Contribution ID: **317** Type: **not specified**

Extending eBPF to GPU Device Contexts

Widely used for ML workloads, GPUs are typically SIMT accelerators with threads in warps on SMs, organized into blocks, launched as kernels, using multi-level memory hierarchies (registers, shared/LDS, L2, device memory) and limited preemption. This complexity creates rich but challenging behavior patterns for observability and customization. Today, many tracing tools for GPU workloads sit at the CPU boundary (e.g. probes on CUDA userspace libraries or kernel drivers), which gives you host-side events, but treats the device as a black box: little visibility inside a running kernel, weak linkage to stalls or memory traffic, and no safe way to adapt behavior in-flight. GPU specific profilers(e.g. CUPTI, GTPin, Nvbit, Neutrino) provide device-side visibility, but they are often siloed from eBPF pipelines, make it harder to corelate with events on CPUs.

We prototype offloading eBPF into GPU device contexts by defining GPU-side attach points (CUDA device function entry/exit, thread begin/end, barrier/sync, memory ops, etc) and compiling eBPF programs into device bytecode (PTX/SPIR-V), with verifier, helper, and map support for on-device execution. Built on top of bpftime, this approach can be 3-10x faster than NVBit, is not vendor-locked, and works with Linux kernel eBPF programs like kprobes and uprobes. This enables GPU extensions like fine-grained profiling at the GPU thread, warp or instruction level, adaptive GPU kernel optimization, and programmable scheduling across SMs with eBPF. It can also help accelerate some existing eBPF applications.

The goal of this talk is to explore the usecases, challenges and lessons learned from extending eBPF's programming model to GPUs.

https://github.com/eunomia-bpf/bpftime/tree/master/example/gpu

Primary authors: ZHENG, YUSHENG; YU, Tong (PLCT); YANG, Yiwei (UCSC)

Presenters: ZHENG, YUSHENG; YU, Tong (PLCT)

Session Classification: eBPF Track

Track Classification: eBPF Track