eGPU: Extending eBPF Programmability and Observability to GPUs

Yiwei Yang, Yu Tong, Yusheng Zheng, Andrew Quinn
Center for Research in Systems and Storage
University of California, Santa Cruz

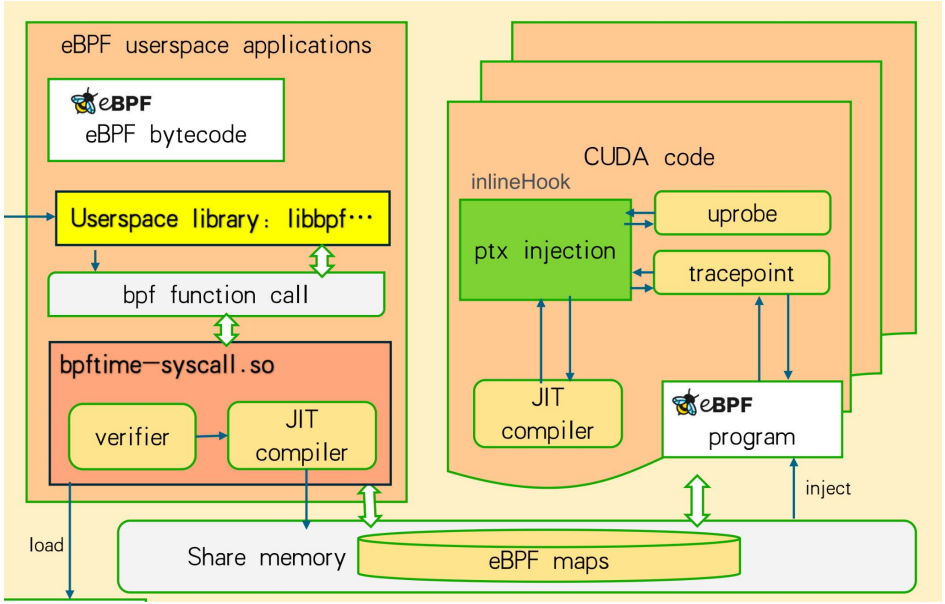
UC SANTA CRUZ
BaskinEngineering
CRSS

Introduction to GPU Observability & Motivation

- **Rapid Growth of GPU Workloads**
  - HPC simulations, large-scale ML/LLM, data analytics
  - Need low-overhead observability to ensure performance
- **Traditional Profiling Challenges**
  - Often intrusive; limited real-time insight
- **Goal**: Combine real-time GPU telemetry & dynamic instrumentation to identify bottlenecks quickly

Introducing eGPU

- Core Idea: Extend eBPF support to the GPU!
  a. Advantages:
     i. Add GPU telemetry for profiling.
     ii. Integrated into current workflows.
     iii. Lightweight!



Comparison with Other Approaches

- **NVIDIA Nsight/CUPTI**
  - Deep profiling but intrusive, dedicated runs
- **Strobelight/BPF (Meta)**
  - Hooks CPU-level calls; less granular inside GPU kernels
- **eGPU Advantages**
  - Dynamic injection
  - Fine-grained event-level instrumentation in GPU

eGPU Challenges and Solutions

- Must inject new eGPU tracing logic into running GPU kernel. But, current systems do not support self-modifying kernels!
  - Key Idea: use checkout and restore techniques to replace running PTX program (cite).
- Must coherently share memory across the GPU and CPU for storing telemetry data.
  - Add support to eBPF shared memory maps using CPU and CUDA atomic primitives.

Performance Evaluation Setup

- **Hardware**
  - Dual-Socket Intel Xeon + NVIDIA P40 GPU
- **Tests**
  - Compare eGPU vs. native GPU kernels + nvbit
  - Microbenchmarks (small kernels) & streaming tasks
- **Metrics**
  - End-to-end latency
  - Throughput (tasks/sec)
  - Overhead vs. baseline