PMTest: A Fast and Flexible Testing Framework for Persistent Memory Programs

Sihang Liu, University of Virginia
Yizhou Wei, University of Virginia
Jishen Zhao, UC San Diego
Aasheesh Kolli, Penn State University, VMware Research
Samira Khan, University of Virginia

Abstract

Recent non-volatile memory technologies such as 3D XPoint and NVDIMMs have enabled persistent memory (PM) systems that can manipulate persistent data directly in memory. This advancement of memory technology has spurred the development of a new set of crash-consistent software (CCS) for PM - applications that can recover persistent data from memory in a consistent state in the event of a crash (e.g., power failure). CCS developed for persistent memory ranges from kernel modules to user-space libraries and custom applications. However, ensuring crash consistency in CCS is difficult and error-prone. Programmers typically employ low-level hardware primitives or transactional libraries to enforce ordering and durability guarantees that are required for ensuring crash consistency. Due to the reordering by the hardware, programmers cannot *test* whether the order specified in the CCS will *not* result in an ordering that *violates* the crash consistency requirement.

We believe that there is an urgent need for developing a testing framework that helps programmers identify crash consistency bugs in their CCS. We find that prior testing tools lack generality, i.e., they work only for one specific CCS or memory persistency model and/or introduce significant performance overhead. To overcome these drawbacks, we propose PMTest[1], a crash consistency testing framework that is both flexible and fast. PMTest provides flexibility by providing two basic assertion-like software checkers to test two fundamental characteristics of all CCS: the ordering and durability guarantee. These checkers can also serve as the building blocks of other application-specific, high-level checkers. PMTest enables fast testing by deducing the persist order without exhausting *all* possible orders. In the evaluation with eight programs, PMTest not only identified 45 synthetic crash consistency bugs, but also detected 3 new bugs in a file system (PMFS) and in applications developed using a transactional library (PMDK), while on average being 7.1× faster than the state-of-the-art tool.

[1] PMTest is available at https://pmtest.persistentmemory.org.

ASPLOS '19, April 13–17, 2019, Providence, RI, USA

ACM ISBN 978-1-4503-6240-5/19/04…$15.00
https://doi.org/10.1145/3297858.3304015

CCS Concepts • **Hardware → Emerging technologies**; • **Software and its engineering → Software testing and debugging**.

Keywords Persistent Memory, Crash Consistency, Debugging, Testing

ACM Reference Format:
Sihang Liu, Yizhou Wei, Jishen Zhao, Aasheesh Kolli, and Samira Khan. 2019. PMTest: A Fast and Flexible Testing Framework for Persistent Memory Programs. In *2019 Architectural Support for Programming Languages and Operating Systems (ASPLOS '19), April 13–17, 2019, Providence, RI, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3297858.3304015

1 Introduction

Persistent Memory (PM) technologies offer the persistence of disks combined with performance close to that of DRAM, blurring the divide between memory and storage [34, 40, 45, 67]. PMs are expected to be placed alongside DRAM on the system's memory bus and be accessed via a byte-addressable load/store interface, providing an opportunity to manipulate persistent data directly in-place in memory. Programs can recover their updated in-memory persistent data even in the event of a crash (e.g., power failure). However, such a recovery requires a guarantee that persistent data is always in a consistent state – a requirement referred to as the crash consistency guarantee. A variety of applications have taken crash consistency into consideration. File systems carefully orchestrate meta-data management to ensure that the files are recoverable [10, 16, 42, 63, 66, 73], while databases use intricate logging mechanisms to provide ACID guarantees for transactions [1, 2, 23, 50, 65]. Apart from relying on file systems and databases for crash consistency [1, 2, 10, 16, 23, 42, 50, 63, 65, 66, 73], the advent of PMs makes it possible for applications to manage crash consistency directly using PM's load/store interface and thereby, improve performance by avoiding costly system calls. For this reason, a variety of custom crash-consistent applications [7, 17, 33, 70, 72] and user-space libraries (e.g., NV-Heaps [9], Mnemosyne [64], PMDK [33]) have been developed for PM systems. Moving forward, we expect that