WrAP: Write Aside Persistence for Storage Class Memory in High Performance Computing

Ellis R. Giles
Rice University
ACM Student Member

Peter Varman
(Advisor)
Rice University

Kshitij Doshi
(Advisor)
Intel Corporation

***Abstract*—Many I/O-intensive High Performance Computing applications like Map Reduce and database systems are incorporating in-memory computing technology to overcome traditional storage bandwidth bottlenecks. The volatile nature of DRAM makes these systems vulnerable to system crashes.**

Software based Write Aside Persistence is presented that provides atomic durability and consistency for persistent, byte-addressable memory writes, while ensuring fast paths to data in processor caches, DRAM, and persistent memory tiers.

I. Introduction and Problem Overview

This research examines the use of emerging byte-addressable persistent memory (also called Storage Class Memory or SCM) as a replacement for traditional non-volatile storage in emerging data intensive applications on high performance computing systems. A growing number of applications use in-memory database technologies that operate almost entirely from DRAM (*e.g.* SAP HANA, IBM solidDB, voltDB, and Neo4J). DRAM-based data access has advantages: high throughputs and real-time response times and fine-grained memory word (or cache-line) accessibility. The volatile nature of DRAM makes persistent applications vulnerable to system crashes, often requiring lengthy ad-hoc checkpointing techniques to maintain a persistent copy of the data on non-volatile storage. Figure 1 shows how SCM technology combines the fast, cache-line granularity access of DRAM with the persistence of disk.

The problem addressed is ensuring a sequence of stores to arbitrary SCM addresses is performed atomically, even in the presence of machine failure. Many problems arise such as making sure that memory writes are actually written to SCM and spurious cache evictions before a transaction commit do not corrupt data structures.

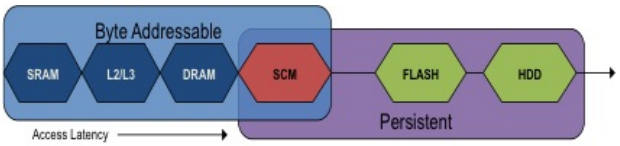

Fig. 1. Fast, Byte-Addressable SCM

II. Write Aside Persistence

Software based Write Aside Persistence is presented as a lightweight library for supporting atomic writes to storage class memory similar to a hardware based solution.

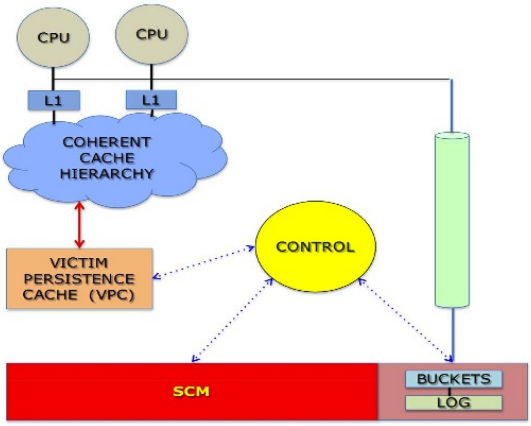

Fig. 2. Hardware WrAP

In a hardware WrAP shown in Figure 2, the memory architecture is augmented with a victim cache that catches persistent memory cache evictions from the cache hierarchy [1]. A WrAP controller is added that manages a victim cache and

SCM based log area. When data is written to SCM, it is streamed to the log and, potentially cached, home memory location.

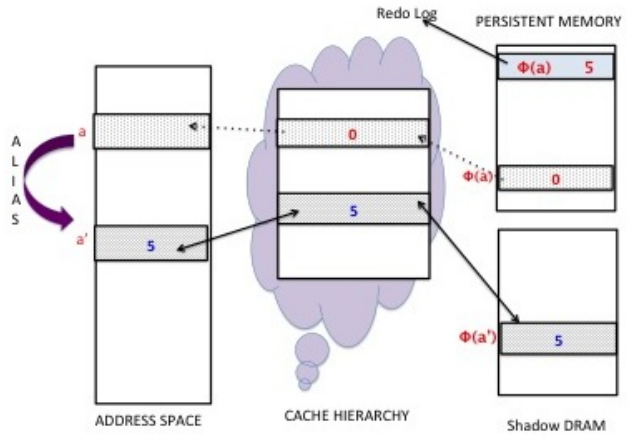

Fig. 3. Software WrAP and Alias Table

In the presented software version of WrAP, a DRAM based software Alias Table is used which prevents cache evictions to the original SCM memory location. The software version also has streaming writes to persistent log locations, but instead of writing data to the original SCM location, data is written into a fast DRAM based Alias Table. A WrAP read, reads data from the Alias Table if the entry is in the table or from the original location otherwise. On a WrAP close, data can be written to the main SCM location from the log location or the Alias Table if the data hasn't been modified by other threads. Optionally, the Alias Table may be purged asynchronously. Figure 3 shows the software based WrAP approach.

III. Selected Results

For evaluation, I examined an array update speedup of the SoftWrap implementation over a traditional Copy-On-Write approach using a PIN based simulator. A SCM slowdown factor of four over DRAM was used with a varying number of transactional write operations per wrap. The total number of system cycles over an average of ten runs was recorded and compared to the Copy-On-Write approach. The results for random, cache line and elemental sequential are shown in Figure 4.

Graph data structure creation was also examined in the Graph 500 Benchmark. The graph data structure is allocated in persistent memory using the

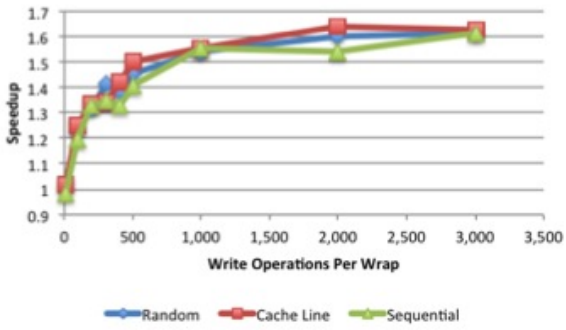

Fig. 4. Speedup for SoftWrap for Various Array Update Methods

pmalloc call in the WrAP API. Every access to the Graph 500 data structure in SCM is accessed through WrAP. Creation time is shown in Figure 5. SoftWrAP is twice as fast for reliable graph construction and node insertions when compared to a Copy-On-Write approach.

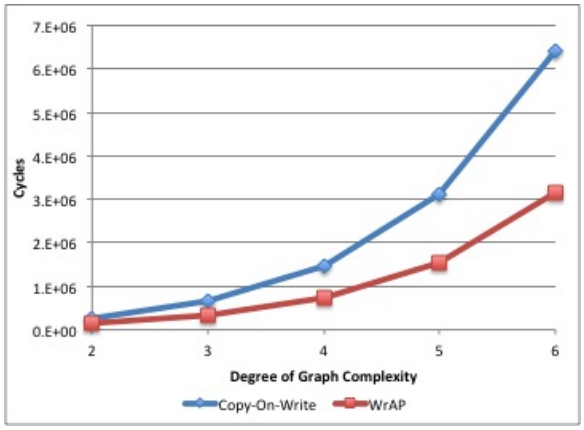

Fig. 5. Graph 500 Benchmark Construction Times

IV. Conclusions

A software based WrAP utilizes a fast DRAM based Alias Table for SCM atomic persistence. It shows significant performance gains over traditional Copy-On-Write, or UnDo log, based approaches. WrAP allows for byte addressability and atomic persistence of SCM devices while benefitting from caching without slow, front end synchronous operations.

References

[1] Giles, E., Doshi, K., and Varman, P. Bridging the programming gap between persistent and volatile memory using WrAP. In *ACM Computing Frontiers* (2013).