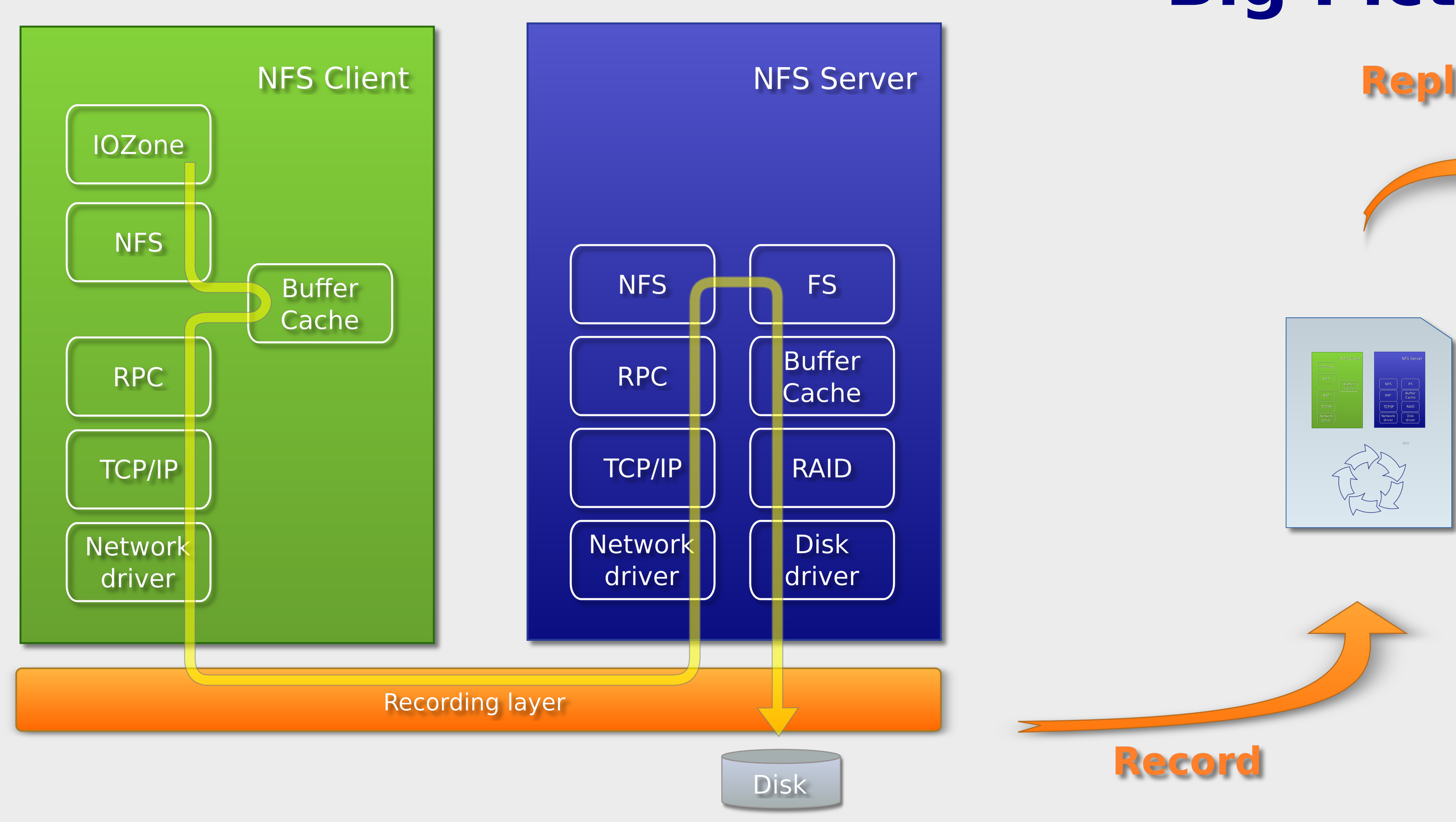
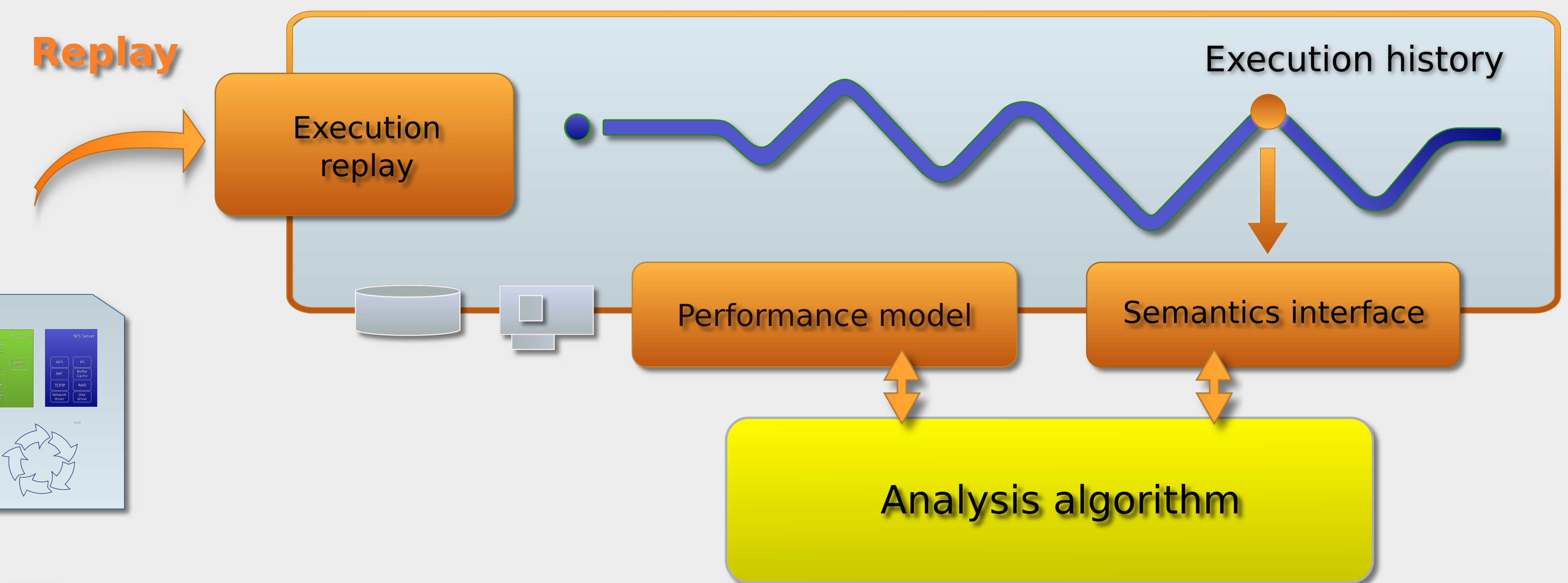
A Replay-based Approach to Performance Analysis

Anton Burtsev, Eric Eide, John Regehr
University of Utah, School of Computing

Big Picture





NFS benchmark: IOZone, a filesystem benchmark, runs over an NFS-mounted filesystem on a client machine. Processing of every filesystem write involves two machines and multiple operating system components until it reaches the physical disk. Request path is shown with a yellow line. Configuration of the NFS protocol, number of NFS server threads, size of TCP/IP buffers, configuration of RAID and disk write buffering, and many other factors affect performance of the system.

Analysis framework: We 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. An analysis framework is 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 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

Motivation

Complex systems
- Multiple engineering optimizations
- Composed of simple components that promptly evolve into complex artifacts
- Emergent behavior

Performance is determined by
- Availability of data
- Latencies of communication
- Delays of synchronization
- Efficiency of scheduling

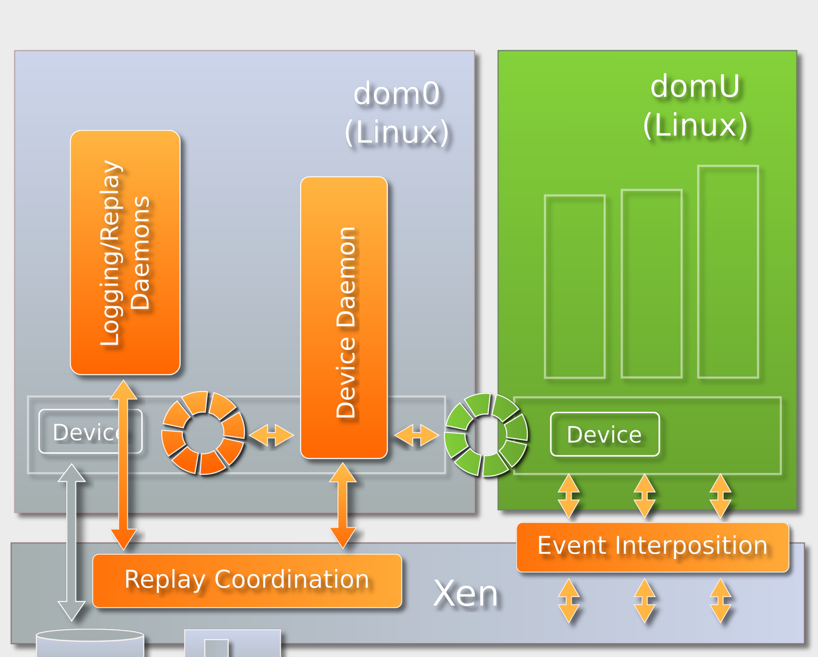
Analysis requires reasoning about
- Dynamic state of multiple components, buffers, and caches
- Control and data flow between them
- Performance of individual requests (slow and fast paths)
- Availability of resources for pipelined and parallel execution

Existing approaches are inherently limited
- Strict requirement of low run-time overhead
- Collect only minimal subset of the run-time state
- No means to correlate collected data with actual system's state

Idea

- Capture complete execution with deterministic replay
- Run analysis offline
  - An old idea (Balzer, AFIPS'69), which is enabled by recent advances in virtualization (Xu et al., MoBS'07)

Execution replay



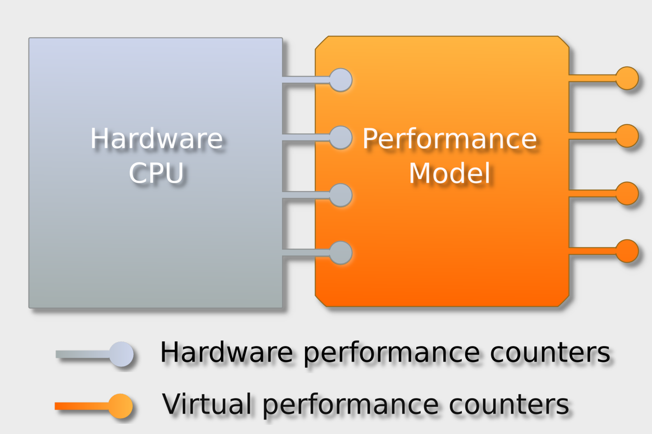
Goal:
- Realistic systems
- Realistic workloads

Combine recording mechanisms with a full-featured VMM
- Low-overhead recording
- Analysis of the entire software stack

Logging and replay infrastructure: four logging and replay components and a high-bandwidth communication channel across them are designed to implement lightweight recording of all nondeterministic events.

This material is based upon work supported by the National Science Foundation under Grants No. 0524096, 0709427. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

Performance model



Re-execution approach to performance
- Identical hardware
- Recreate conditions of the original run

Performance counters
- Export performance for analysis
- Simple linear model
- Need to record only time

Virtual performance counters: account for effects of replay mechanisms, and translate performance between original and replay runs.

Analysis interface

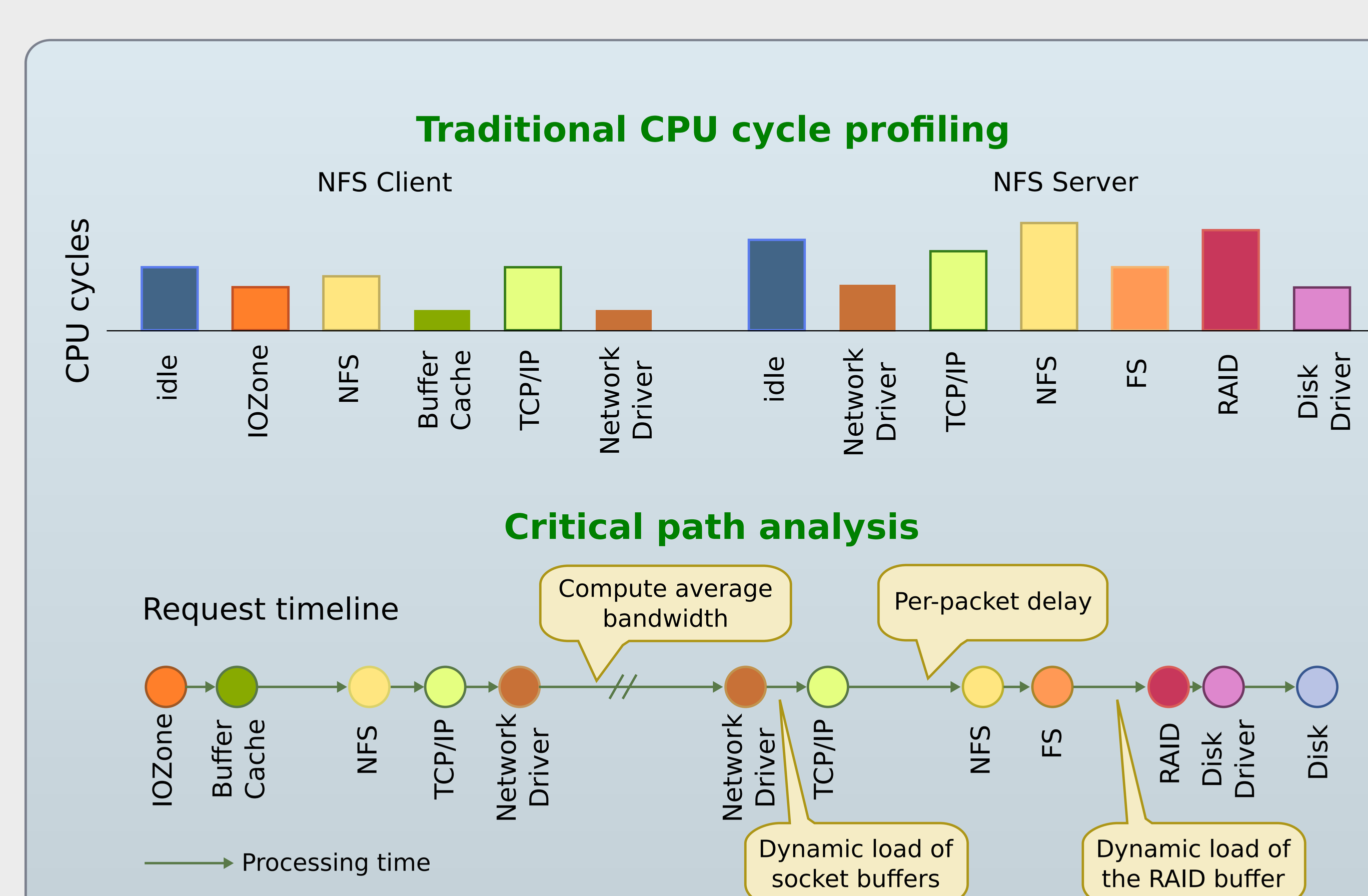
Analysis is driven by changes in run-time state of the system
- "Big step" semantics

Binary rewriting to instrument execution (SystemTap)
- Safe analysis language

```
probe kernel.function(*@mm/*.c).call
      { called[probefunc()] <<< 1 }
```

We integrate it with deterministic replay mechanisms

Insight into analysis algorithms



Multiple replay sessions allow us to run different analyses and collect various performance measurements in a consistent way.

- Automatic generation of the request-processing path
- Fine-grain performance model
- Automatic search for transient performance anomalies
- Combination with static analysis