Data-centric Profiling for CPUs and GPU

Working group session report out

Participants:

- Laksono Adhianto
- Terry Jones
- Edgar Leon
- Treece Burgess
- Holly Wilper
- Dave Kaeli
- David Boehme
- Giovanni Baraldi
- Anthony Danalis

Requirements:

- Users want to know:
  - Which program variables and data structures are responsible for high memory latency
  - Desire to work at a source code level - make data actionable
  - For every memory access:
    - Execution time breakdown
    - Size of the object
    - Pinned versus paged memory
  - Report cache misses L1/L2/L3?
  - Provide suggestions for optimization: tiling? fusion?
- Tool developers want to know:
  - Instruction addresses
  - Data addresses
  - Support for user annotations tied to memory addresses

Challenges:

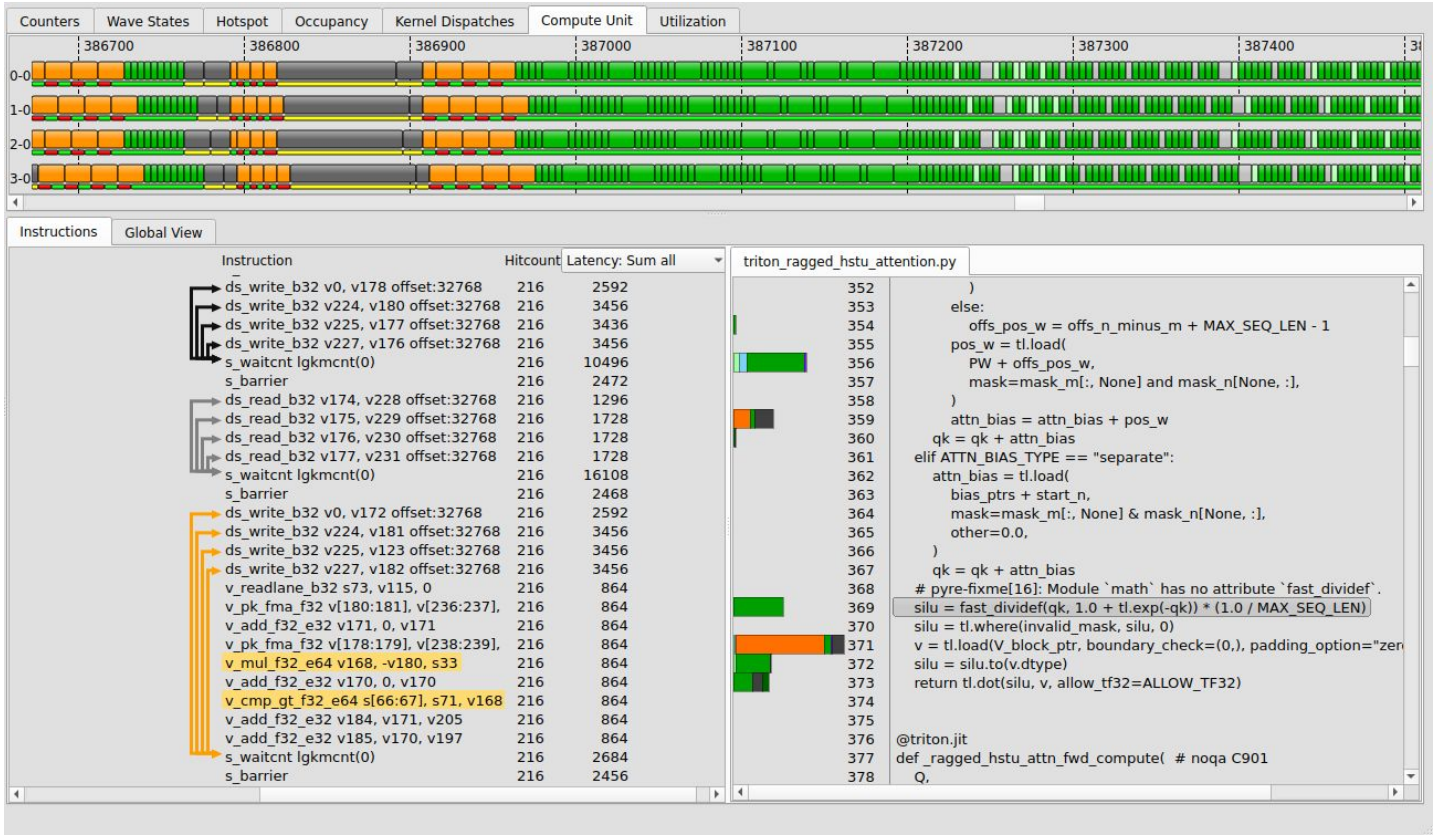
- Unable to attribute delays to program variables due to aliasing
- How can we preserve mappings across runs
- How can we avoid introducing a lot of profiling/tracing overhead
- How can we intercept and track memory allocations

Current GPUs Support:

- NVIDIA
  - Hardware counters
    - Source of memory latency
    - Instruction Pointers
  - Nsight Compute
- AMD
  - Hardware counters
    - Source of latency: L1/L2
    - Instruction Pointers
  - Rocprof thread tracing
  - Rocprof trace decoder (library + API)
- Intel
  - Hardware counters?

AMD rocprof compute viewer

Thread Trace Data



https://github.com/ROCm/rocprof-compute-viewer/blob/amd-mainline/docs/data/isaview.png

Asks:

NVIDIA

- Report memory latency on a per instruction basis
- Add this capability to NVIDIA NSight compute (if it is not already there)
- Can you provide specific hardware counters to help?

AMD

- Report memory latency on a per instruction basis
- Combine hardware counter values with thread traces (AMD Advanced Thread Tracer)

Intel

- ???