Benchmarks Can Make Sense
Anton Burtsev, Eric Eide, Mike Hibler, John Regehr
Flux Research Group, School of Computing

Motivation
Benchmarks are designed to compare systems
- Return a single number (e.g. throughput, average delay, completion time, CPU utilization, transactions per second)
No way to identify performance bottlenecks
- A simple configuration error can invalidate all results
- Netfork file system (NFS) benchmark below:
  - No disk write buffering (50%)
  - Synchronous NFS mount (60%)
  - Default NFS request buffer (30%)
No way to debug the system

Goal
Benchmarks should make more sense
- Verify benchmark setup
- Aid performance analysis
- Performance bottlenecks, verbose performance model

Implementation
Full-system replay:
- Xen virtual machine monitor
- Support for production workloads
We extend it with a low-overhead recording
- Cooperative logging for network of machines
- Versioning storage for deterministic disk communication

Performance model:
Do performance measures stay sound during replay?

Replay
1) Set branch counters to overflow (cause exception)
2) Iterate in a single-step CPU mode to a target IP
3) Inject external event
Analyze ILP during original and replay run

Analysis interface:
Additional research needed to understand which information can actually help performance analysis
- Performance metrics
- Functional properties of the system

Big Picture

NFS benchmark: IOZone, a filesystem benchmark, runs over an NFS mounted file system on a client machine. Processing of every file-system write involves two machines and multiple operating system components until it reaches the physical disk (request path is shown with a yellow line).

Challenges
Efficient full-system replay
- Replay may interfere with the execution of a system
Debugging and analysis interface
- Non-intrusive probes
- DTrace-like language interface to collect information about execution

Status
Implemented a basic deterministic logging and replay infrastructure
- Can replay beginning of the Linux boot (650K instructions)
- Replay mechanisms are designed to treat the state of a guest system as a set of memory pages
- Right choice to support heterogeneity of replay in the future
Support most non-deterministic events
- Lack support for logging device driver communication