Vorpal: Vector Clock Ordering For Large Persistent Memory Systems

Kunal Korgaonkar
kkorgaon@eng.ucsd.edu
UC San Diego

Joseph Izraelevitz
jizraelevitz@eng.ucsd.edu
UC San Diego

Jishen Zhao
jzhao@eng.ucsd.edu
UC San Diego

Steven Swanson
swanson@eng.ucsd.edu
UC San Diego

ABSTRACT

In systems with non-volatile main memories (NVMMs), programmers must carefully control the order in which writes become persistent. Otherwise, what will remain in persistence after a crash may be unusable upon recovery. Prior art has already explored semantic models for specifying this persist order, but most enforcement algorithms for the order are not scalable to large server machines because they assume that the machine contains only one or two memory controllers. In this paper, we describe a collection of provably-correct algorithms for enforcing the persist-order across writes, generated at many different cores, and persisted across numerous different memory controllers. Relative to existing solutions, our algorithms improve performance by 48% by reducing both traffic and serialization overheads.

ACM Reference Format:
Kunal Korgaonkar, Joseph Izraelevitz, Jishen Zhao, and Steven Swanson. 2019. Vorpal: Vector Clock Ordering For Large Persistent Memory Systems. In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29-August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3293611.3331598

1 INTRODUCTION

Non-volatile main memories (NVMMs), such as phase-change memory (PCM) [46], resistive RAM (ReRAM) [38], and 3D Xpoint [31], are likely to bring profound changes to many aspects of computer systems. Like DRAM, these devices are *byte-addressable*, and programs interact with them using the traditional load/store (read-/write) interface. Unlike DRAM, but like disk, these devices are *non-volatile* — their contents are guaranteed to survive a power outage. As such, NVMMs are poised to provide a medium for fast, durable storage. Given their non-volatile property, NVMM are also referred to as *persistent* memories.

Unfortunately, from the perspective of durable storage, NVMMs are somewhat undermined by the fact that processor caches and registers lose their contents on power failure. Consequently, barring additional effort, data in NVMMs is likely to be unusable after a power failure because processor caches and registers lose their contents on power failure — what will survive in NVMM is that data that was not held within the cache hierarchy at the time of failure. As a consequence, NVMMs require programs to carefully control the order in which their writes reach persistent memory. A failure in the middle of a linked-list insertion, for example, may lead to a post-crash dangling reference, if the "next" pointer of the predecessor node is written into persistent memory while the pointed (inserted) node itself remains in a cache.

To give programmers a tool to manage the order of writes to persistent memory, modern ISAs provide a *memory persistency model*, which is a set of instructions that control the order in which writes to NVMM become persistent, such that they will be visible after a power failure [35]. The memory persistency model is specified in conjunction with of the similar, yet, different, *memory consistency model*, which describes the order in which concurrent threads' updates become visible to each other. These models are generally orthogonal (two ISAs might implement the same persistency model on top of different consistency models), but their interactions are generally tightly coupled — a persistency model might leverage the memory consistency model to induce orderings.

A growing body of research [20–23, 32, 35, 40, 41] explores ISAs and supporting micro-architectures to enforce these ordering constraints. Despite this previous work, scalability of persistence ordering support still remains an issue for NVMMs. Most previous work only considers the ordering of NVMM writes through one or two memory controllers [4, 20–23, 32, 40], but modern servers can include many memory controllers scattered across multiple sockets [15]. Enforcing ordering constraints between writes generated at many different cores and written into NVMM (that is, *persisted*) at many different memory controllers requires an efficient algorithm to capture and enforce these constraints.

In this paper, we propose a set of novel distributed algorithms for ordering writes to persistent memory on a large, shared-memory computer system. Our collection of algorithms, which we call Vector ORdered Persistence ALgorithms (Vorpal), uses a vector clock [12, 30] representation to (1) construct a partial order over the writes to NVMM and (2) efficiently enforce their order at multiple memory controllers.

In particular, this paper makes the following contributions:

- We describe a basic vector clock algorithm for enforcing write order to persistent memories and demonstrate that it can implement the acquire-release memory persistency (ARP) model, defined in Section 2.2.3.
- We address the scalability problems in vector clocks as a write ordering mechanism in cache-coherent, shared-memory

This work was supported in part by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. We thank our shepherd, Marc Shapiro, for his valuable guidance in writing.

PODC '19, July 29-August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07…$15.00
https://doi.org/10.1145/3293611.3331598