Tracing and Profiling of GPU-Accelerated Software

Progress Report Meeting
May 5, 2017

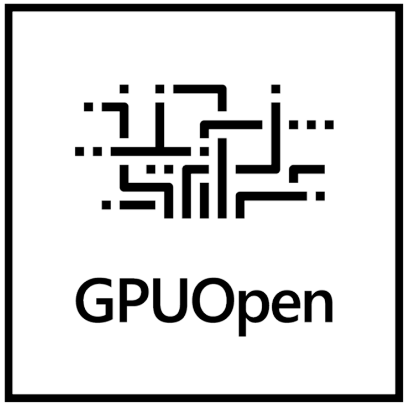
Paul Margheritta Michel Dagenais

DORSAL lab
École Polytechnique de Montréal

Introduction

- **GPU**: for graphics and general purpose (GPGPU)
- So many **cores**! We lack tools for that.
- Example: Radeon R9 Nano from AMD, **4096** cores
- We need to address issues related to **GPU specifics** and **highly parallel systems**

Software context



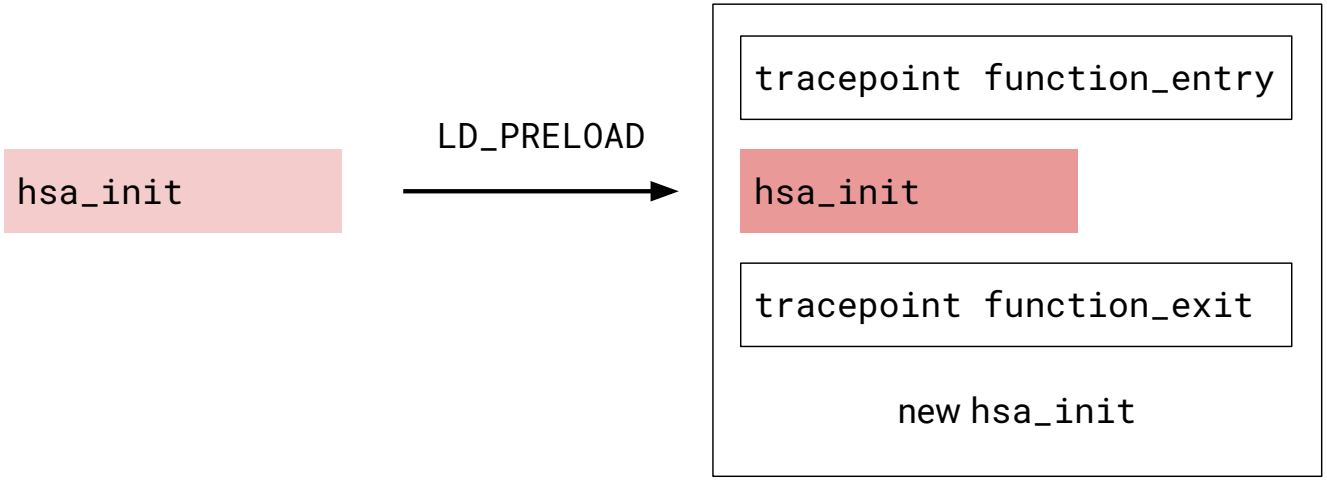
CODE XL

- **ROCm** (Radeon Open Compute): open-source platform for GPU development
- **HSA** (Heterogeneous System Architecture): runtime and API used to launch compute kernels
- **CodeXL**: open-source debugging and performance analysis tool for HSA and OpenCL

Research goals

- Analyze current **tracing and profiling mechanisms**
- Explore **AMD initiatives** for performance analysis on GPUs
- Provide **tracing** with LTTng in the HSA runtime
- Design **views** in Trace Compass for better understanding

Common techniques



- **Intercepting** and replacing symbols in the HSA runtime
- Early solution: changing links in the **API function table**
- More flexibility with **preloaded libraries**: build a collection of libraries that intercept API calls and other functions and preload them with `LD_PRELOAD`

Call stack events

- All **API functions** instrumented at entry and exit
- Generation of interception sources **automated** with Python scripts

```
[14:09:46.995847859] (+0.000000212) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_agent_iterate_regions" }
[14:09:46.995848105] (+0.000000246) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_allocate" }
[14:09:46.995873645] (+0.000025540) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_allocate" }
[14:09:46.995876201] (+0.000002556) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_queue_load_write_index_relaxed" }
[14:09:46.995877499] (+0.000001298) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_queue_load_write_index_relaxed" }
[14:09:46.995878536] (+0.000001037) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_queue_store_write_index_relaxed" }
[14:09:46.995879172] (+0.000000636) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_queue_store_write_index_relaxed" }
[14:09:46.995880363] (+0.000001191) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_signal_store_relaxed" }
[14:09:46.995881433] (+0.000001070) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_signal_store_relaxed" }
[14:09:46.998467990] (+0.002586557) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_signal_destroy" }
[14:09:46.998497508] (+0.000029518) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_signal_destroy" }
[14:09:46.998498496] (+0.000000988) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_free" }
[14:09:46.998629735] (+0.000131239) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_free" }
[14:09:46.998630386] (+0.000000651) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_free" }
[14:09:46.998757302] (+0.000126916) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_free" }
[14:09:46.998757819] (+0.000000517) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_free" }
[14:09:46.998769716] (+0.000011897) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_memory_free" }
[14:09:46.998772760] (+0.000003044) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_executable_destroy" }
[14:09:46.998804488] (+0.000031728) paul-gpu hsa_runtime:function_exit: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_executable_destroy" }
[14:09:46.998805579] (+0.000001091) paul-gpu hsa_runtime:function_entry: { cpu_id = 2 }, { vtid = 9639 }, { name = "hsa_code_object_destroy" }
```

Queue profiling events

- **User-mode queues** are used to dispatch functions to be executed on the GPU
- Gives the **state** of the user-mode queues
- Information about the **AQL packets** sent to the queues is also available

```
[14:10:52.461396161] (+?.??????????) paul-gpu hsa_runtime:function_entry: { cpu_id = 0 }, { vtid = 9699 }, { name = "hsa_init" }
[14:10:52.501221557] (+0.039825396) paul-gpu hsa_runtime:queue_created: { cpu_id = 2 }, { vtid = 9707 }, { agent_handle = 0x202ACA0, queue_id = 3 }
[14:10:52.501288925] (+0.000067368) paul-gpu hsa_runtime:queue_created: { cpu_id = 0 }, { vtid = 9706 }, { agent_handle = 0x202ACA0, queue_id = 1 }
[14:10:52.501530608] (+0.000241683) paul-gpu hsa_runtime:queue_created: { cpu_id = 4 }, { vtid = 9708 }, { agent_handle = 0x202ACA0, queue_id = 2 }
[14:10:52.502734250] (+0.001203642) paul-gpu hsa_runtime:queue_created: { cpu_id = 4 }, { vtid = 9705 }, { agent_handle = 0x202ACA0, queue_id = 4 }
[14:10:52.518108672] (+0.015374422) paul-gpu hsa_runtime:aql_kernel_dispatch_packet_submitted: { cpu_id = 7 }, { vtid = 9709 }, { packet_id = 0, agent_handle = 0x202ACA0, queue_id = 3, kernel_object = 0x9082C100
0, kernel_name = "&__vector_copy_kernel" }
[14:10:52.518155840] (+0.000047168) paul-gpu hsa_runtime:aql_kernel_dispatch_packet_submitted: { cpu_id = 1 }, { vtid = 9712 }, { packet_id = 0, agent_handle = 0x202ACA0, queue_id = 4, kernel_object = 0x9082C200
0, kernel_name = "&__vector_copy_kernel" }
[14:10:52.518693246] (+0.000537406) paul-gpu hsa_runtime:aql_kernel_dispatch_packet_submitted: { cpu_id = 2 }, { vtid = 9711 }, { packet_id = 0, agent_handle = 0x202ACA0, queue_id = 2, kernel_object = 0x9082C300
0, kernel_name = "&__vector_copy_kernel" }
[14:10:52.518741624] (+0.000048378) paul-gpu hsa_runtime:aql_kernel_dispatch_packet_submitted: { cpu_id = 6 }, { vtid = 9710 }, { packet_id = 0, agent_handle = 0x202ACA0, queue_id = 1, kernel_object = 0x9082C400
0, kernel_name = "&__vector_copy_kernel" }
[14:10:52.525311682] (+0.006570058) paul-gpu hsa_runtime:queue_destroyed: { cpu_id = 1 }, { vtid = 9707 }, { queue_id = 3 }
[14:10:52.525721880] (+0.000410198) paul-gpu hsa_runtime:queue_destroyed: { cpu_id = 4 }, { vtid = 9706 }, { queue_id = 1 }
[14:10:52.526094019] (+0.000372139) paul-gpu hsa_runtime:queue_destroyed: { cpu_id = 0 }, { vtid = 9705 }, { queue_id = 4 }
[14:10:52.526648647] (+0.000554628) paul-gpu hsa_runtime:queue_destroyed: { cpu_id = 6 }, { vtid = 9708 }, { queue_id = 2 }
[14:10:52.527073072] (+0.000424425) paul-gpu hsa_runtime:function_exit: { cpu_id = 1 }, { vtid = 9699 }, { name = "hsa_shut_down" }
```

Kernels timing

- A **profiled queue** allows us to get timing information
- **Start/end timestamps** are aligned on the monotonic clock of the system
- This information is obtained in an **asynchronous** way

```
[14:11:28.944146571] (+?.?????????) paul-gpu hsa_runtime:kernel_start_nm: { cpu_id = 7 }, { vtid = 9774 }, { kernel_object = 0x9076C2000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, que
ue_id = 3, timestamp = 17806973 }
[14:11:28.944154047] (+0.000007476) paul-gpu hsa_runtime:kernel_end_nm: { cpu_id = 7 }, { vtid = 9774 }, { kernel_object = 0x9076C2000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, queue
_id = 3, timestamp = 18909375 }
[14:11:28.944155089] (+0.000001042) paul-gpu hsa_runtime:kernel_start_nm: { cpu_id = 7 }, { vtid = 9774 }, { kernel_object = 0x9076C1000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, que
ue_id = 2, timestamp = 18053853 }
[14:11:28.944155430] (+0.000000341) paul-gpu hsa_runtime:kernel_end_nm: { cpu_id = 7 }, { vtid = 9774 }, { kernel_object = 0x9076C1000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, queue
_id = 2, timestamp = 19037855 }
[14:11:28.945675306] (+0.001519876) paul-gpu hsa_runtime:kernel_start_nm: { cpu_id = 5 }, { vtid = 9773 }, { kernel_object = 0x9076C3000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, que
ue_id = 1, timestamp = 21805361 }
[14:11:28.945688051] (+0.000012745) paul-gpu hsa_runtime:kernel_end_nm: { cpu_id = 5 }, { vtid = 9773 }, { kernel_object = 0x9076C3000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, queue
_id = 1, timestamp = 22329208 }
[14:11:28.947846320] (+0.002158269) paul-gpu hsa_runtime:kernel_start_nm: { cpu_id = 4 }, { vtid = 9772 }, { kernel_object = 0x9076C4000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, que
ue_id = 4, timestamp = 24535137 }
[14:11:28.947852196] (+0.000005876) paul-gpu hsa_runtime:kernel_end_nm: { cpu_id = 4 }, { vtid = 9772 }, { kernel_object = 0x9076C4000, kernel_name = "&__vector_copy_kernel", agent_handle = 0x7F792B1CEDC0, queue
_id = 4, timestamp = 25095618 }
```

Performance counters

- Performance counters provide **low-level**, hardware-related data
- The **SoftCP mode** is used to define pre- and post-dispatch callbacks
- Those callbacks open and close **contexts** useful for the collection of performance counters
- In the **multi-threaded case**, we need a lock on the opening of a context

```
[14:11:44.921782396] (+0.000004902) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9821 }, { kernel_object = 0x9097CF000, counter_index = 6, counter_name = "FlatVMemInsts", value =
 2 }
[14:11:44.921786558] (+0.000004162) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9821 }, { kernel_object = 0x9097CF000, counter_index = 8, counter_name = "FlatLDSInsts", value =
0 }
[14:11:44.921797410] (+0.000010852) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9821 }, { kernel_object = 0x9097CF000, counter_index = 13, counter_name = "FetchSize", value = 40
98.12 }
[14:11:44.921806586] (+0.000009176) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9821 }, { kernel_object = 0x9097CF000, counter_index = 14, counter_name = "WriteSize", value = 40
96 }
[14:11:44.921825138] (+0.000018552) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9821 }, { kernel_object = 0x9097CF000, counter_index = 15, counter_name = "CacheHit", value = 0.0
457317 }
[14:11:44.928077153] (+0.006252015) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9819 }, { kernel_object = 0x9097D0000, counter_index = 0, counter_name = "Wavefronts", value = 32
768 }
[14:11:44.928083043] (+0.000005890) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9819 }, { kernel_object = 0x9097D0000, counter_index = 1, counter_name = "VALUInsts", value = 9 }
[14:11:44.928087767] (+0.000004724) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9819 }, { kernel_object = 0x9097D0000, counter_index = 2, counter_name = "SALUInsts", value = 2 }
[14:11:44.928098791] (+0.000011024) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9819 }, { kernel_object = 0x9097D0000, counter_index = 3, counter_name = "VFetchInsts", value = 0
 }
[14:11:44.928103431] (+0.000004640) paul-gpu hsa_runtime:kernel_perf_counter_float64_nm: { cpu_id = 2 }, { vtid = 9819 }, { kernel_object = 0x9097D0000, counter_index = 4, counter_name = "SFetchInsts", value = 2
 }
```

Linux kernel events

- Some trace points are already defined in the **AMD Linux kernel drivers**
- Some other trace points may be **added**
- Comes in addition with **user space tracing** for more information

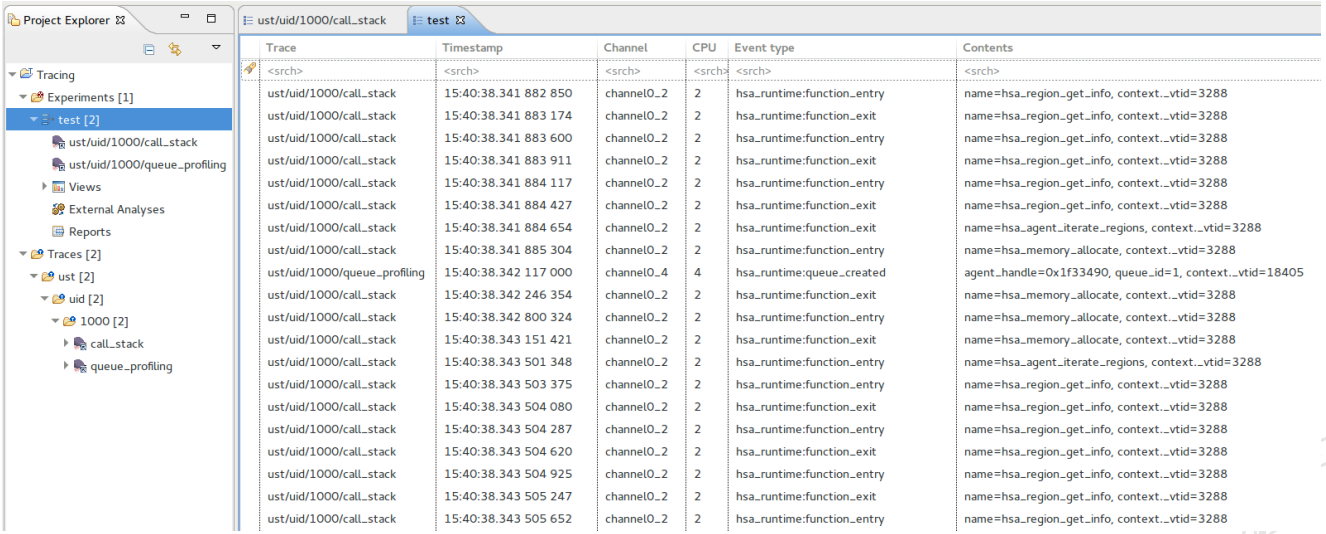
Post-tracing processing

- The 4 user-space tracing targets are mutually **incompatible**
- Traces will have to be **collected separately**, in multiple runs, and then merged or reduced

`call_stack`	(no prerequisite)
`queue_profiling`	(requires a profiled queue with no kernel timing)
`kernel_times`	(requires a profiled queue with kernel timing)
`perf_counters`	(requires specific contexts to be open)

Combining data from multiple runs

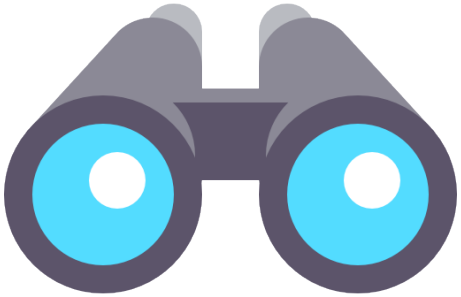
- Early solution: using **Babeltrace** Python bindings
- **Trace Compass experiments** allow merging and offsetting of traces
- Mechanisms for sorting and merging have been proposed for **Chromium traces** and could be re-used



Trace	Timestamp	Channel	CPU	Event type	Contents
<srch>	<srch>	<srch>	<srch>	<srch>	<srch>
ust/uid/1000/call_stack	15:40:38.341 882 850	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 883 174	channel0_2	2	hsa_runtime:function_exit	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 883 600	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 883 911	channel0_2	2	hsa_runtime:function_exit	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 884 117	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 884 427	channel0_2	2	hsa_runtime:function_exit	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 884 654	channel0_2	2	hsa_runtime:function_exit	name=hsa_agent_iterate_regions, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.341 885 304	channel0_2	2	hsa_runtime:function_entry	name=hsa_memory_allocate, context._vtid=3288
ust/uid/1000/queue_profiling	15:40:38.342 117 000	channel0_4	4	hsa_runtime:queue_created	agent_handle=0x1f33490, queue_id=1, context._vtid=18405
ust/uid/1000/call_stack	15:40:38.342 246 354	channel0_2	2	hsa_runtime:function_exit	name=hsa_memory_allocate, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.342 800 324	channel0_2	2	hsa_runtime:function_entry	name=hsa_memory_allocate, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 151 421	channel0_2	2	hsa_runtime:function_exit	name=hsa_memory_allocate, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 501 348	channel0_2	2	hsa_runtime:function_entry	name=hsa_agent_iterate_regions, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 503 375	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 504 080	channel0_2	2	hsa_runtime:function_exit	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 504 287	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 504 620	channel0_2	2	hsa_runtime:function_exit	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 504 925	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 505 247	channel0_2	2	hsa_runtime:function_exit	name=hsa_region_get_info, context._vtid=3288
ust/uid/1000/call_stack	15:40:38.343 505 652	channel0_2	2	hsa_runtime:function_entry	name=hsa_region_get_info, context._vtid=3288

Future work

- Adapt to current work to **OpenCL** applications
- Find more **generic solutions** for trace merging
- Provide more advanced **Linux kernel** tracing



Thank you!

Any questions?

paul.margheritta@polymtl.ca