### Fast, Flexible, and Comprehensive Bug Detection for Persistent Memory Programs

Bang Di¹, Jiawen Liu², Hao Chen¹, and Dong Li²

¹Hunan University
²University of California, Merced

#### Introduction

- Developing a crash-consistent PM application is challenging.
- Bugs are easily caused by missing data durability or violating ordering guarantee.
- PM debuggers are often slow, which makes bug detection infeasible.
  - Pmemcheck: 218x slowdown (two hours).
  - XFDetector: 1000x slowdown (nine hours).
- Some debuggers such as PMTest are fast but have low bug coverage.

#### Characterization of PM Programs

- We study store, writeback and fence instructions, and characterize how they are interleaved and distributed.
- **Pattern 1**: for most stores, the data durability is guaranteed by the nearest fence.
- **Inspiration from pattern 1**: tree re-organization in the traditional bookkeeping has few opportunities to gain performance benefit, because information of most stores is deleted in a short term.
- **Pattern 2**: stores in a writeback interval are highly likely to be flushed together by the same single writeback.
- **Inspiration from pattern 2**: it is promising to collectively maintain and update persistency status of memory locations for high performance.
- **Pattern 3**: store happens much more frequently than writeback and fence.
- **Inspiration from pattern 3**: we must efficiently process store instructions.

#### Design of PMDebugger

- PMDebugger instruments memory store, writeback and fence instructions.
- **A hybrid PM-aware data structure**. PMDebugger uses a combination of an array (named memory location array) and an AVL tree.
- The memory location array consists of metadata and memory location information.
- The memory location information is collected from store instructions.
- The metadata maintains the information for writeback intervals.

#### Processing instructions.

- The key idea is to use the metadata to collectively maintain memory location information in the array.
- **Supporting relaxed persistency models**.
  - Epoch model: add a flag to indicate if the store is in an epoch section.
  - Strand model: set up our hybrid PM-aware data structure for each strand section.
- **Detecting bugs**. PMDebugger automatically checks memory location information according to bug-checking rules.

#### Evaluation

- We characterize PM programs and get three PM program patterns, then we introduce a fast, flexible and comprehensive PM debugger based on these PM program characterizations.
- Compared with XFDetector and Pmemcheck, PMDebugger leads to 49.3x and 2.5x speedup; compared with PMTest, PMDebugger has comparable performance, without heavily relying on programmer annotations, and detects 38 more bugs.
- PMDebugger finds 21 new bugs in PMDK and Memcached.

#### Contribution Summary

- PMDebugger instruments memory store, writeback and fence instructions.
- The memory location array consists of metadata and memory location information.
- The memory location information is collected from store instructions.
- The metadata maintains the information for writeback intervals.

- **Performance**.
  - Compared with Pmemcheck: 2.5x speedup.
  - Compared with XFDetector: 49.3x speedup.
  - Compared with PMTest: less than 2x slowdown but more bug coverage.

- **Bug detection capability**. Using a dataset containing 78 bugs to evaluate.
  - Pmemcheck detects 55 bugs; PMTest detects 61 bugs; PMDebugger detects 78 bugs.
  - **New bugs**. PMDebugger finds 21 new bug and these bugs were not reported before by state-of-the-arts.
  - 19 new bugs in Memcached.
  - 2 new bugs in PMDK.

- New bugs. PMDebugger finds 21 new bugs in PMDK and Memcached.

*The full paper of this poster is accepted into ASPLOS'21*

Artifact is available at: https://github.com/PASAUUCMerced/PMDebugger