Overview of ROCm and Compilers

Samuel Antao
STMS, LUMI CoE Lead
Data Center GPU, AMD

AMD @HLRS

AMD
together we advance_

Thanks to all the AMD staff for their contributions

Suyash Tandon
Justin Chang
Julio Maia
Noel Chalmers
Paul T. Bauman
Nicholas Curtis
Nicholas Malaya
Alessandro Fanfarillo
Jose Noudohouenou
Chip Freitag
Damon McDougall
Noah Wolfe
Jakub Kurzak
Samuel Antao
George Markomanolis
Bob Robey
Gina Sitaraman

Agenda

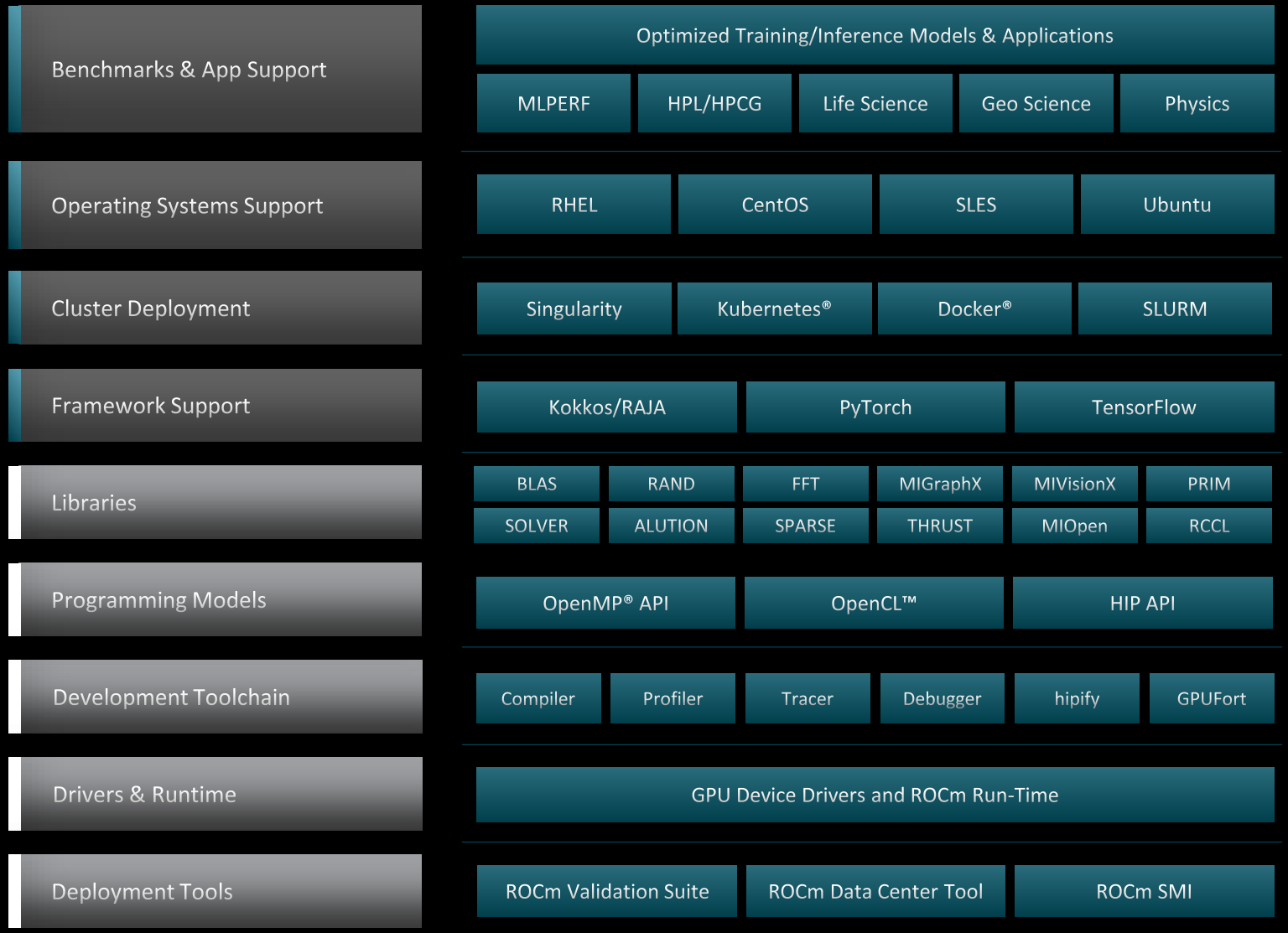
1. ROCm Software Ecosystem
2. Compilers for AMD GPUs

2. ROCm Software Ecosystem

Open Software Platform For GPU Compute

AMD ROCm

- Unlocked GPU Power To Accelerate Computational Tasks
- Optimized for HPC and Deep Learning Workloads at Scale
- Open Source Enabling Innovation, Differentiation, and Collaboration



AMD ROCm 5.0

DEMOCRATIZING EXASCALE FOR ALL

EXPANDING SUPPORT & ACCESS

- Support for Radeon Pro W6800 Workstation GPUs
- Remote access through the AMD Accelerator Cloud

OPTIMIZING PERFORMANCE

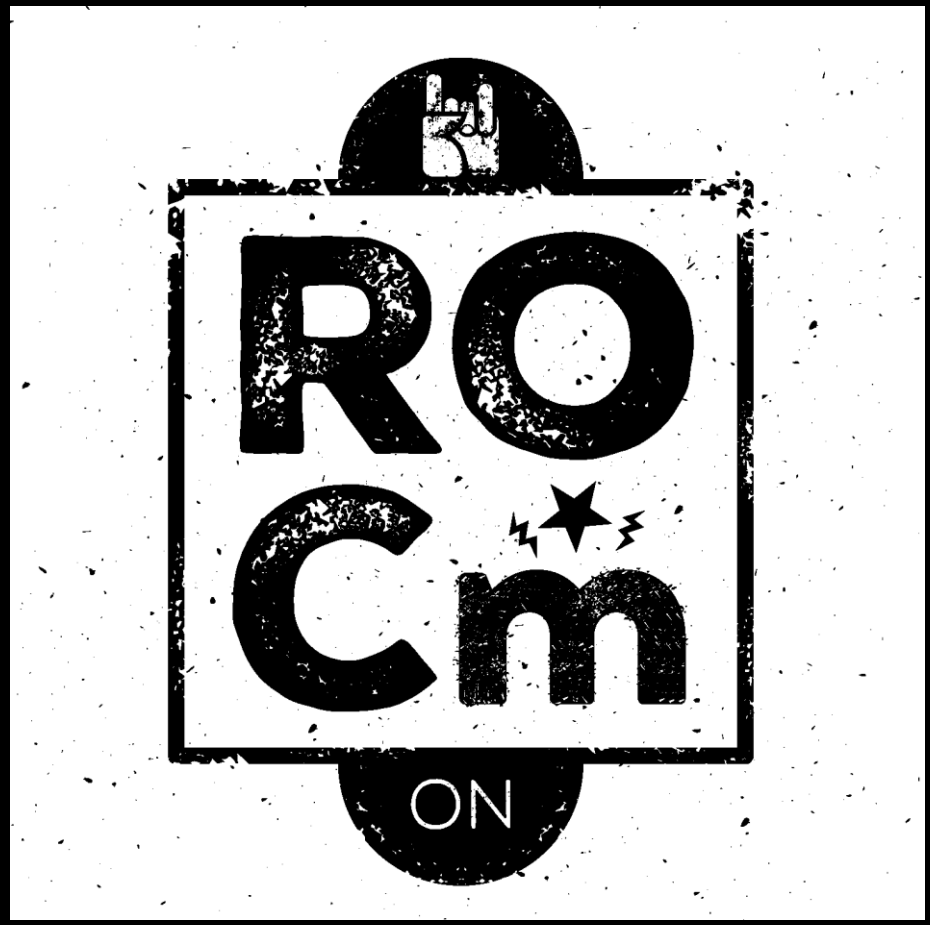
- MI200 Optimizations: FP64 Matrix ops, Improved Cache
- Improved launch latency and kernel performance

ENABLING DEVELOPER SUCCESS

- HPC Apps & ML Frameworks on AMD InfinityHub
- Streamlined and improved tools increasing productivity

ROCm Software Ecosystem

- **H**eterogeneous-compute **I**nterface for **P**ortability (HIP) is part of a larger software distribution called ROCm
- Install instructions and documentation:
  - https://rocm.docs.amd.com/en/latest/deploy/linux/quick_start.html
  - https://gpuopen.com/learn/amd-lab-notes/amd-lab-notes-rocm-installation-readme/
- The ROCm package provides libraries and programming tools for developing HPC and ML applications on AMD GPUs
- All the ROCm environment and the libraries are provided from the supercomputer, usually, there is no need to install something yourselves
- Heterogeneous System Architecture (HSA) runtime is an API that exposes the necessary interfaces to access and interact with the hardware driven by AMDGPU driver



ROCm GPU Libraries

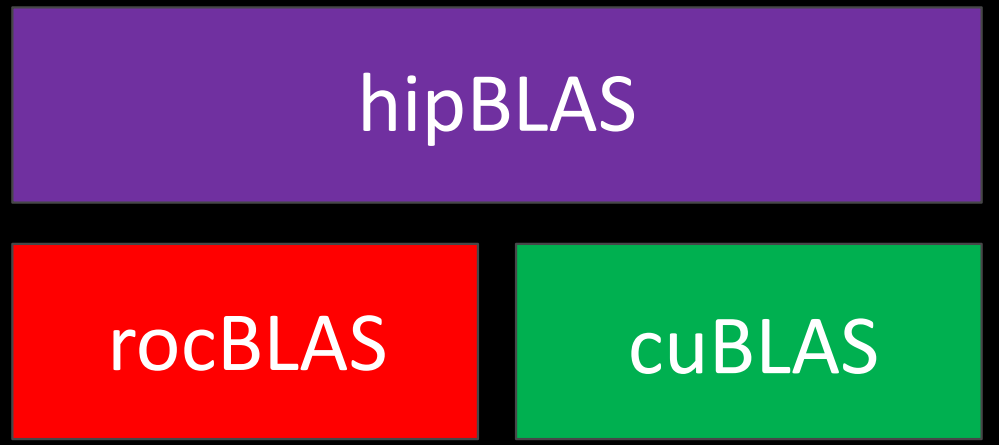
ROCm provides several GPU math libraries

- Typically, two versions:
  - roc* -> AMD GPU library, usually written in HIP
  - hip* -> Thin interface between roc* and Nvidia cu* library

When developing an application meant to target both CUDA and AMD devices, use the hip* libraries (portability)

When developing an application meant to target only AMD devices, may prefer the roc* library API (performance).

- Some roc* libraries perform **better** by using addition APIs not available in the cu* equivalents



AMD Math Library Equivalents: “Decoder Ring”

CUBLAS	**ROCBLAS**	Basic Linear Algebra Subroutines
CUFFT	**ROCFFT**	Fast Fourier Transforms
CURAND	**ROCRAND**	Random Number Generation
THRUST	**ROCTHRUST**	C++ Parallel Algorithms
CUB	**ROCPRIM**	Optimized Parallel Primitives

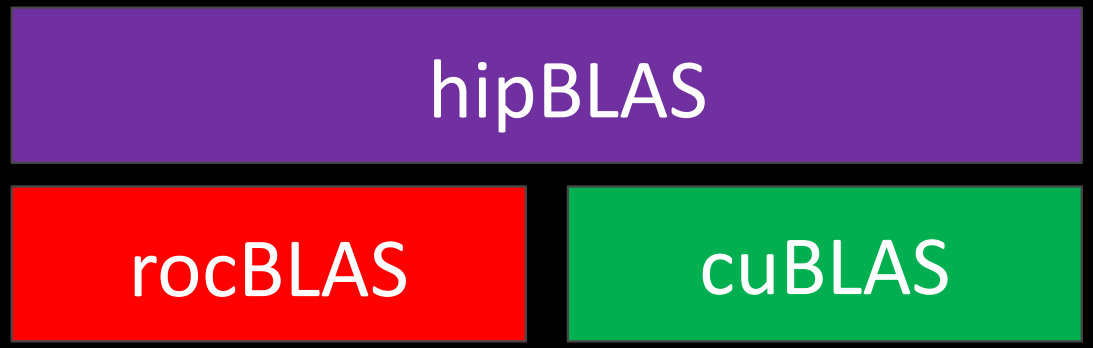
AMD Math Library Equivalents: “Decoder Ring”

CUSPARSE	**ROCSPARSE**	Sparse BLAS, SpMV, etc.
CUSOLVER	**ROCSOLVER**	Linear Solvers
AMGX	**ROCALUTION**	Solvers and preconditioners for sparse linear systems

GITHUB.COM/ROCM-DEVELOPER-TOOLS/HIP → HIP_PORTING_GUIDE.MD FOR A COMPLETE LIST

AMD GPU Libraries: BLAS

- rocBLAS – `sudo apt install rocblas`
  - Source code: https://github.com/ROCmSoftwarePlatform/rocBLAS
  - Documentation: https://rocblas.readthedocs.io/en/latest/
  - Basic linear algebra functionality
    - axpy, gemv, trsm, etc
  - Use hipBLAS if you need portability between AMD and NVIDIA devices
- hipBLAS - `sudo apt install hipblas`
  - Documentation: https://github.com/ROCmSoftwarePlatform/hipBLAS/wiki/Exported-functions
  - Use this if you need portability between AMD and NVIDIA
  - It is just a thin wrapper:
    - It can dispatch calls to rocBLAS for AMD devices
    - It can dispatch calls to cuBLAS for NVIDIA devices



AMD GPU Libraries: rocBLAS example

- rocBLAS
  - Documentation: https://rocblas.readthedocs.io/en/latest/
  - Level 1, 2, and 3 functionality
    - axpy, gemv, trsm, etc
  - Note: rocBLAS syntax matches BLAS closer than hipBLAS or cuBLAS
    - Use hipBLAS only if you need portability between AMD and NVIDIA devices
  - Link with: `-lrocblas`

```
#include <rocblas.h>

int main(int argc, char ** argv) {
  rocblas_int N = 500000;

  // Allocate device memory
  double * dx, * dy;
  hipMalloc(&dx, sizeof(double) * N);
  hipMalloc(&dy, sizeof(double) * N);

  // Allocate host memory (and fill up the arrays) here
  std::vector<double> hx(N), hy(N);

  // Copy host arrays to device
  hipMemcpy(dx, hx.data(), sizeof(double) * N, hipMemcpyHostToDevice);
  hipMemcpy(dy, hy.data(), sizeof(double) * N, hipMemcpyHostToDevice);

  const double alpha = 1.0;
  rocblas_handle handle;
  rocblas_create_handle(&handle);
  rocblas_status status;
  status = rocblas_daxpy(handle, N, &alpha, dx, 1, dy, 1);
  rocblas_destroy_handle(handle);

  // Copy result back to host
  hipMemcpy(hy.data(), dy, sizeof(double) * N, hipMemcpyDeviceToHost);
  hipFree(dx);
  hipFree(dy);
  return 0;
}
```

Some Links to Key Libraries

- BLAS
  - rocBLAS (https://github.com/ROCmSoftwarePlatform/rocBLAS)
  - hipBLAS (https://github.com/ROCmSoftwarePlatform/hipBLAS)
- FFTs
  - rocFFT (https://github.com/ROCmSoftwarePlatform/rocFFT)
  - hipFFT (https://github.com/ROCmSoftwarePlatform/hipFFT)
- Random number generation
  - rocRAND (https://github.com/ROCmSoftwarePlatform/rocRAND)
- Sparse linear algebra
  - rocSPARSE (https://github.com/ROCmSoftwarePlatform/rocSPARSE)
  - hipSPARSE (https://github.com/ROCmSoftwarePlatform/hipSPARSE)
- Iterative solvers
  - rocALUTION (https://github.com/ROCmSoftwarePlatform/rocALUTION)
- Parallel primitives
  - rocPRIM (https://github.com/ROCmSoftwarePlatform/rocPRIM)
  - hipCUB (https://github.com/ROCmSoftwarePlatform/hipCUB)

AMD Machine Learning Library Support

Machine Learning Frameworks:

- Tensorflow: https://github.com/ROCmSoftwarePlatform/tensorflow-upstream
- Pytorch: https://github.com/ROCmSoftwarePlatform/pytorch
- Caffe: https://github.com/ROCmSoftwarePlatform/hipCaffe

Machine Learning Libraries:

- MIOpen (similar to cuDNN): https://github.com/ROCmSoftwarePlatform/MIOpen
- Tensile (GEMM Autotuner): https://github.com/ROCmSoftwarePlatform/Tensile
- RCCL (ROCm analogue of NCCL): https://github.com/ROCmSoftwarePlatform/rccl
- Horovod (Distributed ML): https://github.com/ROCmSoftwarePlatform/horovod

Benchmarks:

- DeepBench: https://github.com/ROCmSoftwarePlatform/DeepBench
- MLPerf: https://mlperf.org

Usage of hipcc

Usage is straightforward. Accepts all/any flags that clang accepts, e.g.,

```
hipcc --offload-arch=gfx90a dotprod.cpp -o dotprod
```

Set HIPCC_VERBOSE=7 to see a bunch of useful information

- Compile and link lines
- Various paths

```
$ HIPCC_VERBOSE=7 hipcc --offload-arch=gfx90a dotprod.cpp -o dotprod
HIP_PATH=/opt/rocm-5.2.0
HIP_PLATFORM=amd
HIP_COMPILER=clang
HIP_RUNTIME=rocclr
ROCM_PATH=/opt/rocm-5.2.0
...
hipcc-args: --offload-arch=gfx90a dotprod.cpp -o dotprod
hipcc-cmd: /opt/rocm-5.2.0/llvm/bin/clang++ -stdc=c++11 -hc  -D__HIPCC__ -isystem /opt/rocm-
5.2.0/llvm/lib/clang/14.0.0/include
-isystem /opt/rocm-5.2.0/has/include -isystem /opt/rocm-5.2.0/include –offload-arch=gfx90a –O3 ...
```

- You can use also *`hipcc -v`* … to print some information
- With the command *`hipconfig`* you can see many information about environment variables declaration

OpenMP Offload GPU Support

- ROCm and AOMP
  - ROCm supports both HIP and OpenMP
  - AOMP: the AMD OpenMP research compiler, it is used to prototype the new OpenMP features for ROCm
- HPE Compilers
  - Provides offloading support to AMD GPUs, through OpenMP, HIP, and OpenACC (only for Fortran)
- GNU compilers:
  - Provide OpenMP and OpenACC offloading support for AMD GPUs
  - GCC 11: Supports AMD GCN gfx908
  - GCC 13: Supports AMD GCN gfx90a

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.

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.

Questions?


AMD