CSE 231—Advanced Operating Systems
"ReVirt"
or "What can we do with virtualization??"

Andrew Quinn

A note on the Authors. The advisor on the Revirt paper has been accused of heinous crimes. Incidentally, he was my advisor when he was arrested. I have no idea if he did it, nor is it even reasonable for me to postulate.

This puts this paper in a tough spot. Revit is one of my favorite papers and was instrumental in my thesis. While Revirt isn't the first paper on record and replay, it is the genesis of tons of work on record and replay in the OS, SE, PL, and architecture communities (and, won a test of time award as a result).

Ultimately, I decided to review this paper mainly because academics do not make money from us studying their work; we gain knowledge while they gain very little. Moreover, I decided to cover the work with two main stipulations. First, while we often discuss the people behind the research, we won't do that here. Second, publicity, citations, and review are not equivalent to and endorsement of the people who made the work (as stated above, I have no idea of the authors guilt or innocence).

Summary. The trickiest technical bit in this paper is the discussion for how they handle interrupts. The challenge is that the system needs to ensure that each interrupt is delivered at precisely the same time during replay as it was during recording. Some system prevent interrupts from being raised at arbitrary points in the execution [2, 3]. However, ReVirt instead guarantees that each interrupt will be raised before the same instruction.

The strawman approach instruments the program and counts all executed instructions during recording and replay. Unfortunately, this would be painfully slow. Note, however, that there are other counting schemes that

can still uniquely identify each execution instruction. For example, each pair of (branch count (`bc`), program counter (`pc`)) uniquely defines an executed instruction, so an interrupt raised before the same (bc, pc) pair during recording and replay will be correct.

ReVirt uses to Intel's performance counters to accelerate the pausing before a branch counter. The system tells the performance counters to raise an interrupt after $\texttt{bc} - 1$ branches and then the processor single steps to `pc`. Unfortuantely, performance counters have "skid"—they may raise an execution a few instructions after the `bc`'s branch. So, ReVirt actually chooses an arbitrary value that will always be larger than the skew, and raises the interrupt after $\texttt{bc} - 128$ instructions and then single-steps to the right spot.

Discussion. A major challenge facing record and replay systems is how they will handle concurrency. In particular, how can you handle record and replay of data races? Actually recording and replaying races will be overly expensive, so what can you do?

1. SMP-Revirt [4]: use page-protections!
2. DeLorean [7]: use hardware!
3. ODR [1]/PRES [11]: treat replay as search!
4. Arnold [3]/Castor [6]: Races are bugs! No need to support!

Why might we record an execution? There have been a few major reasons and techniques explored in the literature today:

1. Debugging [10]
2. Fault Tolerance [2]
3. Online analysis [9, 12]
4. Intrusion Analysis [5]
5. Workload Capture [8]

Record and replay is a super important approach in the research community for many of these purposes.

References

[1] Gautam Altekar and Ion Stoica. Odr: Output-deterministic replay for multicore debugging. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP '09, page 193–206, New York, NY, USA, 2009. Association for Computing Machinery.

[2] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, SOSP '95, page 1–11, New York, NY, USA, 1995. Association for Computing Machinery.

[3] David Devecsery, Michael Chow, Xianzheng Dou, Jason Flinn, and Peter M. Chen. Eidetic systems. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, page 525–540, USA, 2014. USENIX Association.

[4] George W. Dunlap, Dominic G. Lucchetti, Michael A. Fetterman, and Peter M. Chen. Execution replay of multiprocessor virtual machines. In *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '08, page 121–130, New York, NY, USA, 2008. Association for Computing Machinery.

[5] Ashlesha Joshi, Samuel T. King, George W. Dunlap, and Peter M. Chen. Detecting past and present intrusions through vulnerability-specific predicates. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP '05, page 91–104, New York, NY, USA, 2005. Association for Computing Machinery.

[6] Ali José Mashtizadeh, Tal Garfinkel, David Terei, David Mazieres, and Mendel Rosenblum. Towards practical default-on multi-core record/replay. *SIGPLAN Not.*, 52(4):693–708, apr 2017.

[7] Pablo Montesinos, Luis Ceze, and Josep Torrellas. Delorean: Recording and deterministically replaying shared-memory multiprocessor execution ef?ciently. *SIGARCH Comput. Archit. News*, 36(3):289–300, jun 2008.

[8] Satish Narayanasamy, Cristiano Pereira, Harish Patil, Robert Cohn, and Brad Calder. Automatic logging of operating system effects to guide

application-level architecture simulation. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '06/Performance '06, page 216–227, New York, NY, USA, 2006. Association for Computing Machinery.

[9] Edmund B. Nightingale, Daniel Peek, Peter M. Chen, and Jason Flinn. Parallelizing security checks on commodity hardware. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIII, page 308–318, New York, NY, USA, 2008. Association for Computing Machinery.

[10] Robert O'Callahan, Chris Jones, Nathan Froyd, Kyle Huey, Albert Noll, and Nimrod Partush. Engineering record and replay for deployability. In *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '17, page 377–389, USA, 2017. USENIX Association.

[11] Soyeon Park, Yuanyuan Zhou, Weiwei Xiong, Zuoning Yin, Rini Kaushik, Kyu H. Lee, and Shan Lu. Pres: Probabilistic replay with execution sketching on multiprocessors. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP '09, page 177–192, New York, NY, USA, 2009. Association for Computing Machinery.

[12] Kaushik Veeraraghavan, Dongyoon Lee, Benjamin Wester, Jessica Ouyang, Peter M. Chen, Jason Flinn, and Satish Narayanasamy. Doubleplay: Parallelizing sequential logging and replay. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, page 15–26, New York, NY, USA, 2011. Association for Computing Machinery.