Introduction to OmniTools

Nicholas Curtis, Jonathan Madsen, Keith Lowery, Xiaomin Lu, George Markomanolis, Cole Ramos, Karl W. Schulz, Noah Wolfe

Developing Applications with the AMD ROCm Ecosystem

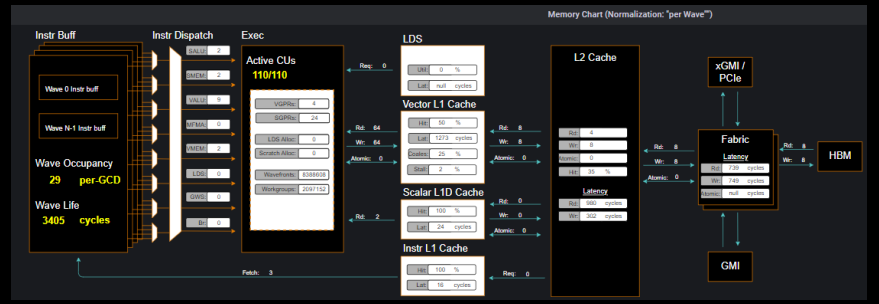
Background – AMD Profilers

- rocprof
  - github.com/ROCm-Developer-Tools/rocprofiler
  - Raw collection of GPU counters and traces
  - Counter collection driven by user provided input files
  - Counter results output in CSV
  - Trace collection support for:
    - HIP
    - HSA
    - GPU
  - Traces visualized with Perfetto

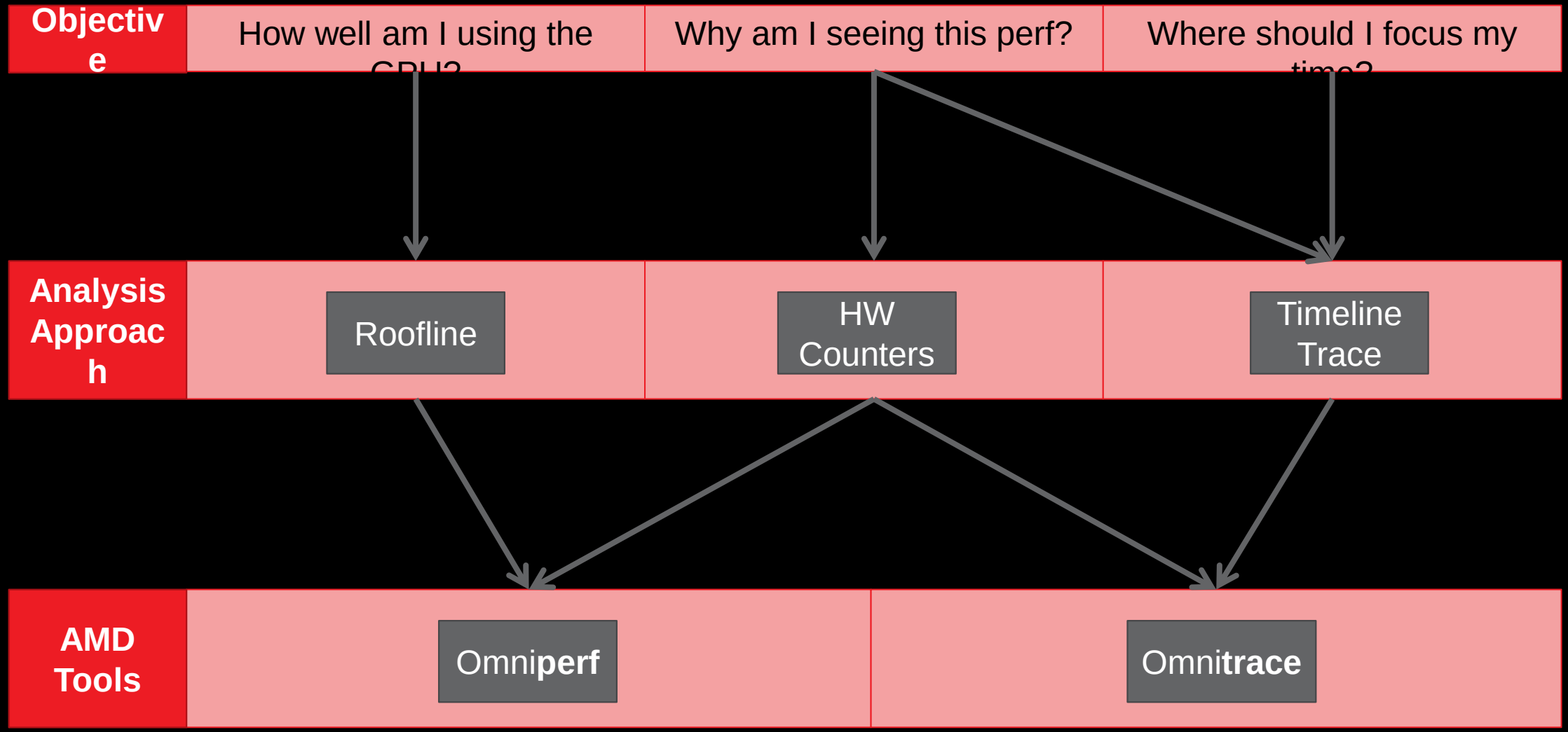
- Omni**trace**
  - github.com/AMDResearch/omnitrace
  - Comprehensive trace collection and visualization of CPU+GPU
  - Includes support for:
    - HIP, HSA, GPU
    - OpenMP®
    - MPI
    - Kokkos
    - Pthreads
    - Multi-GPU
  - Visualizations with Perfetto

- Omni**perf**
  - github.com/AMDResearch/omniperf
  - Automated collection, analysis and visualization of performance counters
  - Includes support for:
    - GPU Speed-of-Light Analysis
    - Memory Chart Analysis
    - Roofline Analysis
    - Kernel comparison
  - Visualizations with Grafana or standalone GUI

	A	B	C	D	E
1	Name	Calls	TotalDura	AverageNs	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 Profilers



Omnitrace

Omnitrace: Application Profiling, Tracing, and Analysis

- It is an AMD Research tool, repository: https://github.com/AMDResearch/omnitrace
- It is not part of ROCm stack
- Omnitrace is a comprehensive profiling and tracing tool for parallel applications written in C, C++, Fortran, HIP, OpenCL™, and Python™ which execute on the CPU or CPU+GPU
- Data collection modes:
  - Dynamic instrumentation
  - Statistical sampling
  - Process-level sampling
  - Critical trace generation
- Data analysis:
  - High-level summary profiles
  - Comprehensive traces
  - Critical trace analysis
- Parallelism support: HIP, HSA, Pthreads, MPI, Kokkos, OpenMP®
- GPU Metrics: GPU hardware counters, HIP/HSA API, HIP kernel tracing, HSA operation tracing, memory/power/temperature/utilization
- CPU Metrics: Hardware counters, timing metrics, memory metrics, network statistics, I/O, and more

Installation (if required)

- Instructions for binary installation
- Visit the Omnitrace releases page: https://github.com/AMDResearch/omnitrace/releases
- Select the version that matches your operating system, ROCm version, etc.
- For an HPE/AMD system, we select OpenSuse operating system
- For example, download the installer *omnitrace-1.7.2-opensuse-15.4-ROCm-50300-PAPI-OMPT-Python3.sh*
- Any user can install it in his project but it should not be required
- There are rpm and deb files for installation also
- Full documentation: https://amdresearch.github.io/omnitrace/

```
wget https://github.com/AMDResearch/omnitrace/releases/download/v1.7.2/omnitrace-1.7.2-opensuse-15.4-ROCm-50300-PAPI-OMPT-Python3.sh

mkdir /opt/omnitrace/
chmod +x omnitrace-1.7.2-opensuse-15.4-ROCm-50300-PAPI-OMPT-Python3.sh
./omnitrace-1.7.2-opensuse-15.4-ROCm-50300-PAPI-OMPT-Python3.sh --prefix=/opt/omnitrace --exclude-subdir
export PATH=/opt/omnitrace/:$PATH
source omnitrace_installation_path/share/omnitrace/setup-env.sh
```

Omnitrace instrumentation modes

- Runtime instrumentation: Dynamic binary instrumentation, it can instrument a lot of data and increased overhead
- Sampling instrumentation (omnitrace-sample)
- Attaching to a process (-p)
- Binary rewriting (-o)
  - It will not instrument the dynamically-linked libraries, thus lower overhead and faster execution
  - This approach is recommended when the target executable uses process-level parallelism (e.g. MPI)
  - To instrument dynamic libraries: https://amdresearch.github.io/omnitrace/instrumenting.html#binary-rewriting-a-library

For problems, create an issue here: https://github.com/AMDResearch/omnitrace/issues
Documentation: https://amdresearch.github.io/omniperf/

Execution

- Runtime instrumentation

```
srun … omnitrace <omnitrace-options> -- <exe> [<exe-options>]
```

- Sampling instrumentation

```
srun … omnitrace-sample <omnitrace-options> -- <exe> [<exe-options>]
```

- Binary rewriting

```
srun … omnitrace <omnitrace-options> -o <name-of-new-exe-or-library> -- <exe-or-library>
srun … <name-of-new-exe>
```

Omnitrace configuration (I)

srun -n 1 --gpus 1 omnitrace-avail --categories omnitrace

ENVIRONMENT VARIABLE	VALUE	CATEGORIES
OMNITRACE_CONFIG_FILE	%env{HOME}%/.omnitrace.cfg;%env{HOME}%/.omnitrace.json	config, core, libomnitrace, omnitrace, timemory
OMNITRACE_CRITICAL_TRACE	false	backend, critical_trace, custom, libomnitrace, omnitrace
OMNITRACE_OUTPUT_PATH	omnitrace-%tag%-output	filename, io, libomnitrace, omnitrace, timemory
OMNITRACE_OUTPUT_PREFIX		filename, io, libomnitrace, omnitrace, timemory
OMNITRACE_PERFETTO_BACKEND	inprocess	custom, libomnitrace, omnitrace, perfetto
OMNITRACE_PERFETTO_BUFFER_SIZE_KB	1024000	custom, data, libomnitrace, omnitrace, perfetto
OMNITRACE_PERFETTO_FILL_POLICY	discard	custom, data, libomnitrace, omnitrace, perfetto
OMNITRACE_PROCESS_SAMPLING_DURATION	-1	custom, libomnitrace, omnitrace, process_sampling, sampling
OMNITRACE_PROCESS_SAMPLING_FREQ	0	custom, libomnitrace, omnitrace, process_sampling
OMNITRACE_ROCM_EVENTS		custom, hardware_counters, libomnitrace, omnitrace, rocm, rocprofiler
OMNITRACE_SAMPLING_CPUS	0-3	custom, libomnitrace, omnitrace, process_sampling
OMNITRACE_SAMPLING_DELAY	0.5	custom, libomnitrace, omnitrace, process_sampling, sampling
OMNITRACE_SAMPLING_DURATION	0	custom, libomnitrace, omnitrace, process_sampling, sampling
OMNITRACE_SAMPLING_FREQ	100	custom, libomnitrace, omnitrace, process_sampling, sampling
OMNITRACE_SAMPLING_GPUS	all	custom, libomnitrace, omnitrace, process_sampling, rocm, rocm_smi
OMNITRACE_TIMEMORY_COMPONENTS	wall_clock,cpu_clock,page_rss,cpu_util,papi_vector	component, custom, libomnitrace, omnitrace, timemory
OMNITRACE_TIME_OUTPUT	true	filename, io, libomnitrace, omnitrace, timemory
OMNITRACE_USE_KOKKOSP	false	backend, custom, kokkos, libomnitrace, omnitrace
OMNITRACE_USE_PERFETTO	true	backend, custom, libomnitrace, omnitrace, perfetto
OMNITRACE_USE_PID	false	custom, filename, io, libomnitrace, omnitrace
OMNITRACE_USE_PROCESS_SAMPLING	true	backend, custom, libomnitrace, omnitrace, process_sampling, sampling
OMNITRACE_USE_RCCLP	false	backend, custom, libomnitrace, omnitrace, rccl, rocm
OMNITRACE_USE_ROCM_SMI	true	backend, custom, libomnitrace, omnitrace, rocm, rocm_smi
OMNITRACE_USE_ROCPROFILER	true	backend, custom, libomnitrace, omnitrace, rocm, rocprofiler
OMNITRACE_USE_ROCTRACER	true	backend, custom, libomnitrace, omnitrace, rocm, roctracer
OMNITRACE_USE_ROCTX	false	backend, custom, libomnitrace, omnitrace, rocm, roctracer, roctx
OMNITRACE_USE_SAMPLING	false	backend, custom, libomnitrace, omnitrace, sampling
OMNITRACE_USE_TIMEMORY	true	backend, custom, libomnitrace, omnitrace, timemory
OMNITRACE_VERBOSE	0	core, debugging, libomnitrace, omnitrace, timemory

Omnitrace configuration (II)

srun -n 1 --gpus 1 omnitrace-avail --categories omnitrace --brief --description

```
[omnitrace] /proc/sys/kernel/perf_event_paranoid has a value of 3. Disabling PAPI (requires a value <= 1)...
[omnitrace] In order to enable PAPI support, run 'echo N | sudo tee /proc/sys/kernel/perf_event_paranoid' where N is < 2
```

ENVIRONMENT VARIABLE	DESCRIPTION
OMNITRACE_CONFIG_FILE	Configuration file for omnitrace
OMNITRACE_CRITICAL_TRACE	Enable generation of the critical trace
OMNITRACE_OUTPUT_PATH	Explicitly specify the output folder for results
OMNITRACE_OUTPUT_PREFIX	Explicitly specify a prefix for all output files
OMNITRACE_PERFETTO_BACKEND	Specify the perfetto backend to activate. Options are: 'inprocess', 'system', or 'all'
OMNITRACE_PERFETTO_BUFFER_SIZE_KB	Size of perfetto buffer (in KB)
OMNITRACE_PERFETTO_FILL_POLICY	Behavior when perfetto buffer is full. 'discard' will ignore new entries, 'ring_buffer' will overwrite old entries
OMNITRACE_PROCESS_SAMPLING_DURATION	If > 0.0, time (in seconds) to sample before stopping. If less than zero, uses OMNITRACE_SAMPLING_DURATION
OMNITRACE_PROCESS_SAMPLING_FREQ	Number of measurements per second when OMNITTRACE_USE_PROCESS_SAMPLING=ON. If set to zero, uses OMNITRACE_SAMPLING_FREQ value
OMNITRACE_ROCM_EVENTS	ROCm hardware counters. Use ':device=N' syntax to specify collection on device number N, e.g. ':device=0'. If no device specification is provided, the event is collected on every available device
OMNITRACE_SAMPLING_CPUS	CPUs to collect frequency information for. Values should be separated by commas and can be explicit or ranges, e.g. 0,1,5-8. An empty value implies 'all' and 'none' suppresses all CPU frequency sampling
OMNITRACE_SAMPLING_DELAY	Time (in seconds) to wait before the first sampling signal is delivered, increasing this value can fix deadlocks during init
OMNITRACE_SAMPLING_DURATION	If > 0.0, time (in seconds) to sample before stopping
OMNITRACE_SAMPLING_FREQ	Number of software interrupts per second when OMNITTRACE_USE_SAMPLING=ON
OMNITRACE_SAMPLING_GPUS	Devices to query when OMNITRACE_USE_ROCM_SMI=ON. Values should be separated by commas and can be explicit or ranges, e.g. 0,1,5-8. An empty value implies 'all' and 'none' suppresses all GPU sampling
OMNITRACE_TIMEMORY_COMPONENTS	List of components to collect via timemory (see `omnitrace-avail -C`)
OMNITRACE_TIME_OUTPUT	Output data to subfolder w/ a timestamp (see also: TIME_FORMAT)
OMNITRACE_USE_KOKKOSP	Enable support for Kokkos Tools
OMNITRACE_USE_PERFETTO	Enable perfetto backend
OMNITRACE_USE_PID	Enable tagging filenames with process identifier (either MPI rank or pid)
OMNITRACE_USE_PROCESS_SAMPLING	Enable a background thread which samples process-level and system metrics such as the CPU/GPU freq, power, memory usage, etc.
OMNITRACE_USE_RCCLP	Enable support for ROCm Communication Collectives Library (RCCL) Performance
OMNITRACE_USE_ROCM_SMI	Enable sampling GPU power, temp, utilization, and memory usage
OMNITRACE_USE_ROCPROFILER	Enable ROCm hardware counters
OMNITRACE_USE_ROCTRACER	Enable ROCm API and kernel tracing
OMNITRACE_USE_ROCTX	Enable ROCtx API. Warning! Out-of-order ranges may corrupt perfetto flamegraph
OMNITRACE_USE_SAMPLING	Enable statistical sampling of call-stack
OMNITRACE_USE_TIMEMORY	Enable timemory backend
OMNITRACE_VERBOSE	Verbosity level

Create a configuration file

- Use a name of non-existing config file

```
srun -n 1 omnitrace-avail -G omnitrace.cfg
[omnitrace-avail] Outputting text configuration file './omnitrace.cfg'...
```

- To add also description for each variable

```
srun -n 1 omnitrace-avail -G omnitrace_all.cfg --all
[omnitrace-avail] Outputting text configuration file './omnitrace_all.cfg’...
```

- Declare which cfg file to use :

```
export OMNITRACE_CONFIG_GILE=/path/omnitrace.cfg
```

Executing MatrixTranspose

- Non instrumented execution

```
time srun -n 1 --gpus 1 ./MatrixTranspose
real    0m1.245s
```

- Dynamic instrumentation

```
time srun -n 1 -gpus 1 omnitrace -- ./MatrixTranspose
[omnitrace][exe]
[omnitrace][exe] command ::
'/pfs/lustrep4/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/MatrixTranspose'...
[omnitrace][exe]
…
[omnitrace][118151][metadata]> Outputting 'omnitrace-MatrixTranspose-output/2022-10-16_22.53/metadata-118151.json' and 'omnitrace-MatrixTranspose-output/2022-10-16_22.53/functions-118151.json'
[omnitrace][118151][0][omnitrace_finalize] Finalized
[706.822]       perfetto.cc:57383 Tracing session 1 ended, total sessions:0
[omnitrace][exe] End of omnitrace
real    1m27.841s
```

Identify overhead

Command: `nm --demangle MatrixTranspose | egrep -i ' (t|u) '`

```
000000000020d080 t _GLOBAL__sub_I_MatrixTranspose.cpp
000000000020c970 T __device_stub__warmup()
000000000020ca40 T matrixTransposeCPUReference(float*, float*, unsigned int)
000000000020c9c0 T __device_stub__matrixTranspose(float*, float*, int)
                 U std::ctype<char>::_M_widen_init() const
                 U std::ostream::put(char)
                 U std::ostream::flush()
                 U std::ios_base::Init::Init()
                 U std::ios_base::Init::~Init()
                 U std::basic_ostream<char, std::char_traits<char> >& std::__ostream_insert<char, std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&, char const*, long)
                 U std::__throw_bad_cast()
                 U __cxa_atexit
000000000020c930 t __do_global_dtors_aux
                 U __hipPopCallConfiguration
                 U __hipPushCallConfiguration
                 U __hipRegisterFatBinary
                 U __hipRegisterFunction
                 U __hipUnregisterFatBinary
000000000020cfd0 t __hip_module_ctor
000000000020d060 t __hip_module_dtor
000000000020d12e T __libc_csu_fini
000000000020d0ae T __libc_csu_init
                 U __libc_start_main
000000000020d178 t _fini
000000000020d160 t _init
000000000020c890 T _start
000000000020d14e t atexit
000000000020c8c0 t deregister_tm_clones
000000000020c960 t frame_dummy
                 U free
                 U hipFree
                 U hipGetDeviceProperties
                 U hipLaunchKernel
                 U hipMalloc
                 U hipMemcpy
000000000020cb00 T main
                 U malloc
                 U printf
                 U puts
000000000020c8f0 t register_tm_clones
                 U strlen
```

Available functions to instrument

```
srun -n 1 --gpus 1 omnitrace -v -1 --simulate --print-available functions --
./MatrixTranspose
```

```
[omnitrace][exe]
[omnitrace][exe] command :: '/pfs/lustrep4/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/MatrixTranspose'...
[omnitrace][exe]

[available] ../sysdeps/x86_64/crti.S:
[available]    [_fini][3]
[available]    [_init][7]

[available] ../sysdeps/x86_64/start.S:
[available]    [_start][12]

[available] /home/omnitrace/build-release/opensuse-15.2-rocm-5.0-python/external/dyninst/tbb/src/TBB-External/src/../src/old/concurrent_queue_v2.cpp:
[available]    [tbb::internal::concurrent_queue_base::concurrent_queue_base][51]
[available]    [tbb::internal::concurrent_queue_base::internal_pop][27]
[available]    [tbb::internal::concurrent_queue_base::internal_pop_if_present][50]
[available]    [tbb::internal::concurrent_queue_base::internal_push][46]
[available]    [tbb::internal::concurrent_queue_base::internal_push_if_not_full][54]
[available]    [tbb::internal::concurrent_queue_base::internal_set_capacity][5]
[available] MatrixTranspose:
[available]    [__device_stub__matrixTranspose][26]
[available]    [__device_stub__warmup][17]
[available]    [__do_global_dtors_aux][9]
[available]    [__hip_module_ctor][33]
[available]    [__hip_module_dtor][8]
[available]    [frame_dummy][4]
[available]    [targ20d132][1]

[available] MatrixTranspose.cpp:
[available]    [_GLOBAL__sub_I_MatrixTranspose.cpp][8]
[available]    [main][278]
[available]    [matrixTransposeCPUReference][51]

[available] atexit.c:
[available]    [atexit][5]

[available] elf-init.c:
[available]    [__libc_csu_fini][3]
[available]    [__libc_csu_init][36]
```

More than 36000 functions

Custom including/excluding functions

- Include functions

```
srun -n 1 --gpus 1 omnitrace -v -1 --simulate --print-available functions -I
'function_name1' 'function_name2' -- ./MatrixTranspose
```

- Exclude functions

```
srun -n 1 --gpus 1 omnitrace -v -1 --simulate --print-available functions -E
'function_name1' 'function_name2' -- ./MatrixTranspose
```

The above commands include the simulate flag that it will demonstrate the available functions but it will not run the MatrixTranspose executable

Decreasing profiling overhead

- Binary rewriting and print available functions

```
srun -n 1 --gpus 1 omnitrace -v -1 --print-available functions -o matrix.inst --
./MatrixTranspose
```

```
[omnitrace][exe]
[omnitrace][exe] command :: '/pfs/lustrep4/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/MatrixTranspose'...
[omnitrace][exe]
[omnitrace][exe] Resolved 'libomnitrace-rt.so' to '/pfs/lustrep4/scratch/project_462000075/markoman/omnitrace_install/lib/libomnitrace-rt.so.11.0.1'...
[omnitrace][exe] DYNINST_API_RT: /pfs/lustrep4/scratch/project_462000075/markoman/omnitrace_install/lib/libomnitrace-rt.so.11.0.1
[omnitrace][exe] instrumentation target: /pfs/lustrep4/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/MatrixTranspose
[omnitrace][exe] Opening '/pfs/lustrep4/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/MatrixTranspose' for binary rewrite... Done
[omnitrace][exe] Getting the address space image, modules, and procedures...
[omnitrace][exe]
[omnitrace][exe] Found 16 functions in 6 modules in instrumentation target
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/available.json'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/available.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/overlapping.json'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/overlapping.txt'... Done
[omnitrace][exe] function: 'main' ... found
[omnitrace][exe] function: 'omnitrace_user_start_trace' ... not found
[omnitrace][exe] function: 'omnitrace_user_stop_trace' ... not found
[omnitrace][exe] function: 'MPI_Init' ... not found
[omnitrace][exe] function: 'MPI_Init_thread' ... not found
[omnitrace][exe] function: 'MPI_Finalize' ... not found
[omnitrace][exe] function: 'MPI_Comm_rank' ... not found
[omnitrace][exe] function: 'MPI_Comm_size' ... not found
[omnitrace][exe] Resolved 'libomnitrace-dl.so' to '/pfs/lustrep4/scratch/project_462000075/markoman/omnitrace_install/lib/libomnitrace-dl.so.1.6.0'...
[omnitrace][exe] loading library: '/pfs/lustrep4/scratch/project_462000075/markoman/omnitrace_install/lib/libomnitrace-dl.so.1.6.0'...
[omnitrace][exe] Finding instrumentation functions...
[omnitrace][exe] function: 'omnitrace_init' ... found
[omnitrace][exe] function: 'omnitrace_finalize' ... found
[omnitrace][exe] function: 'omnitrace_set_env' ... found
[omnitrace][exe] function: 'omnitrace_set_mpi' ... found
[omnitrace][exe] function: 'omnitrace_push_trace' ... found
[omnitrace][exe] function: 'omnitrace_pop_trace' ... found
[omnitrace][exe] function: 'omnitrace_register_source' ... found
[omnitrace][exe] function: 'omnitrace_register_coverage' ... found
[omnitrace][exe] Resolved 'libomnitrace-dl.so' to '/pfs/lustrep4/scratch/project_462000075/markoman/omnitrace_install/lib/libomnitrace-dl.so.1.6.0'...
[omnitrace][exe] Adding main entry snippets...
[omnitrace][exe] Adding main exit snippets...
[omnitrace][exe]
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/available.json'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/available.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/instrumented.json'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/instrumented.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/excluded.json'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/excluded.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/overlapping.json'... Done
[omnitrace][exe] Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.21_PM/instrumentation/overlapping.txt'... Done

[instrumented] MatrixTranspose.cpp:
[instrumented]    [main][278]
```

- Default instrumentation is main function and functions of 1024 instructions and more (for CPU)
- To instrument routines with for example 50 instructions, add the option "–i 50" to instrument function of 50 instructions and above (move overhead)

Executing the new binary

```
time srun -n 1 --gpus 1 ./matrix.inst
```

```
[omnitrace][omnitrace_init_tooling] Instrumentation mode: Trace


[omnitrace] /proc/sys/kernel/perf_event_paranoid has a value of 3. Disabling PAPI (requires a value <= 1)...
[omnitrace] In order to enable PAPI support, run 'echo N | sudo tee /proc/sys/kernel/perf_event_paranoid' where N is < 2
[730.689]       perfetto.cc:55910 Configured tracing session 1, #sources:1, duration:0 ms, #buffers:1, total buffer size:1024000 KB, total sessions:1, uid:0 session name: ""

Device name
Device name
[omnitrace][91915][1][hip_activity_callback]    1 :: CopyHostToDevice     :: CopyHostToDevice     :: cid=7, time_ns=(357731149538957:357731140299748) delta=-9239209, device_id=0, stream_id=0, pid=0, tid=0
PASSED!

[omnitrace][91915][0][omnitrace_finalize] finalizing...


[omnitrace][91915][0][omnitrace_finalize] omnitrace/process/91915 : 0.471434 sec wall_clock,  217.600 MB peak_rss,  210.379 MB page_rss, 0.480000 sec cpu_clock,  101.8 % cpu_util [laps: 1]
[omnitrace][91915][0][omnitrace_finalize] omnitrace/process/91915/thread/0 : 0.471373 sec wall_clock, 0.237256 sec thread_cpu_clock,   50.3 % thread_cpu_util,  217.600 MB peak_rss [laps: 1]

[omnitrace][91915][0][omnitrace_finalize] Finalizing perfetto...
[omnitrace][91915][perfetto]> Outputting '/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/omnitrace-matrix.inst-output/2022-11-14_12.33_PM/perfetto-trace.proto' (1008.42 KB / 1.01 MB / 0.00 GB)... Done
[omnitrace][91915][roctracer]> Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.33_PM/roctracer.json'
[omnitrace][91915][roctracer]> Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.33_PM/roctracer.txt'
[omnitrace][91915][wall_clock]> Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.33_PM/wall_clock.json'
[omnitrace][91915][wall_clock]> Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.33_PM/wall_clock.txt'
[omnitrace][91915][manager::finalize][metadata]> Outputting 'omnitrace-matrix.inst-output/2022-11-14_12.33_PM/metadata.json' and 'omnitrace-matrix.inst-output/2022-11-14_12.33_PM/functions.json'
[omnitrace][91915][0][omnitrace_finalize] Finalized
[731.210]       perfetto.cc:57383 Tracing session 1 ended, total sessions:0

real    0m0.803s
```

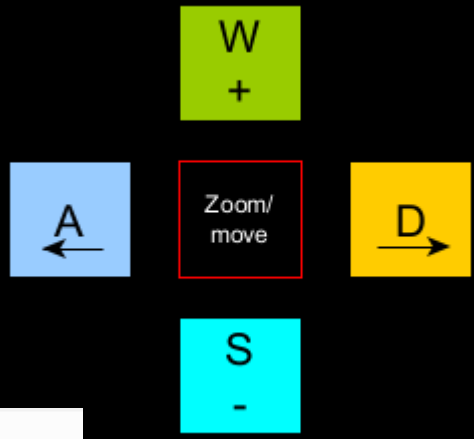
Check the list of the GPU calls instrumented

omnitrace-matrix.inst-output/2022-11-14_12.33_PM/roctracer.txt

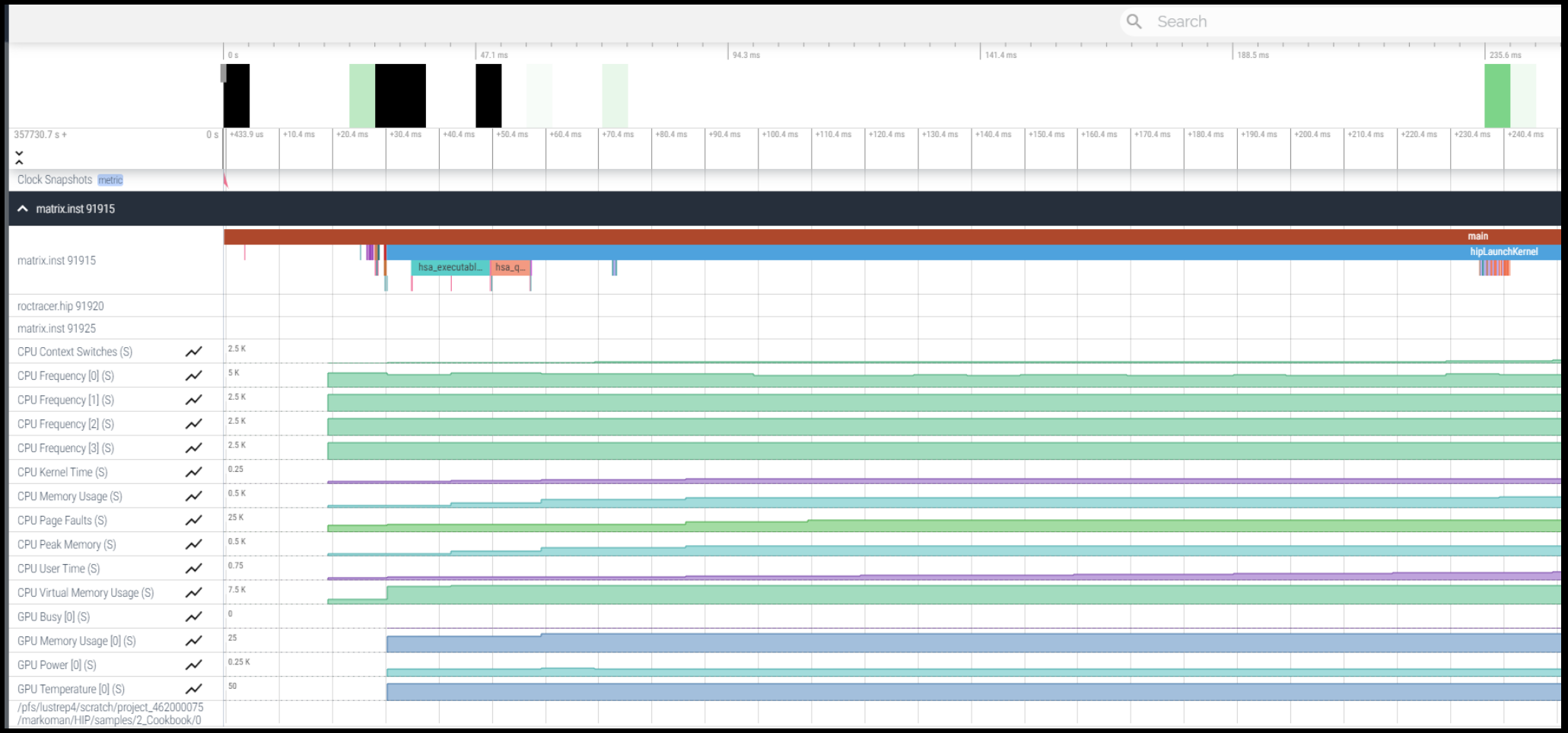
ROCM TRACER (ACTIVITY API)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	% SELF
|0>>> pthread_create	5	0	roctracer	sec	0.001036	0.000207	100.0
|2>>> |_start_thread	-	1	-	-	-	-	-
|2>>> |_hsa_amd_memory_pool_allocate	5	2	roctracer	sec	0.000750	0.000150	100.0
|2>>> |_hsa_iterate_agents	2	2	roctracer	sec	0.000018	0.000009	100.0
|2>>> |_hsa_amd_agents_allow_access	4	2	roctracer	sec	0.000118	0.000030	100.0
|2>>> |_hsa_agent_iterate_isas	1	2	roctracer	sec	0.000001	0.000001	100.0
|2>>> |_hsa_signal_create	15	2	roctracer	sec	0.000068	0.000005	100.0
|2>>> |_hsa_executable_load_agent_code_object	1	2	roctracer	sec	0.014825	0.014825	100.0
|2>>> |_hsa_amd_memory_lock_to_pool	3	2	roctracer	sec	0.000538	0.000179	100.0
|2>>> |_hsa_signal_silent_store_relaxed	5	2	roctracer	sec	0.000001	0.000000	100.0
|2>>> |_hsa_queue_add_write_index_screlease	3	2	roctracer	sec	0.000001	0.000000	100.0
|2>>> |_hsa_signal_store_screlease	4	2	roctracer	sec	0.000001	0.000000	100.0
|2>>> |_hsa_amd_signal_async_handler	3	2	roctracer	sec	0.000001	0.000000	100.0
|2>>> |_hsa_signal_wait_scacquire	5	2	roctracer	sec	0.009013	0.001803	100.0
|2>>> |_hsa_signal_load_relaxed	7	2	roctracer	sec	0.000003	0.000000	100.0
|2>>> |_hsa_queue_load_read_index_relaxed	2	2	roctracer	sec	0.000000	0.000000	100.0
|2>>> |_hsa_signal_destroy	1	2	roctracer	sec	0.000000	0.000000	100.0
|2>>> |_hsa_amd_memory_unlock	2	2	roctracer	sec	0.000098	0.000049	100.0
|2>>> |_hsa_queue_load_read_index_scacquire	2	2	roctracer	sec	0.000000	0.000000	100.0
|2>>> |_hsa_amd_memory_async_copy	1	2	roctracer	sec	0.000002	0.000002	100.0
|4>>> |_start_thread	-	1	-	-	-	-	-
|4>>> |_hsa_amd_memory_pool_allocate	1	2	roctracer	sec	0.000092	0.000092	100.0
|4>>> |_hsa_signal_create	11	2	roctracer	sec	0.000003	0.000000	100.0
|4>>> |_hsa_executable_load_agent_code_object	1	2	roctracer	sec	0.005452	0.005452	100.0
|4>>> |_hsa_queue_load_read_index_relaxed	1	2	roctracer	sec	0.000000	0.000000	100.0
|4>>> |_hsa_amd_memory_lock_to_pool	1	2	roctracer	sec	0.000068	0.000068	100.0
|4>>> |_hsa_queue_load_read_index_scacquire	1	2	roctracer	sec	0.000000	0.000000	100.0
|4>>> |_hsa_signal_load_relaxed	5	2	roctracer	sec	0.000001	0.000000	100.0
|4>>> |_hsa_signal_destroy	2	2	roctracer	sec	0.000000	0.000000	100.0
|4>>> |_hsa_signal_wait_scacquire	2	2	roctracer	sec	0.000182	0.000091	100.0
|4>>> |_hsa_amd_memory_unlock	1	2	roctracer	sec	0.000043	0.000043	100.0
|4>>> |_hsa_amd_memory_async_copy	1	2	roctracer	sec	0.000304	0.000304	100.0
|4>>> |_hsa_signal_store_screlease	1	2	roctracer	sec	0.000000	0.000000	100.0
|4>>> |_hsa_amd_memory_pool_free	1	2	roctracer	sec	0.000062	0.000062	100.0
|5>>> |_start_thread	-	1	-	-	-	-	-
|5>>> |_hsa_signal_create	8	2	roctracer	sec	0.000001	0.000000	100.0
|5>>> |_hsa_queue_add_write_index_screlease	1	2	roctracer	sec	0.000000	0.000000	100.0
|5>>> |_hsa_signal_store_screlease	2	2	roctracer	sec	0.000001	0.000001	100.0
|5>>> |_hsa_signal_silent_store_relaxed	2	2	roctracer	sec	0.000000	0.000000	100.0
|5>>> |_hsa_signal_load_relaxed	1	2	roctracer	sec	0.000000	0.000000	100.0
|5>>> |_hsa_amd_memory_pool_free	1	2	roctracer	sec	0.000047	0.000047	100.0
|3>>> |_start_thread	-	1	-	-	-	-	-
|3>>> |_hsa_queue_create	1	2	roctracer	sec	0.007257	0.007257	100.0
|3>>> |_hsa_signal_create	10	2	roctracer	sec	0.000003	0.000000	100.0
|3>>> |_hsa_signal_load_relaxed	3	2	roctracer	sec	0.000001	0.000000	100.0
|3>>> |_hsa_queue_load_read_index_scacquire	1	2	roctracer	sec	0.000000	0.000000	100.0
|3>>> |_hsa_queue_load_read_index_relaxed	1	2	roctracer	sec	0.000000	0.000000	100.0
|3>>> |_hsa_amd_memory_async_copy	1	2	roctracer	sec	0.000281	0.000281	100.0
|1>>> |_start_thread	-	1	-	-	-	-	-
|0>>> hipGetDeviceProperties	1	0	roctracer	sec	0.000000	0.000000	0.0
|0>>> hipMalloc	2	0	roctracer	sec	0.000000	0.000000	0.0
|0>>> hipLaunchKernel	2	0	roctracer	sec	0.000000	0.000000	0.0
|0>>> hipMemcpy	3	0	roctracer	sec	0.000000	0.000000	0.0
|0>>> hipFree	2	0	roctracer	sec	0.000000	0.000000	0.0
|0>>> |_warmup()	1	1	roctracer	sec	0.000001	0.000001	100.0
|0>>> |_matrixTranspose(float*, float*, int)	1	1	roctracer	sec	0.000085	0.000085	100.0

Visualizing trace

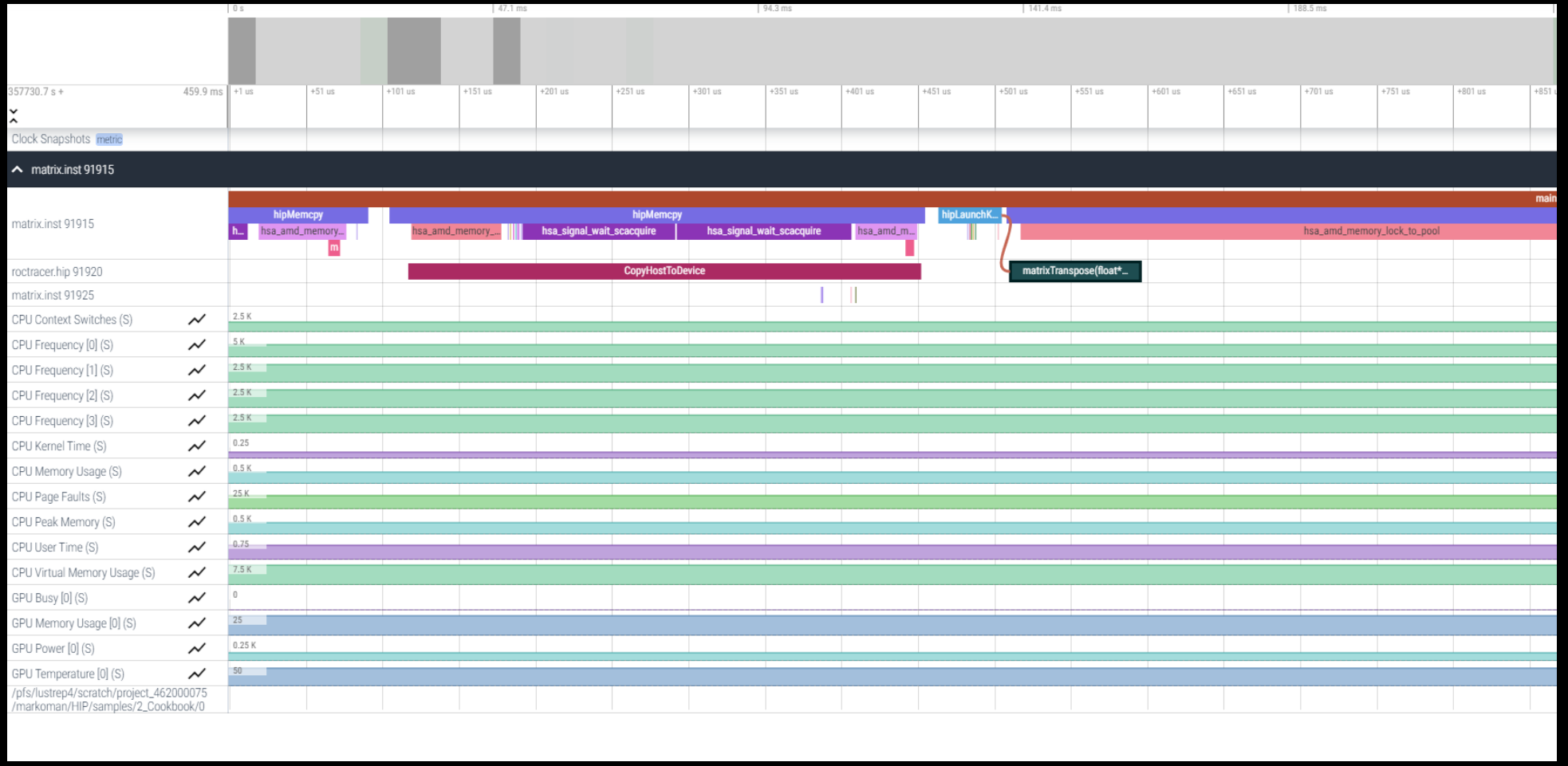


- Copy the perfetto-trace.proto to your laptop
- Go to https://ui.perfetto.dev/ click open trace and select the perfetto-trace.proto



Visualizing trace

- Copy the perfetto-trace.proto to your laptop
- Go to https://ui.perfetto.dev/ click open trace and select the perfetto-trace.proto



Hardware counters (I)

```
srun -n 1 --gpus 1 omnitrace-avail --all
```

GPU		
SQ_INSTS_VMEM_WR:device=0	true	Number of VMEM write instructions issued (including FLAT). (per-simd, emulated)
SQ_INSTS_VMEM_RD:device=0	true	Number of VMEM read instructions issued (including FLAT). (per-simd, emulated)
SQ_INSTS_SALU:device=0	true	Number of SALU instructions issued. (per-simd, emulated)
SQ_INSTS_SMEM:device=0	true	Number of SMEM instructions issued. (per-simd, emulated)
SQ_INSTS_FLAT:device=0	true	Number of FLAT instructions issued. (per-simd, emulated)
SQ_INSTS_FLAT_LDS_ONLY:device=0	true	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:device=0	true	Number of LDS instructions issued (including FLAT). (per-simd, emulated)
SQ_INSTS_GDS:device=0	true	Number of GDS instructions issued. (per-simd, emulated)
SQ_WAIT_INST_LDS:device=0	true	Number of wave-cycles spent waiting for LDS instruction issue. In units of 4 cycles. (per-simd, nondeterministic)
SQ_ACTIVE_INST_VALU:device=0	true	regspec 71? Number of cycles the SQ instruction arbiter is working on a VALU instruction. (per-simd, nondeterministic)
SQ_INST_CYCLES_SALU:device=0	true	Number of cycles needed to execute non-memory read scalar operations. (per-simd, emulated)
SQ_THREAD_CYCLES_VALU:device=0	true	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:device=0	true	Number of cycles LDS is stalled by bank conflicts. (emulated)
TCC_HIT[0]:device=0	true	Number of cache hits.
TCC_HIT[1]:device=0	true	Number of cache hits.

...

FETCH_SIZE:device=0	true	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:device=0	true	The total kilobytes written to the video memory. This is measured with all extra fetches and any cache or memory effects taken into account.
WRITE_REQ_32B:device=0	true	The total number of 32-byte effective memory writes.
GPUBusy:device=0	true	The percentage of time GPU was busy.
Wavefronts:device=0	true	Total wavefronts.
VALUInsts:device=0	true	The average number of vector ALU instructions executed per work-item (affected by flow control).
SALUInsts:device=0	true	The average number of scalar ALU instructions executed per work-item (affected by flow control).
VFetchInsts:device=0	true	The average number of vector fetch instructions from the video memory executed per work-item (affected by flow control). Excludes FLAT instructions that fetch...
SFetchInsts:device=0	true	The average number of scalar fetch instructions from the video memory executed per work-item (affected by flow control).
VWriteInsts:device=0	true	The average number of vector write instructions to the video memory executed per work-item (affected by flow control). Excludes FLAT instructions that write t...
FlatVMemInsts:device=0	true	The average number of FLAT instructions that read from or write to the video memory executed per work item (affected by flow control). Includes FLAT instructi...
LDSInsts:device=0	true	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 writ...
FlatLDSInsts:device=0	true	The average number of FLAT instructions that read or write to LDS executed per work item (affected by flow control).
GDSInsts:device=0	true	The average number of GDS read or GDS write instructions executed per work item (affected by flow control).
VALUUtilization:device=0	true	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 mu...
VALUBusy:device=0	true	The percentage of GPUTime vector ALU instructions are processed. Value range: 0% (bad) to 100% (optimal).
SALUBusy:device=0	true	The percentage of GPUTime scalar ALU instructions are processed. Value range: 0% (bad) to 100% (optimal).
FetchSize:device=0	true	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:device=0	true	The total kilobytes written to the video memory. This is measured with all extra fetches and any cache or memory effects taken into account.
MemWrites32B:device=0	true	The total number of effective 32B write transactions to the memory
L2CacheHit:device=0	true	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:device=0	true	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 a...
MemUnitStalled:device=0	true	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:device=0	true	The percentage of GPUTime the Write unit is stalled. Value range: 0% to 100% (bad).
ALUStalledByLDS:device=0	true	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...
LDSBankConflict:device=0	true	The percentage of GPUTime LDS is stalled by bank conflicts. Value range: 0% (optimal) to 100% (bad).

Commonly Used Counters

- VALUUtilization: The percentage of ALUs active in a wave. Low VALUUtilization is likely due to high divergence or a poorly sized grid
- VALUBusy: The percentage of GPUTime vector ALU instructions are processed. Can be thought of as something like compute utilization
- FetchSize: The total kilobytes fetched from global memory
- WriteSize: The total kilobytes written to global memory
- L2CacheHit: The percentage of fetch, write, atomic, and other instructions that hit the data in L2 cache
- MemUnitBusy: The percentage of GPUTime the memory unit is active. The result includes the stall time
- MemUnitStalled: The percentage of GPUTime the memory unit is stalled
- WriteUnitStalled: The percentage of GPUTime the write unit is stalled

Full list at: https://github.com/ROCm-Developer-Tools/rocprofiler/blob/amd-master/test/tool/metrics.xml

Hardware counters (II)

- Declare in your cfg file the metrics you want to profile
- For example, profile metrics only for the GPU with id 0:

```
OMNITRACE_ROCM_EVENTS = GPUBusy:device=0,Wavefronts:device=0,
VALUBusy:device=0,L2CacheHit:device=0,MemUnitBusy:device=0
```

- Profile for all the participated GPUs:

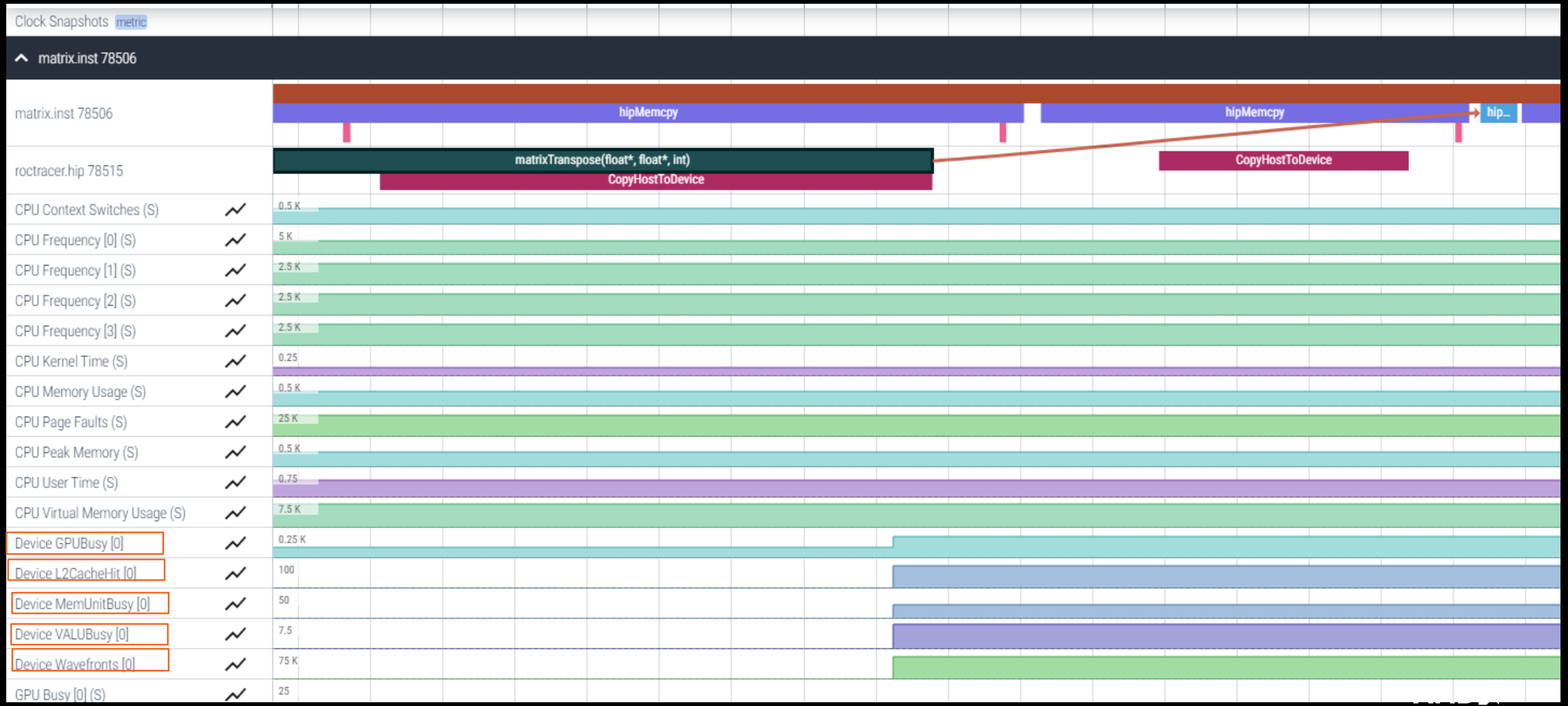
```
OMNITRACE_ROCM_EVENTS = GPUBusy,Wavefronts,VALUBusy,L2CacheHit,MemUnitBusy
```

Execution with hardware counters

```
srun -n 1 --gpus 1 ./matrix.inst
```

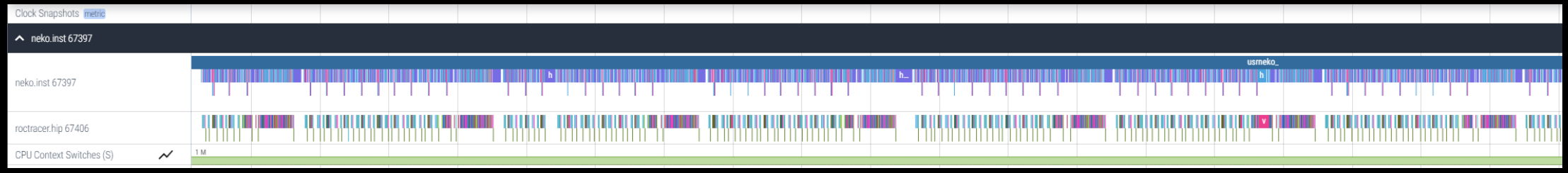
```
[omnitrace] /proc/sys/kernel/perf_event_paranoid has a value of 3. Disabling PAPI (requires a value <= 2)...
[omnitrace] In order to enable PAPI support, run 'echo N | sudo tee /proc/sys/kernel/perf_event_paranoid' where N is <= 2
[297.589]       perfetto.cc:55910 Configured tracing session 1, #sources:1, duration:0 ms, #buffers:1, total buffer size:1024000 KB, total sessions:1, uid:0 session name: ""
Device name
Device name

PASSED!
[omnitrace][78506][0][omnitrace_finalize] finalizing...
[omnitrace][78506][0][omnitrace_finalize]
[omnitrace][78506][0][omnitrace_finalize] omnitrace/process/78506 : 0.717209 sec wall_clock,  219.768 MB peak_rss,  212.754 MB page_rss, 0.740000 sec cpu_clock,  103.2 % cpu_util [laps: 1]
[omnitrace][78506][0][omnitrace_finalize] omnitrace/process/78506/thread/0 : 0.715605 sec wall_clock, 0.233719 sec thread_cpu_clock,   32.7 % thread_cpu_util,  219.768 MB peak_rss [laps: 1]
[omnitrace][78506][0][omnitrace_finalize]
[omnitrace][78506][0][omnitrace_finalize] Finalizing perfetto...
[omnitrace][78506][perfetto]> Outputting '/scratch/project_462000075/markoman/HIP/samples/2_Cookbook/0_MatrixTranspose/omnitrace-matrix.inst-output/2022-11-16_00.45/perfetto-trace.proto' (95.15 KB / 0.10 MB / 0.00 GB)... Done
[omnitrace][78506][0][omnitrace_finalize] Finalization metrics: 0.137393 sec wall_clock,    0.000 MB peak_rss,    1.085 MB page_rss, 0.130000 sec cpu_clock,   94.6 % cpu_util
[omnitrace][78506][rocprof-device-0-GPUBusy]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-GPUBusy.json'
[omnitrace][78506][rocprof-device-0-GPUBusy]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-GPUBusy.txt'
[omnitrace][78506][rocprof-device-0-Wavefronts]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-Wavefronts.json'
[omnitrace][78506][rocprof-device-0-Wavefronts]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-Wavefronts.txt'
[omnitrace][78506][rocprof-device-0-VALUBusy]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-VALUBusy.json'
[omnitrace][78506][rocprof-device-0-VALUBusy]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-VALUBusy.txt'
[omnitrace][78506][rocprof-device-0-L2CacheHit]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-L2CacheHit.json'
[omnitrace][78506][rocprof-device-0-L2CacheHit]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-L2CacheHit.txt'
[omnitrace][78506][rocprof-device-0-MemUnitBusy]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-MemUnitBusy.json'
[omnitrace][78506][rocprof-device-0-MemUnitBusy]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/rocprof-device-0-MemUnitBusy.txt'
[omnitrace][78506][roctracer]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/roctracer.json'
[omnitrace][78506][roctracer]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/roctracer.txt'
[omnitrace][78506][sampling_gpu_memory_usage]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_memory_usage.json'
[omnitrace][78506][sampling_gpu_memory_usage]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_memory_usage.txt'
[omnitrace][78506][sampling_gpu_power]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_power.json'
[omnitrace][78506][sampling_gpu_power]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_power.txt'
[omnitrace][78506][sampling_gpu_temperature]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_temperature.json'
[omnitrace][78506][sampling_gpu_temperature]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_temperature.txt'
[omnitrace][78506][sampling_gpu_busy_percent]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_busy_percent.json'
[omnitrace][78506][sampling_gpu_busy_percent]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/sampling_gpu_busy_percent.txt'
[omnitrace][78506][wall_clock]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/wall_clock.json'
[omnitrace][78506][wall_clock]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/wall_clock.txt'
[omnitrace][78506][metadata]> Outputting 'omnitrace-matrix.inst-output/2022-11-16_00.45/metadata-78506.json' and 'omnitrace-matrix.inst-output/2022-11-16_00.45/functions-78506.json'
[omnitrace][78506][0][omnitrace_finalize] Finalized
[303.572]       perfetto.cc:57383 Tracing session 1 ended, total sessions:0
```

Visualization with hardware counters

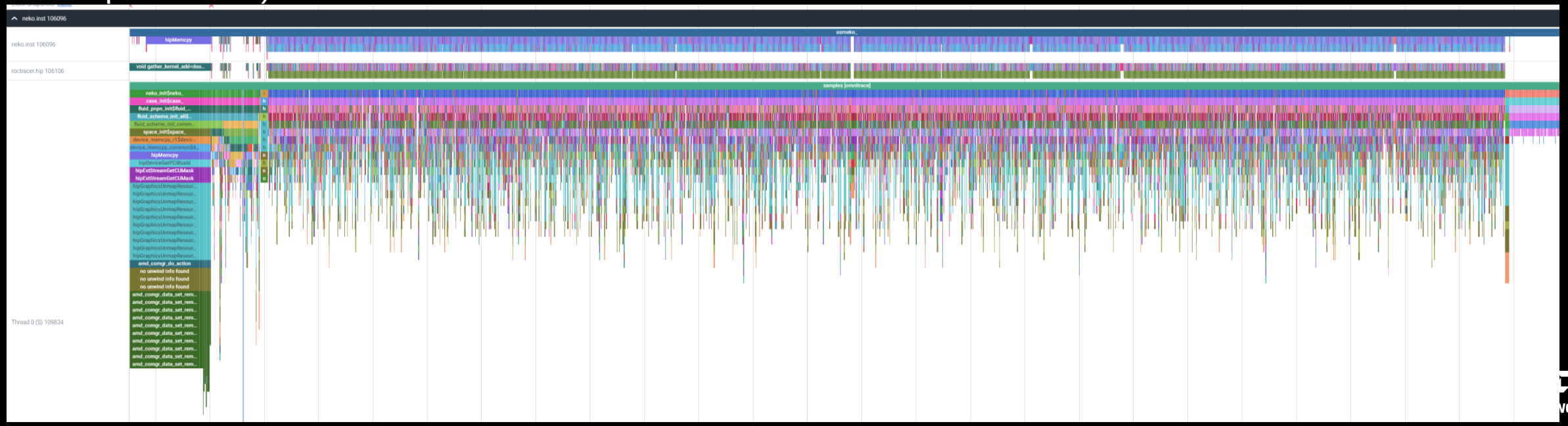


Sampling call-stack (I)

- Another application with OMNITRACE_USE_SAMPLING = false

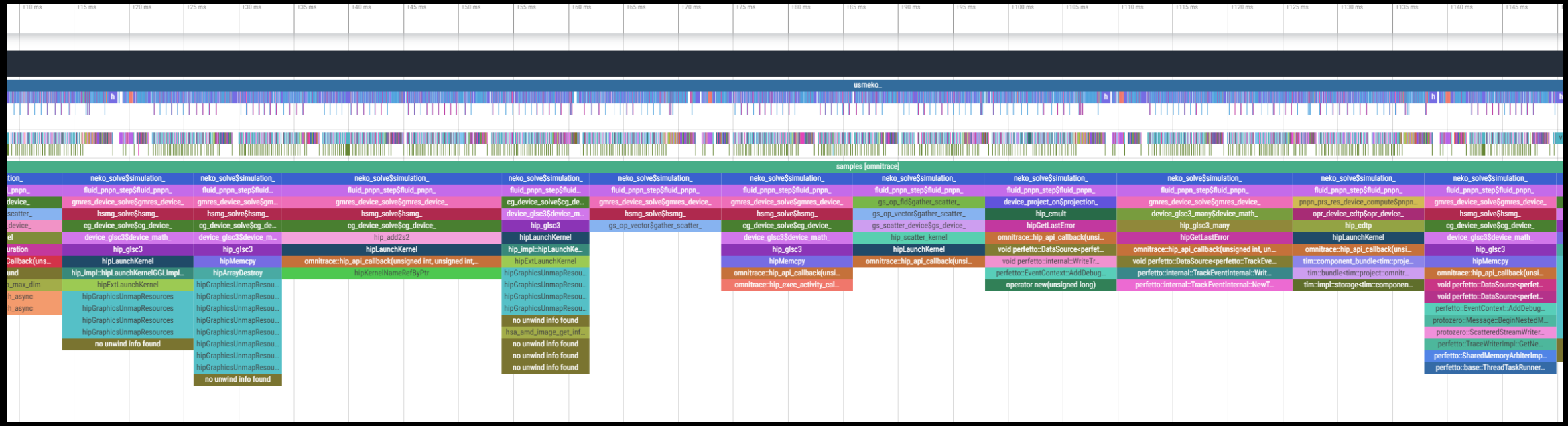


- With OMNITRACE_USE_SAMPLING = true and OMNITRACE_SAMPLING_FREQ = 100 (100 samples per second)



Sampling call-stack (II)

- Zoom in call-stack sampling



How to see kernels timing?

- omnitrace-binary-output/timestamp/wall_clock.txt

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> main	1	0	wall_clock	sec	21.811922	21.811922	21.811922	21.811922	0.000000	0.000000	46.3
|0>>> |_mbind	23	1	wall_clock	sec	0.000041	0.000002	0.000001	0.000004	0.000000	0.000001	100.0
|0>>> |_pthread_create	1	1	wall_clock	sec	0.023345	0.023345	0.023345	0.023345	0.000000	0.000000	100.0
|1>>> |_start_thread	-	2	-	-	-	-	-	-	-	-	-
|0>>> |_hipDeviceGetName	1	1	wall_clock	sec	0.001030	0.001030	0.001030	0.001030	0.000000	0.000000	100.0
|0>>> |_hipMalloc	1076	1	wall_clock	sec	0.019050	0.000018	0.000001	0.000583	0.000000	0.000046	100.0
|0>>> |_hipMemcpy	92578	1	wall_clock	sec	6.052626	0.000065	0.000001	0.181018	0.000000	0.000605	99.7
|0>>> |_mbind	146	2	wall_clock	sec	0.000167	0.000001	0.000001	0.000003	0.000000	0.000001	100.0
|0>>> |_void gather_kernel_add<double>(double*, int, int, int const*, double const*, int, int const*, int, int cons...	52100	2	wall_clock	sec	0.001629	0.000000	0.000000	0.000006	0.000000	0.000000	100.0
|0>>> |_void scatter_kernel<double>(double*, int, int const*, double*, int, int const*, int, int const*, int const*)	52106	2	wall_clock	sec	0.002148	0.000000	0.000000	0.000248	0.000000	0.000001	100.0
|0>>> |_void coef_generate_dxyz_kernel<double, 8, 1024>(double*, double*, double*, double*, double*, double*, doubl...	1	2	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> |_void coef_generate_drst_kernel<double>(double*, double*, double*, double*, double*, double*, double*, doubl...	3	2	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> |_void coef_generate_geo_kernel<double, 8, 1024>(double*, double*, double*, double*, double*, double*, double...	1	2	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> |_void invcol1_kernel<double>(double*, int)	509	2	wall_clock	sec	0.000016	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> |_void glsum_kernel<double>(double const*, double*, int)	3	2	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> |_void reduce_kernel<double>(double*, int)	78705	2	wall_clock	sec	0.003255	0.000000	0.000000	0.000001	0.000000	0.000000	100.0

How to see kernels timing? (II

- Add/edit in your omnitrace.cfg file, OMNITRACE_USE_TIMEMORY = true and OMNITRACE_FLAT_PROFILE = true

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> usrneko_	1	0	wall_clock	sec	24.024075	24.024075	24.024075	24.024075	0.000000	0.000000	100.0
|0>>> mbind	580	0	wall_clock	sec	0.000540	0.000001	0.000000	0.000004	0.000000	0.000000	100.0
|0>>> pthread_create	1	0	wall_clock	sec	0.006690	0.006690	0.006690	0.006690	0.000000	0.000000	100.0
|0>>> hipDeviceGetName	1	0	wall_clock	sec	0.000632	0.000632	0.000632	0.000632	0.000000	0.000000	100.0
|0>>> hipMalloc	1076	0	wall_clock	sec	0.029519	0.000027	0.000001	0.000373	0.000000	0.000061	100.0
|0>>> hipMemcpy	92578	0	wall_clock	sec	6.805347	0.000074	0.000001	0.621693	0.000004	0.002046	100.0
|0>>> hipDeviceSynchronize	20	0	wall_clock	sec	0.020044	0.001002	0.000002	0.002453	0.000000	0.000698	100.0
|0>>> hipLaunchKernel	510053	0	wall_clock	sec	4.547851	0.000009	0.000004	0.014506	0.000000	0.000030	100.0
|0>>> hipGetLastError	510053	0	wall_clock	sec	0.762807	0.000001	0.000001	0.031479	0.000000	0.000055	100.0
|0>>> void gather_kernel_add<double>(double*, int, int, int const*, double const*, int, int const*, int, int const*, ...	54121	0	wall_clock	sec	0.001754	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void scatter_kernel<double>(double*, int, int const*, double*, int, int const*, int, int const*, int const*)	54121	0	wall_clock	sec	0.002088	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> hipFree	937	0	wall_clock	sec	0.016387	0.000017	0.000002	0.001981	0.000000	0.000097	100.0
|0>>> hip_coef_generate_dxyzdrst	3	0	wall_clock	sec	0.006214	0.002071	0.000063	0.006060	0.000012	0.003455	100.0
|0>>> void coef_generate_dxyz_kernel<double, 8, 1024>(double*, double*, double*, double*, double*, double*, double*, ...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void coef_generate_drst_kernel<double>(double*, double*, double*, double*, double*, double*, double*, double*, ...	3	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> hip_coef_generate_geo	3	0	wall_clock	sec	0.000125	0.000042	0.000032	0.000055	0.000000	0.000012	100.0
|0>>> void coef_generate_geo_kernel<double, 8, 1024>(double*, double*, double*, double*, double*, double*, double con...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void invcol1_kernel<double>(double*, int)	509	0	wall_clock	sec	0.000017	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> hipHostMalloc	16	0	wall_clock	sec	0.000871	0.000054	0.000035	0.000071	0.000000	0.000014	100.0
|0>>> void glsum_kernel<double>(double const*, double*, int)	3	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void reduce_kernel<double>(double*, int)	78719	0	wall_clock	sec	0.003757	0.000000	0.000000	0.000271	0.000000	0.000001	100.0
|0>>> void cfill_kernel<double>(double*, double, int)	2014	0	wall_clock	sec	0.000066	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void jacobi_kernel<double, 8>(double*, double const*, double const*, double const*, double const*, double const...	502	0	wall_clock	sec	0.000016	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void col2_kernel<double>(double*, double const*, int)	10501	0	wall_clock	sec	0.000915	0.000000	0.000000	0.000574	0.000000	0.000006	100.0
|0>>> void coef_generate_dxyz_kernel<double, 2, 1024>(double*, double*, double*, double*, double*, double*, double*, ...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void coef_generate_geo_kernel<double, 2, 1024>(double*, double*, double*, double*, double*, double*, double con...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void coef_generate_dxyz_kernel<double, 4, 1024>(double*, double*, double*, double*, double*, double*, double*, ...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void coef_generate_geo_kernel<double, 4, 1024>(double*, double*, double*, double*, double*, double*, double con...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> hipMemcpyAsync	10012	0	wall_clock	sec	0.081116	0.000008	0.000004	0.000568	0.000000	0.000011	100.0
|0>>> void jacobi_kernel<double, 2>(double*, double const*, double const*, double const*, double const*, double const...	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> void tnsr3d_kernel<double>(double*, int, double const*, int, double const*, double const*, double const*)	11011	0	wall_clock	sec	0.000388	0.000000	0.000000	0.000003	0.000000	0.000000	100.0
|0>>> void cfl_kernel<double, 8, 1024>(double, double const*, double const*, double const*, double const*, double con...	501	0	wall_clock	sec	0.000028	0.000000	0.000000	0.000000	0.000000	0.000000	100.0

User API

- Omnitrace provides an API to control the instrumentation

API Call	Description
`int` **`omnitrace_user_start_trace(void)`**	Enable tracing on this thread and all subsequently created threads
`int` **`omnitrace_user_stop_trace(void)`**	Disable tracing on this thread and all subsequently created threads
`int` **`omnitrace_user_start_thread_trace(void)`**	Enable tracing on this specific thread. Does not apply to subsequently created threads
`int` **`omnitrace_user_stop_thread_trace(void)`**	Disable tracing on this specific thread. Does not apply to subsequently created threads

All the API calls: https://amdresearch.github.io/omnitrace/user_api.html

MPI

- We use the example omnitrace/examples/mpi/mpi.cpp
- Compile and run it to check the output, then create an instrumented binary

```
srun -n 1 omnitrace -o mpi.inst -- ./mpi
srun -n 2 ./mpi.inst
```

MPI 0

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> main	1	0	wall_clock	sec	2.308613	2.308613	2.308613	2.308613	0.000000	0.000000	86.7
|0>>> |_MPI_Init_thread	1	1	wall_clock	sec	0.298743	0.298743	0.298743	0.298743	0.000000	0.000000	99.5
|0>>> |_mbind	10	2	wall_clock	sec	0.000011	0.000001	0.000001	0.000002	0.000000	0.000001	100.0
|0>>> |_pthread_create	2	2	wall_clock	sec	0.001410	0.000705	0.000564	0.000847	0.000000	0.000200	0.0
|2>>> |_start_thread	1	3	wall_clock	sec	0.195632	0.195632	0.195632	0.195632	0.000000	0.000000	100.0
|1>>> |_start_thread	-	3	-	-	-	-	-	-	-	-	-
|0>>> |_pthread_create	1	1	wall_clock	sec	0.001182	0.001182	0.001182	0.001182	0.000000	0.000000	0.0
|3>>> |_start_thread	1	2	wall_clock	sec	0.002902	0.002902	0.002902	0.002902	0.000000	0.000000	62.7
|3>>> |_MPI_Comm_size	13	3	wall_clock	sec	0.000031	0.000002	0.000000	0.000024	0.000000	0.000006	100.0
|3>>> |_MPI_Comm_rank	5	3	wall_clock	sec	0.000004	0.000001	0.000000	0.000000	0.000000	0.000000	100.0
|3>>> |_MPI_Barrier	6	3	wall_clock	sec	0.000972	0.00					
|3>>> |_MPI_Send	8	3	wall_clock	sec	0.000017	0.00					
|3>>> |_MPI_Recv	8	3	wall_clock	sec	0.000021	0.00					
|3>>> |_MPI_Alltoall	8	3	wall_clock	sec	0.000030	0.00					
|3>>> |_MPI_Comm_dup	1	3	wall_clock	sec	0.000008	0.00					
|0>>> |_pthread_join	2	1	wall_clock	sec	0.007953	0.00					

MPI 1

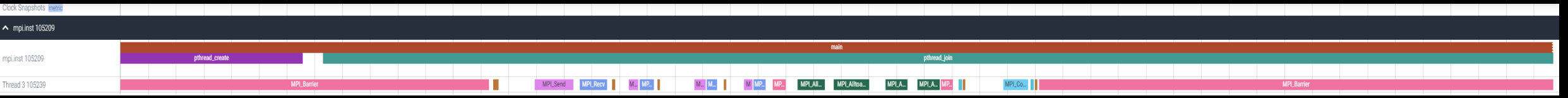
REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> main	1	0	wall_clock	sec	2.306350	2.306350	2.306350	2.306350	0.000000	0.000000	86.8
|0>>> |_MPI_Init_thread	1	1	wall_clock	sec	0.293291	0.293291	0.293291	0.293291	0.000000	0.000000	99.2
|0>>> |_mbind	10	2	wall_clock	sec	0.000014	0.000001	0.000001	0.000004	0.000000	0.000001	100.0
|0>>> |_pthread_create	2	2	wall_clock	sec	0.002338	0.001169	0.000897	0.001441	0.000000	0.000384	0.0
|2>>> |_start_thread	1	3	wall_clock	sec	0.193902	0.193902	0.193902	0.193902	0.000000	0.000000	100.0
|1>>> |_start_thread	-	3	-	-	-	-	-	-	-	-	-
|0>>> |_pthread_create	1	1	wall_clock	sec	0.006592	0.006592	0.006592	0.006592	0.000000	0.000000	0.0
|3>>> |_start_thread	1	2	wall_clock	sec	0.007850	0.007850	0.007850	0.007850	0.000000	0.000000	16.4
|3>>> |_MPI_Comm_size	13	3	wall_clock	sec	0.000031	0.000002	0.000000	0.000024	0.000000	0.000007	100.0
|3>>> |_MPI_Comm_rank	5	3	wall_clock	sec	0.000009	0.000002	0.000000	0.000006	0.000000	0.000002	100.0
|3>>> |_MPI_Barrier	6	3	wall_clock	sec	0.006405	0.001068	0.000001	0.005604	0.000005	0.002244	100.0
|3>>> |_MPI_Send	8	3	wall_clock	sec	0.000020	0.000003	0.000001	0.000012	0.000000	0.000004	100.0
|3>>> |_MPI_Recv	8	3	wall_clock	sec	0.000027	0.000003	0.000002	0.000009	0.000000	0.000002	100.0
|3>>> |_MPI_Alltoall	8	3	wall_clock	sec	0.000060	0.000007	0.000003	0.000011	0.000000	0.000003	100.0
|3>>> |_MPI_Comm_dup	1	3	wall_clock	sec	0.000008	0.000008	0.000008	0.000008	0.000000	0.000000	100.0
|0>>> |_pthread_join	2	1	wall_clock	sec	0.005277	0.002638	0.001800	0.003477	0.000001	0.001186	100.0

MPI visualizing one Perfetto per MPI process

MPI 0

MPI 1

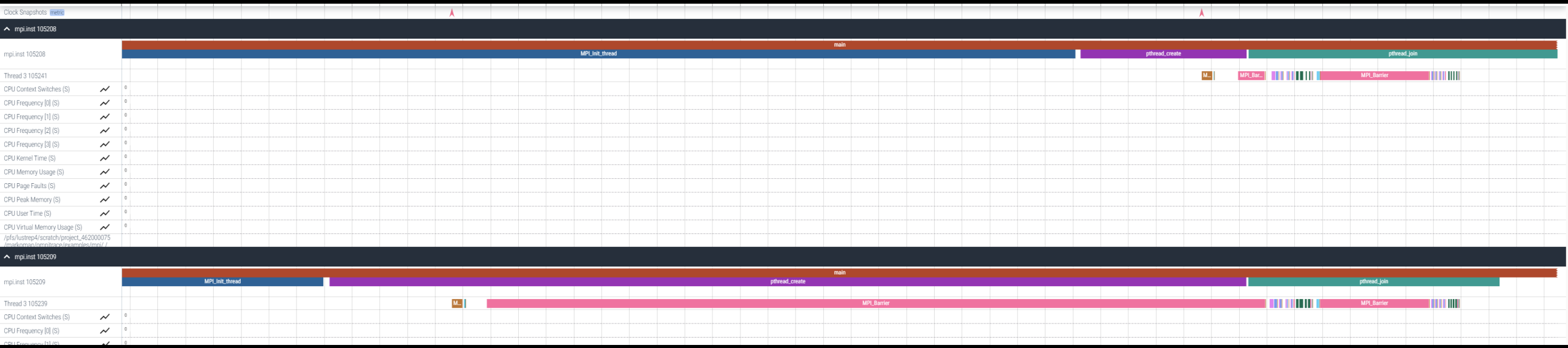


Visualizing all the MPI processes in once

- Merge the Perfetto files:

```
cat omnitrace-mpi.inst-output/timestamp/perfetto-trace-0.proto omnitrace-mpi.inst-output/timestamp/perfetto-trace-1.proto > allprocesses.proto
```

- For large number or processes a different approach is required if willing to visualize many processes



OpenMP®

- We use the example /omnitrace/examples/openmp/
- Build the code:

```
cmake –B build .
```

- We use the openmp-lu binary, execution:

```
export OPENMP_NUM_THREADS=4
srun –n 1 –c 4 ./openmp-lu
```

- Create a new instrumented binary:

```
srun -n 1 omnitrace -o openmp-lu.inst -- ./openmp-lu
```

OpenMP® (II)

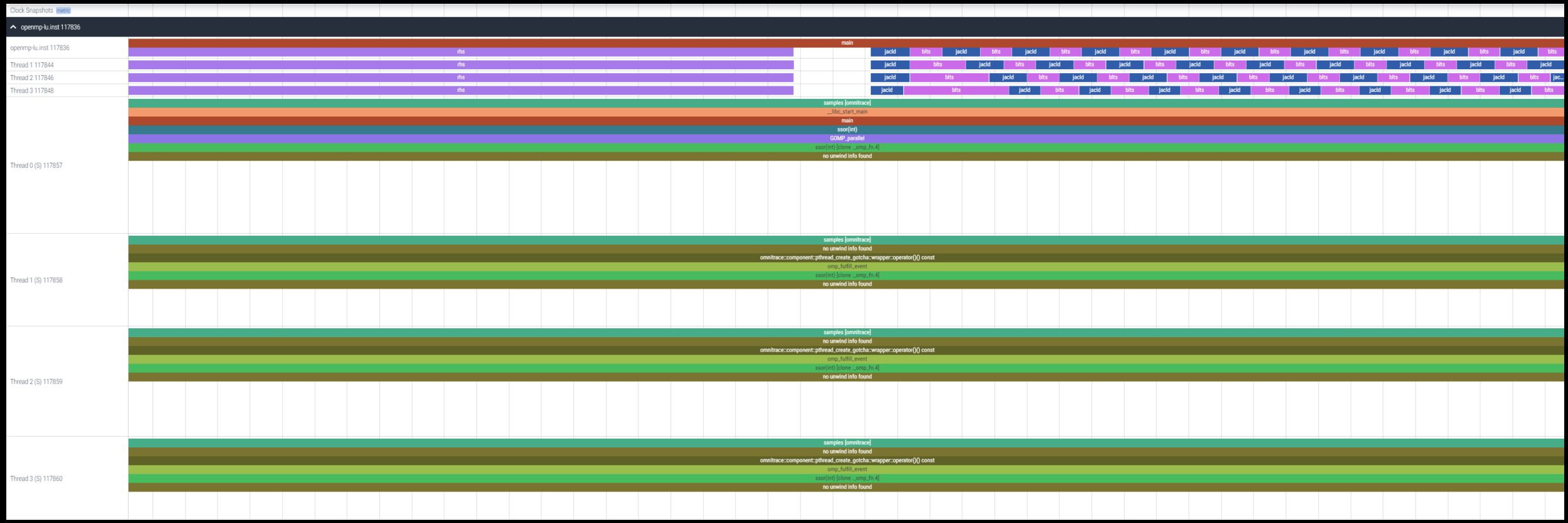
- Execution:

srun -n 1 –c 4 ./openmp-lu.inst

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> main	1	0	wall_clock	sec	1.096702	1.096702	1.096702	1.096702	0.000000	0.000000	9.2
|0>>> |_pthread_create	3	1	wall_clock	sec	0.002931	0.000977	0.000733	0.001420	0.000000	0.000385	0.0
|3>>> |_start_thread	1	2	wall_clock	sec	2.451520	2.451520	2.451520	2.451520	0.000000	0.000000	57.7
|3>>> |_erhs	1	3	wall_clock	sec	0.001906	0.001906	0.001906	0.001906	0.000000	0.000000	100.0
|3>>> |_rhs	153	3	wall_clock	sec	0.229893	0.001503	0.001410	0.001893	0.000000	0.000116	100.0
|3>>> |_jacld	3473	3	wall_clock	sec	0.170568	0.000049	0.000047	0.000135	0.000000	0.000005	100.0
|3>>> |_blts	3473	3	wall_clock	sec	0.232512	0.000067	0.000040	0.000959	0.000000	0.000034	100.0
|3>>> |_jacu	3473	3	wall_clock	sec	0.166229	0.000048	0.000046	0.000148	0.000000	0.000005	100.0
|3>>> |_buts	3473	3	wall_clock	sec	0.236484	0.000068	0.000041	0.000391	0.000000	0.000031	100.0
|2>>> |_start_thread	1	2	wall_clock	sec	2.452309	2.452309	2.452309	2.452309	0.000000	0.000000	58.1
|2>>> |_erhs	1	3	wall_clock	sec	0.001895	0.001895	0.001895	0.001895	0.000000	0.000000	100.0
|2>>> |_rhs	153	3	wall_clock	sec	0.229776	0.001502	0.001410	0.001893	0.000000	0.000115	100.0
|2>>> |_jacld	3473	3	wall_clock	sec	0.204609	0.000059	0.000057	0.000152	0.000000	0.000006	100.0
|2>>> |_blts	3473	3	wall_clock	sec	0.192986	0.000056	0.000047	0.000358	0.000000	0.000026	100.0
|2>>> |_jacu	3473	3	wall_clock	sec	0.199029	0.000057	0.000055	0.000188	0.000000	0.000007	100.0
|2>>> |_buts	3473	3	wall_clock	sec	0.198972	0.000057	0.000048	0.000372	0.000000	0.000026	100.0
|1>>> |_start_thread	1	2	wall_clock	sec	2.453072	2.453072	2.453072	2.453072	0.000000	0.000000	58.6
|1>>> |_erhs	1	3	wall_clock	sec	0.001905	0.001905	0.001905	0.001905	0.000000	0.000000	100.0
|1>>> |_rhs	153	3	wall_clock	sec	0.229742	0.001502	0.001410	0.001894	0.000000	0.000115	100.0
|1>>> |_jacld	3473	3	wall_clock	sec	0.206418	0.000059	0.000057	0.000934	0.000000	0.000016	100.0
|1>>> |_blts	3473	3	wall_clock	sec	0.186097	0.000054	0.000047	0.000344	0.000000	0.000023	100.0
|1>>> |_jacu	3473	3	wall_clock	sec	0.198689	0.000057	0.000055	0.000186	0.000000	0.000006	100.0
|1>>> |_buts	3473	3	wall_clock	sec	0.192470	0.000055	0.000048	0.000356	0.000000	0.000022	100.0
|0>>> |_erhs	1	1	wall_clock	sec	0.001961	0.001961	0.001961	0.001961	0.000000	0.000000	100.0
|0>>> |_rhs	153	1	wall_clock	sec	0.229889	0.001503	0.001410	0.001891	0.000000	0.000116	100.0
|0>>> |_jacld	3473	1	wall_clock	sec	0.208903	0.000060	0.000057	0.000359	0.000000	0.000017	100.0
|0>>> |_blts	3473	1	wall_clock	sec	0.172646	0.000050	0.000047	0.000822	0.000000	0.000020	100.0
|0>>> |_jacu	3473	1	wall_clock	sec	0.202130	0.000058	0.000055	0.000350	0.000000	0.000016	100.0
|0>>> |_buts	3473	1	wall_clock	sec	0.176975	0.000051	0.000048	0.000377	0.000000	0.000016	100.0
|0>>> |_pintgr	1	1	wall_clock	sec	0.000054	0.000054	0.000054	0.000054	0.000000	0.000000	100.0

OpenMP® visualization



Python™

- The omnitrace Python package is installed in /path/omnitrace_install/lib/pythonX.Y/site-packages/omnitrace
- Setup the environment

```
export PYTHONPATH=/path/omnitrace/lib/python/site-packages/:${PYTHONPATH}
```

- We use the Fibonacci example:

omnitrace/examples/python/source.py

- Execute:

```
srun -n 1 --gpus 1 omnitrace-python ./external.py
```

There will be a new directory called omnitrace-source-output with contents

Python documentation: https://amdresearch.github.io/omnitrace/python.html

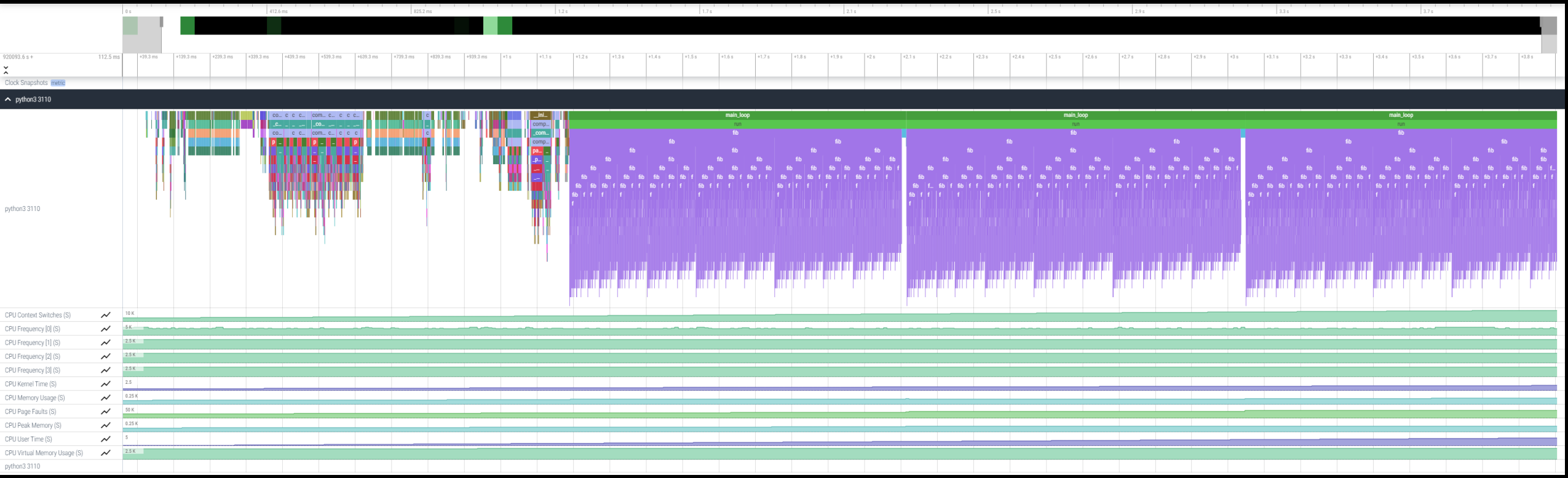
Python™ (II)

- omnitrace-source-output/timestamp/wall_clock.txt

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> main_loop	3	0	wall_clock	sec	2.786075	0.928692	0.926350	0.932130	0.000009	0.003042	0.0
|0>>> |_run	3	1	wall_clock	sec	2.785799	0.928600	0.926250	0.932037	0.000009	0.003043	0.0
|0>>> |_fib	3	2	wall_clock	sec	2.750104	0.916701	0.914454	0.919577	0.000007	0.002619	0.0
|0>>> |_fib	6	3	wall_clock	sec	2.749901	0.458317	0.348962	0.567074	0.013958	0.118145	0.0
|0>>> |_fib	12	4	wall_clock	sec	2.749511	0.229126	0.133382	0.350765	0.006504	0.080650	0.0
|0>>> |_fib	24	5	wall_clock	sec	2.748734	0.114531	0.050867	0.217030	0.002399	0.048977	0.1
|0>>> |_fib	48	6	wall_clock	sec	2.747118	0.057232	0.019302	0.134596	0.000806	0.028396	0.1
|0>>> |_fib	96	7	wall_clock	sec	2.743922	0.028583	0.007181	0.083350	0.000257	0.016026	0.2
|0>>> |_fib	192	8	wall_clock	sec	2.737564	0.014258	0.002690	0.051524	0.000079	0.008887	0.5
|0>>> |_fib	384	9	wall_clock	sec	2.724966	0.007096	0.000973	0.031798	0.000024	0.004865	0.9
|0>>> |_fib	768	10	wall_clock	sec	2.699251	0.003515	0.000336	0.019670	0.000007	0.002637	1.9
|0>>> |_fib	1536	11	wall_clock	sec	2.648006	0.001724	0.000096	0.012081	0.000002	0.001417	3.9
|0>>> |_fib	3072	12	wall_clock	sec	2.545260	0.000829	0.000016	0.007461	0.000001	0.000758	8.0
|0>>> |_fib	6078	13	wall_clock	sec	2.342276	0.000385	0.000016	0.004669	0.000000	0.000404	16.0
|0>>> |_fib	10896	14	wall_clock	sec	1.967475	0.000181	0.000015	0.002752	0.000000	0.000218	28.6
|0>>> |_fib	15060	15	wall_clock	sec	1.404069	0.000093	0.000015	0.001704	0.000000	0.000123	43.6
|0>>> |_fib	14280	16	wall_clock	sec	0.791873	0.000055	0.000015	0.001044	0.000000	0.000076	58.3
|0>>> |_fib	8826	17	wall_clock	sec	0.330189	0.000037	0.000015	0.000620	0.000000	0.000050	70.9
|0>>> |_fib	3456	18	wall_clock	sec	0.096120	0.000028	0.000015	0.000380	0.000000	0.000034	81.0
|0>>> |_fib	822	19	wall_clock	sec	0.018294	0.000022	0.000015	0.000209	0.000000	0.000024	88.9
|0>>> |_fib	108	20	wall_clock	sec	0.002037	0.000019	0.000016	0.000107	0.000000	0.000015	94.9
|0>>> |_fib	6	21	wall_clock	sec	0.000104	0.000017	0.000016	0.000019	0.000000	0.000001	100.0
|0>>> |_inefficient	3	2	wall_clock	sec	0.035450	0.011817	0.010096	0.012972	0.000002	0.001519	95.8
|0>>> |__sum	3	3	wall_clock	sec	0.001494	0.000498	0.000440	0.000537	0.000000	0.000051	100.0

Visualizing Python™ Perfeto tracing



Omnitrace-sample

- For easy usage of Omnitrace there is also the omnitrace-sample that does sampling with less overhead.
- It provides less overhead but you need to be sure that you do not miss information
- Not all the declarations of a cfg file apply, for example to use hardware counters, ou need to execute the following command:

```
srun -n 1 omnitrace-sample -TPHD -G
"GPUBusy:device=0,Wavefronts:device=0,VALUBusy:device=0,L2CacheHit:device=0,MemUnitBusy:device=0” -- ./binary
```

See omnitrace-sample -h for more information

Tips & Tricks

- My Perfetto timeline seems weird how can I check the clock skew?
  - OMNITRACE_VERBOSE equal to 1 or higher for verbose mode and it will print the timestamp skew
- Omnitrace takes too long time in the finalization, how to check which part takes a lot of time?
  - Use OMNITRACE_VERBOSE equal to 1 or higher for verbose mode
- It takes too long time to map rocm-smi samples to the kernels
  - Use temporarily OMNITRACE_USE_ROCM_SMI=OF
- If you are doing binary rewriting and you do not get information about kernels, declare:
  - HSA_TOOLS_LIB=libomnitrace.so in the environment and be sure that OMNITRACE_USE_ROCTRACER=ON in the cfg file
- My HIP application hangs in different points, what to do?
  - Try to set HSA_ENABLE_INTERRUPT=0 in the environment, this handles different how HIP is notified that GPU kernels completed
- It is preferred to use binary rewriting for MPI applications, in order to write one file per MPI process, and not aggregated, use: OMNITRACE_USE_PID=ON
- My Perfetto trace is too big, can I decrease it?
  - Yes, with v1.7.3 and later declare OMNITRACE_PERFETTO_ANNOTATIONS to false.
- Full documentation: https://amdresearch.github.io/omnitrace/

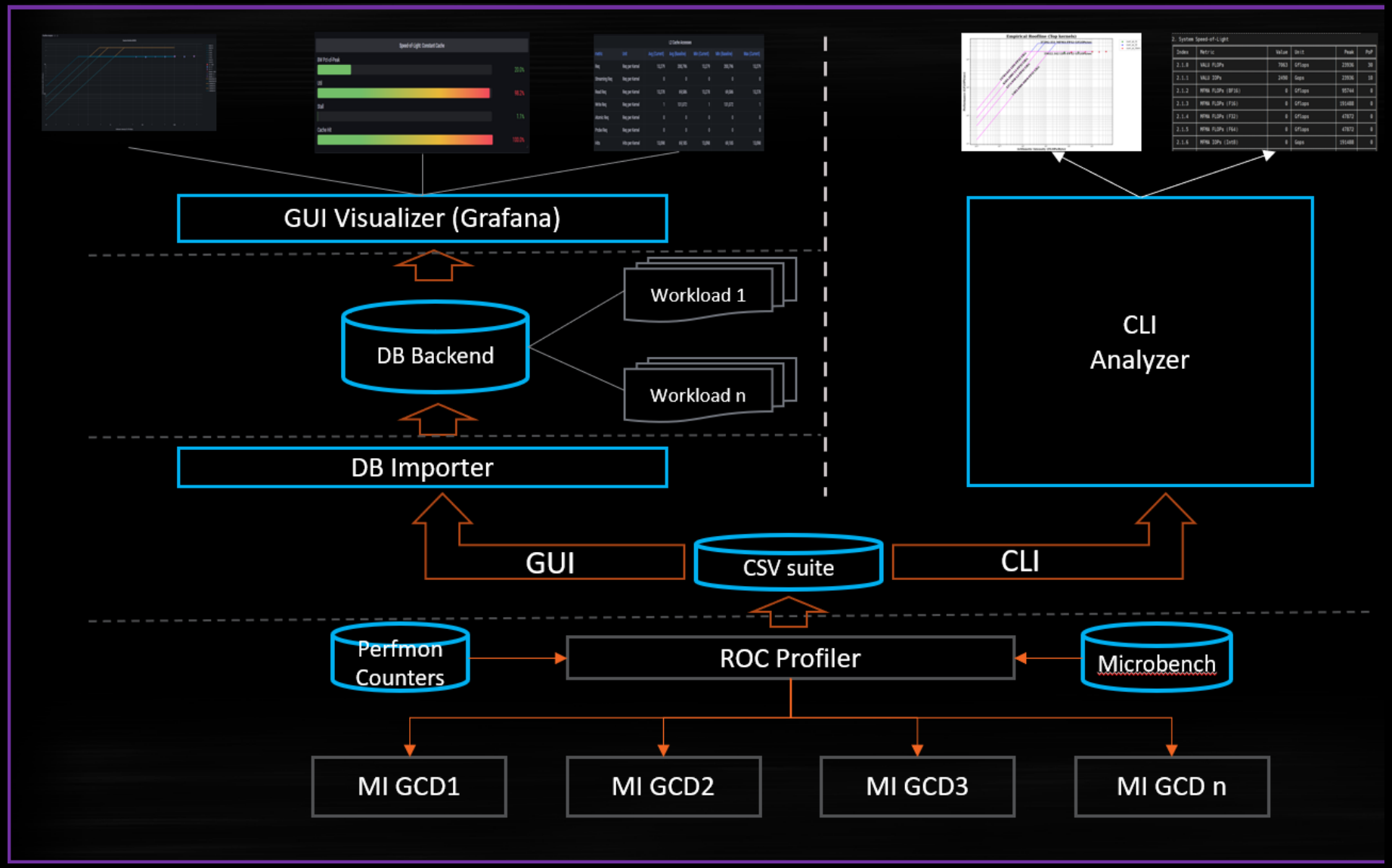
Omniperf

Omniperf

- The Omniperf executes the code as many times required based on the job submission
- Without specific option the application will be executed many times with various hardware counters (more than 100), so this can take long time. It does not mean that all the counters will provide useful data.
- There are various options for filtering (kernel, metric) even to execute mainly for roofline analysis
- There are many data per metric/HW and we will show a few, Omniperf provides tables for every metric
- With Omniperf first we profile, then we analyze and then we can import to database or visualize with standalone GUI
- The Omniperf targets MI100 and MI200 and later future generation AMD GPUs

- For problems, create an issue here: https://github.com/AMDResearch/omniperf/issues

Omniperf Architecture



Omniperf features

Omniperf Features	
MI200 support	Roofline Analysis Panel (*Supported on MI200 only, SLES 15 SP3 or RHEL8*)
MI100 support	Command Processor (CP) Panel
Standalone GUI Analyzer	Shader Processing Input (SPI) Panel
Grafana/MongoDB GUI Analyzer	Wavefront Launch Panel
Dispatch Filtering	Compute Unit - Instruction Mix Panel
Kernel Filtering	Compute Unit - Pipeline Panel
GPU ID Filtering	Local Data Share (LDS) Panel
Baseline Comparison	Instruction Cache Panel
Multi-Normalizations	Scalar L1D Cache Panel
System Info Panel	Texture Addresser and Data Panel
System Speed-of-Light Panel	Vector L1D Cache Panel
Kernel Statistic Panel	L2 Cache Panel
Memory Chart Analysis Panel	L2 Cache (per-Channel) Panel

Client-side installation (if required)

- Download the latest version from here: https://github.com/AMDResearch/omniperf/releases

```
wget https://github.com/AMDResearch/omniperf/releases/download/v1.0.4/omniperf-
1.0.4.tar.gz

tar zxvf omniperf-1.0.4.tar.gz
module load rocm
cd omniperf-1.0.4/
python3 -m pip install -t ${INSTALL_DIR}/python-libs -r requirements.txt
mkdir build
cd build
export PYTHONPATH=$INSTALL_DIR/python-libs
cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR}/1.0.4 \
        -DPYTHON_DEPS=${INSTALL_DIR}/python-libs \
        -DMOD_INSTALL_PATH=${INSTALL_DIR}/modulefiles ..
make install
export PATH=$INSTALL_DIR/1.0.4/bin:$PATH
```

Omniperf modes

- Profiling

```
omniperf profile  -n workload_name [profile options] [roofline options] --
<profile_cmd>
```

- Analysis

```
omniperf analyze -p workloads/workload_name/mi200/
```

- GUI import

```
omniperf database --import [CONNECTION OPTIONS]
```

- GUI standalone

```
omniperf analyze -p workloads/workload_name/mi200/ --gui
```

Then follow the instructions to open the web page for the GUI

Omniperf Profiling

- We use the example sample/vcopy.cpp from the Omniperf installation folder (cp /global/training/enccs/omniperf/1.0.4/share/sample/vcopy.cpp .)
- Compile with hipcc, let's call the binary vcopy
- Load Omniperf module
- Profiling with the default set pf data for all kernels, execute:

```
srun -n 1 --gpus 1 omniperf profile -n vcopy_all -- ./vcopy 1048576 256
```

```
…
-------------
Profile only
-------------

omniperf ver:  1.0.4
Path:  /pfs/lustrep4/scratch/project_462000075/markoman/omniperf-1.0.4/build/workloads
Target:  mi200
Command:  ./vcopy 1048576 256
Kernel Selection:  None
Dispatch Selection:  None
IP Blocks: All
```

In this case we call the workload name "vcopy_all" and after the "--" everything is about the application we execute. In this case, the application will be executed many times for collecting different metrics, if the application takes significant time to run once, then this could b not the optimum approach.

At the end of the execution, we have a folder workloads/vcopy_all/mi200/
You can see all the options with the command `omniperf profile --help`

Omniperf Analyze

- We use the example sample/vcopy.cpp from the Omniperf installation folder

```
srun -n 1 --gpus 1 omniperf analyze -p workloads/vcopy_all/mi200/ &>
vcopy_analyze.txt
```

0. Top Stat

	KernelName	Count	Sum(ns)	Mean(ns)	Median(ns)	Pct
0	vecCopy(double*, double*, double*, int, int) [clone .kd]	1	341123.00	341123.00	341123.00	100.00

2. System Speed-of-Light

Index	Metric	Value	Unit	Peak	PoP
2.1.0	VALU FLOPs	0.00	Gflop	23936.0	0.0
2.1.1	VALU IOPs	89.14	Giop	23936.0	0.37242200388114116
2.1.2	MFMA FLOPs (BF16)	0.00	Gflop	95744.0	0.0
2.1.3	MFMA FLOPs (F16)	0.00	Gflop	191488.0	0.0
2.1.4	MFMA FLOPs (F32)	0.00	Gflop	47872.0	0.0
2.1.5	MFMA FLOPs (F64)	0.00	Gflop	47872.0	0.0
2.1.6	MFMA IOPs (Int8)	0.00	Giop	191488.0	0.0
2.1.7	Active CUs	58.00	Cus	110	52.72727272727273
2.1.8	SALU Util	3.69	Pct	100	3.6862586934167525
2.1.9	VALU Util	5.90	Pct	100	5.895531580380328
2.1.10	MFMA Util	0.00	Pct	100	0.0
2.1.11	VALU Active Threads/Wave	32.71	Threads	64	51.10526315789473
2.1.12	IPC - Issue	0.98	Instr/cycle	5	19.576640831930312

7.1 Wavefront Launch Stats

Index	Metric	Avg	Min	Max	Unit
7.1.0	Grid Size	1048576.00	1048576.00	1048576.00	Work items
7.1.1	Workgroup Size	256.00	256.00	256.00	Work items
7.1.2	Total Wavefronts	16384.00	16384.00	16384.00	Wavefronts
7.1.3	Saved Wavefronts	0.00	0.00	0.00	Wavefronts
7.1.4	Restored Wavefronts	0.00	0.00	0.00	Wavefronts
7.1.5	VGPRs	44.00	44.00	44.00	Registers
7.1.6	SGPRs	48.00	48.00	48.00	Registers
7.1.7	LDS Allocation	0.00	0.00	0.00	Bytes
7.1.8	Scratch Allocation	16496.00	16496.00	16496.00	Bytes

Omniperf Analyze (II)

- Execute omniperf analyze –h to see various options
- Use specific IP block (-b)
- Top kernel:

```
srun -n 1 --gpus 1 omniperf analyze -p workloads/vcopy_all/mi200/ -b 0
```

- IP Block of wavefronts: `srun -n 1 --gpus 1 omniperf analyze -p workloads/vcopy_all/mi200/ -b 7.1.2`

```
--------------------------------------------------------------------------------
0. Top Stat
```

	KernelName	Count	Sum(ns)	Mean(ns)	Median(ns)	Pct
0	vecCopy(double*, double*, double*, int, int) [clone .kd]	1	20960.00	20960.00	20960.00	100.00

```
--------------------------------------------------------------------------------
7. Wavefront
7.1 Wavefront Launch Stats
```

Index	Metric	Avg	Min	Max	Unit
7.1.2	Total Wavefronts	16384.00	16384.00	16384.00	Wavefronts

Omniperf Analyze (III)

omniperf analyze -h

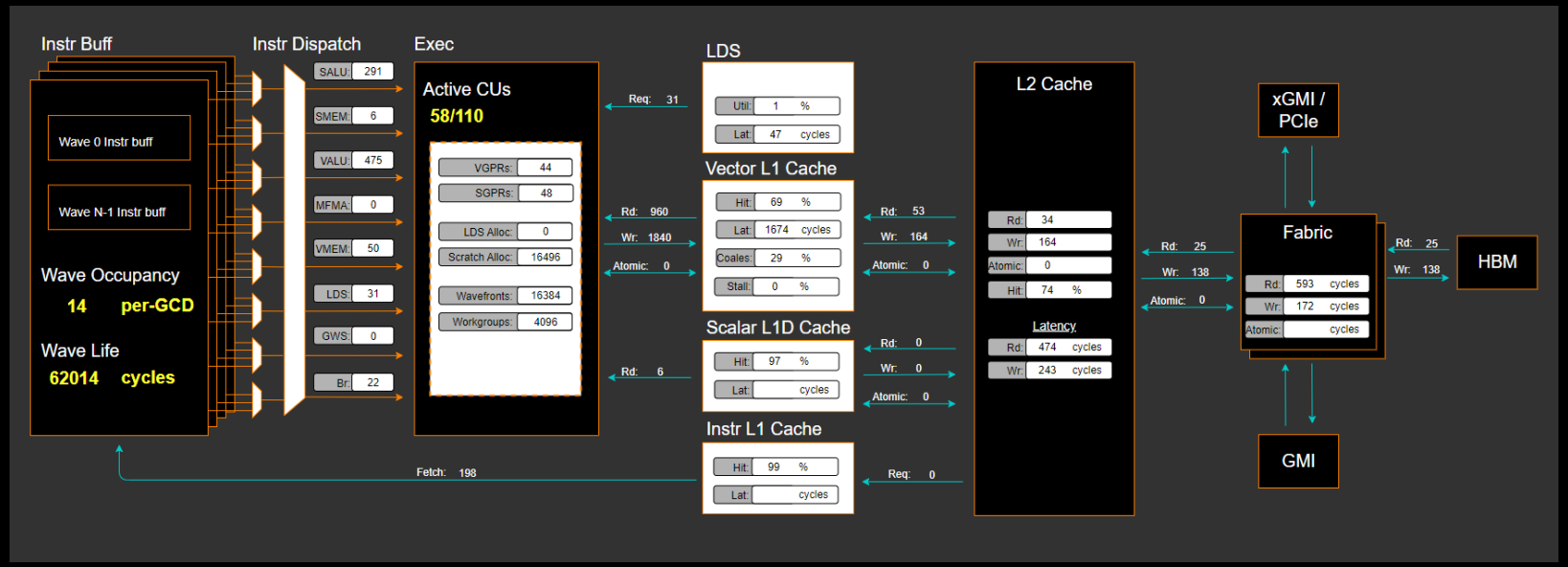
```
Help:
  -h, --help                            show this help message and exit

General Options:
  -v, --version                         show program's version number and exit
  -V, --verbose                         Increase output verbosity

Analyze Options:
  -p  [ ...], --path  [ ...]                        Specify the raw data root dirs or desired results directory.
  -o , --output                                     Specify the output file.
  --list-kernels                                    List kernels.
  --list-metrics                                    List metrics can be customized to analyze on specific arch:
                                                       gfx906
                                                       gfx908
                                                       gfx90a
  -b  [ ...], --filter-metrics  [ ...]              Specify IP block/metric Ids from --list-metrics.
  -k  [ ...], --filter-kernels  [ ...]              Specify kernel id from --list-kernels.
  --filter-dispatch-ids  [ ...]                     Specify dispatch IDs.
  --filter-gpu-ids  [ ...]                          Specify GPU IDs.
  -n , --normal-unit                                Specify the normalization unit: (DEFAULT: per_wave)
                                                       per_wave
                                                       per_cycle
                                                       per_second
  --config-dir                                      Specify the directory of customized configs.
  -t , --time-unit                                  Specify display time unit in kernel top stats: (DEFAULT: ns)
                                                       s
                                                       ms
                                                       us
                                                       ns
  --decimal                                         Specify the decimal to display. (DEFAULT: 2)
  --cols  [ ...]                                    Specify column indices to display.
  -g                                                Debug single metric.
  --dependency                                      List the installation dependency.
  --gui [GUI]                                       Activate a GUI to interate with Omniperf metrics.
                                                    Optionally, specify port to launch application (DEFAULT: 8050)
```

Omniperf Analyze with standalone GUI

- Download the data on your computer (workloads/vcopy_all/), install Omniperf without ROCm, and execute:

```
omniperf analyze -p workloads/vcopy_all/mi200/ --gui
```

Open web page http://172.21.7.117:8050/



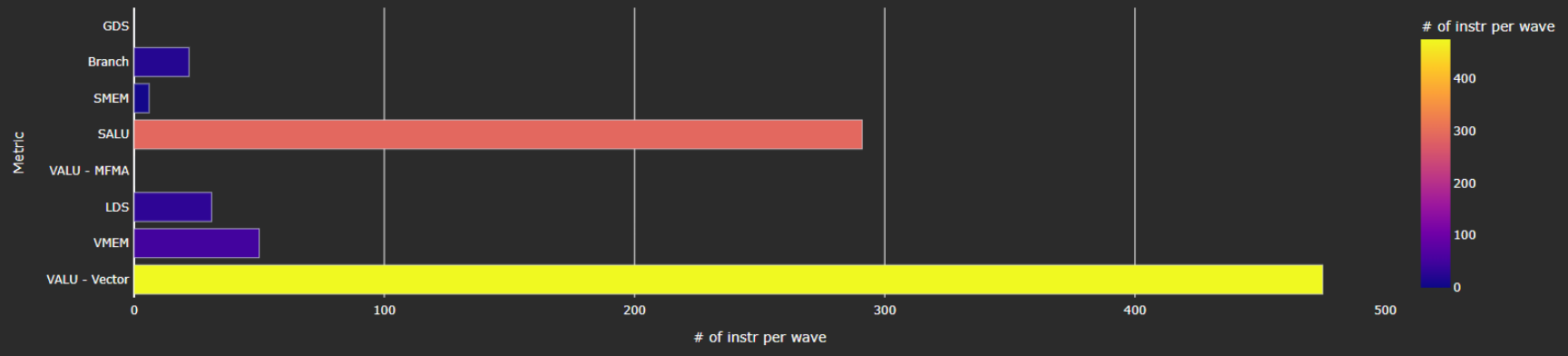
Omniperf Analyze with standalone GUI (II)

2. System Speed-of-Light

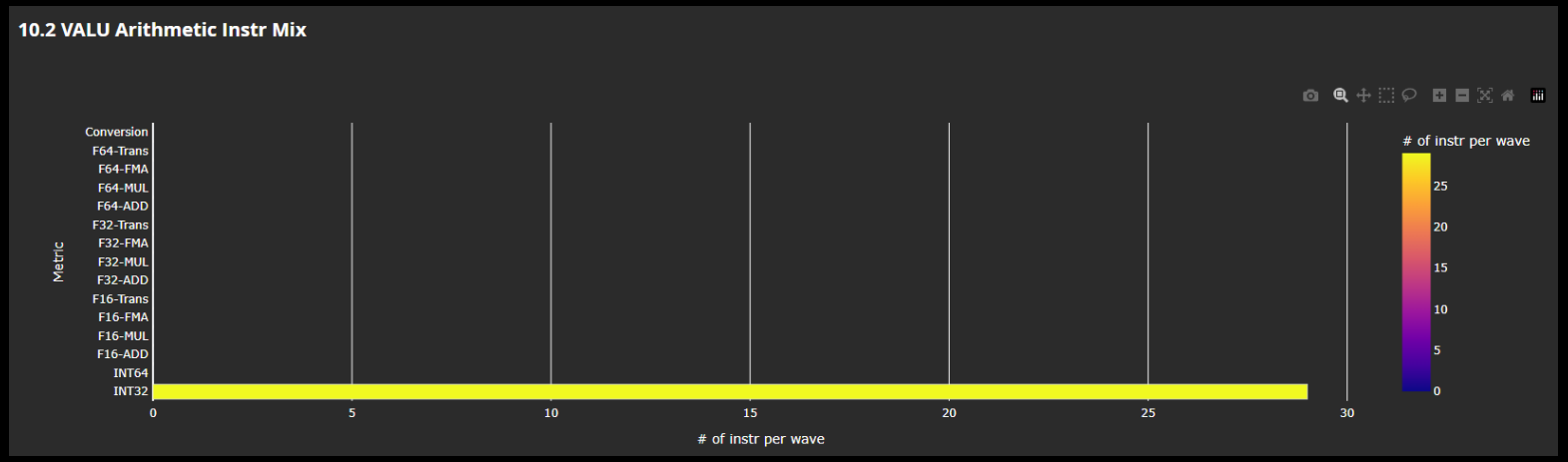
Metric	Value	Unit	Peak	PoP
VALU FLOPs	0.00	Gflop	23936.00	0.00
VALU IOPs	89.14	Giop	23936.00	0.37
MFMA FLOPs (BF16)	0.00	Gflop	95744.00	0.00
MFMA FLOPs (F16)	0.00	Gflop	191488.00	0.00
MFMA FLOPs (F32)	0.00	Gflop	47872.00	0.00
MFMA FLOPs (F64)	0.00	Gflop	47872.00	0.00
MFMA IOPs (Int8)	0.00	Giop	191488.00	0.00
Active CUs	58.00	Cus	110.00	52.73

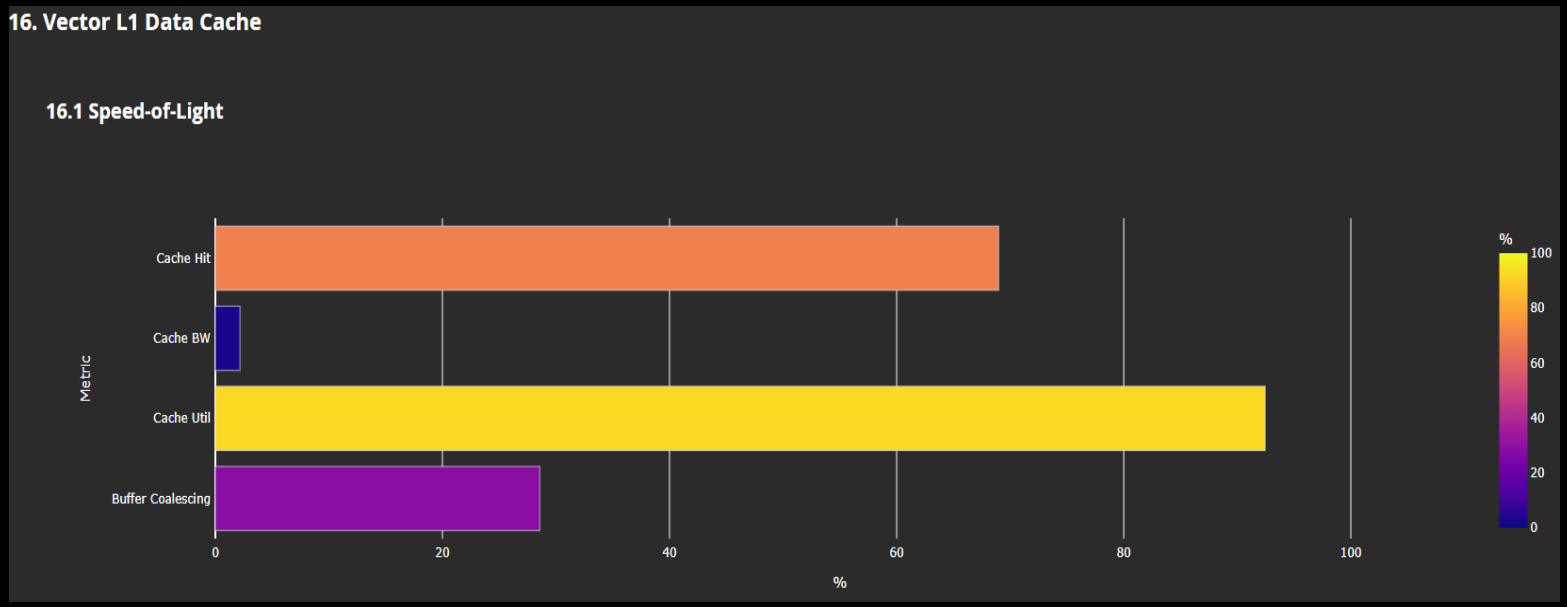
10. Compute Units - Instruction Mix

10.1 Instruction Mix



Omniperf Analyze with standalone GUI (III)



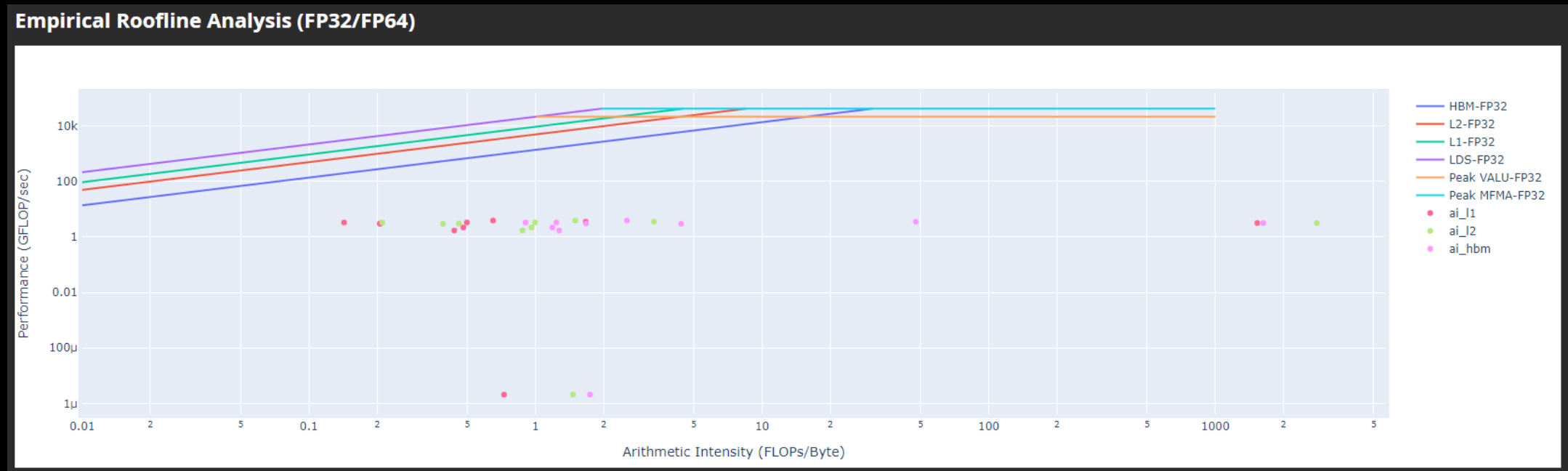


Roofline Analysis

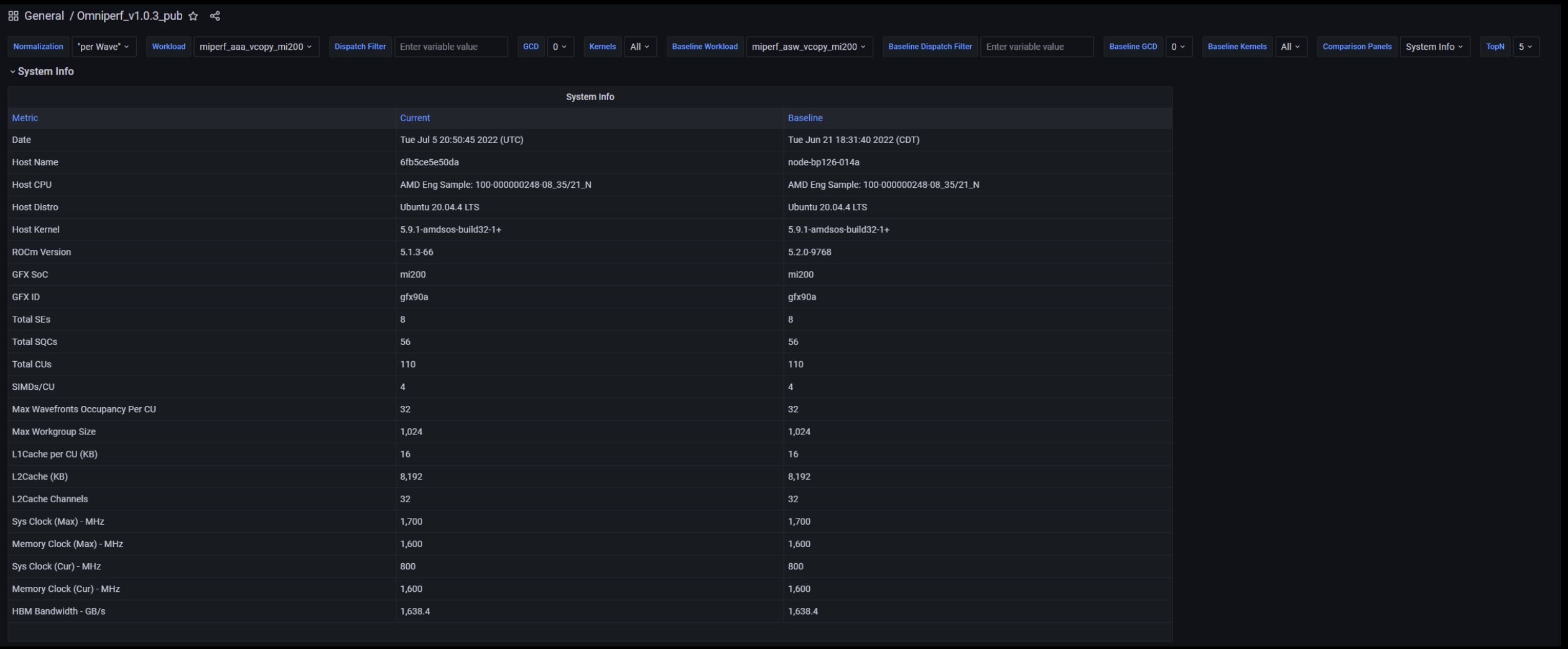
- Profile with roofline:

```
srun -n 1 --gpus 1 omniperf profile -n roofline_case_app --roof-only --
./app
```

- Prepare GUI:
  Copy the workload to your computer
  Execute: omniperf analyze -p workloads/roofline_case_app/mi200/ --gui
  Open the web page http://172.21.7.117:8050/



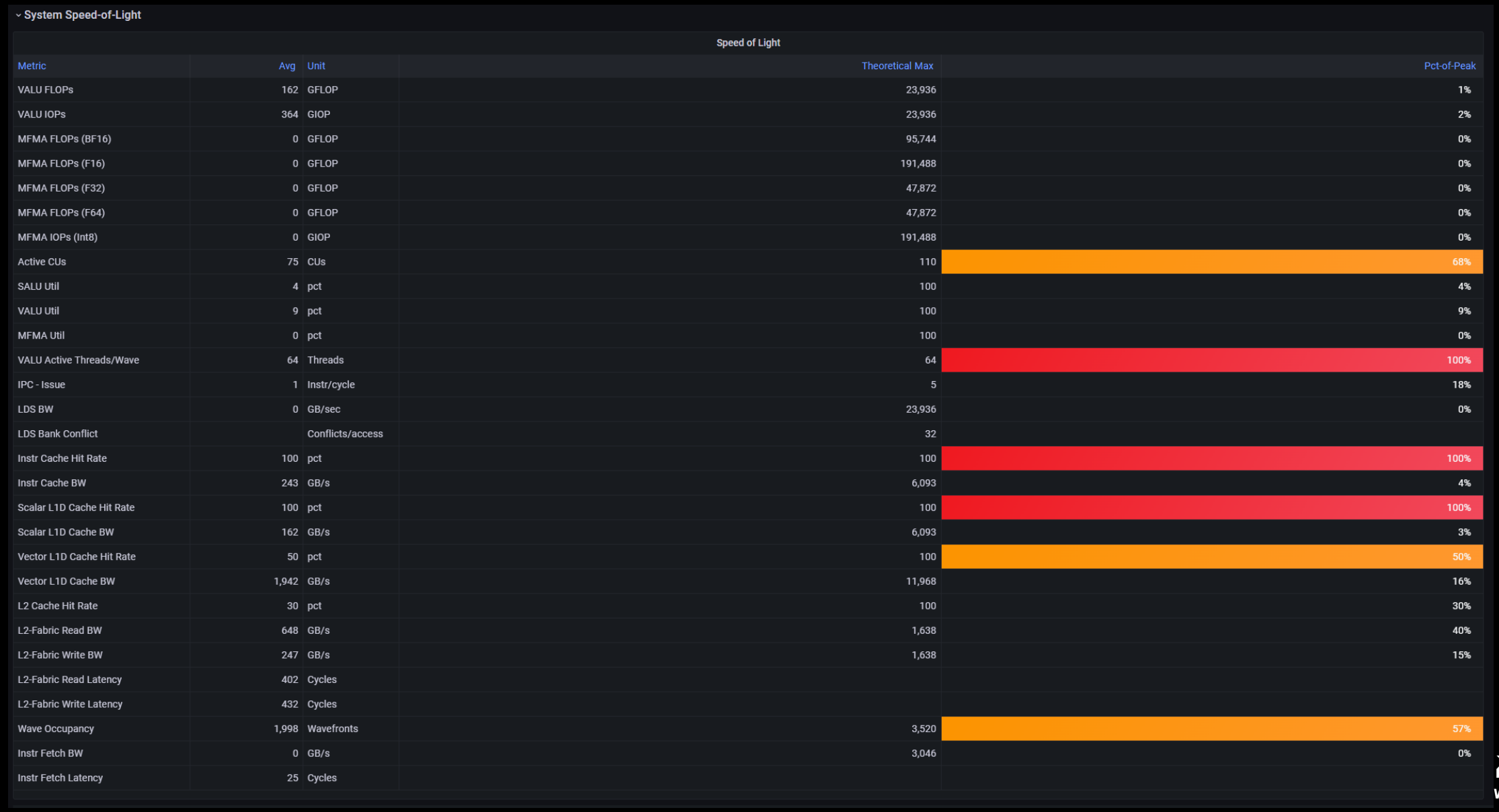
Grafana – System Info



System Info

Metric	Current	Baseline
Date	Tue Jul 5 20:50:45 2022 (UTC)	Tue Jun 21 18:31:40 2022 (CDT)
Host Name	6fb5ce5e50da	node-bp126-014a
Host CPU	AMD Eng Sample: 100-000000248-08_35/21_N	AMD Eng Sample: 100-000000248-08_35/21_N
Host Distro	Ubuntu 20.04.4 LTS	Ubuntu 20.04.4 LTS
Host Kernel	5.9.1-amdsos-build32-1+	5.9.1-amdsos-build32-1+
ROCm Version	5.1.3-66	5.2.0-9768
GFX SoC	mi200	mi200
GFX ID	gfx90a	gfx90a
Total SEs	8	8
Total SQCs	56	56
Total CUs	110	110
SIMDs/CU	4	4
Max Wavefronts Occupancy Per CU	32	32
Max Workgroup Size	1,024	1,024
L1Cache per CU (KB)	16	16
L2Cache (KB)	8,192	8,192
L2Cache Channels	32	32
Sys Clock (Max) - MHz	1,700	1,700
Memory Clock (Max) - MHz	1,600	1,600
Sys Clock (Cur) - MHz	800	800
Memory Clock (Cur) - MHz	1,600	1,600
HBM Bandwidth - GB/s	1,638.4	1,638.4

Grafana – System Speed-of-Light

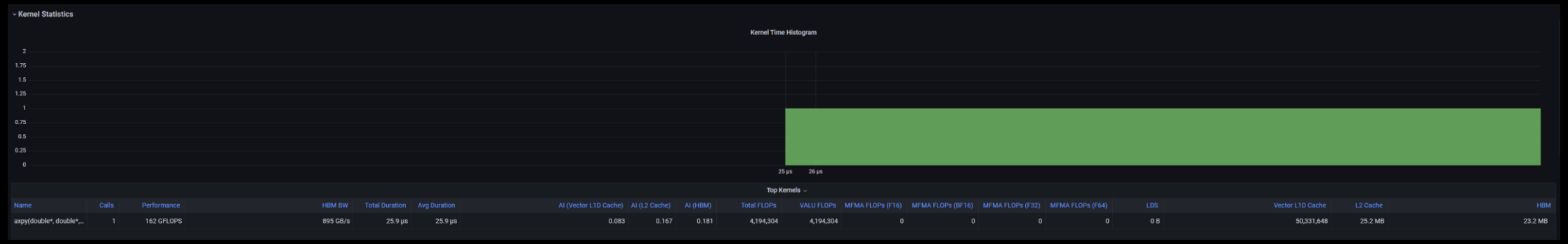


System Speed-of-Light

Speed of Light

Metric	Avg	Unit	Theoretical Max	Pct-of-Peak
VALU FLOPs	162	GFLOP	23,936	1%
VALU IOPs	364	GIOP	23,936	2%
MFMA FLOPs (BF16)	0	GFLOP	95,744	0%
MFMA FLOPs (F16)	0	GFLOP	191,488	0%
MFMA FLOPs (F32)	0	GFLOP	47,872	0%
MFMA FLOPs (F64)	0	GFLOP	47,872	0%
MFMA IOPs (Int8)	0	GIOP	191,488	0%
Active CUs	75	CUs	110	68%
SALU Util	4	pct	100	4%
VALU Util	9	pct	100	9%
MFMA Util	0	pct	100	0%
VALU Active Threads/Wave	64	Threads	64	100%
IPC - Issue	1	Instr/cycle	5	18%
LDS BW	0	GB/sec	23,936	0%
LDS Bank Conflict		Conflicts/access	32	
Instr Cache Hit Rate	100	pct	100	100%
Instr Cache BW	243	GB/s	6,093	4%
Scalar L1D Cache Hit Rate	100	pct	100	100%
Scalar L1D Cache BW	162	GB/s	6,093	3%
Vector L1D Cache Hit Rate	50	pct	100	50%
Vector L1D Cache BW	1,942	GB/s	11,968	16%
L2 Cache Hit Rate	30	pct	100	30%
L2-Fabric Read BW	648	GB/s	1,638	40%
L2-Fabric Write BW	247	GB/s	1,638	15%
L2-Fabric Read Latency	402	Cycles		
L2-Fabric Write Latency	432	Cycles		
Wave Occupancy	1,998	Wavefronts	3,520	57%
Instr Fetch BW	0	GB/s	3,046	0%
Instr Fetch Latency	25	Cycles		

Grafana- Kernel Statistics

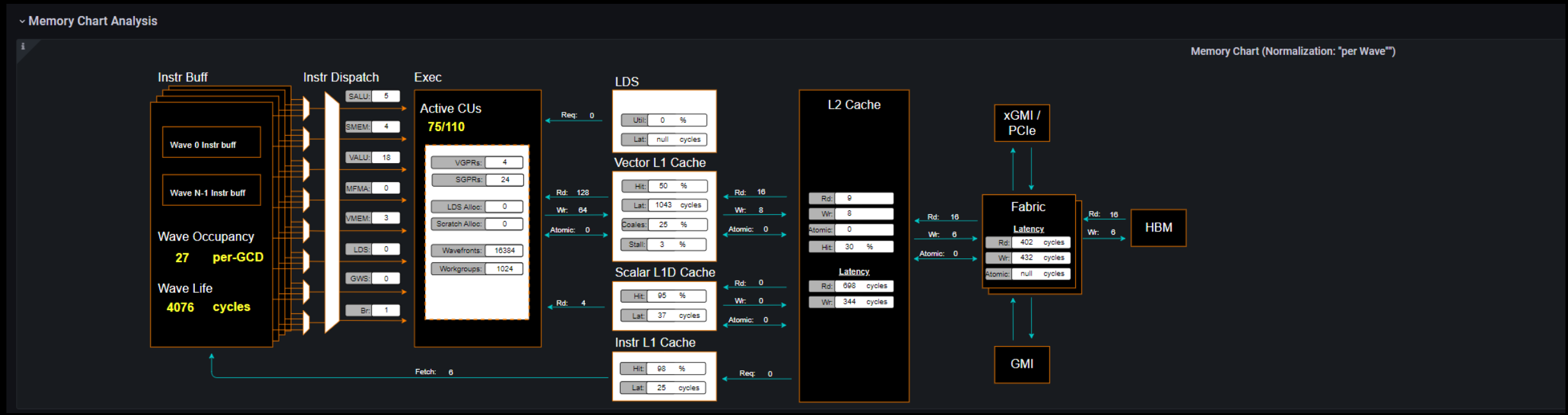


Name	Calls	Performance	HBM BW	Total Duration	Avg Duration	AI (Vector L1D Cache)	AI (L2 Cache)	AI (HBM)	Total FLOPs	VALU FLOPs	MFMA FLOPs (F16)	MFMA FLOPs (BF16)	MFMA FLOPs (F32)	MFMA FLOPs (F64)	LDS	Vector L1D Cache	L2 Cache	HBM
axpy(double*, double*,...	1	162 GFLOPS	895 GB/s	25.9 µs	25.9 µs	0.083	0.167	0.181	4,194,304	4,194,304	0	0	0	0	0 B	50,331,648	25.2 MB	23.2 MB

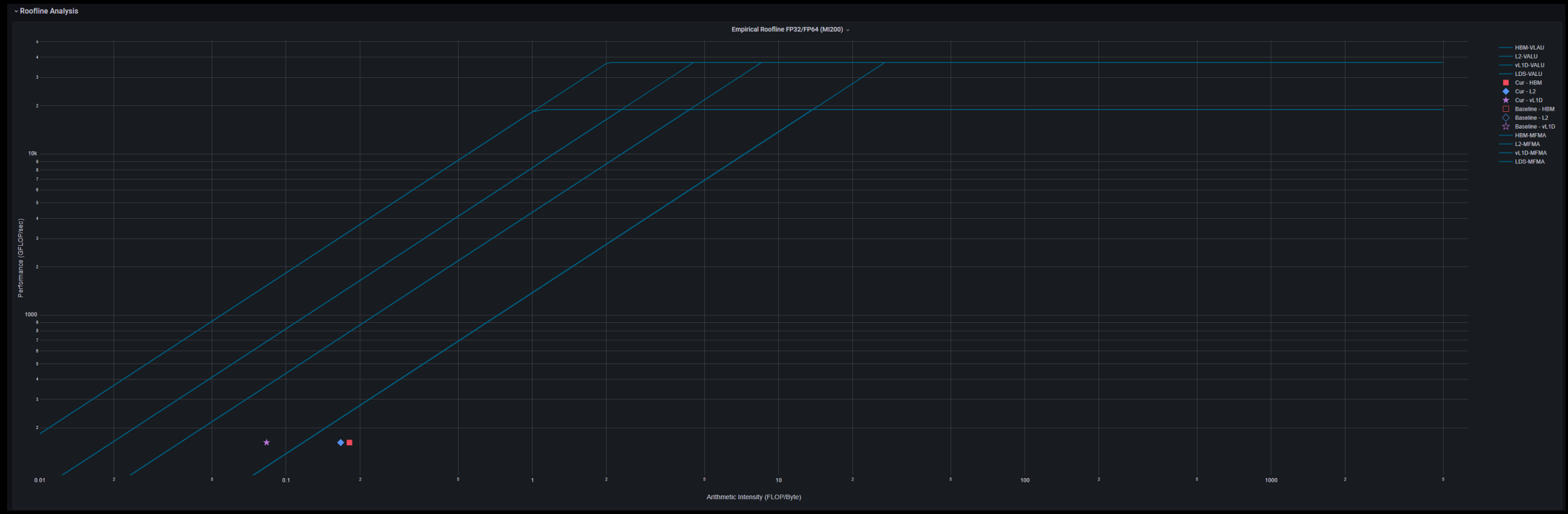
Top Dispatches

Dispatch	Calls	Performance	HBM BW	Total Duration	Avg Duration	AI (Vector L1D Cache)	AI (L2 Cache)	AI (HBM)	Total FLOPs	VALU FLOPs	MFMA FLOPs (F16)	MFMA FLOPs (BF16)	MFMA FLOPs (F32)	MFMA FLOPs (F64)	LDS	Vector L1D Cache	L2 Cache	HBM
0	1	162 GFLOPS	895 GB/s	25.9 µs	25.9 µs	0.083	0.167	0.181	4,194,304	4,194,304	0	0	0	0	0 B	50,331,648	25.2 MB	23.2 MB

Grafana – Mmeory Chart Analysis



Grafana - Roofline



Grafana – Wavefront & Compute Unit

Wavefront

Wavefront Launch Stats

Metric	Avg	Min	Max	Unit
Grid Size	1,048,576	1,048,576	1,048,576	Work Items
Workgroup Size	1,024	1,024	1,024	Work Items
Total Wavefronts	16,384	16,384	16,384	Wavefronts
Saved Wavefronts	0	0	0	Wavefronts
Restored Wavefronts	0	0	0	Wavefronts
VGPRs	4	4	4	Registers
SGPRs	24	24	24	Registers
LDS Allocation	0	0	0	Bytes
Scratch Allocation	0	0	0	Bytes

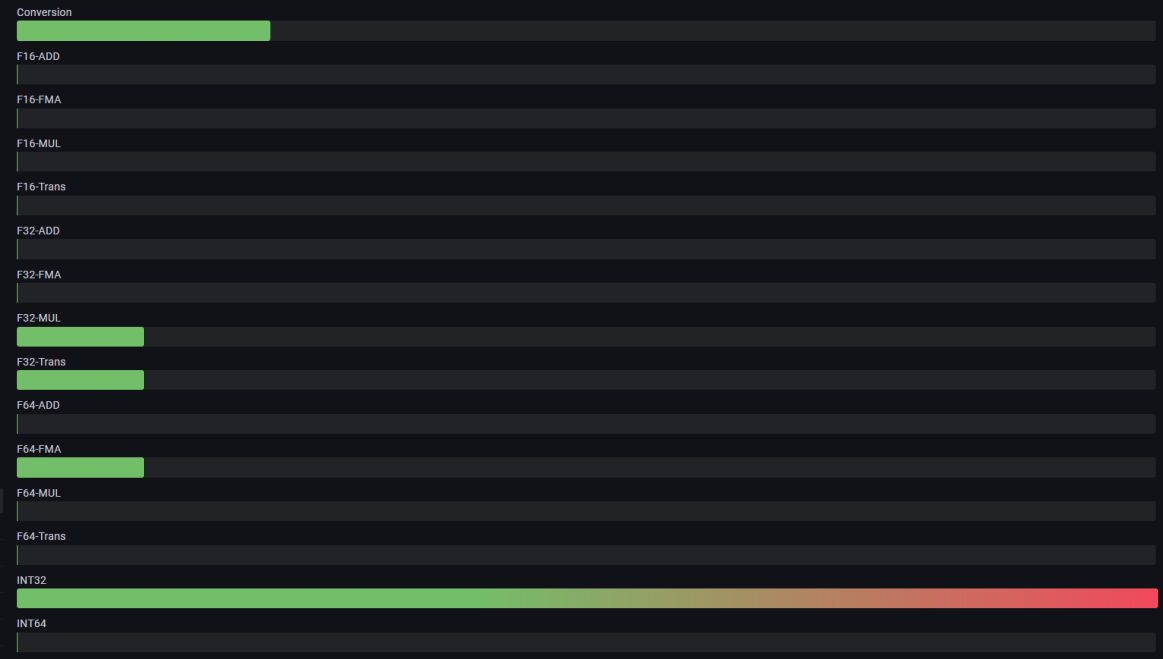
Wavefront Runtime Stats

Metric	Avg	Min	Max	Unit
Kernel Time (Nanosec)	25,920	25,920	25,920	ns
Kernel Time (Cycles)	34,367	34,367	34,367	Cycle
Instr/wavefront	36	36	36	Instr/wavefront
Wave Cycles	4,076	4,076	4,076	Cycles/wave
Dependency Wait Cycles	3,683	3,683	3,683	Cycles/wave
Issue Wait Cycles	780	780	780	Cycles/wave
Active Cycles	140	140	140	Cycles/wave
Wavefront Occupancy	1,998	1,998	1,998	Wavefronts

Compute Unit - Instruction Mix

Instruction Mix

VALU Arithmetic Instr Mix



VMEM Instr Mix

VMEM Instr	Count
Buffer Instr	0
Buffer Read	0
Buffer Write	0
Buffer Atomic	0
Flat Instr	3
Flat Read	2
Flat Write	1
Flat Atomic	0

MFMA Arithmetic Instr Mix

MFMA Instr	Count
MFMA-I8	0

Grafana – Instruction Cache & Scalar L1 Data Cache

Instruction Cache Accesses

Metric	Avg (Current)	Min (Current)	Max (Current)	Unit
Req	6	6	6	Req per Wave
Hits	6	6	6	Hits per Wave
Misses - Non Duplicated	0	0	0	Misses per Wave
Misses - Duplicated	0	0	0	Misses per Wave
Cache Hit	98	98	98	pct

Scalar L1D Cache - L2 Interface

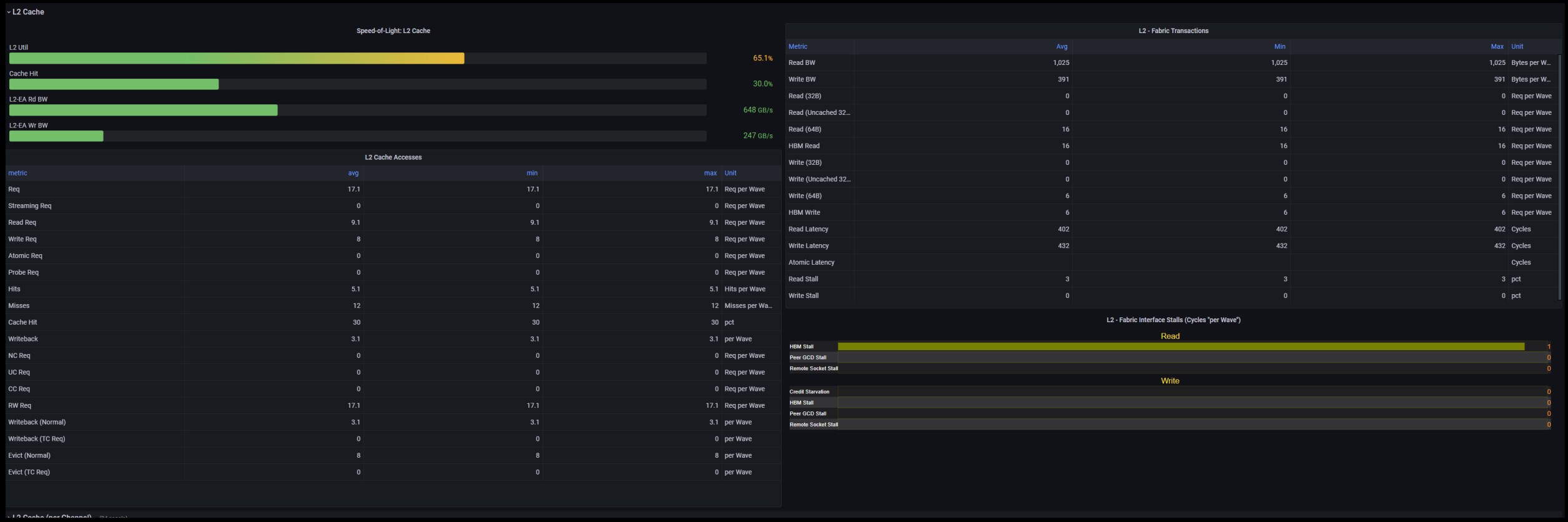
Metric	Mean	Min	Max	Unit
Read Req	0.007	0.007	0.007	Req per Wave
Write Req	0	0	0	Req per Wave
Atomic Req	0	0	0	Req per Wave
Stall	0	0	0	Cycles per Wave

Scalar L1D Cache Accesses

Metric	Avg (Current)	Min (Current)	Max (Current)	Unit
Req	4	4	4	Req per Wave
Hits	4	4	4	Req per Wave
Misses - Non Duplicated	0	0	0	Req per Wave
Misses- Duplicated	0	0	0	Req per Wave
Cache Hit	95	95	95	pct
Read Req (Total)	4	4	4	Req per Wave
Atomic Req	0	0	0	Req per Wave
Read Req (1 DWord)	2	2	2	Req per Wave
Read Req (2 DWord)	1	1	1	Req per Wave
Read Req (4 DWord)	1	1	1	Req per Wave
Read Req (8 DWord)	0	0	0	Req per Wave
Read Req (16 DWord)	0	0	0	Req per Wave

Grafana – Vector L1 Data Cache

Grafana – L2 Cache



Grafana – L2 Cache (per Channel)

DISCLAIMERS AND ATTRIBUTIONS

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.

OpenCL is a trademark of Apple Inc. used by permission by Khronos Group, Inc.
The OpenMP name and the OpenMP logo are registered trademarks of the OpenMP Architecture Review Board


AMD