A Replay-based Approach to Performance Analysis

Anton Burtsev   Eric Eide   John Regehr
University of Utah, School of Computing
{aburtsev, eeide, regehr}@cs.utah.edu

The performance analysis of complex software systems is a difficult task. Designed to deliver the highest performance, modern systems utilize multiple forms of parallelism and a number of engineering optimizations. The performance of such systems is largely determined by the availability of data, latency of communication, delay-free synchronization, and efficiency of scheduling algorithms.

There is an inherent problem in the structure of performance analysis as it exists today. The strict requirement of low run-time overhead forces analysis tools to collect only the smallest subset of the system's state. The entire analysis is performed offline. Lacking complete state of the system, the analysis has no means to reason about collected data with respect to execution history and run-time state of the system. Analysis tools are mostly aimed at conducting lightweight performance measurements and provide no means for explaining, solving, and predicting performance.

Our work argues for a new approach to performance analysis. Two properties are essential for evolving existing approaches to performance analysis. First, analysis algorithms should have an access to the complete execution history and run-time state of a system. Second, the complexity of analysis algorithms should not be restricted by the requirements of a low run-time overhead.

Our goal is to turn performance, a dynamic property of a particular execution, into a static property that can be analyzed separately from an actual instance of a running system. We rely on the old idea of decoupling analysis from execution (Balzer, AFIPS'69). Recent advances in virtualization enable the application of this idea to complex, production-quality systems by demonstrating full-system execution recording with an overhead of only several percent (Xu et al., MoBS'07).

Analysis framework. We present a new analysis framework aimed to provide general support for replay-based performance analysis. We view it as a low-level mechanism that exports the behavior of a system to a higher level, at which analysis is formulated in a platform-independent manner. Several mechanisms implement this idea.

Execution recording. Full-system virtualization has been demonstrated as an efficient platform for recording the execution of systems. We are extending Xen, a full-featured, open-source virtualization platform, with deterministic replay capabilities. Xen offers an excellent virtualization performance and is widely deployed as a virtualization platform in commercial cloud datacenters and large-scale academic research facilities.

Performance model. We will recreate performance properties of an original run by re-executing the system on the same hardware as during the original run. In contrast to replicating a complex hardware state of the CPU in a simulator, replay on real hardware has the potential to provide us with a simpler replay architecture, better replay performance, and higher realism of the performance model.

Our system will provide a uniform interface to the performance of a system during replay via the virtual performance counters. Virtual counters virtualize the hardware performance counters exposed as a low-level interface by modern CPUs. Virtual counters account for the impact of replay mechanisms on the performance of a system, and report performance as it would be observed during the original run.

Analysis interface. We make a key observation that any performance analysis infers general properties of a system by observing changes in its state. A range of dynamic analyses require tracking the state of a system on every instruction. Such fine-grain intervention with the system requires implementation of complex binary translation mechanisms that merge analysis and system code for efficient execution.

Fortunately, a number of performance analyses can be specified in a coarser manner and require an invocation only when a specific a specific point in execution is reached. We leverage this observation to avoid the complexity of integrating binary translation mechanisms into our system. Instead, we rely on binary rewriting to trigger invocation of the analysis code at any place of the system's execution path.

Vision for novel analysis techniques. Several unique properties of our approach enable new ways of analyzing the performance of complex systems. The determinism of analyses and the availability of the global run-time state of the system and its execution history provide support for analysis of transient performance anomalies, evaluating effects of multiple interleaving bottlenecks, and correlating the performance behavior of a system with its functional properties.

The availability of explicit control flow and the possibility of computing request-processing paths enable automatic identification of critical execution paths and performance bottlenecks. By examining the run-time state of a system, it is possible to collect a variety of traditional performance measurements, e.g., disk I/O throughput, network throughput, and so on. Different performance metrics can be collected over multiple replay runs. The determinism of replay ensures consistency of these metrics.

The possibility of replaying a system multiple times provides a way to measure performance of basic instruction blocks. This fine-grain performance profile can be used as an input to static approaches to performance analyses, which execute the system in an abstract space (Wilhelm et al., ACM TECS'08).

Students: Burtsev. No demo is planned.