XPS:EXPL:FP:Architecture and Software for Scalable Persistent Memory

Peter Varman
Rice University
Houston, Texas
pjv@rice.edu

NSF XPS Workshop, June 1-2, 2015, Arlington, VA
Outline

- Background
- WrAP Architecture
- Hardware Controller Approach
- Software Approach (SoftWrAP)
Background

Storage Class Memory (SCM):
- Byte-Addressable
- High-Speed
- Density
- DRAM form factors
- No Passive Power Consumption
- Persistent

- Types:
  - Memristor
  - Magnetic Spin
  - Phase Change Memory (PCM)

- Supports New Applications (no longer require slow, block-based persistence)

Adapted from ISCA 2009 (Qureshi et al)
Background

• Supports New Data Management Applications

  • In Memory Databases (IMDB) & Caches
    – CSQL, Volt DB, SAP HANA and Memcached
    – Fast Access Speeds
    – Low Passive Power Consumption
    – Persistence
    – Fast Crash Recovery

  • Support for Unstructured Pointer Based Data
    – New Graph Databases (Neo4J)

• Single programmer interface
  – Blurs distinction between in-memory and disk-based data
Durability Problem

• Durability of writes (even a single write) not guaranteed

• A=1 is only in cache hierarchy and not in memory
Durability Problem

- Durability of writes are not guaranteed.
- A=1 is now only in a store buffer and not memory.
- Unfortunate side effect of also invalidating cache entry.
Durability Problem

Fortunately, 2/2015 Intel ISA Manual introduces:

- **CLWB** – write back
- **PCOMMIT** – persistent commit - $$$

Durability of a store, A=1 now in persistent memory.

Problem solved?

What about multiple stores?
Failure Atomicity

- Sequence of coupled writes that must be updated as a group
  - Pointer changes to a linked data structure

- Failure can leave data structure in SCM in inconsistent state
Another problem

Cache evictions can leave SCM in inconsistent state.
• Background

• WrAP Architecture

• Hardware Controller Approach

• Software Approach (SoftWrAP)
Write Aside Persistence (WrAP)

- Propagate updates along two paths
- Use cache hierarchy for value communication
- Log updates to SCM as write-combined streaming stores
- Avoid fine-grained synchronous front-end operations
  - **Hardware Wrap**: Victim Cache to hold cache spills
  - **SoftWrAP**: Alias SCM locations to DRAM locations
• Background
• WrAP Architecture
• Hardware Controller Approach
• Software Approach (SoftWrAP)
WrAP Architecture

Write Aside Persistence

- **Victim Persistence Cache**
  - Contains evicted entries from open wraps

- **Background Logging Path**
  - SCM Address/Value Pairs
  - Log in persistent memory
  - Streaming stores

```
wrapOpen();
*X=5; *Y=7;
wrapClose();
```
Hardware Controller

Accepted -- Computer Architecture Letters (2015)
Outline

- Background
- WrAP Architecture
- Hardware Controller Approach
- Software Approach (SoftWrAP)
**SoftWrAP Approach**

- **Aliasing**
  
  SCM locations aliased to DRAM locations

**wrapOpen()**
Creates Log

**wrapStore(x, val)**
Streams location x and value to log
Writes x to Alias Table

**wrapLoad(x)**
Load x from alias table or SCM if not present

**wrapClose()**
Close log & PCOMMIT
// Can process table.
STORE a, 5
STORE a', 5
Append to Log
## Alias Table

### Hash Table A
State: Retiring

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>M</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Z</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>N</td>
<td></td>
<td>1024</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Hash Table B
State: Active

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>X</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>Y</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Z</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Hash(W)

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Data Object

#### Hash Table A
- State: Retiring

#### Hash Table B
- State: Active

- Double Buffered (2 Hash Tables)
- Concurrent Retirement
- Supports primitives and object types
- Reads check both tables
- 5 table states.
- Locks only on state change and openWrap.


• Giles E., Doshi K., Varman P., “Free Atomic Consistency in Storage Class Memory Using Software-Based Write-Aside Persistence”, (Short Paper + Poster) CF’15, Ischia, Italy

Related Work

Transaction Support for Storage Class Memory:

• Battery Backup Based Work
  – Whole System Persistence

• Recoverable Memory Techniques

• Up-Front Cache Line Changes & Data Copying
  – BPFS (epoch barriers on cache eviction, copy-on-write)
  – NV-Heaps (Logging)
  – Kiln (NV-Victim Cache provides transaction buffering; 2 phase commit)

• Software Control Layer and Compiler
  – Mnemosyne (STM based interception of all reads and writes; undo)
  – ATLAS (Automatically generates atomic regions; Undo Log)

• Software Solutions and Versioning
  – Consistent Durable Data Structures (versioning and garbage collection)
  – REWIND (In-place updates; atomic double linked list)
Summary

- Storage Class Memories are an exciting new area of research that will give rise to new software applications and architectures.

- WrAP Architecture presented is a fast, straightforward approach to ensure transactional support in writing byte-addressed persistent data.

- Scaling to distributed storage

- Wrapping complete applications