Time Travel for Closed Distributed Systems

Anton Burtsev Prashanth Radhakrishnan Mike Hibler Jay Lepreau

School of Computing, University of Utah
www.flux.utah.edu

The ability to replay an execution is a powerful aid in analyzing and debugging long-running distributed applications. Our goal is to time-travel a network of hundreds or thousands of virtual machines spread across multiple physical hosts in the Emulab testbed environment. Our time-travel system will provide the ability to navigate through the original execution history, replay it from an arbitrary point, and—we believe—even change its state without proscribing further replay.

The existing work, on which we build, either implements deterministic time-travel of a single VM or is restricted to multiple VMs running on a single physical host. Our work differs in two ways:

- During replay we optionally allow relaxed determinism, which permits a user to mutate system state during replay. Such mutations are important for debugging, but violate determinism. Given the best-effort service model of the Internet and the unreliability of distributed entities, we conjecture that large classes of applications are tolerant of non-deterministic replay: they must already cope with lost, reordered, and delayed packets, as well as unexpected failures of all kinds.

- We emphasize scalability in the system's design. Towards this goal we spread virtual machines across physical hosts, employ cooperative replay, optionally relax determinism, and assume a "closed world."

Experiments in Emulab typically operate in a closed world, in which nodes within an experiment communicate only with each other. Operating a time-travel system in a closed world offers opportunities to improve its scalability and to explore special characteristics, such as relaxed determinism. In Emulab we can also make the assumption that we have a reliable, low latency network fabric for the control plane, simplifying the overall system and bounding clock skew to the microsecond-range using Veitch's techniques.[1]

It is interesting to review the closed-world assumption for the deterministic and non-deterministic cases. The assumption improves *scalability* of the former, by not having to log messages from the external world. However, a closed world is required for the latter's *correctness* (e.g., a message to the external world during non-deterministic replay may not elicit a response because the external world is non-cooperative).

When complete, our work will help answer three questions: Is non-deterministic replay useful for many applications? Does non-deterministic replay have performance advantages? How fast is a physically distributed time-travel system?

The system is designed and partially implemented. Our poster will describe the motivation, design, research questions, and outline the prototype implementation.

Implementation: Like Chen and others, we run all nodes under the control of a virtual machine monitor. Our Xen VM encapsulation enables transparent tracking of non-deterministic events in the system (e.g., timer and I/O interrupts). Control over the system's virtual memory lets us checkpoint a running VM without suspending its execution. For faster time-travel we combine logging with periodic consistent checkpointing. We use existing non-blocking checkpoint techniques which tag all outbound packets with an "epoch-id" to demarcate checkpoint epochs.

To recreate the original execution, we log all non-deterministic events during the original run and replay them during the subsequent "time-travel" runs. We provide efficient navigation within the execution history by combining logging with periodic distributed checkpointing. To address scalability issues we employ cooperative logging, where time-traveling peers cooperate to recreate the original run. For example, messages from the peers need not be logged because they will be resent during replay. For large long-running distributed systems, cooperative logging mitigates the problem of log storage space exhaustion.

Students: Burtsev, Radhakrishnan. Corresponding author: aburtsev@cs.utah.edu.

[1] D. Veitch, S. Babu, A. Pasztor, "Robust Synchronization of Software Clocks Across the Internet," IMC 2004.