AMD GPU Counters Tool Integration via ROCProf SDK

Scalable Tools ‘24 Breakout Session
Tuesday 2-3:30p - Mountain

Participants

- Jen Green, Sandia Nat’l Labs
- Ben Welton, AMD
- John Mellor-Crummey, Rice University
- Stepan Vanecek, TU Munich

Expected Usage

- Iterate over all of the devices
- Create a context per device
- Determine what counters need to be configured for each device
  - A “counter” may collect a raw metric or a derived metric
- Set up the counter configuration for each context
  - Can do this iteratively to determine what HW counters can be fit into the same profile set
- Enable the counter configuration

Problem

- During rocprofiler-sdk initialization, it is not yet known what agents will be used by an application.
  - For instance, one may limit the devices to be used by setting ROCR_VISIBLE_DEVICES
- Don’t want to turn on collection for devices that might not be used by an application
  - May interfere with programs running on other device

A New Approach

Register to receive a callback when the runtime is initialized to find out what devices are to be turned on

Rocprofiler-sdk To Do List

- Fix driver+rocprofiler-sdk to allow PC sampling and counter collection to occur in the same execution
  - Disable clock frequency adjustments for each
- Add new registration API to inform tool what GPUs are in use
- Header file only utility functions to ease tool development

Questions

- Rocprofiler-sdk design
  - GCD vs. agent?
    - How do we know what GCD my MPI rank is using?
  - Can PC sampling and counters be measured at the same time without distorting one another's measurements?
    - GUESS: The only thing that should be affected is time-based measurements
- Tool design
  - How should counters be presented to an application developer?
    - One aggregate count per counter?
    - Per dimension data?

Breakout Session Transcript Summary

https://github.com/jennfshr/amdgpu/blob/main/README.md

Tool Startup Notes (refer to samples/counter_collection/client.cpp)

1. Tool init, create context - can be multiple
2. Create output buffer that stores data, flushed when full via buffer_callback (or manual flush) - performance improvement and less app interruption
    a. Size? Will capture the reponses from 4 devices (e.g.)
3. get_gpu_device_agents(); to determine what can and cannot be instrumented on the machine (need to look at available gpus)
4. For each agent, create a profile, includes the counters you want to collect:
    a. `ROCPROFILER_CALL( … rocprofiler_query_counter_info(...)`
5. rocprofier_create_profile_config… can be called iteratively
    a. Pass the address of the profile, id is returned, use this to find the limit?
6. Some concerns are duplication of code that ROCPROF supplies, so considering header only libraries for generic tool consumption
7. Defer initialization of profile OR defer selection of preestablished profiles
8. Avoid global locking of all GPUs on the system, since that's essentially what counter collection is doing - so decide on what devices to lock is key
9. Counter dims applies to counter entries only, each counter on specific hardware will have an associated set of dims, will return all possible dimensions