

eBPF Kernel Tracing Overhead & Performance Handbook

Analyzing Kprobe Traps, Cache Thrashing, and Production Cloud Latency

Author: Lesty Tech Systems Engineering Group | Date: October 2026 | Verified Technical Documentation

1. The eBPF Execution Model Inside Linux

Extended Berkeley Packet Filter (eBPF) provides an in-kernel virtual machine executing JIT-compiled bytecode. While eBPF eliminates user-space context switches for network packet filtering (XDP), deep observability tools attach probes to high-frequency kernel symbols.

When dynamic probes (kprobes) are attached, the kernel replaces the target instruction with an INT3 breakpoint or a ftrace trampoline. When the instruction executes, the CPU dumps register state to the stack, invoking the BPF program.

2. The Hidden Cost of Microscopic Traps

On high-throughput systems processing millions of network packets per second, executing a kprobe on every network socket receive or system call induces catastrophic L1/L2 data cache evictions and pipeline stalls.

CRITICAL: Unoptimized eBPF monitoring suites can consume between 15% and 28% of total host CPU cycles simply managing ring buffer synchronizations and per-CPU hash map lookups.

Empirical Laboratory & Architectural Telemetry

Mechanism	Kprobe (Dynamic Trap)	Raw Tracepoint	Fentry / Fexit (Modern)
Instruction Replacement	INT3 Breakpoint / Trap	Static Jump Table	Direct Call Trampoline
Latency per Invocation	120 - 180 ns	15 - 25 ns	8 - 14 ns
Cache Pollution	Severe (Context Dump)	Minimal	Negligible
Overhead @ 500k ops/s	18.4% CPU Core Saturation	3.2% CPU Core Load	1.6% CPU Core Load