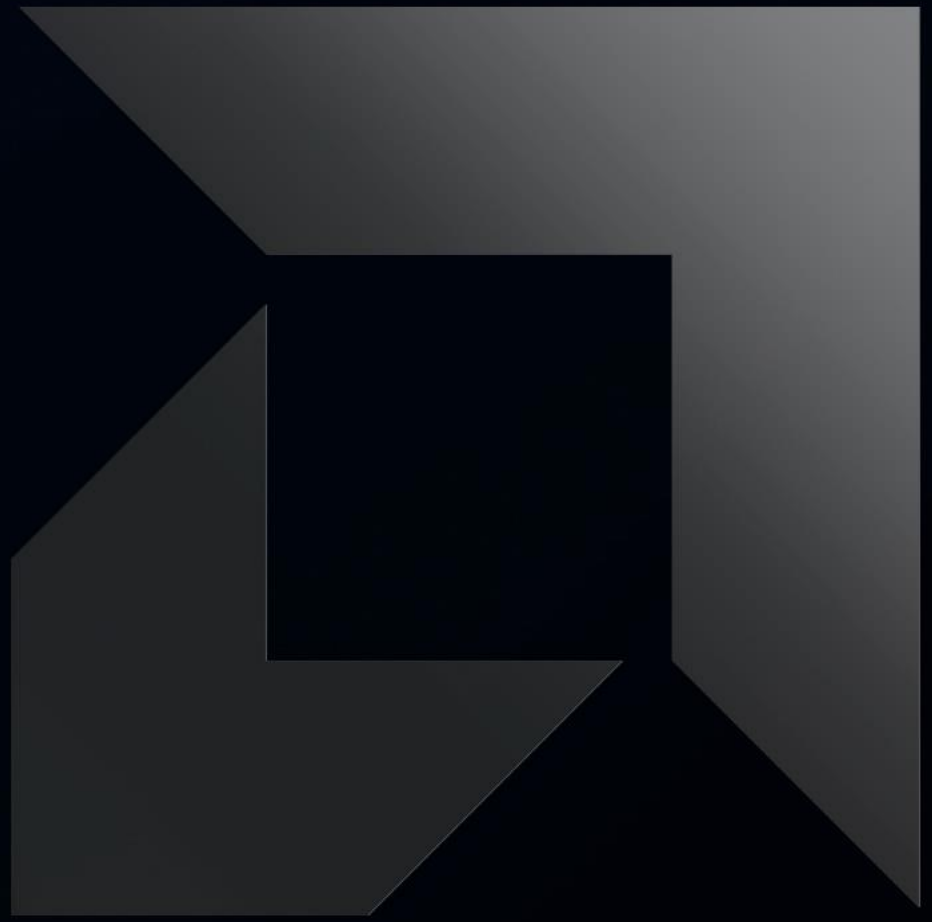


Application Analysis using Omniperf

Alessandro Fanfarillo, Thomas Gibson

AMD
together we advance_

Background: AMD Profiling Tools

ROC Profiler

- github.com/ROCm-Developer-Tools/rocprofiler
- Raw collection of GPU counters and traces
  - Counter collection driven by user-provided input files
- Counter collection output in CSV
- Supported tracing collection
  - HIP
  - HSA
  - GPU
- Traces can be visualized with Perfetto

Omnitrace

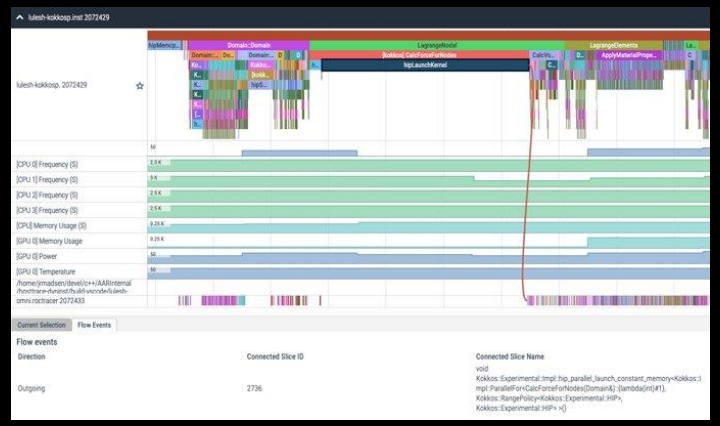
- github.com/AMDResearch/omnitrace
- Comprehensive trace collection and visualization of CPU + GPU activity
- Support for
  - HIP, HSA, GPU
  - OpenMP®
  - MPI
  - Kokkos
  - Pthreads
  - Multi-GPU
- Traces can be visualized with Perfetto

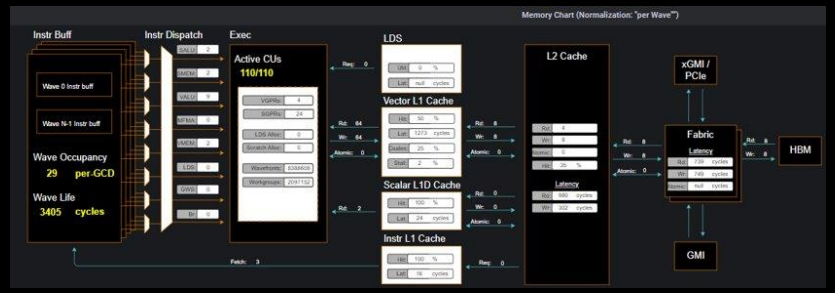
Focus of this talk

Omniperf

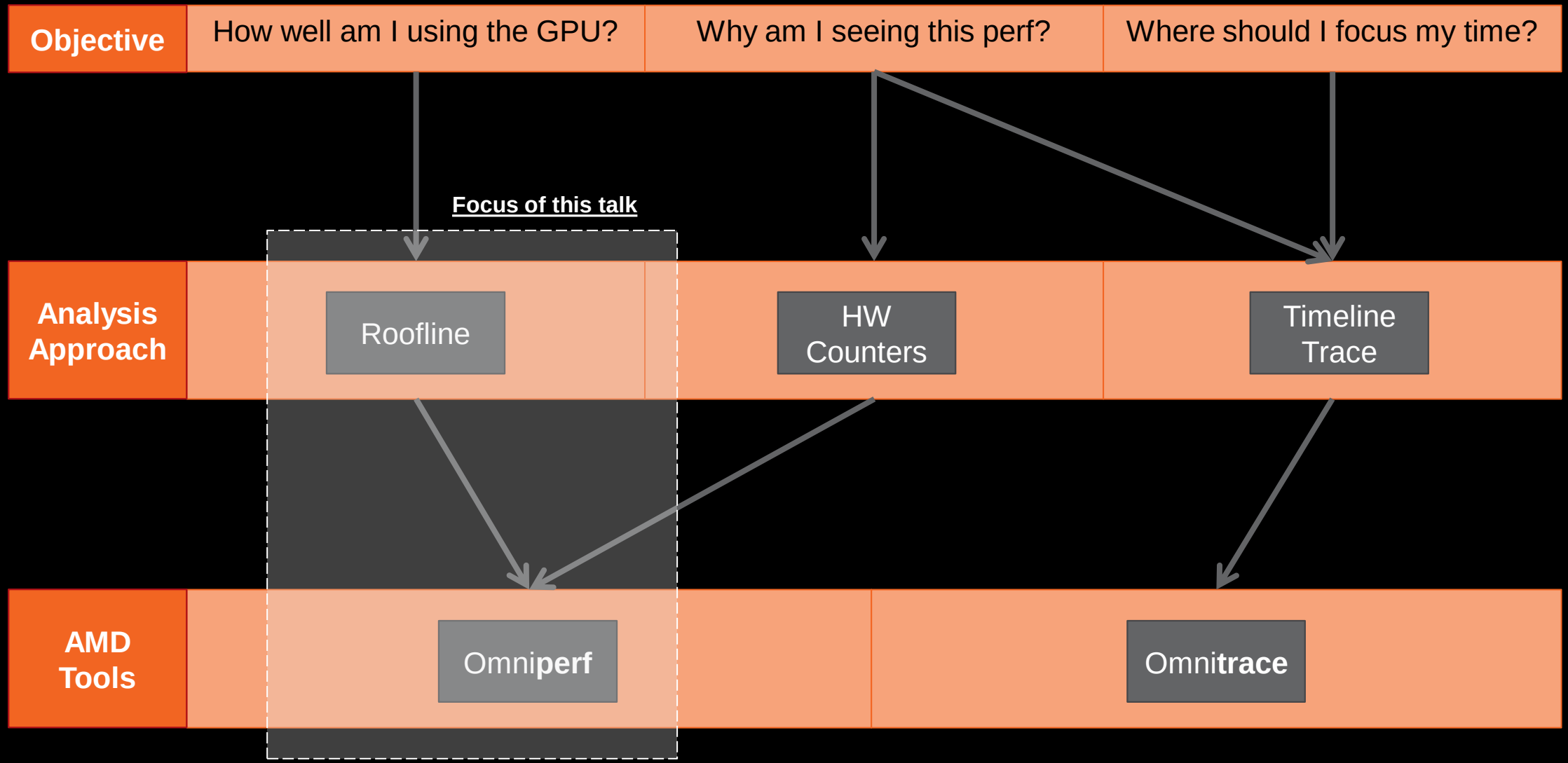
- github.com/AMDResearch/omniperf
- Comprehensive collection and visualization of performance counters
- Support for
  - GPU Speed-of-Light Analysis
  - Empirical Roofline Analysis
  - Memory Chart Analysis
  - Kernel Comparisons
- Can be visualized with Grafana

	A	B	C	D	E
1	Name	Calls	TotalDura	AverageN	Percentage
2	hipMemcpyAsync	99	3.22E+10	3.25E+08	44.14872
3	hipEventSynchronize	330	2.42E+10	73394557	33.225
4	hipMemsetAsync	87	7.76E+09	89232696	10.64953
5	hipHostMalloc	9	5.41E+09	6.01E+08	7.415198
6	hipDeviceSynchronize	28	1.32E+09	47006288	1.805515
7	hipHostFree	17	1.05E+09	61534688	1.435014
8	hipMemcpy	41	8.11E+08	19791876	1.113161
9	hipLaunchKernel	1856	58082083	31294	0.079676
10	hipStreamCreate	2	46380834	23190417	0.063625
11	hipMemset	2	18847246	9423623	0.025854
12	hipStreamDestroy	2	15183338	7591669	0.020828
13	hipFree	38	8269713	217624	0.011344
14	hipEventRecord	330	2520035	7636	0.003457
15	hipMalloc	30	1484804	49493	0.002037
16	__hipPopCallConfigura	1856	229159	123	0.000314
17	__hipPushCallConfigur	1856	224177	120	0.000308
18	hipGetLastError	1494	100458	67	0.000138
19	hipEventCreate	330	76675	232	0.000105
20	hipEventDestroy	330	64671	195	8.87E-05
21	hipGetDevicePropertie	47	51808	1102	7.11E-05
22	hipGetDevice	64	11611	181	1.59E-05
23	hipSetDevice	1	401	401	5.50E-07
24	hipGetDeviceCount	1	220	220	3.02E-07



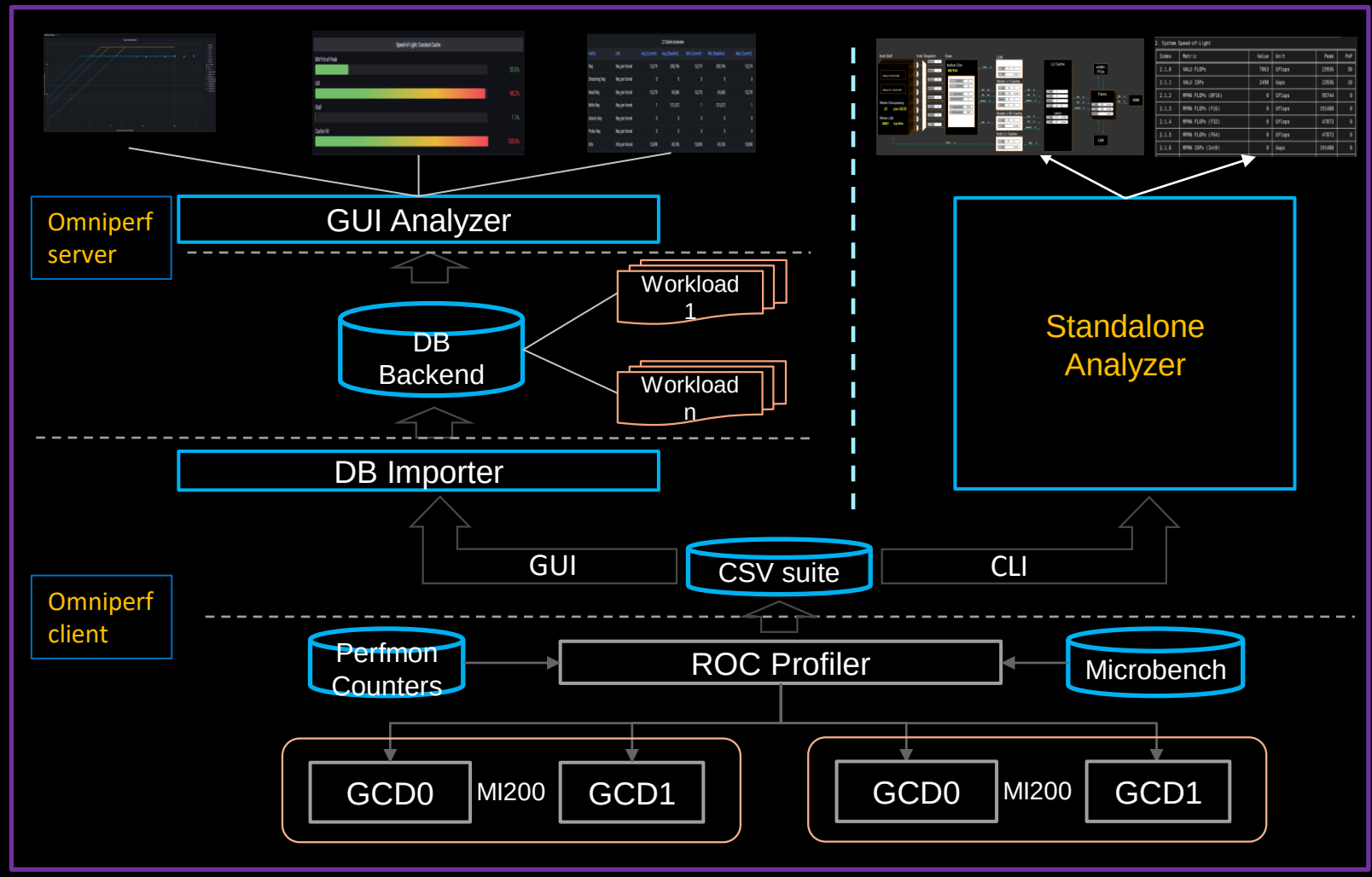


Background: AMD Profiling Tools

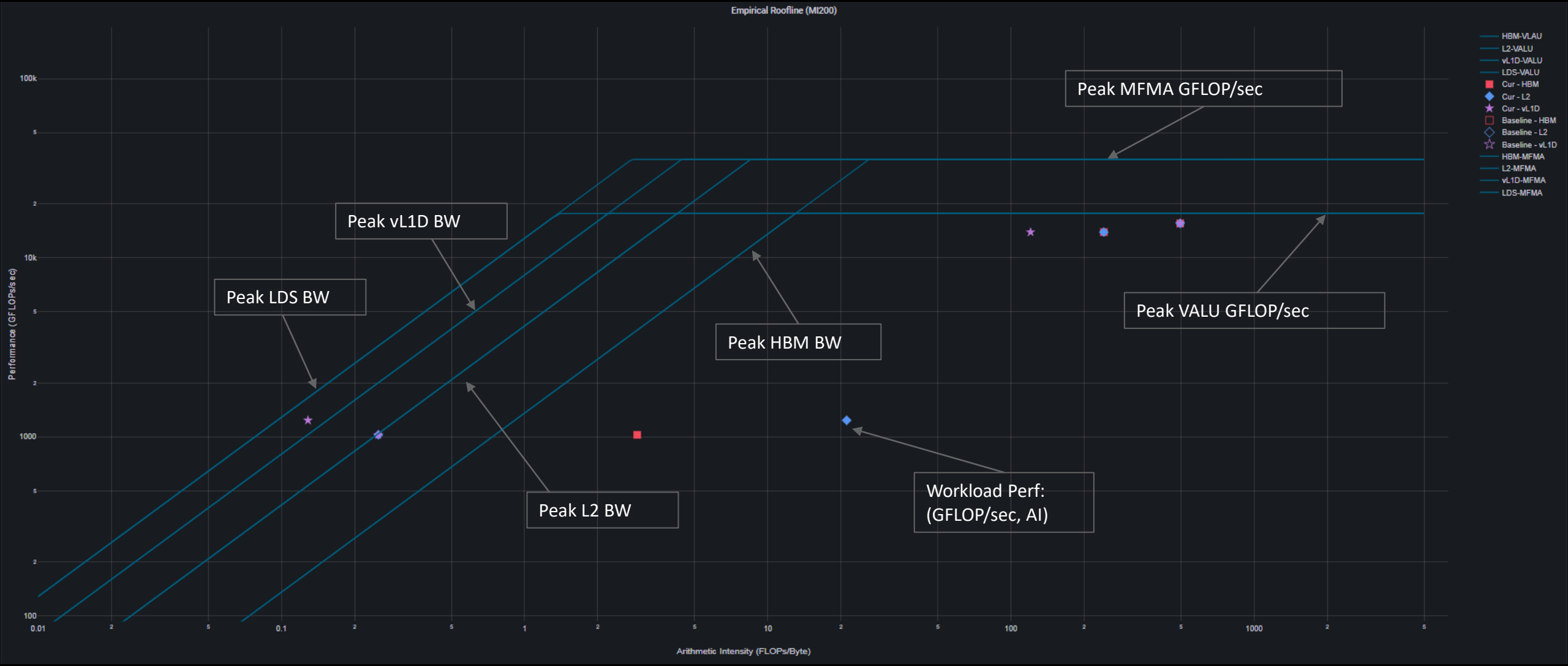


What is Omniperf?

- Open-source repo
  - github.com/AMDResearch/omniperf
- Built on top of ROC Profiler
  - Drives PMC collection
- Integrated Performance Analyzer for AMD GPUs
  - Roofline Analyzer
  - Mem. Chart Analyzer
  - Speed-of-Light
  - Kernel Comparisons
  - Flexible Filtering/Normalization
  - Comprehensive Profiling:
    - Wavefront Dispatching
    - Shader Compute
    - LDS Accesses
    - L1/L2 Cache Accesses
    - HBM Accesses
- User-interfaces
  - Grafana™ Based GUI
  - **Standalone GUI**

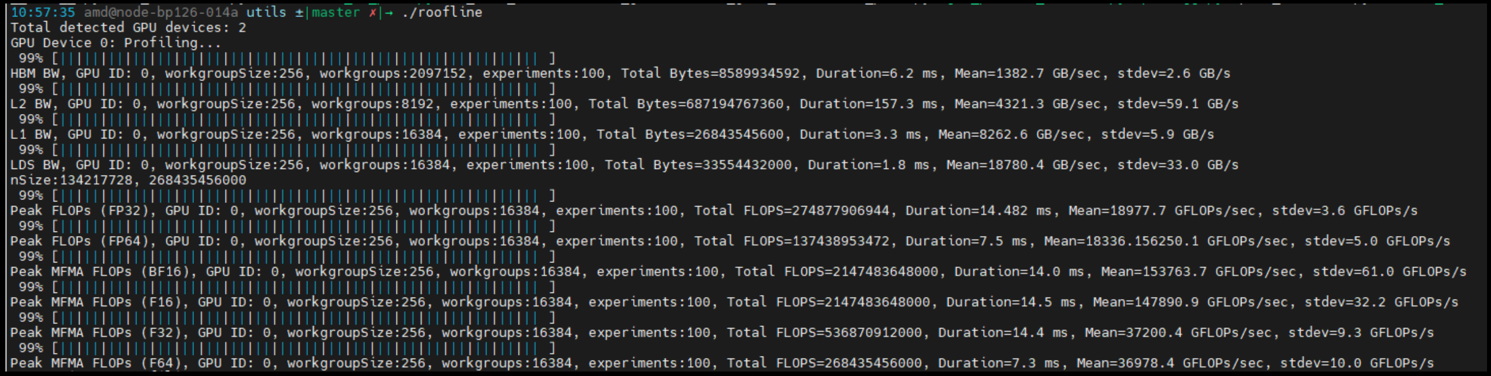


Roofline Analysis on MI200 GPUs



Roofline Analysis on MI200 GPUs: roofline benchmarking

- Empirical roofline benchmarking
  - Measure achievable peak FLOPS
    - VALU: F32, F64
    - MFMA: F16, F32, BF16, F32, F64
  - Measure achievable peak BW
    - LDS
    - Vector L1D Cache
    - L2 Cache
    - HBM



```
10:57:35 amd@node-bp126-014a utils ±|master ✗|→ ./roofline
Total detected GPU devices: 2
GPU Device 0: Profiling...
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
HBM BW, GPU ID: 0, workgroupSize:256, workgroups:2097152, experiments:100, Total Bytes=8589934592, Duration=6.2 ms, Mean=1382.7 GB/sec, stdev=2.6 GB/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
L2 BW, GPU ID: 0, workgroupSize:256, workgroups:8192, experiments:100, Total Bytes=687194767360, Duration=157.3 ms, Mean=4321.3 GB/sec, stdev=59.1 GB/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
L1 BW, GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total Bytes=26843545600, Duration=3.3 ms, Mean=8262.6 GB/sec, stdev=5.9 GB/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
LDS BW, GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total Bytes=33554432000, Duration=1.8 ms, Mean=18780.4 GB/sec, stdev=33.0 GB/s
nSize:134217728, 268435456000
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
Peak FLOPs (FP32), GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total FLOPS=274877906944, Duration=14.482 ms, Mean=18977.7 GFLOPs/sec, stdev=3.6 GFLOPs/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
Peak FLOPs (FP64), GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total FLOPS=137438953472, Duration=7.5 ms, Mean=18336.156250.1 GFLOPs/sec, stdev=5.0 GFLOPs/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
Peak MFMA FLOPs (BF16), GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total FLOPS=2147483648000, Duration=14.0 ms, Mean=153763.7 GFLOPs/sec, stdev=61.0 GFLOPs/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
Peak MFMA FLOPs (F16), GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total FLOPS=2147483648000, Duration=14.5 ms, Mean=147890.9 GFLOPs/sec, stdev=32.2 GFLOPs/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
Peak MFMA FLOPs (F32), GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total FLOPS=536870912000, Duration=14.4 ms, Mean=37200.4 GFLOPs/sec, stdev=9.3 GFLOPs/s
 99% [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ]
Peak MFMA FLOPs (F64), GPU ID: 0, workgroupSize:256, workgroups:16384, experiments:100, Total FLOPS=268435456000, Duration=7.3 ms, Mean=36978.4 GFLOPs/sec, stdev=10.0 GFLOPs/s
```

- Internally developed micro-benchmark algorithms
  - Peak VALU FLOP: axpy
  - Peak MFMA FLOP: Matrix multiplication based on MFMA intrinsic
  - Peak LDS/vL1D/L2 BW: Pointer chasing
  - Peak HBM BW: Streaming copy
- Note: empirical benchmark runs may not happen if Omniperf version not compatible with installed ROCm version

Getting Started with Omniperf

AMD
together we advance_

Getting started with Omniperf: Client

- Step 0: Install ROCm™ 5.1+
  - Fresh installation: Introduction to ROCm Installation Guide for Linux® (amd.com)

- Step 1: Clone Omniperf

```
git clone https://github.com/AMDResearch/omniperf.git
```

- Step 2: Install dependencies (requires CMake 3.19+ and Python 3.7+)

```
$cd omniperf
$export PATH=/path/to/cmake/bin:$PATH
$export INSTALL_DIR=/path/to/target-installation-dir/omniperf
$python3 -m pip install --system -t ${INSTALL_DIR}/python-libs -r requirements.txt
```

- Step 3: Build and install Omniperf client

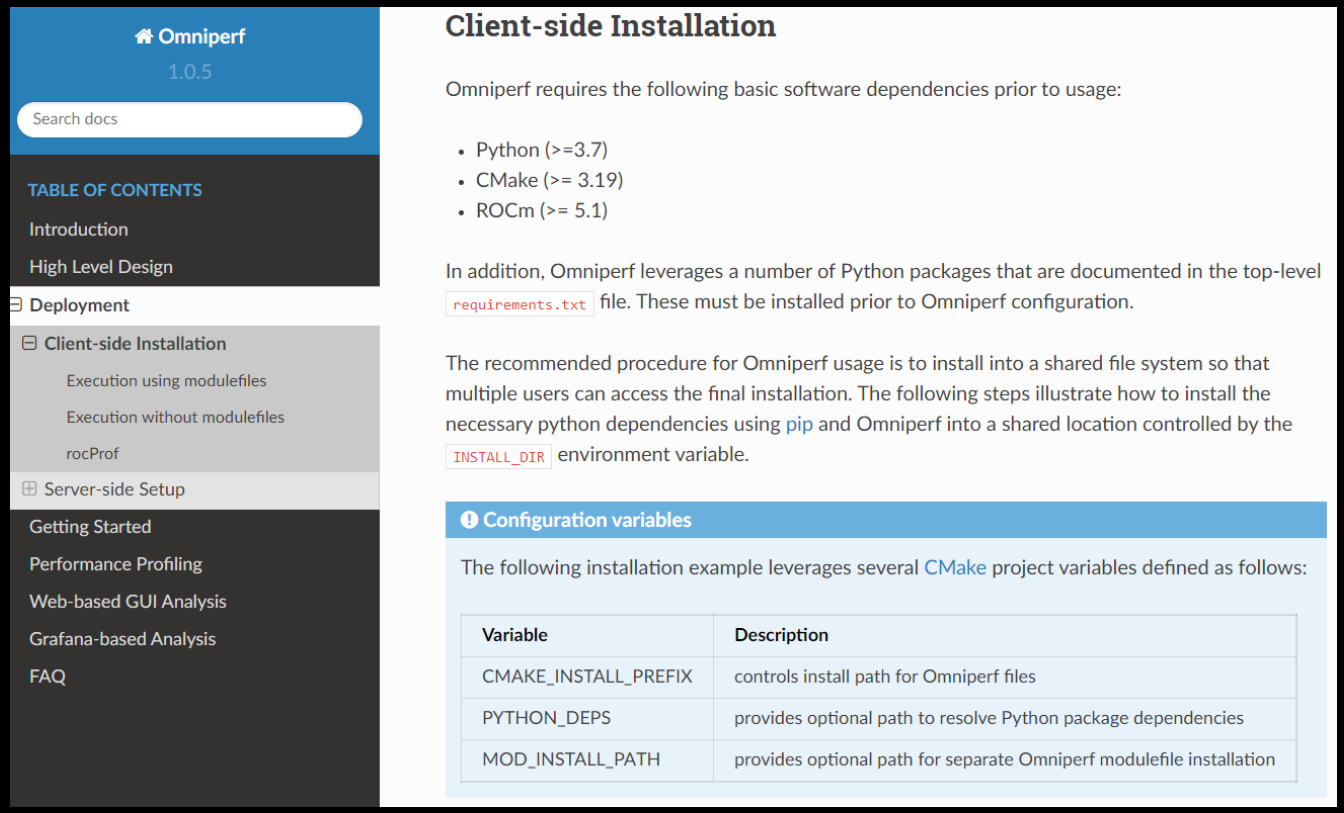
```
$cd build
$cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR}/1.0.4 -DPYTHON_DEPS=${INSTALL_DIR}/python-libs
-DMOD_INSTALL_PATH=${INSTALL_DIR}/modulefiles ..
$make install
```

Getting started with Omniperf: Client

- Step 5: Sanity Check

```
$export PATH=$INSTALL_DIR/1.0.4/bin:$PATH
$export PYTHONPATH=$INSTALL_DIR/python-libs
$export
ROOFLINE_BIN=/path/to/installed/omniperf/roofline
$omniperf -version
  ----------------------------------------
  Omniperf version: 1.0.4 (release)
  Git revision:     065b4b7
  ----------------------------------------
```

- Online Documentation for Omniperf installation:
  - amdresearch.github.io/omniperf/installation.html#client-side-installation
  - Includes instructions for systems using modulefiles



Getting started with Omniperf: Server

Can be installed locally on your workstation. Profile remotely, visualize locally

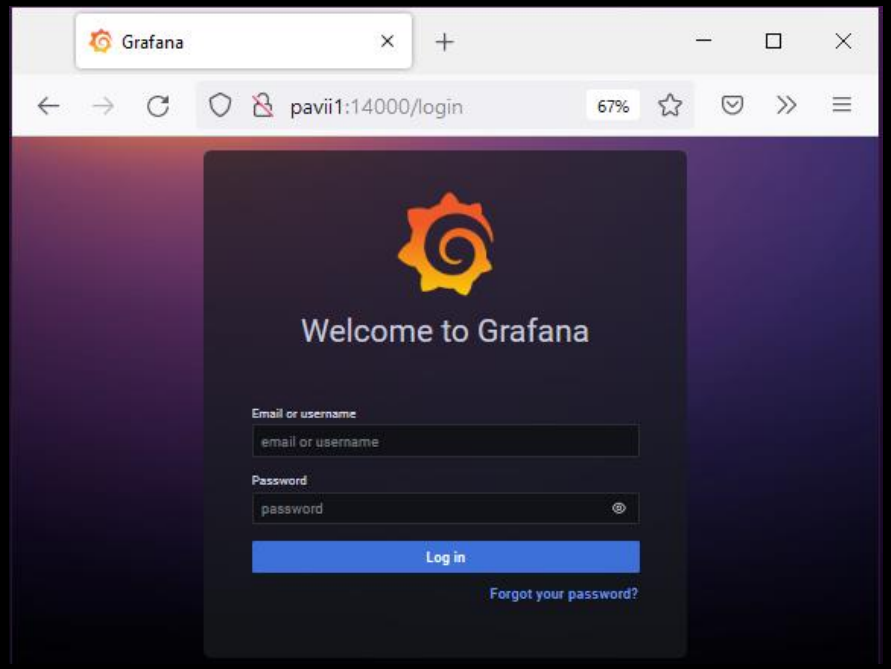
- Step 1: setup persistent Docker® storage

```
$sudo mkdir -p /usr/local/persist
$cd /usr/local/persist/
$sudo mkdir -p grafana-storage mongodb
$sudo docker volume create --driver local --opt type=none --opt device=/usr/local/persist/grafana-storage --opt o=bind grafana-storage
$sudo docker volume create --driver local --opt type=none --opt device=/usr/local/persist/mongodb --opt o=bind grafana-mongo-db
```

- Step 2: start the Omniperf server

```
$sudo docker-compose build
$sudo docker-compose up -d
```

- Step 3: configure server
  - Refer to: https://amdresearch.github.io/omniperf/



Omniperf Demonstration

AMD
together we advance_

Omniperf: “Hello World” Example

- Step 1: Compile workload

```
$mkdir test && cd test
$cp $OMNIPERF_HOME/sample/vcopy.cpp .
$hipcc vcopy.cpp -o vcopy
$./vcopy 1048576 256
  Finished allocating vectors on the CPU
  Finished allocating vectors on the GPU
  Finished copying vectors to the GPU
  sw thinks it moved 1.000000 KB per wave
  Total threads: 1048576, Grid Size: 4096 block Size:256, Wavefronts:16384:
  Launching the  kernel on the GPU
  Finished executing kernel
  Finished copying the output vector from the GPU to the CPU
  Releasing GPU memory
  Releasing CPU memory
```

- Step 2: Profile workload

```
$omniperf profile -n vcopy_demo -- ./vcopy 1048576 256
```

Omniperf: Live Demonstration on HPC examples

- axpy kernel
- Thread divergence

DISCLAIMERS AND ATTRIBUTIONS


AMD