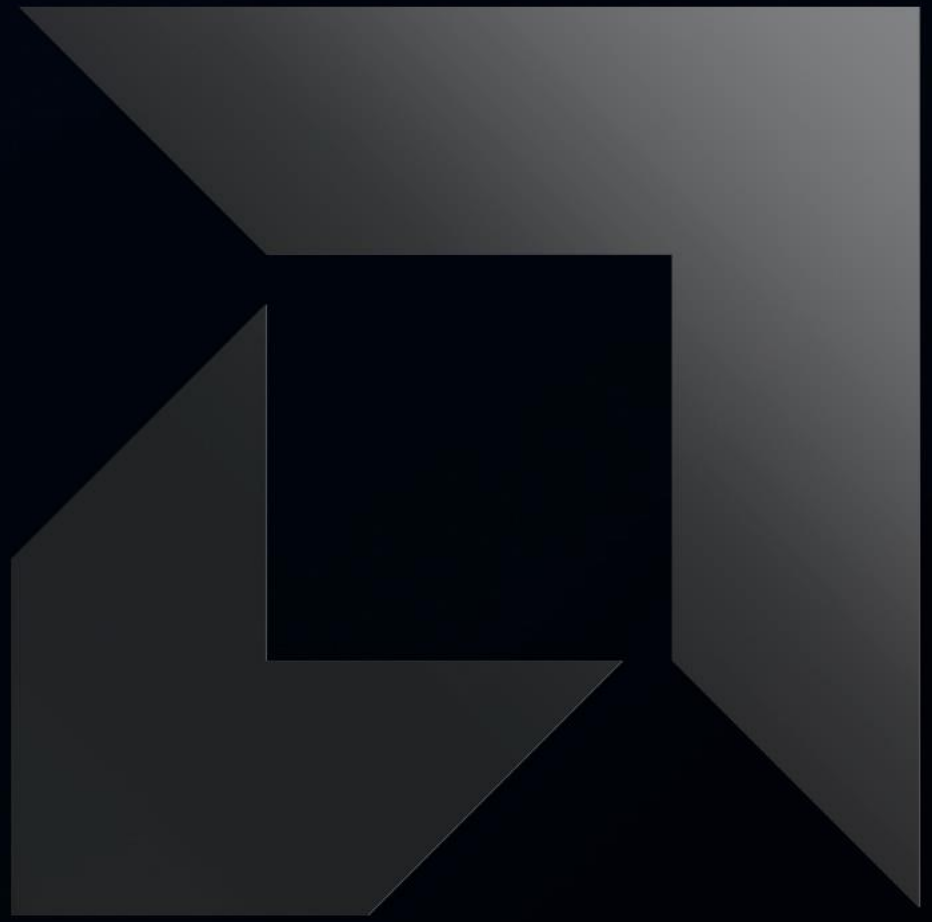


System Profiling with Omnitrace

Presenter: Sam Antao

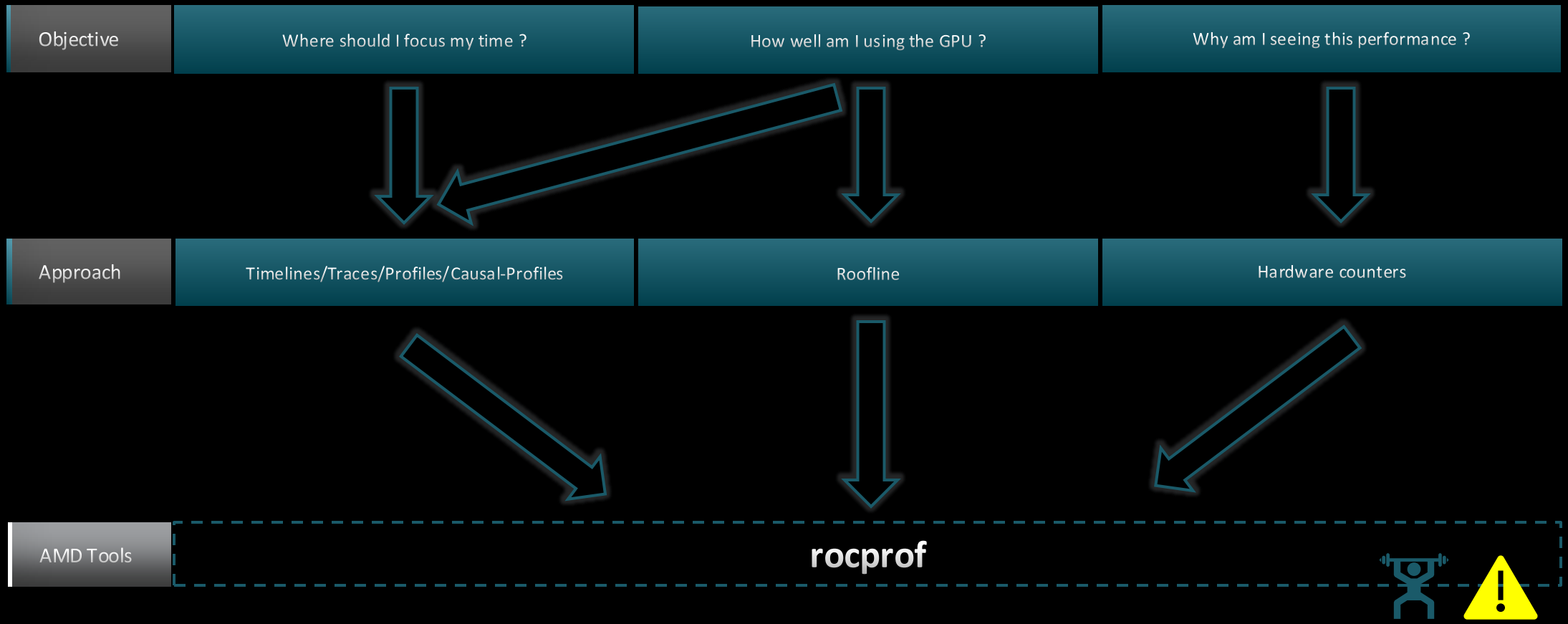
LUMI Comprehensive Course

Oct 31st, 2024

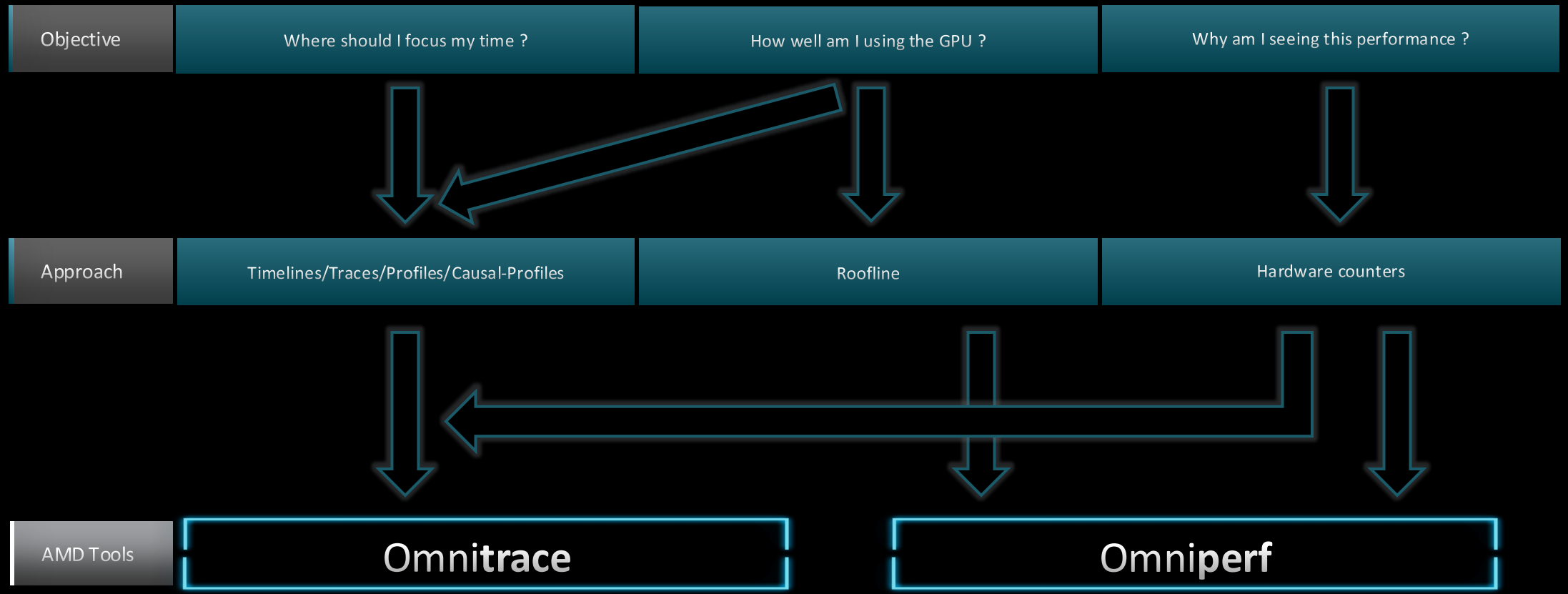
AMD

together we advance_

Background – AMD Profilers



Background – AMD Profilers



AMD Profilers with Timeline Profiling Support

ROC-profiler (rocprof)

Hardware Counters	Raw collection of GPU counters and traces	
	Counter collection with user input files	Counter results printed to a CSV
Traces and timelines	Trace collection support for	
	CPU to GPU copy	HIP API
	HSA API	GPU Kernels

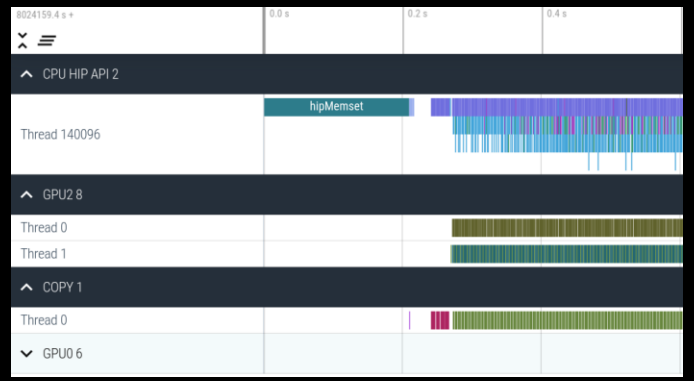
Omnitrace

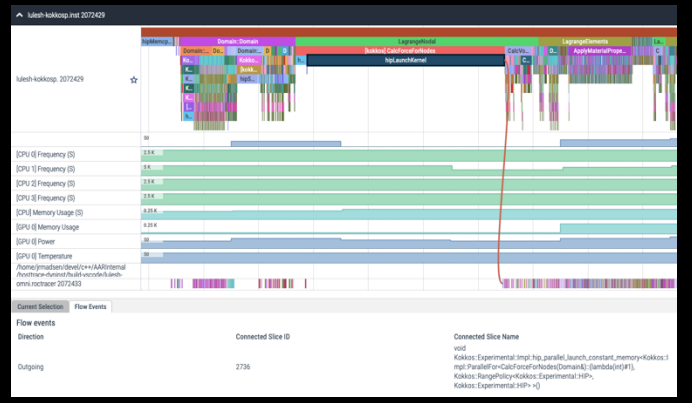
Trace collection	Hardware counters (rocprof)	Sampling
	Runtime instrumentation with standard executable	Binary rewrite Pre-instruments executable
Adds to rocprof	PAPI	DynInst
	Perf	Profile
	uProf	OpenMP®
	MPI	Kokkos
	p-threads	rocm-smi

Visualization

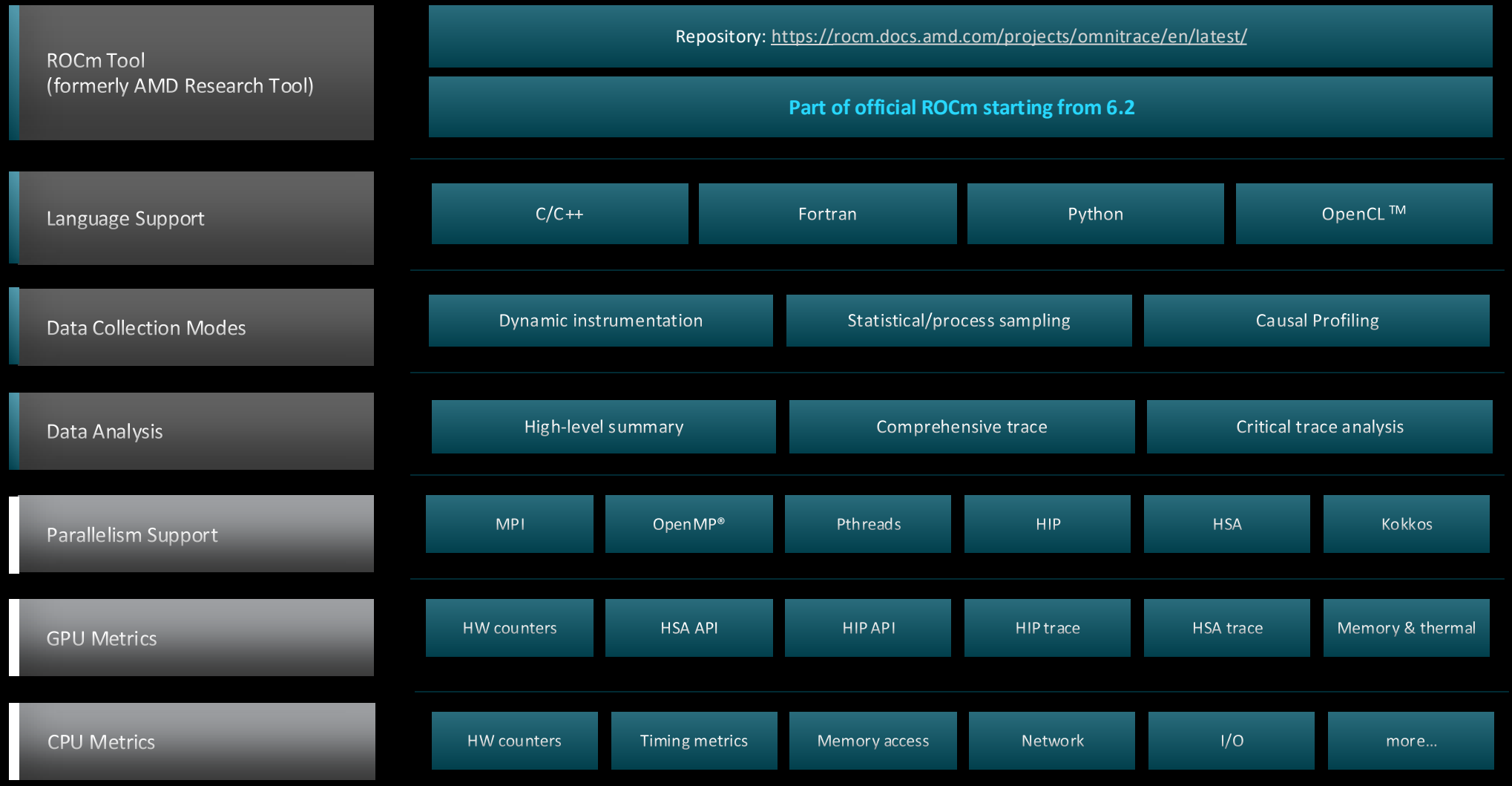
Traces visualized with Perfetto

	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





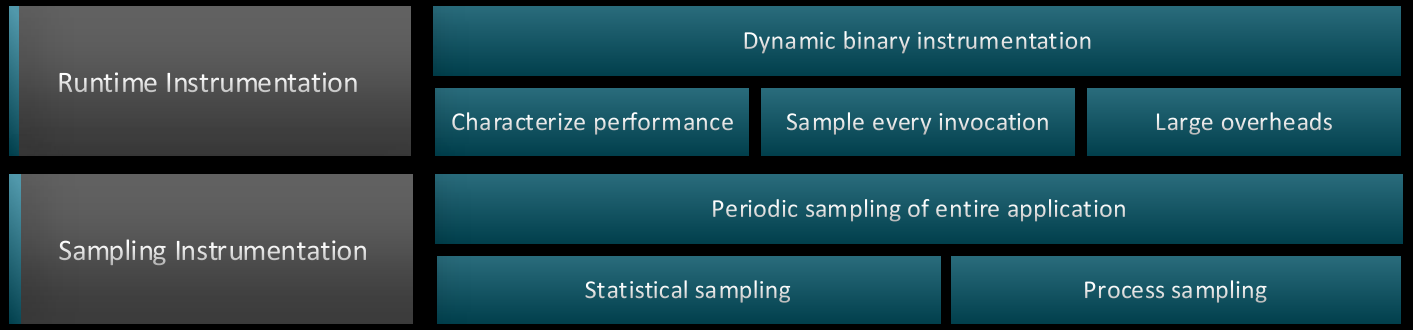
Omnitrace: Application Profiling, Tracing, and Analysis



New features constantly being added

Refer to current documentation for recent updates

Omnitrace functioning Modes



Basic command-line syntax:

```
$ omnitrace [omnitrace-options] -- <CMD> <ARGS>
```

For more information or help use -h/--help/? flags:

```
$ omnitrace -h
```

Can also execute on systems using a job scheduler. For example, with SLURM, an interactive session can be used as:

```
$ srun [options] omnitrace [omnitrace-options] -- <CMD> <ARGS>
```

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

Omnitrace Configuration Options

```
$ omnitrace-avail --categories [options]
```

Get more information about run-time settings, data collection capabilities, and available hardware counters. For more information or help use `-h/--help` flags:

```
$ omnitrace-avail -h
```

Collect information for Omnitrace-related settings using shorthand `-c` for `--categories`:

```
$ omnitrace-avail -c rocm
```

ENVIRONMENT VARIABLE	VALUE	CATEGORIES
OMNITRACE_ROCM_EVENTS		custom, hardware_counters, libomnitrace, omnitrace, rocm, rocprofiler
OMNITRACE_SAMPLING_GPUS	0	custom, libomnitrace, omnitrace, process_sampling, rocm, rocm_smi
OMNITRACE_USE_RCCLP	false	backend, custom, libomnitrace, omnitrace, rccl, rocm
OMNITRACE_USE_ROCM_SMI	true	backend, custom, libomnitrace, omnitrace, process_sampling, 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	true	backend, custom, libomnitrace, omnitrace, rocm, roctracer, roctx

Shows all runtime settings that may be tuned for rocm

Omnitrace Configuration File

```
$ omnitrace-avail --categories [options]
```

Get more information about run-time settings, data collection capabilities, and available hardware counters. For more information or help use `-h/--help` flags:

```
$ omnitrace-avail -h
```

Collect information for omnitrace-related settings using shorthand `-c` for `--categories`:

```
$ omnitrace-avail -c omnitrace
```

For brief description, use the options:

```
$ omnitrace-avail -bd
```

ENVIRONMENT VARIABLE	DESCRIPTION
OMNITRACE_CAUSAL_BINARY_EXCLUDE	Excludes binaries matching the list of provided regexes from causal experiments (separated by tab, sem...
OMNITRACE_CAUSAL_BINARY_SCOPE	Limits causal experiments to the binaries matching the provided list of regular expressions (separated...
OMNITRACE_CAUSAL_DELAY	Length of time to wait (in seconds) before starting the first causal experiment
OMNITRACE_CAUSAL_DURATION	Length of time to perform causal experimentation (in seconds) after the first experiment has started. ...
OMNITRACE_CAUSAL_FUNCTION_EXCLUDE	Excludes functions matching the list of provided regexes from causal experiments (separated by tab, se...
OMNITRACE_CAUSAL_FUNCTION_SCOPE	List of <function> regex entries for causal profiling (separated by tab, semi-colon, and/or quotes (si...
OMNITRACE_CAUSAL_RANDOM_SEED	Seed for random number generator which selects speedups and experiments -- please note that the lines ...
OMNITRACE_CAUSAL_SOURCE_EXCLUDE	Excludes source files or source file + lineno pair (i.e. <file> or <file>:<line>) matching the list of...
OMNITRACE_CAUSAL_SOURCE_SCOPE	Limits causal experiments to the source files or source file + lineno pair (i.e. <file> or <file>:<lin...
OMNITRACE_CONFIG_FILE	Configuration file for omnitrace
OMNITRACE_CRITICAL_TRACE	Enable generation of the critical trace
OMNITRACE_ENABLED	Activation state of timemory
OMNITRACE_OUTPUT_PATH	Explicitly specify the output folder for results
OMNITRACE_OUTPUT_PREFIX	Explicitly specify a prefix for all output files
OMNITRACE_PAPI_EVENTS	PAPI presets and events to collect (see also: papi_avail)
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...
OMNITRACE_PROCESS_SAMPLING_DURATION	If > 0.0, time (in seconds) to sample before stopping. If less than zero, uses OMNITRACE_SAMPLING_DURA...
OMNITRACE_PROCESS_SAMPLING_FREQ	Number of measurements per second when OMNITTRACE_USE_PROCESS_SAMPLING=ON. If set to zero, uses OMNITR...
OMNITRACE_ROCM_EVENTS	ROCm hardware counters. Use ':device=N' syntax to specify collection on device number N, e.g. ':device...
OMNITRACE_SAMPLING_CPUS	CPUs to collect frequency information for. Values should be separated by commas and can be explicit or...
OMNITRACE_SAMPLING_DELAY	Time (in seconds) to wait before the first sampling signal is delivered, increasing this value can fix...
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 expli...

Create a config file

Create a config file in `$HOME`:

```
$ omnitrace-avail -G $HOME/.omnitrace.cfg
```

To add description of all variables and settings, use:

```
$ omnitrace-avail -G $HOME/.omnitrace.cfg --all
```

Modify the config file `$HOME/.omnitrace.cfg` as desired to enable and change settings:

```
<snip>
OMNITRACE_TRACE                                 = true
OMNITRACE_PROFILE                               = true
OMNITRACE_USE_SAMPLING                          = false
OMNITRACE_USE_ROCTRACER                         = true
OMNITRACE_USE_ROCM_SMI                          = true
OMNITRACE_USE_MPIP                              = true
OMNITRACE_USE_PID                               = true
OMNITRACE_USE_ROCPROFILER                       = true
OMNITRACE_USE_ROCTX                             = true
<snip>
```

Contents of the config file

Declare which config file to use by setting the environment:

```
$ export OMNITRACE_CONFIG_FILE=/path-to/.omnitrace.cfg
```

Binary Rewrite

Binary Rewrite

```
$ omnitrace-instrument [omnitrace-options] -o <new-name-of-exec> -- <CMD> <ARGS>
```

Generating a new executable/library with instrumentation built-in:

```
$ omnitrace-instrument -o Jacobi_hip.inst -- ./Jacobi_hip
```

This new binary will have instrumented functions

Subroutine Instrumentation

Default instrumentation is main function and functions of 1024 instructions and more (for CPU)

To instrument routines with 50 or more cycles, add option "`-i 500`" (more overhead)

```
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libgcc_s-8-20210514.so.1'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libnss_compat-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libnss_dns-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libnss_files-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libpthread-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libresolv-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/librt-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libstdc++.so.6.0.25'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libthread_db-1.0.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libutil-2.28.so'...
[omnitrace][exe] [internal] parsing library: '/usr/lib64/libz.so.1.2.11'...
[omnitrace][exe] [internal] binary info processing required 0.666 sec and 110.500 MB
[omnitrace][exe] Processing 9 modules...
[omnitrace][exe] Processing 9 modules... Done (0.001 sec, 0.000 MB)
[omnitrace][exe] Found 'MPI_Init' in '/home/ssitaram/git/HPCTrainingExamples/HIP/jacobi/Jacobi_hip'. Enabling MPI support...
[omnitrace][exe] Finding instrumentation functions...
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/available.json'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/available.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/instrumented.json'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/instrumented.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/excluded.json'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/excluded.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/overlapping.json'... Done
[omnitrace][exe] Outputting 'omnitrace-Jacobi_hip.inst-output/2023-03-15_12.57/instrumentation/overlapping.txt'... Done
[omnitrace][exe]
[omnitrace][exe] The instrumented executable image is stored in '/home/ssitaram/git/HPCTrainingExamples/HIP/jacobi/Jacobi_hip.inst'
[omnitrace][exe] Getting linked libraries for /home/ssitaram/git/HPCTrainingExamples/HIP/jacobi/Jacobi_hip...
[omnitrace][exe] Consider instrumenting the relevant libraries...
[omnitrace][exe]
[omnitrace][exe]        /lib64/libgcc_s.so.1
[omnitrace][exe]        /lib64/libpthread.so.0
[omnitrace][exe]        /lib64/libm.so.6
[omnitrace][exe]        /lib64/librt.so.1
[omnitrace][exe]        /home/ssitaram/cp2k-hip/libs/install/openmpi/lib/libmpi.so.40
[omnitrace][exe]        /opt/rocm-5.4.3//lib/libroctx64.so.4
[omnitrace][exe]        /opt/rocm-5.4.3//lib/libroctracer64.so.4
[omnitrace][exe]        /opt/rocm-5.4.3/hip/lib/libamdhip64.so.5
[omnitrace][exe]        /lib64/libstdc++.so.6
[omnitrace][exe]        /lib64/libc.so.6
[omnitrace][exe]        /lib64/ld-linux-x86-64.so.2
```

Path to new instrumented binary

Run Instrumented Binary

Binary Rewrite

```
$ omnitrace-instrument [omnitrace-options] -o <new-name-of-exec> -- <CMD> <ARGS>
```

Generating a new executable/library with instrumentation built-in:

```
$ omnitrace-instrument -o Jacobi_hip.inst -- ./Jacobi_hip
```

Run the instrumented binary:

```
$ mpirun -np 1 omnitrace-run -- ./Jacobi_hip.inst -g 1 1
```

Subroutine Instrumentation

Default instrumentation is main function and functions of 1024 instructions and more (for CPU)

To instrument routines with 50 or more cycles, add option "`-i 50`" (more overhead)

Binary rewrite is recommended for runs with multiple ranks as Omnitrace produces separate output files for each rank

```
[omnitrace][3624331][omnitrace_init_tooling] Instrumentation mode: Trace

   omnitrace v1.8.0
[953.765]       perfetto.cc:58656 Configured tracing session 1, #sources:1, duration:0 ms, #buffers:1, total buffer size:1024000 KB, total sessions:1, uid:0 session name: ""
Topology size: 1 x 1
Local domain size (current node): 4096 x 4096
[omnitrace][0][pid=3624331] MPI rank: 0 (0), MPI size: 1 (1)
Global domain size (all nodes): 4096 x 4096
Rank 0 selecting device 0 on host TheraC60
Starting Jacobi run.
Iteration:   0 - Residual: 0.022108
Iteration: 100 - Residual: 0.000625
Iteration: 200 - Residual: 0.000371
Iteration: 300 - Residual: 0.000274
Iteration: 400 - Residual: 0.000221
Iteration: 500 - Residual: 0.000187
Iteration: 600 - Residual: 0.000163
Iteration: 700 - Residual: 0.000145
Iteration: 800 - Residual: 0.000131
Iteration: 900 - Residual: 0.000120
Iteration: 1000 - Residual: 0.000111
Stopped after 1000 iterations with residue 0.000111
Total Jacobi run time: 1.5470 sec.
Measured lattice updates: 10.84 GLU/s (total), 10.84 GLU/s (per process)
Measured FLOPS: 184.36 GFLOPS (total), 184.36 GFLOPS (per process)
Measured device bandwidth: 1.04 TB/s (total), 1.04 TB/s (per process)

[omnitrace][3624331][0][omnitrace_finalize] finalizing...
[omnitrace][3624331][0][omnitrace_finalize]
[omnitrace][3624331][0][omnitrace_finalize] omnitrace/process/3624331 : 2.364423 sec wall_clock,  645.964 MB peak_rss,  388.739 MB page_rss, 4.330000 sec cpu_clock,  183.1 % cpu_util [laps: 1]
[omnitrace][3624331][0][omnitrace_finalize] omnitrace/process/3624331/thread/0 : 2.355893 sec wall_clock, 1.293230 sec thread_cpu_clock,   54.9 % thread_cpu_util,  645.964 MB peak_rss [laps: 1]
[omnitrace][3624331][0][omnitrace_finalize] omnitrace/process/3624331/thread/1 : 2.345084 sec wall_clock, 0.000261 sec thread_cpu_clock,    0.0 % thread_cpu_util,  642.676 MB peak_rss [laps: 1]
[omnitrace][3624331][0][omnitrace_finalize]
[omnitrace][3624331][0][omnitrace_finalize] Finalizing perfetto...
```

Generates traces for application run

Kernel Durations

```
$ cat omnitrace-Jacobi_hip.inst-output/2024-01-01_13.57/wall_clock-0.txt
```

If you do not see a `wall_clock.txt` dumped by Omnitrace, try modify the config file `$HOME/.omnitrace.cfg` and enable `OMNITRACE_PROFILE` (or prepend to your `mpirun` command):

```
…
OMNITRACE_PROFILE                                   = true
…
```

Durations

Call Stack

	0>>>	_MPI_Allreduce	1	5	wall_clock	sec	0.000012	0.000012	0.000012	0.000012	0.000000	0.000000	100.0
	0>>>	_hipDeviceSynchronize	1	5	wall_clock	sec	0.000019	0.000019	0.000019	0.000019	0.000000	0.000000	94.4
	0>>>	_NormKernel1(int, double, double, double const*, double*)	1	6	wall_clock	sec	0.000001	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
	0>>>	_NormKernel2(int, double const*, double*)	1	6	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
	0>>>	_MPI_Barrier	1	5	wall_clock	sec	0.000001	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
	0>>>	_hipEventRecord	2	5	wall_clock	sec	0.000027	0.000014	0.000011	0.000016	0.000000	0.000003	100.0
	0>>>	_Halo D2H::Halo Exchange	1	5	wall_clock	sec	1.628420	1.628420	1.628420	1.628420	0.000000	0.000000	0.0
	0>>>	_hipStreamSynchronize	1	6	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
	0>>>	_MPI Exchange::Halo Exchange	1	6	wall_clock	sec	1.628395	1.628395	1.628395	1.628395	0.000000	0.000000	0.0
	0>>>	_MPI_Waitall	1	7	wall_clock	sec	0.000002	0.000002	0.000002	0.000002	0.000000	0.000000	100.0
	0>>>	_Halo H2D::Halo Exchange	1	7	wall_clock	sec	1.628104	1.628104	1.628104	1.628104	0.000000	0.000000	0.0
	0>>>	_hipStreamSynchronize	1	8	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
	0>>>	_hipLaunchKernel	5	8	wall_clock	sec	0.000615	0.000123	0.000005	0.000578	0.000000	0.000254	99.6
	0>>>	_mbind	1	9	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
	0>>>	_hipMemcpy	1	8	wall_clock	sec	0.001122	0.001122	0.001122	0.001122	0.000000	0.000000	99.9
	0>>>	_LocalLaplacianKernel(int, int, int, double, double, double const*, double*)	1	9	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
	0>>>	_HaloLaplacianKernel(int, int, int, double, double, double const*, double const*, double*)	1	9	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
	0>>>	_JacobiIterationKernel(int, double, double, double const*, double const*, double*, double*)	1	9	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0

Kernel Durations – Flat Profile

Edit in your `omnitrace.cfg` (or prepend to your `mpirun` command):

```
OMNITRACE_PROFILE                                = true
OMNITRACE_FLAT_PROFILE                           = true
```

Use flat profile to see aggregate duration of kernels and functions

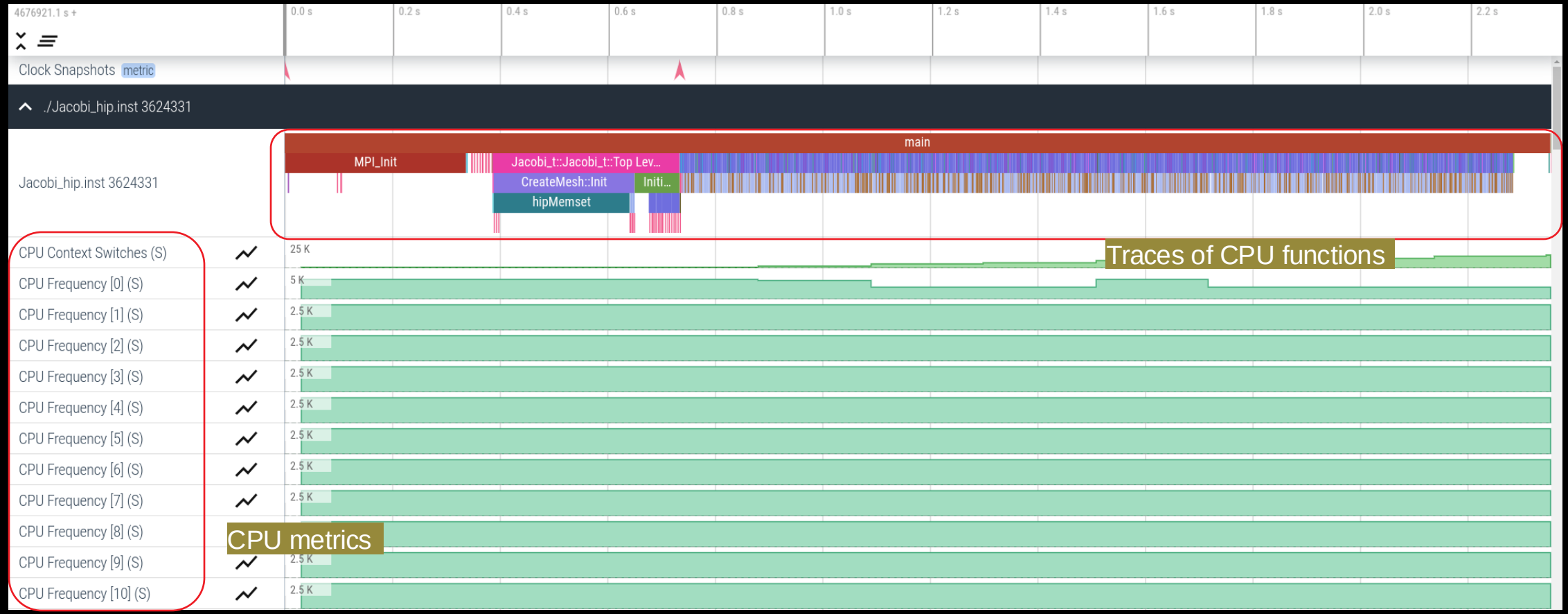
REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> main	1	0	wall_clock	sec	82.739099	82.739099	82.739099	82.739099	0.000000	0.000000	100.0
|0>>> MPI_Init	1	0	wall_clock	sec	34.056610	34.056610	34.056610	34.056610	0.000000	0.000000	100.0
|0>>> pthread_create	3	0	wall_clock	sec	0.014644	0.004881	0.001169	0.011974	0.000038	0.006145	100.0
|0>>> mbind	285	0	wall_clock	sec	0.001793	0.000006	0.000005	0.000020	0.000000	0.000002	100.0
|0>>> MPI_Comm_dup	1	0	wall_clock	sec	0.000212	0.000212	0.000212	0.000212	0.000000	0.000000	100.0
|0>>> MPI_Comm_rank	1	0	wall_clock	sec	0.000041	0.000041	0.000041	0.000041	0.000000	0.000000	100.0
|0>>> MPI_Comm_size	1	0	wall_clock	sec	0.000004	0.000004	0.000004	0.000004	0.000000	0.000000	100.0
|0>>> hipInit	1	0	wall_clock	sec	0.000372	0.000372	0.000372	0.000372	0.000000	0.000000	100.0
|0>>> hipGetDeviceCount	1	0	wall_clock	sec	0.000017	0.000017	0.000017	0.000017	0.000000	0.000000	100.0
|0>>> MPI_Allgather	1	0	wall_clock	sec	0.000009	0.000009	0.000009	0.000009	0.000000	0.000000	100.0
|0>>> hipSetDevice	1	0	wall_clock	sec	0.000024	0.000024	0.000024	0.000024	0.000000	0.000000	100.0
|0>>> hipHostMalloc	3	0	wall_clock	sec	0.126827	0.042276	0.000176	0.126453	0.005314	0.072900	100.0
|0>>> hipMalloc	7	0	wall_clock	sec	0.000458	0.000065	0.000024	0.000178	0.000000	0.000052	100.0
|0>>> hipMemset	1	0	wall_clock	sec	35.770403	35.770403	35.770403	35.770403	0.000000	0.000000	100.0
|0>>> hipStreamCreate	2	0	wall_clock	sec	0.016750	0.008375	0.005339	0.011412	0.000018	0.004295	100.0
|0>>> hipMemcpy	1005	0	wall_clock	sec	8.506781	0.008464	0.000610	0.039390	0.000023	0.004844	100.0
|0>>> hipEventCreate	2	0	wall_clock	sec	0.000037	0.000018	0.000016	0.000021	0.000000	0.000003	100.0
|0>>> hipLaunchKernel	5002	0	wall_clock	sec	0.181301	0.000036	0.000025	0.012046	0.000000	0.000278	100.0
|0>>> MPI_Allreduce	1003	0	wall_clock	sec	0.002009	0.000002	0.000001	0.000022	0.000000	0.000001	100.0
|0>>> hipDeviceSynchronize	1001	0	wall_clock	sec	0.016813	0.000017	0.000015	0.000043	0.000000	0.000004	100.0
|0>>> MPI_Barrier	3	0	wall_clock	sec	0.000007	0.000002	0.000001	0.000004	0.000000	0.000001	100.0
|0>>> hipEventRecord	2000	0	wall_clock	sec	0.046701	0.000023	0.000020	0.000225	0.000000	0.000006	100.0
|0>>> hipStreamSynchronize	2000	0	wall_clock	sec	0.030366	0.000015	0.000013	0.000382	0.000000	0.000009	100.0
|0>>> MPI_Waitall	1000	0	wall_clock	sec	0.001665	0.000002	0.000002	0.000007	0.000000	0.000000	100.0
|0>>> NormKernel1(int, double, double, double const*, double*)	1001	0	wall_clock	sec	0.001502	0.000002	0.000001	0.000006	0.000000	0.000000	100.0
|0>>> NormKernel2(int, double const*, double*)	1000	0	wall_clock	sec	0.001972	0.000002	0.000001	0.000003	0.000000	0.000001	100.0
|0>>> LocalLaplacianKernel(int, int, int, double, double, double const*, double*)	1000	0	wall_clock	sec	0.001488	0.000001	0.000001	0.000007	0.000000	0.000000	100.0
|0>>> HaloLaplacianKernel(int, int, int, double, double, double const*, double const*, double*)	1000	0	wall_clock	sec	0.001465	0.000001	0.000001	0.000007	0.000000	0.000000	100.0
|0>>> hipEventElapsedTime	1000	0	wall_clock	sec	0.015060	0.000015	0.000014	0.000041	0.000000	0.000002	100.0
|0>>> JacobiIterationKernel(int, double, double, double const*, double const*, double*, double*)	1000	0	wall_clock	sec	0.002598	0.000003	0.000001	0.000006	0.000000	0.000001	100.0
|0>>> pthread_join	1	0	wall_clock	sec	0.000396	0.000396	0.000396	0.000396	0.000000	0.000000	100.0
|0>>> hipFree	4	0	wall_clock	sec	0.000526	0.000131	0.000021	0.000243	0.000000	0.000091	100.0
|0>>> hipHostFree	2	0	wall_clock	sec	0.000637	0.000318	0.000287	0.000350	0.000000	0.000044	100.0
|3>>> start_thread	1	0	wall_clock	sec	0.004802	0.004802	0.004802	0.004802	0.000000	0.000000	100.0
|1>>> start_thread	1	0	wall_clock	sec	81.987779	81.987779	81.987779	81.987779	0.000000	0.000000	100.0
|2>>> start_thread	-	0	-	-	-	-	-	-	-	-	-

Visualizing Trace (1/3)

Use Perfetto

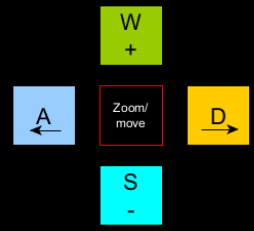
Copy `perfetto-trace-0.proto` to your laptop, go to https://ui.perfetto.dev/, click "Open trace file", select `perfetto-trace-0.proto`



Visualizing Trace (2/3)

Use Perfetto

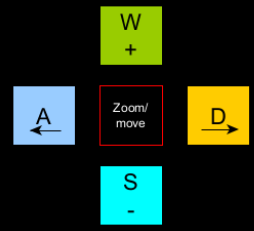
Zoom in to investigate regions of interest

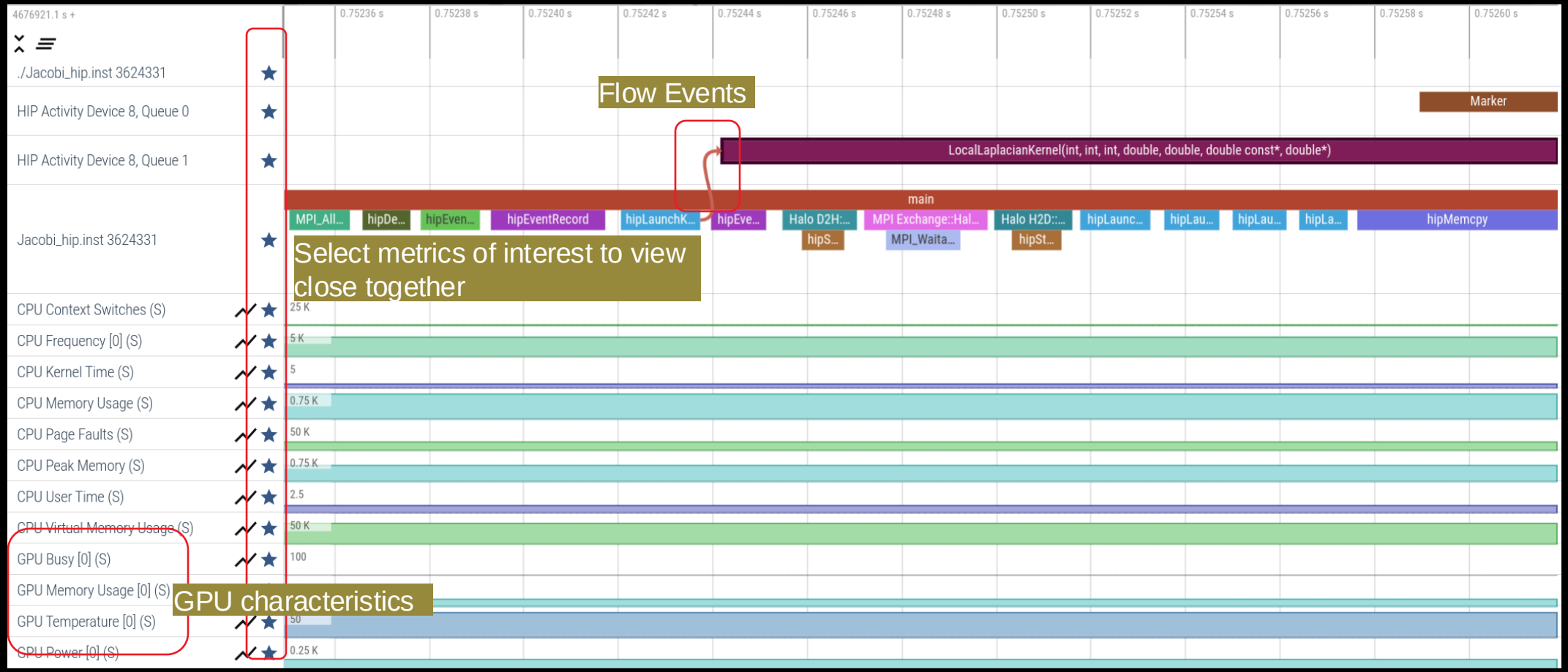


Visualizing Trace (3/3)

Use Perfetto

Zoom in to investigate regions of interest





Hardware Counters – List All

```
$ omnitrace-avail --all
```

Components, Categories

COMPONENT	AVAILABLE	VALUE_TYPE	STRING_IDS	FILENAME	DESCRIPTION	CATEGORY
allinea_map	false	void	"allinea", "allinea_map", "forge"		Controls the AllineaMAP sampler.	category::external, os::supports_linux, t...
caliper_marker	false	void	"cali", "caliper", "caliper_marker"		Generic forwarding of markers to Caliper ...	category::external, os::supports_unix, tp...
caliper_config	false	void	"caliper_config"		Caliper configuration manager.	category::external, os::supports_unix, tp...
caliper_loop_marker	false	void	"caliper_loop_marker"		Variant of caliper_marker with support fo...	category::external, os::supports_unix, tp...
cpu_clock	true	long	"cpu_clock"	cpu_clock	Total CPU time spent in both user- and ke...	project::timemory, category::timing, os::...
cpu_util	true	std::pair<long, long>	"cpu_util", "cpu_utilization"	cpu_util	Percentage of CPU-clock time divided by w...	project::timemory, category::timing, os::...
craypat counters	false	std::vector<unsigned long, std::allocato...	"craypat counters"	craypat counters	Names and value of any counter events tha...	category::external, os::supports_linux, t...

Environment Variables

ENVIRONMENT VARIABLE	VALUE	DATA TYPE	DESCRIPTION	CATEGORIES
OMNITRACE_CAUSAL_BINARY_EXCLUDE		string	Excludes binaries matching the list of pr...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_BINARY_SCOPE	%MAIN%	string	Limits causal experiments to the binaries...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_DELAY	0	double	Length of time to wait (in seconds) befor...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_DURATION	0	double	Length of time to perform causal experime...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_FUNCTION_EXCLUDE		string	Excludes functions matching the list of p...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_FUNCTION_SCOPE		string	List of <function> regex entries for caus...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_RANDOM_SEED	0	unsigned long	Seed for random number generator which se...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_SOURCE_EXCLUDE		string	Excludes source files or source file + li...	analysis, causal, custom, libomnitrace, o...
OMNITRACE_CAUSAL_SOURCE_SCOPE		string	Limits causal experiments to the source f...	analysis, causal, custom, libomnitrace, o...

HARDWARE COUNTER	AVAILABLE	DESCRIPTION
CPU		
PAPI_L1_DCM	true	Level 1 data cache misses
PAPI_L1_ICM	false	Level 1 instruction cache misses
PAPI_L2_DCM	true	Level 2 data cache misses
PAPI_L2_ICM	true	Level 2 instruction cache misses
PAPI_L3_DCM	false	Level 3 data cache misses
PAPI_L3_ICM	false	Level 3 instruction cache misses
PAPI_L1_TCM		Level 1 cache misses

CPU Hardware Counters

perf::CYCLES	true	PERF_COUNT_HW_CPU_CYCLES
perf::CYCLES:u=0	true	perf::CYCLES + monitor at user level
perf::CYCLES:k=0	true	perf::CYCLES + monitor at kernel level
perf::CYCLES:h=0	true	perf::CYCLES + monitor at hypervisor level
perf::CYCLES:period=0	true	perf::CYCLES + sampling period
perf::CYCLES:freq=0	true	perf::CYCLES + sampling frequency (Hz)
perf::CYCLES:precise=0	true	perf::CYCLES + precise event sampling
perf::CYCLES:excl=0	true	perf::CYCLES + exclusive access

TCC_NORMAL_WRITEBACK_sum:device=0	true	Number of writebacks due to requests that...
TCC_ALL_TC_OP_WB_WRITEBACK_sum:device=0	true	Number of writebacks due to all TC_OP wri...
TCC_NORMAL_EVICT_sum:device=0	true	Number of evictions due to requests that ...
TCC_ALL_TC_OP_INV_EVICT_sum:device=0	true	Number of evictions due to all TC_OP inva...
TCC_EA_RDREQ_DRAM_sum:device=0	true	Number of TCC/EA read requests (either 32...
TCC_EA_WRREQ_DRAM_sum:device=0	true	Number of TCC/EA write requests (either 3...
FETCH_SIZE:device=0	true	The total kilobytes fetched from the vide...
WRITE_SIZE:device=0	true	The total kilobytes written to the video ...
WRITE_REQ_32B:device=0	true	The total number of 32-byte effective mem...
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 instruct...
SALUInsts:device=0	true	The average number of scalar ALU instruct...
SFetchInsts:device=0	true	The average number of scalar fetch instru...
GDSInsts:device=0	true	The average number of GDS read or GDS wri...
MemUnitBusy:device=0	true	The percentage of GPUTime the memory unit...
ALUStalledByLDS:device=0	true	The percentage of GPUTime ALU units are s...

GPU Hardware Counters

A very small subset of the counters shown here

Configure Omnitrace to Collect GPU Hardware Counters

Modify config file

Modify the config file `$HOME/.omnitrace.cfg` to add desired metrics and for concerned `GPU#ID`:

```
…
OMNITRACE_ROCM_EVENTS = FetchSize:device=0, VALUUtilization:device=0, MemUnitBusy:device=0
…
```

To profile desired metrics for all participating GPUs:

```
…
OMNITRACE_ROCM_EVENTS = FetchSize, VALUUtilization, MemUnitBusy
…
```

Note: currently experiencing issues with ROCm 6.2.1, fix coming soon

Full list of GPU metrics at https://github.com/ROCm/rocprofiler/blob/amd-staging/test/tool/metrics.xml

Execution with Hardware Counters

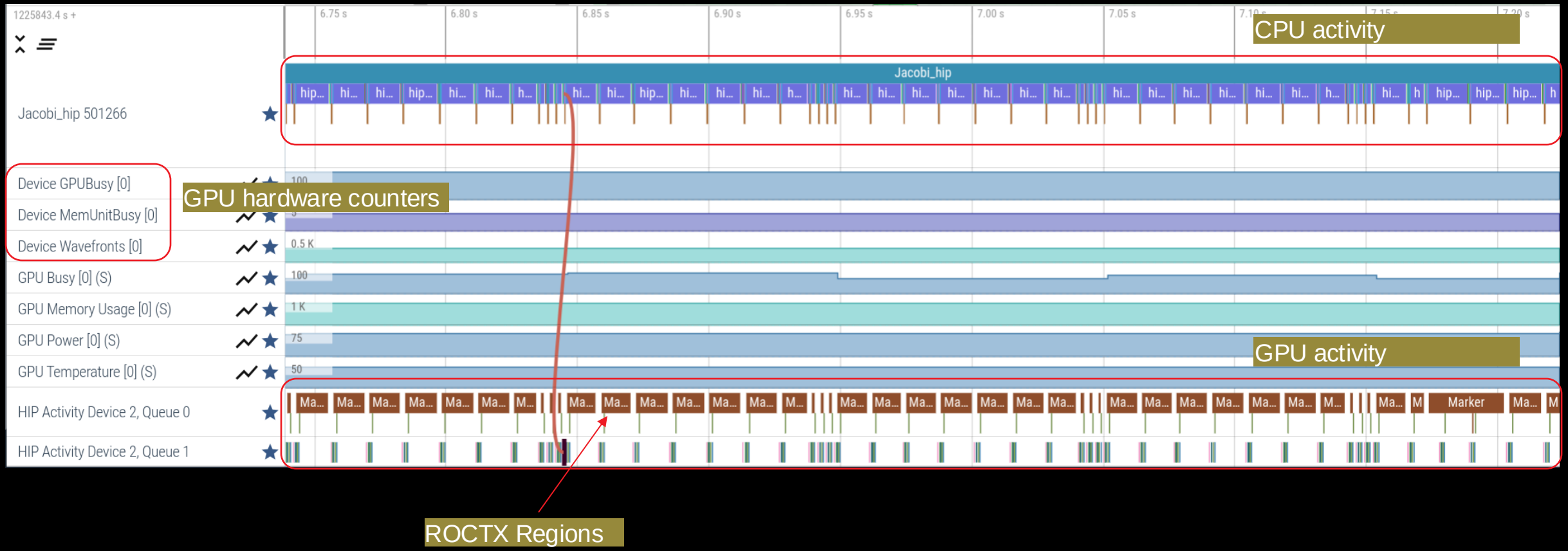
After modifying .cfg file to set up `OMNITRACE_ROCM_EVENTS` with GPU metrics run:

```
$ mpirun -np 1 omnitrace-run -- ./Jacobi_hip.inst -g 1 1
```

```
[omnitrace][1056814][0][omnitrace_finalize]
[omnitrace][1056814][0][omnitrace_finalize] Finalizing perfetto...
[omnitrace][1056814][perfetto]> Outputting '/datasets/teams/dcgpu_training/lstanisi/test_hackmd3/HPCTrainingExamples/HIP/jacobi/omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/perfetto-trace-0.proto' (8130.87 KB / 8.13 MB / 0.01 GB)... Done
[omnitrace][1056814][rocprof-device-0-FetchSize]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/rocprof-device-0-FetchSize-0.json'
[omnitrace][1056814][rocprof-device-0-FetchSize]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/rocprof-device-0-FetchSize-0.txt'
[omnitrace][1056814][rocprof-device-0-VALUUtilization]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/rocprof-device-0-VALUUtilization-0.json'
[omnitrace][1056814][rocprof-device-0-VALUUtilization]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/rocprof-device-0-VALUUtilization-0.txt'
[omnitrace][1056814][rocprof-device-0-MemUnitBusy]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/rocprof-device-0-MemUnitBusy-0.json'
[omnitrace][1056814][rocprof-device-0-MemUnitBusy]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/rocprof-device-0-MemUnitBusy-0.txt'
[omnitrace][1056814][wall_clock]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/wall_clock-0.json'
[omnitrace][1056814][wall_clock]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/wall_clock-0.txt'
[omnitrace][1056814][roctracer]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/roctracer-0.json'
[omnitrace][1056814][roctracer]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/roctracer-0.txt'
[omnitrace][1056814][metadata]> Outputting 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/metadata-0.json' and 'omnitrace-Jacobi_hip.inst-output/2024-10-02_10.36/functions-0.json'
```

GPU hardware counters

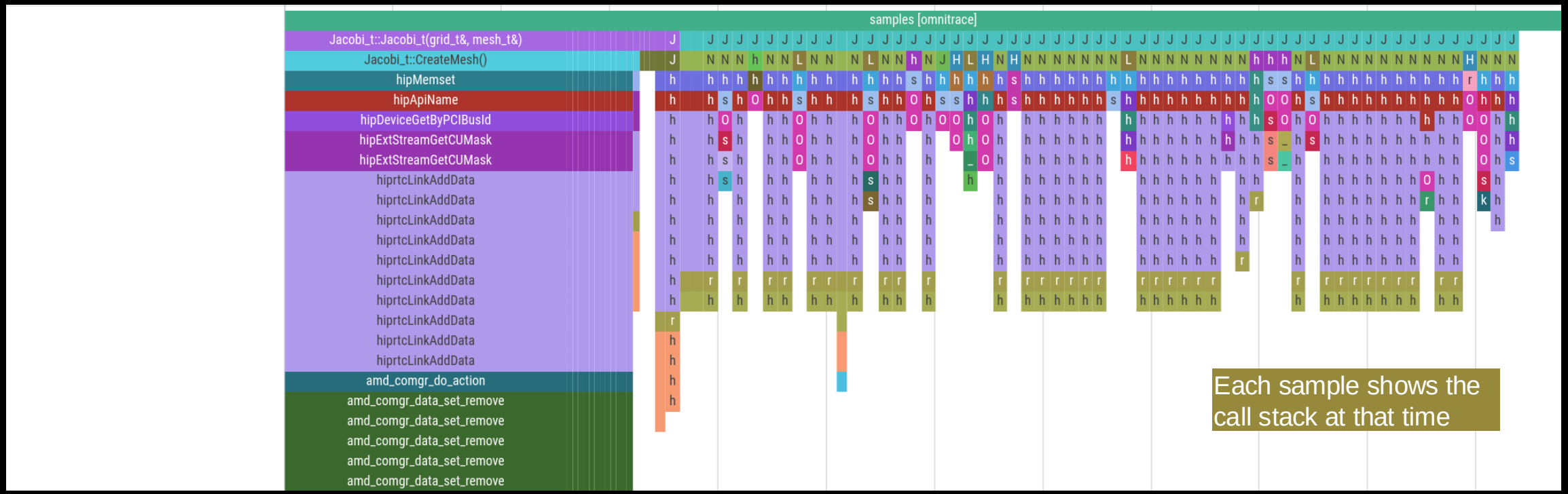
Visualization with Hardware Counters



Sampling CPU Call-Stack (1/2)

`OMNITRACE_USE_SAMPLING = true; OMNITRACE_SAMPLING_FREQ = 100` (100 samples per second)

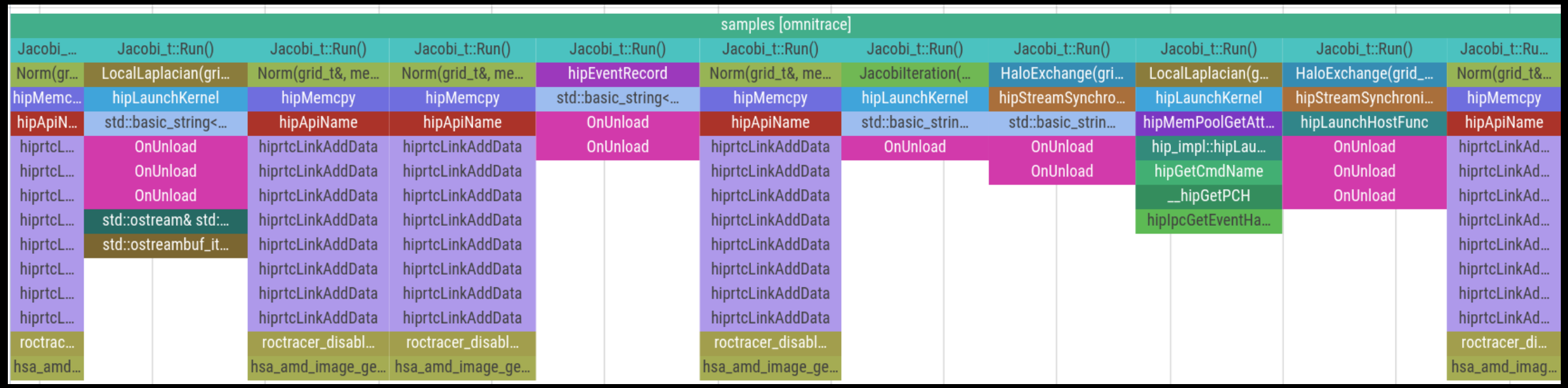
Alternatively run with `omnitrace-sample`



Scroll down all the way in Perfetto to see the sampling output

Sampling CPU Call-Stack (2/2)

Zoom in call-stack sampling



Thread 0 (S) 3625610 ← Sampling data is annotated with (S)

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
`int` **`omnitrace_user_push_region(void)`**	Start user defined region
`int` **`omnitrace_user_pop_region(void)`**	End user defined region, FILO (first in last out) is expected

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

OpenMP®

We use the example omnitrace/examples/openmp/

Build the code with CMake:

```
$ cmake -B build
```

Use the openmp-lu binary, which can be executed with:

```
$ export OMP_NUM_THREADS=4
$ srun -n 1 -c 4 ./openmp-lu
```

Create a new instrumented binary:

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

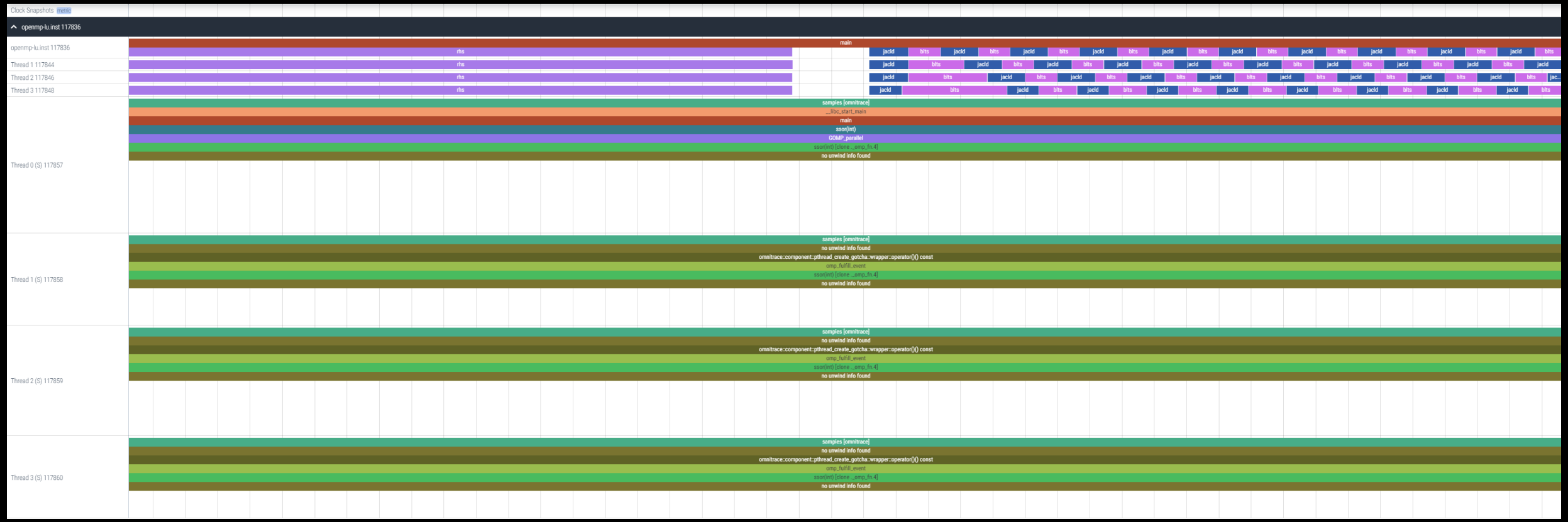
Execute the new binary:

```
$ srun -n 1 -c 4 omnitrace-run -- ./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 in omnitrace/examples/python/source.py

Execute the python program with:

```
$ omnitrace-python ./external.py
```

Profiled data is dumped in output directory:

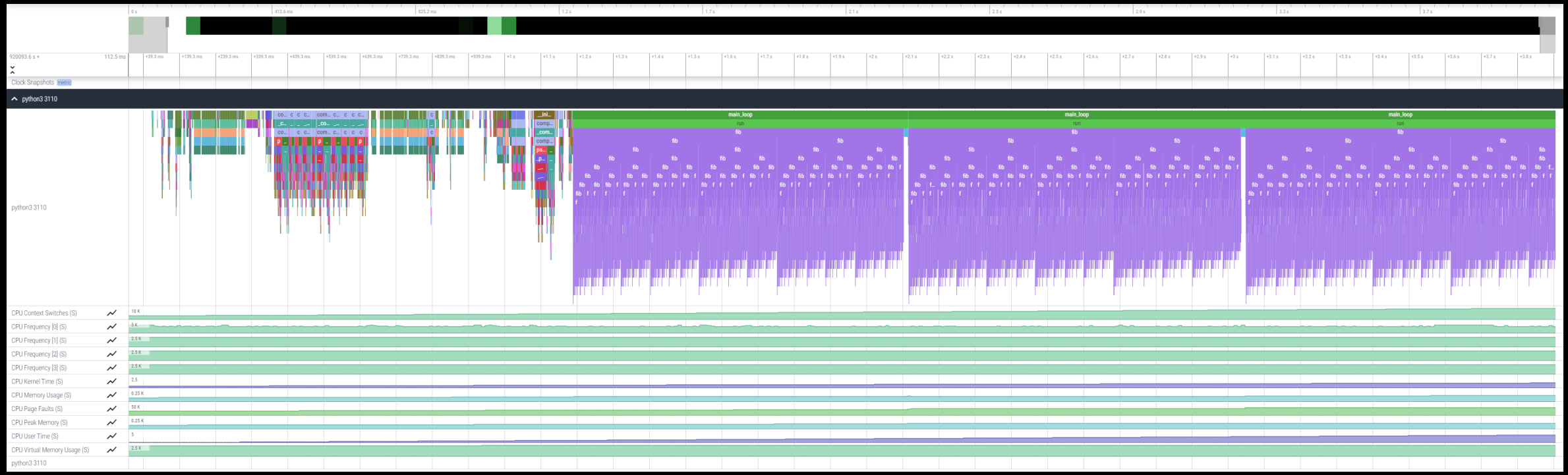
```
$ cat omnitrace-source-output/timestamp/wall_clock.txt
```

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

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™ Perfetto Tracing



Summary

- Omnitrace - powerful tool to understand CPU + GPU activity on AMD GPUs
  - Ideal for an initial look at how an application runs
  - Easy to visualize traces in Perfetto

- Leverages several other tools and combines their data into a comprehensive output files
  - Some tools used are AMDµProf, rocprofiler, rocm-smi, roctracer, perf, etc.

- **Helps users analyze overlaps between CPU/GPU compute and communication**

Hands-on exercises

https://hackmd.io/@sfantao/lumi-training-ams-2024#Omnitrace

We welcome you to explore our HPC Training Examples repo:

https://github.com/amd/HPCTrainingExamples

A table of contents for the READMEs if available at the top-level **README** in the repo

Relevant exercises for this presentation located in **Omnitrace** directory.

Link to instructions on how to run the tests: **Omnitrace/README.md and subdirectories**

DISCLAIMERS AND ATTRIBUTIONS


AMD