SSP: Eliminating Redundant Writes in Failure-Atomic NVRAMs via Shadow Sub-Paging

Yuanjiang Ni
UC Santa Cruz

Jishen Zhao
UC San Diego

Heiner Litz
UC Santa Cruz

Daniel Bittman
UC Santa Cruz

Ethan L. Miller
UC Santa Cruz
Pure Storage

ABSTRACT

Non-Volatile Random Access Memory (NVRAM) technologies are closing the performance gap between traditional storage and memory. However, the integrity of persistent data structures after an unclean shutdown remains a major concern. Logging is commonly used to ensure consistency of NVRAM systems, but it imposes significant performance overhead and causes additional wear out by writing extra data into NVRAM. Our goal is to eliminate the extra writes that are needed to achieve consistency. SSP (i) exploits a novel cache-line-level remapping mechanism to eliminate redundant data copies in NVRAM, (ii) minimizes the storage overheads using page consolidation and (iii) removes failure-atomicity overheads from the critical path, significantly improving the performance of NVRAM systems. Our evaluation results demonstrate that SSP reduces overall write traffic by up to 1.8×, reduces extra NVRAM writes in the critical path by up to 10× and improves transaction throughput by up to 1.6×, compared to a state-of-the-art logging design.

CCS CONCEPTS

• **Information systems** → *Phase change memory*; • **Computer systems organization** → *Reliability*.

KEYWORDS

shadow sub-paging, NVRAM, failure-atomicity

ACM Reference Format:
Yuanjiang Ni, Jishen Zhao, Heiner Litz, Daniel Bittman, and Ethan L. Miller. 2019. SSP: Eliminating Redundant Writes in Failure-Atomic NVRAMs via Shadow Sub-Paging. In *Proceedings of The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'19).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3352460.3358326

MICRO'19, October 12-16, 2019, Columbus, OH, USA

ACM ISBN 978-1-4503-6938-1/19/10…$15.00
https://doi.org/10.1145/3352460.3358326

1 INTRODUCTION

Non-Volatile Random Access Memory (NVRAM) is becoming a reality, as technologies such as STT-RAM [20], PCM [39] (Phase-Change Memory) and memristors [45] show DRAM-like performance and disk-like persistence. Recently, Intel has announced its Optane DC persistent memory [17], further facilitating the acceptance of NVRAM as a new storage tier. NVRAM on the memory bus ("persistent memory") introduces a new storage interface: applications use CPU load/store instructions to directly operate on the storage medium. This model removes the need to maintain separate data formats for memory and storage. The resulting programs are streamlined and reduce overheads, for instance, by avoiding serialization and deserialization of the in-memory data structures.

Since data in persistent memory survives a power cycle, this model requires mechanisms to ensure that persisted data is reusable by preserving application consistency in the presence of failures. Durable transactions provide a straightforward abstraction to support consistency—programmers only need to specify the code that should be part of a failure-atomic section. The updates within the failure-atomic section are guaranteed to be executed indivisibly (all or nothing).

Prior work on providing failure-atomicity for persistent memory [4, 5, 14, 18, 19, 44, 49] has focused on the following challenges: First, the limited write endurance [23] of NVRAM compared to DRAM, second, the latency overheads introduced for guaranteeing failure-atomicity, by memory fences, flushes and logging and third, the inability to amortize overheads when operating on byte addressable NVRAM. Prior work has approached the above challenges with three techniques: write-ahead logging, log-structuring and shadow paging.

Write-ahead logging [4, 9, 16, 49] ensures storage consistency with explicit data copying. Several previous works [4, 9, 16, 49] employ fine-grained logging to avoid unnecessary memory bandwidth consumption. Unfortunately, logging, even at finest granularity, still causes redundant writes that introduce bandwidth overheads [54] and cause additional wear out. Log-structured stores only maintain a single copy of a data element, but they must adjust a mapping table on each write that references the new element. Furthermore, the log-structured approach usually maintains mappings to data elements of large size to reduce the capacity overhead of the mapping table in comparison to the actual data. Unfortunately, this introduces fragmentation, requires large page writes and introduces garbage collection overheads, similar to those incurred by NAND Flash [24]. It is required to use a more flexible, and hence complex, mapping scheme [14] to reduce the otherwise prohibitive metadata