# Tracing from within GPU

Balboul anas  Michel Dagenais

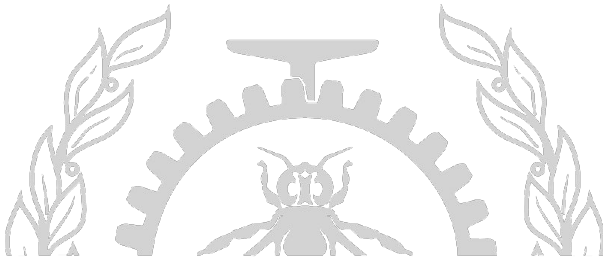
May 10, 2018

Polytechnique Montréal

Laboratoire **DORSAL**

# Agenda

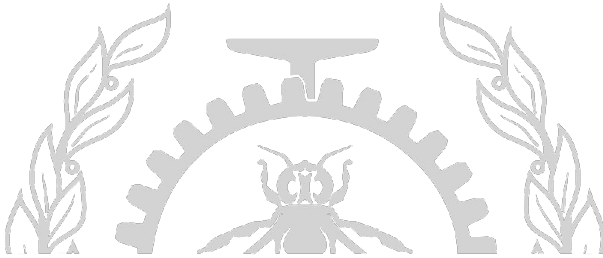
- Introduction
- Literature review
- AMD GCN3 Instruction Set
- Future Work
- Conclusion



# Introduction

**Why tracing a GPU program(Kernel) ?**

- We all know the importance of tracing …
- Many solutions exist to trace a GPU.
- Most of this solution trace the framework for writing programs that execute across heterogeneous platforms (Like OpenCL, HSA, Cuda).
- The need of a way to trace the kernel execution on a lower level.

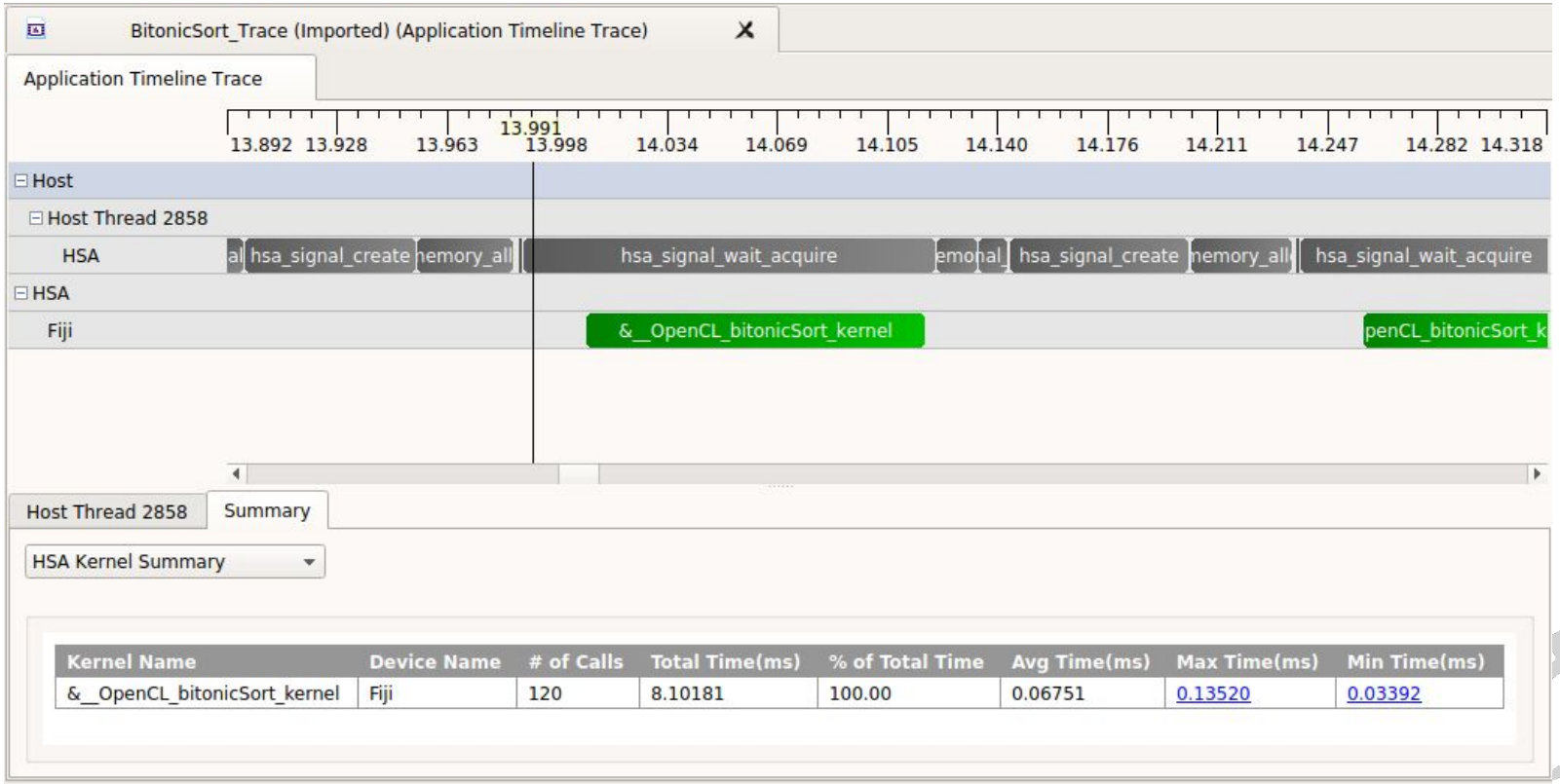


# Introduction

## Definition of the problem

•Because AMD's tools are open source we choose to work with their tools and their hardware.

•Here is an example of a GPU program traced using CodeXL



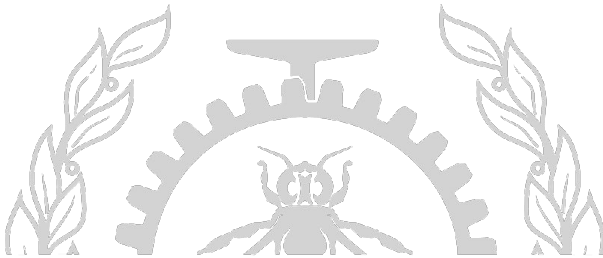

| Kernel Name | Device Name | # of Calls | Total Time(ms) | % of Total Time | Avg Time(ms) | Max Time(ms) | Min Time(ms) |
|---|---|---|---|---|---|---|---|
| &__OpenCL_bitonicSort_kernel | Fiji | 120 | 8.10181 | 100.00 | 0.06751 | 0.13520 | 0.03392 |

# Introduction

## Why tracing kernel from within the GPU is important:

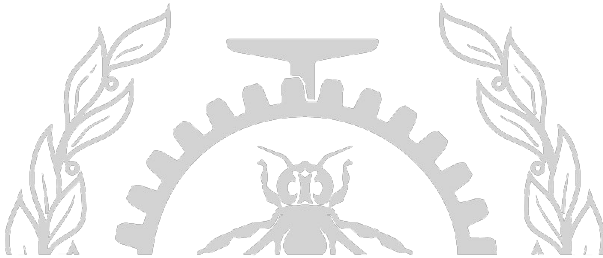
- Here is an example of a kernel that copy a buffer ‘in’ into a buffer ‘out’ :

```
47  prog kernel &__vector_copy_kernel(
48          kernarg_u64 %in,
49          kernarg_u64 %out)
50  {
51  @__vector_copy_kernel_entry:
52          // BB#0:                                // %entry
53          workitemabsid_u32       $s0, 0;
54          cvt_s64_s32     $d0, $s0;
55          shl_u64 $d0, $d0, 2;
56          ld_kernarg_align(8)_width(all)_u64      $d1, [%out];
57          add_u64 $d1, $d1, $d0;
58          ld_kernarg_align(8)_width(all)_u64      $d2, [%in];
59          add_u64 $d0, $d2, $d0;
60          ld_global_u32   $s0, [$d0];
61          st_global_u32   $s0, [$d1];
62          ret;
63  };
```



# Literature review

- Paul Margheritta, Michel R. Dagenais : (Tracing of software applications that uses a GPU)
  - LTTNG-HSA: set of libraries that are meant to be preloaded when executing a GPU-accelerated program.
  - Each library hook into the HSA API calls and insert an LTTng tracepoint.
  - Using GPU performance API to collect some performance counters from the GPU.
  - Unify all the traces in one CTF trace using merging and sorting techniques.



# AMD GCN3 Instruction Set

- Some instructions of AMD GCN3 Instruction Set Architecture that can be used to collect informations :

*Instruction* **S_MEMTIME**

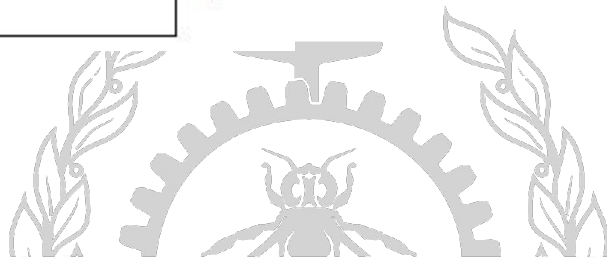
*Description* Return current 64-bit timestamp.This “time” is a free-running clock counter based on the shader core clock.

*Microcode* SMEM Opcode 36 (0x24)

*Instruction* **S_TRAP**

*Description* Enter the trap handler. TrapID = SIMM16[7:0]. Wait for all instructions to complete, save {pc_rewind,trapID,pc} into ttmp0,1; load TBA into PC, set PRIV=1 and continue. A trapID of zero is not allowed.

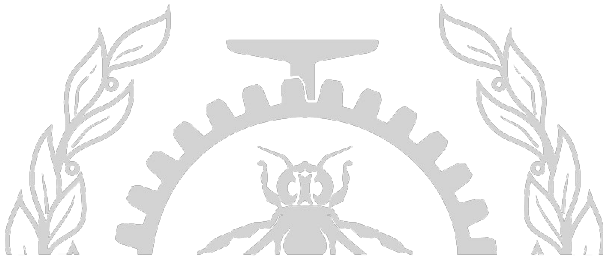
*Microcode* SOPP Opcode 18 (0x12)

# AMD GCN3 Instruction Set

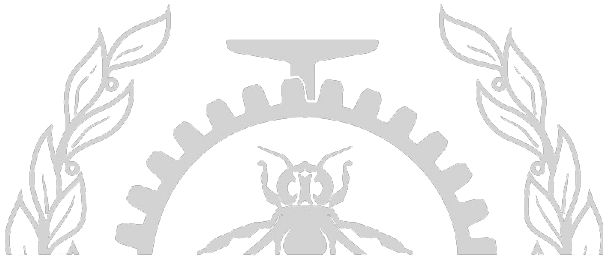
- Some status registers of AMD GCN3 Instruction Set Architecture that can be used to collect informations :

| | | |
|---|---|---|
| IN_TG | 11 | Wavefront is a member of a work-group of more than one wavefront. |
| IN_BARRIER | 12 | Wavefront is waiting at a barrier. |
| HALT | 13 | Wavefront is halted or scheduled to halt.<br>HALT can be set by the host through wavefront-control messages, or by the shader. This bit is ignored while in the trap handler (PRIV = 1); it also is ignored if a host-initiated trap is received (request to enter the trap handler). |
| TRAP | 14 | Wavefront is flagged to enter the trap handler as soon as possible. |

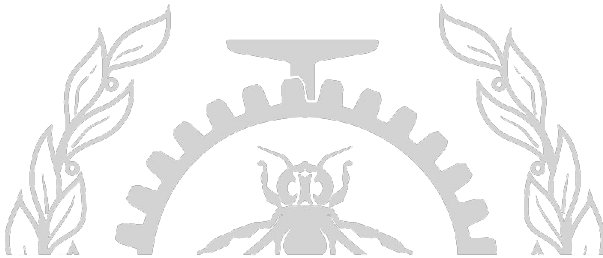


# Future Work

- Tracing and doing more experiences to have a better understanding about the execution of a kernel in a low level using HSA framework.
- Developing new techniques and exploring already existing ones to collect low level information during the execution of a kernel.
- Overcoming performance problems and obstacles that will be induced by the nature of parallel programs.



# Questions?



# References


- *Paul M., & Dagenais, M. R. (2018). : LTTNG-HSA: BRINGING LTTNG TRACING TO HSA-BASED GPU RUNTIMES*