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

Extending eBPF to GPU Device and Driver Contexts

Friday 12 December 2025 16:00 (30 minutes)

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 with complex CPU side orchestration in kernel driver and userspace runtime. 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 user-space libraries or kernel drivers), which give you host-side events but treat the device as a black box: there is 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 and make it harder to correlate with events on CPUs. At the same time, modern GPU stacks increasingly embed critical one-size-fits-all policies like memory management, scheduling, and observability inside closed-source vendor libraries and opaque driver modules, making them difficult to adapt for dynamic workloads.

We prototype offloading eBPF into GPU device contexts by defining GPU-side attach points (CUDA device function entry/exit, thread begin/end, 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 3–10× faster than NVBit. This enables GPU device extensions like fine-grained profiling at the instruction level and programmable scheduling across SMs with eBPF.

We also introduce gpu_ext, and extend the Linux GPU driver with verified eBPF attach points, allowing adaptive customization of GPU scheduling and memory management policies at runtime that work together with the GPU device extensions.

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

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

https://github.com/eunomia-bpf/gpu_ext

https://arxiv.org/abs/2512.12615

Primary authors: ZHENG, Yusheng (eunomia-bpf); YU, Tong (eunomia-bpf); YANG, Yiwei (eunomia-bpf)

Presenters: ZHENG, Yusheng (eunomia-bpf); YU, Tong (eunomia-bpf)

Session Classification: eBPF Track

Track Classification: eBPF Track