Binary Tools for GPU Call Path Tracing

Working Group Outbrief

Scalable Tools Workshop 2025

Attending members

Ronak Chauhan, Matin Raayai-Ardakani, Barton Miller, Hsuan-heng Wu, Angus He, Yumeng Liu, David Kaeli, Sebastien Darche

Yumeng’s Instrumentation

- Based on NVBit
- Information collected: Global warp id, mask, PC, global timestamp
- Collected at function entry, function exit, call site
- “CUDA-style” instrumentation : select a single active thread per warp to push record
    - It would be equivalent to scalar instructions
- One single shared tracing buffer between warps
- Build HPCToolkit CCT from the buffer (on the fly)

NVBit Channels

- Two buffers, one in host memory and the other on the device
- Once the buffer is full, a flag is set to tell the host to copy the data from device to host
- Device is busy-waiting on a variable on unified memory while the host retrieves the data
- Compensate for waiting times (distortion) by ignoring the time that the host copies the data
- Is it possible with the HSA memory model ?

PC

- Object code is randomly loaded in memory - so the PCs from the trace are also shifted randomly
- Would need to store (or compute) relative PC from where the object is loaded
-

Dyninst

- Can’t do it (yet!)
- Scalar instrumentation
- Need channel implementation
- Timestamp is rather easy
- Flat wave id is hard to compute, surprisingly enough
- Dynamic dispatch and function pointers must be handled separately
    - What if one of the kernel’s arguments is a function pointer?

Instrumentation variables

- Should they exist per-kernel, per-stream, per-launch ?

Side-note : Tail-call optimization

- Do GPU compilers implement TCO (and are instrumentation tool capable to recognize it) ?