Using Rocprofiler-sdk

Working Group Outbrief

Scalable Tools Workshop 2025

attendees

- Yuning
- Ronak
- Sebastien
- Yumeng
- Matin
- Jonathon
- James
- John
- Ben

Experiment with HPCToolkit + Rocprofiler-sdk staging

- Goal: one version of HPCToolkit + Rocprofiler-sdk 1.0.0 for any ROCm version
- Problem: this didn't seem to work in preliminary experiments
- Identified some problems
  - The Rocprofiler-sdk has some new changes relating to page migration, requiring edits to HPCToolkit
  - HPCToolkit isn't properly linking in librocprofiler-sdk from a location outside /opt/rocm-x.y.z
- Fixed the linking problem by informing meson about rocprofiler-sdk location outside rocm
- Now: problem with include path ordering /opt/rocm appearing before rocprofiler-sdk staging
  - Meson + Cmake configuration needs some inspection
    - Later determined that meson alphabetizes include paths!

AMD GPU Advanced Thread Tracing (ATT)

- Will have decoder, does it work if we want to send records from device to host? And How?
  - Save this conversation for tutorial
- Concern: decoder is usually not light-weight, and not recommended for runtime
- There will be a tutorial about ATT in the future
  - AMD’s “Giovanni” will provide more info about it

Causal Profiling

Causal profiling measures optimization potential

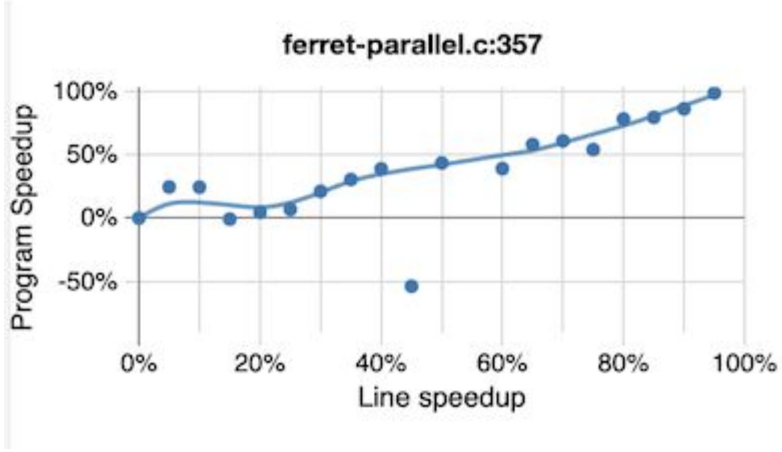
It predicts what the impact of optimizing code will have on overall throughput or latency.

Artifact: Coz - https://github.com/plasma-umass/coz

Video: https://www.youtube.com/watch?v=jE0V-p1odPg&t=28s

Profiles generated by Coz show the "bang for buck" of optimizing a line of code in an application. In the below profile, almost every effort to optimize the performance of this line of code directly leads to an increase in overall performance, making it an excellent candidate for optimization efforts.

How can we apply causal profiling effectively for GPU-accelerated code? Don’t neglect the CPU!



Causal Profiling for GPU-accelerated Code?

- Goal: understand optimization opportunities for GPU-accelerated code
    - Coz inserts delays at source lines in code to see what lines deliver performance that matters
        - E.g. code in loops matters more than code outside
- How: use causal profiling by inserting delays in GPU code
    - Jonathan
        - Possibly insert delays in GPU code with host trap handler
        - See how the idea can shed light on where improvements to GPU code matter
        - For bigger applications, probably want to focus on the kernel level first
            - Too many places to check them all?
        - Issues
            - Programs often have phase-like behavior
                - What comes early may matter less than what comes afterward (setup vs. run)
            - Don't want to focus instrumentation on the wrong places
    - Output: identify GPU source lines where improvement would accelerate an application the most

Progress of causal profiling

Recent paper on causal profiling

- Identifying {On-/Off-CPU} Bottlenecks Together with Blocked Samples. *OSDI 2024*
- https://www.usenix.org/system/files/osdi24-ahn.pdf