use the following command to update the added repository.

```
$ sudo zypper ref
```

### 3.6.4.2.5.6 Install ROCm Meta-Packages

Install the ROCm package by typing the command below:

```
$ sudo zypper --gpg-auto-import-keys install <package-name>
```

Specify the name of the meta-package name as <package-name>, which you want to install, in the command given above. For example,

- \$ sudo zypper --gpg-auto-import-keys install rocm-hip-sdk
- \$ sudo zypper --gpg-auto-import-keys install rocm-hip-sdk rocm-ocl-sdk

### 3.6.4.3 Verification Process

#### 3.6.4.3.1 Verifying ROCm Installation

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

```
/opt/rocm-<version>/bin/rocmfinfo
```

OR

```
/opt/rocm-<version>/opencl/bin/clinfo
```

**NOTE:** For convenience, users may add the ROCm binaries in your PATH, as shown in the example below.

```
$ echo 'export PATH=$PATH:/opt/rocm-<version>/bin:/opt/rocm-<version>/opencl/bin'
```

#### 3.6.4.3.2 Verifying Package Installation

Users can use the following commands to ensure the packages are installed successfully.

Linux Distro	Command
Ubuntu/Debian	\$ sudo apt list --installed
RHEL/CentOS	\$ sudo yum list installed
OpenSUSE / SLES	\$ sudo zypper search --installed-only

## 3.6.5 ROCm Stack Uninstallation

Uninstallation of ROCm entails removing ROCm packages, tools, and libraries from the system.

### 3.6.5.1 Uninstalling ROCm Stack

#### 3.6.5.1.1 Removing ROCm Toolkit and Driver

This section describes the uninstallation process in detail. The following methods remove the ROCm stack from the system.

#### 3.6.5.1.2 Choosing an Uninstallation Method

You can uninstall using the following methods:

- Uninstallation using the Uninstall Script
- Package Manager uninstallation

##### 3.6.5.1.2.1 Uninstallation Using Uninstall Script

The following commands uninstall all installed ROCm packages:

```
$ sudo amdgpu-uninstall
```

**NOTE:** amdgpu-uninstall ignores all parameters/arguments and uninstalls all ROCm packages.

Refer to the Uninstall Kernel Mode Driver section to uninstall the kernel-mode driver.

##### 3.6.5.1.2.2 Uninstallation Using Package Manager

The Package Manager uninstallation offers a method for a clean uninstallation process for ROCm. This section describes how to uninstall the ROCm for various Linux distributions.

Use the following commands to remove the specific meta-packages from the system.

#### Uninstalling Specific Meta-packages

Use the following command to uninstall specific meta-packages. You may specify the name of the meta-package name as <package-name> you want to uninstall in the command given below.

#### UBUNTU/DEBIAN

```
$ sudo apt autoremove <package-name>
```

#### RHEL/CentOS

```
$ sudo yum remove <package-name>
```

#### SLES/OPENSUSE

```
$ sudo zypper remove <package-name>
```

#### Complete Uninstallation of ROCm Packages

If you want to uninstall all installed ROCm packages, use the following command as uninstallation of rocm-core package removes all the ROCm specific packages from the system.

#### UBUNTU/DEBIAN

```
$ sudo apt autoremove rocm-core
```

### RHEL/CentOS

```
$ sudo yum remove rocm-core
```

### SLES/OPENSUSE

```
$ sudo zypper remove rocm-core
```

**NOTE:** The command above removes all ROCm-specific packages.

Refer to the Uninstall Kernel Mode Driver section below to uninstall the kernel-mode driver uninstallation.

### Uninstall Kernel Mode Driver

Users can uninstall the kernel-mode driver by entering the following command on the system.

### UBUNTU/DEBIAN

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

### RHEL/CentOS

```
$ sudo yum remove amdgpu-dkms
```

### SLES/OPENSUSE

```
$ sudo zypper remove amdgpu-dkms
```

### Remove ROCm and AMDGPU Repositories

#### UBUNTU/DEBIAN

Use the following commands to remove the AMDGPU and ROCm repository from the Ubuntu/Debian system:

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

Clear cache and clean the system.

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

Reboot the system.

```
$ sudo reboot
```

### RHEL/CentOS

This section describes the process of removing AMDGPU and ROCm repositories from the RHEL/CentOS environment.

Remove the reference to the AMDGPU and ROCm repository from the system using the following instructions

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

Clear cache and clean the system.



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

Restart the system.

```
$ sudo reboot
```

## SLES/OPENSUSE

This section describes the process of removing AMDGPU and ROCm repositories from the SLES/OPENSUSE environment.

Remove the reference to the amdgpu and ROCm repository from the system with the commands below.

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

Clear cache and clean the system.

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

Restart the system.

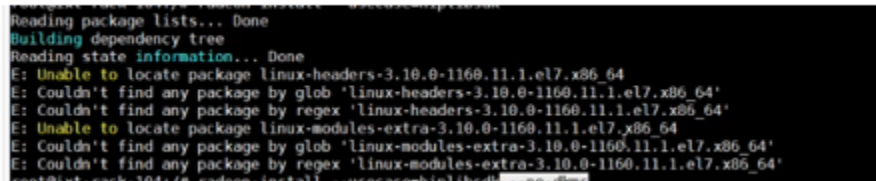
```
$ sudo reboot
```

## 3.6.6 Troubleshooting

### Issue

If the amdgpu-install script is executed inside Docker, the system may display the following error while installing various use cases.

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



```
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package linux-headers-3.10.0-1160.11.1.el7.x86_64
E: Could not find any package by glob 'linux-headers-3.10.0-1160.11.1.el7.x86_64'
E: Could not find any package by regex 'linux-headers-3.10.0-1160.11.1.el7.x86_64'
E: Unable to locate package linux-modules-extra-3.10.0-1160.11.1.el7.x86_64
E: Could not find any package by glob 'linux-modules-extra-3.10.0-1160.11.1.el7.x86_64'
E: Could not find any package by regex 'linux-modules-extra-3.10.0-1160.11.1.el7.x86_64'
root@ist-rock-164: /# rpmdev-install --usecase=hiplibdl --no-dkms
```

### Resolution

When the installation is initiated in Docker, the installer tries to install the use case along with the kernel-mode driver. However, the kernel-mode driver cannot be installed in a Docker system. To skip the installation of the kernel-mode driver, proceed with the option `--no-dkms`, as shown in the command below.

```
$ sudo amdgpu-install --usecase=rocm --no-dkms
```

### 3.6.7 Frequently Asked Questions

*Can users install multiple packages at the same time with the installer script?*

Yes, users can install multiple packages at the same time with the installer script. Provide package names in the `--usecase` parameter, separated by a comma, as shown below.

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

*How to list all the possible inputs for the `--usecase` parameter in the `amdgpu-install` script?*

The following command lists all the possible options for `--usecase`

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

*What are the available options other than the `--usecase` in the `amdgpu-install` script?*

The following command lists all possible options users can provide in the `amdgpu-install` script.

```
$ sudo amdgpu-install --help
```

*How to check if the kernel module is installed successfully?*

Type the following command on the system.

```
$ sudo dkms status
```

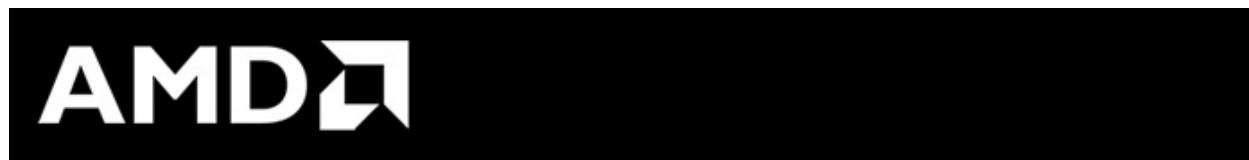
The command displays the output in the following format if the installation of the kernel module is successful.

```
amdgpu, 4.3-52.el7, 3.10.0-1160.11.1.el7.x86_64, x86_64: installed (original_module_↵  
↵exists)
```

*Does the Docker container support command - `$ sudo SUSEConnect --product PackageHub/15.2/x86_64`?*

Users do not need to execute the following command in Docker container.

```
$ sudo SUSEConnect --product PackageHub/15.2/x86_64
```



## 3.7 HIP Installation v4.5

HIP can be easily installed using the pre-built binary packages with the package manager for your platform.

### 3.7.1 HIP Prerequisites

HIP code can be developed either on AMD ROCm platform using HIP-Clang compiler, or a CUDA platform with NVCC installed.

### 3.7.2 AMD Platform

```
sudo apt install mesa-common-dev
sudo apt install clang
sudo apt install comgr
sudo apt-get -y install rocm-dkms
```

HIP-Clang is the compiler for compiling HIP programs on AMD platform.

HIP-Clang can be built manually:

```
git clone -b roc-4.5.x https://github.com/RadeonOpenCompute/llvm-project.git
cd llvm-project
mkdir -p build && cd build
cmake -DCMAKE_INSTALL_PREFIX=/opt/rocm/llvm -DCMAKE_BUILD_TYPE=Release -DLLVM_ENABLE_
↳ASSERTIONS=1 -
DLLVM_TARGETS_TO_BUILD="AMDGPU;X86" - DLLVM_ENABLE_PROJECTS="clang;lld;compiler-rt" ..
↳/llvm
make -j
sudo make install
```

The ROCm device library can be manually built as following,

```
export PATH=/opt/rocm/llvm/bin:$PATH
git clone -b roc-4.5.x https://github.com/RadeonOpenCompute/ROCm-Device-Libs.git
cd ROCm-Device-Libs
mkdir -p build && cd build
CC=clang CXX=clang++ cmake -DLLVM_DIR=/opt/rocm/llvm -DCMAKE_BUILD_TYPE=Release -
↳DLLVM_ENABLE_WERROR=1 -DLLVM_ENABLE_ASSERTIONS=1 -DCMAKE_INSTALL_PREFIX=/opt/rocm ..
make -j
sudo make install
```

### 3.7.3 NVIDIA Platform

HIP-nvcc is the compiler for HIP program compilation on NVIDIA platform.

- Add the ROCm package server to your system as per the OS-specific guide available [here](#).
- Install the “hip-runtime-nvidia” and “hip-devel” package. This will install CUDA SDK and the HIP porting layer.

```
apt-get install hip-runtime-nvidia hip-devel
```

- Default paths and environment variables:
  - By default HIP looks for CUDA SDK in /usr/local/cuda (can be overridden by setting CUDA\_PATH env variable).
  - By default HIP is installed into /opt/rocm/hip (can be overridden by setting HIP\_PATH environment variable).
  - Optionally, consider adding /opt/rocm/bin to your path to make it easier to use the tools.

## 3.7.4 Building HIP from Source

### 3.7.4.1 Get HIP source code

```
git clone -b rocm-4.5.x https://github.com/ROCm-Developer-Tools/hipamd.git
git clone -b rocm-4.5.x https://github.com/ROCm-Developer-Tools/hip.git
git clone -b rocm-4.5.x https://github.com/ROCm-Developer-Tools/ROCclr.git
git clone -b rocm-4.5.x https://github.com/RadeonOpenCompute/ROCm-OpenCL-Runtime.git
```

### 3.7.4.2 Set the environment variables

```
export HIPAMD_DIR="$(readlink -f hipamd) "
export HIP_DIR="$(readlink -f hip) "
export ROCclr_DIR="$(readlink -f ROCclr) "
export OPENCL_DIR="$(readlink -f ROCm-OpenCL-Runtime) "
```

ROCclr is defined on AMD platform that HIP use Radeon Open Compute Common Language Runtime (ROCclr), which is a virtual device interface that HIP runtimes interact with different backends.

See <https://github.com/ROCm-Developer-Tools/ROCclr>

HIPAMD repository provides implementation specifically for AMD platform. See <https://github.com/ROCm-Developer-Tools/hipamd>

### 3.7.4.3 Build HIP

```
cd "$HIPAMD_DIR"
mkdir -p build; cd build
cmake -DHIP_COMMON_DIR=$HIP_DIR -DAMD_OPENCL_PATH=$OPENCL_DIR -DROCCLR_PATH=$ROCCLR_
DIR -DCMAKE_PREFIX_PATH="/opt/rocm/" -DCMAKE_INSTALL_PREFIX=$PWD/install ..
make -j$(nproc)
sudo make install
```

**Note:** If you don't specify CMAKE\_INSTALL\_PREFIX, hip runtime will be installed to "/opt/rocm/hip". By default, release version of AMDHIP is built.

### 3.7.4.4 Default paths and environment variables

- By default HIP looks for HSA in /opt/rocm/hsa (can be overridden by setting HSA\_PATH environment variable).
- By default HIP is installed into /opt/rocm/hip (can be overridden by setting HIP\_PATH environment variable).
- By default HIP looks for clang in /opt/rocm/llvm/bin (can be overridden by setting HIP\_CLANG\_PATH environment variable)
- By default HIP looks for device library in /opt/rocm/lib (can be overridden by setting DEVICE\_LIB\_PATH environment variable)
- Optionally, consider adding /opt/rocm/bin to your PATH to make it easier to use the tools
- Optionally, set HIPCC\_VERBOSE=7 to output the command line for compilation

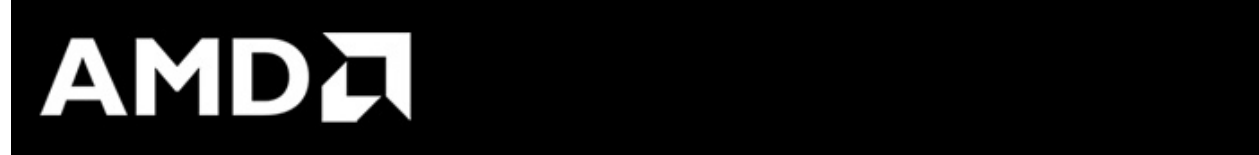
After installation, make sure HIP\_PATH is pointed to /where/to/install/hip

### 3.7.4.5 Verify your installation

Run hipconfig (instructions below assume default installation path):

```
/opt/rocm/bin/hipconfig --full
```

Compile and run the square sample. You can access the square sample at,  
[https://github.com/ROCm-Developer-Tools/HIP/tree/main/samples/0\\_Intro/square](https://github.com/ROCm-Developer-Tools/HIP/tree/main/samples/0_Intro/square)



## 3.8 ROCm Installation v4.3

- *Deploying ROCm*
- *Prerequisites*
- *Supported Operating Systems*
  - *Ubuntu*
  - *CentOS RHEL*
  - *SLES 15 Service Pack 2*
- *ROCm Installation Known Issues and Workarounds*
- *Getting the ROCm Source Code*

### 3.8.1 Deploying ROCm

AMD hosts both Debian and RPM repositories for the ROCm v4.x packages.

The following directions show how to install ROCm on supported Debian-based systems such as Ubuntu 18.04.x

**Note:** These directions may not work as written on unsupported Debian-based distributions. For example, newer versions of Ubuntu may not be compatible with the rock-dkms kernel driver. In this case, you can exclude the rocm-dkms and rock-dkms packages.

**Note:** You must use either ROCm or the amdgpu-pro driver. Using both drivers will result in an installation error.

**Important - Mellanox ConnectX NIC Users:** If you are using Mellanox ConnectX NIC, you must install Mellanox OFED before installing ROCm.

For more information about installing Mellanox OFED, refer to:

<https://docs.mellanox.com/display/MLNXOFEDv461000/Installing+Mellanox+OFED>

### 3.8.1.1 ROCm Repositories

Use the following ROCm repositories for the required major and point releases:

- Major releases - <https://repo.radeon.com/rocm/yum/rpm/>
- Point releases - <https://repo.radeon.com/rocm/yum/4.3/>

### 3.8.1.2 Base Operating System Kernel Upgrade

For SUSE, it is strongly recommended to follow the steps below when upgrading the base operating system kernel:

1. Remove *rock-dkms* before the upgrade.
2. Install the new kernel.
3. Reboot the system.
4. Reinstall *rock-dkms*.

Implementing these steps ensures correct loading of *amdgpu* and *amdkfd* after the kernel upgrade and prevents any issue caused by an incomplete DKMS upgrade. Fedora and Ubuntu do not have this restriction.

## 3.8.2 Prerequisites

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

OS	Kernel
<b>SLES15 SP2</b>	5.3.18-24.49
<b>RHEL 7.9</b>	3.10.0-1160.6.1.el7
<b>CentOS 7.9</b>	3.10.0-1127
<b>RHEL 8.3</b>	4.18.0-193.1.1.el8
<b>CentOS 8.3</b>	4.18.0-193.el8
<b>Ubuntu 18.04.5</b>	5.4.0-71-generic
<b>Ubuntu 20.04.2</b>	5.8.0-48-generic

**Note:** Ubuntu versions lower than 18 are no longer supported.

**Note:** AMD ROCm only supports Long Term Support (LTS) versions of Ubuntu. Versions other than LTS may work with ROCm, however, they are not officially supported.

### 3.8.2.1 Perl Modules for HIP-Base Package

The hip-base package has a dependency on Perl modules that some operating systems may not have in their default package repositories. Use the following commands to add repositories that have the required Perl packages:

- For SLES 15 SP2

```
sudo zypper addrepo
```

For more information, see

[https://download.opensuse.org/repositories/devel:languages:perl/SLE\\_15/devel:languages:perl.repo](https://download.opensuse.org/repositories/devel:languages:perl/SLE_15/devel:languages:perl.repo)

- For CentOS8.3

```
sudo yum config-manager --set-enabled powertools
```

- For RHEL8.3

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
```

### 3.8.2.2 Complete Reinstallation OF AMD ROCm V4.3 Recommended

Complete uninstallation of previous ROCm versions is required before installing a new version of ROCm. An upgrade from previous releases to AMD ROCm v4.3 is not supported.

**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.

- 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 unchanged `- /opt/rocm/opencl/bin/x86_64/clinfo`.

**Note:** After an operating system upgrade, AMD ROCm may upgrade automatically and result in an error. This is because AMD ROCm does not support upgrades currently. You must uninstall and reinstall AMD ROCm after an operating system upgrade.

### 3.8.2.3 Multi-version Installation Updates

With the AMD ROCm v4.3 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 should be created for each multi-version rocm `<= 4.3.0`
  - command: `echo <version> | sudo tee /opt/rocm-<version>/info/version`
  - example: `echo 4.3.0 | sudo tee /opt/rocm-4.3.0/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.

**Note:** Before updating to the latest version of the operating system, delete the ROCm packages to avoid DKMS-related issues.

### 3.8.3 Setting Permissions for Groups

This section provides steps to add any current user to a video group to access GPU resources.

1. Issue the following command to check the groups in your system:

```
groups
```

2. Add yourself to the video group using the following instruction:

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

For all ROCm supported operating systems, continue to use *video group*. By default, you can add any future users to the video and render groups.

**Note:** *render group* is required only for Ubuntu v20.04.

3. To add future users to the video and render groups, run the following command:

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

### 3.8.4 Supported Operating Systems

#### 3.8.4.1 Ubuntu

**Note:** AMD ROCm only supports Long Term Support (LTS) versions of Ubuntu. Versions other than LTS may work with ROCm, however, they are not officially supported.

##### 3.8.4.1.1 Installing a ROCm Package from a Debian Repository

To install from a Debian Repository:

1. Run the following code to ensure that your system is up to date:

```
sudo apt update
sudo apt dist-upgrade
sudo apt install libnuma-dev
sudo reboot
```

2. Add the ROCm apt repository.



For Debian-based systems like Ubuntu, configure the Debian ROCm repository as follows:

Key: <https://repo.radeon.com/rocm/rocm.gpg.key>

```
sudo apt install wget gnupg2

wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | sudo apt-key add -

echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/<ROCm_version#>/ ubuntu main' | sudo tee /etc/apt/sources.list.d/rocm.list
```

For example

For the current version of ROCm, ensure you replace `<ROCm_version#>` with `debian`.

```
echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/debian/ ubuntu main' | sudo tee /etc/apt/sources.list.d/rocm.list
```

For older versions of ROCm, replace `<ROCm_version#>` with any ROCm versions number like 4.3.1, 4.3 or 4.2.

For example,

```
echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/4.3/ ubuntu main' | sudo tee /etc/apt/sources.list.d/rocm.list
```

**Note:** For ROCm v4.1 and lower, use `'xenial main'`, instead of `'ubuntu main'`, as shown below.

```
wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | sudo apt-key add -

echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/<ROCm_version#>/ xenial main' | sudo tee /etc/apt/sources.list.d/rocm.list
```

For example,

```
echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/4.1/ ubuntu main' | sudo tee /etc/apt/sources.list.d/rocm.list
```

**Note:** For developer systems or Docker containers (where it could be beneficial to use a fixed ROCm version), select a versioned repository from:

<https://repo.radeon.com/rocm/apt/>

The gpg key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm apt repository.

```
wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | sudo apt-key add -
```

The current `rocm.gpg.key` is not available in a standard key ring distribution, but has the following sha1sum hash:

```
777947b2579611bf4d377687b5013c69642c5762 rocm.gpg.key
```

3. Install the ROCm meta-package. Update the appropriate repository list and install the `rocm-dkms` meta-package:

```
sudo apt update

sudo apt install rocm-dkms && sudo reboot
```

4. Restart the system.

5. After restarting the system, run the following commands to verify that the ROCm installation is successful. If you see your GPUs listed by both commands, the installation is considered successful.

```
/opt/rocm/bin/rocminfo
/opt/rocm/rocm/bin/rocmclinfo
```

Note: To run the ROCm programs, add the ROCm binaries in your PATH.

```
echo 'export PATH=$PATH:/opt/rocm/bin:/opt/rocm/rocm/bin:/opt/rocm/rocmcl/bin'
↪ | sudo tee -a /etc/profile.d/rocm.sh
```

### 3.8.4.1.2 Uninstalling ROCm Packages from Ubuntu

To uninstall the ROCm packages from Ubuntu 20.04 or Ubuntu 18.04.5, run the following command:

```
sudo apt autoremove rocm-opencl rocm-dkms rocm-dev rocm-utils && sudo reboot
```

### 3.8.4.1.3 Using Debian-based ROCm with Upstream Kernel Drivers

You can install ROCm user-level software without installing AMD's custom ROCk kernel driver. The kernel used must have the *HSA kernel driver* option enabled and compiled into the *amdgpu* kernel driver. To install only ROCm user-level software, run the following commands instead of installing rocm-dkms:

```
sudo apt update
sudo apt install rocm-dev
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video" | sudo tee /etc/
↪udev/rules.d/70-kfd.rules
```

## 3.8.4.2 CentOS RHEL

This section describes how to install ROCm on supported RPM-based systems such as CentOS/RHEL.

### 3.8.4.2.1 Preparing RHEL for Installation

RHEL is a subscription-based operating system. You must enable the external repositories to install on the devtoolset-7 environment and the dkms support files.

Note: The following steps do not apply to the CentOS installation.

1. The subscription for RHEL must be enabled and attached to a pool ID. See the Obtaining an RHEL image and license page for instructions on registering your system with the RHEL subscription server and attaching to a pool id.
2. Enable the following repositories for RHEL v7.x:

```
sudo subscription-manager repos --enable rhel-server-rhsc1-7-rpms
sudo subscription-manager repos --enable rhel-7-server-optional-rpms
sudo subscription-manager repos --enable rhel-7-server-extras-rpms
```

3. Enable additional repositories by downloading and installing the epel-release-latest-7/epel-release-latest-8 repository RPM:

```
sudo rpm -ivh <repo>
```

For more details,

- see <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm> for RHEL v7.x
- see <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm> for RHEL v8.x

4. Install and set up Devtoolset-7.

**Note:** Devtoolset is not required for CentOS/RHEL v8.x

To setup the Devtoolset-7 environment, follow the instructions on this page: <https://www.softwarecollections.org/en/scls/rhscl/devtoolset-7/>

Note: devtoolset-7 is a software collections package and is not supported by AMD.

5. Add the ROCm GPG key

```
sudo rpm --import https://repo.radeon.com/rocm/rocm.gpg.key
```

### 3.8.4.2.1.1 Installing CentOS for DKMS

Use the dkms tool to install the kernel drivers on CentOS/RHEL:

```
sudo yum install -y epel-release
sudo yum install -y dkms kernel-headers-`uname -r` kernel-devel-`uname -r`
```

### 3.8.4.2.2 Installing ROCm

To install ROCm on your system, follow the instructions below:

1. Delete the previous versions of ROCm before installing the latest version.
2. Create a /etc/yum.repos.d/rocm.repo file with the following contents:
  - CentOS/RHEL 7.x : <https://repo.radeon.com/rocm/yum/rpm>
  - CentOS/RHEL 8.x : <https://repo.radeon.com/rocm/centos8/rpm>

```
[ROCM]
name=ROCM
baseurl=https://repo.radeon.com/rocm/yum/rpm
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

**Note:** The URL of the repository must point to the location of the repositories' repodata database. For developer systems or Docker containers (where it could be beneficial to use a fixed ROCm version), select a versioned repository from:

<https://repo.radeon.com/rocm/yum/>

3. Install ROCm components using the following command:

```
sudo yum install rocm-dkms && sudo reboot
```

4. Restart the system. The rock-dkms component is installed and the /dev/kfd device is now available.
5. Restart the system.
6. Test the ROCm installation.

### 3.8.4.2.3 Testing the ROCm Installation

After restarting the system, run the following commands to verify that the ROCm installation is successful. If you see your GPUs listed, you are good to go!

```
/opt/rocm/bin/rocminfo
/opt/rocm/rocm-opencl/bin/clinfo
```

**Note:** Add the ROCm binaries in your PATH for easy implementation of the ROCm programs.

```
echo 'export PATH=$PATH:/opt/rocm/bin:/opt/rocm/profiler/bin:/opt/rocm/rocm-opencl/bin' |
↪ sudo tee -a /etc/profile.d/rocm.sh
```

### 3.8.4.2.4 Compiling Applications Using HCC, HIP, and Other ROCm Software

To compile applications or samples, run the following command to use gcc-7.2 provided by the devtoolset-7 environment:

```
scl enable devtoolset-7 bash
```

### 3.8.4.2.5 Uninstalling ROCm from CentOS/RHEL

To uninstall the ROCm packages, run the following command:

```
sudo yum autoremove rocm-opencl rocm-dkms rock-dkms
```

### 3.8.4.2.6 Using ROCm on CentOS/RHEL with Upstream Kernel Drivers

You can install ROCm user-level software without installing AMD's custom ROCk kernel driver. The kernel used must have the *HSA kernel driver* option enabled and compiled into the *amdgpu* kernel driver. To install only ROCm user-level software, run the following commands instead of installing rocm-dkms:

```
sudo yum install rocm-dev
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video" | sudo tee /etc/
↪ udev/rules.d/70-kfd.rules
sudo reboot
```

**Note:** Ensure you restart the system after ROCm installation.

### 3.8.4.2.7 Installing Development Packages for Cross Compilation

You can develop and test ROCm packages on different systems. For example, some development or build systems may not have an AMD GPU installed. In this scenario, you can avoid installing the ROCm kernel driver on your development system. Instead, install the following development subset of packages:

```
sudo yum install rocm-dev
```

**Note:** To execute ROCm-enabled applications, you will require a system installed with the full ROCm driver stack.

### 3.8.4.3 SLES 15 Service Pack 2

The following section tells you how to perform an install and uninstall ROCm on SLES 15 SP 2.

**Note:** For SUSE-based distributions (SLE, OpenSUSE, etc), upgrading the base kernel after installing ROCm may result in a broken installation. This is due to policies regarding unsupported kernel modules. To mitigate this, make the following change before initializing the amdgpu module:

```
#Allow Unsupported Driver and Load Driver
cat <<EOF | tee /etc/modprobe.d/10-unsupported-modules.conf
allow_unsupported_modules 1
EOF
```

For more information, refer to <https://www.suse.com/support/kb/doc/?id=000016939>

#### Installation

1. Install the “dkms” package.

```
sudo SUSEConnect --product PackageHub/15.2/x86_64
sudo zypper install dkms
```

2. Add the ROCm repo.

```
sudo zypper clean -all
sudo zypper addrepo https://download.opensuse.org/repositories/devel:languages:perl/
↪SLE_15/devel:languages:perl.repo
sudo zypper ref
sudo rpm --import https://repo.radeon.com/rocm/rocm.gpg.key
sudo zypper --gpg-auto-import-keys install rocm-dkms
sudo reboot
```

**Note:** For developer systems or Docker containers (where it could be beneficial to use a fixed ROCm version), select a versioned repository from:

<https://repo.radeon.com/rocm/zypp/>

3. Run the following command once

```
cat <<EOF | sudo tee /etc/modprobe.d/10-unsupported-modules.conf
allow_unsupported_modules 1
EOF
sudo modprobe amdgpu
```

4. Verify the ROCm installation.
5. Run `/opt/rocm/bin/rocminfo` and `/opt/rocm/ocl/bin/clinfo` commands to list the GPUs and verify that the ROCm installation is successful.
6. Restart the system.
7. Test the basic ROCm installation.
8. After restarting the system, run the following commands to verify that the ROCm installation is successful. If you see your GPUs listed by both commands, the installation is considered successful.

```
/opt/rocm/bin/rocminfo
/opt/rocm/ocl/bin/clinfo
```

**Note:** To run the ROCm programs more efficiently, add the ROCm binaries in your PATH.

```
echo 'export PATH=$PATH:/opt/rocm/bin:/opt/rocm/profiler/bin:/opt/rocm/rocm/bin' | sudo tee -a /etc/profile.d/rocm.sh
```

### Using ROCm on SLES with Upstream Kernel Drivers

```
sudo zypper install rocm-dev
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video" | sudo tee /etc/udev/rules.d/70-kfd.rules
sudo reboot
```

### Uninstallation

To uninstall, use the following command:

```
sudo zypper remove rocm-ocl rocm-dkms rock-dkms
```

Note: Ensure all other installed packages/components are removed. Note: Ensure all the content in the /opt/rocm directory is completely removed. If the command does not remove all the ROCm components/packages, ensure you remove them individually.

#### 3.8.4.3.1 Performing an OpenCL-only Installation of ROCm

Some users may want to install a subset of the full ROCm installation. If you are trying to install on a system with a limited amount of storage space, or which will only run a small collection of known applications, you may want to install only the packages that are required to run OpenCL applications. To do that, you can run the following installation command instead of the command to install rocm-dkms.

```
sudo yum install rock-dkms rocm-ocl-devel && sudo reboot
```

## 3.8.5 ROCm Installation Known Issues and Workarounds

The ROCm platform relies on some closed source components to provide functionalities like HSA image support. These components are only available through the ROCm repositories, and they may be deprecated or become open source components in the future. These components are made available in the following packages:

- hsa-ext-rocr-dev

## 3.8.6 Getting the ROCm Source Code

AMD ROCm is built from open source software. It is, therefore, possible to modify the various components of ROCm by downloading the source code and rebuilding the components. The source code for ROCm components can be cloned from each of the GitHub repositories using git. For easy access to download the correct versions of each of these tools, the ROCm repository contains a repo manifest file called default.xml. You can use this manifest file to download the source code for ROCm software.

The repo tool from Google® allows you to manage multiple git repositories simultaneously. Run the following commands to install the repo:

```
mkdir -p ~/bin/
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
```

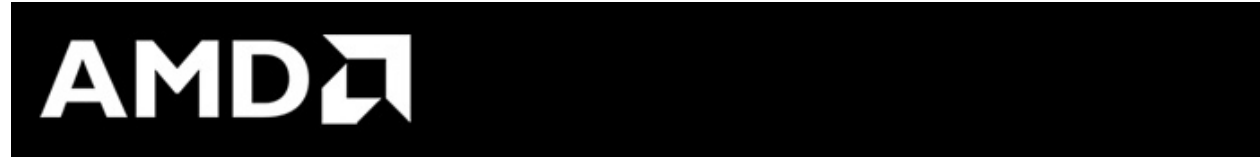
Note: You can choose a different folder to install the repo into if you desire. ~/bin/ is used as an example.

### 3.8.7 Downloading the ROCm Source Code

The following example shows how to use the repo binary to download the ROCm source code. If you choose a directory other than `~/bin/` to install the repo, you must use that chosen directory in the code as shown below:

```
mkdir -p ~/ROCM/  
cd ~/ROCM/  
~/bin/repo init -u https://github.com/RadeonOpenCompute/ROCM.git -b roc-4.3.x  
repo sync
```

**Note:** Using this sample code will cause the repo to download the open source code associated with this ROCm release. Ensure that you have ssh-keys configured on your machine for your GitHub ID prior to the download.



## 3.9 Multi Version Installation

Users can install and access multiple versions of the ROCm toolkit simultaneously.

Previously, users could install only a single version of the ROCm toolkit.

Now, users have the option to install multiple versions simultaneously and toggle to the desired version of the ROCm toolkit. From the v3.3 release, multiple versions of ROCm packages can be installed in the `/opt/rocm-<version>` folder.

### 3.9.1 Prerequisites

Ensure the existing installations of ROCm, including `/opt/rocm`, are completely removed before the ROCm toolkit installation. The ROCm package requires a clean installation.

- To install a single instance of ROCm, use the `rocm-dkms` or `rocm-dev` packages to install all the required components. This creates a symbolic link `/opt/rocm` pointing to the corresponding version of ROCm installed on the system.
- To install individual ROCm components, create the `/opt/rocm` symbolic link pointing to the version of ROCm installed on the system. For example, `# ln -s /opt/rocm-4.0.0 /opt/rocm`
- To install multiple instance ROCm packages, create `/opt/rocm` symbolic link pointing to the version of ROCm installed/used on the system. For example, `# ln -s /opt/rocm-4.0.0 /opt/rocm`

**Note:** The Kernel Fusion Driver (KFD) must be compatible with all versions of the ROCm software installed on the system.

### 3.9.2 Before You Begin

Review the following important notes:

#### Single Version Installation

To install a single instance of the ROCm package, access the non-versioned packages.

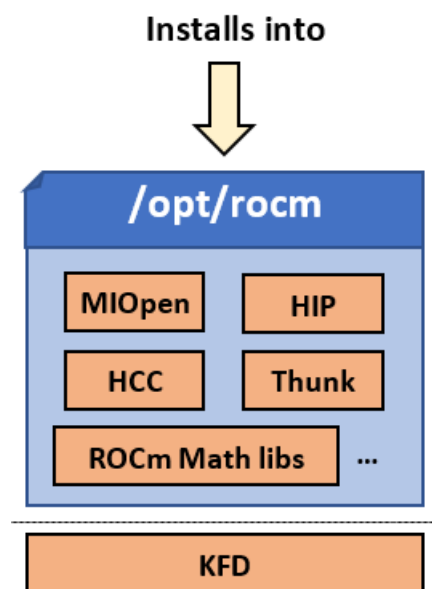
**Note:** You must not install any components from the multi-instance set.

For example,

- rocm-dkms
- rocm-dev
- hip

A fresh installation of single-version installation will install the new version in the `/opt/rocm-<version>` folder.

## ROCm 3.0



#### Multi Version Installation

- To install a multi-instance of the ROCm package, access the versioned packages and components.

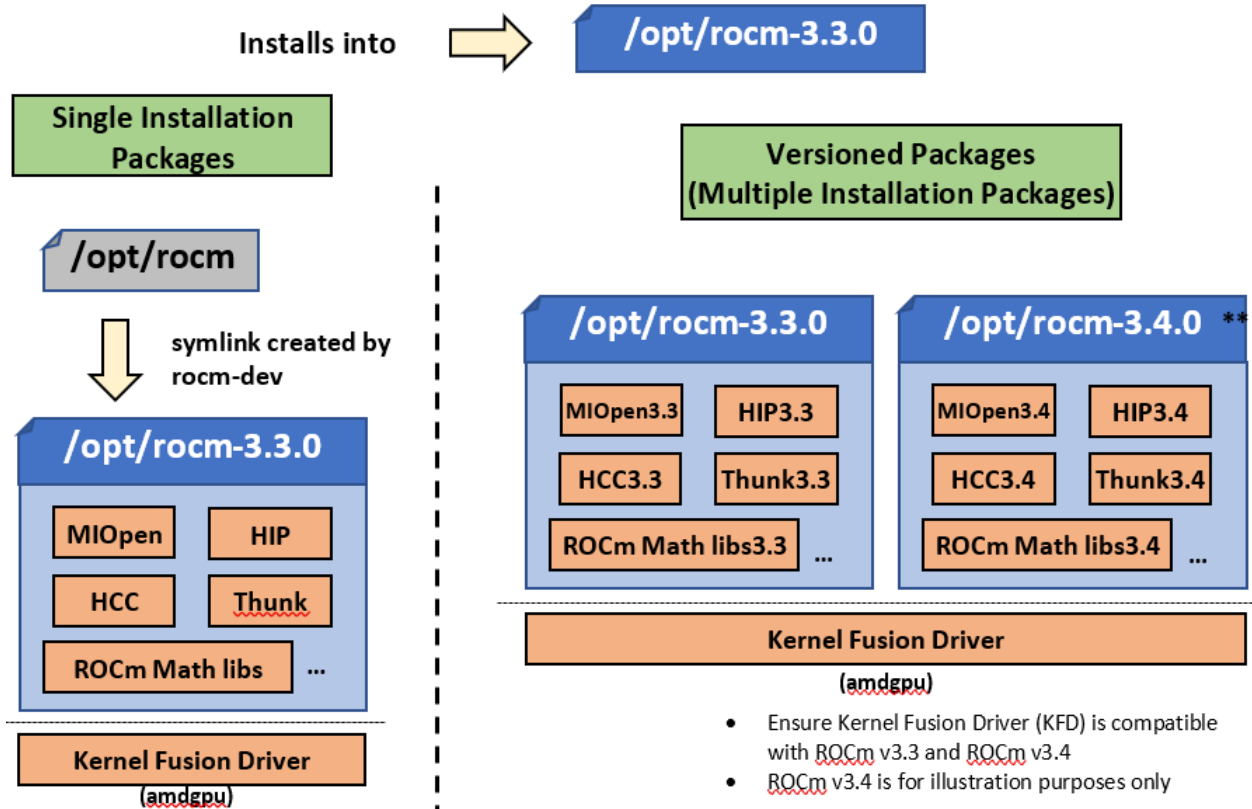
For example,

- rocm-dev4.2.0
- hip4.2.0
- kernel/firmware package doesn't have multi version so it should be installed using "apt/yum/zypper install rock-dkms".
- The new multi-instance package enables you to install two versions of the ROCm toolkit simultaneously and provides the ability to toggle between the two versioned packages.
- The ROCm-DEV package does not create symlinks



- Users must create symlinks if required
- Multi-version installation with previous ROCm versions is not supported
- Kernel Fusion Driver (KFD) must be compatible with all versions of ROCm installations

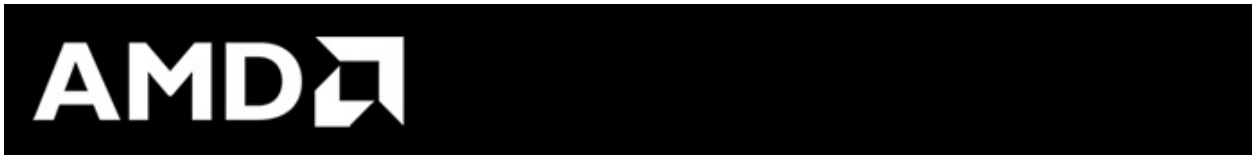
## ROCm v3.3



**IMPORTANT:** A single instance ROCm package cannot co-exist with the multi-instance package.

**NOTE:** The multi-instance installation applies only to ROCm v3.3 and above. This package requires a fresh installation after the complete removal of existing ROCm packages. The multi-version installation is not backward compatible.

**Note:** If you install the multi-instance version of AMD ROCm and create a sym-link to `/opt/rocm`, you must run 'ldconfig' to ensure the software stack functions correctly with the sym-link.



## 3.10 Using CMake with AMD ROCm

Most components in AMD ROCm support CMake 3.5 or higher out-of-the-box and do not require any special Find modules. A Find module is often used by downstream to find the files by guessing locations of files with platform-specific hints. Typically, the Find module is required when the upstream is not built with CMake or the package configuration files are not available.

AMD ROCm provides the respective *config-file* packages, and this enables `find_package` to be used directly. AMD ROCm does not require any Find module as the *config-file* packages are shipped with the upstream projects.

### 3.10.1 Finding Dependencies

When dependencies are not found in standard locations such as `/usr` or `/usr/local`, then the `CMAKE_PREFIX_PATH` variable can be set to the installation prefixes. This can be set to multiple locations with a semicolon separating the entries.

There are two ways to set this variable:

- Pass the flag when configuring with `-DCMAKE_PREFIX_PATH=...`. This approach is preferred when users install the components in custom locations.
- Append the variable in the `CMakeLists.txt` file. This is useful if the dependencies are found in a common location. For example, when the binaries provided on [repo.radeon.com](https://repo.radeon.com) are installed to `/opt/rocm`, you can add the following line to a `CMakeLists.txt` file

```
list (APPEND CMAKE_PREFIX_PATH /opt/rocm/hip /opt/rocm)
```

### 3.10.2 Using HIP in CMake

There are two ways to use HIP in CMake:

- Use the HIP API without compiling the GPU device code. As there is no GPU code, any C or C++ compiler can be used. The `find_package(hip)` provides the `hip::host` target to use HIP in this context

```
# Search for rocm in common locations
list(APPEND CMAKE_PREFIX_PATH /opt/rocm/hip /opt/rocm)
# Find hip
find_package(hip)
# Create the library
add_library(myLib ...)
# Link with HIP
target_link_libraries(myLib hip::host)
```

---

**Note:** The `hip::host` target provides all the usage requirements needed to use HIP without compiling GPU device code.

---

- Use HIP API and compile GPU device code. This requires using a device compiler. The compiler for CMake can be set using either the `CMAKE_C_COMPILER` and `CMAKE_CXX_COMPILER` variable or using the `CC` and `CXX` environment variables. This can be set when configuring CMake or put into a CMake toolchain file. The device compiler must be set to a compiler that supports AMD GPU targets, which is usually Clang.

The `find_package(hip)` provides the `hip::device` target to add all the flags for device compilation

```
# Search for rocm in common locations
list(APPEND CMAKE_PREFIX_PATH /opt/rocm/hip /opt/rocm)
# Find hip
find_package(hip)
# Create library
add_library(myLib ...)
# Link with HIP
target_link_libraries(myLib hip::device)
```

This project can then be configured with:

```
cmake -DCMAKE_C_COMPILER=/opt/rocm/llvm/bin/clang -DCMAKE_CXX_COMPILER=/opt/rocm/llvm/
↪bin/clang++ ..
```

Which uses the device compiler provided from the binary packages from [repo.radeon.com](https://repo.radeon.com).

---

**Note:** Compiling for the GPU device requires at least C++11. This can be enabled by setting CMAKE\_CXX\_STANDARD or setting the correct compiler flags in the CMake toolchain.

---

The GPU device code can be built for different GPU architectures by setting the GPU\_TARGETS variable. By default, this is set to all the currently supported architectures for AMD ROCm. It can be set by passing the flag during configuration with -DGPU\_TARGETS=gfx900. It can also be set in the CMakeLists.txt as a cached variable before calling find\_package(hip):

```
# Set the GPU to compile for
set(GPU_TARGETS "gfx900" CACHE STRING "GPU targets to compile for")
# Search for rocm in common locations
list(APPEND CMAKE_PREFIX_PATH /opt/rocm/hip /opt/rocm)
# Find hip
find_package(hip)
```

### 3.10.3 Using AMD ROCm Libraries

Libraries such as rocBLAS, MIOpen, and others support CMake users as well.

As illustrated in the example below, to use MIOpen from CMake, you can call find\_package(miopen), which provides the MIOpen CMake target. This can be linked with target\_link\_libraries:

```
# Search for rocm in common locations
list(APPEND CMAKE_PREFIX_PATH /opt/rocm/hip /opt/rocm)
# Find miopen
find_package(miopen)
# Create library
add_library(myLib ...)
# Link with miopen
target_link_libraries(myLib MIOpen)
```

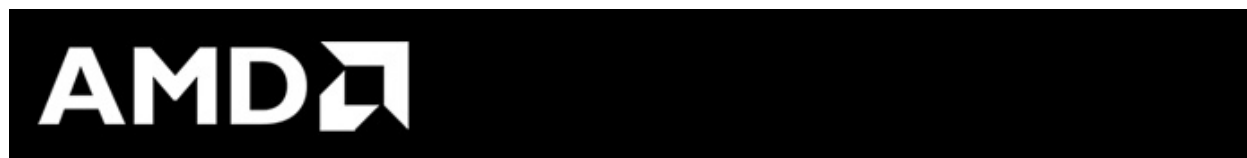
---

**Note:** Most libraries are designed as host-only API, so using a GPU device compiler is not necessary for downstream projects unless it uses the GPU device code.

---

### 3.10.4 ROCm CMake Packages

Component	Package	Targets
HIP	hip	hip::host, hip::device
rocPRIM	rocprim	roc::rocprim
roc-Thrust	roc-thrust	roc::rocthrust
hipCUB	hipcub	hip::hipcub
rocRAND	rocrand	roc::rocrand
rocBLAS	rocblas	roc::rocblas
roc-SOLVER	rocsolver	roc::rocsolver
hip-BLAS	hipblas	roc::hipblas
rocFFT	rocfft	roc::rocfft
hipFFT	hipfft	hip::hipfft
roc-SPARSE	rocsparse	roc::rocsparse
hipSPARSE	hipspars	roc::hipspars
rocA-LU-TION	rocalution	roc::rocalution
RCCL	rccl	rccl
MIOpen	miopen	MIOpen
MI-GraphX	mi-graphx	migraphx::migraphx, migraphx::migraphx_c, migraphx::migraphx_cpu, migraphx::migraphx_gpu, migraphx::migraphx_onnx, migraphx::migraphx_tf



## 3.11 Mesa Multimedia Installation

### 3.11.1 Prerequisites

- Ensure you have ROCm installed on the system.

For ROCm installation instructions, see

[https://rocm.docs.amd.com/en/latest/Installation\\_Guide/Installation-Guide.html](https://rocm.docs.amd.com/en/latest/Installation_Guide/Installation-Guide.html)

### 3.11.1.1 System Prerequisites

The following operating systems are supported for Mesa Multimedia:

- Ubuntu 18.04.3
- Ubuntu 20.04, including dual kernel

### 3.11.1.2 Installation Prerequisites

1. Obtain the AMDGPU driver from <https://www.amd.com/en/support/kb/release-notes/rn-amdgpu-unified-linux-20-45> for the appropriate distro version.
2. Follow the pre-installation instructions at <https://amdgpu-install.readthedocs.io/en/latest/> (from “Preamble” to “Using the amdgpu-install Script” sections).
3. Proceed with the installation instructions as documented in the next section.

## 3.11.2 Installation Instructions

1. Use the following installation instructions to install Mesa Multimedia:

```
| ./amdgpu-install -y --no-dkms
```

### Note:

Run it from the directory where the download is unpacked. The download and install instructions are:

```
$ cd ~/Downloads
$ tar -Jxvf amdgpu-pro-YY.XX-NNNNNN.tar.xz
$ cd ~/Downloads/amdgpu-pro-YY.XX-NNNNNN
$ ./amdgpu-install -y --no-dkms
```

### 2. gstreamer Installation

```
sudo apt-get -y install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-
↳ plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-plugins-bad gstreamer1.0-
↳ vaapi gstreamer1.0-libav gstreamer1.0-tools

sudo apt-get -y install gst-omx-listcomponents gstreamer1.0-omx-bellagio-config
↳ gstreamer1.0-omx-generic gstreamer1.0-omx-generic-config
```

### 3. Utilities Installation

```
sudo apt-get -y install mediainfo ffmpeg

sudo reboot

# Check amdgpu loading status after reboot

dmesg | grep -i initialized
```

(continues on next page)

(continued from previous page)

```
Sep 24 13:00:42 jz-tester kernel: [ 277.120055] [drm] VCN decode and encode_
↳ initialized successfully.

Sep 24 13:00:42 jz-tester kernel: [ 277.121654] [drm] Initialized amdgpu 3.34.0_
↳ 20150101 for 0000:03:00.0 on minor 1
```

#### 4. Configure Running Environment Variables

```
export BELLAGIO_SEARCH_PATH=/opt/amdgpu/lib/x86_64-linux-gnu/libomxil-bellagio0:/opt/
↳ amdgpu/lib/libomxil-bellagio0

export GST_PLUGIN_PATH=/opt/amdgpu/lib/x86_64-linux-gnu/gstreamer-1.0/

export GST_VAAPI_ALL_DRIVERS=1

export OMX_RENDER_NODE=/dev/dri/renderD128
```

### 3.11.3 Check Installation

#### 1. Ensure you perform an installation check.

```
omxregister-bellagio -v

Scanning directory /opt/amdgpu/lib/libomxil-bellagio0/

Scanning library /opt/amdgpu/lib/libomxil-bellagio0/libomx_mesa.so

Component OMX.mesa.video_decoder registered with 0 quality levels

Specific role OMX.mesa.video_decoder.mpeg2 registered

Specific role OMX.mesa.video_decoder.avc registered

Specific role OMX.mesa.video_decoder.hevc registered

Component OMX.mesa.video_encoder registered with 0 quality levels

Specific role OMX.mesa.video_encoder.avc registered

2 OpenMAX IL ST static components in 1 library successfully scanned
```

```
gst-inspect-1.0 omx
```

#### Plugin Details

Name	OMX
Description	GStreamer OpenMAX Plug-ins
Filename	/usr/lib/x86_64-linux-gnu/ gstreamer-1.0/libgstomx.so
Version	1.12.4
License	LGPL
Source module	gst-omx
Source release date	2017-12-07
Binary package	GStreamer OpenMAX Plug-ins source release
Origin URL	Unknown package origin

```
omxmpeg2dec: OpenMAX MPEG2 Video Decoder
```

```
omxh264dec: OpenMAX H.264 Video Decoder
```

```
omxh264enc: OpenMAX H.264 Video Encoder
```

### 3. Features

```
+-- 3 elements
```

```
gst-inspect-1.0 vaapi
```

### Plugin Details

Name	vaapi
Description	VA-API based elements
Filename	/usr/lib/x86_64-linux-gnu/ gstreamer-1.0/libgstvaapi.so
Version	1.14.5
License	LGPL
Source module	gstreamer-vaapi
Source release date	2019-05-29
Binary package	gstreamer-vaapi
Origin URL	<a href="http://bugzilla.gnome.org/enter_bug.cgi?product=GStreamer">http://bugzilla.gnome.org/enter_bug.cgi?product=GStreamer</a>

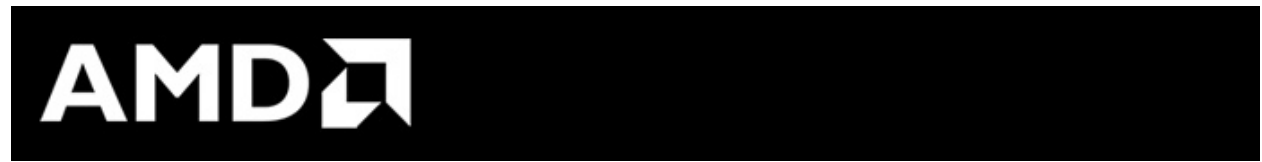
```
vaapijpegdec: VA-API JPEG decoder
vaapimpeg2dec: VA-API MPEG2 decoder
vaapih264dec: VA-API H264 decoder
vaapivc1dec: VA-API VC1 decoder
vaapivp9dec: VA-API VP9 decoder
vaapih265dec: VA-API H265 decoder
vaapiopostproc: VA-API video postprocessing
vaapidcodebin: VA-API Decode Bin
vaapisink: VA-API sink
vaapih265enc: VA-API H265 encoder
vaapih264enc: VA-API H264 encoder
```

### 11 Features

(continues on next page)

(continued from previous page)

```
+++ 11 elements
```



## 3.12 Tools Installation

### 3.12.1 ROCTracer

ROC-tracer library: Runtimes Generic Callback/Activity APIs.

The goal of the implementation is to provide a generic installation independent from the specific runtime profiler to trace API and asynchronous activity.

The following API provides the functionality to register runtimes API callbacks and asynchronous activity records pool support.

#### 3.12.1.1 ROC-TX library: code annotation events API

**Includes basic API:** roctxMark, roctxRangePush, roctxRangePop

#### 3.12.1.2 Usage

##### 3.12.1.2.1 rocTracer API

To use the rocTracer API, you need the API header to link your application with roctracer .so library:

- API header: `/opt/rocm/roctracer/include/roctracer.h`
- .so library: `/opt/rocm/lib/libroctracer64.so`

##### 3.12.1.2.2 rocTX API

To use the rocTX API, you need the API header to link your application with roctx .so library:

- API header: `/opt/rocm/roctracer/include/roctx.h`
- .so library: `/opt/rocm/lib/libroctx64.so`



### 3.12.1.2.3 Library source tree

- doc - documentation
- inc/roctracer.h - rocTracer library public API header
- inc/roctx.h - rocTX library public API header
- **src - Library sources**
  - core - rocTracer library API sources
  - roctx - rocTX library API sources
  - util - library utils sources
- **test - test suit**
  - MatrixTranspose - test based on HIP MatrixTranspose sample

### 3.12.1.2.4 API Description

‘roctracer’ / ‘rocTX’ profiling C API specification

### 3.12.1.2.5 Code examples

- test/MatrixTranspose\_test/MatrixTranspose.cpp
- test/MatrixTranspose/MatrixTranspose.cpp

### 3.12.1.2.6 Build and run test

Prerequisites

- ROCm
- Python modules: CppHeaderParser, argparse

1. Install *CppHeaderParser*, *argparse*

```
sudo pip install CppHeaderParser argparse
```

2. Clone development branch of ROCTracer

```
git clone -b amd-master https://github.com/ROCm-Developer-Tools/roctracer
```

3. Set environment

```
export CMAKE_PREFIX_PATH=/opt/rocm
```

4. Use custom HIP version

```
export HIP_PATH=/opt/rocm/hip
```

5. Build roctracer library

```
export CMAKE_BUILD_TYPE=<debug|release> # release by default
cd <your path>/roctracer && BUILD_DIR=build HIP_VDI=1 ./build.sh
```

## 6. Build and run test

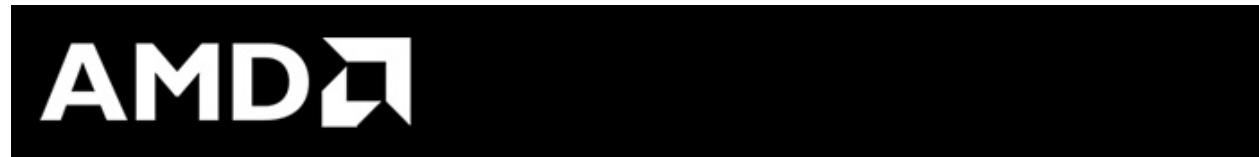
```
make mytest  
run.sh
```

## 7. Install

```
make install
```

or

```
make package && dpkg -i *.deb
```



## 3.13 Software Stack for AMD GPU

### 3.13.1 Machine Learning and High Performance Computing Software Stack for AMD GPU v4.1

#### 3.13.1.1 ROCm Binary Package Structure

ROCm is a collection of software ranging from drivers and runtimes to libraries and developer tools. In AMD's package distributions, these software projects are provided as a separate packages. This allows users to install only the packages they need, if they do not wish to install all of ROCm. These packages will install most of the ROCm software into `/opt/rocm/` by default.

The packages for each of the major ROCm components are:

##### 3.13.1.1.1 ROCm Core Components

- ROCK Kernel Driver: `rock-dkms` `rock-dkms-firmware`
- ROCr Runtime: `hsa-rocr-dev`
- ROCt Thunk Interface: `hsakmt-roct`, `hsakmt-roct-dev`

#### 3.13.1.1.2 ROCm Support Software

- ROCm SMI: `rocm-smi`
- ROCm cmake: `rocm-cmake`
- rocminfo: `rocminfo`
- ROCm Bandwidth Test: `rocm_bandwidth_test`

#### 3.13.1.1.3 ROCm Compilers

- Clang compiler: `llvm-amdgpu`
- HIP: `hip_base`, `hip_doc`, `hip_rocclr`, `hip_samples`
- ROCm Clang-OCCL Kernel Compiler: `rocm-clang-ocl`

#### 3.13.1.1.4 ROCm Device Libraries

- ROCm Device Libraries: `rocm-device-libs`
- ROCm OpenCL: `rocm-openccl`, `rocm-openccl-devel` (on RHEL/CentOS), `rocm-openccl-dev` (on Ubuntu)

#### 3.13.1.1.5 ROCm Development ToolChain

- Asynchronous Task and Memory Interface (ATMI): `atmi`
- ROCm Debug Agent: `rocm_debug_agent`
- ROCm Code Object Manager: `comgr`
- ROC Profiler: `rocprofiler-dev`
- ROC Tracer: `roctracer-dev`

#### 3.13.1.1.6 ROCm Libraries

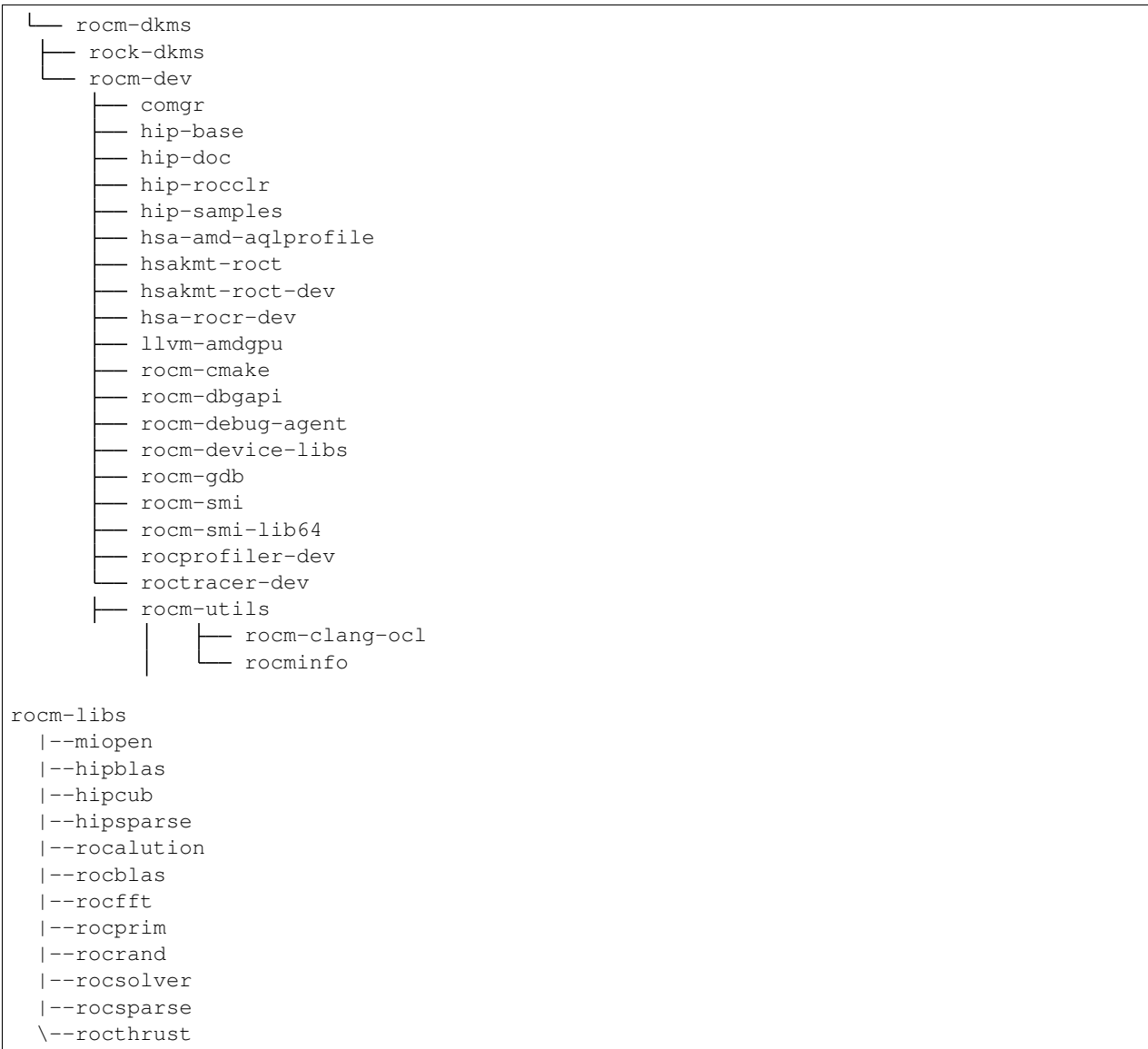
- rocALUTION: `rocalution`
- rocBLAS: `rocblas`
- hipBLAS: `hipblas`
- hipCUB: `hipCUB`
- rocFFT: `rocfft`
- rocRAND: `rocrand`
- rocSPARSE: `rocsparse`
- hipSPARSE: `hipsparse`
- ROCm SMI Lib: `rocm-smi-lib64`
- rocThrust: `rocThrust`
- MIOpen: `MIOpen-HIP` (for the HIP version), `MIOpen-OpenCL` (for the OpenCL version)

- MIOpenGEMM: `miopengemm`
- MIVisionX: `mivisionx`
- RCCL: `rccl`

To make it easier to install ROCm, the AMD binary repositories provide a number of meta-packages that will automatically install multiple other packages. For example, `rocm-dkms` is the primary meta-package that is used to install most of the base technology needed for ROCm to operate. It will install the `rock-dkms` kernel driver, and another meta-package (`rocm-dev`) which installs most of the user-land ROCm core components, support software, and development tools.

The `rocm-utils` meta-package will install useful utilities that, while not required for ROCm to operate, may still be beneficial to have. Finally, the `rocm-libs` meta-package will install some (but not all) of the libraries that are part of ROCm.

The chain of software installed by these meta-packages is illustrated below:



These meta-packages are not required but may be useful to make it easier to install ROCm on most systems.

Note: Some users may want to skip certain packages. For instance, a user that wants to use the upstream kernel drivers (rather than those supplied by AMD) may want to skip the rocm-dkms and rock-dkms packages. Instead, they could directly install rocm-dev.

Similarly, a user that only wants to install OpenCL support instead of HCC and HIP may want to skip the rocm-dkms and rocm-dev packages. Instead, they could directly install rock-dkms, rocm-ocl, and rocm-ocl-dev and their dependencies.

#### 3.13.1.1.6.1 ROCm Platform Packages

The following platform packages are for ROCm v4.1.0:

#### 3.13.1.1.7 Drivers, ToolChains, Libraries, and Source Code

The latest supported version of the drivers, tools, libraries and source code for the ROCm platform have been released and are available from the following GitHub repositories:

##### ROCm Core Components

- [ROCK Kernel Driver](#)
- [ROCr Runtime](#)
- [ROCr Thunk Interface](#)

##### ROCm Support Software

- [ROCm SMI](#)
- [ROCm cmake](#)
- [rocm-info](#)
- [ROCm Bandwidth Test](#)

##### ROCm Compilers

- [HIP](#)
- [ROCm Clang-OpenCL Kernel Compiler](#)

Example Applications:

- [HIP Examples](#)

##### ROCm Device Libraries and Tools

- [ROCm Device Libraries](#)
- [ROCm OpenCL Runtime](#)
- [ROCm LLVM OCL](#)
- [ROCm Device Libraries OCL](#)
- [Asynchronous Task and Memory Interface](#)
- [ROCr Debug Agent](#)
- [ROCm Code Object Manager](#)
- [ROC Profiler](#)
- [ROC Tracer](#)

- AOMP
- Radeon Compute Profiler
- ROCm Validation Suite

### ROCm Libraries

- rocBLAS
- hipBLAS
- rocFFT
- rocRAND
- rocSPARSE
- hipSPARSE
- rocALUTION
- MIOpenGEMM
- mi open
- rocThrust
- ROCm SMI Lib
- RCCL
- MIVisionX
- hipCUB
- AMDMIGraphX

#### 3.13.1.1.7.1 List of ROCm Packages for Supported Operating Systems

#### 3.13.1.1.7.2 ROCm-Library Meta Packages

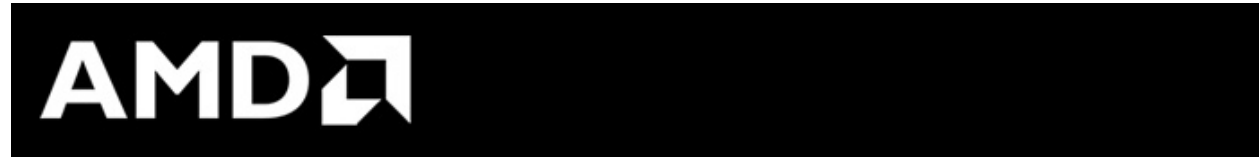
Package	Debian	RPM
rocFFT	Yes	Yes
rocRAND	Yes	Yes
rocBLAS	Yes	Yes
rocSPARSE	Yes	Yes
rocALUTION	Yes	Yes
rocPRIM	Yes	Yes
rocTHRUST	Yes	Yes
rocSOLVER	Yes	Yes
hipBLAS	Yes	Yes
hipSPARSE	Yes	Yes
hipcub	Yes	Yes

### 3.13.1.1.7.3 Meta Packages

Package	Debian	RPM
ROCm Master Package	rocm	rocm-1.6.77-Linux.rpm
ROCm Developer Master Package	rocm-dev	rocm-dev-1.6.77-Linux.rpm
ROCm Libraries Master Package	rocm-libs	rocm-libs-1.6.77-Linux.rpm
ATMI	atmi	atmi-0.3.7-45-gde867f2-Linux.rpm
HIP Core	hip_base	hip_base-1.2.17263.rpm
HIP Documents	hip_doc	hip_doc-1.2.17263.rpm
HIP Compiler	hip_hcc	hip_hcc-1.2.17263.rpm
HIP Samples	hip_samples	hip_samples-1.2.17263.rpm.
HIPBLAS	hipblas	hipblas-0.4.0.3-Linux.rpm
MIOpen OpenCL Lib	miopen-opencl.	MIOpen-OpenCL-1.0.0-Linux.rpm
rocBLAS	rocblas	rocblas-0.4.2.3-Linux.rpm
rocFFT	rocfft	rocm-device-libs-0.0.1-Linux.rpm
ROCm Device Libs	rocm-device-libs	rocm-device-libs-0.0.1-Linux.rpm
ROCm OpenCL for Dev with CL headers	rocm-opencl-dev	rocm-opencl-devel-1.2.0-1424893.x86_64.rpm
ROCm GDB	rocm-gdb	rocm-gdb-1.5.265-gc4fb045.x86_64.rpm
RCP profiler	rocm-profiler	rocm-profiler-5.1.6386-gbaddcc9.x86_64.rpm
ROCm SMI Tool	rocm-smi	rocm-smi-1.0.0_24_g68893bc-1.x86_64.rpm
ROCm Utilities	rocm-utils	rocm-utils-1.0.0-Linux.rpm

## 3.14 Hardware and Software Support Information

- [Hardware and Software Support](#)
- [Radeon Instinct™ GPU-Powered HPC Solutions](#)



## 3.15 AMD Instinct™ High Performance Computing and Tuning Guide

HPC workloads have unique requirements. The default hardware and BIOS configurations for OEM platforms may not provide optimal performance for HPC workloads. To help enable optimal HPC settings on a per-platform and workload level, this guide calls out:

- BIOS settings that can impact performance
- hardware configuration best practices
- supported versions of operating systems
- workload-specific recommendations for optimal BIOS and operating system settings

There is also a discussion on the AMD Instinct™ software development environment, including information on how to install and run the DGEMM and STREAM benchmarks as well as GROMACS. This guidance provides a good starting point but is not exhaustively tested across all compilers.

Prerequisites to understanding this document and to perform tuning of HPC applications include:

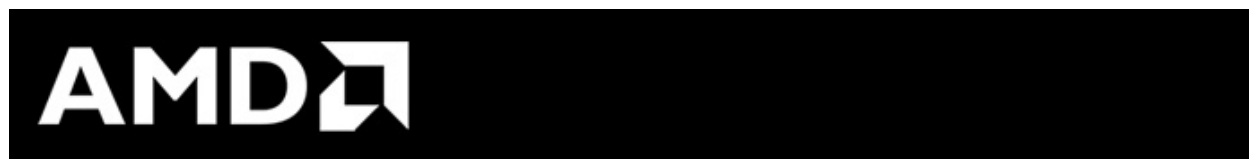
- Experience configuring servers
- Administrative access to the Server's Management Interface (BMC)
- Administrative access to the operating system
- Familiarity with OEMs Server's Management Interface (BMC) is strongly recommended
- Familiarity with the OS specific tools for configuration, monitoring and troubleshooting is strongly recommended

This document provides guidance on tuning systems with AMD Instinct™ accelerators for High Performance Computing (HPC) workloads. This document is not an all-inclusive guide, and some items referred to may have similar, but different, names in various OEM systems (for example, OEM-specific BIOS settings). This document also provides suggestions on items that should be the initial focus of additional, application-specific tuning.

This document is based on the AMD EPYC™ 7002 series processor family (former codename “Rome”). One can expect very similar results for the AMD EYPC™ 7003 series processor family (former codename “Milan”). Specific differences in the configuration options or performance obtained will be explicitly called out through the document where needed.

While this guide is a good starting point, developers are encouraged to perform their own performance testing for additional tuning.

For more details, refer to the [AMD Instinct™ High Performance Computing and Tuning Guide](#)



## 3.16 HIP Programming Guide v4.5

Heterogeneous-Computing Interface for Portability (HIP) is a C++ dialect designed to ease conversion of CUDA applications to portable C++ code. It provides a C-style API and a C++ kernel language. The C++ interface can use templates and classes across the host/kernel boundary.

The HIPify tool automates much of the conversion work by performing a source-to-source transformation from CUDA to HIP. HIP code can run on AMD hardware (through the HCC compiler) or NVIDIA hardware (through the NVCC compiler) with no performance loss compared with the original CUDA code.

Programmers familiar with other GPGPU languages will find HIP easy to learn and use. AMD platforms implement this language using the HC dialect providing similar low-level control over the machine.

Use HIP when converting CUDA applications to portable C++ and for new projects that require portability between AMD and NVIDIA. HIP provides a C++ development language and access to the best development tools on both platforms.



### 3.16.1 Programming Guide (PDF)

You can access and download the latest version of the HIP Programming Guide.

[Download PDF](#)

### 3.16.2 Related Topics

#### 3.16.2.1 HIP API Guide

You can access the Doxygen-generated HIP API Guide at the following location:

<https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD-HIP-API-4.5.pdf>

#### 3.16.2.2 HIP\_Supported\_CUDA\_API\_Reference\_Guide

You can access and download the latest version of the HIP-Supported CUDA API Reference Guide.

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Supported\\_CUDA\\_API\\_Reference\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Supported_CUDA_API_Reference_Guide.pdf)

#### 3.16.2.3 AMD ROCm Compiler Reference Guide

You can access and download the AMD ROCm Compiler Reference Guide at,

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_Compiler\\_Reference\\_Guide\\_v4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_Compiler_Reference_Guide_v4.5.pdf)

#### 3.16.2.4 HIP Installation Instructions

For HIP installation instructions, refer to

[https://rocmdocs.amd.com/en/latest/Installation\\_Guide/HIP-Installation.html](https://rocmdocs.amd.com/en/latest/Installation_Guide/HIP-Installation.html)

#### 3.16.2.5 HIP FAQ

- [HIP-FAQ](#)

## 3.17 HIP API Documentation v4.5

You can access the latest Doxygen-generated HIP API Guide at the following location:

<https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD-HIP-API-4.5.pdf>

## 3.18 HIP-Supported CUDA API Reference Guide v4.5

You can access the latest Reference guide at,

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Supported\\_CUDA\\_API\\_Reference\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Supported_CUDA_API_Reference_Guide.pdf)

## 3.19 AMD ROCm Compiler Reference Guide v4.5

You can access and download the AMD ROCm Compiler Reference Guide at,

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_Compiler\\_Reference\\_Guide\\_v4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_Compiler_Reference_Guide_v4.5.pdf)

### 3.19.1 Supported CUDA APIs

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Supported\\_CUDA\\_API\\_Reference\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Supported_CUDA_API_Reference_Guide.pdf)

To access the following supported CUDA APIs, see

[https://rocm.docs.amd.com/en/latest/Programming\\_Guides/Programming-Guides.html#hip-faq-porting-guide-and-programming-guide](https://rocm.docs.amd.com/en/latest/Programming_Guides/Programming-Guides.html#hip-faq-porting-guide-and-programming-guide)

- Runtime API
- Driver API
- cuComplex API
- cuBLAS
- cuRAND
- cuDNN
- cuFFT
- cuSPARSE

### 3.19.2 Deprecated HIP APIs

#### 3.19.2.1 HIP Context Management APIs

CUDA supports cuCtx API, the Driver API that defines “Context” and “Devices” as separate entities. Contexts contain a single device, and a device can theoretically have multiple contexts. HIP initially added limited support for APIs to facilitate easy porting from existing driver codes. The APIs are marked as deprecated now as there is a better alternate interface (such as hipSetDevice or the stream API) to achieve the required functions.

- hipCtxPopCurrent
- hipCtxPushCurrent
- hipCtxSetCurrent
- hipCtxGetCurrent
- hipCtxGetDevice
- hipCtxGetApiVersion

- hipCtxGetCacheConfig
- hipCtxSetCacheConfig
- hipCtxSetSharedMemConfig
- hipCtxGetSharedMemConfig
- hipCtxSynchronize
- hipCtxGetFlags
- hipCtxEnablePeerAccess
- hipCtxDisablePeerAccess

## 3.20 OpenCL Programming Guide

- [Opencl-Programming-Guide](#)
- [Optimization-Opencl](#)

## 3.21 OpenMP Support

### 3.21.1 Overview

The ROCm installation includes an LLVM-based implementation that fully supports the OpenMP 4.5 standard and a subset of the OpenMP 5.0 standard. Fortran, C/C++ compilers, and corresponding runtime libraries are included. Along with host APIs, the OpenMP compilers support offloading code and data onto GPU devices. The GPUs supported are the same as those supported by this ROCm release. This document briefly describes the installation location of the OpenMP toolchain and example usage of device offloading.

### 3.21.2 Installation

The OpenMP toolchain is automatically installed as part of the standard ROCm installation and is available under `/opt/rocm-{version}/llvm`. The sub-directories are:

- `bin`: Compilers (flang and clang) and other binaries
- `examples`: How to compile and run these programs is shown in the usage section below.
- `include`: Header files
- `lib`: Libraries including those required for target offload
- `lib-debug`: Debug versions of the above libraries

### 3.21.3 Usage

The example programs can be compiled and run by pointing the environment variable AOMP to the OpenMP install directory. For example:

```
% export AOMP=/opt/rocm-{version}/llvm
% cd $AOMP/examples/openmp/veccopy
% make run
```

The above invocation of Make will compile and run the program. Note, the options that are required for target offload from an OpenMP program:

```
-target x86_64-pc-linux-gnu -fopenmp -fopenmp-targets=amdgcN-amd-amdhsa -Xopenmp-
↪target=amdgcN-amd-amdhsa -march=<gpu-arch>
```

The value of gpu-arch can be obtained by running the following command:

```
% /opt/rocm-{version}/bin/rocmInfo | grep gfx
```

### 3.21.4 Helpful Tips

Setting the environment variable LIBOMPTARGET\_KERNEL\_TRACE while running an OpenMP program produces valuable information. Among other details, a value of 1 will lead the runtime to emit the number of teams and threads for every kernel run on the GPU. A value of 2 leads additionally to a trace of implementation-level APIs and corresponding timing information.

## 3.22 ROCm Libraries

Libraries are listed alphabetically below.

[hipSOLVER User Guide](#)

[MIGraphX 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](#)

### 3.22.1 Deprecated Libraries

#### 3.22.1.1 hipeigen

Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. hipeigen has been upstream to the main project at <https://eigen.tuxfamily.org/>.

## 3.23 Deep Learning

### 3.23.1 MIOpen API

- [MIOpen API](#)
- [MIOpenGEMM API](#)

### 3.23.2 TensorFlow

#### 3.23.2.1 AMD ROCm Tensorflow v1.15 Release

We are excited to announce the release of ROCm enabled TensorFlow v1.15 for AMD GPUs.

**In this release we have the following features enabled on top of upstream TF1.15 enhancements:**

- We integrated ROCm RCCL library for mGPU communication, details in [RCCL github repo](#)
- XLA backend is enabled for AMD GPUs, the functionality is complete, performance optimization is in progress.

#### 3.23.2.2 AMD ROCm Tensorflow v2.2.0-beta1 Release

In addition to Tensorflow v1.15 release, we also enabled Tensorflow v2.2.0-beta1 for AMD GPUs. The TF-ROCm 2.2.0-beta1 release supports Tensorflow V2 API. Both whl packages and docker containers are available below.

#### 3.23.2.3 Tensorflow Installation

1. Install the open-source AMD ROCm 3.3 stack. For details, see [here](#)
2. Install other relevant ROCm packages.

```
sudo apt update
sudo apt install rocm-libs miopen-hip cxx-activtylogger rccl
```

3. Install TensorFlow itself (via the Python Package Index).

```
sudo apt install wget python3-pip
# Pip3 install the whl package from PyPI
pip3 install --user tensorflow-rocm #works only with python3.8 or prior
```

Tensorflow v2.2.0 is installed.

### 3.23.2.3.1 Tensorflow ROCm port: Basic installation on RHEL

The following instructions provide a starting point for using the TensorFlow ROCm port on RHEL.

**Note** It is recommended to start with a clean RHEL 8.2 system.

#### 3.23.2.3.1.1 Install ROCm

1. Use the instructions below to add the ROCm repository.

```
export RPM_ROCM_REPO=https://repo.radeon.com/rocm/yum/3.7
```

2. Install the following packages.

```
# Enable extra repositories
yum --enablerepo=extras install -y epel-release

# Install required base build and packaging commands for ROCm
yum -y install \
    bc \
    cmake \
    cmake3 \
    dkms \
    dpkg \
    elfutils-libelf-devel \
    expect \
    file \
    gettext \
    gcc-c++ \
    git \
    libgcc \
    ncurses \
    ncurses-base \
    ncurses-libs \
    numactl-devel \
    numactl-libs \
    libunwind-devel \
    libunwind \
    llvm \
    llvm-libs \
    make \
    pciutils \
    pciutils-devel \
    pciutils-libs \
    python36 \
    python36-devel \
    pkgconfig \
    qemu-kvm \
    wget
```

3. Install ROCm packages.

```
# Add the ROCm package repo location
echo -e "[ROCM]\nname=ROCM\nbaseurl=$RPM_ROCM_REPO\nenabled=1\npgpcheck=0" >> /etc/
↪yum.repos.d/rocm.repo
```

(continues on next page)

(continued from previous page)

```
# Install the ROCm rpms
sudo yum clean all
sudo yum install -y rocm-dev
sudo yum install -y hipblas hipcub hipsparse miopen-hip miopengemm rccl rocblas_
↪rocfft rocprim rocrand
```

#### 4. Ensure the ROCm target list is set up.

```
bash -c 'echo -e "gfx803\ngfx900\ngfx906\ngfx908" >> $ROCM_PATH/bin/target.lst'
```

#### 5. Install the required Python packages.

```
pip3.6 install --user \
    cget \
    pyyaml \
    pip \
    setuptools==39.1.0 \
    virtualenv \
    absl-py \
    six==1.10.0 \
    protobuf==3.6.1 \
    numpy==1.18.2 \
    scipy==1.4.1 \
    scikit-learn==0.19.1 \
    pandas==0.19.2 \
    gnureadline \
    bz2file \
    wheel==0.29.0 \
    portpicker \
    werkzeug \
    grpcio \
    astor \
    gast \
    termcolor \
    h5py==2.8.0 \
    keras_preprocessing==1.0.5
```

#### 6. Install TensorFlow.

```
# Install ROCm manylinux WHL
wget <location of WHL file>
pip3.6 install --user ./tensorflow*linux_x86_64.whl
```

### 3.23.2.4 Tensorflow benchmarking

Clone the repository of bench test and run it

```
cd ~ && git clone https://github.com/tensorflow/benchmarks.git
python3 ~/benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=resnet50
```

### 3.23.2.5 Tensorflow Installation with Docker

**Note:** firstly, configure docker environment for ROCm (information [here](#))

Pull the docker images for Tensorflow releases with ROCm backend support. The size of these docker images is about 7 Gb.

```
sudo docker pull rocm/tensorflow:latest
```

Launch the downloaded docker image

```
alias drun='sudo docker run -it --network=host --device=/dev/kfd --device=/dev/dri --
↪ipc=host --shm-size 16G --group-add video --cap-add=SYS_PTRACE --security-opt_
↪seccomp=unconfined -v $HOME/dockerx:/dockerx'

#Run it
drun rocm/tensorflow:latest
```

More information about tensorflow docker images can be found [here](#)

### 3.23.2.6 Tensorflow More Resources

The official github repository is [here](#)

## 3.23.3 MIOpen

### 3.23.3.1 ROCm MIOpen v2.0.1 Release

Announcing our new Foundation for Deep Learning acceleration MIOpen 2.0 which introduces support for Convolution Neural Network (CNN) acceleration — built to run on top of the ROCm software stack!

This release includes the following:

- This release contains bug fixes and performance improvements.
- Additionally, the convolution algorithm Implicit GEMM is now enabled by default
- **Known issues:**
  - Backward propagation for batch normalization in fp16 mode may trigger NaN in some cases
  - Softmax Log mode may produce an incorrect result in back propagation
- [Source code](#)
- **Documentation**
  - [MIOpen](#)
  - [MIOpenGemm](#)

#### Changes:

- Added Winograd multi-pass convolution kernel
- Fixed issue with hip compiler paths
- Fixed immediate mode behavior with auto-tuning environment variable
- Fixed issue with system find-db in-memory cache, the fix enable the cache by default
- Improved logging



- Improved how symbols are hidden in the library
- Updated default behavior to enable implicit GEMM

### 3.23.3.2 Porting from cuDNN to MIOpen

The [porting guide](#) highlights the key differences between the current cuDNN and MIOpen APIs.

### 3.23.3.3 The ROCm 3.3 has prebuilt packages for MIOpen

Install the ROCm MIOpen implementation (assuming you already have the ‘rocm’ and ‘rocm-opengl-dev’ package installed):

MIOpen can be installed on Ubuntu using

```
apt-get
```

#### For just OpenCL development

```
sudo apt-get install miopengemm miopen-opengl
```

#### For HIP development

```
sudo apt-get install miopengemm miopen-hip
```

Or you can build from [source code](#)

Currently both the backends cannot be installed on the same system simultaneously. If a different backend other than what currently exists on the system is desired, please uninstall the existing backend completely and then install the new backend.

## 3.23.4 PyTorch

### 3.23.4.1 Building PyTorch for ROCm

This is a quick guide to setup PyTorch with ROCm support inside a docker container. Assumes a .deb based system. See [ROCm install](#) for supported operating systems and general information on the ROCm software stack.

Note: Currently, ROCm install version 3.3 is required.

1. Install or update rocm-dev on the host system:

```
sudo apt-get install rocm-dev  
or  
sudo apt-get update  
sudo apt-get upgrade
```

### 3.23.4.2 Recommended: Install using published PyTorch ROCm docker image:

2. Obtain docker image:

```
docker pull rocm/pytorch:rocm4.0_ubuntu18.04_py3.6_pytorch
```

3. Start a docker container using the downloaded image:

```
sudo docker run -it -v $HOME:/data --privileged --rm --device=/dev/kfd --device=/dev/
↳dri --group-add video rocm/pytorch:rocm3.7_ubuntu16.04_py3.6_pytorch
```

4. Confirm working installation:

```
PYTORCH_TEST_WITH_ROCM=1 python3.6 test/run_test.py --verbose
```

**Note:** Compilation and installation must be correct for the tests to be successful.

5. Install torchvision:

```
pip install torchvision
```

This step is optional but most PyTorch scripts will use torchvision to load models. E.g., running the pytorch examples requires torchvision.

### 3.23.4.3 Option 2: Install using PyTorch upstream docker file

1. Clone PyTorch repository on the host:

```
cd ~
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git submodule init
git submodule update
```

2. Build PyTorch docker image:

```
cd pytorch/docker/caffe2/jenkins
./build.sh py2-clang7-rocmdeb-ubuntu16.04

A message "Successfully built <image_id>" indicates a successful completion of this_
↳step.
```

**Note:** These steps are not tested and validated on other software versions.

3. Start a docker container using the new image:

```
sudo docker run -it -v $HOME:/data --privileged --rm --device=/dev/kfd --device=/dev/
↳dri --group-add video <image_id>
```

**Note:** This will mount your host home directory on /data in the container.

4. Change to previous PyTorch checkout from within the running docker:

```
cd /data/pytorch
```

5. Build PyTorch for ROCm:

Unless you are running a gfx900/Vega10-type GPU (MI25, Vega56, Vega64,...), explicitly export the GPU architecture to build for, e.g.: export HCC\_AMDGPU\_TARGET=gfx906

then

```
.jenkins/pytorch/build.sh
```

This will hipify the PyTorch sources first, and then compile using 4 concurrent jobs. Note, the docker image requires 16 GB of RAM.

6. Confirm working installation:

```
PYTORCH_TEST_WITH_ROCM=1 python test/run_test.py --verbose
```

No tests will fail if the compilation and installation is correct.

7. Install torchvision:

```
pip install torchvision
```

This step is optional; however, most PyTorch scripts use torchvision to load models. For example, running the pytorch examples requires torchvision.

8. Commit the container to preserve the pytorch install (from the host):

```
sudo docker commit <container_id> -m 'pytorch installed'
```

#### 3.23.4.4 Option 3: Install using minimal ROCm docker file

1. Download dockerfile based on the OS choose: Recommend to use - Dockerfile-<OS distro>-complete to get all the ROCm Math libs installed which are required for PyTorch.

Dockerfile

2. Build docker image:

```
sudo docker build -f ./Dockerfile-<OS distro>-complete .
```

The message “Successfully built <image\_id>” indicates a successful completion of this step.

3. Start a docker container using the new image:

```
sudo docker run -it -v $HOME:/data --privileged --rm --device=/dev/kfd --device=/dev/
↵dri --group-add video <image_id>
```

Note: This will mount your host home directory on /data in the container.

4. Clone pytorch master (on to the host):

```
cd ~
git clone https://github.com/pytorch/pytorch.git or git clone https://github.com/
↵ROCmSoftwarePlatform/pytorch.git
cd pytorch
git submodule init
git submodule update --init --recursive'
```

5. Run “hipify” to prepare source code (in the container):

```
python3 tools/amd_build/build_amd.py
```

#### 6. Build and install pytorch:

By default pytorch is built for all supported AMD GPU targets like gfx900/gfx906/gfx908 (MI25, MI50, MI60, MI100, ...) This can be overwritten using `export PYTORCH_ROCM_ARCH=gfx900;gfx906;gfx908`

then

```
USE_ROCM=1 MAX_JOBS=4 python3 setup.py install --user
```

Use `MAX_JOBS=n` to limit peak memory usage. If building fails try falling back to fewer jobs. 4 jobs assume available main memory of 16 GB or larger.

#### 7. Confirm working installation:

```
PYTORCH_TEST_WITH_ROCM=1 python3 test/run_test.py --verbose
```

No tests will fail if the compilation and installation is correct.

#### 8. Install torchvision:

```
pip3 install --user "git+https://github.com/pytorch/vision.git"
```

This step is optional. However, most PyTorch scripts will use torchvision to load models. For example, running the PyTorch examples requires torchvision.

#### 9. Commit the container to preserve the pytorch install (from the host):

```
sudo docker commit <container_id> -m 'pyTorch installed'
```

### 3.23.4.5 PyTorch examples

#### 1. Clone the PyTorch examples repository:

```
git clone https://github.com/pytorch/examples.git && cd examples/
```

#### 2. Download pip requiremenst:

```
pip3 install -r mnist/requirements.txt
```

#### 3. Run individual example: Super-resolution training and running

```
cd super_resolution/

# download dataset for training and run learning
python3 main.py --upscale_factor 3 --batchSize 4 --testBatchSize 100 --nEpochs 30 --
→lr 0.001

# test work super resolution effect
python3 super_resolve.py --input_image dataset/BSDS300/images/test/16077.jpg \
--model model_epoch_30.pth --output_filename out.png
```

#### 4. Open *out.png* and *dataset/BSDS300/images/test/16077.jpg* files to see result

### 3.23.4.6 Building Caffe2 for ROCm

This is a quick guide to setup Caffe2 with ROCm support inside docker container and run on AMD GPUs. Caffe2 with ROCm support offers complete functionality on a single GPU achieving great performance on AMD GPUs using both native ROCm libraries and custom hip kernels. This requires your host system to have rocm-3.3s drivers installed. Please refer to [ROCm install](#) to install ROCm software stack. If your host system doesn't have docker installed, please refer to [docker install](#). It is recommended to add the user to the docker group to run docker as a non-root user, please refer [here](#).

**This guide provides two options to run Caffe2.**

1. Launch the docker container using a docker image with Caffe2 installed.
2. Build Caffe2 from source inside a Caffe2 ROCm docker image.

### 3.23.4.7 Option 1: Docker image with Caffe2 installed:

This option provides a docker image which has Caffe2 installed. Users can launch the docker container and train/run deep learning models directly. This docker image will run on both gfx900(Vega10-type GPU - MI25, Vega56, Vega64,...) and gfx906(Vega20-type GPU - MI50, MI60)

1. Launch the docker container

```
docker run -it --network=host --device=/dev/kfd --device=/dev/dri --group-add video_
rocm/pytorch:rocm3.7_ubuntu16.04_py3.6_caffe2
```

This will automatically download the image if it does not exist on the host. You can also pass -v argument to mount any data directories on to the container.

### 3.23.4.8 Option 2: Install using Caffe2 ROCm docker image:

1. Clone PyTorch repository on the host:

```
cd ~
git clone --recurse-submodules https://github.com/pytorch/pytorch.git
cd pytorch
git submodule update --init --recursive
```

2. Launch the docker container

```
docker pull rocm/pytorch:rocm3.7_ubuntu16.04_py3.6_caffe2
docker run -it --network=host --device=/dev/kfd --device=/dev/dri --group-add video -
-v $PWD:/pytorch rocm/pytorch:rocm3.7_ubuntu16.04_py3.6_caffe2
```

3. Build Caffe2 from source

```
cd /pytorch
```

If running on gfx900/vega10-type GPU(MI25, Vega56, Vega64,...)

```
./jenkins/caffe2/build.sh
```

If running on gfx906/vega20-type GPU(MI50, MI60)

```
HCC_AMDGPU_TARGET=gfx906 ./jenkins/caffe2/build.sh
```

### 3.23.4.9 Test the Caffe2 Installation

To validate Caffe2 installation, run

#### 1. Test Command

```
cd ~ && python -c 'from caffe2.python import core' 2>/dev/null && echo "Success" ||  
↪ echo "Failure"
```

#### 2. Running unit tests in Caffe2

```
cd /pytorch  
.jenkins/caffe2/test.sh
```

### 3.23.4.10 Run benchmarks

Caffe2 benchmarking script supports the following networks MLP, AlexNet, OverFeat, VGGA, Inception

To run benchmarks for networks MLP, AlexNet, OverFeat, VGGA, Inception run the command from pytorch home directory replacing `<name_of_the_network>` with one of the networks.

```
python caffe2/python/convnet_benchmarks.py --batch_size 64 --model <name_of_the_  
↪ network> --engine MIOOPEN
```

### 3.23.4.11 Running example scripts

Please refer to the example scripts in `caffe2/python/examples`. It currently has `resnet50_trainer.py` which can run ResNet's, ResNeXt's with various layer, groups, depth configurations and `char_rnn.py` which uses RNNs to do character level prediction.

### 3.23.4.12 Building own docker images

After cloning the pytorch repository, you can build your own Caffe2 ROCm docker image. Navigate to pytorch repo and run

```
cd docker/caffe2/jenkins  
./build.sh py2-clang7-rocmdeb-ubuntu16.04
```

This should complete with a message “Successfully built `<image_id>`” which can then be used to install Caffe2 as in Option 2 above.

## 3.24 MIVisionX

MIVisionX toolkit is a set of comprehensive computer vision and machine intelligence libraries, utilities, and applications bundled into a single toolkit. AMD MIVisionX delivers highly optimized open source implementation of the [Khronos OpenVX™](#) and OpenVX™ Extensions along with Convolution Neural Net Model Compiler & Optimizer supporting [ONNX](#), and [Khronos NNEF™](#) exchange formats. The toolkit allows for rapid prototyping and deployment of optimized workloads on a wide range of computer hardware, including small embedded x86 CPUs, APUs, discrete GPUs, and heterogeneous servers.

- [AMD OpenVX](#)
- [AMD OpenVX Extensions](#)

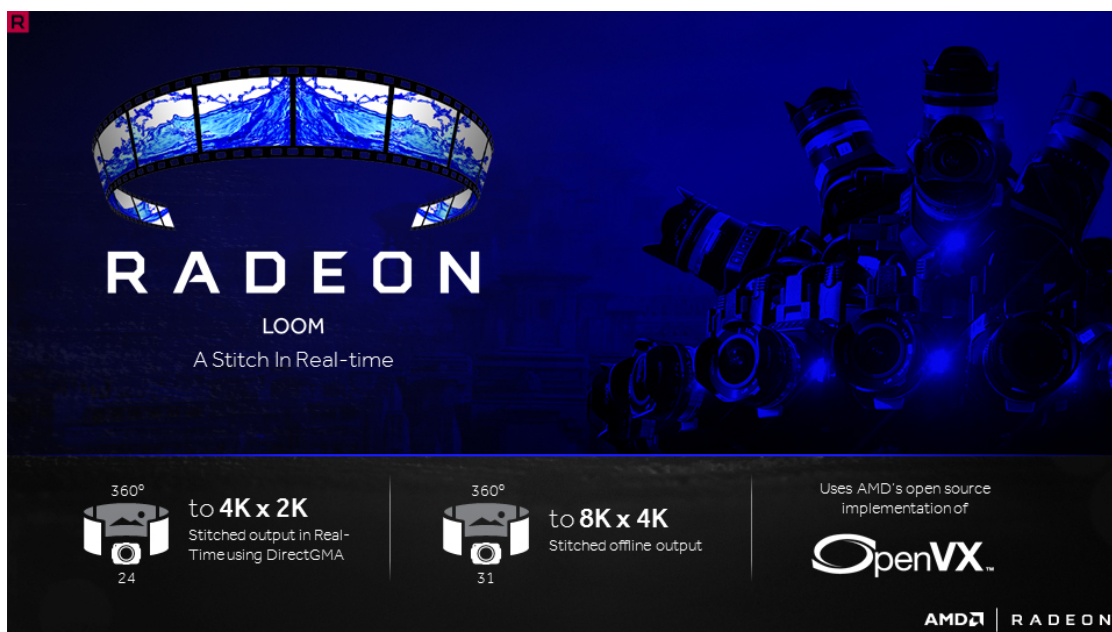
- Loom 360 Video Stitch Library
  - Neural Net Library
  - OpenCV Extension
  - RPP Extension
  - WinML Extension
- Applications
- Neural Net Model Compiler and Optimizer
- RALI
- Samples
- Toolkit
- Utilities
  - Inference Generator
  - Loom Shell
  - RunCL
  - RunVX
- Prerequisites
- Build and Install MIVisionX
- Verify the Installation
- Docker
- Release Notes



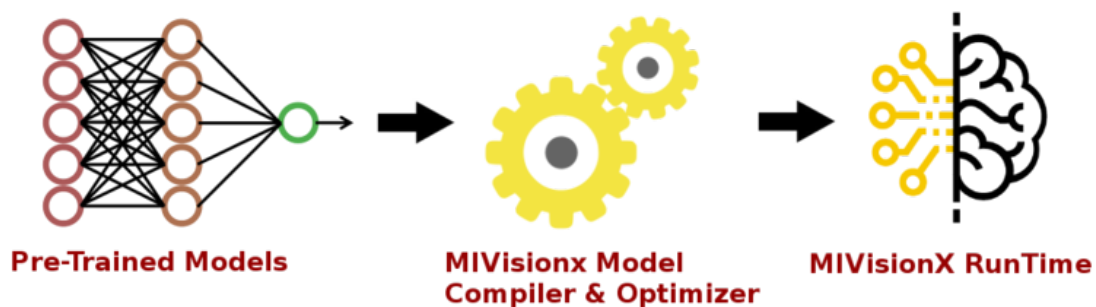
AMD OpenVX [[amd\\_openvx](#)] is a highly optimized open source implementation of the [Khronos OpenVX](#) computer vision specification. It allows for rapid prototyping as well as fast execution on a wide range of computer hardware, including small embedded x86 CPUs and large workstation discrete GPUs.

The OpenVX framework provides a mechanism to add new vision functions to OpenVX by 3rd party vendors. This project has below mentioned OpenVX [modules](#) and utilities to extend [amd\\_openvx](#) project, which contains the AMD OpenVX Core Engine.

- [amd\\_loomsl](#): AMD Radeon Loom stitching library for live 360 degree video applications.

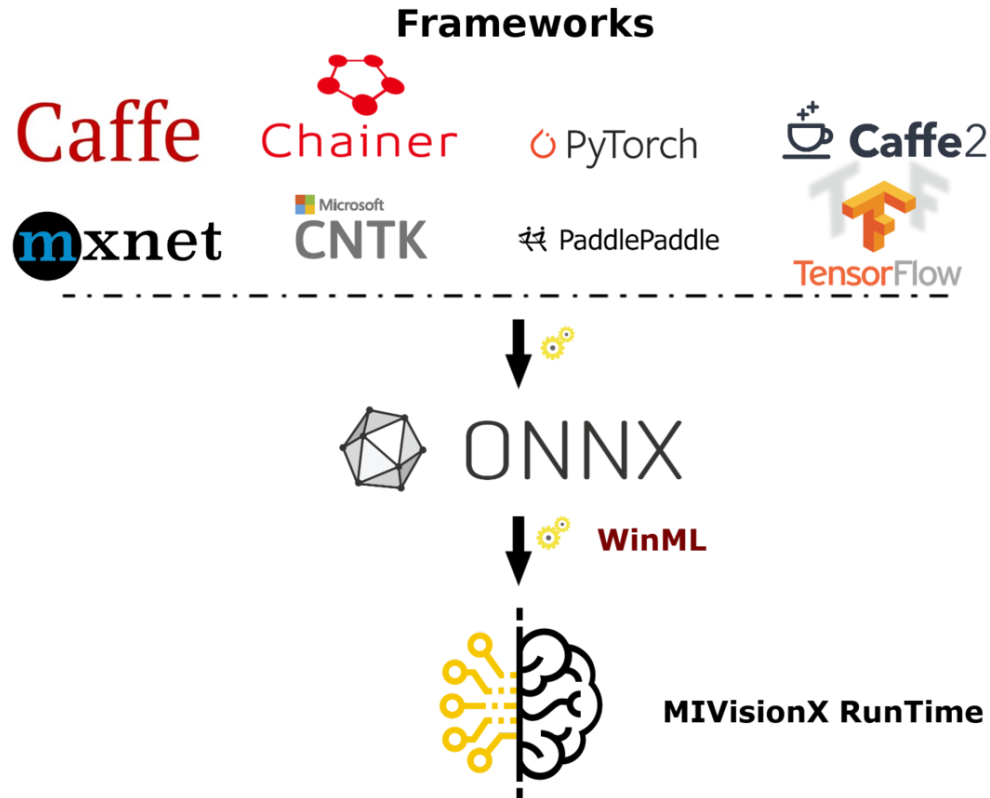


- `amd_nn`: OpenVX neural network module



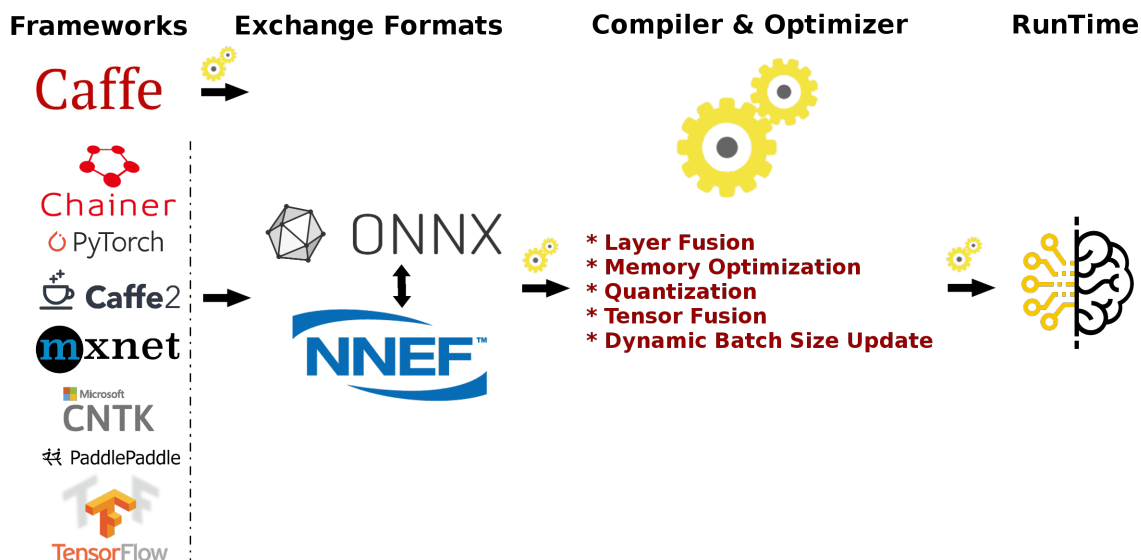
- `amd_opencv`: OpenVX module that implements a mechanism to access OpenCV functionality as OpenVX kernels
- `amd_winml`: WinML extension will allow developers to import a pre-trained ONNX model into an OpenVX graph and add hundreds of different pre & post processing vision/generic/user-defined functions, available in OpenVX and OpenCV interop, to the input and output of the neural net model. This will allow developers to build an end to end application for inference.





MIVisionX has a number of [applications](#) built on top of OpenVX modules, it uses AMD optimized libraries to build applications which can be used to prototype or used as models to develop a product.

- [Cloud Inference Application](#): This sample application does inference using a client-server system.
- [Digit Test](#) This sample application is used to recognize hand written digits.
- [MIVisionX OpenVX Classification](#): This sample application shows how to run supported pre-trained caffe models with MIVisionX RunTime.
- [MIVisionX WinML Classification](#): This sample application shows how to run supported ONNX models with MIVisionX RunTime on Windows.
- [MIVisionX WinML YoloV2](#): This sample application shows how to run tiny yolov2(20 classes) with MIVisionX RunTime on Windows.
- [External Applications](#)



Neural Net Model Compiler & Optimizer `model_compiler` converts pre-trained neural net models to MIVisionX run-time code for optimized inference.

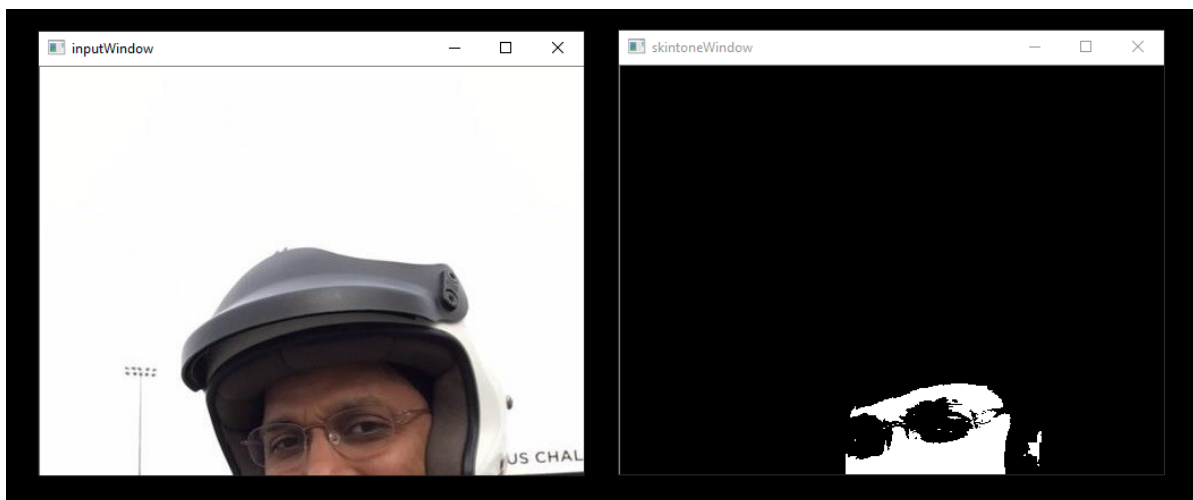
The Radeon Augmentation Library `RALI` is designed to efficiently decode and process images and videos from a variety of storage formats and modify them through a processing graph programmable by the user.

MIVisionX samples using OpenVX and OpenVX extension libraries

### GDF - Graph Description Format

MIVisionX samples using `runvx` with GDF

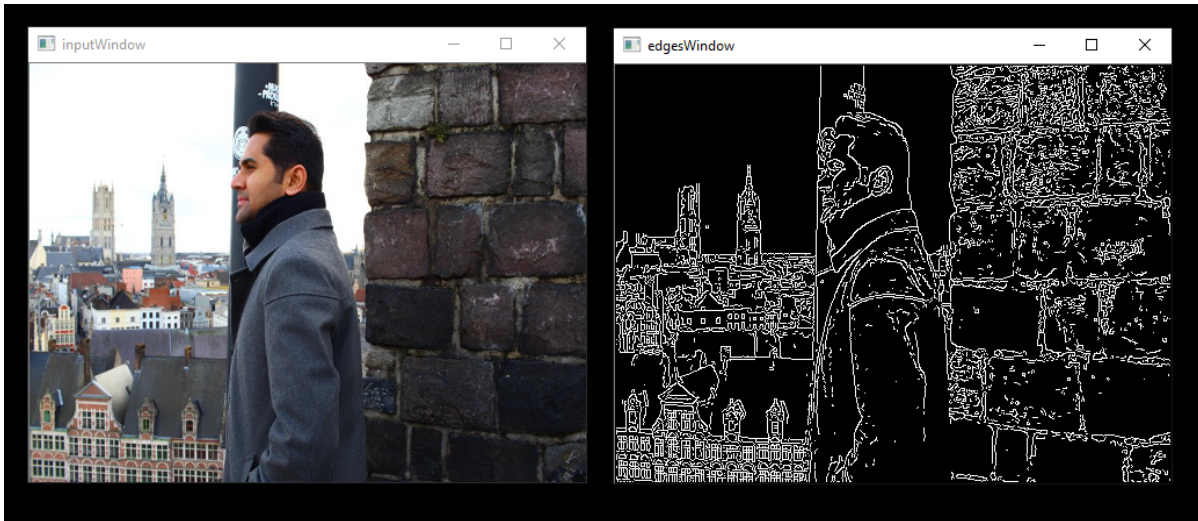
`skintonedetect.gdf`



usage:

```
runvx skintonedetect.gdf
```

`canny.gdf`



usage:

```
runvx canny.gdf
```

### **skintonedetect-LIVE.gdf**

Using live camera

usage:

```
runvx -frames:live skintonedetect-LIVE.gdf
```

### **canny-LIVE.gdf**

Using live camera

usage:

```
runvx -frames:live canny-LIVE.gdf
```

### **OpenCV\_orb-LIVE.gdf**

Using live camera

usage:

```
runvx -frames:live OpenCV_orb-LIVE.gdf
```

**Note:** More samples available on [GitHub](#)

**MIVisionX Toolkit**, is a comprehensive set of help tools for neural net creation, development, training, and deployment. The Toolkit provides you with helpful tools to design, develop, quantize, prune, retrain, and infer your neural network work in any framework. The Toolkit is designed to help you deploy your work to any AMD or 3rd party hardware, from embedded to servers.

MIVisionX provides you with tools for accomplishing your tasks throughout the whole neural net life-cycle, from creating a model to deploying them for your target platforms.

- **inference\_generator**: generate inference library from pre-trained Caffe models
- **loom\_shell**: an interpreter to prototype 360 degree video stitching applications using a script
- **RunVX**: command-line utility to execute OpenVX graph described in GDF text file

- **RunCL**: command-line utility to build, execute, and debug OpenCL programs
- CPU: SSE4.1 or above CPU, 64-bit
- GPU: **GFX7 or above** [optional]
- APU: Carrizo or above [optional]

**Note:** Some modules in MIVisionX can be built for CPU only. To take advantage of advanced features and modules we recommend using AMD GPUs or AMD APUs.

### Windows

- Windows 10
- Windows SDK
- Visual Studio 2017
- Install the latest drivers and *OpenCL SDK* <<https://github.com/GPUOpen-LibrariesAndSDKs/OCL-SDK/releases/tag/1.0>>\_
- **OpenCV**
  - Set OpenCV\_DIR environment variable to OpenCV/build folder
  - Add %OpenCV\_DIR%\x64vc14bin or %OpenCV\_DIR%\x64vc15bin to your PATH

### Linux

- Install **ROCm**
- ROCm CMake, MIOpenGEMM & MIOpen for Neural Net Extensions (vx\_nn)
- CMake 2.8 or newer [download](#)
- Qt Creator for [Cloud Inference Client](#)
- **Protobuf for inference generator & model compiler**
  - install libprotobuf-dev and protobuf-compiler needed for vx\_nn
- **OpenCV** <<https://github.com/opencv/opencv/releases/tag/3.4.0>>\_
  - Set OpenCV\_DIR environment variable to OpenCV/build folder
- **FFMPEG - Optional**
  - FFMPEG is required for amd\_media & mv\_deploy modules

For the convenience of the developer, we here provide the setup script which will install all the dependencies required by this project.

**MIVisionX-setup.py**- This script builds all the prerequisites required by MIVisionX. The setup script creates a deps folder and installs all the prerequisites, this script only needs to be executed once. If -d option for directory is not given the script will install deps folder in '~/' directory by default, else in the user specified folder.

### Prerequisites for running the scripts

- ubuntu 16.04/18.04 or CentOS 7.5/7.6
- **ROCm supported hardware**
- **ROCm**

usage:

```
python MIVisionX-setup.py --directory [setup directory - optional]
                        --installer [Package management tool - optional]
→ (default:apt-get) [options: Ubuntu:apt-get;CentOS:yum]]
                        --miopen      [MIOpen Version - optional (default:2.1.0)]
                        --miopengemm[MIOpenGEMM Version - optional (default:1.1.5)]
                        --ffmpeg      [FFMPEG Installation - optional (default:no)]
→ [options:Install ffmpeg - yes]]
                        --rpp         [RPP Installation - optional (default:yes)]
→ [options:yes/no]]
```

**Note:** use `--installer yum` for CentOS

## Windows

### Using .msi packages

- [MIVisionX-installer.msi](#): MIVisionX
- [MIVisionX\\_WinML-installer.msi](#): MIVisionX for WinML

### Using Visual Studio 2017 on 64-bit Windows 10

- Install [OpenCL\\_SDK](#)
- Install [OpenCV](#) with/without [contrib](#) to support camera capture, image display, & opencv extensions
  - Set `OpenCV_DIR` environment variable to OpenCV/build folder
  - Add `%OpenCV_DIR%\x64vc14bin` or `%OpenCV_DIR%\x64vc15bin` to your PATH
- Use `MIVisionX.sln` to build for x64 platform

**NOTE:** `vx_nn` is not supported on Windows in this release

## Linux

### Using apt-get/yum

#### Prerequisites

- Ubuntu 16.04/18.04 or CentOS 7.5/7.6
- [ROCm supported hardware](#)
- [ROCm](#)

### Ubuntu

```
sudo apt-get install mivisionx
```

### CentOS

```
sudo yum install mivisionx
```

#### Note:

- `vx_winml` is not supported on linux
- source code will not available with apt-get/yum install
- executables placed in `/opt/rocm/mivisionx/bin` and libraries in `/opt/rocm/mivisionx/lib`
- OpenVX and module header files into `/opt/rocm/mivisionx/include`
- model compiler, toolkit, & samples placed in `/opt/rocm/mivisionx`
- Package (.deb & .rpm) install requires OpenCV v3.4.0 to execute AMD OpenCV extensions

## Using MIVisionX-setup.py and CMake on Linux (Ubuntu 16.04/18.04 or CentOS 7.5/7.6) with ROCm

- Install [ROCm](#)
- Use the below commands to setup and build MIVisionX

```
git clone https://github.com/GPUOpen-ProfessionalCompute-Libraries/MIVisionX.git
cd MIVisionX
```

```
python MIVisionX-setup.py --directory [setup directory - optional]
                        --installer [Package management tool - optional]
→ (default:apt-get) [options: Ubuntu:apt-get;CentOS:yum]
                        --miopen [MIOpen Version - optional (default:2.1.0)]
                        --miopengemm [MIOpenGEMM Version - optional (default:1.1.5)]
                        --ffmpeg [FFMPEG Installation - optional (default:no)]
→ [options:Install ffmpeg - yes]]
                        --rpp [RPP Installation - optional (default:yes)]
→ [options:yes/no]]
```

**Note:** Use `--installer yum` for CentOS

```
mkdir build
cd build
cmake ../
make -j8
sudo make install
```

**Note:**

- vx\_winml is not supported on Linux
- the installer will copy all executables into `/opt/rocm/mivisionx/bin` and libraries into `/opt/rocm/mivisionx/lib`
- the installer also copies all the OpenVX and module header files into `/opt/rocm/mivisionx/include` folder

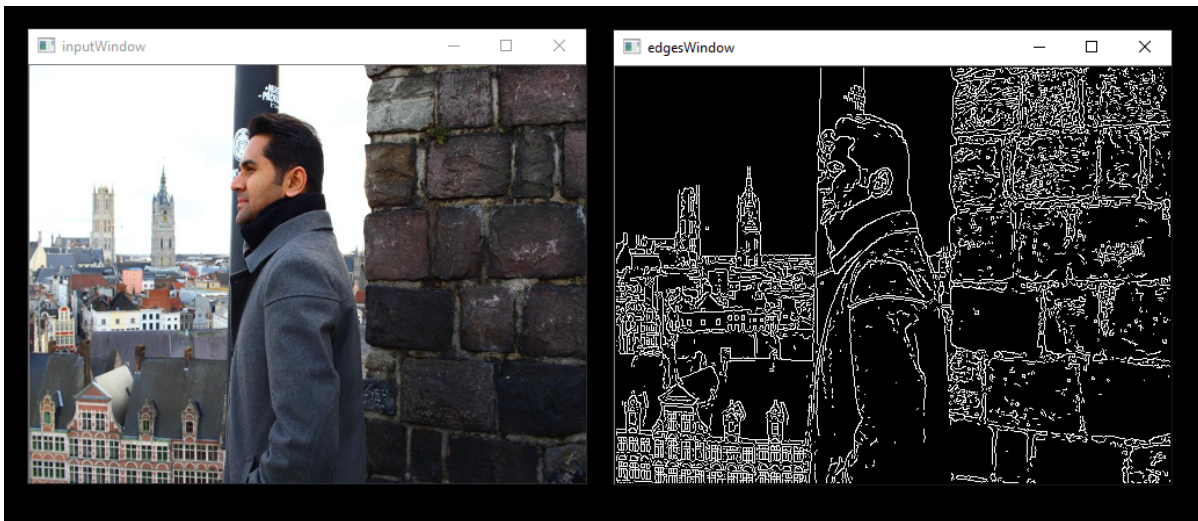
## Using CMake on Linux (Ubuntu 16.04 64-bit or CentOS 7.5 / 7.6 ) with ROCm

- Install [ROCm](#)
- **git clone, build and install other ROCm projects (using cmake and `% make install`) in the below order for vx\_nn.**
  - [rocm-cmake](#)
  - [MIOpenGEMM](#)
  - [MIOpen](#) – make sure to use `-DMIOPEN_BACKEND=OpenCL` option with cmake
- install [protobuf](#)
- install [OpenCV](#)
- install [FFMPEG n4.0.4](#) - Optional
- **build and install (using cmake and `% make install`)**
  - executables will be placed in bin folder
  - libraries will be placed in lib folder
  - the installer will copy all executables into `/opt/rocm/mivisionx/bin` and libraries into `/opt/rocm/lib`
  - the installer also copies all the OpenVX and module header files into `/opt/rocm/mivisionx/include` folder

- add the installed library path to LD\_LIBRARY\_PATH environment variable (default /opt/rocm/mivisionx/lib)
- add the installed executable path to PATH environment variable (default /opt/rocm/mivisionx/bin)

## Linux

- The installer will copy all executables into /opt/rocm/mivisionx/bin and libraries into /opt/rocm/mivisionx/lib
- The installer also copies all the OpenVX and OpenVX module header files into /opt/rocm/mivisionx/include folder
- Apps, Samples, Documents, Model Compiler and Toolkit are placed into /opt/rocm/mivisionx
- Run samples to verify the installation
  - **Canny Edge Detection**



```
export PATH=$PATH:/opt/rocm/mivisionx/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/rocm/mivisionx/lib
runvx /opt/rocm/mivisionx/samples/gdf/canny.gdf
```

Note: More samples are available [here](#)

MIVisionX provides developers with docker images for Ubuntu 16.04, Ubuntu 18.04, CentOS 7.5, & CentOS 7.6. Using docker images developers can quickly prototype and build applications without having to be locked into a single system setup or lose valuable time figuring out the dependencies of the underlying software.

## MIVisionX Docker

- [Ubuntu 16.04](#)
- [Ubuntu 18.04](#)
- [CentOS 7.5](#)
- [CentOS 7.6](#)

## Docker Workflow Sample on Ubuntu 16.04/18.04

### Prerequisites

- Ubuntu 16.04/18.04
- [rocm supported hardware](#)

## Workflow

### Step 1 - Install rocm-dkms

```
sudo apt update
sudo apt dist-upgrade
sudo apt install libnuma-dev
sudo reboot
```

```
wget -qO - https://repo.radeon.com/rocm/apt/debian/rocm.gpg.key | sudo apt-key add -
echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/debian/ xenial main' | sudo_
↳tee /etc/apt/sources.list.d/rocm.list
sudo apt update
sudo apt install rocm-dkms
sudo reboot
```

### Step 2 - Setup Docker

```
sudo apt-get install curl
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
↳$(lsb_release -cs) stable"
sudo apt-get update
apt-cache policy docker-ce
sudo apt-get install -y docker-ce
sudo systemctl status docker
```

### Step 3 - Get Docker Image

```
sudo docker pull mivisionx/ubuntu-16.04
```

### Step 4 - Run the docker image

```
sudo docker run -it --device=/dev/kfd --device=/dev/dri --cap-add=SYS_RAWIO --device=/
↳dev/mem --group-add video --network host mivisionx/ ubuntu-16.04
```

- **Optional: Map localhost directory on the docker image**

- option to map the localhost directory with trained caffe models to be accessed on the docker image.
- usage: `-v {LOCAL_HOST_DIRECTORY_PATH}:{DOCKER_DIRECTORY_PATH}`

```
sudo docker run -it -v /home/./root/hostDrive/ --device=/dev/kfd --device=/dev/dri --
↳cap-add=SYS_RAWIO --device=/dev/mem --group-add video --network host mivisionx/
↳ubuntu-16.04
```

### Note: Display option with docker

- Using host display

```
xhost +local:root
sudo docker run -it --device=/dev/kfd --device=/dev/dri --cap-add=SYS_RAWIO --device=/
↳dev/mem --group-add video
--network host --env DISPLAY=unix$DISPLAY --privileged --volume $XAUTH:/root/.
↳Xauthority
--volume /tmp/.X11-unix/./tmp/.X11-unix mivisionx/ubuntu-16.04:latest
```

- Test display with MIVisionX sample



```
export PATH=$PATH:/opt/rocm/mivisionx/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/rocm/mivisionx/lib
runvx /opt/rocm/mivisionx/samples/gdf/canny.gdf
```

**Known issues**

- Package (.deb & .rpm) install requires OpenCV v3.4.0 to execute AMD OpenCV extensions

**Tested configurations**

- Windows 10
- Linux: Ubuntu - 16.04/18.04 & CentOS - 7.5/7.6
- ROCm: rocm-dkms - 2.9.6
- rocm-cmake - [github master:ac45c6e](#)
- MIOpenGEMM - 1.1.5
- MIOpen - 2.1.0
- Protobuf - V3.5.2
- OpenCV - 3.4.0
- Dependencies for all the above packages

## 3.25 AMD ROCm Profiler

### 3.25.1 Overview

The rocProf is a command line tool implemented on the top of rocProfiler and rocTracer APIs. Source code for rocProf can be found at GitHub: <https://github.com/ROCm-Developer-Tools/rocprofiler/blob/amd-master/bin/rocprof>

This command line tool is implemented as a script which is setting up the environment for attaching the profiler and then run the provided application command line. The tool uses two profiling plugins loaded by ROC runtime and based on rocProfiler and rocTracer for collecting metrics/counters, HW traces and runtime API/activity traces. The tool consumes an input XML or text file with counters list or trace parameters and provides output profiling data and statistics in various formats as text, CSV and JSON traces. Google Chrome tracing can be used to visualize the JSON traces with runtime API/activity timelines and per kernel counters data.

### 3.25.2 Profiling Modes

‘rocprof’ can be used for GPU profiling using HW counters and application tracing.

### 3.25.2.1 GPU profiling

GPU profiling is controlled with input file which defines a list of metrics/counters and a profiling scope. An input file is provided using option ‘-i’. Output CSV file with a line per submitted kernel is generated. Each line has kernel name, kernel parameters and counter values. By option ‘—stats’ the kernel execution stats can be generated in CSV format. Currently profiling has limitation of serializing submitted kernels. An example of input file:

```
# Perf counters group 1
pmc : Wavefronts VALUInsts SALUInsts SFetchInsts
# Perf counters group 2
pmc : TCC_HIT[0], TCC_MISS[0]
# Filter by dispatches range, GPU index and kernel names
# supported range formats: "3:9", "3:", "3"
range: 1 : 4
gpu: 0 1 2 3
kernel: simple Pass1 simpleConvolutionPass2
```

An example of profiling command line for ‘MatrixTranspose’ application

```
$ rocprof -i input.txt MatrixTranspose
RPL: on '191018_011134' from '/....rocprofiler_pkg' in '/....MatrixTranspose'
RPL: profiling '"/MatrixTranspose"'
RPL: input file 'input.txt'
RPL: output dir '/tmp/rpl_data_191018_011134_9695'
RPL: result dir '/tmp/rpl_data_191018_011134_9695/input0_results_191018_011134'
ROCProfiler: rc-file '/....rpl_rc.xml'
ROCProfiler: input from "/tmp/rpl_data_191018_011134_9695/input0.xml"
  gpu_index =
  kernel =
  range =
  4 metrics
    L2CacheHit, VFetchInsts, VWriteInsts, MemUnitStalled
  0 traces
Device name Ellesmere [Radeon RX 470/480/570/570X/580/580X]
PASSED!

ROCProfiler: 1 contexts collected, output directory /tmp/rpl_data_191018_011134_9695/
→input0_results_191018_011134
RPL: '/....MatrixTranspose/input.csv' is generated
```

#### 3.25.2.1.1 Counters and metrics

There are two profiling features, metrics and traces. Hardware performance counters are treated as the basic metrics and the formulas can be defined for derived metrics. Counters and metrics can be dynamically configured using XML configuration files with counters and metrics tables:

- Counters table entry, basic metric: counter name, block name, event id
- Derived metrics table entry: metric name, an expression for calculation the metric from the counters

Metrics XML File Example:

```
<gfx8>
  <metric name=L1_CYCLES_COUNTER block=L1 event=0 descr="L1 cache cycles"></metric>
  <metric name=L1_MISS_COUNTER block=L1 event=33 descr="L1 cache misses"></metric>
  . . .
</gfx8>
```

(continues on next page)

(continued from previous page)

```

<gfx9>
    . . .
</gfx9>

<global>
  <metric
    name=L1_MISS_RATIO
    expr=L1_CYCLES_COUNT/L1_MISS_COUNTER
    descry="L1 miss rate metric"
  ></metric>
</global>

```

### 3.25.2.1.1.1 Metrics query

Available counters and metrics can be queried by options ‘—list-basic’ for counters and ‘—list-derived’ for derived metrics. The output for counters indicates number of block instances and number of block counter registers. The output for derived metrics prints the metrics expressions. Examples:

```

$ rocprof --list-basic
RPL: on '191018_014450' from '/opt/rocm/rocprofiler' in '/..../MatrixTranspose'
ROCProfiler: rc-file '/..../rpl_rc.xml'
Basic HW counters:
  gpu-agent0 : GRBM_COUNT : Tie High - Count Number of Clocks
    block GRBM has 2 counters
  gpu-agent0 : GRBM_GUI_ACTIVE : The GUI is Active
    block GRBM has 2 counters
    . . .
  gpu-agent0 : TCC_HIT[0-15] : Number of cache hits.
    block TCC has 4 counters
  gpu-agent0 : TCC_MISS[0-15] : Number of cache misses. UC reads count as misses.
    block TCC has 4 counters
    . . .

$ rocprof --list-derived
RPL: on '191018_015911' from '/opt/rocm/rocprofiler' in '/home/evgeny/work/BUILD/0_
↳MatrixTranspose'
ROCProfiler: rc-file '/home/evgeny/rpl_rc.xml'
Derived metrics:
  gpu-agent0 : TCC_HIT_sum : Number of cache hits. Sum over TCC instances.
    TCC_HIT_sum = sum(TCC_HIT,16)
  gpu-agent0 : TCC_MISS_sum : Number of cache misses. Sum over TCC instances.
    TCC_MISS_sum = sum(TCC_MISS,16)
  gpu-agent0 : TCC_MC_RDREQ_sum : Number of 32-byte reads. Sum over TCC instaces.
    TCC_MC_RDREQ_sum = sum(TCC_MC_RDREQ,16)
    . . .

```

### 3.25.2.1.1.2 Metrics collecting

Counters and metrics accumulated per kernel can be collected using input file with a list of metrics, see an example in 2.1. Currently profiling has limitation of serializing submitted kernels. The number of counters which can be dumped by one run is limited by GPU HW by number of counter registers per block. The number of counters can be different for different blocks and can be queried, see 2.1.1.1.

### 3.25.2.1.1.3 Blocks instancing

GPU blocks are implemented as several identical instances. To dump counters of specific instance square brackets can be used, see an example in 2.1. The number of block instances can be queried, see 2.1.1.1.

### 3.25.2.1.1.4 HW limitations

The number of counters which can be dumped by one run is limited by GPU HW by number of counter registers per block. The number of counters can be different for different blocks and can be queried, see 2.1.1.1.

- Metrics groups

To dump a list of metrics exceeding HW limitations the metrics list can be split on groups. The tool supports automatic splitting on optimal metric groups:

```
$ rocprof -i input.txt ./MatrixTranspose
RPL: on '191018_032645' from '/opt/rocm/rocprofiler' in '/..../'
↳MatrixTranspose'
RPL: profiling './MatrixTranspose'
RPL: input file 'input.txt'
RPL: output dir '/tmp/rpl_data_191018_032645_12106'
RPL: result dir '/tmp/rpl_data_191018_032645_12106/input0_results_
↳191018_032645'
ROCProfiler: rc-file '/....'/rpl_rc.xml'
ROCProfiler: input from "/tmp/rpl_data_191018_032645_12106/input0.xml
↳"
    gpu_index =
    kernel =
    range =
    20 metrics
        Wavefronts, VALUInsts, SALUInsts, SFetchInsts, FlatVMemInsts,
↳LDSInsts, FlatLDSInsts, GDSInsts, VALUUtilization, FetchSize,
↳WriteSize, L2CacheHit, VWriteInsts, GPUBusy, VALUBusy, SALUBusy,
↳MemUnitStalled, WriteUnitStalled, LDSBankConflict, MemUnitBusy
    0 traces
Device name Ellesmere [Radeon RX 470/480/570/570X/580/580X]

Input metrics out of HW limit. Proposed metrics group set:
    group1: L2CacheHit VWriteInsts MemUnitStalled WriteUnitStalled
↳MemUnitBusy FetchSize FlatVMemInsts LDSInsts VALUInsts SALUInsts
↳SFetchInsts FlatLDSInsts GPUBusy Wavefronts
    group2: WriteSize GDSInsts VALUUtilization VALUBusy SALUBusy
↳LDSBankConflict

ERROR: rocprofiler_open(), Construct(), Metrics list exceeds HW
↳limits
```

(continues on next page)

(continued from previous page)

```
Aborted (core dumped)
Error found, profiling aborted.
```

- Collecting with multiple runs

To collect several metric groups a full application replay is used by defining several ‘pmc:’ lines in the input file, see 2.1.

### 3.25.2.2 Application tracing

Supported application tracing includes runtime API and GPU activity tracing. Supported runtimes are: ROCr (HSA API) and HIP. Supported GPU activity: kernel execution, async memory copy, barrier packets. The trace is generated in JSON format compatible with Chrome tracing. The trace consists of several sections with timelines for API trace per thread and GPU activity. The timelines events show event name and parameters. Supported options: ‘—hsa-trace’, ‘—hip-trace’, ‘—sys-trace’, where ‘sys trace’ is for HIP and HSA combined trace.

#### 3.25.2.2.1 HIP runtime trace

The trace is generated by option ‘—hip-trace’ and includes HIP API timelines and GPU activity at the runtime level.

#### 3.25.2.2.2 ROCr runtime trace

The trace is generated by option ‘—hsa-trace’ and includes ROCr API timelines and GPU activity at AQL queue level. Also, can provide counters per kernel.

#### 3.25.2.2.3 KFD driver trace

The trace is generated by option ‘—kfd-trace’ and includes KFD Thunk API timelines.

It is planned to include memory allocations/migration activity tracing.

#### 3.25.2.2.4 Code annotation

Support for application code annotation. Start/stop API is supported to programmatically control the profiling. A ‘roctx’ library provides annotation API. Annotation is visualized in JSON trace as a separate “Markers and Ranges” timeline section.

##### 3.25.2.2.4.1 Start/stop API

```
// Tracing start API
void roctracer_start();

// Tracing stop API
void roctracer_stop();
```

#### 3.25.2.2.4.2 rocTX basic markers API

```
// A marker created by given ASCII message
void roctxMark(const char* message);

// Returns the 0 based level of a nested range being started by given message,
↳ associated to this range.
// A negative value is returned on the error.
int roctxRangePush(const char* message);

// Marks the end of a nested range.
// Returns the 0 based level the range.
// A negative value is returned on the error.
int roctxRangePop();
```

#### 3.25.2.3 Multiple GPUs profiling

The profiler supports multiple GPU's profiling and provide GPI id for counters and kernels data in CSV output file. Also, GPU id is indicating for respective GPU activity timeline in JSON trace.

### 3.25.3 Profiling control

Profiling can be controlled by specifying a profiling scope, by filtering trace events and specifying interesting time intervals.

#### 3.25.3.1 Profiling scope

Counters profiling scope can be specified by GPU id list, kernel name substrings list and dispatch range. Supported range formats examples: "3:9", "3:", "3". You can see an example of input file in 2.1.

#### 3.25.3.2 Tracing control

Tracing can be filtered by events names using profiler input file and by enabling interesting time intervals by command line option.

##### 3.25.3.2.1 Filtering Traced APIs

A list of traced API names can be specified in profiler input file. An example of input file line for ROCr runtime trace (HSA API):

```
hsa:hsa_queue_create hsa_amd_memory_pool_allocate
```

### 3.25.3.2.2 Tracing period

Tracing can be disabled on start so it can be enabled with start/stop API:

```
--trace-start <on|off>
```

Trace can be dumped periodically with initial delay, dumping period length and rate:

```
--trace-period <dealy:length:rate>
```

### 3.25.3.3 Concurrent kernels

Currently concurrent kernels profiling is not supported, which is a planned feature. Kernels are serialized.

### 3.25.3.4 Multi-processes profiling

Multi-processes profiling is not currently supported.

### 3.25.3.5 Errors logging

Profiler errors are logged to global logs:

```
/tmp/aql_profile_log.txt
/tmp/rocprofiler_log.txt
/tmp/roctracer_log.txt
```

## 3.25.4 3rd party visualization tools

‘rocprof’ produces JSON trace, which is compatible with Chrome Tracing. Chrome Tracing is an internal trace visualization tool in Google Chrome.

For more information about Chrome Tracing, see <https://aras-p.info/blog/2017/01/23/Chrome-Tracing-as-Profiler-Frontend/>

## 3.25.5 Runtime Environment Setup

You must set the ‘PATH’ environment variable to the ROCm bin directory. This enables the profiler to find the correct ROCm setup and get ROCm info metadata. For example, “*export PATH=\$PATH:/opt/rocm/bin*”.

### 3.25.6 Command line options

The command line options can be printed with option ‘-h’:

```
rocprof [-h] [--list-basic] [--list-derived] [-i <input .txt/.xml file>] [-o <output_
↪CSV file>] <app command line>
```

Options:

```
-h - this help
--verbose - verbose mode, dumping all base counters used in the input metrics
--list-basic - to print the list of basic HW counters
--list-derived - to print the list of derived metrics with formulas
--cmd-qts <on/off> - quoting profiled cmd line [on]

-i <.txt|.xml file> - input file
    Input file .txt format, automatically rerun application for every pmc line:
```

```
    # Perf counters group 1
    pmc : Wavefronts VALUInsts SALUInsts SFetchInsts FlatVMemInsts LDSInsts_
↪FlatLDSInsts GDSInsts FetchSize
    # Perf counters group 2
    pmc : VALUUtilization,WriteSize L2CacheHit
    # Filter by dispatches range, GPU index and kernel names
    # supported range formats: "3:9", "3:", "3"
    range: 1 : 4
    gpu: 0 1 2 3
    kernel: simple Pass1 simpleConvolutionPass2
```

Input file .xml format, for single profiling run:

```
    # Metrics list definition, also the form "<block-name>:<event-id>" can be used
    # All defined metrics can be found in the 'metrics.xml'
    # There are basic metrics for raw HW counters and high-level metrics for_
↪derived counters
    <metric name=SQ:4,SQ_WAVES,VFetchInsts
    ></metric>

    # Filter by dispatches range, GPU index and kernel names
    <metric
    # range formats: "3:9", "3:", "3"
    range=""
    # list of gpu indexes "0,1,2,3"
    gpu_index=""
    # list of matched sub-strings "Simple1,Conv1,SimpleConvolution"
    kernel=""
    ></metric>
```

```
-o <output file> - output CSV file [<input file base>.csv]
```

The output CSV file columns meaning in the columns order:

```
Index - kernels dispatch order index
KernelName - the dispatched kernel name
gpu-id - GPU id the kernel was submitted to
queue-id - the ROCm queue unique id the kernel was submitted to
queue-index - The ROCm queue write index for the submitted AQL packet
tid - system application thread id which submitted the kernel
grd - the kernel's grid size
wgr - the kernel's work group size
lds - the kernel's LDS memory size
```

(continues on next page)



(continued from previous page)

```

scr - the kernel's scratch memory size
vgpr - the kernel's VGPR size
sgpr - the kernel's SGPR size
fbar - the kernel's barriers limitation
sig - the kernel's completion signal
... - The columns with the counters values per kernel dispatch
DispatchNs/BeginNs/EndNs/CompleteNs - timestamp columns if time-stamping was
↳enabled

-d <data directory> - directory where profiler store profiling data including thread
↳traces [/tmp]
    The data directory is removing automatically if the directory is matching the
↳temporary one, which is the default.
-t <temporary directory> - to change the temporary directory [/tmp]
    By changing the temporary directory you can prevent removing the profiling data
↳from /tmp or enable removing from not '/tmp' directory.

--basenames <on|off> - to turn on/off truncating of the kernel full function names
↳till the base ones [off]
--timestamp <on|off> - to turn on/off the kernel dispatches timestamps, dispatch/
↳begin/end/complete [off]
    Four kernel timestamps in nanoseconds are reported:
        DispatchNs - the time when the kernel AQL dispatch packet was written to the
↳queue
        BeginNs - the kernel execution begin time
        EndNs - the kernel execution end time
        CompleteNs - the time when the completion signal of the AQL dispatch packet was
↳received

--ctx-limit <max number> - maximum number of outstanding contexts [0 - unlimited]
--heartbeat <rate sec> - to print progress heartbeats [0 - disabled]

--stats - generating kernel execution stats, file <output name>.stats.csv
--roctx-trace - to enable roctx application code annotation trace
    Will show the application code annotation in JSON trace "Markers and Ranges"
↳section.
--sys-trace - to trace HIP/HSA APIs and GPU activity, generates stats and JSON trace
↳chrome-tracing compatible
--hip-trace - to trace HIP, generates API execution stats and JSON file chrome-
↳tracing compatible
--hsa-trace - to trace HSA, generates API execution stats and JSON file chrome-
↳tracing compatible
--kfd-trace - to trace KFD, generates API execution stats and JSON file chrome-
↳tracing compatible
    Generated files: <output name>.<domain>_stats.txt <output name>.json
    Traced API list can be set by input .txt or .xml files.
    Input .txt:
        hsa: hsa_queue_create hsa_amd_memory_pool_allocate
    Input .xml:
        <trace name="HSA">
            <parameters list="hsa_queue_create, hsa_amd_memory_pool_allocate">
            </parameters>
        </trace>

--trace-start <on|off> - to enable tracing on start [on]
--trace-period <delay:length:rate> - to enable trace with initial delay, with
↳periodic sample length and rate

```

(continues on next page)

(continued from previous page)

```

Supported time formats: <number(m|s|ms|us)>
--obj-tracking <on|off> - to turn on/off kernels code objects tracking [off]
To support V3 code objects.

Configuration file:
You can set your parameters defaults preferences in the configuration file 'rpl_rc.xml'
→'. The search path sequence: ./home/ evgeny:<package path>
First the configuration file is looking in the current directory, then in your home,
→and then in the package directory.
Configurable options: 'basenames', 'timestamp', 'ctx-limit', 'heartbeat', 'obj-
→tracking'.
An example of 'rpl_rc.xml':
<defaults
  basenames=off
  timestamp=off
  ctx-limit=0
  heartbeat=0
  obj-tracking=off
></defaults>

```

### 3.25.7 Publicly available counters and metrics

The following counters are publicly available for commercially available VEGA10/20 GPUs.

Counters:

- GRBM\_COUNT : Tie High - Count Number of Clocks
- GRBM\_GUI\_ACTIVE : The GUI is Active
- SQ\_WAVES : Count number of waves sent to SQs. (per-simd, emulated, global)
- SQ\_INSTS\_VALU : Number of VALU instructions issued. (per-simd, emulated)
- SQ\_INSTS\_VMEM\_WR : Number of VMEM write instructions issued (including FLAT).
 →(per-simd, emulated)
- SQ\_INSTS\_VMEM\_RD : Number of VMEM read instructions issued (including FLAT). (per-
 →simd, emulated)
- SQ\_INSTS\_SALU : Number of SALU instructions issued. (per-simd, emulated)
- SQ\_INSTS\_SMEM : Number of SMEM instructions issued. (per-simd, emulated)
- SQ\_INSTS\_FLAT : Number of FLAT instructions issued. (per-simd, emulated)
- SQ\_INSTS\_FLAT\_LDS\_ONLY : Number of FLAT instructions issued that read/wrote only
 →from/to LDS (only works if EARLY\_TA\_DONE is enabled). (per-simd, emulated)
- SQ\_INSTS\_LDS : Number of LDS instructions issued (including FLAT). (per-simd,
 →emulated)
- SQ\_INSTS\_GDS : Number of GDS instructions issued. (per-simd, emulated)
- SQ\_WAIT\_INST\_LDS : Number of wave-cycles spent waiting for LDS instruction issue.
 →In units of 4 cycles. (per-simd, nondeterministic)
- SQ\_ACTIVE\_INST\_VALU : regspec 71? Number of cycles the SQ instruction arbiter is
 →working on a VALU instruction. (per-simd, nondeterministic)
- SQ\_INST\_CYCLES\_SALU : Number of cycles needed to execute non-memory read scalar
 →operations. (per-simd, emulated)
- SQ\_THREAD\_CYCLES\_VALU : Number of thread-cycles used to execute VALU operations
 →(similar to INST\_CYCLES\_VALU but multiplied by # of active threads). (per-simd)
- SQ\_LDS\_BANK\_CONFLICT : Number of cycles LDS is stalled by bank conflicts.
 →(emulated)
- TA\_TA\_BUSY[0-15] : TA block is busy. Perf\_Windowing not supported for this
 →counter.
- TA\_FLAT\_READ\_WAVEFRONTS[0-15] : Number of flat opcode reads processed by the TA.

(continues on next page)

(continued from previous page)

- `TA_FLAT_WRITE_WAVEFRONTS[0-15]` : Number of flat opcode writes processed by the TA.
- `TCC_HIT[0-15]` : Number of cache hits.
- `TCC_MISS[0-15]` : Number of cache misses. UC reads count as misses.
- `TCC_EA_WRREQ[0-15]` : Number of transactions (either 32-byte or 64-byte) going over the `TC_EA_wrreq` interface. Atomics may travel over the same interface and are generally classified as write requests. This does not include probe commands.
- `TCC_EA_WRREQ_64B[0-15]` : Number of 64-byte transactions going (64-byte write or CMPSWAP) over the `TC_EA_wrreq` interface.
- `TCC_EA_WRREQ_STALL[0-15]` : Number of cycles a write request was stalled.
- `TCC_EA_RDREQ[0-15]` : Number of TCC/EA read requests (either 32-byte or 64-byte)
- `TCC_EA_RDREQ_32B[0-15]` : Number of 32-byte TCC/EA read requests
- `TCP_TCP_TA_DATA_STALL_CYCLES[0-15]` : TCP stalls TA data interface. Now Windowed.

The following derived metrics have been defined and the profiler metrics XML specification can be found at: <https://github.com/ROCm-Developer-Tools/rocprofiler/blob/amd-master/test/tool/metrics.xml>.

#### Metrics:

- `TA_BUSY_avr` : TA block is busy. Average over TA instances.
- `TA_BUSY_max` : TA block is busy. Max over TA instances.
- `TA_BUSY_min` : TA block is busy. Min over TA instances.
- `TA_FLAT_READ_WAVEFRONTS_sum` : Number of flat opcode reads processed by the TA. Sum over TA instances.
- `TA_FLAT_WRITE_WAVEFRONTS_sum` : Number of flat opcode writes processed by the TA. Sum over TA instances.
- `TCC_HIT_sum` : Number of cache hits. Sum over TCC instances.
- `TCC_MISS_sum` : Number of cache misses. Sum over TCC instances.
- `TCC_EA_RDREQ_32B_sum` : Number of 32-byte TCC/EA read requests. Sum over TCC instances.
- `TCC_EA_RDREQ_sum` : Number of TCC/EA read requests (either 32-byte or 64-byte). Sum over TCC instances.
- `TCC_EA_WRREQ_sum` : Number of transactions (either 32-byte or 64-byte) going over the `TC_EA_wrreq` interface. Sum over TCC instances.
- `TCC_EA_WRREQ_64B_sum` : Number of 64-byte transactions going (64-byte write or CMPSWAP) over the `TC_EA_wrreq` interface. Sum over TCC instances.
- `TCC_WRREQ_STALL_max` : Number of cycles a write request was stalled. Max over TCC instances.
- `TCC_MC_WRREQ_sum` : Number of 32-byte effective writes. Sum over TCC instances.
- `FETCH_SIZE` : The total kilobytes fetched from the video memory. This is measured with all extra fetches and any cache or memory effects taken into account.
- `WRITE_SIZE` : The total kilobytes written to the video memory. This is measured with all extra fetches and any cache or memory effects taken into account.
- `GPUBusy` : The percentage of time GPU was busy.
- `Wavefronts` : Total wavefronts.
- `VALUInsts` : The average number of vector ALU instructions executed per work-item (affected by flow control).
- `SALUInsts` : The average number of scalar ALU instructions executed per work-item (affected by flow control).
- `VFetchInsts` : The average number of vector fetch instructions from the video memory executed per work-item (affected by flow control). Excludes FLAT instructions that fetch from video memory.
- `SFetchInsts` : The average number of scalar fetch instructions from the video memory executed per work-item (affected by flow control).
- `VWriteInsts` : The average number of vector write instructions to the video memory executed per work-item (affected by flow control). Excludes FLAT instructions that write to video memory.
- `FlatVMemInsts` : The average number of FLAT instructions that read from or write to the video memory executed per work item (affected by flow control). Excludes FLAT instructions that read from or write to scratch.

(continued from previous page)

- LDSInsts : The average number of LDS read or LDS write instructions executed per work item (affected by flow control). Excludes FLAT instructions that read from or write to LDS.
- FlatLDSInsts : The average number of FLAT instructions that read or write to LDS executed per work item (affected by flow control).
- GDSInsts : The average number of GDS read or GDS write instructions executed per work item (affected by flow control).
- VALUUtilization : The percentage of active vector ALU threads in a wave. A lower number can mean either more thread divergence in a wave or that the work-group size is not a multiple of 64. Value range: 0% (bad), 100% (ideal - no thread divergence).
- VALUBusy : The percentage of GPUTime vector ALU instructions are processed. Value range: 0% (bad) to 100% (optimal).
- SALUBusy : The percentage of GPUTime scalar ALU instructions are processed. Value range: 0% (bad) to 100% (optimal).
- Mem32Bwrites :
- FetchSize : The total kilobytes fetched from the video memory. This is measured with all extra fetches and any cache or memory effects taken into account.
- WriteSize : The total kilobytes written to the video memory. This is measured with all extra fetches and any cache or memory effects taken into account.
- L2CacheHit : The percentage of fetch, write, atomic, and other instructions that hit the data in L2 cache. Value range: 0% (no hit) to 100% (optimal).
- MemUnitBusy : The percentage of GPUTime the memory unit is active. The result includes the stall time (MemUnitStalled). This is measured with all extra fetches and writes and any cache or memory effects taken into account. Value range: 0% to 100% (fetch-bound).
- MemUnitStalled : The percentage of GPUTime the memory unit is stalled. Try reducing the number or size of fetches and writes if possible. Value range: 0% (optimal) to 100% (bad).
- WriteUnitStalled : The percentage of GPUTime the Write unit is stalled. Value range: 0% to 100% (bad).
- ALUStalledByLDS : The percentage of GPUTime ALU units are stalled by the LDS input queue being full or the output queue being not ready. If there are LDS bank conflicts, reduce them. Otherwise, try reducing the number of LDS accesses if possible. Value range: 0% (optimal) to 100% (bad).
- LDSBankConflict : The percentage of GPUTime LDS is stalled by bank conflicts. Value range: 0% (optimal) to 100% (bad).

## 3.26 AMD ROCProfiler API

ROC profiler library. Profiling with perf-counters and derived metrics. Library supports GFX8/GFX9.

HW specific low-level performance analysis interface for profiling of GPU compute applications. The profiling includes HW performance counters with complex performance metrics.

GitHub: <https://github.com/ROCm-Developer-Tools/rocprofiler>

### Metrics

- [The link to profiler default metrics XML specification.](#)

### API specification

- [API specification at the GitHub.](#)

### To get sources

To clone ROC Profiler from GitHub:

```
git clone https://github.com/ROCm-Developer-Tools/rocprofiler
```

The library `source` tree:

```
* bin
  * rocprof - Profiling tool run script
* doc - Documentation
* inc/rocprofiler.h - Library public API
* src - Library sources
  * core - Library API sources
  * util - Library utils sources
  * xml - XML parser
* test - Library test suite
  * tool - Profiling tool
    * tool.cpp - tool sources
    * metrics.xml - metrics config file
  * ctrl - Test controll
  * util - Test utils
  * simple_convolution - Simple convolution test kernel
```

## Build

Build environment:

```
export CMAKE_PREFIX_PATH=<path to hsa-runtime includes>:<path to hsa-runtime library>
export CMAKE_BUILD_TYPE=<debug|release> # release by default
export CMAKE_DEBUG_TRACE=1 # to enable debug tracing
```

**To Build with the current installed ROCm:**

```
To build and install to /opt/rocm/rocprofiler
export CMAKE_PREFIX_PATH=/opt/rocm/include/hsa:/opt/rocm
cd ../rocprofiler
mkdir build
cd build
cmake ..
make
make install
```

**Internal ‘simple\_convolution’ test run script:**

```
cd ../rocprofiler/build
./run.sh
```

**To enable error messages logging to ‘/tmp/rocprofiler\_log.txt’:**

```
export ROCPROFILER_LOG=1
```

**To enable verbose tracing:**

```
export ROCPROFILER_TRACE=1
```

## 3.27 AMD ROCTracer API

ROCTracer library, Runtimes Generic Callback/Activity APIs. The goal of the implementation is to provide a generic independent from specific runtime profiler to trace API and asynchronous activity.

The API provides functionality for registering the runtimes API callbacks and asynchronous activity records pool support.

GitHub: <https://github.com/ROCm-Developer-Tools/roctracer>

### API specification

- [API specification at the GitHub.](#)

### To get sources

To clone ROC Tracer from GitHub:

```
git clone -b amd-master https://github.com/ROCm-Developer-Tools/roctracer

The library source tree:

* inc/roctracer.h - Library public API
* src - Library sources
  * core - Library API sources
  * util - Library utils sources
* test - test suit
  * MatrixTranspose - test based on HIP MatrixTranspose sample
```

### Build and run test

```
- Python is required
  The required modules: CppHeaderParser, argparse.
  To install:
  sudo pip install CppHeaderParser argparse

- To customize environment, below are defaults
  export HIP_PATH=/opt/rocm/HIP
  export HCC_HOME=/opt/rocm/hcc/
  export CMAKE_PREFIX_PATH=/opt/rocm

- Build ROCTracer
  export CMAKE_BUILD_TYPE=<debug|release> # release by default
  cd <your path>/roctracer && mkdir build && cd build && cmake -DCMAKE_INSTALL_PREFIX=/
  ↪opt/rocm .. && make -j <nproc>

- To build and run test
  make mytest
  run.sh

- To install
  make install
  or
  make package && dpkg -i *.deb
```

## 3.28 AMD ROCm Debugger

The AMD ROCm Debugger (ROCgdb) is the AMD ROCm source-level debugger for Linux based on the GNU Debugger (GDB). It enables heterogeneous debugging on the AMD ROCm platform of an x86-based host architecture along with AMD GPU architectures and supported by the *AMD Debugger API*.

The AMD ROCm Debugger is installed by the rocm-gdb package. The rocm-gdb package is part of the rocm-dev meta-package, which is in the rocm-dkms package.

The current AMD ROCm Debugger (ROCgdb) is an initial prototype that focuses on source line debugging. Note, symbolic variable debugging capabilities are not currently supported.

You can use the standard GDB commands for both CPU and GPU code debugging. For more information about ROCgdb, refer to the ROCgdb User Guide, which is installed at:

- /opt/rocm/share/info/gdb.info as a texinfo file
- /opt/rocm/share/doc/gdb/gdb.pdf as a PDF file

The AMD ROCm Debugger User Guide is available as a PDF at:

[https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger\\_User\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger_User_Guide.pdf)

For more information about GNU Debugger (GDB), refer to the GNU Debugger (GDB) web site at: <http://www.gnu.org/software/gdb>

## 3.29 AMD Debugger API

### 3.29.1 Introduction

The AMD Debugger API (ROCdbgapi) is a library that provides all the support necessary for a debugger and other tools to perform low level control of the execution and inspection of execution state of AMD™ commercially available GPU architectures.

For the AMD Debugger API Guide, see

For more information about the AMD ROCm ecosystem, see:

- <https://rocmdocs.amd.com/>

[https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger\\_API\\_Guide.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCDebugger_API_Guide.pdf)

### 3.29.2 Build the AMD Debugger API Library

The ROCdbgapi library can be built on Ubuntu 16.04, Ubuntu 18.04, Centos 8.1, RHEL 8.1, and SLES 15 Service Pack 1.

Building the ROCdbgapi library has the following prerequisites:

1. A C++14 compiler such as GCC 5 or Clang 3.4.
2. AMD Code Object Manager Library (ROCcomgr) which can be installed as part of the AMD ROCm release by the comgr package.
3. ROCm CMake modules which can be installed as part of the AMD ROCm release by the rocm-cmake package.

An example command-line to build and install the ROCdbgapi library on Linux is:

```
cd rocdbgapi
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=../install ..
make
```

You may substitute a path of your own choosing for CMAKE\_INSTALL\_PREFIX.

The built ROCdbgapi library will be placed in:

- build/include/amd-dbgapi.h
- build/librocm-dbgapi.so\*

To install the ROCdbgapi library:

```
make install
```

The installed ROCdbgapi library will be placed in:

- ../install/include/amd-dbgapi.h
- ../install/lib/librocm-dbgapi.so\*
- ../install/share/amd-dbgapi/LICENSE.txt
- ../install/share/amd-dbgapi/README.md

To use the ROCdbgapi library, the ROCcomgr library must be installed. This can be installed as part of the AMD ROCm release by the comgr package:

- libamd\_comgr.so.1

### 3.29.3 Build the AMD Debugger API Specification Documentation

Generating the *AMD Debugger API Specification* documentation has the following prerequisites:

1. For Ubuntu 16.04 and Ubuntu 18.04 the following adds the needed packages:

```
apt install doxygen graphviz texlive-full
```

NOTE: The doxygen 1.8.13 that is installed by Ubuntu 18.04 has a bug that prevents the PDF from being created. doxygen 1.8.11 can be built from source to avoid the issue.

2. For CentOS 8.1 and RHEL 8.1 the following adds the needed packages:

```
yum install -y doxygen graphviz texlive texlive-xtab texlive-multirow \
texlive-sectsty texlive-tocloft texlive-tabu texlive-adjustbox
```

NOTE: The doxygen 1.8.14 that is installed by CentOS 8.1 and RHEL 8.1, has a bug that prevents the PDF from being created. doxygen 1.8.11 can be built from source to avoid the issue.

3. For SLES 15 Service Pack 15 the following adds the needed packages:

```
zypper in doxygen graphviz texlive-scheme-medium texlive-hanging \
texlive-stackengine texlive-tocloft texlive-etoc texlive-tabu
```

An example command-line to generate the HTML and PDF documentation after running the above cmake is:

```
make doc
```



The generated ROCdbgapi library documentation is put in:

- `doc/html/index.html`
- `doc/latex/refman.pdf`

If the ROCdbgapi library PDF documentation has been generated, `make install` will place it in:

- `../install/share/doc/amd-dbgapi/amd-dbgapi.pdf`

### 3.29.4 Known Limitations and Restrictions

You can refer to the following sections in the *AMD Debugger API Specification* documentation for:

- *Supported AMD GPU Architectures* provides the list of supported AMD GPU architectures.
- *Known Limitations and Restrictions* provides information about known limitations and restrictions.

The ROCdbgapi library is compatible with the following interface versions:

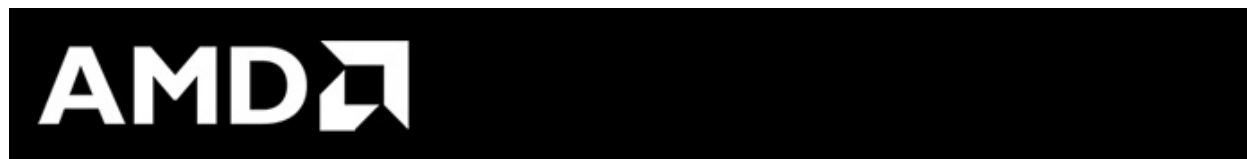
- *AMD GPU Driver Version*
  - See `KFD_IOCTL_MAJOR_VERSION` and `KFD_IOCTL_MINOR_VERSION` in `src/linux/kfd_ioctl.h` which conform to [semver](#).
- *AMD GPU Driver Debug ioctl Version*
  - See `KFD_IOCTL_DBG_MAJOR_VERSION` and `KFD_IOCTL_DBG_MINOR_VERSION` in `src/linux/kfd_ioctl.h` which conform to [semver](#).
- *ROCm Runtime `r_debug` ABI Version*
  - See `ROCR_RDEBUG_VERSION` in `src/rocr_rdebug.h`.
- *Architectures and Firmware Versions*
  - See `s_gfxip_lookup_table` in `src/os_driver.cpp`.

### 3.29.5 Disclaimer

The information contained herein is for informational purposes only and is subject to change without notice. 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 products are as set forth in a signed agreement between the parties or in AMD™ Standard Terms and Conditions of Sale.

AMD, the AMD Arrow logo, ROCm® and combinations thereof are trademarks of Advanced Micro Devices, Inc. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. PCIe® is a registered trademark of PCI-SIG Corporation. RedHat® and the Shadowman logo are registered trademarks of Red Hat, Inc. [www.redhat.com](http://www.redhat.com) in the U.S. and other countries. SUSE® is a registered trademark of SUSE LLC in the United States and other countries. Ubuntu® and the Ubuntu logo are registered trademarks of Canonical Ltd. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Copyright (c) 2019-2021 Advanced Micro Devices, Inc. All rights reserved.



## 3.30 ROCm™ Data Center Tool

The ROCm™ Data Center Tool™ simplifies the administration and addresses key infrastructure challenges in AMD GPUs in cluster and datacenter environments. The main features are:

- GPU telemetry
- GPU statistics for jobs
- Integration with third-party tools
- Open source

The tool can be used in stand-alone mode if all components are installed. However, the existing management tools can use the same set of features available in a library format.

Refer to the Starting RDC section in the ROCm Data Center Tool User Guide for details on different modes of operation.

### 3.30.1 Objective

This user guide is intended to:

- Provide an overview of the ROCm Data Center Tool features
- Describe how system administrators and Data Center (or HPC) users can administer and configure AMD GPUs
- Describe the components
- Provide an overview of the open source developer handbook

### 3.30.2 Target Audience

The audience for the AMD ROCm Data Center™ tool consists of:

- Administrators: The tool will provide cluster administrator with the capability of monitoring, validating, and configuring policies.
- HPC Users: Provides GPU centric feedback for their workload submissions
- OEM: Add GPU information to their existing cluster management software
- Open Source Contributors: RDC is open source and will accept contributions from the community

### 3.30.3 Download AMD ROCm Data Center Tool User Guide

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_ROCm\\_DataCenter\\_Tool\\_User\\_Guide\\_v4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_ROCm_DataCenter_Tool_User_Guide_v4.5.pdf)

### 3.30.4 Download AMD ROCm Data Center Tool API Guide

[https://github.com/RadeonOpenCompute/ROCm/blob/master/RDC\\_API\\_Manual\\_4.5.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/RDC_API_Manual_4.5.pdf)

## 3.31 AMD ROCm Debug Agent Library

### 3.31.1 Introduction

The AMD ROCm Debug Agent (ROCdebug-agent) is a library that can be loaded by the ROCm Platform Runtime (ROCr) to provide the following functionality:

- Print the state of all AMD GPU wavefronts that caused a queue error (for example, causing a memory violation, executing an `s_trap 2`, or executing an illegal instruction).
- Print the state of all AMD GPU wavefronts by sending a SIGQUIT signal to the process (for example, by pressing `Ctrl-\`) while the program is executing.

This functionality is provided for all AMD GPUs supported by the ROCm Debugger API Library (ROCdbgapi).

### 3.31.2 Usage

To display the source text location with the machine code instructions around the wavefronts' pc, compile the AMD GPU code objects with `-ggdb`. In addition, `-O0`, while not required, will help the source text location displayed to be more intuitive as higher optimization levels can reorder machine code instructions. If `-ggdb` is not used, source line information will not be available and only machine code instructions starting at the wavefronts' pc will be printed. For example:

```
/opt/rocm/bin/hipcc -O0 -ggdb -o my_program my_program.cpp
```

To use the ROCdebug-agent set the `HSA_TOOLS_LIB` environment variable to the file name or path of the library. For example:

```
HSA_TOOLS_LIB=/opt/rocm/lib/librocm-debug-agent.so.2 ./my_program
```

If the application encounters a triggering event, it will print the state of some or all AMD GPU wavefronts. For example, a sample print out is:

```
Queue error (HSA_STATUS_ERROR_EXCEPTION: An HSAIL operation resulted in a hardware_
↳exception.)

-----
wave_1: pc=0x7fd4f100d0e8 (stopped, reason: ASSERT_TRAP)

system registers:
      m0: 00000000      status: 00012461      trapsts: 20000000      ↳
↳mode: 000003c0
      ttmp4: 00000001      ttmp5: 00000000      ttmp6: f51a0080      ↳
↳ttmp7: 000000d5
      ttmp8: 00000000      ttmp9: 00000000      ttmp10: 00000000      ↳
↳ttmp11: 000000c0
```

(continues on next page)

(continued from previous page)

```

        tttmp13: 00000000
        exec: 0000000000000001          vcc: 0000000000000000
        xnack_mask: 0000000000012460   flat_scratch: 00807fac01000000

scalar registers:
        s0: f520c000          s1: 00007fd5          s2: 00000000
↪s3: 00ea4fac
        s4: f51a0080          s5: 00007fd5          s6: f520c000
↪s7: 00007fd5
        s8: f1002000          s9: 00007fd4          s10: 00000000
↪s11: 00000000
        s12: f1000000          s13: 00007fd4          s14: f1001000
↪s15: 00007fd4
        s16: f5186070          s17: 00007fd5          s18: f100e070
↪s19: 00007fd4
        s20: f5186070          s21: 00007fd5          s22: f100e070
↪s23: 00007fd4
        s24: 00004000          s25: 00010000

vector registers:
        v0: [0] 00000000 [1] f1002004 [2] f1002008 [3] f100200c [4] f1002010 [5]
↪f1002014 [6] f1002018 [7] f100201c [8] f1002020 [9] f1002024 [10] f1002028 [11]
↪f100202c [12] f1002030 [13] f1002034 [14] f1002038 [15] f100203c [16] f1002040 [17]
↪f1002044 [18] f1002048 [19] f100204c [20] f1002050 [21] f1002054 [22] f1002058 [23]
↪f100205c [24] f1002060 [25] f1002064 [26] f1002068 [27] f100206c [28] f1002070 [29]
↪f1002074 [30] f1002078 [31] f100207c [32] f1002080 [33] f1002084 [34] f1002088 [35]
↪f100208c [36] f1002090 [37] f1002094 [38] f1002098 [39] f100209c [40] f10020a0 [41]
↪f10020a4 [42] f10020a8 [43] f10020ac [44] f10020b0 [45] f10020b4 [46] f10020b8 [47]
↪f10020bc [48] f10020c0 [49] f10020c4 [50] f10020c8 [51] f10020cc [52] f10020d0 [53]
↪f10020d4 [54] f10020d8 [55] f10020dc [56] f10020e0 [57] f10020e4 [58] f10020e8 [59]
↪f10020ec [60] f10020f0 [61] f10020f4 [62] f10020f8 [63] f10020fc
        v1: [0] 00000000 [1] 00007fd4 [2] 00007fd4 [3] 00007fd4 [4] 00007fd4 [5]
↪00007fd4 [6] 00007fd4 [7] 00007fd4 [8] 00007fd4 [9] 00007fd4 [10] 00007fd4 [11]
↪00007fd4 [12] 00007fd4 [13] 00007fd4 [14] 00007fd4 [15] 00007fd4 [16] 00007fd4 [17]
↪00007fd4 [18] 00007fd4 [19] 00007fd4 [20] 00007fd4 [21] 00007fd4 [22] 00007fd4 [23]
↪00007fd4 [24] 00007fd4 [25] 00007fd4 [26] 00007fd4 [27] 00007fd4 [28] 00007fd4 [29]
↪00007fd4 [30] 00007fd4 [31] 00007fd4 [32] 00007fd4 [33] 00007fd4 [34] 00007fd4 [35]
↪00007fd4 [36] 00007fd4 [37] 00007fd4 [38] 00007fd4 [39] 00007fd4 [40] 00007fd4 [41]
↪00007fd4 [42] 00007fd4 [43] 00007fd4 [44] 00007fd4 [45] 00007fd4 [46] 00007fd4 [47]
↪00007fd4 [48] 00007fd4 [49] 00007fd4 [50] 00007fd4 [51] 00007fd4 [52] 00007fd4 [53]
↪00007fd4 [54] 00007fd4 [55] 00007fd4 [56] 00007fd4 [57] 00007fd4 [58] 00007fd4 [59]
↪00007fd4 [60] 00007fd4 [61] 00007fd4 [62] 00007fd4 [63] 00007fd4
        v2: [0] 22222222 [1] 11111125 [2] 1111111b [3] 11111123 [4] 1111111d [5]
↪1111111c [6] 1111111a [7] 1111111d [8] 1111111a [9] 1111111b [10] 1111111c [11]
↪11111118 [12] 11111123 [13] 1111111c [14] 11111119 [15] 11111117 [16] 1111111d [17]
↪11111114 [18] 1111111b [19] 11111117 [20] 1111111a [21] 1111111d [22] 11111118 [23]
↪11111120 [24] 11111118 [25] 1111111c [26] 1111111d [27] 1111111e [28] 1111111a [29]
↪11111122 [30] 1111111e [31] 11111120 [32] 11111123 [33] 11111119 [34] 1111111c [35]
↪1111111d [36] 11111116 [37] 1111111a [38] 1111111d [39] 1111111c [40] 11111113 [41]
↪11111115 [42] 1111111d [43] 1111111f [44] 1111111e [45] 1111111c [46] 1111111f [47]
↪1111111e [48] 11111117 [49] 11111115 [50] 1111111a [51] 11111121 [52] 1111111f [53]
↪1111111b [54] 1111111b [55] 11111124 [56] 11111116 [57] 11111125 [58] 11111123 [59]
↪1111111b [60] 1111111a [61] 11111119 [62] 11111118 [63] 11111123
        v3: [0] 11111111 [1] 11111111 [2] 11111111 [3] 11111111 [4] 11111111 [5]
↪11111111 [6] 11111111 [7] 11111111 [8] 11111111 [9] 11111111 [10] 11111111 [11]
↪11111111 [12] 11111111 [13] 11111111 [14] 11111111 [15] 11111111 [16] 11111111 [17]
↪11111111 [18] 11111111 [19] 11111111 [20] 11111111 [21] 11111111 [22] 11111111 [23]
↪11111111 [24] 11111111 [25] 11111111 [26] 11111111 [27] 11111111 [28]
↪11111111 [30] 11111111 [31] 11111111 [32] 11111111 [33] 11111111 [34] 11111111 [35]
↪11111111 [36] 11111111 [37] 11111111 [38] 11111111 [39] 11111111 [40] 11111111 [41]
↪11111111 [42] 11111111 [43] 11111111 [44] 11111111 [45] 11111111 [46] 11111111 [47]
↪11111111 [48] 11111111 [49] 11111111 [50] 11111111 [51] 11111111 [52] 11111111 [53]
↪11111111 [54] 11111111 [55] 11111111 [56] 11111111 [57] 11111111 [58] 11111111 [59]
↪11111111 [60] 11111111 [61] 11111111 [62] 11111111 [63] 11111111

```

(continued from previous page)

```

v4: [0] f10115b0 [1] 0000000a [2] 00000005 [3] 00000009 [4] 00000004 [5]
→00000001 [6] 00000001 [7] 0000000a [8] 00000004 [9] 00000005 [10] 00000008 [11]
→00000002 [12] 00000008 [13] 00000001 [14] 00000006 [15] 00000005 [16] 00000005 [17]
→00000001 [18] 00000001 [19] 00000002 [20] 00000006 [21] 00000006 [22] 00000002 [23]
→0000000a [24] 00000001 [25] 00000001 [26] 0000000a [27] 00000006 [28] 00000001 [29]
→00000008 [30] 0000000a [31] 00000009 [32] 00000009 [33] 00000007 [34] 0000000a [35]
→00000007 [36] 00000003 [37] 00000003 [38] 00000008 [39] 00000001 [40] 00000001 [41]
→00000002 [42] 00000005 [43] 00000009 [44] 00000005 [45] 00000005 [46] 0000000a [47]
→00000003 [48] 00000004 [49] 00000001 [50] 00000002 [51] 0000000a [52] 0000000a [53]
→00000001 [54] 00000007 [55] 0000000a [56] 00000004 [57] 0000000a [58] 00000008 [59]
→00000006 [60] 00000008 [61] 00000001 [62] 00000004 [63] 00000009

v5: [0] 00007fd4 [1] 00007fd4 [2] 00007fd4 [3] 00007fd4 [4] 00007fd4 [5]
→00007fd4 [6] 00007fd4 [7] 00007fd4 [8] 00007fd4 [9] 00007fd4 [10] 00007fd4 [11]
→00007fd4 [12] 00007fd4 [13] 00007fd4 [14] 00007fd4 [15] 00007fd4 [16] 00007fd4 [17]
→00007fd4 [18] 00007fd4 [19] 00007fd4 [20] 00007fd4 [21] 00007fd4 [22] 00007fd4 [23]
→00007fd4 [24] 00007fd4 [25] 00007fd4 [26] 00007fd4 [27] 00007fd4 [28] 00007fd4 [29]
→00007fd4 [30] 00007fd4 [31] 00007fd4 [32] 00007fd4 [33] 00007fd4 [34] 00007fd4 [35]
→00007fd4 [36] 00007fd4 [37] 00007fd4 [38] 00007fd4 [39] 00007fd4 [40] 00007fd4 [41]
→00007fd4 [42] 00007fd4 [43] 00007fd4 [44] 00007fd4 [45] 00007fd4 [46] 00007fd4 [47]
→00007fd4 [48] 00007fd4 [49] 00007fd4 [50] 00007fd4 [51] 00007fd4 [52] 00007fd4 [53]
→00007fd4 [54] 00007fd4 [55] 00007fd4 [56] 00007fd4 [57] 00007fd4 [58] 00007fd4 [59]
→00007fd4 [60] 00007fd4 [61] 00007fd4 [62] 00007fd4 [63] 00007fd4

v6: [0] 00007ffe [1] 00007ffe [2] 00007ffe [3] 00007ffe [4] 00007ffe [5]
→00007ffe [6] 00007ffe [7] 00007ffe [8] 00007ffe [9] 00007ffe [10] 00007ffe [11]
→00007ffe [12] 00007ffe [13] 00007ffe [14] 00007ffe [15] 00007ffe [16] 00007ffe [17]
→00007ffe [18] 00007ffe [19] 00007ffe [20] 00007ffe [21] 00007ffe [22] 00007ffe [23]
→00007ffe [24] 00007ffe [25] 00007ffe [26] 00007ffe [27] 00007ffe [28] 00007ffe [29]
→00007ffe [30] 00007ffe [31] 00007ffe [32] 00007ffe [33] 00007ffe [34] 00007ffe [35]
→00007ffe [36] 00007ffe [37] 00007ffe [38] 00007ffe [39] 00007ffe [40] 00007ffe [41]
→00007ffe [42] 00007ffe [43] 00007ffe [44] 00007ffe [45] 00007ffe [46] 00007ffe [47]
→00007ffe [48] 00007ffe [49] 00007ffe [50] 00007ffe [51] 00007ffe [52] 00007ffe [53]
→00007ffe [54] 00007ffe [55] 00007ffe [56] 00007ffe [57] 00007ffe [58] 00007ffe [59]
→00007ffe [60] 00007ffe [61] 00007ffe [62] 00007ffe [63] 00007ffe

v7: [0] 3d3495ac [1] bd0dfb7a [2] bcc1143a [3] bca64d59 [4] bc112d79 [5]
→3cbcc8c8 [6] 3ce69f7c [7] 3de967fe [8] bdee8d4d [9] 3c9e426b [10] bc6d380f [11]
→3c18495c [12] be38843f [13] bd5a1da8 [14] 3d80c7e4 [15] bc978798 [16] 3cd52d8d [17]
→bd58d230 [18] 3e2e91ac [19] bca54a71 [20] 3c3cea13 [21] 3c888a4b [22] 3de0a868 [23]
→3d220de3 [24] 3ce4d6f8 [25] bc033ce0 [26] bb38519f [27] b9a4b621 [28] bd800802 [29]
→bdb04d27 [30] bc826d02 [31] bd4aa05d [32] 3dae9483 [33] b921dac8 [34] 3d194f79 [35]
→bdlccbd9 [36] bd45f9c5 [37] bc1b4cb0 [38] 3db1ab4b [39] 3e0487ab [40] 3d37f334 [41]
→3b983eb8 [42] 3caba2a4 [43] bd8944ea [44] be01bee7 [45] bbbf22d8 [46] 3d076472 [47]
→bd2eb34c [48] 3c3da426 [49] 3d754b6d [50] 3c08a069 [51] bcdeca32 [52] be12e2e4 [53]
→3c92d0e2 [54] 3d1480e4 [55] 3d817751 [56] 3db0072c [57] 3d6fc70b [58] bd6a67a1 [59]
→3da0f9ed [60] 3b67b5e6 [61] bdb8002e [62] 3cd0a9b9 [63] 386eee2b

```

Local memory content:

```

0x0000: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0020: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0040: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0060: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0080: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x00a0: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x00c0: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x00e0: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0100: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0120: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0140: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111

```

(continues on next page)

(continued from previous page)

```

0x0160: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x0180: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x01a0: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x01c0: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111
0x01e0: 22222222 11111111 22222222 11111111 22222222 11111111 22222222 11111111

Disassembly for function vector_add_assert_trap(int*, int*, int*):
  code object: file:///rocm-debug-agent/build/test/rocm-debug-agent-test
  ↪ #offset=14309&size=31336
    loaded at: [0x7fd4f100c000-0x7fd4f100e070]

/rocm-debug-agent/test/vector_add_assert_trap.cpp:
55      c[gid] = a[gid] + b[gid] + (lds_check[0] >> 32);
      0x7fd4f100d0c4 <+196>:      s_waitcnt vmcnt(0) lgkmcnt(0)
      0x7fd4f100d0c8 <+200>:      v_add3_u32 v2, v2, v4, v3
      0x7fd4f100d0d0 <+208>:      global_store_dword v[0:1], v2, off
      0x7fd4f100d0d8 <+216>:      s_or_saveexec_b64 s[0:1], s[0:1]
      0x7fd4f100d0dc <+220>:      s_xor_b64 exec, exec, s[0:1]
      0x7fd4f100d0e0 <+224>:      s_cbranch_execz 65503 # 0x7fd4f100d060 <vector_add_
  ↪ assert_trap(int*, int*, int*)+96>

53      __builtin_trap ();
      0x7fd4f100d0e4 <+228>:      s_mov_b64 s[0:1], s[6:7]
=> 0x7fd4f100d0e8 <+232>:      s_trap 2
      0x7fd4f100d0ec <+236>:      s_endpgm

End of disassembly.
Aborted (core dumped)

```

The supported triggering events are:

- **Memory fault**

A memory fault happens when an AMD GPU accesses a page that is not accessible. The information about the memory fault is printed. For example:

```

System event (HSA_AMD_GPU_MEMORY_FAULT_EVENT: page not present or supervisor_
  ↪ privilege, write access to a read-only page)
Faulting page: 0x7fbe4cc01000

```

There could be multiple memory faults, but the information about only one is printed.

A memory fault does not specify the wavefront that caused it. However, the stop reason for each wavefront is available. For example:

```

wave_0: pc=0x7fbe4cc0d0b4 (stopped, reason: MEMORY_VIOLATION)

```

- **Assert trap**

This occurs when an `s_trap 2` instruction is executed. The `__builtin_trap()` language builtin, or `llvm.trap` LLVM IR instruction, can be used to generate this AMD GPU instruction.

- **Illegal instruction**

This occurs when the hardware detects an illegal instruction.

- **SIGQUIT ``(Ctrl-)``**

A SIGQUIT signal can be sent to a process with the `kill -s SIGQUIT <pid>` command or by pressing `Ctrl-\'`. See the `--disable-linux-signals` option for more information.

### 3.31.3 Options

Options are passed through the `ROCM_DEBUG_AGENT_OPTIONS` environment variable. For example:

```
ROCM_DEBUG_AGENT_OPTIONS="--all --save-code-objects" \
HSA_TOOLS_LIB=librocm-debug-agent.so.2 ./my_program
```

The supported options are:

- `--a`, `--all`

Prints all wavefronts.

If not specified, only wavefronts that have a triggering event are printed.

- `--s [DIR]`, `--save-code-objects[=DIR]`

Saves all loaded code objects. If the directory is not specified, the code objects are saved in the current directory.

The file name in which the code object is saved is the same as the code object URI with special characters replaced by `'_'`. For example, the code object URI:

```
file:///rocm-debug-agent/rocm-debug-agent-test#offset=14309&size=31336
```

is saved in a file with the name:

```
file____rocm-debug-agent_rocm-debug-agent-test_offset_14309_size_31336
```

- `--o <file-path>`, `--output=<file-path>`

Saves the output produced by the ROCdebug-agent in the specified file.

By default, the output is redirected to `stderr`.

- `--d`, `--disable-linux-signals`

Disables installing a SIGQUIT signal handler, so that the default Linux handler may dump a core file.

By default, the ROCdebug-agent installs a SIGQUIT handler to print the state of all wavefronts when a SIGQUIT signal is sent to the process.

- `--l <log-level>`, `--log-level=<log-level>`

Changes the ROCdebug-agent and ROCdbgapi log level. The log level can be `none`, `info`, `warning`, or `error`.

The default log level is `none`.

- `--h`, `--help`

Displays a usage message and aborts the process.

### 3.31.4 Build the ROCdebug-agent library

The ROCdebug-agent library can be built on Ubuntu 18.04, Ubuntu 20.04, Centos 8.1, RHEL 8.1, and SLES 15 Service Pack 1.

Building the ROCdebug-agent library has the following prerequisites:

1. A C++17 compiler such as GCC 7 or Clang 5.
2. The AMD ROCm software stack which can be installed as part of the AMD ROCm release by the `rocm-dev` package.

3. For Ubuntu 18.04 the following adds the needed packages:

```
apt install libelf-dev libdw-dev
```

4. For CentOS 8.1 and RHEL 8.1 the following adds the needed packages:

```
yum install elfutils-libelf-devel elfutils-devel
```

5. For SLES 15 Service Pack 1 the following adds the needed packages:

```
zypper install libelf-devel libdw-devel
```

6. Python version 3.6 or later is required to run the tests.

An example command-line to build and install the ROCdebug-agent library on Linux is:

```
cd rocm-debug-agent
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=../install ..
make
```

Use the CMAKE\_INSTALL\_PREFIX to specify where the ROCdebug-agent library should be installed. The default location is /usr.

Use CMAKE\_MODULE\_PATH to specify a ' ; ' separated list of paths that will be used to locate cmake modules. It is used to locate the HIP cmake modules required to build the tests. The default is /opt/rocm/hip/cmake

The built ROCdebug-agent library will be placed in:

- build/librocm-debug-agent.so.2\*

To install the ROCdebug-agent library:

```
make install
```

The installed ROCdebug-agent library will be placed in:

- <install-prefix>/lib/librocm-debug-agent.so.2\*
- <install-prefix>/bin/rocm-debug-agent-test
- <install-prefix>/bin/run-test.py
- <install-prefix>/share/rocm-debug-agent/LICENSE.txt
- <install-prefix>/share/rocm-debug-agent/README.md

To use the ROCdebug-agent library, the ROCdbgapi library must be installed. This can be installed as part of the ROCm release by the rocm-dbgapi package.

### 3.31.5 Test the ROCdebug-agent library

To test the ROCdebug-agent library:

```
make test
```

The output should be:



```
Running tests...
Test project /rocm-debug-agent/build
  Start 1: rocm-debug-agent-test
1/1 Test #1: rocm-debug-agent-test ..... Passed    1.59 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =  1.59 sec
```

Tests can be run individually outside of the CTest harness. For example:

```
HSA_TOOLS_LIB=librocm-debug-agent.so.2 test/rocm-debug-agent-test 0
HSA_TOOLS_LIB=librocm-debug-agent.so.2 test/rocm-debug-agent-test 1
HSA_TOOLS_LIB=librocm-debug-agent.so.2 test/rocm-debug-agent-test 2
```

### 3.31.6 Known Limitations and Restrictions

- A disassembly of the wavefront faulting PC is only provided if it is within a code object.

### 3.31.7 Disclaimer

The information contained herein is for informational purposes only and is subject to change without notice. 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.

AMD®, the AMD Arrow logo, ROCm® and combinations thereof are trademarks of Advanced Micro Devices, Inc. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. RedHat® and the Shadowman logo are registered trademarks of Red Hat, Inc. www.redhat.com in the U.S. and other countries. SUSE® is a registered trademark of SUSE LLC in the United States and other countries. Ubuntu® and the Ubuntu logo are registered trademarks of Canonical Ltd. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Copyright (c) 2018-2020 Advanced Micro Devices, Inc. All rights reserved. For the latest HIP Programming Guide documentation, refer to the PDF version at:

[https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD\\_HIP\\_Programming\\_Guide\\_v4.3.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_HIP_Programming_Guide_v4.3.pdf)

## 3.32 System Level Debug

### 3.32.1 ROCm Language & System Level Debug, Flags, and Environment Variables

Kernel options to avoid Ethernet port getting renamed every time you change graphics cards  
 net.ifnames=0 biosdevname=0

### **3.32.1.1 ROCr Error Code**

- 2 Invalid Dimension
- 4 Invalid Group Memory
- 8 Invalid (or Null) Code
- 32 Invalid Format </li>
- 64 Group is too large
- 128 Out of VGPR's
- 0x80000000 Debug Trap

### **3.32.1.2 Command to dump firmware version and get Linux Kernel version**

- `sudo cat /sys/kernel/debug/dri/1/amdgpu_firmware_info`
- `uname -a`

### **3.32.1.3 Debug Flags**

Debug messages when developing/debugging base ROCm driver. You could enable the printing from libhsakmt.so by setting an environment variable, HSAKMT\_DEBUG\_LEVEL. Available debug levels are 3~7. The higher level you set, the more messages will print.

- `export HSAKMT_DEBUG_LEVEL=3` : only `pr_err()` will print.
- `export HSAKMT_DEBUG_LEVEL=4` : `pr_err()` and `pr_warn()` will print.
- `export HSAKMT_DEBUG_LEVEL=5` : We currently don't implement "notice". Setting to 5 is same as setting to 4.
- `export HSAKMT_DEBUG_LEVEL=6` : `pr_err()`, `pr_warn()`, and `pr_info` will print.
- `export HSAKMT_DEBUG_LEVEL=7` : Everything including `pr_debug` will print.

### **3.32.1.4 ROCr level env variable for debug**

- `HSA_ENABLE_SDMA=0`
- `HSA_ENABLE_INTERRUPT=0`
- `HSA_SVM_GUARD_PAGES=0`
- `HSA_DISABLE_CACHE=1`

### 3.32.1.5 Turn Off Page Retry on GFX9/Vega devices

- `sudo -s`
- `echo 1 > /sys/module/amdkfd/parameters/noretry`

### 3.32.1.6 HIP Environment Variables

### 3.32.1.7 OpenCL Debug Flags

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

### 3.32.1.8 PCIe-Debug

Refer here for PCIe-Debug

**More information here on how to debug and profile HIP applications**

- [HIP-Debugging](#)
- [HIP-Profilng](#)

## 3.33 ROCmValidationSuite

The ROCm Validation Suite (RVS) is a system administrator's and cluster manager's tool for detecting and troubleshooting common problems affecting AMD GPU(s) running in a high-performance computing environment, enabled using the ROCm software stack on a compatible platform.

The RVS is a collection of tests, benchmarks and qualification tools each targeting a specific sub-system of the ROCm platform. All of the tools are implemented in software and share a common command line interface. Each set of tests are implemented in a “module” which is a library encapsulating the functionality specific to the tool. The CLI can specify the directory containing modules to use when searching for libraries to load. Each module may have a set of options that it defines and a configuration file that supports its execution.

### 3.33.1 ROCmValidationSuite Modules

#### GPU Properties – GPUP

The GPU Properties module queries the configuration of a target device and returns the device's static characteristics. These static values can be used to debug issues such as device support, performance and firmware problems.

#### GPU Monitor – GM module

The GPU monitor tool is capable of running on one, some or all of the GPU(s) installed and will report various information at regular intervals. The module can be configured to halt another RVS modules execution if one of the quantities exceeds a specified boundary value.

#### PCI Express State Monitor – PESM module?

The PCIe State Monitor tool is used to actively monitor the PCIe interconnect between the host platform and the GPU. The module will register a “listener” on a target GPU's PCIe interconnect, and log a message whenever it detects a state change. The PESM will be able to detect the following state changes:

- PCIe link speed changes
- GPU power state changes

### PCI Express Qualification Tool – PEQT module

The PCIe Qualification Tool consists is used to qualify the PCIe bus on which the GPU is connected. The qualification test will be capable of determining the following characteristics of the PCIe bus interconnect to a GPU:

- Support for Gen 3 atomic completers
- DMA transfer statistics
- PCIe link speed
- PCIe link width

### P2P Benchmark and Qualification Tool – PBQT module

The P2P Benchmark and Qualification Tool is designed to provide the list of all GPUs that support P2P and characterize the P2P links between peers. In addition to testing for P2P compatibility, this test will perform a peer-to-peer throughput test between all P2P pairs for performance evaluation. The P2P Benchmark and Qualification Tool will allow users to pick a collection of two or more GPUs on which to run. The user will also be able to select whether or not they want to run the throughput test on each of the pairs.

Please see the web page “ROCm, a New Era in Open GPU Computing” to find out more about the P2P solutions available in a ROCm environment.

### PCI Express Bandwidth Benchmark – PEBB module

The PCIe Bandwidth Benchmark attempts to saturate the PCIe bus with DMA transfers between system memory and a target GPU card’s memory. The maximum bandwidth obtained is reported to help debug low bandwidth issues. The benchmark should be capable of targeting one, some or all of the GPUs installed in a platform, reporting individual benchmark statistics for each.

### GPU Stress Test - GST module

The GPU Stress Test runs a Graphics Stress test or SGEMM/DGEMM (Single/Double-precision General Matrix Multiplication) workload on one, some or all GPUs. The GPUs can be of the same or different types. The duration of the benchmark should be configurable, both in terms of time (how long to run) and iterations (how many times to run).

The test should be capable driving the power level equivalent to the rated TDP of the card, or levels below that. The tool must be capable of driving cards at TDP-50% to TDP-100%, in 10% incremental jumps. This should be controllable by the user.

### Input EDPp Test - IET module

The Input EDPp Test generates EDP peak power on all input rails. This test is used to verify if the system PSU is capable of handling the worst case power spikes of the board. Peak Current at defined period = 1 minute moving average power.

Examples and about config files [link](#).

## 3.33.2 Prerequisites

Ubuntu :

```
sudo apt-get -y update && sudo apt-get install -y libpci3 libpci-dev doxygen unzip_
↪cmake git
```

CentOS :

```
sudo yum install -y cmake3 doxygen pciutils-devel rpm rpm-build git gcc-c++
```

RHEL :

```
sudo yum install -y cmake3 doxygen rpm rpm-build git gcc-c++

wget http://mirror.centos.org/centos/7/os/x86_64/Packages/pciutils-devel-3.5.1-3.el7.
↪x86_64.rpm

sudo rpm -ivh pciutils-devel-3.5.1-3.el7.x86_64.rpm
```

SLES :

```
sudo SUSEConnect -p sle-module-desktop-applications/15.1/x86_64

sudo SUSEConnect --product sle-module-development-tools/15.1/x86_64

sudo zypper install -y cmake doxygen pciutils-devel libpci3 rpm git rpm-build gcc-c++
```

### 3.33.3 Install ROCm stack, rocblas and rocm\_smi64

Install ROCm stack for Ubuntu/CentOS, Refer <https://github.com/RadeonOpenCompute/ROCm>

Install rocBLAS and rocm\_smi64 :

Ubuntu :

```
sudo apt-get install rocblas rocm_smi64
```

CentOS & RHEL :

```
sudo yum install rocblas rocm_smi64
```

SUSE :

```
sudo zypper install rocblas rocm_smi64
```

**Note:** If rocm\_smi64 is already installed but “/opt/rocm/rocm\_smi/ path doesn’t exist. Do below:

Ubuntu : sudo dpkg -r rocm\_smi64 && sudo apt install rocm\_smi64

CentOS & RHEL : sudo rpm -e rocm\_smi64 && sudo yum install rocm\_smi64

SUSE : sudo rpm -e rocm\_smi64 && sudo zypper install rocm\_smi64

### 3.33.4 Building from Source

This section explains how to get and compile current development stream of RVS.

**Clone repository**

```
git clone https://github.com/ROCm-Developer-Tools/ROCmValidationSuite.git
```

**Configure and build RVS:**

```
cd ROCmValidationSuite
```

If OS is Ubuntu and SLES, use cmake

```
cmake ./ -B./build  
make -C ./build
```

If OS is CentOS and RHEL, use cmake3

```
cmake3 ./ -B./build  
make -C ./build
```

**Build package:**

```
cd ./build  
make package
```

Note: based on your OS, only DEB or RPM package will be built. You may ignore an error for the unrelated configuration

**Install package:**

```
Ubuntu : sudo dpkg -i rocm-validation-suite*.deb  
CentOS & RHEL & SUSE : sudo rpm -i --replacefiles --nodeps rocm-validation-suite*.rpm
```

### Running RVS

Running version built from source code:

```
cd ./build/bin  
sudo ./rvs -d 3  
sudo ./rvsqa.new.sh ; It will run complete rvs test suite
```

### 3.33.5 Regression

Regression is currently implemented for PQT module only. It comes in the form of a Python script `run_regression.py`.

The script will first create valid configuration files on `$RVS_BUILD/regression` folder. It is done by invoking `prq_create_conf.py` script to generate valid configuration files. If you need different tests, modify the `prq_create_conf.py` script to generate them.

Then, it will iterate through generated files and invoke RVS to specifying also JSON output and `-d 3` logging level.

Finally, it will iterate over generated JSON output files and search for ERROR string. Results are written into `$RVS_BUILD/regression/regression_res` file.

Results are written into `$RVS_BUILD/regression/`

#### Environment variables

Before running the `run_regression.py` you first need to set the following environment variables for location of RVS source tree and build folders (adjust for your particular clone):

```
export WB=/work/yourworkfolder  
export RVS=$WB/ROCmValidationSuite  
export RVS_BUILD=$RVS/./build
```

#### Running the script

Just do:

```
cd $RVS/regression
./run_regression.py
```

## 3.34 System Management Interface

A System Management Interface (SMI) event interface is added to the kernel and a ROCm SMI library for system administrators to get notified when specific events occur. On the kernel side, AMDKFD\_IOC\_SMI\_EVENTS input/output control is enhanced to allow notifications propagation to user mode through the event channel.

On the ROCm SMI lib side, APIs are added to set an event mask and receive event notifications with a timeout option. Further, ROCm SMI API details can be found in the PDF generated by Doxygen from source or by referring to the `rocm_smi.h` header file (see the `rsmi_event_notification_*` functions).

For more information, download the latest System Management Interface API guide at:

<https://github.com/RadeonOpenCompute/ROCm>

### 3.34.1 ROCm SMI library

### 3.34.2 ROCm System Management Interface (ROCm SMI) Library

The ROCm System Management Interface Library, or ROCm SMI library, is part of the Radeon Open Compute ROCm software stack . It is a C library for Linux that provides a user space interface for applications to monitor and control GPU applications.

#### 3.34.2.1 Important note about Versioning and Backward Compatibility

The ROCm SMI library is currently under development, and therefore subject to change either at the ABI or API level. The intention is to keep the API as stable as possible even while in development, but in some cases we may need to break backwards compatibility in order to ensure future stability and usability. Following [Semantic Versioning](#) rules, while the ROCm SMI library is in high state of change, the major version will remain 0, and backward compatibility is not ensured.

Once new development has leveled off, the major version will become greater than 0, and backward compatibility will be enforced between major versions.

### 3.34.3 Building ROCm SMI

#### 3.34.3.1 Additional Required software for building

In order to build the ROCm SMI library, the following components are required. Note that the software versions listed are what was used in development. Earlier versions are not guaranteed to work:

- CMake (v3.5.0)
- g++ (5.4.0)

In order to build the latest documentation, the following are required:

- Doxygen (1.8.11)
- latex (pdfTeX 3.14159265-2.6-1.40.16)

The source code for ROCm SMI is available on [Github](#).

After the the ROCm SMI library git repository has been cloned to a local Linux machine, building the library is achieved by following the typical CMake build sequence. Specifically,

```
$ mk -p build
$ cd build
$ cmake <location of root of ROCm SMI library CMakeLists.txt>
$ make
# Install library file and header; default location is /opt/rocm
$ make install
```

The built library will appear in the build folder.

### 3.34.3.2 Building Documentation

The documentation PDF file can be built with the following steps (continued from the steps above):

```
$ make doc
$ cd latex
$ make
```

The reference manual, refman.pdf will be in the latex directory upon a successful build.

### 3.34.3.3 Building Tests

In order to verify the build and capability of ROCm SMI on your system and to see an example of how ROCm SMI can be used, you may build and run the tests that are available in the repo. To build the tests, follow these steps:

```
# Set environment variables used in CMakeLists.txt file
$ ROCM_DIR=<location of ROCm SMI library>
$ mkdir <location for test build>
$ cd <location for test build>
$ cmake -DROCM_DIR=<location of ROCM SMI library .so> <ROCm SMI source root>/tests/
↪ rocm_smi_test
$ make
```

To run the test, execute the program rsmitst that is built from the steps above.

## 3.34.4 Usage Basics

### 3.34.4.1 Device Indices

Many of the functions in the library take a “device index”. The device index is a number greater than or equal to 0, and less than the number of devices detected, as determined by `rsmi_num_monitor_devices()`. The index is used to distinguish the detected devices from one another. It is important to note that a device may end up with a different index after a reboot, so an index should not be relied upon to be constant over reboots.



### 3.34.4.2 Hello ROCm SMI

The only required ROCm-SMI call for any program that wants to use ROCm-SMI is the `rsmi_init()` call. This call initializes some internal data structures that will be used by subsequent ROCm-SMI calls.

When ROCm-SMI is no longer being used, `rsmi_shut_down()` should be called. This provides a way to do any releasing of resources that ROCm-SMI may have held. In many cases, this may have no effect, but may be necessary in future versions of the library.

A simple “Hello World” type program that displays the device ID of detected devices would look like this:

```
#include <stdint.h>
#include "rocm_smi/rocm_smi.h"
int main() {
    rsmi_status_t ret;
    uint32_t num_devices;
    uint64_t dev_id;

    // We will skip return code checks for this example, but it
    // is recommended to always check this as some calls may not
    // apply for some devices or ROCm releases

    ret = rsmi_init(0);
    ret = rsmi_num_monitor_devices(&num_devices);

    for (int i=0; i < num_devices; ++i) {
        ret = rsmi_dev_id_get(i, &dev_id);
        // dev_id holds the device ID of device i, upon a
        // successful call
    }
    ret = rsmi_shut_down();
    return 0;
}
```

## 3.34.5 SYSFS Interface

### 3.34.5.1 Naming and data format standards for sysfs files

The `libsensors` library offers an interface to the raw sensors data through the `sysfs` interface. Since `lm-sensors 3.0.0`, `libsensors` is completely chip-independent. It assumes that all the kernel drivers implement the standard `sysfs` interface described in this document. This makes adding or updating support for any given chip very easy, as `libsensors`, and applications using it, do not need to be modified. This is a major improvement compared to `lm-sensors 2`.

Note that motherboards vary widely in the connections to sensor chips. There is no standard that ensures, for example, that the second temperature sensor is connected to the CPU, or that the second fan is on the CPU. Also, some values reported by the chips need some computation before they make full sense. For example, most chips can only measure voltages between 0 and +4V. Other voltages are scaled back into that range using external resistors. Since the values of these resistors can change from motherboard to motherboard, the conversions cannot be hard coded into the driver and have to be done in user space.

For this reason, even if we aim at a chip-independent `libsensors`, it will still require a configuration file (e.g. `/etc/sensors.conf`) for proper values conversion, labeling of inputs and hiding of unused inputs.

An alternative method that some programs use is to access the `sysfs` files directly. This document briefly describes the standards that the drivers follow, so that an application program can scan for entries and access this data in a simple and consistent way. That said, such programs will have to implement conversion, labeling and hiding of inputs. For this reason, it is still not recommended to bypass the library.

Each chip gets its own directory in the sysfs `/sys/devices` tree. To find all sensor chips, it is easier to follow the device symlinks from `/sys/class/hwmon/hwmon*`.

Up to lm-sensors 3.0.0, libsensors looks for hardware monitoring attributes in the “physical” device directory. Since lm-sensors 3.0.1, attributes found in the hwmon “class” device directory are also supported. Complex drivers (e.g. drivers for multifunction chips) may want to use this possibility to avoid namespace pollution. The only drawback will be that older versions of libsensors won’t support the driver in question.

All sysfs values are fixed point numbers.

There is only one value per file, unlike the older `/proc` specification. The common scheme for files naming is: `<type><number>_<item>`. Usual types for sensor chips are “in” (voltage), “temp” (temperature) and “fan” (fan). Usual items are “input” (measured value), “max” (high threshold, “min” (low threshold). Numbering usually starts from 1, except for voltages which start from 0 (because most data sheets use this). A number is always used for elements that can be present more than once, even if there is a single element of the given type on the specific chip. Other files do not refer to a specific element, so they have a simple name, and no number.

Alarms are direct indications read from the chips. The drivers do NOT make comparisons of readings to thresholds. This allows violations between readings to be caught and alarmed. The exact definition of an alarm (for example, whether a threshold must be met or must be exceeded to cause an alarm) is chip-dependent.

When setting values of hwmon sysfs attributes, the string representation of the desired value must be written, note that strings which are not a number are interpreted as 0! For more on how written strings are interpreted see the “sysfs attribute writes interpretation” section at the end of this file.

[0-*	denotes any positive number starting from 0
[1-*	denotes any positive number starting from 1
RO	read only value
WO	write only value
RW	read/write value

Read/write values may be read-only for some chips, depending on the hardware implementation.

All entries (except name) are optional, and should only be created in a given driver if the chip has the feature.

### 3.34.6 Global Attributes

name	<p>The chip name. This should be a short, lowercase string, not containing whitespace, dashes, or the wildcard character '*'. This attribute represents the chip name.</p> <p>It is the only mandatory attribute. I2C devices get this attribute created automatically.</p> <p>RO</p>
update_interval	<p>The interval at which the chip will update readings.</p> <p>Unit: millisecond</p> <p>RW</p> <p>Some devices have a variable update rate or interval. This attribute can be used to change it to the desired value.</p>



### 3.34.7 Voltages

in[0-*]_min	Voltage min value. Unit: millivolt RW
in[0-*]_lcrit	Voltage critical min value. Unit: millivolt RW If voltage drops to or below this limit, the system may take drastic action such as power down or reset. At the very least, it should report a fault.
in[0-*]_max	Voltage max value. Unit: millivolt RW
in[0-*]_crit	Voltage critical max value. Unit: millivolt RW If voltage reaches or exceeds this limit, the system may take drastic action such as power down or reset. At the very least, it should report a fault.
in[0-*]_input	Voltage input value. Unit: millivolt RO Voltage measured on the chip pin. Actual voltage depends on the scaling resistors on the motherboard, as recommended in the chip datasheet. This varies by chip and by motherboard. Because of this variation, values are generally NOT scaled by the chip driver, and must be done by the application. However, some drivers (notably lm87 and via686a) do scale, because of internal resistors built into a chip. These drivers will output the actual voltage. Rule of thumb: drivers should report the voltage values at the “pins” of the chip.
in[0-*]_average	Average voltage Unit: millivolt
<b>3.34. System Management Interface</b>	<b>RO</b> <b>153</b>
in[0-*]_lowest	Historical minimum voltage

Also see the Alarms section for status flags associated with voltages.



### 3.34.8 Fans

fan[1-*]_min	Fan minimum value Unit: revolution/min (RPM) RW
fan[1-*]_max	Fan maximum value Unit: revolution/min (RPM) Only rarely supported by the hardware. RW
fan[1-*]_input	Fan input value. Unit: revolution/min (RPM) RO
fan[1-*]_div	Fan divisor. Integer value in powers of two (1, 2, 4, 8, 16, 32, 64, 128). RW Some chips only support values 1, 2, 4 and 8. Note that this is actually an internal clock divisor, which affects the measurable speed range, not the read value.
fan[1-*]_pulses	Number of tachometer pulses per fan revolution. Integer value, typically between 1 and 4. RW This value is a characteristic of the fan connected to the device's input, so it has to be set in accordance with the fan model. Should only be created if the chip has a register to configure the number of pulses. In the absence of such a register (and thus attribute) the value assumed by all devices is 2 pulses per fan revolution.
fan[1-*]_target	Desired fan speed Unit: revolution/min (RPM) RW Only makes sense if the chip supports closed-loop fan speed
156	control based on the fan speed <b>Chapter 3: ROCm Learning Center</b>
fan[1-*]_label	Suggested fan channel label.



Also see the Alarms section for status flags associated with fans.



### 3.34.9 Pulse with Modulation

pwm[1- <i>*</i> ]	Pulse width modulation fan control. Integer value in the range 0 to 255 RW 255 is max or 100%.
pwm[1- <i>*</i> ].enable	Fan speed control method: 0: no fan speed control (i.e. fan at full speed) 1: manual fan speed control enabled (using pwm[1- <i>*</i> ]) 2+: automatic fan speed control enabled Check individual chip documentation files for automatic mode details. RW
pwm[1- <i>*</i> ].mode	0: DC mode (direct current) 1: PWM mode (pulse-width modulation) RW
pwm[1- <i>*</i> ].freq	Base PWM frequency in Hz. Only possibly available when pwmN_mode is PWM, but not always present even then. RW
pwm[1- <i>*</i> ].auto_channels_temp	Select which temperature channels affect this PWM output in auto mode. Bitfield, 1 is temp1, 2 is temp2, 4 is temp3 etc. . . Which values are possible depend on the chip used. RW
pwm[1-].auto_point[1-].pwm pwm[1-].auto_point[1-].temp pwm[1-].auto_point[1-].temp_hyst	Define the PWM vs temperature curve. Number of trip points is chip-dependent. Use this for chips which associate trip points to PWM output channels. RW
temp[1-].auto_point[1-].pwm temp[1-].auto_point[1-].temp temp[1-].auto_point[1-].temp_hyst	Define the PWM vs temperature curve. Number of trip points is chip dependent. Use this for chips which associate trip points to temperature channels. RW
<b>3.34. System Management Interface</b>	

There is a third case where trip points are associated to both PWM output channels and temperature channels: the PWM values are associated to PWM output channels while the temperature values are associated to temperature channels. In that case, the result is determined by the mapping between temperature inputs and PWM outputs. When several temperature inputs are mapped to a given PWM output, this leads to several candidate PWM values. The actual result is up to the chip, but in general the highest candidate value (fastest fan speed) wins.



### 3.34.10 Temperatures

temp[1-*]_type	<p>Sensor type selection. Integers 1 to 6 RW 1: CPU embedded diode 2: 3904 transistor 3: thermal diode 4: thermistor 5: AMD AMDSI 6: Intel PECI Not all types are supported by all chips</p>
temp[1-*]_max	<p>Temperature max value. Unit: millidegree Celsius (or millivolt, see below) RW</p>
temp[1-*]_min	<p>Temperature min value. Unit: millidegree Celsius RW</p>
temp[1-*]_max_hyst	<p>Temperature hysteresis value for max limit. Unit: millidegree Celsius Must be reported as an absolute temperature, NOT a delta from the max value. RW</p>
temp[1-*]_min_hyst	<p>Temperature hysteresis value for min limit. Unit: millidegree Celsius Must be reported as an absolute temperature, NOT a delta from the min value. RW</p>
temp[1-*]_input	<p>Temperature input value. Unit: millidegree Celsius RO</p>
temp[1-*]_crit	<p>Temperature critical max value, typically greater than corresponding temp_max values. Unit: millidegree Celsius</p>
162	<p>RW <b>Chapter 3. ROCm Learning Center</b></p>
temp[1-*]_crit_hyst	<p>Temperature hysteresis value for critical limit.</p>

Some chips measure temperature using external thermistors and an ADC, and report the temperature measurement as a voltage. Converting this voltage back to a temperature (or the other way around for limits) requires mathematical functions not available in the kernel, so the conversion must occur in user space. For these chips, all temp\* files described above should contain values expressed in millivolt instead of millidegree Celsius. In other words, such temperature channels are handled as voltage channels by the driver.

Also see the Alarms section for status flags associated with temperatures.





### 3.34.11 Currents

curr[1-*]_max	Current max value Unit: milliampere RW
curr[1-*]_min	Current min value. Unit: milliampere RW
curr[1-*]_lcrit	Current critical low value Unit: milliampere RW
curr[1-*]_crit	Current critical high value. Unit: milliampere RW
curr[1-*]_input	Current input value Unit: milliampere RO
curr[1-*]_average	Average current use Unit: milliampere RO
curr[1-*]_lowest	Historical minimum current Unit: milliampere RO
curr[1-*]_highest	Historical maximum current Unit: milliampere RO
curr[1-*]_reset_history	Reset currX_lowest and currX_highest WO
curr_reset_history	
<b>3.34. System Management Interface</b>	Reset currX_lowest and currX_highest for all sensors <b>165</b> WO
curr[1-*]_enable	

Also see the Alarms section for status flags associated with currents.



### 3.34.12 Power

power[1-*]_average	Average power use Unit: microWatt RO
power[1-*]_average_interval	Power use averaging interval. A poll notification is sent to this file if the hardware changes the averaging interval. Unit: milliseconds RW
power[1-*]_average_interval_max	Maximum power use averaging interval Unit: milliseconds RO
power[1-*]_average_interval_min	Minimum power use averaging interval Unit: milliseconds RO
power[1-*]_average_highest	Historical average maximum power use Unit: microWatt RO
power[1-*]_average_lowest	Historical average minimum power use Unit: microWatt RO
power[1-*]_average_max	A poll notification is sent to power[1-*]_average when power use rises above this value. Unit: microWatt RW
power[1-*]_average_min	A poll notification is sent to power[1-*]_average when power use sinks below this value. Unit: microWatt RW
power[1-*]_input	Instantaneous power use Unit: microWatt

Also see the Alarms section for status flags associated with power readings.

### 3.34.13 Energy

energy[1-*]_input	Cumulative energy use Unit: microJoule RO
energy[1-*]_enable	Enable or disable the sensors When disabled the sensor read will return -ENODATA 1: Enable 0: Disable RW

### 3.34.14 Humidity

humidity[1-*]_input	Humidity Unit: milli-percent (per cent mille, pcm) RO
humidity[1-*]_enable	Enable or disable the sensors When disabled the sensor read will return -ENODATA 1: Enable 0: Disable RW

### 3.34.15 Alarms

Each channel or limit may have an associated alarm file, containing a boolean value. 1 means than an alarm condition exists, 0 means no alarm.

Usually a given chip will either use channel-related alarms, or limit-related alarms, not both. The driver should just reflect the hardware implementation.

in[0-*]_alarm curr[1-*]_alarm power[1-*]_alarm fan[1-*]_alarm temp[1-*]_alarm	Channel alarm 0: no alarm 1: alarm RO
---	--

OR

in[0-*]_min_alarm in[0-*]_max_alarm in[0-*]_lcrit_alarm in[0-*]_crit_alarm curr[1-*]_min_alarm curr[1-*]_max_alarm curr[1-*]_lcrit_alarm curr[1-*]_crit_alarm power[1-*]_cap_alarm power[1-*]_max_alarm power[1-*]_crit_alarm fan[1-*]_min_alarm fan[1-*]_max_alarm temp[1-*]_min_alarm temp[1-*]_max_alarm temp[1-*]_lcrit_alarm temp[1-*]_crit_alarm temp[1-*]_emergency_alarm	Limit alarm 0: no alarm 1: alarm RO
---	--

Each input channel may have an associated fault file. This can be used to notify open diodes, unconnected fans etc. where the hardware supports it. When this boolean has value 1, the measurement for that channel should not be trusted.

fan[1-*]_fault temp[1-*]_fault	Input fault condition 0: no fault occurred 1: fault condition RO
-----------------------------------	---

Some chips also offer the possibility to get beeped when an alarm occurs:

beep_enable	<p>Master beep enable</p> <p>0: no beeps</p> <p>1: beeps</p> <p>RW</p>
in[0-*]_beep curr[1-*]_beep fan[1-*]_beep temp[1-*]_beep	<p>Channel beep</p> <p>0: disable</p> <p>1: enable</p> <p>RW</p>

In theory, a chip could provide per-limit beep masking, but no such chip was seen so far.

Old drivers provided a different, non-standard interface to alarms and beeps. These interface files are deprecated, but will be kept around for compatibility reasons:

alarms	<p>Alarm bitmask.</p> <p>RO</p> <p>Integer representation of one to four bytes.</p> <p>A ‘1’ bit means an alarm.</p> <p>Chips should be programmed for ‘comparator’ mode so that the alarm will ‘come back’ after you read the register if it is still valid.</p> <p>Generally a direct representation of a chip’s internal alarm registers; there is no standard for the position of individual bits. For this reason, the use of this interface file for new drivers is discouraged. Use individual *_alarm and *_fault files instead.</p> <p>Bits are defined in kernel/include/sensors.h.</p>
beep_mask	<p>Bitmask for beep.</p> <p>Same format as ‘alarms’ with the same bit locations, use discouraged for the same reason. Use individual *_beep files instead.</p> <p>RW</p>

### 3.34.16 Intrusion detection

intrusion[0-*]_alarm	Chassis intrusion detection 0: OK 1: intrusion detected RW Contrary to regular alarm flags which clear themselves automatically when read, this one sticks until cleared by the user. This is done by writing 0 to the file. Writing other values is unsupported.
intrusion[0-*]_beep	Chassis intrusion beep 0: disable 1: enable RW

### 3.34.17 Average Sample Configuration

Devices allowing for reading {in,power,curr,temp}\_average values may export attributes for controlling number of samples used to compute average.

samples	Sets number of average samples for all types of measurements. RW
in_samples	Sets number of average samples for specific type of measurements.
power_samples	Note that on some devices it won't be possible to set all of
curr_samples	them to different values so changing one might also change
curr_samples	some others. RW



### 3.34.18 sysfs attribute writes interpretation

hwmon sysfs attributes always contain numbers, so the first thing to do is to convert the input to a number, there are 2 ways to do this depending whether the number can be negative or not: `unsigned long u = simple_strtoul(buf, NULL, 10); long s = simple_strtol(buf, NULL, 10);`

With *buf* being the buffer with the user input being passed by the kernel. Notice that we do not use the second argument of `strto[u]l`, and thus cannot tell when 0 is returned, if this was really 0 or is caused by invalid input. This is done deliberately as checking this everywhere would add a lot of code to the kernel.

Notice that it is important to always store the converted value in an unsigned long or long, so that no wrap around can happen before any further checking.

After the input string is converted to an (unsigned) long, the value should be checked if its acceptable. Be careful with further conversions on the value before checking it for validity, as these conversions could still cause a wrap around before the check. For example do not multiply the result, and only add/subtract if it has been divided before the add/subtract.

What to do if a value is found to be invalid, depends on the type of the sysfs attribute that is being set. If it is a continuous setting like a `tempX_max` or `inX_max` attribute, then the value should be clamped to its limits using `clamp_val(value, min_limit, max_limit)`. If it is not continuous like for example a `tempX_type`, then when an invalid value is written, `-EINVAL` should be returned.

Example1, `temp1_max`, register is a signed 8 bit value (-128 - 127 degrees):

```
long v = simple_strtol(buf, NULL, 10) / 1000;
v = clamp_val(v, -128, 127);
/* write v to register */
```

Example2, fan divider setting, valid values 2, 4 and 8:

```
unsigned long v = simple_strtoul(buf, NULL, 10);

switch (v) {
case 2: v = 1; break;
case 4: v = 2; break;
case 8: v = 3; break;
default:
    return -EINVAL;
}
/* write v to register */
```

### 3.34.19 Performance

The `pcie_bw` sysfs file will report the usage of the PCIe bus over the last second, as a string with 3 integers: “bytes-received bytes-sent mps”. As there is no efficient way to calculate the size of each packet transmitted to and from the GPU in real time, the maximum payload size (mps), or the largest size of a PCIe packet, is included. The estimated bandwidth can then be calculated using by “bytes-received\*mps + bytes-sent\*mps” sed and multiplied by the number of packets received and sent.

### 3.34.20 KFD Topology

Application software needs to understand the properties of the underlying hardware to leverage the performance capabilities of the platform for feature utilization and task scheduling. The sysfs topology exposes this information in a loosely hierarchal order. The information is populated by the KFD driver is gathered from ACPI (CRAT) and AMDGPU base driver.

The sysfs topology is arranged hierarchically as following. The root directory of the topology is `/sys/devices/virtual/kfd/kfd/topology/nodes/`

Based on the platform inside this directory there will be sub-directories corresponding to each HSA Agent. A system with N HSA Agents will have N directories as shown below.

```
/sys/devices/virtual/kfd/kfd/topology/nodes/0/  
/sys/devices/virtual/kfd/kfd/topology/nodes/1/  
.  
.  
/sys/devices/virtual/kfd/kfd/topology/nodes/N-1/
```

### 3.34.21 HSA Agent Information

The HSA Agent directory and the sub-directories inside that contains all the information about that agent. The following are the main information available.

### 3.34.22 Node Information

This is available in the root directory of the HSA agent. This provides information about the compute capabilities of the agent which includes number of cores or compute units, SIMD count and clock speed.

### 3.34.23 Memory

The memory bank information attached to this agent is populated in “mem\_banks” subdirectory. `/sys/devices/virtual/kfd/kfd/topology/nodes/N/mem_banks`

### 3.34.24 Cache

The caches available for this agent is populated in “cache” subdirectory `/sys/devices/virtual/kfd/kfd/topology/nodes/N/cache`

### 3.34.25 IO-LINKS

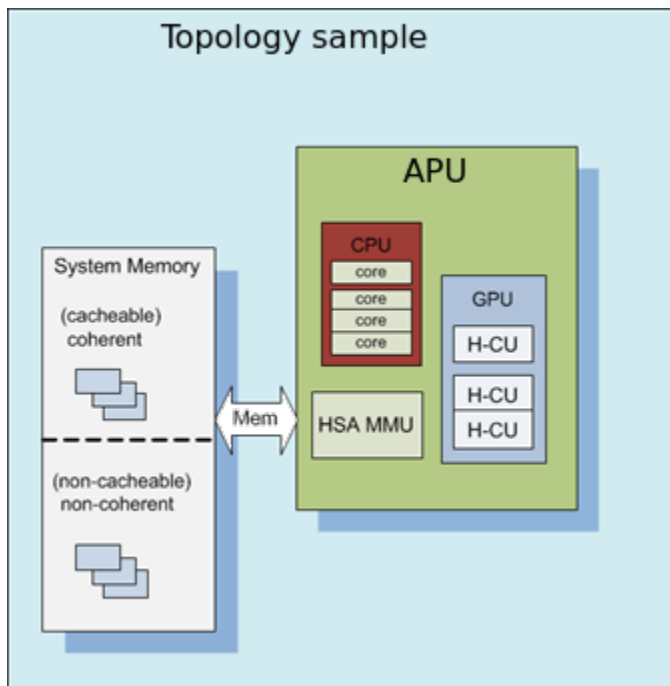
The IO links provides HSA agent interconnect information with latency (cost) between agents. This is useful for peer-to-peer transfers.

### 3.34.26 How to use topology information

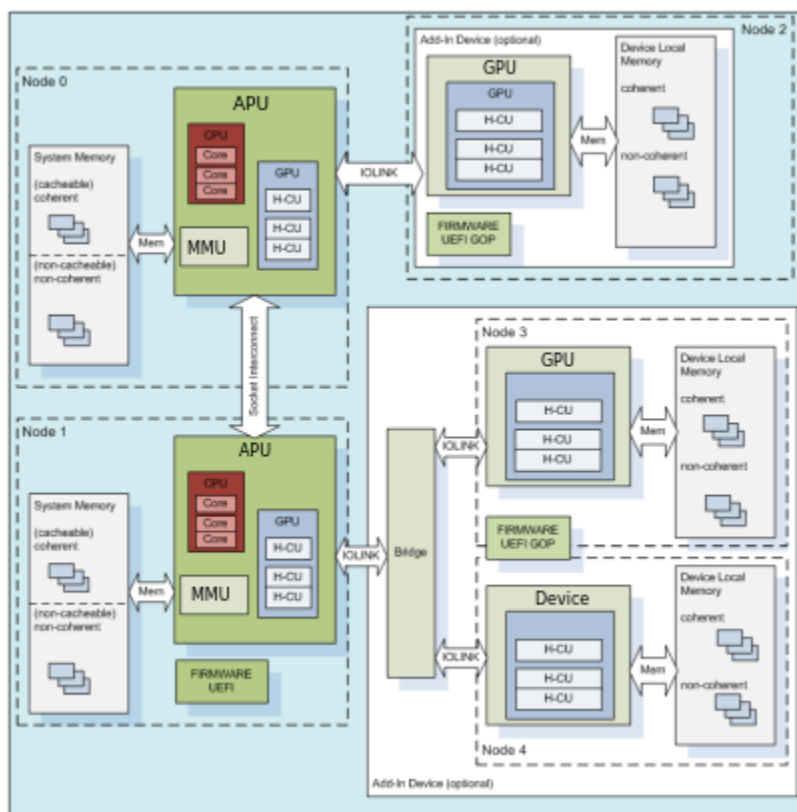
The information provided in sysfs should not be directly used by application software. Application software should always use Thunk library API (libhsakmt) to access topology information. Please refer to Thunk API for more information.

The data are associated with a node ID, forming a per-node element list which references the elements contained at relative offsets within that list. A node associates with a kernel agent or agent. Node ID's should be 0-based, with the "0" ID representing the primary elements of the system (e.g., "boot cores", memory) if applicable. The enumeration order and—if applicable—values of the ID should match other information reported through mechanisms outside of the scope of the requirements;

For example, the data and enumeration order contained in the ACPI SRAT table on some systems should match the memory order and properties reported through HSA. Further detail is out of the scope of the System Architecture and outlined in the Runtime API specification.



Each of these nodes is interconnected with other nodes in more advanced systems to the level necessary to adequately describe the topology.



Where applicable, the node grouping of physical memory follows NUMA principles to leverage memory locality in software when multiple physical memory blocks are available in the system and agents have a different “access cost” (e.g., bandwidth/latency) to that memory.

#### KFD Topology structure for AMDGPU :

```
sysfsclasskfd
sysfsclasskfdtopology
sysfsclasskfdtopologynodes0
sysfsclasskfdtopologynodes0iolinks01
sysfsclasskfdtopologynodes0membanks0
sysfs-class-kfd-topology-nodes-N-caches
```

```
[--setsclk LEVEL [LEVEL ...]] [--setmclk LEVEL [LEVEL ...]] [--setpcie LEVEL [LEVEL ...]] [--set-
slevel
```

### 3.34.27 SMI Event Interface and Library

An SMI event interface is added to the kernel and ROCm SMI lib for system administrators to get notified when specific events occur. On the kernel side, AMDKFD\_IOC\_SMI\_EVENTS input/output control is added to allow notifications propagation to user mode through the event channel.

On the ROCm SMI lib side, APIs are added to set an event mask and receive event notifications with a timeout option. Further, ROCm SMI API details can be found in the PDF generated by Doxygen from source or by referring to the rocm\_smi.h header file (see the rsmi\_event\_notification\_\* functions).

### 3.34.28 ROCR\_VISIBLE\_DEVICES

It is possible to rearrange or isolate the collection of ROCm GPU/GCD devices that are available on a ROCm platform. This can be achieved at the start of an application by way of ROCR\_VISIBLE\_DEVICES environment variable.

To make devices visible to an application, they must be specified as a comma-separated list of enumerable devices, where devices are identified by their enumeration index or UUID.

For example, consider a ROCm platform with the following devices:

Device	Enumeration Index	UUID
Device 1	0	GPU-365628c172834d70
Device 2	1	GPU-368988c172123d70
Device 3	2	GPU-367458c172345d70
Device 4	4	GPU-363688c172386d70

To use devices 0 and 2 from the above-mentioned ROCm platform and to enumerate them in that order, one can employ ROCR\_VISIBLE\_DEVICES in the following ways:

- ROCR\_VISIBLE\_DEVICES=0,2
- ROCR\_VISIBLE\_DEVICES=0,GPU-367458c172345d70
- ROCR\_VISIBLE\_DEVICES=GPU-365628c172834d70,2
- ROCR\_VISIBLE\_DEVICES=GPU-365628c172834d70,GPU-363688c172386d70

Cooperative applications can use this to effectively allocate GPU/GCDs among themselves.

#### 3.34.28.1 Interaction between ROCR\_VISIBLE\_DEVICES and CUDA\_VISIBLE\_DEVICES

The ROCR\_VISIBLE\_DEVICES (RVD) environment is defined by ROCm stack to operate at the ROCr level. The ROCr implementation surfaces all GPU devices when users have not explicitly defined the environment. If defined, ROCr surfaces only those GPU devices that fulfil user requests.

CUDA\_VISIBLE\_DEVICES (CVD) controls the subset of GPU devcies that are available to an application. It builds on GPU devices surfaced by ROCr. The CVD value is legal only if it is a subset of the GPU device indices surfaced by ROCr.

This is best illustrated by the following example:

1. Consider a system that has 8 devices - 0, 1, 2, 3, 4, 5, 6, 7
2. User specifies RVD to select 4 devices - 4, 5, 6, 7
3. These four devices will be available to application as 0, 1, 2, 3

Note the indices of GPU devices as they become available to an application. Users can specify CVD to select a subset of these 4 devices. For example, they can specify CVD as 1,2 or 1,3 or 0,3 or 3,2 etc

Setting both RVD and CVD is typically unnecessary and may be harmful. Use of both environments can play a role when using multiple GPUs and mixing high level languages within a single process. For example, if a single Linux process uses both HIP and OpenCL and wants to use two GPUs such that HIP uses one GPU and OpenCL uses the other, then RVD will select the two GPUs that are assigned to the process, and CVD will select a single GPU (index 0 or 1), from those allowed by RVD, for use by HIP. OpenCL has its own variable enabling it to use the other GPU as allowed by RVD.

Usually, users will not need per language controls either because the process only runs one language or the languages need to cooperate within the same device and will be best served by RVD alone.

It is therefore recommended that ROCm applications use RVD.

### 3.34.29 Device cgroup

At a system administration level, the GPU/GCD isolation is possible using the device control group (cgroup). For all the AMD GPUs in a compute node, the ROCK-Kernel-Driver exposes a single compute device file `/dev/kfd` and a separate (Direct Rendering Infrastructure) render device files `/dev/dri/renderDN` for each device. To participate in the Linux kernel's cgroup infrastructure, the ROCK driver relies on the render device files.

For example, consider a compute node with the two AMD GPUs. The ROCK-Kernel-Driver exposes the following device files:

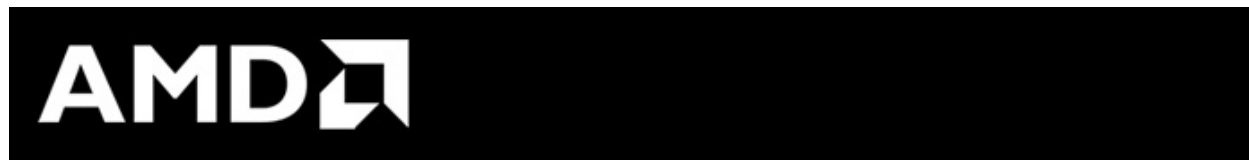
```
crw-rw-rw- 1 root root 240, 0 Apr 22 10:31 /dev/kfd
crw-rw---- 1 root video 226, 128 Apr 22 10:31 /dev/dri/renderD128
crw-rw---- 1 root video 226, 129 Apr 22 10:31 /dev/dri/renderD129
```

A ROCm application running on this compute node can use both GPUs only if it has access to all the above-listed device files. The administrator can restrict the devices an application can access by using device cgroup. The device cgroup subsystem allows or denies access to devices by applications in a cgroup. If a cgroup has whitelisted only `/dev/kfd` and `/dev/dri/renderD129`, then applications in that cgroup will have access only to that single GPU.

Refer to the Linux kernel's cgroup documentation for information on how to create a cgroup and whitelist devices.

For cgroup-v1, refer <https://www.kernel.org/doc/Documentation/cgroup-v1/devices.txt>

For cgroup-v2, refer <https://www.kernel.org/doc/Documentation/cgroup-v2.txt>



## 3.35 ROCm Command Line Interface

### 3.35.1 Clock and Temperature Management

This repository includes the AMD ROCm-SMI tool. This tool exposes functionality for clock and temperature management of the ROCm-enabled system.

For detailed and up to date usage information, use:

```
/opt/rocm/bin/rocm-smi -h
```

Or see below for information on:

- Optional Arguments
- Display Options
- Topology
- Pages Information
- Hardware-related Information
- Software-related/controlled Information
- Set Options
- Reset Options
- Auto-response Options
- Output Options

## Installation

You may find rocm-smi at the following location after installing the rocm package:

```
/opt/rocm/bin/rocm-smi
```

Alternatively, you may clone this repository and run the tool directly.

## Version

The SMI will report a “version” which is the version of the kernel installed:

AMD ROCm System Management Interface v\$(uname)

For ROCK installations, this will be the AMDGPU module version (e.g. 5.0.71) For non-ROCK or monolithic ROCK installations, this will be the kernel version, which will be equivalent to the following bash command:

```
$(uname -a) | cut -d ' ' -f 3)
```

## Usage

For detailed and up to date usage information, see:

```
/opt/rocm/bin/rocm-smi -h
```

For your convenience, the output from the -h flag is as follows:

AMD ROCm System Management Interface | ROCM-SMI version: 1.4.1 | Kernel version: 5.6.20

**usage: rocm-smi [-h] [-d DEVICE [DEVICE ...]] [-alldevices] [-showhw] [-a] [-i] [-v] [-showdriverversion] [-showfwinfo [BLOCK [BLOCK ...]]] [-showmclkrange] [-showmemvendor] [-showsclkrange] [-showproductname] [-showserial] [-showuniqueid] [-showvoltage] [-showbus] [-showpagesinfo] [-showpendingpages] [-showretiredpages] [-showunreservablepages] [-f] [-P] [-t] [-u] [-showmemuse] [-showvoltage] [-b] [-c] [-g] [-l] [-M] [-m] [-o] [-p] [-S] [-s] [-showmeminfo TYPE [TYPE ...]] [-showpids] [-showpidgpus [SHOWPIDGPUS [SHOWPIDGPUS ...]]] [-showreplaycount] [-showrasinfo [SHOWRASINFO [SHOWRASINFO ...]]] [-showvc] [-showxgmierr] [-showtopo] [-showtopoweight] [-showtopohops] [-showtopotype] [-showtoponuma] [-r] [-resetfans] [-resetprofile] [-resetpoweroverdrive] [-resetxgmierr] [-setsclk LEVEL [LEVEL ...]] [-setmclk LEVEL [LEVEL ...]] [-setpcie LEVEL [LEVEL ...]] [-setslevel SCLKLEVEL SCLK SVOLT] [-setmlevel MCLKLEVEL MCLK MVOLT] [-setvc POINT SCLK SVOLT]**

[--setsrange MINMAX SCLK] [--setmrange MINMAX SCLK] [--setfan LEVEL] [--setperflvel LEVEL] [--setoverdrive %] [--setmemoverdrive %] [--setpoweroverdrive WATTS] [--setprofile SETPROFILE] [--rasenable BLOCK ERRTYPE] [--rasdisable BLOCK ERRTYPE] [--rasinject BLOCK] [--gpureset] [--load FILE | --save FILE] [--autorespond RESPONSE] [--loglevel LEVEL] [--json] [--csv]

Optional Arguments	
-h, --help	show this help message and exit
--gpureset	Reset specified GPU (One GPU must be specified)
--load FILE	Load Clock, Fan, Performance and Profile settings from FILE
--save FILE	Save Clock, Fan, Performance and Profile settings to FILE

-d DEVICE [DEVICE ...], --device DEVICE [DEVICE ...] Execute command on specified device

Display Options	
--alldvices	
--showhw	Show Hardware details
-a, --showallinfo	Show Temperature, Fan and Clock values
Topology	
-i, --showid	Show GPU ID
-v, --showvbi	Show VBIOS version
--showdriverversion	Show kernel driver version
--showfwinfo [BLOCK [BLOCK ...]]	Show FW information
--showmclkrange	Show mclk range
--showmemvendor	Show GPU memory vendor
--showscclkrange	Show scclk range
--showproductname	Show SKU/Vendor name
--showserial	Show GPU's Serial Number
--showuniqueid	Show GPU's Unique ID
--showvoltage	Show voltage range
--showbus	Show PCI bus number
Pages Information	
--showpagesinfo	Show retired, pending and unreservable pages
--showpendingpages	Show pending retired pages
--showretiredpages	Show retired pages
--showunreservablepages	Show unreservable pages
Hardware-related Information	
-f, --showfan	Show current fan speed
-P, --showpower	Show current Average Graphics Package Power Consumption
-t, --showtemp	Show current temperature
-u, --showuse	Show current GPU use
--showmemuse	Show current GPU memory used
--showvoltage	Show current GPU voltage
Software-related/controlled information	
-b, --showbw	Show estimated PCIe use
-c, --showclocks	Show current clock frequencies
-g, --showguclocks	Show current GPU clock frequencies
-l, --showprofile	Show Compute Profile attributes
-M, --showmaxpower	Show maximum graphics package power this GPU will consume
-m, --showmemoverdrive	Show current GPU Memory Clock OverDrive level
-o, --showoverdrive	Show current GPU Clock OverDrive level
-p, --showperflvel	Show current DPM Performance Level



Table 1 – continued from previous page

-S, -showclkvolt	Show supported GPU and Memory Clocks and Voltages
-s, -showclkfrq	Show supported GPU and Memory Clock
-showmeminfo TYPE [TYPE ...]	Show Memory usage information for given block(s) TYPE
-showpids	Show current running KFD PIDs
-showpidgpus [SHOWPIDGPUS [SHOWPIDGPUS ...]]	Show GPUs used by specified KFD PIDs (all if no arg given)
-showreplaycount	Show PCIe Replay Count
-showrasinfo [SHOWRASINFO [SHOWRASINFO ...]]	Show RAS enablement information and error counts for the specified block
-showvc	Show voltage curve
-showxgmierr	Show XGMI error information since last read
-showtopo	Show hardware topology information
-showtopoweight	Shows the relative weight between GPUs
-showtopohops	Shows the number of hops between GPUs
-showtopotype	Shows the link type between GPUs
-showtoponuma	Shows the numa nodes
<b>Set Options</b>	
-setclk LEVEL [LEVEL ...]	Set GPU Clock Frequency Level(s) (requires manual Perf level)
-setmclk LEVEL [LEVEL ...]	Set GPU Memory Clock Frequency Level(s) (requires manual Perf level)
-setpcie LEVEL [LEVEL ...]	Set PCIE Clock Frequency Level(s) (requires manual Perf level)
-setslevel SCLKLEVEL SCLK SVOLT	Change GPU Clock frequency (MHz) and Voltage (mV) for a specific Level
-setmlevel MCLKLEVEL MCLK MVOLT	Change GPU Memory clock frequency (MHz) and Voltage for (mV) a specific Level
-setvc POINT SCLK SVOLT	Change SCLK Voltage Curve (MHz mV) for a specific point
-setsrange MINMAX SCLK	Set min(0) or max(1) SCLK speed
-setmrange MINMAX MCLK	Set min(0) or max(1) MCLK speed
-setfan LEVEL	Set GPU Fan Speed (Level or %)
-setperflevel LEVEL	Set Performance Level
-setoverdrive %	Set GPU OverDrive level (requires manual high Perf level)
-setmemoverdrive %	Set GPU Memory Overclock OverDrive level (requires manual high Perf level)
-setpoweroverdrive WATTS	Set the maximum GPU power using Power OverDrive in Watts
-setprofile SETPROFILE	Specify Power Profile level (#) or a quoted string of CUSTOM Profile attributes
-rasenable BLOCK ERRTYPE	Enable RAS for specified block and error type
-rasdisable BLOCK ERRTYPE	Disable RAS for specified block and error type
-rasinject BLOCK	Inject RAS poison for specified block (ONLY WORKS ON UNSECURED GPUs)
<b>Reset Options</b>	
-r, -resetclocks	Reset clocks and OverDrive to default
-resetfans	Reset fans to automatic (driver) control
-resetprofile	Reset Power Profile back to default
-resetpoweroverdrive	Set the maximum GPU power back to the device default state
-resetxgmierr	Reset XGMI error count
<b>Auto-response Options</b>	
-autorespond RESPONSE	Response to automatically provide for all prompts (NOT RECOMMENDED)
<b>Output Options</b>	
-loglevel LEVEL	How much output will be printed for what program is doing, one of debug, info, warn, error
-json	Print output in JSON format
-csv	Print output in CSV format

### Detailed Option Descriptions

**-setclk/-setmclk # [# # ...]:** This allows you to set a mask for the levels. For example, if a GPU has 8 clock levels, you can set a mask to use levels 0, 5, 6 and 7 with `-setclk 0 5 6 7`. This will only use the base level, and the top 3 clock levels. This will allow you to keep the GPU at base level when there is no GPU load, and the top 3 levels when the GPU load increases.

**–setfan LEVEL:** This sets the fan speed to a value ranging from 0 to 255 (not from 0-100%). If the level ends with a %, the fan speed is calculated as  $pct * maxlevel / 100$  (maxlevel is usually 255, but is determined by the ASIC) .. NOTE:

While the hardware **is** usually capable of overriding this value when required, it **is** recommended to **not** set the fan level lower than the default value **for** extended periods of time

**–setperflevel LEVEL:** This lets you use the pre-defined Performance Level values, which can include: auto (Automatically change PowerPlay values based on GPU workload) low (Keep PowerPlay values low, regardless of workload) high (Keep PowerPlay values high, regardless of workload) manual (Only use values defined in sysfs values)

**–setoverdrive/–setmemoverdrive #:** **DEPRECATED IN NEWER KERNEL VERSIONS (use –set-slevel/–setmlevel instead)** This sets the percentage above maximum for the max Performance Level. For example, –setoverdrive 20 will increase the top sclk level by 20%. If the maximum sclk level is 1000MHz, then –setoverdrive 20 will increase the maximum sclk to 1200MHz

**–setpoweroverdrive/–resetpoweroverdrive #:** This allows users to change the maximum power available to a GPU package. The input value is in Watts. This limit is enforced by the hardware, and some cards allow users to set it to a higher value than the default that ships with the GPU. This Power OverDrive mode allows the GPU to run at higher frequencies for longer periods of time, though this may mean the GPU uses more power than it is allowed to use per power supply specifications. Each GPU has a model-specific maximum Power OverDrive that it will take; attempting to set a higher limit than that will cause this command to fail.

**–setprofile SETPROFILE:** The Compute Profile accepts 1 or n parameters, either the Profile to select (see –show-profile for a list of preset Power Profiles) or a quoted string of values for the CUSTOM profile.

NOTE: These values can vary based on the ASIC, and may include: SCLK\_PROFILE\_ENABLE - Whether or not to apply the 3 following SCLK settings (0=disable,1=enable)

NOTE: This is a hidden field. If set to 0, the following 3 values are displayed as ‘-‘ SCLK\_UP\_HYST

- Delay before sclk is increased (in milliseconds) SCLK\_DOWN\_HYST
- Delay before sclk is decreased (in milliseconds) SCLK\_ACTIVE\_LEVEL
- Workload required before sclk levels change (in %) MCLK\_PROFILE\_ENABLE
- Whether or not to apply the 3 following MCLK settings (0=disable,1=enable)

NOTE: This is a hidden field. If set to 0, the following 3 values are displayed as ‘-‘ MCLK\_UP\_HYST

- Delay before mclk is increased (in milliseconds) MCLK\_DOWN\_HYST
- Delay before mclk is decreased (in milliseconds) MCLK\_ACTIVE\_LEVEL
- Workload required before mclk levels change (in %)

BUSY\_SET\_POINT - Threshold for raw activity level before levels change  
 FPS - Frames Per Second  
 USE\_RLC\_BUSY - When set to 1, DPM is switched up as long as RLC busy message is received  
 MIN\_ACTIVE\_LEVEL - Workload required before levels change (in %)

---

**Note:** When a compute queue is detected, these values will be automatically applied to the system Compute Power Profiles are only applied when the Performance Level is set to “auto”

The CUSTOM Power Profile is only applied when the Performance Level is set to “manual” so using this flag will automatically set the performance level to “manual”

It is not possible to modify the non-CUSTOM Profiles. These are hard-coded by the kernel

---

**-P, –showpower:** Show Average Graphics Package power consumption

“Graphics Package” refers to the GPU plus any HBM (High-Bandwidth memory) modules, if present

**-M, --showmaxpower:** Show the maximum Graphics Package power that the GPU will attempt to consume. This limit is enforced by the hardware.

**--loglevel:** This will allow the user to set a logging level for the SMI's actions. Currently this is only implemented for sysfs writes, but can easily be expanded upon in the future to log other things from the SMI

**--showmeminfo:** This allows the user to see the amount of used and total memory for a given block (vram, vis\_vram, gtt). It returns the number of bytes used and total number of bytes for each block 'all' can be passed as a field to return all blocks, otherwise a quoted-string is used for multiple values (e.g. "vram vis\_vram") vram refers to the Video RAM, or graphics memory, on the specified device vis\_vram refers to Visible VRAM, which is the CPU-accessible video memory on the device gtt refers to the Graphics Translation Table

**-b, --showbw:** This shows an approximation of the number of bytes received and sent by the GPU over the last second through the PCIe bus. Note that this will not work for APUs since data for the GPU portion of the APU goes through the memory fabric and does not 'enter/exit' the chip via the PCIe interface, thus no accesses are generated, and the performance counters can't count accesses that are not generated. NOTE: It is not possible to easily grab the size of every packet that is transmitted in real time, so the kernel estimates the bandwidth by taking the maximum payload size (mps), which is the max size that a PCIe packet can be. and multiplies it by the number of packets received and sent. This means that the SMI will report the maximum estimated bandwidth, the actual usage could (and likely will be) less

**--showrasinfo:** This shows the RAS information for a given block. This includes enablement of the block (currently GFX, SDMA and UMC are the only supported blocks) and the number of errors ue - Uncorrectable errors ce - Correctable errors

### Clock Type Descriptions

DCEFCLK - DCE (Display) FCLK - Data fabric (VG20 and later) - Data flow from XGMI, Memory, PCIe SCLK - GFXCLK (Graphics core)

---

**Note:** SOCCLK split from SCLK as of Vega10. Pre-Vega10 they were both controlled by SCLK

---

MCLK - GPU Memory (VRAM) PCLK - PCIe bus

---

**Note:** This gives 2 speeds, PCIe Gen1 x1 and the highest available based on the hardware

---

SOCCLK - System clock (VG10 and later)- Data Fabric (DF), MM HUB, AT HUB, SYSTEM HUB, OSS, DFD Note - DF split from SOCCLK as of Vega20. Pre-Vega20 they were both controlled by SOCCLK

**--gpureset:** This flag will attempt to reset the GPU for a specified device. This will invoke the GPU reset through the kernel debugfs file amdgpu\_gpu\_recover. Note that GPU reset will not always work, depending on the manner in which the GPU is hung.

**--showdriverversion:** This flag will print out the AMDGPU module version for amdgpu-pro or ROCK kernels. For other kernels, it will simply print out the name of the kernel (uname)

**--showserial:** This flag will print out the serial number for the graphics card NOTE: This is currently only supported on Vega20 server cards that support it. Consumer cards and cards older than Vega20 will not support this feature.

**--showproductname:** This uses the pci.ids file to print out more information regarding the GPUs on the system. 'update-pciids' may need to be executed on the machine to get the latest PCI ID snapshot, as certain newer GPUs will not be present in the stock pci.ids file, and the file may even be absent on certain OS installation types

**--showpagesinfo | --showretiredpages | --showpendingpages | --showunreservablepages:** These flags display the different "bad pages" as reported by the kernel. The three types of pages are: Retired pages (reserved pages) - These pages are reserved and are unable to be used Pending pages - These pages are pending for reservation, and will be reserved/retired Unreservable pages - These pages are not reservable for some reason.

**–showmemuse | –showuse | –showmeminfo –showuse and –showmemuse** are used to indicate how busy the respective blocks are. For example, for **–showuse** (`gpu_busy_percent` sysfs file), the SMU samples every ms or so to see if any GPU block (RLC, MEC, PFP, CP) is busy. If so, that's 1 (or high). If not, that's 0 (low). If we have 5 high and 5 low samples, that means 50% utilization (50% GPU busy, or 50% GPU use). The windows and sampling vary from generation to generation, but that is how GPU and VRAM use is calculated in a generic sense. **–showmeminfo** (and **VRAM%** in concise output) will show the amount of VRAM used (visible, total, GTT), as well as the total available for those partitions. The percentage shown there indicates the amount of used memory in terms of current allocations

#### OverDrive settings

- Enabling OverDrive requires both a card that support OverDrive and a driver parameter that enables its use.
- Because OverDrive features can damage your card, most workstation and server GPUs cannot use OverDrive.
- Consumer GPUs that can use OverDrive must enable this feature by setting bit 14 in the `amdgpu` driver's `ppfeaturemask` module parameter

For OverDrive functionality, the OverDrive bit (bit 14) must be enabled (by default, the OverDrive bit is disabled on the ROCK and upstream kernels). This can be done by setting `amdgpu.ppfeaturemask` accordingly in the kernel parameters, or by changing the default value inside `amdgpu_drv.c` (if building your own kernel).

As an example, if the `ppfeaturemask` is set to `0xffffbfff` (`111111111111111101111111111111`), then enabling the OverDrive bit would make it `0xffffffff` (`111111111111111111111111111111`). These are the flags that require OverDrive functionality to be enabled for the flag to work:

```
--showclkvolt
--showvoltage
--showvc
--showscclrange
--showmclrange
--setslevel
--setmlevel
--setoverdrive
--setpoweroverdrive
--resetpoweroverdrive
--setvc
--setsrange
--setmrange
```

#### Testing changes

After making changes to the SMI, run the test script to ensure that all functionality remains intact before uploading the patch. This can be done using:

```
./test-rocm-smi.sh /opt/rocm/bin/rocm-smi
```

The test can run all flags for the SMI, or specific flags can be tested with the `-s` option.

Any new functionality added to the SMI should have a corresponding test added to the test script.

### 3.35.2 SDMA Usage Per-process

The SDMA usage per-process is available using the following command,

```
$ rocm-smi -showpids
```

### 3.35.3 Hardware Topology

This feature provides a matrix representation of the GPUs present in a system by providing information of the manner in which the nodes are connected.

Weight Between Two GPUs			
	GPU0	GPU1	GPU2
GPU0	0	72	72
GPU1	72	0	40
GPU2	72	40	0

Hops Between Two GPUs			
	GPU0	GPU1	GPU2
GPU0	0	3	3
GPU1	3	0	2
GPU2	3	2	0

This is represented in terms of weights, hops, and link types between two given GPUs. It also provides the numa node and the CPU affinity associated with every GPU.

Link Type Between Two GPUs			
	GPU0	GPU1	GPU2
GPU0	0	PCIE	PCIE
GPU1	PCIE	0	PCIE
GPU2	PCIE	PCIE	0

Numa Nodes	
GPU[0]	: (Topology) Numa Node: 1
GPU[0]	: (Topology) Numa Affinity: 1
GPU[1]	: (Topology) Numa Node: 0
GPU[1]	: (Topology) Numa Affinity: 0
GPU[2]	: (Topology) Numa Node: 0
GPU[2]	: (Topology) Numa Affinity: 0

For more information about ROCm SMI API libraries, refer to the ROCm SMI API Guide at [https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm\\_SMI\\_API\\_Guide\\_v3.10.pdf](https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm_SMI_API_Guide_v3.10.pdf)

## 3.36 GCN ISA Manuals

### 3.36.1 GCN 1.1

ISA Manual for Hawaii [pdf](#)

### 3.36.2 GCN 2.0

ISA Manual for Fiji and Polaris [pdf](#)

### 3.36.3 Vega

- [testdocbook](#)

### 3.36.4 Inline GCN ISA Assembly Guide

#### 3.36.4.1 The Art of AMDGCN Assembly: How to Bend the Machine to Your Will

The ability to write code in assembly is essential to achieving the best performance for a GPU program. We have previously described how to combine several languages in a single program using ROCm and Hsaco. This article explains how to produce Hsaco from assembly code and also takes a closer look at some new features of the GCN architecture. I'd like to thank Ilya Perminov of Luxsoft for co-authoring this blog post. Programs written for GPUs should achieve the highest performance possible. Even carefully written ones, however, won't always employ 100% of the GPU's capabilities. Some reasons are the following:

- The program may be written in a high level language that does not expose all of the features available on the hardware.
- The compiler is unable to produce optimal ISA code, either because the compiler needs to 'play it safe' while adhering to the semantics of a language or because the compiler itself is generating un-optimized code.

Consider a program that uses one of GCN's new features (source code is available on [GitHub](#)). Recent hardware architecture updates—DPP and DS Permute instructions—enable efficient data sharing between wavefront lanes. To become more familiar with the instruction set, review the [GCN ISA Reference Guide](#). Note: the assembler is currently experimental; some of syntax we describe may change.

#### 3.36.4.2 DS Permute Instructions

Two new instructions, `ds_permute_b32` and `ds_bpermute_b32`, allow VGPR data to move between lanes on the basis of an index from another VGPR. These instructions use LDS hardware to route data between the 64 lanes, but they don't write to LDS memory. The difference between them is what to index: the source-lane ID or the destination-lane ID. In other words, `ds_permute_b32` says "put my lane data in lane i," and `ds_bpermute_b32` says "read data from lane i." The GCN ISA Reference Guide provides a more formal description. The test kernel is simple: read the initial data and indices from memory into GPRs, do the permutation in the GPRs and write the data back to memory. An analogous OpenCL kernel would have this form:

```
__kernel void hello_world(__global const uint * in, __global const uint * index, __
↪global uint * out)
{
    size_t i = get_global_id(0);
```

(continues on next page)

(continued from previous page)

```

    out[i] = in[ index[i] ];
}

```

### 3.36.4.3 Passing Parameters to a Kernel

Formal HSA arguments are passed to a kernel using a special read-only memory segment called kernarg. Before a wavefront starts, the base address of the kernarg segment is written to an SGPR pair. The memory layout of variables in kernarg must employ the same order as the list of kernel formal arguments, starting at offset 0, with no padding between variables—except to honor the requirements of natural alignment and any align qualifier. The example host program must create the kernarg segment and fill it with the buffer base addresses. The HSA host code might look like the following:

```

/*
 * This is the host-side representation of the kernel arguments that the simplePermute_
 * ↪kernel expects.
 */
struct simplePermute_args_t {
    uint32_t * in;
    uint32_t * index;
    uint32_t * out;
};
/*
 * Allocate the kernel-argument buffer from the correct region.
 */
hsa_status_t status;
simplePermute_args_t * args = NULL;
status = hsa_memory_allocate(kernarg_region, sizeof(simplePermute_args_t), (void**)&
 * ↪args));
assert(HSA_STATUS_SUCCESS == status);
aql->kernarg_address = args;
/*
 * Write the args directly to the kernargs buffer;
 * the code assumes that memory is already allocated for the
 * buffers that in_ptr, index_ptr and out_ptr point to
 */
args->in = in_ptr;
args->index = index_ptr;
args->out = out_ptr;

```

The host program should also allocate memory for the in, index and out buffers. In the GitHub repository, all the run-time-related stuff is hidden in the Dispatch and Buffer classes, so the sample code looks much cleaner:

```

// Create Kernarg segment
if (!AllocateKernarg(3 * sizeof(void*))) { return false; }

// Create buffers
Buffer *in, *index, *out;
in = AllocateBuffer(size);
index = AllocateBuffer(size);
out = AllocateBuffer(size);

// Fill Kernarg memory
Kernarg(in); // Add base pointer to "in" buffer
Kernarg(index); // Append base pointer to "index" buffer
Kernarg(out); // Append base pointer to "out" buffer

```

Initial Wavefront and Register State To launch a kernel in real hardware, the run time needs information about the kernel, such as

- The LDS size
- The number of GPRs
- Which registers need initialization before the kernel starts

All this data resides in the `amd_kernel_code_t` structure. A full description of the structure is available in the [AMDGPU-ABI](#) specification. This is what it looks like in source code:

```
.hsa_code_object_version 2,0
.hsa_code_object_isa 8, 0, 3, "AMD", "AMDGPU"

.text
.p2align 8
.amdgpu_hsa_kernel hello_world

hello_world:

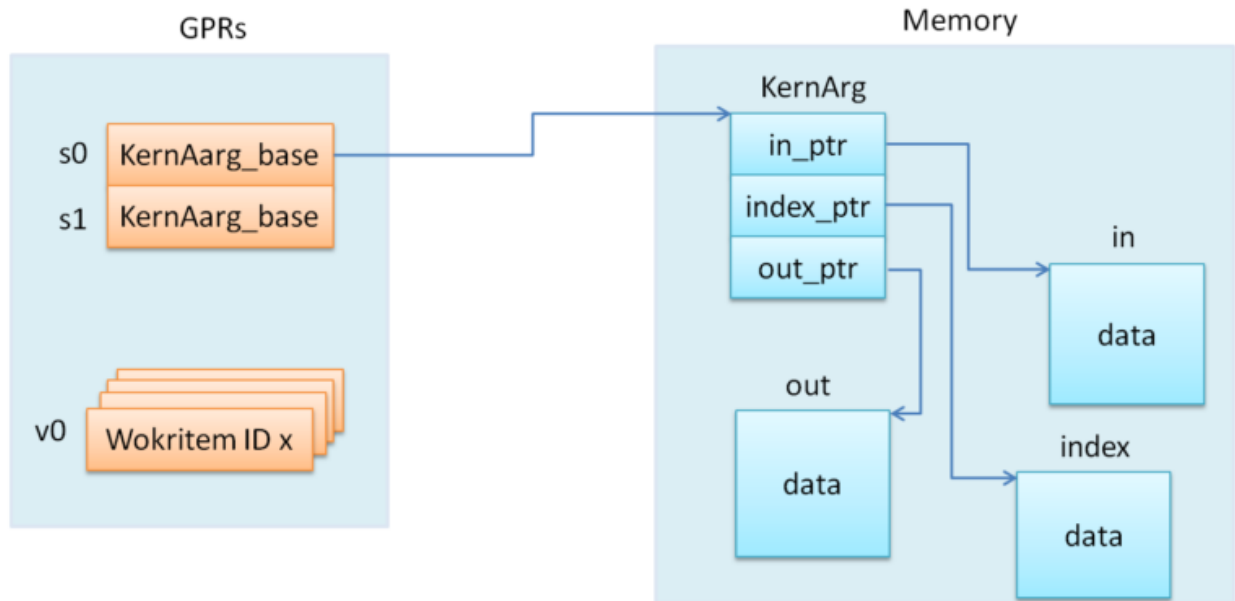
.amd_kernel_code_t
enable_sgpr_kernarg_segment_ptr = 1
is_ptr64 = 1
compute_pgm_rsrc1_vgprs = 1
compute_pgm_rsrc1_sgprs = 0
compute_pgm_rsrc2_user_sgpr = 2
kernarg_segment_byte_size = 24
wavefront_sgpr_count = 8
workitem_vgpr_count = 5
.end_amd_kernel_code_t

s_load_dwordx2  s[4:5], s[0:1], 0x10
s_load_dwordx4  s[0:3], s[0:1], 0x00
v_lshlrev_b32  v0, 2, v0
s_waitcnt      lgkmcnt(0)
v_add_u32      v1, vcc, s2, v0
v_mov_b32      v2, s3
v_addc_u32     v2, vcc, v2, 0, vcc
v_add_u32      v3, vcc, s0, v0
v_mov_b32      v4, s1
v_addc_u32     v4, vcc, v4, 0, vcc
flat_load_dword v1, v[1:2]
flat_load_dword v2, v[3:4]
s_waitcnt      vmcnt(0) & lgkmcnt(0)
v_lshlrev_b32  v1, 2, v1
ds_bpermute_b32 v1, v1, v2
v_add_u32      v3, vcc, s4, v0
v_mov_b32      v2, s5
v_addc_u32     v4, vcc, v2, 0, vcc
s_waitcnt      lgkmcnt(0)
flat_store_dword v[3:4], v1
s_endpgm
```

Currently, a programmer must manually set all non-default values to provide the necessary information. Hopefully, this situation will change with new updates that bring automatic register counting and possibly a new syntax to fill that structure. Before the start of every wavefront execution, the GPU sets up the register state on the basis of the `enable_sgpr_*` and `enable_vgpr_*` flags. VGPR `v0` is always initialized with a work-item ID in the x dimension. Registers `v1` and `v2` can be initialized with work-item IDs in the y and z dimensions, respectively. Scalar GPRs can be initialized with a work-group ID and work-group count in each dimension, a dispatch ID, and pointers to



kernarg, the aql packet, the aql queue, and so on. Again, the AMDGPU-ABI specification contains a full list in the section on initial register state. For this example, a 64-bit base kernarg address will be stored in the s[0:1] registers (enable\_sgpr\_kernarg\_segment\_ptr = 1), and the work-item thread ID will occupy v0 (by default). Below is the scheme showing initial state for our kernel.



#### 3.36.4.4 The GPR Counting

The next `amd_kernel_code_t` fields are obvious: `is_ptr64 = 1` says we are in 64-bit mode, and `kernarg_segment_byte_size = 24` describes the kernarg segment size. The GPR counting is less straightforward, however. The `workitem_vgpr_count` holds the number of vector registers that each work item uses, and `wavefront_sgpr_count` holds the number of scalar registers that a wavefront uses. The code above employs v0–v4, so `workitem_vgpr_count = 5`. But `wavefront_sgpr_count = 8` even though the code only shows s0–s5, since the special registers VCC, FLAT\_SCRATCH and XNACK are physically stored as part of the wavefront’s SGPRs in the highest-numbered SGPRs. In this example, FLAT\_SCRATCH and XNACK are disabled, so VCC has only two additional registers. In current GCN3 hardware, VGPRs are allocated in groups of 4 registers and SGPRs in groups of 16. Previous generations (GCN1 and GCN2) have a VGPR granularity of 4 registers and an SGPR granularity of 8 registers. The fields `compute_pgm_rsrc1_*gprs` contain a device-specific number for each register-block type to allocate for a wavefront. As we said previously, future updates may enable automatic counting, but for now you can use following formulas for all three GCN GPU generations:

```
compute_pgm_rsrc1_vgprs = (workitem_vgpr_count-1)/4
compute_pgm_rsrc1_sgprs = (wavefront_sgpr_count-1)/8
```

Now consider the corresponding assembly:

```
// initial state:
//   s[0:1] - kernarg base address
//   v0 - workitem id

s_load_dwordx2 s[4:5], s[0:1], 0x10 // load out_ptr into s[4:5] from kernarg
s_load_dwordx4 s[0:3], s[0:1], 0x00 // load in_ptr into s[0:1] and index_ptr into
↪ s[2:3] from kernarg
```

(continues on next page)

(continued from previous page)

```

v_lshlrev_b32 v0, 2, v0          // v0 *= 4;
s_waitcnt    lgkmcnt(0)        // wait for memory reads to finish

// compute address of corresponding element of index buffer
// i.e. v[1:2] = &index[workitem_id]
v_add_u32     v1, vcc, s2, v0
v_mov_b32     v2, s3
v_addc_u32    v2, vcc, v2, 0, vcc

// compute address of corresponding element of in buffer
// i.e. v[3:4] = &in[workitem_id]
v_add_u32     v3, vcc, s0, v0
v_mov_b32     v4, s1
v_addc_u32    v4, vcc, v4, 0, vcc

flat_load_dword v1, v[1:2] // load index[workitem_id] into v1
flat_load_dword v2, v[3:4] // load in[workitem_id] into v2
s_waitcnt     vmcnt(0) & lgkmcnt(0) // wait for memory reads to finish

// v1 *= 4; ds_bpermute_b32 uses byte offset and registers are dwords
v_lshlrev_b32 v1, 2, v1

// perform permutation
// temp[thread_id] = v2
// v1 = temp[v1]
// effectively we got v1 = in[index[thread_id]]
ds_bpermute_b32 v1, v1, v2

// compute address of corresponding element of out buffer
// i.e. v[3:4] = &out[workitem_id]
v_add_u32     v3, vcc, s4, v0
v_mov_b32     v2, s5
v_addc_u32    v4, vcc, v2, 0, vcc

s_waitcnt     lgkmcnt(0) // wait for permutation to finish

// store final value in out buffer, i.e. out[workitem_id] = v1
flat_store_dword v[3:4], v1

s_endpgm

```

### 3.36.4.5 Compiling GCN ASM Kernel Into Hsaco

The next step is to produce a Hsaco from the ASM source. LLVM has added support for the AMDGCN assembler, so you can use Clang to do all the necessary magic:

```

clang -x assembler -target amdgcnc--amdhsa -mcpu=fiji -c -o test.o asm_source.s

clang -target amdgcnc--amdhsa test.o -o test.co

```

The first command assembles an object file from the assembly source, and the second one links everything (you could have multiple source files) into a Hsaco. Now, you can load and run kernels from that Hsaco in a program. The [GitHub examples](#) use Cmake to automatically compile ASM sources. In a future post we will cover DPP, another GCN cross-lane feature that allows vector instructions to grab operands from a neighboring lane.

## 3.37 Remote Device Programming

### 3.37.1 ROCmRDMA

#### Peer-to-Peer bridge driver for PeerDirect - Deprecated Repo

This is now included as part of the ROCK [Kernel Driver](#) ROCmRDMA is the solution designed to allow third-party kernel drivers to utilize DMA access to the GPU memory. It allows direct path for data exchange (peer-to-peer) using the standard features of PCI Express.

Currently ROCmRDMA provides the following benefits:

- Direct access to ROCm memory for 3rd party PCIe devices
- Support for PeerDirect(c) interface to offloads the CPU when dealing with ROCm memory for RDMA network stacks;

#### 3.37.1.1 Restrictions and limitations

To fully utilize ROCmRDMA the number of limitation could apply impacting either performance or functionality in the whole:

- It is recommended that devices utilizing ROCmRDMA share the same upstream PCI Express root complex. Such limitation depends on PCIe chipset manufactures and outside of GPU controls;
- To provide peer-to-peer DMA access all GPU local memory must be exposed via PCI memory BARs (so called large-BAR configuration);
- It is recommended to have IOMMU support disabled or configured in pass-through mode due to limitation in Linux kernel to support local PCIe device memory for any form transition others then 1:1 mapping.

#### 3.37.1.2 ROCmRDMA interface specification

The implementation of ROCmRDMA interface can be found in [\[amd\\_rdma.h\]](#) file.

#### 3.37.1.3 API versions

ROCm up to and including v4.1 supported RDMA version 1.0.

ROCm 4.2 has enhanced the API version to 2.0, introduced the following definitions to allow users to detect the API version, and apply conditional compilation as needed:

```
/* API versions:
 * 1.0 Original API until ROCm 4.1, AMD_RDMA_MAJOR/MINOR undefined
 * 2.0 Added IOMMU (dma-mapping) support, removed p2p_info.kfd_proc
 *     Introduced AMD_RDMA_MAJOR/MINOR version definition
 */
#define AMD_RDMA_MAJOR 2
#define AMD_RDMA_MINOR 0
```

### 3.37.1.4 Data structures

```
/**
 * Structure describing information needed to P2P access from another device
 * to specific location of GPU memory
 */
struct amd_p2p_info {
    uint64_t      va;                /**< Specify user virt. address
                                     * which this page table
                                     * described
                                     */
    uint64_t      size;             /**< Specify total size of
                                     * allocation
                                     */
    struct pid     *pid;            /**< Specify process pid to which
                                     * virtual address belongs
                                     */
    struct sg_table *pages;         /**< Specify DMA/Bus addresses */
    void          *priv;            /**< Pointer set by AMD kernel
                                     * driver
                                     */
};
```

```
/**
 * Structure providing function pointers to support rdma/p2p requirements.
 * to specific location of GPU memory
 */
struct amd_rdma_interface {
    int (*get_pages)(uint64_t address, uint64_t length, struct pid *pid,
                    struct device *dma_dev,
                    struct amd_p2p_info **amd_p2p_data,
                    void (*free_callback)(void *client_priv),
                    void *client_priv);
    int (*put_pages)(struct amd_p2p_info **amd_p2p_data);
    int (*is_gpu_address)(uint64_t address, struct pid *pid);
    int (*get_page_size)(uint64_t address, uint64_t length, struct pid *pid,
                        unsigned long *page_size);
};
```

### 3.37.1.5 The function to query ROCmRDMA interface

```
/**
 * amdkfd_query_rdma_interface - Return interface (function pointers table) for
 *                               rdma interface
 *
 * \param interace      - OUT: Pointer to interface
 * \return 0 if operation was successful.
 */
int amdkfd_query_rdma_interface(const struct amd_rdma_interface **rdma);
```

## 3.37.1.6 ROCmRDMA interface functions description

```

/**
 * This function makes the pages underlying a range of GPU virtual memory
 * accessible for DMA operations from another PCIe device
 *
 * \param address - The start address in the Unified Virtual Address
 *                  space in the specified process
 * \param length - The length of requested mapping
 * \param pid     - Pointer to structure pid to which address belongs.
 *                  Could be NULL for current process address space.
 * \param dma_dev - Device that will need a DMA mapping of the memory
 * \param amd_p2p_data - On return: Pointer to structure describing
 *                       underlying pages/locations
 * \param free_callback - Pointer to callback which will be called when access
 *                       to such memory must be stopped immediately: Memory
 *                       was freed, GECC events, etc.
 *                       Client should immediately stop any transfer
 *                       operations and returned as soon as possible.
 *                       After return all resources associated with address
 *                       will be release and no access will be allowed.
 * \param client_priv - Pointer to be passed as parameter on
 *                       'free_callback;
 *
 * \return 0 if operation was successful
 */
int get_pages(uint64_t address, uint64_t length, struct pid *pid,
              struct device *dma_dev, struct amd_p2p_info **amd_p2p_data,
              void (*free_callback)(void *client_priv),
              void *client_priv);

```

```

/**
 * This function release resources previously allocated by get_pages() call.
 * \param p2p_data - A pointer to pointer to amd_p2p_info entries
 *                  allocated by get_pages() call.
 * \return 0 if operation was successful
 */
int put_pages(struct amd_p2p_info **p2p_data)

```

```

/**
 * Check if given address belongs to GPU address space.
 * \param address - Address to check
 * \param pid     - Process to which given address belongs.
 *                  Could be NULL if current one.
 * \return 0       - This is not GPU address managed by AMD driver
 *       1       - This is GPU address managed by AMD driver
 */
int is_gpu_address(uint64_t address, struct pid *pid);

```

```

/**
 * Return the single page size to be used when building scatter/gather table
 * for given range.
 * \param address - Address
 * \param length  - Range length
 * \param pid     - Process id structure. Could be NULL if current one.
 * \param page_size - On return: Page size

```

(continues on next page)

(continued from previous page)

```
* \return 0 if operation was successful
*/
int get_page_size(uint64_t address, uint64_t length, struct pid *pid,
                  unsigned long *page_size);
```

### 3.37.2 UCX

#### What is UCX ?

Unified Communication X (UCX) is a communication library for building Message Passing (MPI), PGAS/OpenSHMEM libraries and RPC/data-centric applications. UCX utilizes high-speed networks for inter-node and shared memory mechanisms for intra-node communication. For more information, visit <http://openucx.github.io/ucx/>

#### How to install UCX with ROCm ?

See [How to install UCX and OpenMPI](#)

#### How to enable ROCm transport during configuration and runtime

Access the following links to enable ROCm transport during configuration and runtime:

- For release builds: `./contrib/configure-release --prefix=/path/to/install --with-rocm=/path/to/rocm`
- For debug builds: `./contrib/configure-devel --prefix=/path/to/install --with-rocm=/path/to/rocm`

### 3.37.3 OpenMPI

#### OpenMPI and OpenSHMEM installation

1. Get latest-and-gratest OpenMPI version:

```
$ git clone https://github.com/open-mpi/ompi.git
```

2. Autogen:

```
$ cd ompi
$ ./autogen.pl
```

3. Configure with UCX

```
$ mkdir build
$ cd build
$ ./configure --prefix=/your_install_path/ --with-ucx=/path_to_ucx_installation
```

4. Build:

```
$ make
$ make install
```

#### Running Open MPI with UCX

Example of the command line (for InfiniBand RC + shared memory):

```
$ mpirun -np 2 -mca pml ucx -x UCX_NET_DEVICES=mlx5_0:1 -x UCX_TLS=rc,sm ./app
```

#### Open MPI runtime optimizations for UCX

- By default OpenMPI enables build-in transports (BTLs), which may result in additional software overheads in the OpenMPI progress function. In order to workaround this issue you may try to disable certain BTLs.

```
$ mpirun -np 2 -mca pml ucx --mca btl ^vader,tcp,openib -x UCX_NET_DEVICES=mlx5_0:1 -
↪x UCX_TLS=rc,sm ./app
```

- OpenMPI version <https://github.com/open-mpi/ompi/commit/066370202dcad8e302f2baf8921e9efd0f1f7dfc> leverages more efficient timer mechanism and there fore reduces software overheads in OpenMPI progress

### MPI and OpenSHMEM release versions tested with UCX master

1. UCX current tarball: <https://github.com/openucx/ucx/archive/master.zip>
2. The table of MPI and OpenSHMEM distributions that are tested with the HEAD of UCX master

MPI/OpenSHMEM	project
OpenMPI/OSHMEM	2.1.0
MPICH	Latest

## 3.37.4 IPC API

### 3.37.4.1 New Datatypes

```
hsa_amd_ipc_memory_handle_t

/** IPC memory handle to by passed from one process to another */
typedef struct hsa_amd_ipc_memory_handle_s {
    uint64_t handle;
} hsa_amd_ipc_memory_handle_t;

hsa_amd_ipc_signal_handle_t

/** IPC signal handle to by passed from one process to another */
typedef struct hsa_amd_ipc_signal_handle_s {
    uint64_t handle;
} hsa_amd_ipc_signal_handle_t;
```

### Memory sharing API

Allows sharing of HSA allocated memory between different processes.

`hsa_amd_ipc_get_memory_handle`

The purpose of this API is to get / export an IPC handle for an existing allocation from pool.

### `hsa_status_t` HSA\_API

`hsa_amd_ipc_get_memory_handle(void *ptr, hsa_amd_ipc_memory_handle_t *ipc_handle);`

where:

IN: ptr - Pointer to memory previously allocated via `hsa_amd_memory_pool_allocate()` call

OUT: ipc\_handle - Unique IPC handle to be used in IPC.

Application must pass this handle to another process.

hsa\_amd\_ipc\_close\_memory\_handle

Close IPC memory handle previously received via “hsa\_amd\_ipc\_get\_memory\_handle()” call .

#### **hsa\_status\_t HSA\_API**

hsa\_amd\_ipc\_close\_memory\_handle(hsa\_amd\_ipc\_memory\_handle\_t ipc\_handle);

where:

IN: ipc\_handle - IPC Handle to close

hsa\_amd\_ipc\_open\_memory\_handle

Open / import an IPC memory handle exported from another process and return address to be used in the current process.

#### **hsa\_status\_t HSA\_API**

hsa\_amd\_ipc\_open\_memory\_handle(hsa\_amd\_ipc\_memory\_handle\_t ipc\_handle, void \*\*ptr);

where:

IN: ipc\_handle - IPC Handle

OUT: ptr- Address which could be used in the given process for access to the memory

Client should call hsa\_amd\_memory\_pool\_free() when access to this resource is not needed any more.

### **Signal sharing API**

Allows sharing of HSA signals between different processes.

hsa\_amd\_ipc\_get\_signal\_handle

The purpose of this API is to get / export an IPC handle for an existing signal.

#### **hsa\_status\_t HSA\_API**

hsa\_amd\_ipc\_get\_signal\_handle(hsa\_signal\_t signal, hsa\_amd\_ipc\_signal\_handle\_t \*ipc\_handle);

where:

IN: signal - Signal handle created as the result of hsa\_signal\_create() call.

OUT: ipc\_handle - Unique IPC handle to be used in IPC.

Application must pass this handle to another process.

hsa\_amd\_ipc\_close\_signal\_handle

Close IPC signal handle previously received via “hsa\_amd\_ipc\_get\_signal\_handle()” call .

#### **hsa\_status\_t HSA\_API**



```
hsa_amd_ipc_close_signal_handle(hsa_amd_ipc_signal_handle_t ipc_handle);
```

where:

IN: ipc\_handle - IPC Handle to close

```
hsa_amd_ipc_open_signal_handle
```

Open / import an IPC signal handle exported from another process and return address to be used in the current process.

### hsa\_status\_t HSA\_API

```
hsa_amd_ipc_open_signal_handle(hsa_amd_ipc_signal_handle_t ipc_handle, hsa_signal_t &signal);
```

where:

IN: ipc\_handle - IPC Handle

OUT: signal - Signal handle to be used in the current process

Client should call hsa\_signal\_destroy() when access to this resource is not needed any more.

### Query API

Query memory information

Allows query information about memory resource based on address. It is partially overlapped with the following requirement Memory info interface so it may be possible to merge those two interfaces.

```
typedef enum hsa_amd_address_info_s {

    /* Return uint32_t / boolean if address was allocated via HSA stack */
    HSA_AMD_ADDRESS_HSA_ALLOCATED = 0x1,

    /** Return agent where such memory was allocated */
    HSA_AMD_ADDRESS_AGENT = 0x2,

    /** Return pool from which this address was allocated */
    HSA_AMD_ADDRESS_POOL = 0x3,

    /** Return size of allocation */
    HSA_AMD_ADDRESS_ALLOC_SIZE = 0x4

} hsa_amd_address_info_t;
```

### hsa\_status\_t HSA\_API

```
hsa_amd_get_address_info(void ptr, hsa_amd_address_info_t attribute, void value);
```

where:

ptr - Address information about which to query

attribute - Attribute to query

### 3.37.5 MPICH

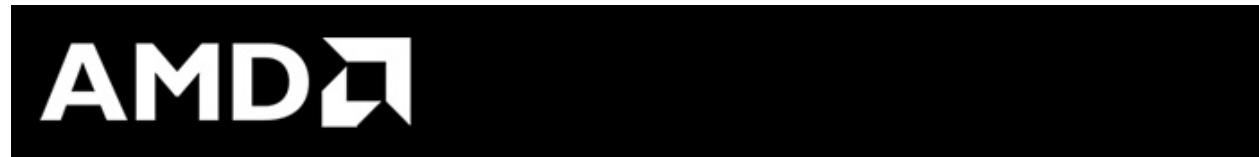
MPICH is a high-performance and widely portable implementation of the MPI-3.1 standard.

For more information about MPICH, refer to <https://www.mpich.org/>

#### 3.37.5.1 Building and Installing MPICH

To build and install MPICH with UCX and ROCm support, see the instructions below.

```
git clone https://github.com/pmodels/mpich.git
cd mpich
git checkout v3.4
git submodule update --init --recursive
./autogen.sh
./configure --prefix=/mpich/install/location --with-device=ch4:ucx --with-ucx=/ucx/
install/location>
make -j && make install
```



## 3.38 v4.1 ROCm Installation

- *Deploying ROCm*
- *Prerequisites*
- *Supported Operating Systems*
  - *Ubuntu*
  - *CentOS RHEL*
  - *SLES 15 Service Pack 2*
- *ROCm Installation Known Issues and Workarounds*
- *Getting the ROCm Source Code*

### 3.38.1 Deploying ROCm

AMD hosts both Debian and RPM repositories for the ROCm v4.x packages.

The following directions show how to install ROCm on supported Debian-based systems such as Ubuntu 18.04.x

**Note:** These directions may not work as written on unsupported Debian-based distributions. For example, newer versions of Ubuntu may not be compatible with the rock-dkms kernel driver. In this case, you can exclude the rocm-dkms and rock-dkms packages.

**Note:** You must use either ROCm or the amdgpu-pro driver. Using both drivers will result in an installation error.

**Important - Mellanox ConnectX NIC Users:** If you are using Mellanox ConnectX NIC, you must install Mellanox OFED before installing ROCm.

For more information about installing Mellanox OFED, refer to:

<https://docs.mellanox.com/display/MLNXOFEDv461000/Installing+Mellanox+OFED>

#### 3.38.1.1 ROCm Repositories

- For major releases - <https://repo.radeon.com/rocm/yum/rpm/>
- For point releases - <https://repo.radeon.com/rocm/yum/4.1.x/>

#### 3.38.1.2 Base Operating System Kernel Upgrade

For SUSE, it is strongly recommended to follow the steps below when upgrading the base operating system kernel:

1. Remove *rock-dkms* before the upgrade.
2. Install the new kernel.
3. Reboot the system.
4. Reinstall *rock-dkms*.

Implementing these steps ensures correct loading of *amdgpu* and *amdkfd* after the kernel upgrade and prevents any issue caused by an incomplete DKMS upgrade. Fedora and Ubuntu do not have this restriction.

### 3.38.2 Prerequisites

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

**Note:** Ubuntu versions lower than 18 are no longer supported.

**Note:** AMD ROCm only supports Long Term Support (LTS) versions of Ubuntu. Versions other than LTS may work with ROCm, however, they are not officially supported.

### 3.38.2.1 Perl Modules for HIP-Base Package

The hip-base package has a dependency on Perl modules that some operating systems may not have in their default package repositories. Use the following commands to add repositories that have the required Perl packages:

- For SLES 15 SP2

```
sudo zypper addrepo
```

For more information, see

[https://download.opensuse.org/repositories/devel:languages:perl/SLE\\_15/devel:languages:perl.repo](https://download.opensuse.org/repositories/devel:languages:perl/SLE_15/devel:languages:perl.repo)

- For CentOS8.3

```
sudo yum config-manager --set-enabled powertools
```

- For RHEL8.3

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
```

### 3.38.2.2 Complete Reinstallation OF AMD ROCm V4.1 Recommended

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.

**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.

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

**Note:** After an operating system upgrade, AMD ROCm may upgrade automatically and result in an error. This is because AMD ROCm does not support upgrades currently. You must uninstall and reinstall AMD ROCm after an operating system upgrade.

**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/rocm/bin/clinfo`.
- For ROCm v3.3 and older releases, the *clinfo* path remains `/opt/rocm/rocm/bin/x86_64/clinfo`.

### 3.38.2.3 Multi-version Installation Updates

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 should 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.

### SETTING PERMISSIONS for GROUPS

**Note:** *render group* is required only for Ubuntu v20.04. For all other ROCm supported operating systems, continue to use *video group*. By default, you must add any future users to the video and render groups.

To add future users to the video and render groups, run the following command:

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

**\*\*Note\*\*:** Before updating to the latest version of the operating system, delete the `rocm` ROCm packages to avoid DKMS-related issues.

## 3.38.3 Supported Operating Systems

### 3.38.3.1 Ubuntu

**Note:** AMD ROCm only supports Long Term Support (LTS) versions of Ubuntu. Versions other than LTS may work with ROCm, however, they are not officially supported.

#### 3.38.3.1.1 Installing a ROCm Package from a Debian Repository

To install from a Debian Repository:

1. Run the following code to ensure that your system is up to date:

```
sudo apt update
sudo apt dist-upgrade
sudo apt install libnuma-dev
sudo reboot
```

2. Add the ROCm apt repository.

For Debian-based systems like Ubuntu, configure the Debian ROCm repository as follows:

**Note:** The public key has changed to reflect the new location. You must update to the new location as the old key will be removed in a future release.

- Old Key: <https://repo.radeon.com/rocm/apt/debian/rocm.gpg.key>
- New Key: <https://repo.radeon.com/rocm/rocm.gpg.key>

```
wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | sudo apt-key add -  
  
echo 'deb [arch=amd64] https://repo.radeon.com/rocm/apt/4.1/ xenial main' | sudo tee /  
→etc/apt/sources.list.d/rocm.list
```

**Note:** For developer systems or Docker containers (where it could be beneficial to use a fixed ROCm version), select a versioned repository from:

<https://repo.radeon.com/rocm/apt/>

The gpg key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm apt repository.

The current rocm.gpg.key is not available in a standard key ring distribution, but has the following sha1sum hash:

```
e85a40d1a43453fe37d63aa6899bc96e08f2817a rocm.gpg.key
```

3. Install the ROCm meta-package. Update the appropriate repository list and install the rocm-dkms meta-package:

```
sudo apt update  
  
sudo apt install rocm-dkms && sudo reboot
```

4. Restart the system.

5. After restarting the system, run the following commands to verify that the ROCm installation is successful. If you see your GPUs listed by both commands, the installation is considered successful.

```
/opt/rocm/bin/rocminfo  
/opt/rocm/ocl/bin/clinco
```

**Note:** To run the ROCm programs, add the ROCm binaries in your PATH.

```
echo 'export PATH=$PATH:/opt/rocm/bin:/opt/rocm/rocmprofiler/bin:/opt/rocm/ocl/bin' <br>  
→| sudo tee -a /etc/profile.d/rocm.sh
```

### 3.38.3.1.2 Uninstalling ROCm Packages from Ubuntu

To uninstall the ROCm packages from Ubuntu 20.04 or Ubuntu 18.04.5, run the following command:

```
sudo apt autoremove rocm-ocl rocm-dkms rocm-dev rocm-utils && sudo reboot
```

### 3.38.3.1.3 Using Debian-based ROCm with Upstream Kernel Drivers

You can install ROCm user-level software without installing AMD's custom ROCk kernel driver. The kernel used must have the *HSA kernel driver* option enabled and compiled into the *amdgpu* kernel driver. To install only ROCm user-level software, run the following commands instead of installing rocm-dkms:

```
sudo apt update  
sudo apt install rocm-dev  
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video" | sudo tee /etc/  
→udev/rules.d/70-kfd.rules
```

### 3.38.3.2 CentOS RHEL

This section describes how to install ROCm on supported RPM-based systems such as CentOS/RHEL.

#### 3.38.3.2.1 Preparing RHEL for Installation

RHEL is a subscription-based operating system. You must enable the external repositories to install on the devtoolset-7 environment and the dkms support files.

Note: The following steps do not apply to the CentOS installation.

1. The subscription for RHEL must be enabled and attached to a pool ID. See the Obtaining an RHEL image and license page for instructions on registering your system with the RHEL subscription server and attaching to a pool id.
2. Enable the following repositories for RHEL v7.x:

```
sudo subscription-manager repos --enable rhel-server-rhsc1-7-rpms
sudo subscription-manager repos --enable rhel-7-server-optional-rpms
sudo subscription-manager repos --enable rhel-7-server-extras-rpms
```

3. Enable additional repositories by downloading and installing the epel-release-latest-7/epel-release-latest-8 repository RPM:

```
sudo rpm -ivh <repo>
```

For more details,

- see <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm> for RHEL v7.x
- see <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm> for RHEL v8.x

4. Install and set up Devtoolset-7.

**Note:** Devtoolset is not required for CentOS/RHEL v8.x

To setup the Devtoolset-7 environment, follow the instructions on this page: <https://www.softwarecollections.org/en/scls/rhsc1/devtoolset-7/>

Note: devtoolset-7 is a software collections package and is not supported by AMD.

#### 3.38.3.2.1.1 Installing CentOS for DKMS

Use the dkms tool to install the kernel drivers on CentOS/RHEL:

```
sudo yum install -y epel-release
sudo yum install -y dkms kernel-headers-`uname -r` kernel-devel-`uname -r`
```

### 3.38.3.2.2 Installing ROCm

To install ROCm on your system, follow the instructions below:

1. Delete the previous versions of ROCm before installing the latest version.
2. Create a `/etc/yum.repos.d/rocm.repo` file with the following contents:
  - CentOS/RHEL 7.x : <https://repo.radeon.com/rocm/yum/rpm>
  - CentOS/RHEL 8.x : <https://repo.radeon.com/rocm/centos8/rpm>

```
[ROCM]
name=ROCM
baseurl=https://repo.radeon.com/rocm/yum/4.1/
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

**Note:** The URL of the repository must point to the location of the repositories' repodata database. For developer systems or Docker containers (where it could be beneficial to use a fixed ROCm version), select a versioned repository from:

<https://repo.radeon.com/rocm/yum/>

3. Install ROCm components using the following command:

```
sudo yum install rocm-dkms && sudo reboot
```

4. Restart the system. The rock-dkms component is installed and the `/dev/kfd` device is now available.
5. Restart the system.
6. Test the ROCm installation.

### 3.38.3.2.3 Testing the ROCm Installation

After restarting the system, run the following commands to verify that the ROCm installation is successful. If you see your GPUs listed, you are good to go!

```
/opt/rocm/bin/rocminfo
/opt/rocm/rocm-opencl/bin/clinfo
```

**Note:** Add the ROCm binaries in your PATH for easy implementation of the ROCm programs.

```
echo 'export PATH=$PATH:/opt/rocm/bin:/opt/rocm/profiler/bin:/opt/rocm/opencl/bin' | sudo tee -a /etc/profile.d/rocm.sh
```



### 3.38.3.2.4 Compiling Applications Using HCC, HIP, and Other ROCm Software

To compile applications or samples, run the following command to use gcc-7.2 provided by the devtoolset-7 environment:

```
scl enable devtoolset-7 bash
```

### 3.38.3.2.5 Uninstalling ROCm from CentOS/RHEL

To uninstall the ROCm packages, run the following command:

```
sudo yum autoremove rocm-openssl rocm-dkms rock-dkms
```

### 3.38.3.2.6 Using ROCm on CentOS/RHEL with Upstream Kernel Drivers

You can install ROCm user-level software without installing AMD's custom ROCk kernel driver. The kernel used must have the *HSA kernel driver* option enabled and compiled into the *amdgpu* kernel driver. To install only ROCm user-level software, run the following commands instead of installing rocm-dkms:

```
sudo yum install rocm-dev
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video"' | sudo tee /etc/
↳udev/rules.d/70-kfd.rules
sudo reboot
```

**Note:** Ensure you restart the system after ROCm installation.

### 3.38.3.2.7 Installing Development Packages for Cross Compilation

You can develop and test ROCm packages on different systems. For example, some development or build systems may not have an AMD GPU installed. In this scenario, you can avoid installing the ROCm kernel driver on your development system. Instead, install the following development subset of packages:

```
sudo yum install rocm-dev
```

**Note:** To execute ROCm-enabled applications, you will require a system installed with the full ROCm driver stack.

## 3.38.3.3 SLES 15 Service Pack 2

The following section tells you how to perform an install and uninstall ROCm on SLES 15 SP 2.

**Note:** For SUSE-based distributions (SLE, OpenSUSE, etc), upgrading the base kernel after installing ROCm may result in a broken installation. This is due to policies regarding unsupported kernel modules. To mitigate this, make the following change before initializing the amdgpu module:

```
#Allow Unsupported Driver and Load Driver
cat <<EOF | tee /etc/modprobe.d/10-unsupported-modules.conf
allow_unsupported_modules 1
EOF
```

For more information, refer to <https://www.suse.com/support/kb/doc/?id=000016939>

## Installation

1. Install the “dkms” package.

```
sudo SUSEConnect --product PackageHub/15.2/x86_64
sudo zypper install dkms
```

2. Add the ROCm repo.

```
sudo zypper clean -all
sudo zypper addrepo https://repo.radeon.com/rocm/zypp/4.1/
sudo zypper ref
sudo rpm --import https://repo.radeon.com/rocm/rocm.gpg.key
sudo zypper --gpg-auto-import-keys install rocm-dkms
sudo reboot
```

**Note:** For developer systems or Docker containers (where it could be beneficial to use a fixed ROCm version), select a versioned repository from:

<https://repo.radeon.com/rocm/zypp/>

3. Run the following command once

```
cat <<EOF | sudo tee /etc/modprobe.d/10-unsupported-modules.conf
allow_unsupported_modules 1
EOF
sudo modprobe amdgpu
```

4. Verify the ROCm installation.
5. Run `/opt/rocm/bin/rocminfo` and `/opt/rocm/rocm/bin/rocmclinfo` commands to list the GPUs and verify that the ROCm installation is successful.
6. Restart the system.
7. Test the basic ROCm installation.
8. After restarting the system, run the following commands to verify that the ROCm installation is successful. If you see your GPUs listed by both commands, the installation is considered successful.

```
/opt/rocm/bin/rocminfo
/opt/rocm/rocm/bin/rocmclinfo
```

**Note:** To run the ROCm programs more efficiently, add the ROCm binaries in your PATH.

```
echo 'export PATH=$PATH:/opt/rocm/bin:/opt/rocm/profiler/bin:/opt/rocm/rocmcl/bin' | sudo tee -a
/etc/profile.d/rocm.sh
```

### Using ROCm on SLES with Upstream Kernel Drivers

```
sudo zypper install rocm-dev
echo 'SUBSYSTEM=="kfd", KERNEL=="kfd", TAG+="uaccess", GROUP="video"' | sudo tee /etc/
udev/rules.d/70-kfd.rules
sudo reboot
```

### Uninstallation

To uninstall, use the following command:

```
sudo zypper remove rocm-opencl rocm-dkms rocmcl-dkms
```

**Note:** Ensure all other installed packages/components are removed. **Note:** Ensure all the content in the `/opt/rocm` directory is completely removed. If the command does not remove all the ROCm components/packages, ensure you remove them individually.

### 3.38.3.3.1 Performing an OpenCL-only Installation of ROCm

Some users may want to install a subset of the full ROCm installation. If you are trying to install on a system with a limited amount of storage space, or which will only run a small collection of known applications, you may want to install only the packages that are required to run OpenCL applications. To do that, you can run the following installation command instead of the command to install rocm-dkms.

```
sudo yum install rocm-dkms rocm-ocl-devel && sudo reboot
```

### 3.38.4 ROCm Installation Known Issues and Workarounds

The ROCm platform relies on some closed source components to provide functionalities like HSA image support. These components are only available through the ROCm repositories, and they may be deprecated or become open source components in the future. These components are made available in the following packages:

- hsa-ext-rocr-dev

### 3.38.5 Getting the ROCm Source Code

AMD ROCm is built from open source software. It is, therefore, possible to modify the various components of ROCm by downloading the source code and rebuilding the components. The source code for ROCm components can be cloned from each of the GitHub repositories using git. For easy access to download the correct versions of each of these tools, the ROCm repository contains a repo manifest file called default.xml. You can use this manifest file to download the source code for ROCm software.

The repo tool from Google® allows you to manage multiple git repositories simultaneously. Run the following commands to install the repo:

```
mkdir -p ~/bin/
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
```

Note: You can choose a different folder to install the repo into if you desire. ~/bin/ is used as an example.

### 3.38.6 Downloading the ROCm Source Code

The following example shows how to use the repo binary to download the ROCm source code. If you choose a directory other than ~/bin/ to install the repo, you must use that chosen directory in the code as shown below:

```
mkdir -p ~/ROCM/
cd ~/ROCM/
~/bin/repo init -u https://github.com/RadeonOpenCompute/ROCM.git -b roc-4.1.x
repo sync
```

**Note:** Using this sample code will cause the repo to download the open source code associated with this ROCm release. Ensure that you have ssh-keys configured on your machine for your GitHub ID prior to the download.