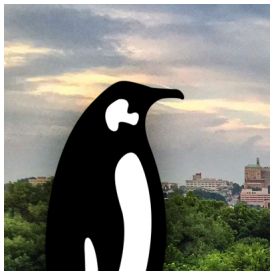

Contribution ID: **295** Type: **not specified**

xprobes: Hybrid User/Kernel eBPF Probes for Cross-Layer Observability

Tuesday, 14 November 2023 10:30 (30 minutes)

eBPF is fundamental for diagnosing performance issues in production environments - where flexible and continuous profiling is key. Understanding, for example, why some functions are taking too long can provide a quick path to uncovering the root cause of performance issues. However, high-level indicators such as high latency are not informative enough: To disambiguate the source of high latency, one must consider not only underlying functions, but also system resources and kernel events - e.g., is the latency of a slow function because of CPU contention? blocking on I/O or locks? or just slow processing?

In this talk we will start by analyzing the shortcomings of widely-used tools (e.g., flamegraphs, distributed tracing, and BCC probes) for this kind of diagnosis. For example, BCC kprobes have visibility over kernel events, but will only attribute their measurements to entire processes (PIDs). Alternatively, BCC uprobes have higher granularity and attribute to specific userspace functions, however, they lack visibility over the system events (e.g., context switches) which influence their measurements. We will propose xprobes, hybrids between uprobes and kprobes with visibility over application functions as well as kernel events. For profiling, an xprobe maintains aggregate distributions for the current set of functions of interest. These aggregates are updated jointly by userspace (e.g., function entry/return) and kernelspace events.

xprobes rely on probe-specific code for merging data from kernel and userspace events. An example is our cpu-latency xprobe, which measures only the on-cpu time of userspace functions of interest. Achieving this is complicated because context switching can happen in-between the function entry and return, meaning that only getting the diffs from function entry/return events will result in wall-clock latency, not actual on-cpu latency distributions of each function. For this xprobe specifically, our solution still involves uprobes recording start and end times, but also has kprobes write information to correction variables so as to account for context switching (e.g., if a function is switched out of the CPU, subtract all the time it's been away from the CPU from the final latency measurement).

We will show how the rich data from xprobes can provide powerful performance insights. For example, if, for a given function, we compare the wall clock latency distributions with distributions from our xprobes (e.g., actual on-cpu latency and I/O latency) we can help disambiguate the root cause of a performance degradation. Finally, we will discuss possible avenues for managing the overheads of xprobes.

Primary authors: CASTANHEIRA, Lucas (CMU); BENSON, Theophilus (Brown University)

Presenter: CASTANHEIRA, Lucas (CMU)

Session Classification: eBPF & Networking

Track Classification: eBPF & Networking Track