AMD

ROCm Software Stack

IWOMP 2020 Vendor Presentation

Greg Rodgers
Derek Bouius

Sept 2020

AMD PUBLIC

[AMD Public Use]

Cautionary Statement

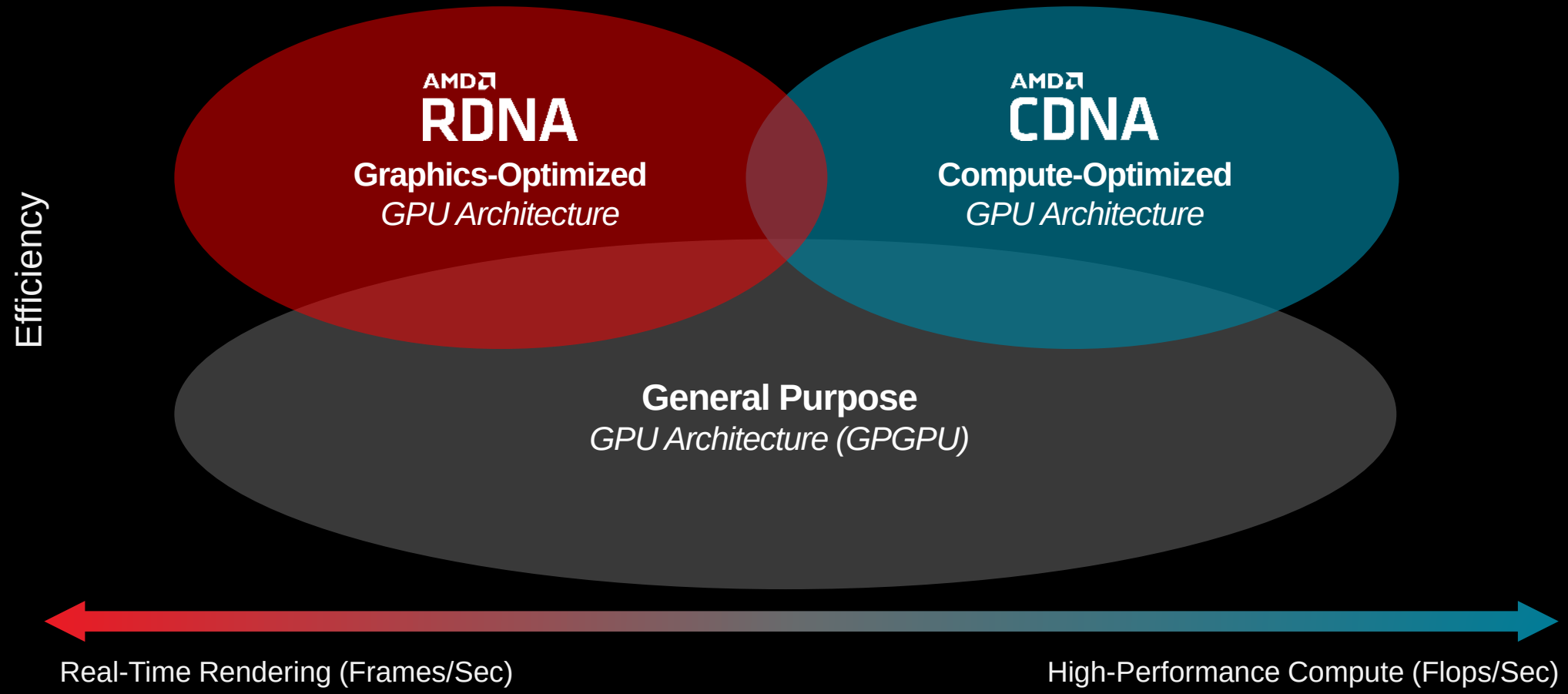
This presentation contains forward-looking statements concerning Advanced Micro Devices, Inc. (AMD) including, but not limited to the features, functionality, availability, timing, expectations and expected benefits of AMD’s products, which are made pursuant to the Safe Harbor provisions of the Private Securities Litigation Reform Act of 1995. Forward-looking statements are commonly identified by words such as "would," "may," "expects," "believes," "plans," "intends," "projects" and other terms with similar meaning. Investors are cautioned that the forward-looking statements in this presentation are based on current beliefs, assumptions and expectations, speak only as of the date of this presentation and involve risks and uncertainties that could cause actual results to differ materially from current expectations. Such statements are subject to certain known and unknown risks and uncertainties, many of which are difficult to predict and generally beyond AMD's control, that could cause actual results and other future events to differ materially from those expressed in, or implied or projected by, the forward-looking information and statements. Investors are urged to review in detail the risks and uncertainties in AMD's Securities and Exchange Commission filings, including but not limited to AMD’s Quarterly Report on Form 10-Q for the quarter ended June 27, 2020.

AMD

[AMD Public Use]

Differentiated Strategy

Optimal Efficiency Through Domain-Specific Optimizations



AMD

AMD EPYC™ CPUs & Radeon Instinct™ GPUs Leading The Exascale Era

>2 ExaFLOPS Expected	Expected to be More Powerful than Today's 200 Fastest Supercomputers Combined	AMD Shipments in 2022

Source: https://www.top500.org/lists/top500/2020/06/

[AMD Public Use]

AMD CDNA Architecture

Compute DNA for the Data Center

Performance
Accelerate ML/HPC with Compute/Tensor OPS

Efficiency
Help Reduce TCO with High Perf-per-Watt

Features
Enhance Enterprise RAS, Security and Virtualization

Scalability
Scale Performance with AMD Infinity Architecture

[AMD Public Use]

What is ROCm™?

An Open Software Platform for GPU-accelerated Computing

AMD ROCm

Frameworks and Applications
TensorFlow, PyTorch, Caffe2

Libraries
MIOpen, roc* libraries

Programming models
HIP, C/C++, Python

Intermediate runtimes/compilers
OpenMP, HIP, OpenCL

Runtimes
ROCm

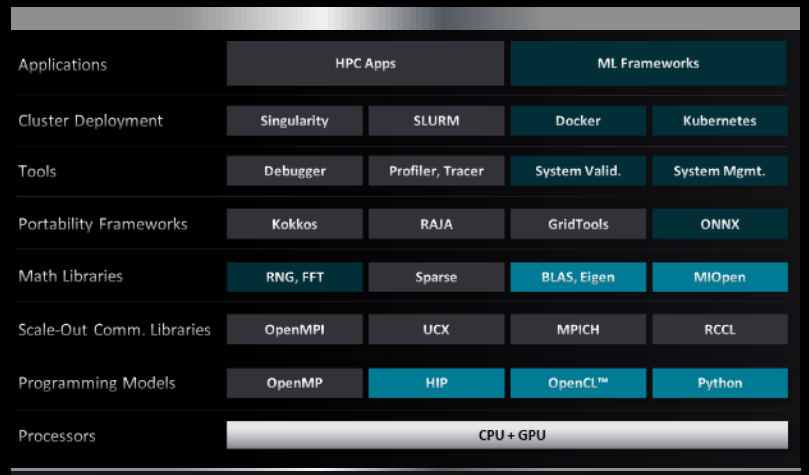
Programmer and system tools
-debug
-profile

[AMD Public Use]

Data Center Software Evolution

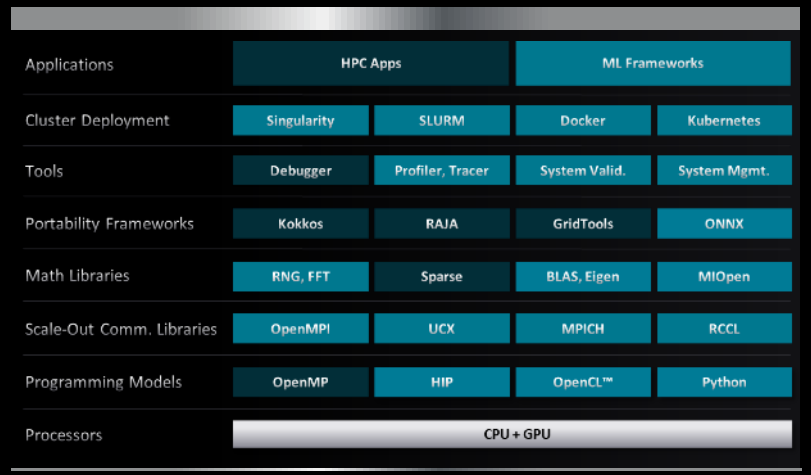
Steady Progress and Growing Ecosystem Support

AMD ROCm



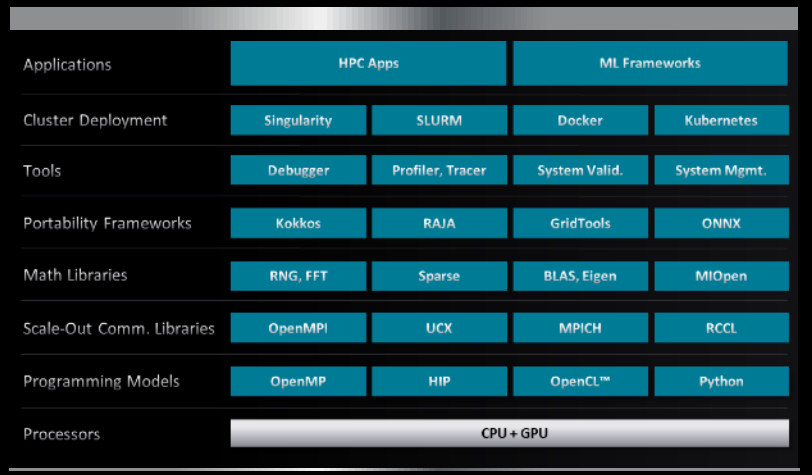
2018: AMD ROCm™ 2.0 Platform

Building the Foundation



2019: AMD ROCm™ 3.0 Platform

Focused on Machine Learning



2020 Plan: AMD ROCm™ 4.0 Platform

Complete Exascale Solution for ML/HPC

ECLIPSE FOUNDATION · Red Hat · slurm · docker · TensorFlow · PYTORCH · MLIR · LLVM COMPILER INFRASTRUCTURE · tvm · CentOS · SUSE · ECP EXASCALE COMPUTING PROJECT · mindtech · ubuntu · OpenMP

Use of third party marks / logos/ products is for informational purposes only and no endorsement of or by AMD is intended or implied. GD-83

Future | Beta/Early | Production

AMD

[AMD Public Use]

Machine Intelligence

Natural Language Processing

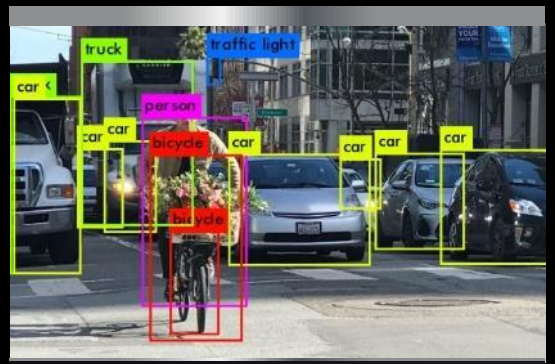


Image Recognition

Recommendation Engines

Industrial Automation

Revolutionizing Applications in Every Field

Exponentially Growing Demands for Performance

AMD Champions Open Source Solutions

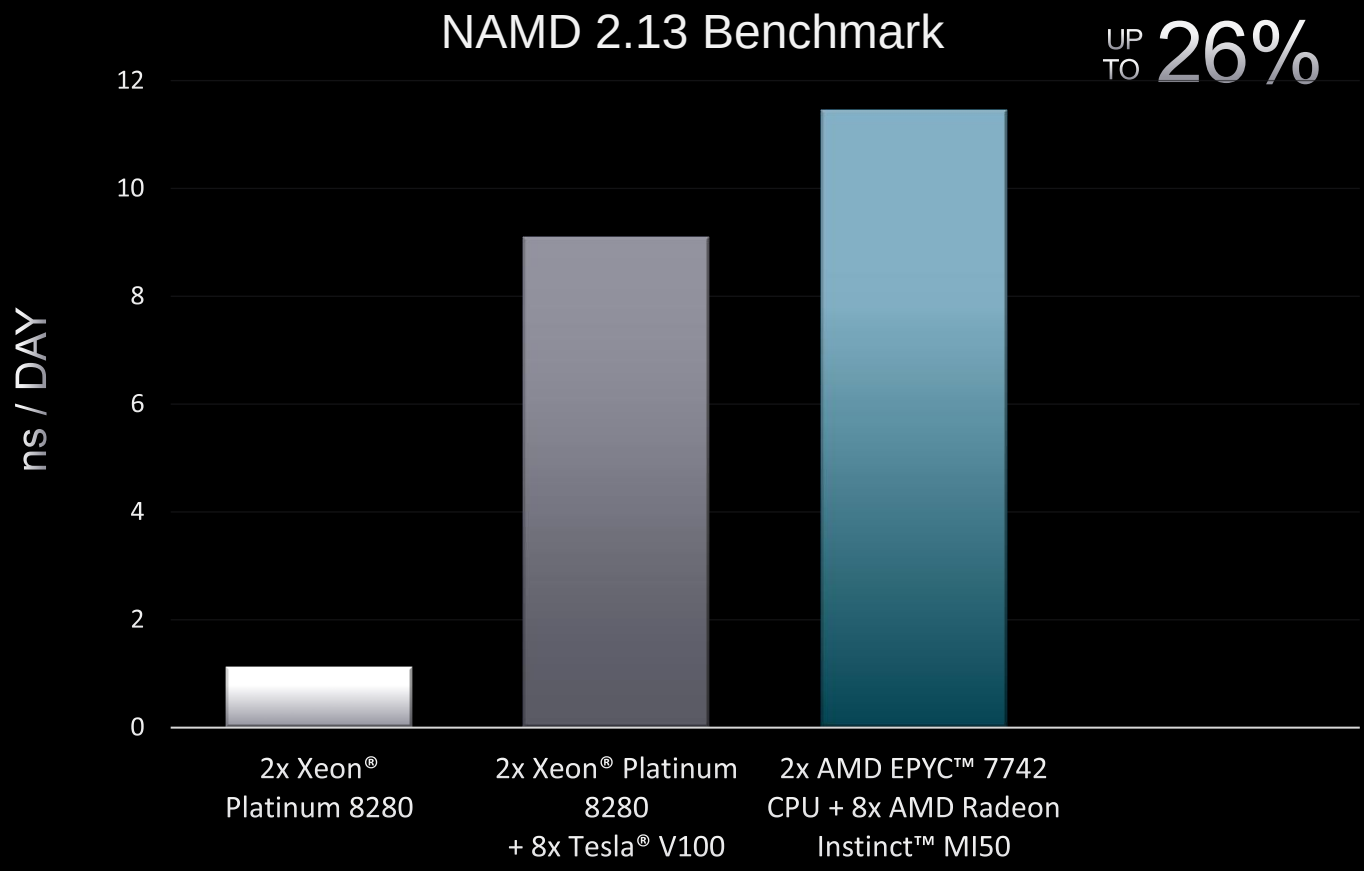
AMD

[AMD Public Use]

AMD CPU + GPU + SW Advantages

Driving High-performance Computing Leadership

- Fully Integrated CPU and GPU Systems and Unified Tools
- Infinity Architecture for Bandwidth and Coherency
- Open Source Software Optimized for Performance



Testing Conducted by AMD performance lab as of 11-10-2019 using NAMD 2.13, STMV 1M Atom benchmark. Best-in-class based on industry-standard pin-based (LGA) X86 processors. Results may vary. (RIV-20)

AMD

[AMD Public Use]

HIP: Multi-Platform Capability for TCO Optimization

Easy to Deploy Porting Capability

CUDA-based application

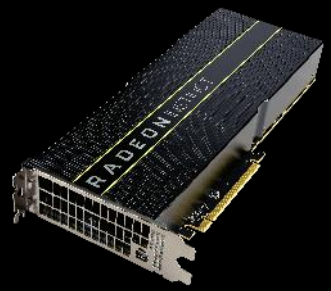
"HIPify"

Virtually Automatic Conversion

Portable HIP C++

Developer maintains HIP port

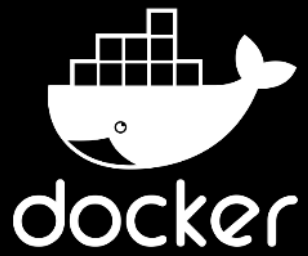
Resulting C++ code runs on NVIDIA or AMD GPUs



AMD

[AMD Public Use]

Fast-Growing ROCm™ Ecosystem



Containers

Sylabs Singularity

Data Center Workload Manager

Container Orchestration

Performance Profiling & System Tracer via PAPI

Eclipse C/C++ Development Tooling Based on ROC-GDB

Upstream ML Frameworks

Exascale Tools, Programming Models and Applications

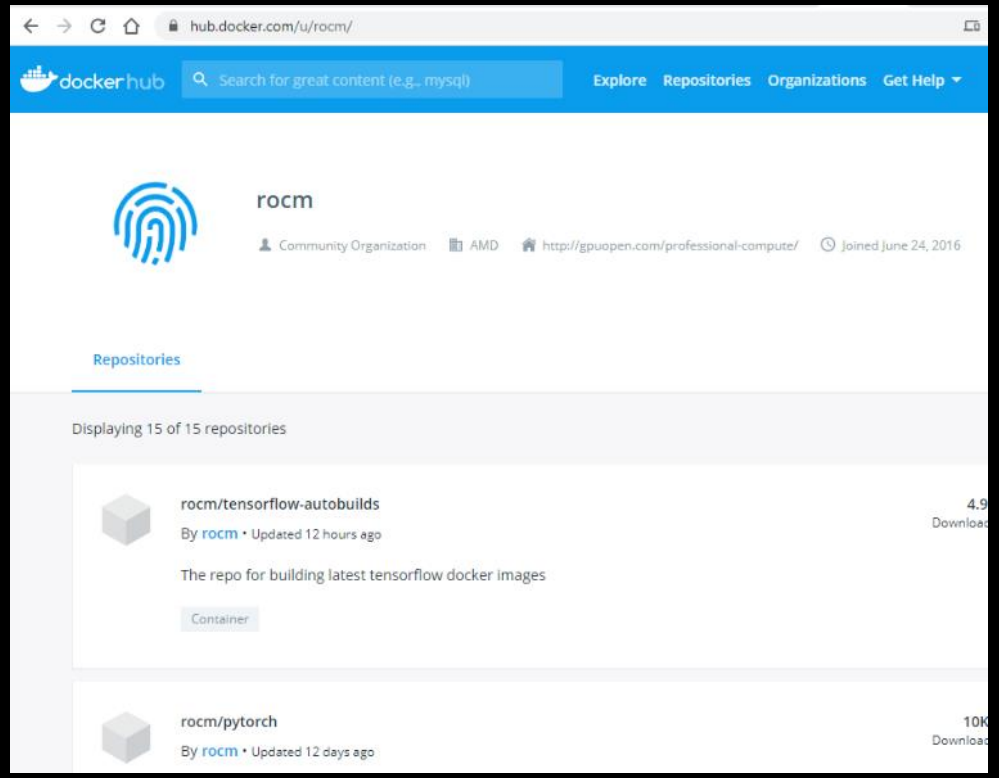
Use of third party marks / logos/ products is for informational purposes only and no endorsement of or by AMD is intended or implied. GD-83

AMD

[AMD Public Use]

Docker®

- Set permissions and add user to docker group
  - groups # identify the groups member
  - sudo usermod -a -G docker $LOGNAME
- ROCm™ Docker Hub
  - https://hub.docker.com/u/rocm/
- Run Docker Image
- docker run -it --network=host --device=/dev/kfd --device=/dev/dri --group-add video --cap-add=SYS_PTRACE --security-opt seccomp=unconfined -v /home/user:/home/user rocm/dev-ubuntu-18.04 bash
- Show running image
  - docker image ls
- Save container to your own image
  - Run docker commit on another terminal window
  - docker commit <container id> <my_docker_image>



AMD

[AMD Public Use]

Machine Learning Models

Deployable Today with Continuous Optimizations

Image Classification	Object Detection	Neural Machine Translation	Reinforcement Learning	Recommender Systems	Generative Models
• ResNet50/101 • ResNet152 • Inception3/4 • VGG16/19 • ShuffleNet • MobileNet • DenseNet • AlexNet • SqueezeNet • GoogleNet • ResNext101	• Faster-RCNN-ResNet50 • Mask-RCNN-ResNet50 • SSD-Resnet50	• GNMT: LSTMs • Translate: LSTMs • BERT: Transformer • GPT-2: Transformer	• Atari • Cart_Pole • VizDoom	• DLRM	• DCGAN • Fast Neural Style Transfer

AMD

AMD GPU Compilers:

C/C++

The GCN ISA is free and open!
https://developer.amd.com/resources/developer-guides-manuals/

HIP (hip-clang)

- HIP (Heterogeneous Interface for Portability) is an interface that provides similar functionality to CUDA API
- Compiles HIP code and emits AMDGCN into binary
- hipcc -> hip-clang -> amdgcn
- Compiles to NVIDIA GPU with NVCC & its tool chain
- All the x86 pieces are dealt with in the same way

AOMP (AMD OpenMP Compiler)

- LLVM
- Compiles C/C++ code with OpenMP “target” pragmas
- Sources feeds into ROCm compiler for future unified LLVM compiler

OpenCL™

- LLVM
- Khronos Industry Standard accelerator language

AMD

AMD GPU Compilers:

Fortran

AOMP

- LLVM clang driver for flang
- Limited flang support for OpenMP 4.5+ target offload
- Will move to F18 in the future
- Feeds into ROCm compiler

hipfort

- New package for HIP and ROCm library APIs in FORTRAN
- Offload kernels to GPU using Fortran 2003 C-binding
- Generated FORTRAN interface and mod files
- Designed for multiple compilers, default is gfortran

Frontier

- See Frontier spec sheet for what is expected to be supported: https://www.olcf.ornl.gov/wp-content/uploads/2019/05/frontier_specsheet.pdf

AMD

[AMD Public Use]

Unified CPU & GPU Debugger

Easily Integrated with Industry Standard Tools

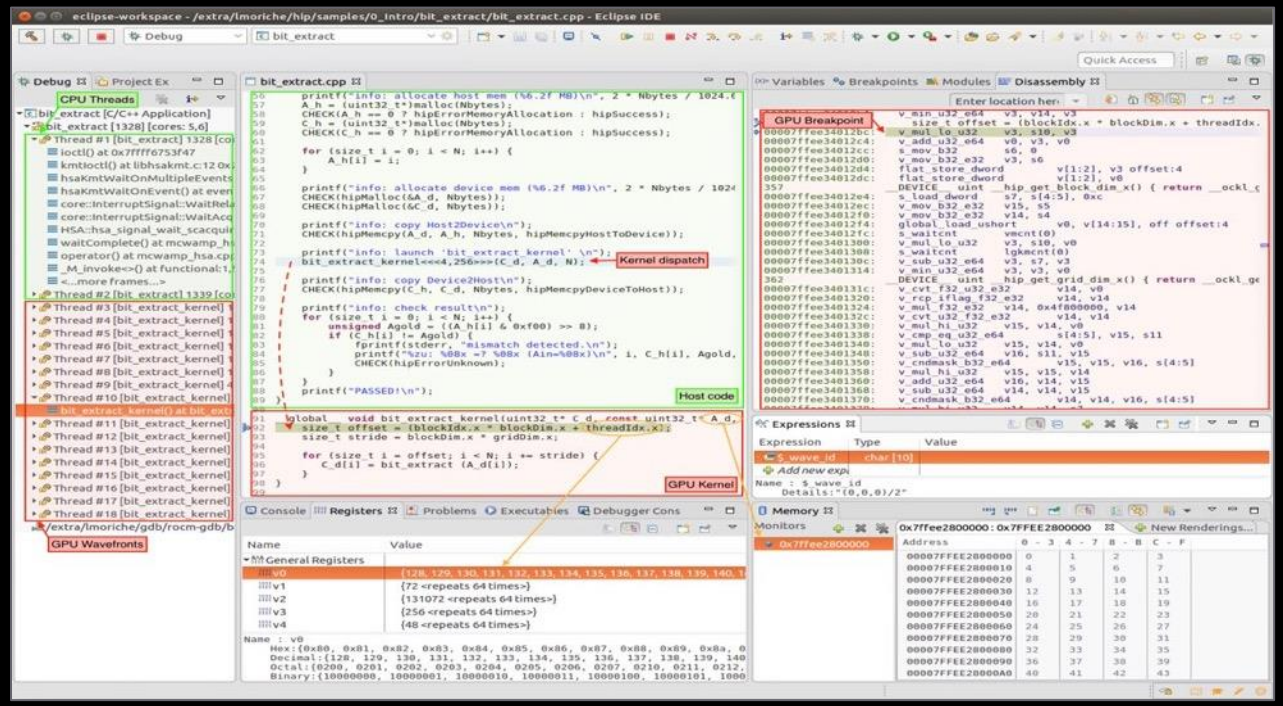
eclipse

THEIA

arm DDT

gdb CLI

CRAY a Hewlett Packard Enterprise company CCDB



ROC-gdb
(gdb with GPU support)

RogueWave SOFTWARE TotalView®

Linux Kernel
(ptrace)

ROC-dbgapi
(GPU Low Level Debug API Library)

GPU Kernel Driver

RADEON INSTINCT

POLYTECHNIQUE MONTRÉAL

Released Q2-2020

AMD

[AMD Public Use]

ROCgdb

- ROCgdb is the ROCm source-level debugger for Linux
- ROCgdb is based on GDB, the GNU source-level debugger
  - https://github.com/ROCm-Developer-Tools/ROCgdb
- Compile executable using hipcc with “--ggdb"
- ROCgdb location:
  - `/opt/rocm/bin/rocgdb`
- To debug an executable
  - `rocgdb $EXE`
- To attach to a running process
  - `rocgdb -p <pid>`

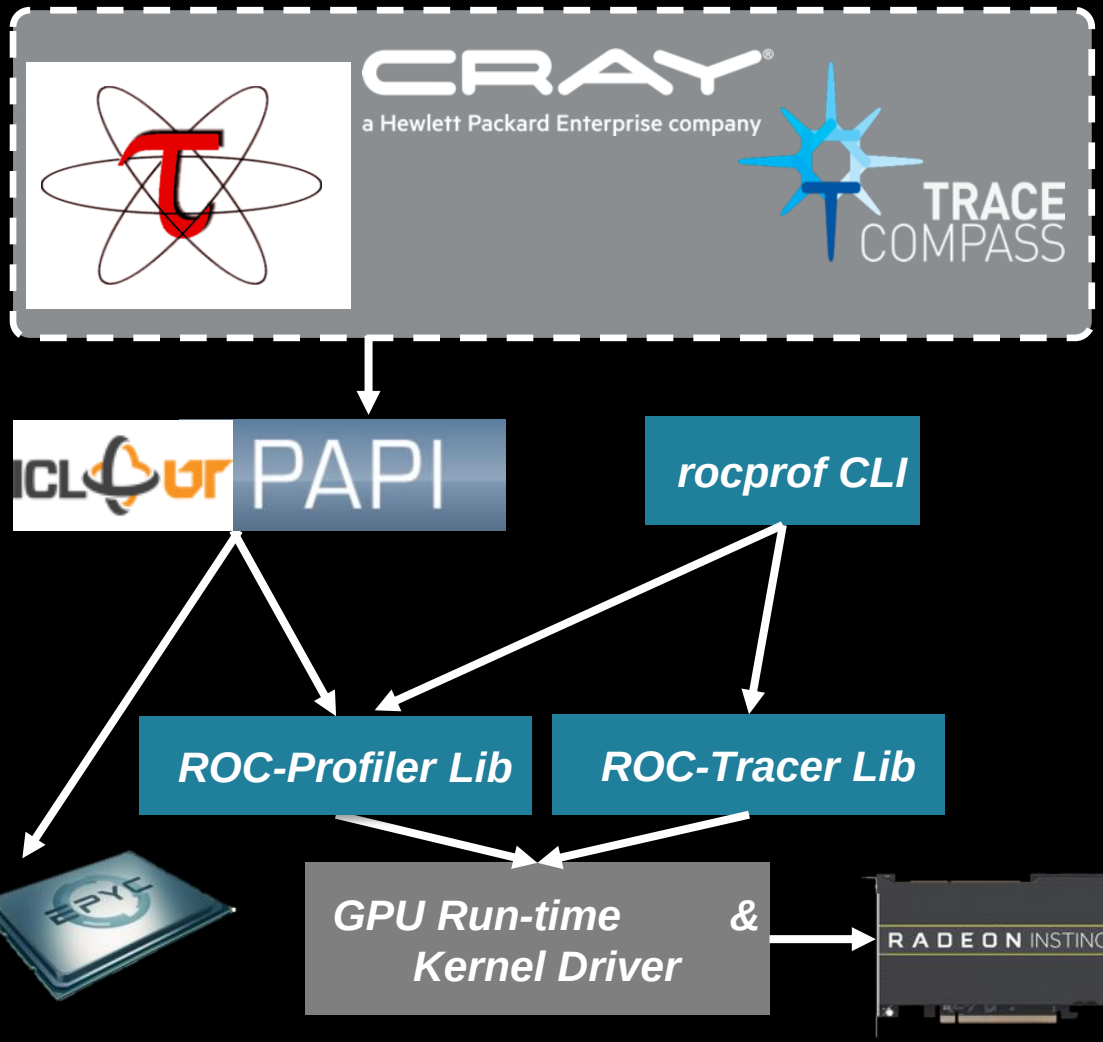
```
(gdb) where
#0  0x000014825547ce57 in sched_yield () from /lib/x86_64-linux-gnu/libc.so.6
#1  0x00001482771954ad in amd::Event::awaitCompletion() () from /opt/rocm/hip/lib/libamdhip64.so.3
#2  0x00001482770ebc2d in ihipMemcpy(void*, void const*, unsigned long, hipMemcpyKind, amd::HostQueue&, bool) ()
   from /opt/rocm/hip/lib/libamdhip64.so.3
#3  0x00001482770ec127 in hipMemcpy () from /opt/rocm/hip/lib/libamdhip64.so.3
#4  0x0000000000409b08 in HPL_pdlange (GRID=<optimized out>, GRID@entry=0x7ffffcdf8770, NORM=<optimized out>,
    NORM@entry=HPL_NORM_1, M=<optimized out>, M@entry=45000, N=<optimized out>, N@entry=45000, NB=<optimized out>,
    NB@entry=384, A=<optimized out>, LDA=<optimized out>) at ../HPL_pdlange.cpp:302
#5  0x00000000004070ce in HPL_pdtest (TEST=TEST@entry=0x7ffffcdf8700, GRID=GRID@entry=0x7ffffcdf8770,
    ALGO=ALGO@entry=0x7ffffcdf8730, N=45000, NB=384) at ../HPL_pdtest.c:376
#6  0x0000000000402b33 in main (ARGC=<optimized out>, ARGV=<optimized out>) at ../HPL_pddriver.c:227
```

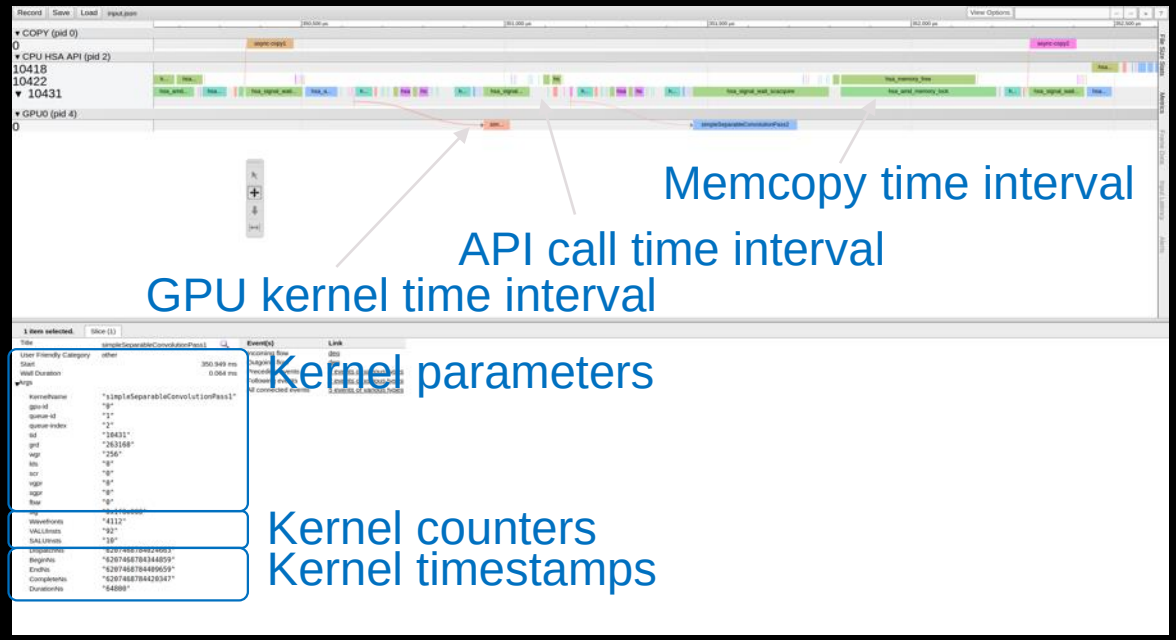

AMD

[AMD Public Use]

ROC-Profiler / Tracer

Easily Integrated with Industry Standard Tools





UNIVERSITY OF OREGON

ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

POLYTECHNIQUE MONTRÉAL

Released Q4-2019

AMD

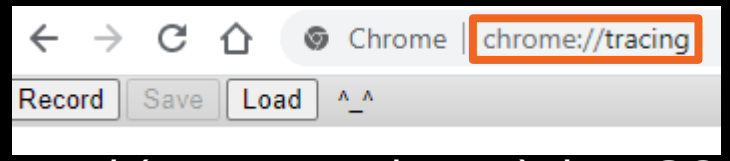
[AMD Public Use]

rocprof

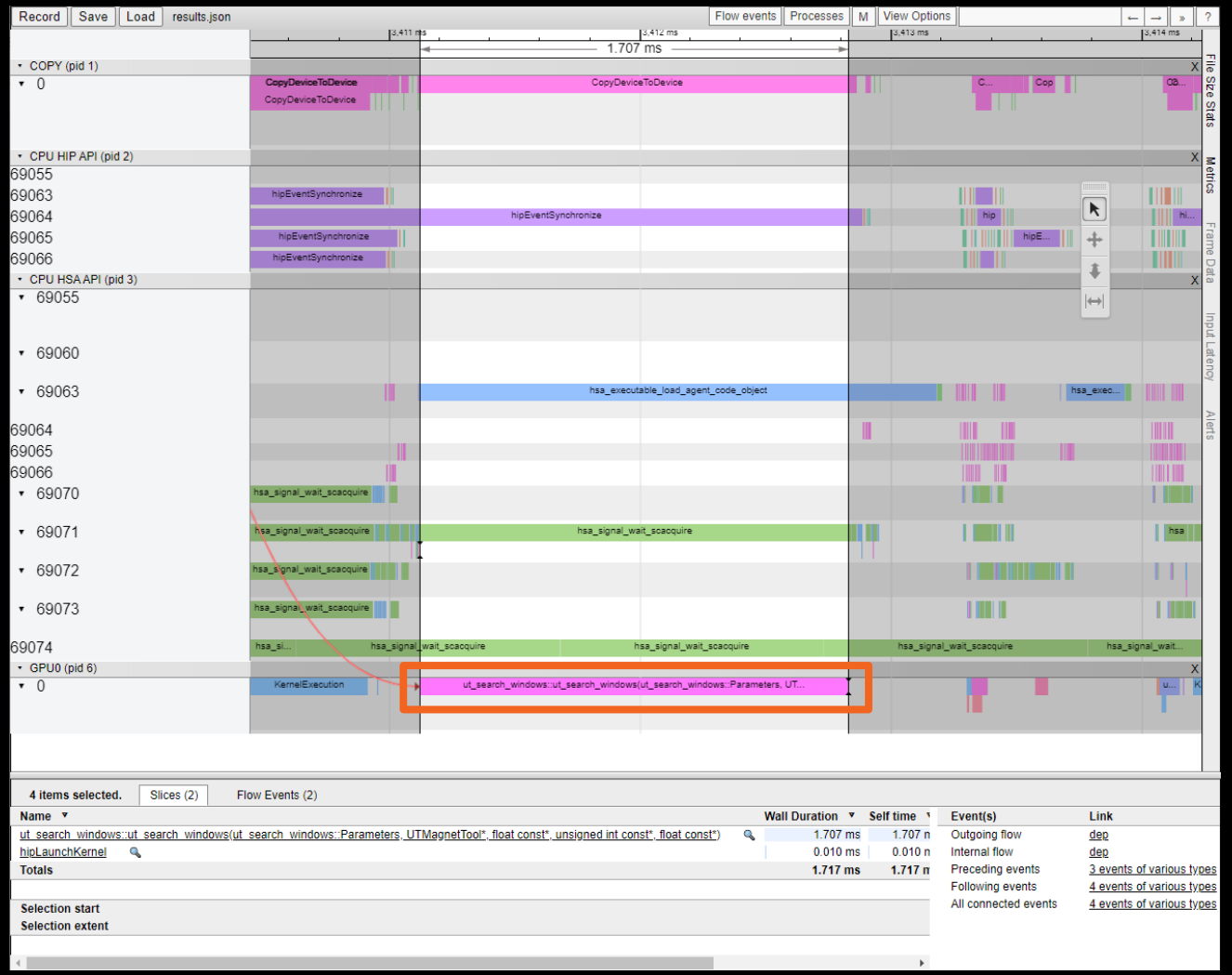
- rocprof is the AMD GPU profiler library
- It profiles with perf-counters and derived metrics
- To run rocprof to generate a kernel profile (text)
  - `rocprof --obj-tracking on --stats $EXE`
  - The default `results.stats.csv` will be generated
  - Comma-separated list of kernel activities

```
"Name","Calls","TotalDurationNs","AverageNs","Percentage"
KernelExecution,1614,473635087,293454,70.2867228678686
"lf_triplet_seeding::lf_triplet_seeding(lf_triplet_seeding::Parameters, LookingF
orward::Constants const*)",27,57000230,2111119,8.458746996112579
"velo_search_by_triplet::velo_search_by_triplet(velo_search_by_triplet::Parameter
rs, VeloGeometry const*)",20,27701080,1385054,4.110797925535989
"velo_calculate_phi_and_sort::velo_calculate_phi_and_sort(velo_calculate_phi_and
_sort::Parameters)",15,11600465,773364,1.721491272443271
```

- Run rocprof to generate a trace file
  - `rocprof --obj-tracking on --sys-trace $EXE`
  - Start Google Chrome
  - Type `chrome://tracing`



- Load (or Drag and Drop) the JSON file to view
- https://github.com/ROCm-Developer-Tools/rocprofiler/




```
--obj-tracking <on|off> - to turn on/off kernels code objects tracking [off]
  To support V3 code object

--stats - generating kernel execution stats, file <output name>.stats.csv

--roctx-trace - to enable rocTX application code annotation trace, "Markers and Ranges" JSON trace section.
--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
--sys-trace - to trace HIP/HSA APIs and GPU activity, generates stats and JSON trace chrome-tracing compatible
  '--hsa-trace' can be used in addition to select activity tracing from HSA (ROCr runtime) level
--kfd-trace - to trace KFD, generates KFD Thunk API execution stats and JSON file chrome-tracing compatible
  Generated files: <output name>.<domain>_stats.txt <output name>.json
```


AMD

[AMD Public Use]

ROCm™ Installation v3.8.0(latest) – Ubuntu® 18.04

1

Ensure that the 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

wget -q -O - http://repo.radeon.com/rocm/apt/debian/rocm.gpg.key | sudo apt-key add -
echo 'deb [arch=amd64] http://repo.radeon.com/rocm/apt/debian/ xenial main' | sudo tee /etc/apt/sources.list.d/rocm.list

3

Install the ROCm meta-package & rocm-dkms meta-package

sudo apt update
sudo apt install –y rocm-dkms miopen-hip rocblas

AMD

[AMD Public Use]

ROCm™ Installation v3.8.0(latest) – Ubuntu®

4

Set permissions and add user to video group

groups *# identify the groups member*

sudo usermod -a -G video $LOGNAME

5

Restart the system

6

Test the basic ROCm installation

/opt/rocm/bin/rocminfo

dpkg -l | grep rocm *#Report installed ROCm versions*

AMD

Visit AMD.com/ROCm

Link to more training information:
https://community.amd.com/community/radeon-instinct-accelerators/blog/

© Copyright Advanced Micro Devices, Inc, All rights reserved.

Thank You!

© Copyright Advanced Micro Devices, Inc, All rights reserved.

[AMD Public Use]

Disclaimers and Attributions

The information contained herein is for informational purposes only, and is subject to change without notice. Timelines, roadmaps, and/or product release dates shown in these slides are plans only and subject to change. “Polaris”, “Vega”, “Radeon Vega”, “Navi”, “Zen” and “Naples” are codenames for AMD architectures, and are not product names.

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.

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. GD-18

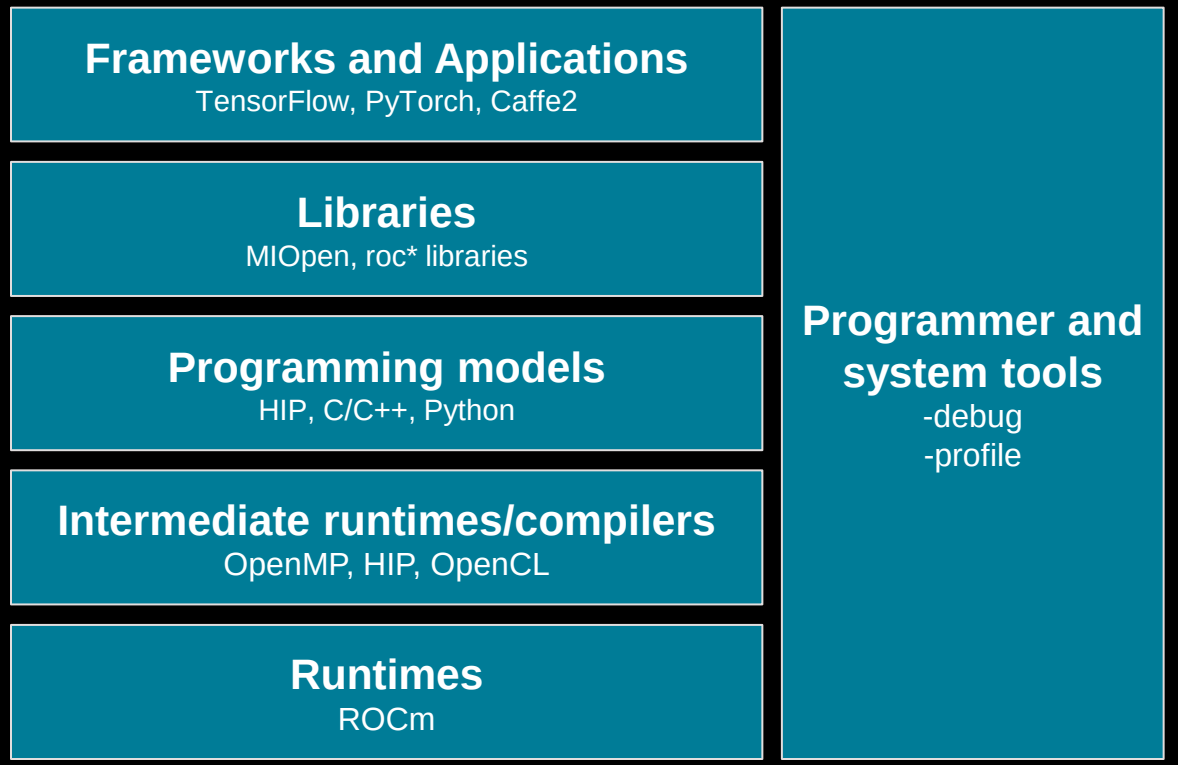
©2020 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, EPYC, ROCm, RDNA, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

[AMD Public Use]

What is ROCm™?

An Open Software Platform for GPU-accelerated Computing

AMD ROCm



AMD

AMD

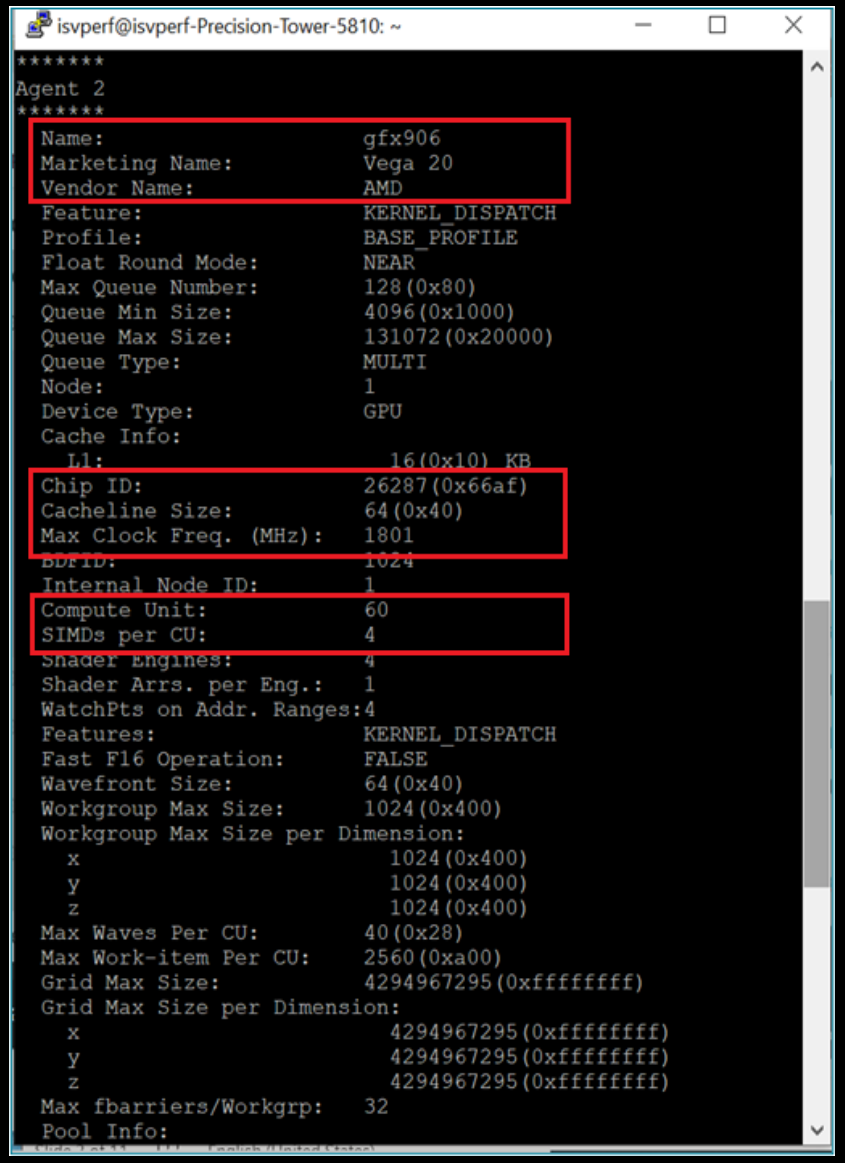
BACKUP DETAIL SLIDES

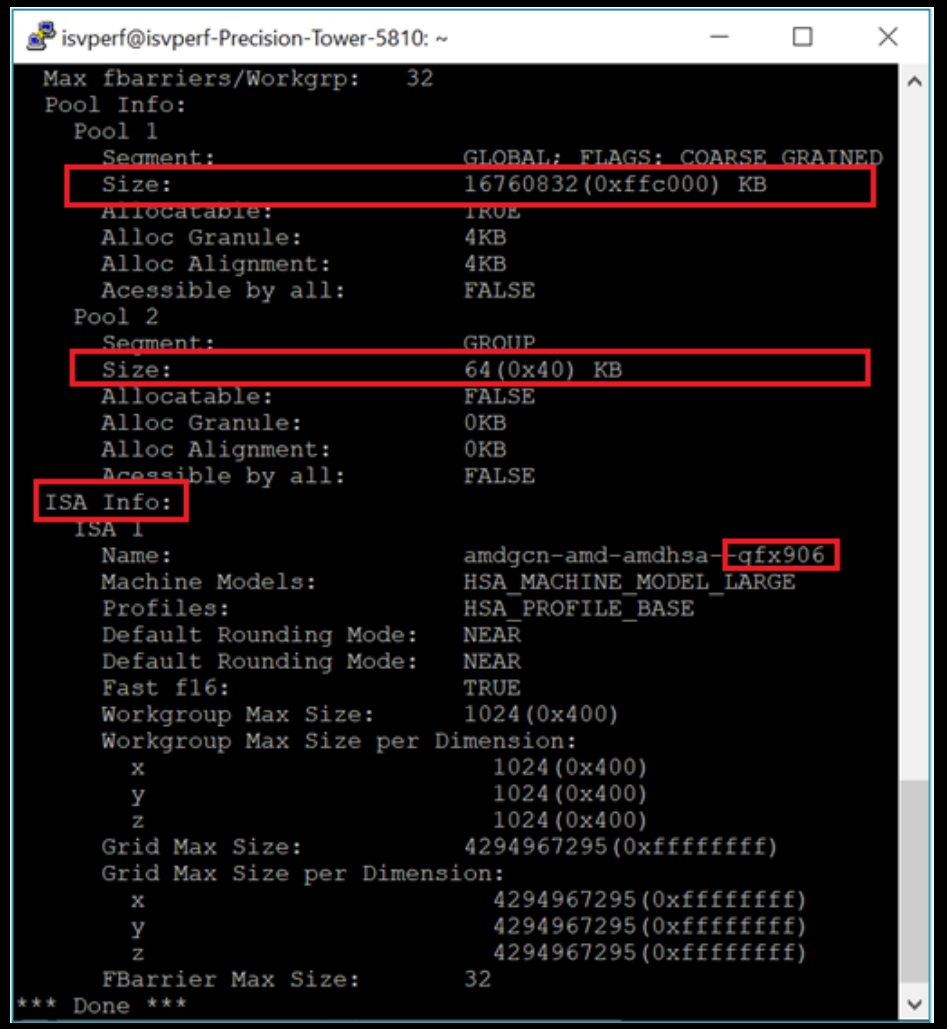
Sept 2020

AMD PUBLIC

[AMD Public Use]

ROCm™ Screen Info





[AMD Public Use]

ROCm™ Version Details

isvperf@isvperf-Precision-Tower-5810: ~

```
isvperf@isvperf-Precision-Tower-5810:~$ dpkg -l | grep rocm
ii  comgr                                 1.6.0.116-rocm-rel-3.0-6-7665c20          amd64        Library to provide support functions
ii  hip-base                              3.0.19493.4438-rocm-rel-3.0-6-36529b1     amd64        HIP: Heterogenous-computing Interface for Portabi
lity [BASE]
ii  hip-doc                               3.0.19493.4438-rocm-rel-3.0-6-36529b1     amd64        HIP: Heterogenous-computing Interface for Portabi
lity [DOCUMENTATION]
ii  hip-hcc                               3.0.19493.4438-rocm-rel-3.0-6-36529b1     amd64        HIP: Heterogenous-computing Interface for Portabi
lity [HCC]
ii  hip-samples                           3.0.19493.4438-rocm-rel-3.0-6-36529b1     amd64        HIP: Heterogenous-computing Interface for Portabi
lity [SAMPLES]
ii  hsa-ext-rocr-dev                      1.1.9.0-rocm-rel-3.0-6-7128d0d            amd64        AMD Heterogeneous System Architecture HSA - Linux
 HSA Runtime extensions for ROCm platforms
ii  hsa-rocr-dev                          1.1.9.0-rocm-rel-3.0-6-7128d0d            amd64        AMD Heterogeneous System Architecture HSA - Linux
 HSA Runtime for Boltzmann (ROCm) platforms
ii  rocm-clang-ocl                        0.5.0.47-rocm-rel-3.0-6-cfddddb           amd64        OpenCL compilation with clang compiler.
ii  rocm-cmake                            0.3.0.134-rocm-rel-3.0-6-e6d1ef3          amd64        rocm-cmake built using CMake
ii  rocm-debug-agent                      1.0.0                                     amd64        Radeon Open Compute (ROCm) Runtime debug agent
ii  rocm-dev                              3.0.6                                     amd64        Radeon Open Compute (ROCm) Runtime software stack
ii  rocm-device-libs                      1.0.0.559-rocm-rel-3.0-6-628eea4          amd64        Radeon Open Compute - device libraries
ii  rocm-dkms                             3.0.6                                     amd64        Radeon Open Compute (ROCm) Runtime software stack
ii  rocm-opencl                           2.0.0-rocm-rel-3.0-6-9a4afec              amd64        OpenCL/ROCm
ii  rocm-opencl-dev                       2.0.0-rocm-rel-3.0-6-9a4afec              amd64        OpenCL/ROCm
ii  rocm-smi                              1.0.0-192-rocm-rel-3.0-6-g01752f2         amd64        System Management Interface for ROCm
ii  rocm-smi-lib64                        2.2.0.8.rocm-rel-3.0-6-8ffe1bc            amd64        ROCm System Management Interface library
ii  rocm-utils                            3.0.6                                     amd64        Radeon Open Compute (ROCm) Runtime software stack
ii  rocminfo                              1.0.0                                     amd64        Radeon Open Compute (ROCm) Runtime rocminfo tool
isvperf@isvperf-Precision-Tower-5810:~$
```

Note: demo purpose only, please check the release notes for the latest rocm lib versions

AMD

[AMD Public Use]

Basic ROCm™ Tools

1. rocm-smi

2. rocm-bandwidth-test (*https://github.com/RadeonOpenCompute/rocm_bandwidth_test*)

```
./rocm-bandwidth-test -b 2,0      # gpu0↔cpu0 bidirectional
./rocm-bandwidth-test -b 2,3      # gpu0↔gpu1 bidirectional
```

3. rocblas-bench (DGEMM, SGEMM)

```
./rocblas-bench -f gemm -r d -m 8640 -n 8640 -k 8640 --transposeB T --initialization
trig_float  -i 200 --device 0 &
```

4. rvs (rocm-validation-suite) *# Cluster management tool*

```
sudo ./rvs -c conf/Artus_dgemm_gst.conf -d 3 -l RVS_dgemm_result.log
```


AMD

[AMD Public Use]

ROCm™ SMI Screen Info

```
isvperf@isvperf-SYS-4029GP-TRT2:~$ rocm-smi


========================ROCm System Management Interface========================
================================================================================
GPU   Temp    AvgPwr   SCLK      MCLK      Fan     Perf   PwrCap   VRAM%   GPU%
1     65.0c   210.0W   1725Mhz   1000Mhz   0.0%    high   225.0W    44%    99%
2     28.0c   26.0W    1725Mhz   1000Mhz   0.0%    high   225.0W     0%    0%
3     30.0c   29.0W    1725Mhz   1000Mhz   0.0%    high   225.0W     0%    0%
4     29.0c   29.0W    1725Mhz   1000Mhz   0.0%    high   225.0W     0%    0%
5     31.0c   26.0W    1725Mhz   1000Mhz   0.0%    high   225.0W     0%    0%
6     29.0c   30.0W    1725Mhz   1000Mhz   0.0%    high   225.0W     0%    0%
7     29.0c   24.0W    1725Mhz   1000Mhz   1.96%   high   225.0W     0%    0%
8     29.0c   28.0W    1725Mhz   1000Mhz   0.0%    high   225.0W     0%    0%
================================================================================
============================End of ROCm SMI Log =================================
isvperf@isvperf-SYS-4029GP-TRT2:~$
```


AMD

[AMD Public Use]

ROCm™ Bandwidth Test – Installation

1

Tools

$sudo bash

2

Add rocm bandwidth-tests

#apt-get -y update && sudo apt-get install -y libpci3 libpci-dev doxygen unzip cmake git
#cd /opt/rocm
#git clone https://github.com/RadeonOpenCompute/rocm_bandwidth_test.git
#cd rocm_bandwidth_test;mkdir ./build;cmake ./ -B./build;make -C ./build

3

Install rocm-bandwitdh-test package from ROCm repo

#apt install -y rocm-bandwidth-test
#exit

4

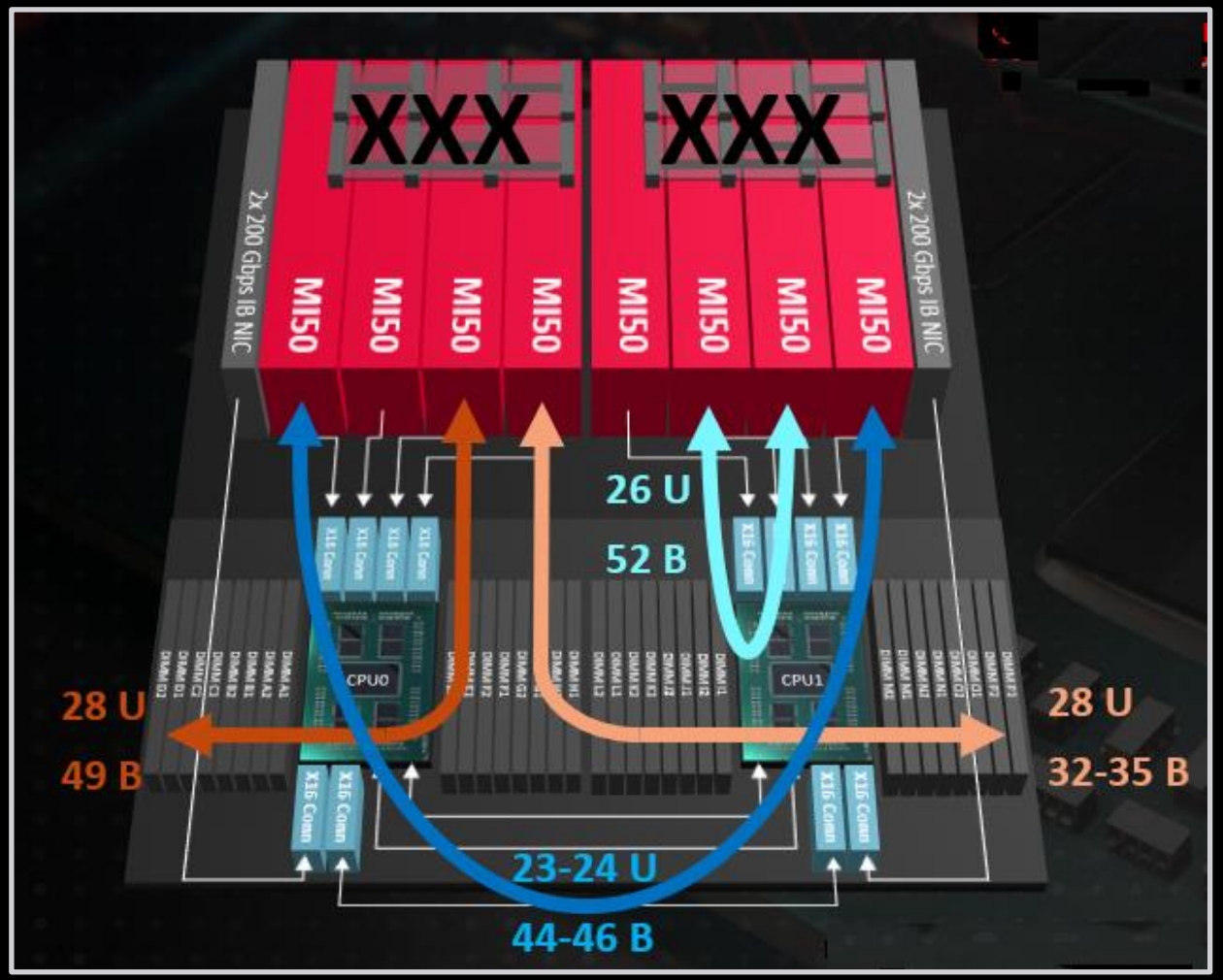
Run RBT

$rocm-bandwidth-test

AMD

[AMD Public Use]

ROCm™ Bandwidth Test – Platform



Note: This is the reference platform with following config:

- Dual Socket AMD EPYC 7742,
- 8x MI50
- 512 GB DDR4 3200
- 960GB NVMe drive
- 256 GB HBM2 @ 8 TB/s

Note: with P2P connected with xGMI, the achievable bandwidth could be updating the light blue arrows from 26 to 32 for unidirectional and 52 to 59 for bidirectional

See End Notes

AMD

[AMD Public Use]

ROCm™ Bandwidth – Bidirectional (*Target-pcie-gen4*)

D/D	CPU 0	CPU 1	GPU 0	GPU 1	GPU 2	GPU 3	GPU 4	GPU 5	GPU 6	GPU 7
CPU 0			49.3	48.9	48.7	49.3	35.2	32.2	32.2	32.1
CPU 1			32.2	34.1	34.2	32.2	48.7	49.3	48.4	49.4
GPU 0	49.3	32.2		52.4	52.1	52.7	45.7	43.7	43.7	44.4
GPU 1	48.9	34.1	52.4		52.1	52.4	45.8	45.4	45.3	45.0
GPU 2	48.7	34.2	52.1	52.1		52.1	45.8	45.6	45.6	45.0
GPU 3	49.3	32.2	52.7	52.4	52.1		45.8	44.6	44.6	43.7
GPU 4	35.2	48.7	45.7	45.8	45.8	45.8		52.3	51.4	52.3
GPU 5	32.2	49.3	43.7	45.4	45.6	44.6	52.3		51.4	52.7
GPU 6	32.2	48.4	43.7	45.3	45.6	44.6	51.4	51.4		51.4
GPU 7	32.1	49.4	44.4	45.0	45.0	43.7	52.3	52.7	51.4	

Note: with GPU (MI50) to GPU connected with xGMI, the achievable bandwidth could be updating the above table's light blue area from ~52 to ~59 for bidirectional

See End Notes

AMD

[AMD Public Use]

rocBLAS – Installation and Build

1

```
# Tools
        $sudo bash
```

2

```
# Add rocBLAS git and build
        #cd /opt/rocm
        #git clone https://github.com/ROCmSoftwarePlatform/rocBLAS.git
        #cd rocBLAS;
        #mkdir ./build;cmake ./ -B./build;make -C ./build;./install.sh -idc
```

3

```
# Install rocBLAS
        #apt install -y rocblas
        #exit
```

4

```
# Run Rocblas-Bench
        $ /opt/rocm/rocBLAS/build/release/clients/staging/ rocblas-bench -f gemm -r d -m 8640 -n
        8640 -k 8640 --transposeB T --initialization trig_float  -i 200 --device 0 &
```


AMD

[AMD Public Use]

rocBLAS-Bench DGEMM Results

```
rocblas-bench INFO: lda < min_lda, set lda = 8640
rocblas-bench INFO: ldb < min_ldb, set ldb = 8640
rocblas-bench INFO: ldc < min_ldc, set ldc = 8640
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,5390.47,239301
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,5346.07,241289
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,5220.58,247088
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,5210.14,247584
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,4984.46,258793
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,4923.73,261986
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,4962.91,259917
transA,transB,M,N,K,alpha,lda,ldb,beta,ldc,rocblas-Gflops,us
N,T,8640,8640,8640,1,8640,8640,0,8640,4886.94,263958
```

Note: the above measurement is collected with a GigaByte Z52 system with 2x2nd Gen EPYC + 8xMI50 with ROCm3.3

See End Notes

[AMD Public Use]

ROCm™ Validation Suite (RVS) Introduction

- The ROCm Validation Suite (RVS) is a system administration and cluster management tool for detecting and troubleshooting common problems affecting AMD GPU(s) running in a high-performance computing environment. RVS is enabled using the ROCm software stack on a compatible platform
- The RVS focuses on software and system configuration issues, diagnostics, topological concerns, and relative systems performance

1. Deployment and Software Issues
2. Hardware Issues and Diagnostics
3. Integration Issues
4. System Stress Checks
5. Troubleshooting
6. Integration into Cluster Scheduler and Cluster Management Applications
7. Help Reduce Downtime and Failed GPU jobs

AMD

[AMD Public Use]

RVS Installation

- Linux® System Support Only
- RVS is Open Source Code

- Ex: ubuntu 18.04 command line

  #git clone https://github.com/ROCm-Developer-Tools/ROCmValidationSuite.git

- Detail configure and build RVS - reference below website

  https://github.com/ROCm-Developer-Tools/ROCmValidationSuite

AMD

[AMD Public Use]

TensorFlow installation: TF-ROCm2.2.0-beta1

1

Install other relevant ROCm packages

sudo apt update

sudo apt install rocm-libs miopen-hip rccl

2

Install TensorFlow (via the Python Package Index)

sudo apt install wget python3-pip

pip3 install --user tensorflow-rocm

Reference: https://rocmdocs.amd.com/en/latest/Deep_learning/Deep-learning.html#tensorflow

Note: some prerequisites libraries need to be installed first such as

$ sudo apt-get install python3-pip

$ sudo pip3 install -U pip

AMD

[AMD Public Use]

Basic Tensorflow Benchmark: CNN-ResNet50

1

Clone from Github

git clone https://github.com/tensorflow/benchmarks.git

2

Pull the Docker® Container

(install docker if necessary following steps @ https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04)

docker pull rocm/tensorflow:rocm3.3-tf1.15-dev

3

Run the Container in Detached mode (will *generate_ID*)

sudo docker run -d -it --network=host -v $HOME:/data --security-opt seccomp=unconfined -v $HOME/dockerx:/dockerx -v /data/imagenet-inception:/imagenet --privileged --device=/dev/kfd --device=/dev/dri --group-add video --cap-add=SYS_PTRACE rocm/tensorflow:rocm3.3-tf1.15-dev

4

Attach to the container with the output ID

docker attach generate_ID

AMD

[AMD Public Use]

Tensorflow Benchmark: CNN-ResNet50 (Cont.)

5

Navigate to the Benchmarks

cd /data/benchmarks/scripts/tf_cnn_benchmarks

6

Run ResNet50 with synthetic data w/o distortions with 1xGPU

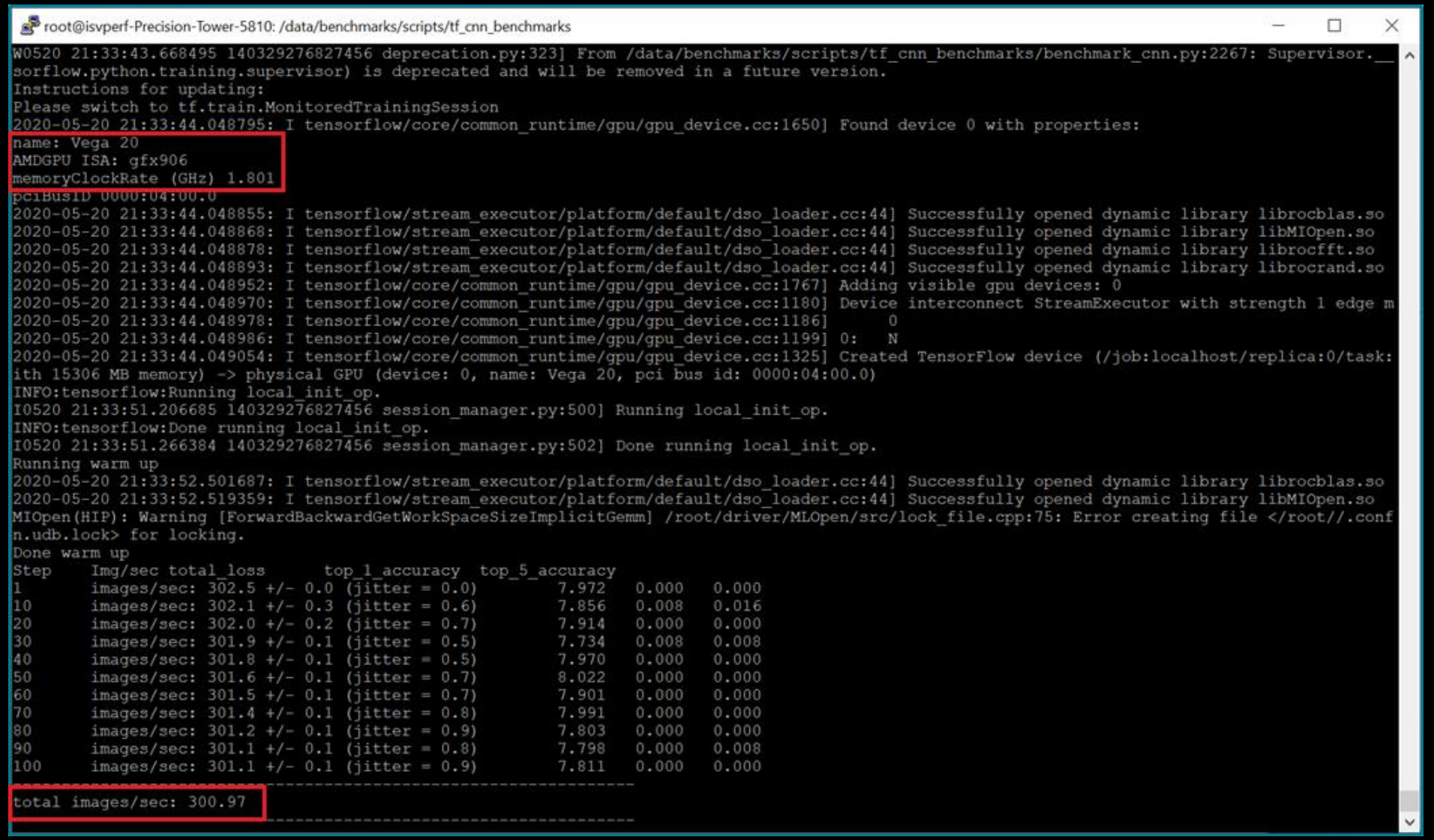
python3 tf_cnn_benchmarks.py --model=resnet50 --batch_size=128 --print_training_accuracy=True --variable_update=parameter_server --local_parameter_device=gpu --num_gpus=1

model: Model to use, e.g. resnet50, inception3, vgg16, and alexnet
num_gpus: Number of GPUs to use
data_dir: Path to data to process. If not set, synthetic data is used
batch_size: Batch size for each GPU
variable_update: The method for managing variables: parameter_server ,replicated, distributed_replicated, independent
local_parameter_device: Device to use as parameter server: cpu or gpu

AMD

[AMD Public Use]

Tensorflow CNN-ResNet50 Screen Capture



See End Notes

[AMD Public Use]

PyTorch Installation – Docker® Image

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”*
2. Obtain Docker image
   - *docker pull rocm/pytorch:rocm3.0_ubuntu16.04_py3.6_pytorch*
3. Clone PyTorch repository on the host
   - *cd ~*
   - *git clone https://github.com/pytorch/pytorch.git*
   - *cd pytorch*
   - *git submodule init*
   - *git submodule update*

AMD

[AMD Public Use]

PyTorch Installation – Build

4 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:rocm3.0_ubuntu16.04_py3.6_pytorch

5 Build PyTorch

cd /data/pytorch
.jenkins/pytorch/build.sh

6 Confirm working installation

PYTORCH_TEST_WITH_ROCM=1 python3.6 test/run_test.py –verbose

7 Install Torchvision & Commit container to preserve Pytorch install

pip install torchvision
sudo docker commit <container_id> -m 'pytorch installed'

AMD

[AMD Public Use]

Basic PyTorch Benchmark – ResNet50

1. Pull the Docker® Image

   docker pull rocm/pytorch:rocm3.3_ubuntu16.04_py3.6_pytorch

2. Run the Docker Container

   alias ptdrun='sudo docker run -it --network=host --device=/dev/kfd --device=/dev/dri --group-add video --cap-add=SYS_PTRACE --security-opt seccomp=unconfined -v $HOME/dockerx:/dockerx --shm-size=64G'
   ptdrun rocm/pytorch:rocm3.3_ubuntu16.04_py3.6_pytorch

3. In docker container, install dependencies and download py script

   *cd ~ && mkdir -p pt-micro-bench && cd pt-micro-bench && rm -rf * && wget https://www.dropbox.com/s/0kh1y41xzx4v8tq/micro_benchmarking_pytorch.py && wget https://raw.githubusercontent.com/wiki/ROCmSoftwarePlatform/pytorch/fp16util.py*
   pip3.6 install torchvision==0.6.0 --no-dependencies

AMD

[AMD Public Use]

PyTorch ResNet50 Benchmark - Screenshot

4

Run ResNet50 Training

export ROCR_VISIBLE_DEVICES=0

python3.6 micro_benchmarking_pytorch.py --network resnet50 --batch-size 128

```
root@isvperf-Precision-Tower-5810:~/pt-micro-bench# python3.6 micro_benchmarking_pytorch.py --network resnet50 --batch-size 128
/root/.local/lib/python3.6/site-packages/torch/cuda/__init__.py:87: UserWarning:
    Found GPU0 Device 66af which is of cuda capability 3.0.
  warnings.warn(old_gpu_warn % (d, name, major, capability[1]))
INFO: running forward and backward for warmup.
INFO: running the benchmark..
OK: finished running benchmark..
--------------------SUMMARY--------------------------
Microbenchmark for network : resnet50
Mini batch size [img] : 128
Time per mini-batch : 0.45507651567459106
Throughput [img/sec] : 281.27138094624996
```


See End Notes

AMD

[AMD Public Use]

rocFFT

- rocFFT is a software library for computing Fast Fourier Transforms (FFT) written in HIP
  - https://github.com/ROCmSoftwarePlatform/rocFFT
- To build the rocfft-rider test, we need to build from source using the flag: `-DBUILD_CLIENTS_TESTS=on`
- Installation
  - `sudo apt -y install libboost-program-options-dev libfftw3-dev`
  - `cd ~`
  - `git clone https://github.com/ROCmSoftwarePlatform/rocFFT.git`
  - `cd rocFFT`
  - `mkdir build; cd build`
  - `cmake .. -DCXX=/opt/rocm/bin/hipcc -DBUILD_CLIENTS_BENCHMARKS=ON -DBUILD_CLIENTS_RIDER=ON -DBUILD_CLIENTS_TESTS=on`
  - `make -j`
- Executables will be in
  - `~/rocFFT/build/clients/staging`
- Run tests with the rocfft-rider benchmark executable. For example:
  - `./rocfft-rider --length $(( 2 ** 24 )) -b 10`
    - `in-place`
    - `Running profile with 1 samples`
    - `length: 16777216`
    - `Execution gpu time: 44.3606 ms`
    - `Execution gflops: 453.841`

AMD

[AMD Public Use]

MPI and UCX

- Using the installation script to install Open MPI with UCX
  - `setup_rocm_ompi_ucx.sh`
- Run as root
  - `sudo su -`
  - `cd /opt/rocm`
  - `./setup_rocm_ompi_ucx.sh true`
- Expected performance
  - XGMI between 2 GPUs
    - 36GB/s bandwidth at 2MB messages
    - 1.8us latency at 1-byte
- For InfiniBand setup
  - Install MLNX_OFED before ROCm install to ensure PeerDirect support is in place for Mellanox drivers
- More info for Open MPI + UCX
  - https://github.com/openucx/ucx/wiki/Build-and-run-ROCM-UCX-OpenMPI
- MPICH support with UCX for AMD GPU also available. To enable MPICH with ROCm-enabled UCX:
  - `./configure --with-device=ch4:ucx --with-ucx=<path/to/ucx/install>`

```
/opt/mpi/ompi/bin/mpirun -np 2 -x UCX_RNDV_THRESH=8192 --mca osc ucx --mca spml
ucx -x LD_LIBRARY_PATH -x UCX_LOG_LEVEL=TRACE_DATA --allow-run-as-root -mca pml
ucx -x UCX_TLS=sm,self,rocm_copy,rocm_ipc,rocm_gdr osu/mpi/pt2pt/osu_bw -d rocm
D D 0 1
--------------------------------------------------------------------------
WARNING: There is at least non-excluded one OpenFabrics device found,
them).  This is most certainly not what you wanted.  Check your
cables, subnet manager configuration, etc.  The openib BTL will be
ignored for this job.

  Local host: ts1-sjc2-03
--------------------------------------------------------------------------
# OSU MPI-ROCM Bandwidth Test v5.3.2
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size      Bandwidth (MB/s)
1                       0.79
2                       0.88
4                       1.63
8                       3.54
16                     12.81
32                     24.41
64                     38.22
128                   131.20
256                   211.42
512                   300.58
1024                  251.70
2048                  261.51
4096                  247.15
8192                  645.69
16384                1313.81
32768                2562.81
65536                4734.22
131072               9717.29
262144              16521.49
524288              25106.44
1048576             32802.49
2097152             36713.70
4194304             34026.49
8388608             35444.23
16777216            34806.10
33554432            34081.91
67108864            33965.73
134217728           33906.72
root@ts1-sjc2-03:/opt/mpi#
```

```
/opt/mpi/ompi/bin/mpirun -np 2 -x UCX_RNDV_THRESH=8192 --mca osc ucx --mca spml
ucx -x LD_LIBRARY_PATH -x UCX_LOG_LEVEL=TRACE_DATA --allow-run-as-root -mca pml
ucx -x UCX_TLS=sm,self,rocm_copy,rocm_ipc,rocm_gdr osu/mpi/pt2pt/osu_latency -d
rocm D D 0 1
--------------------------------------------------------------------------
WARNING: There is at least non-excluded one OpenFabrics device found,
them).  This is most certainly not what you wanted.  Check your
cables, subnet manager configuration, etc.  The openib BTL will be
ignored for this job.

  Local host: ts1-sjc2-03
--------------------------------------------------------------------------
# OSU MPI-ROCM Latency Test v5.3.2
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size          Latency (us)
0                     0.23
1                     1.78
2                     2.80
4                     2.78
8                     2.84
16                    1.78
32                    1.78
64                    1.86
128                   1.52
256                   1.82
512                   2.34
1024                  4.49
2048                  8.30
4096                 17.03
8192                 23.87
16384                24.33
32768                27.88
65536                34.26
131072               47.12
262144               73.91
524288              130.76
1048576             237.76
2097152             455.48
4194304             925.59
8388608            1794.63
16777216           3532.27
33554432           7000.58
67108864          13932.70
134217728         27765.25
root@ts1-sjc2-03:/opt/mpi#
```

[AMD Public Use]

rocHPCG

- rocHPCG is the implementation of HPCG that runs on AMD GPU:
  - https://github.com/ROCmSoftwarePlatform/rocHPCG.git
- To build rocHPCG
  - `git clone https://github.com/ROCmSoftwarePlatform/rocHPCG.git`
  - `cd rocHPCG`
  - `./install.sh`
- The executable will be located at
  - `rocHPCG/build/release/bin/rochpcg`
- The local domain size to run for a 16GB GPU should be “280 280 280”
- A qualified HPCG run would run for 30 minutes
- `rocHPCG/build/release/bin/rochpcg 280 280 280 1860`
  - `DDOT   =   115.8 GFlop/s (  926.2 GB/s)     115.8 GFlop/s per process (  926.2 GB/s per process)`
  - `WAXPBY =    56.9 GFlop/s (  683.1 GB/s)      56.9 GFlop/s per process (  683.1 GB/s per process)`
  - `SpMV   =   112.7 GFlop/s (  710.0 GB/s)     112.7 GFlop/s per process (  710.0 GB/s per process)`
  - `MG     =   159.0 GFlop/s ( 1227.5 GB/s)     159.0 GFlop/s per process ( 1227.5 GB/s per process)`
  - **`Total  =   145.1 GFlop/s ( 1100.1 GB/s)     145.1 GFlop/s per process ( 1100.1 GB/s per process)`**
  - `Final  =   143.7 GFlop/s ( 1089.7 GB/s)     143.7 GFlop/s per process ( 1089.7 GB/s per process)`

AMD

[AMD Public Use]

BabelStream

- BabelStream measures memory transfer rates to/from global device memory on GPUs
- This benchmark is similar in spirit, and based on, the STREAM benchmark for CPUs
- To build BabelStream
  - `git clone https://github.com/UoB-HPC/BabelStream.git`
  - `cd BabelStream; make VERBOSE=1 -f HIP.make`
- To run BabelStream
  - `./hip-stream`
  - `BabelStream`
  - `Version: 3.4`
  - `Implementation: HIP`
  - `Running kernels 100 times`
  - `Precision: double`
  - `Array size: 268.4 MB (=0.3 GB)`
  - `Total size: 805.3 MB (=0.8 GB)`
  - `Using HIP device Vega 20`
  - `Driver: 313700`

Function	MBytes/sec	Min (sec)	Max	Average
Copy	804349.192	0.00067	0.00068	0.00067
Mul	805412.280	0.00067	0.00068	0.00067
Add	775833.239	0.00104	0.00104	0.00104
Triad	**774988.060**	**0.00104**	**0.00104**	**0.00104**
Dot	553257.856	0.00097	0.00099	0.00098

[AMD Public Use]

LAMMPS

- LAMMPS is a popular molecular dynamics simulation application
- LAMMPS has ‘`gpu`’ and ‘`kokkos`’ backends to support AMD GPU. The ‘`gpu`’ backend is shown below.
- Install rocPRIM and hipCUB:
  - `sudo apt install rocprim hipcub`
- Clone the repo:
  - `git clone https://github.com/lammps/lammps.git`
- Get cub 1.8.0 and add it to the LAMMPS libraries:
  - `wget https://github.com/NVlabs/cub/archive/1.8.0.zip`
  - `unzip 1.8.0.zip; mv cub-1.8.0/ lammps/lib/gpu/`
- Edit HIP_ARCH in lammps/lib/gpu/Makefile.hip
  - set `HIP_ARCH = gfx906` for MI50
- Set the following environment variable:
  - `export HIP_PLATFORM=hcc`
  - `cd lammps/lib/gpu; make -f Makefile.hip -j`
  - `Cd lammps/src; make yes-gpu; make hip -j`
- Run the example, in examples/melt or bench/KEPLER:
  - `mpirun -np 1  ../../src/lmp_hip -in in.melt -sf gpu -pk gpu 1`

AMD

[AMD Public Use]

GROMACS

- GROMACS - GROningen MAchine for Chemical Simulations
- Molecular dynamics package mainly designed for simulations of proteins, lipids, and nucleic acids
- The current hipified GROMACS source is in a private repository, enable by request
  - https://github.com/ROCmSoftwarePlatform/Gromacs.git
- Build instructions:
  - `git clone https://github.com/ROCmSoftwarePlatform/Gromacs.git`
  - `cd Gromacs; git checkout develop-2020.1`
  - `mkdir build`
  - `cd build`
  - `rm -rf ../build/*`
  - `cmake -DBUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=Release -DCMAKE_C_COMPILER=mpicc -DCMAKE_CXX_COMPILER=mpicxx -DGMX_MPI=on -DGMX_GPU=on -DGMX_GPU_USE_AMD=on -DGMX_OPENMP=on -DGMX_GPU_DETECTION_DONE=on -DGMX_SIMD=AVX2_256 -DREGRESSIONTEST_DOWNLOAD=OFF -DCMAKE_PREFIX_PATH=/opt/rocm -DCMAKE_INSTALL_PREFIX=$HOME/MI50 ..`
  - `make -j install`
- Alternatively, we can use the GROMACS rocmx docker:
  - https://hub.docker.com/r/rocmx/gromacs
  - `sudo docker run -it -d --network=host -v $HOME:/data --device=/dev/kfd --device=/dev/dri --security-opt seccomp=unconfined --group-add video --name gromacs_docker_script rocmx/hpc:rocm_3.3_hpc_gromacs_2020.1_a`

[AMD Public Use]

NAMD

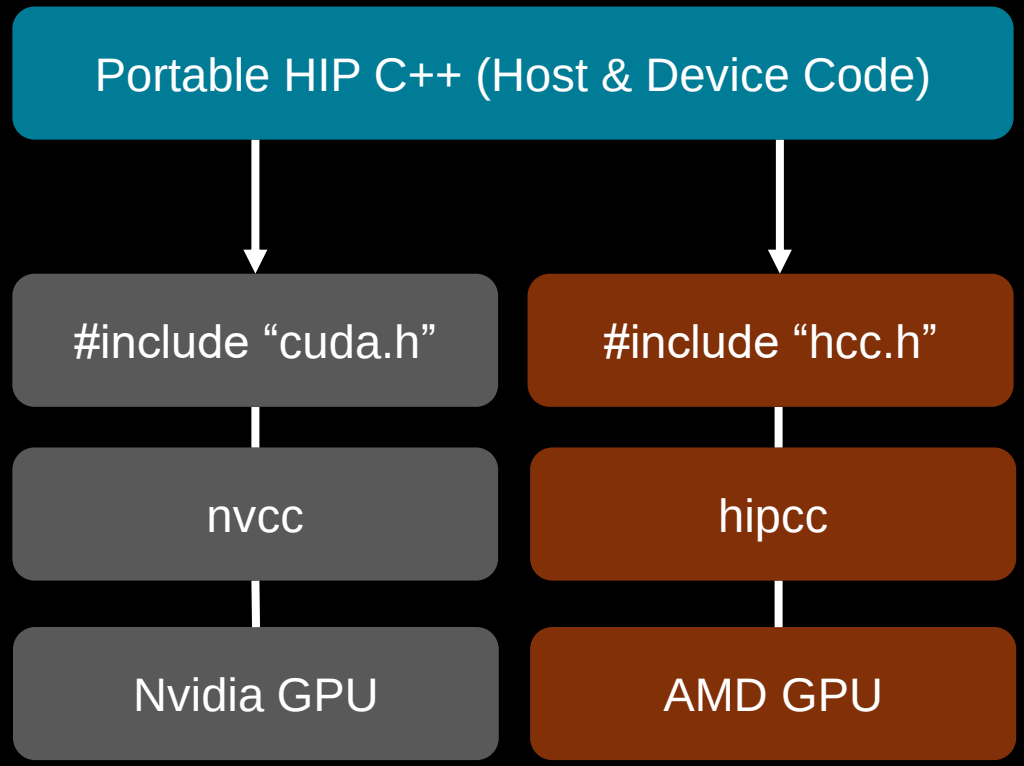
- NAnoscale Molecular Dynamics (NAMD)
  - NAMD is a highly scalable molecular dynamics (MD) code
  - NAMD geared towards the simulation of large biomolecular systems
- The current hipified NAMD source is in a private repository, enable by request
  - https://github.com/ROCmSoftwarePlatform/NAMD
- We can use the NAMD docker to run NAMD
  - `docker run -it --privileged  --device=/dev/kfd --device=/dev/dri/  --cap-add=SYS_RAWIO --device=/dev/mem --group-add video --network host japarada/ubuntu-18.04_namd:rocm-3.3_0416`
  - `cd ~/NAMD/NAMD_benchmarks/`
  - `source ~/namd_hip.rc`
  - `python3 run_benchmarks.py -b apoa1 stmv -c 16-16 -d 0 # 1x GPU`
  - `python3 run_benchmarks.py -b apoa1 stmv -c 16-16 -d 0,1,2,3,4,5,6,7 # 8x GPUs`

[AMD Public Use]

HIP: High Performance & Portable

C++ runtime API and kernel language that allows developers to create portable applications that can run on AMD's accelerators as well as CUDA devices.

- Is open-source
- Provides an API for an application to leverage GPU acceleration for both AMD and CUDA devices
- Syntactically similar to CUDA. Most CUDA API calls can be converted in place: cuda -> hip
- Supports a strong subset of CUDA runtime functionality



[AMD Public Use]

Getting Started with HIP

CUDA VECTOR ADD

```
__global__ void add(int n, double *x, double *y)
{
  int index = blockIdx.x * blockDim.x + threadIdx.x;
  int stride = blockDim.x * gridDim.x;
  for (int i = index; i < n; i += stride)
    {
    y[i] = x[i] + y[i];
    }
}
```

HIP VECTOR ADD

```
__global__ void add(int n, double *x, double *y)
{
  int index = blockIdx.x * blockDim.x + threadIdx.x;
  int stride = blockDim.x * gridDim.x;
  for (int i = index; i < n; i += stride)
    {
    y[i] = x[i] + y[i];
    }
}
```

KERNELS ARE SYNTACTICALLY IDENTICAL

AMD

[AMD Public Use]

Seamless Porting from CUDA APIs

CUDA

```
cudaMemcpyAsync(d_npos,h_npos,sizeof(float4)*SIZE,cudaMemcpyHostToDevice,stream);

cudaMemcpyAsync(d_mask,h_mask,sizeof(MASK_T)*cnt,cudaMemcpyHostToDevice,stream);

calcHHCullenDehnen<<<blocksPerGrid, threadsPerBlock, 0, stream>>>(cnt, SIZE, d_npos, d_mask, rsm);

cudaMemcpyAsync(h_pos,d_npos+(SIZE-cnt),sizeof(float4)*cnt,cudaMemcpyDeviceToHost,stream);

cudaMemcpyAsync(h_mask,d_mask,sizeof(MASK_T)*cnt,cudaMemcpyDeviceToHost,stream);
```

HIP

```
hipMemcpyAsync(d_npos,h_npos,sizeof(float4)*SIZE,hipMemcpyHostToDevice,stream);

hipMemcpyAsync(d_mask,h_mask,sizeof(MASK_T)*cnt,hipMemcpyHostToDevice,stream);

 hipLaunchKernelGGL((calcHHCullenDehnen), dim3(blocksPerGrid), dim3(threadsPerBlock), 0, stream, cnt, SIZE, d_npos, d_mask, rsm);

hipMemcpyAsync(h_pos,d_npos+(SIZE-cnt),sizeof(float4)*cnt,hipMemcpyDeviceToHost,stream);

hipMemcpyAsync(h_mask,d_mask,sizeof(MASK_T)*cnt,hipMemcpyDeviceToHost,stream);
```

[AMD Public Use]

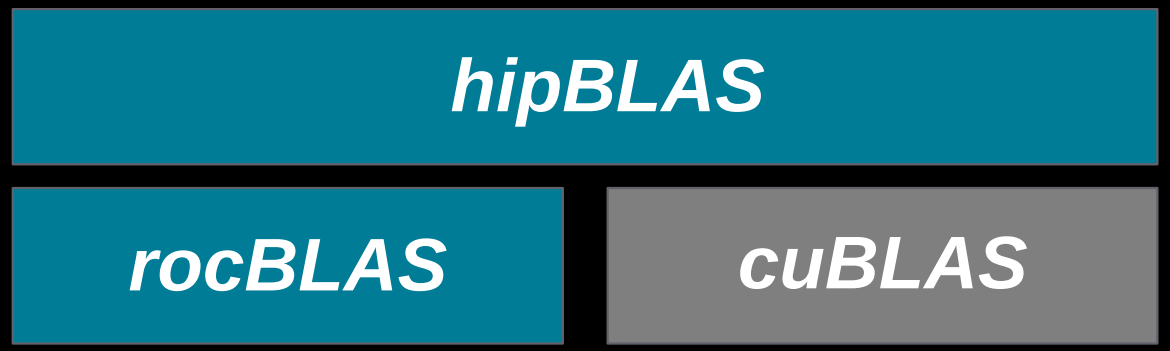
AMD GPU Libraries

A note on naming conventions:

- roc* -> AMDGCN library usually written in HIP
- cu* -> NVIDIA PTX libraries
- hip* -> usually interface layer on top of roc*/cu* backends

hip* libraries:

- Can be compiled by hipcc and can generate a call for the device you have:
  - hipcc->AMDGCN
  - hipcc->nvcc (inlined)->NVPTX



[AMD Public Use]

CUDA Equivalent Libraries

CUDA Library	ROCm Library	Comment
cuBLAS	rocBLAS	Basic Linear Algebra Subroutines
cuFFT	rocFFT	Fast Fourier Transfer Library
cuSPARSE	rocSPARSE	Sparse BLAS + SPMV
cuSolver	rocSolver	Lapack Library
AMG-X	rocALUTION	Sparse iterative solvers & preconditioners with Geometric & Algebraic MultiGrid
Thrust	hipThrust	C++ parallel algorithms library
CUB	rocPRIM	Low Level Optimized Parallel Primitives
cuDNN	MIOpen	Deep learning Solver Library
cuRAND	rocRAND	Random Number Generator Library
EIGEN	EIGEN – HIP port	C++ template library for linear algebra: matrices, vectors, numerical solvers
NCCL	RCCL	Communications Primitives Library based on the MPI equivalents

AMD

HIPIFY Tools:

Converting CUDA Code for Portability

Hipify-perl

- Easy to use –point at a directory and it will attempt to hipify CUDA code
- Very simple string replacement technique: may make incorrect translations
- sed -e ‘s/cuda/hip/g’, (e.g., cudaMemcpy becomes hipMemcpy)
- Recommended for quick scans of projects

Hipify-clang

- Requires clang compiler to parse files and perform semantic translation
- More robust translation of the code
- Generates warnings and assistance for additional analysis
- High quality translation, particularly for cases where the user is familiar with the make system

[AMD Public Use]

Getting QUDA Rocking with HIP

nag®

Experts in numerical software and High Performance Computing

HIP Solutions

- QUDA depends on many CUDA libraries
  - Eigen - **hip support in Eigen's development branch,** CuFFT - **rocFFT + hipFFT,** CuBLAS - **rocBLAS + hipBLAS,** CuRAND - **rocRAND + hipRAND,** Thrust – **hipThrust,** CUB - **hipCUB**
- QUDA is a large project (For a single-person porting project!)
  - 10000 lines of hand-tuned CUDA kernels - **hipify converted these without problems**
  - 35000 lines of header code - **hipify mostly converted these, but needed manual switch to new library dependencies**
  - 74000 lines of library code - **Mostly successful hipify conversion, required some manual changes**
  - 34000 lines of test suite code – **hipify converted without problems**
  - Heavily interconnected due to template use – **No solution to this, it is part of library design**

Results

- Time to running executable on AMD hardware
  - 15 developer-days
- Without AMD tools (hip) and library support, would have taken *significantly longer*.
- Work ongoing for rest of test suite

nag High Performance Computing Consulting | Numerical Algorithms | Software Engineering Services | www.nag.com

AMD

[AMD Public Use]

hipify-perl

- hipify-perl is autogenerated perl-based script which heavily uses regular expressions
- Advantages
  - Ease in use
  - No need to check the input source CUDA code for correctness
  - No dependencies on 3rd party tools, including CUDA
- Disadvantages
  - Limitation in transforming the following constructs
    - macros expansion
    - namespaces
      - redefines of CUDA entities in user namespaces
      - using directive
    - templates (some cases)
    - device/host function calls distinguishing
    - header files correct injection
    - complicated argument lists parsing
- Available in ROCm install:
  - `/opt/rocm/bin/hipify-perl`
  - Convert all files in a directory
  - `/opt/rocm/bin/hipconvertinplace.sh`

AMD

[AMD Public Use]

hipify-clang

- hipify-clang is a clang-based tool for translation CUDA sources into HIP sources
- Translates CUDA source into abstract syntax tree, then traversed by transformation matchers
- After applying all the matchers, the output HIP source is produced
- Advantages:
  - It is a translator, therefore complicated constructs can be parsed successfully, or an error will be reported
  - Supports clang options like -I, -D, --cuda-path, etc
  - Seamless support of new CUDA versions for LLVM Clang
  - Easier to support
- Disadvantages:
  - CUDA should be installed and provided in case of multiple installations by --cuda-path option
  - The input CUDA code needs to be compliable. Incorrect code cannot be translated to HIP
  - Include's and define's should be provided to transform code successfully
- Available in ROCm repo for download:
  - `apt install hipify-clang`
  - Or build from HIPIFY github
  - https://github.com/ROCm-Developer-Tools/HIPIFY

AMD

[AMD Public Use]

Hipify Samples

```
root@ts1-sjc2-03:/opt/rocm/hip/samples/0_Intro/square# cat README.md
# Square.md

Simple test which shows how to use hipify-perl to port CUDA code to HIP.
See related [blog](http://gpuopen.com/hip-to-be-squared-an-introductory-hip-tutorial) that explains the example.
Now it is even simpler and requires no manual modification to the hipified source code - just hipify and compile:

1. Add hip/bin path to the PATH  :
    <code>export PATH=$PATH:[MYHIP]/bin</code>

2. <code>$ make </code>
   Make runs these steps.  This can be performed on either CUDA or AMD platform:
   <code>hipify-perl square.cu > square.cpp </code>    # convert cuda code to hip code
   <code>hipcc square.cpp</code>                       # compile into executable
```

- HIP Samples
  - In `/opt/rocm/hip/samples/0_Intro/square`
- SpecFEM3D Cartesian
  - Fortran code base with one C file to abstract GPU stubs
  - Very clean GPU implementation with all 18 *.cu files contained in one directory:/specfem3d/src/cuda
  - Porting process: (10 min)
    - `~/specfem3d/src$ hipconvertinplace-perl.sh`
  - Minor build changes:
    - Makefile and configuration work – 100 line section to modify and add AMD support
  - Converted 1120 CUDA->HIP refs in 16783 Lines of Code
    - with 1 warning: comment containing the word “CUDA”
    - ~500 were memory management (hipMemcpy, hipFree, hipMemcpyHostToDevice, hipMalloc, hipMemcpyDeviceToHost, hipMemcpy2D, hipMemset, etc)
    - ~250 numeric literal operations
  - 86 kernel launches of 15 separate kernels:

compute_acoustic_vectorial_seismogram_kernel	store_dataT
compute_kernels_hess_ac_cudakernel	kernel_3_acoustic_cuda_device
noise_read_add_surface_movie_cuda_kernel	Kernel_2_noatt_iso_impl
add_sources_el_SIM_TYPE_2_OR_3_kernel	enforce_free_surface_cuda_kernel
assemble_boundary_potential_on_device	compute_coupling_elastic_ac_kernel
compute_stacey_acoustic_kernel	check_array_ispec_kernel
add_sources_ac_SIM_TYPE_2_OR_3_kernel	compute_stacey_elastic_sim3_kernel
assemble_boundary_accel_on_device	

[AMD Public Use]

General information and resources

- ROCm Installation Guide: https://rocmdocs.amd.com/en/latest/Installation_Guide/Installation-Guide.html
- ROCm platform: https://github.com/RadeonOpenCompute/ROCm/
  - With instructions for installing from binary repositories, and links to source repositories for all components
- HIP porting guide: https://github.com/ROCm-Developer-Tools/HIP/blob/master/docs/markdown/hip_porting_guide.md
- ROCm/HIP libraries: https://github.com/ROCmSoftwarePlatform
- rocprofiler: https://github.com/ROCm-Developer-Tools/rocprofiler
  - Collects application traces and performance counters
  - Trace timeline can be visualized with chrome://tracing
- AMD GPU ISA docs: https://developer.amd.com/resources/developer-guides-manuals
- YouTube videos
  - Includes YouTube videos on ROCm software, programming concepts and more details on hardware devices
  - https://community.amd.com/community/radeon-instinct-accelerators/blog/2020/06/10/rocm-open-software-ecosystem-for-accelerated-compute

AMD

[AMD Public Use]

References

- **ROCm™ Installation**

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

- **ROCm Bandwidth Test**

https://github.com/RadeonOpenCompute/rocm_bandwidth_test

- **RVS Installation**

https://github.com/ROCm-Developer-Tools/ROCmValidationSuite

- **TensorFlow Installation & Benchmark**

https://rocmdocs.amd.com/en/latest/Deep_learning/Deep-learning.html#tensorflow

https://github.com/tensorflow/benchmarks/tree/master/scripts/tf_cnn_benchmarks

- **PyTorch Installation & Benchmark**

https://rocmdocs.amd.com/en/latest/Deep_learning/Deep-learning.html#pytorch

https://github.com/ROCmSoftwarePlatform/pytorch/wiki/Performance-analysis-of-PyTorch

- **Link to more training information**

https://community.amd.com/community/radeon-instinct-accelerators/blog/

AMD