Improving the Tooling Interface Ecosystem

Working Group Outbrief

Scalable Tools Workshop 2025

Needs

Must support use of multiple tools

Appropriate sequencing between runtime and tools.

- Static constructors are a bit of a problem – everybody wants to be first
- Hardware needs to be initialized/TLS needs to be initialized/runtimes

Would like a gestalt view of linking of the application

- Where am I in the initialization - no way to see where we are and what is already loaded.

Challenge: Sometimes tools need to attach to a process (GDB/ptrace)

- Need a way to get the current state of the process
- Lots of Linux kernel difficulties, might be made easier for ROCm if GPUs were properly integrated into proc space
- Many many fundamental differences vs. first-party interfaces

Common Requirements

- Multiple tools must be able to monitor at once
- Appropriate sequencing between runtime and tools.
    - Static constructors are a bit of a problem – everybody wants to be first
    - Hardware/TLS/runtimes/etc. need to be initialized at the right time
    - Could possibly be achieved through a “gestalt” view of application initialization
        - So tools that run at odd times can check what is or isn’t loaded yet
- What about tools late-attaching to a process? (Think GDB)
    - Need an interface to get the current state of the process
    - ROCm looking into this for their interface, serious ordering challenges in the Linux kernel
        - Ben: This might be easier if GPUs were actual Linux processes
    - Interface design is vastly different compared to first-party interfaces
- Performance impact when “off” must be near zero

Tooling for orchestration/resource manager

RM: flux/slurm
Orchestration/workflow: kubernetes, merlin
Highly multitool - occurs at multiple levels task level data vs. workflow level

- Things are dynamic

We see things at task level not at the higher level

- Detect conflicts
- See how things fit together

Target Audience

- We want tooling in GPU runtimes
  - HPCToolkit: overall successful collaboration with ROCm, requested Intel copy from them
- We want tooling in CPU runtimes
  - E.g. OpenMP
- We want tooling in workload orchestrators/resource managers
  - Examples: K8s, Merlin, Flux, Slurm
  - Workload performance ≠ task performance
    - Multiple levels of performance to consider, highly multitool
- We want tooling in Glibc
  - Allocator (malloc/free/…) hooks
  - Pthread operations (create/mutex/…) hooks
  - Filesystem I/O (open/close/stat/read/write/realpath/…) hooks
  - …and others

Runtime initialization

We have solved problems with ROCm

Can Intel copy the overall design? (current solution from Intel is not acceptable)

- Pre-and-post thread creation callbacks. Pre-create callback indicates role of the forthcoming thread
- Runtime initialization - tool do your stuff now
- Tool issues upcall saying I’m initializing
- API that enables one to connect GPU operation invocation context to performance information for a GPU event
- PC Sampling with correlation id enables a tool to correlate back to invocation context
- Correlation ids for relating asynchronous activities and the lifetime managment

Make a base standard for tooling and runtime.

Multiple tools need to be able to attach

Action Items

- Collect lessons from ROCm interaction, for GPU tooling interfaces
  - Role markers for runtime-created threads
  - Tool boot trigger on runtime initialization
  - Runtime boot trigger on tool initialization
  - Correlation between GPU performance events and CPU calling context
    - Especially for instruction-level performance (PC sampling)
  - Correlation between related asynchronous events
  - Lifetime management for correlation ids
- Start standardizing the tooling interfaces we want
  - No one likes making standards… but only way to combat API divergence
  - Start with the runtimes we care about and expand from there

The Pitch

- Symbol wrapping is not a sustainable solution
  - Optimized libraries (e.g. CUDA) don’t support wrapped symbols, internal calls avoid PLT
  - “Shadowing” the upstream APIs takes persistent development effort from tool developers
  - Wrong abstraction level, missing mapping information to make sense of performance
- Tooling interfaces in libraries/applications lead to more robust tools
  - Maintained by the actual developers (who understand their APIs!) instead of a third party
  - Less effort for tools to adopt new tech, more man-time can be spent on useful tasks (& interesting problems)
- We want widespread high-quality tooling interfaces
  - What can we do to make this happen?

Resources

- OpenMP / OMPT: https://www.openmp.org/spec-html/5.1/openmpch4.html
- ROCProfiler-SDK: https://github.com/ROCm/rocprofiler-sdk/blob/30e239d1b7bb74df8d255f01956e4631780723d8/source/include/rocprofiler-sdk/registration.h#L222
- PerfStubs: https://github.com/UO-OACISS/perfstubs
- “The Case for a Common Instrumentation Interface for HPC Codes” https://ieeexplore.ieee.org/document/8955675

Random

We need a generic demangler that avoids the baggage of libiberty

- Libc++ doesn’t do Rust, doesn’t have all the features
- LLVM libdemangle is missing features (also LLVM)

Off-topic Discussion: Symbol Demangling

- Background: Binutils (libbfd, libiberty, etc.) is missing shared libraries
  - Binutils doesn’t maintain a stable ABI, so major distros don’t ship shared libraries
  - Some past difficulties getting -fPIC static libraries, but fixed in recent Fedora/RHEL
- Fedora/RHEL: static linking is **STRONGLY** discouraged
- Most usage of libbfd can be replaced by Elfutils
  - HPCToolkit made the jump successfully a few years ago
  - Spindle currently using a mix, but could consider moving to pure Elfutils
- Challenge: libiberty’s symbol demangler has no direct replacement
  - C++ ABI: Only for C++ symbols (not e.g. Rust), missing options, not usable from pure C
  - LLVM libdemangle: Missing options, not usable from pure C, requires building LLVM
  - Need a new “libdemangle” to replace libiberty

Attendees

- Jonathon Anderson
- John Mellor-Crummey
- Ben Woodard
- Matt LeGendre
- Jonathan Madeson
- Add your name here