A Retrospective on: A "Flight Data Recorder" for Enabling Full-system Multiprocessor Deterministic Replay

Min Xu[1], Rastislav Bodik[2], Mark D. Hill[3]

[1]*Unaffiliated* [2]*Google & University of Washington* [3]*Microsoft & University of Wisconsin-Madison*

I. Context

During 2001-02, junior graduate student Min Xu was searching for a problem to work on for his PhD. With his advisors, Prof. Hill (computer architecture) and Prof. Bodik (compilers), we wanted a problem that people would care about if we solved it and we had some talent toward solving it.

We focused on **changes** that might expose new opportunities, as Hill later articulated [1]. In the early 2000s, we saw that three changes were underway:

1) Multicore processors were replacing uniprocessors, implying to us that multithreaded programming would become more important.

2) A new class of programming tools were appearing, including race detectors and reverse-execution debuggers. These tools consumed considerable compute cycles to provide debugging insights. It implied that a programmer's time was increasingly more valuable than compute cycles.

3) Moore's Law was providing bountiful more transistors, leading us to expect that some transistors might be available for "value adds" beyond improving processor performance.

We noted that one value that multicore processors removed was deterministic debugging. Since the dawn of computing, (single-threaded) programs could be debugged using `print` statements or a debugger - **deterministically**. Programs with nondeterministic inputs could be debugged by recording the timing and content of inputs. However, multithreaded programs are nondeterministic due to the interactions of threads in shared memory leading to "Heisenbugs" that may or may not be reproducible. Software debuggers were practical only for programs without data races [2]; more general software record & replay approaches appeared to have prohibitive overhead. We thus hypothesized that hardware might help.

II. The work: Flight Data Recorder (FDR)

A. High level idea and insights

The high level idea was adding modest hardware to collect a (distributed) data log to **record** sufficient multithreaded execution information to enable a deterministic **replay** (either in software or hardware) of the original execution regardless of data races. We called our hardware component a **flight data recorder** (FDR). Like the black box on airplanes, FDR allows one to reconstruct what happened. FDR proved to be a memorable name.

Four insights aided the development of FDR. First, threads only interact on multiprocessor cache coherence events[1], not all loads and stores. This is fortunate as we were experts on coherence. Second, conceiving FDR required us to think "globally" among processor cores, not just within a core as was common for hardware structures, and reason about partial orders amongst instructions within a thread and between threads. We have experiences with both from our prior memory consistency model work [3]. Third, like the second at a different level, we focused on understanding the global temporal behavior as in our critical path work [4]. Fourth, local hardware mechanisms could be used to enable a seemingly sophisticated global algorithm. In particular, with sufficient bookkeeping, FDR hardware did not need to explicitly instantiate graph data structure as common in some software schemes.

[1] We ignore thread interleaving on the same processor core because they can be replayed with determinism on OS interrupts and schedulers.

B. FDR for sequential consistency (SC) model

The core of the FDR paper is about how to augment a multiprocessor system to record minimal log data to capture the total order of all thread's memory accesses, as permitted by the SC memory model. This order is sufficient to enable deterministic replay. **FDR's key insight is that memory accesses from different threads interact primarily via cache coherence messages.** To this end, FDR (a) adds modest information to coherence messages, (b) uses a transitivity reduction algorithm to avoid recording all-but-a-few dependencies between threads and (c) adds a small amount of hardware to compute such dependencies in a sufficient but not absolutely minimal fashion.

C. FDR for total store order (TSO) consistency model

Later we extended FDR to the TSO memory consistency model and improved its logging algorithm by introducing artificial but correct dependency arcs to further reduce the log size [5]. For TSO, we use a hybrid dependency+value recording approach to help the replay process to recreate the original program behavior with only a partial order between threads and a few additional values needed by out-of-order load instructions. Both works are summarized in a Top Picks paper [6].

III. Reflections in 2023

Academic computer architecture work seeks impact on industrial products, especially hardware, and influence on the future literature. To the best of our knowledge, FDR's impact on multicore processor products has been minimal so far. We suspect that this is in part because its value proposition for those designing and selling multicore chips is indirect: the chip vendors sell to system vendors who sell to customers who buy applications written by programmers whose debugging could be aided by FDR. This situation could change as cloud service providers design their own processors to run their own software. In a hardware/software codesign context, we can imagine a future FDR-like hardware that implements deterministic record and replay for either debugging or fault tolerance for extreme mission critical applications.

FDR's literature impact has been more substantial, as demonstrated by over 500 citations[2], in part because FDR showed that efficient hardware support for record-replay was possible. We are impressed by the creativity of subsequent work and discuss a small subset here. DeLorean [7] and Rerun [8] leveraged transactional memory concepts to reduce both hardware and log size. DMP [9] and Calvin [10] went beyond deterministic replay to make all executions deterministic. ODR [11] creatively points out that programmers ultimately care only about the output values. ODR uses this insight to shift cost from recording to replay time by using a solver to reconstruct necessary information not recorded.

Finally, more personally, one of us (Xu), who has spent the last 20 years in the industry, appreciates the profound impact of working on FDR. In particular, FDR showed the importance of better algorithms to make seemingly impossible things possible.

References

[1] Mark Hill, "Increasing Your Research Impact", Computer Architecture Today, https://www.sigarch.org/increasing-your-research-impact/, Aug 12, 2019.

[2] Michiel Ronsse, Koen De Bosschere. "Recplay: a fully integrated practical record/replay system", ACM Trans. Comput. Syst., 17(2):133-152, 1999

[3] Sarita V. Adve, Mark D. Hill, "Weak Ordering - A New Definition", in *Proc. ISCA'90*, p.2-14

[4] Brian Fields, Shai Rubin, Rastislav Bodik, "Focus processor policies via critical path prediction", in *Proc. ISCA'01*, p74-85

[5] Min Xu, Rastislav Bodik, Mark D. Hill, "A regulated transitive reduction (RTR) for longer memory race recording", in *Proc. ASPLOS XII*, p49-60

[6] Min Xu, Rastislav Bodik and Mark D. Hill, "A Hardware Memory Race Recorder for Deterministic Replay," in *IEEE Micro*, vol. 27, no. 1, p48-55

[7] Pablo Montesinos, Luis Ceze, Josep Torrellas, "DeLorean: recording and deterministically replaying shared-memory multiprocessor execution efficiently", in *Proc. ISCA'08*, p289-300

[8] Derek R. Hower, Mark D. Hill, "Rerun: exploiting episodes for lightweight memory race recording", in *Proc. ISCA'08*, p256-276

[9] Joseph Devietti, Brandon Lucia, Luis Ceze, Mark Oskin, "DMP: deterministic shared memory multiprocessing", in *Proc. ASPLOS XIV*, p85-96

[10] Derek Hower, Polina Dudnik, Mark D. Hill, David A. Wood, "Calvin: Deterministic or not? Free will to choose", in *Proc. HPCA'01*, p333-344

[11] Guatam Altekar, Ion Stoica, "ODR: output-deterministic replay for multicore debugging", in *Proc. SOSP'09*, p193-206

[2] Over 500 papers citing FDR per Google Scholar: https://scholar.google.com/scholar?oi=bibs&hl=en&cites=12333304558259539546&as_sdt=5