Hardware Support for ACID Transactions in Persistent Memory

Arpit Joshi, Vijay Nagarajan, Marcelo Cintra, Stratis Viglas

NVMW 2019
Persistent Memory Systems

L1 → LLC → Persistent Memory
Persistent Memory Systems

- Persistent Memory
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data
Persistent Memory Systems

- Persistent Memory
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

System Crashes
Persistent Memory Systems

• Persistent Memory
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

• Crash Consistency
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions
Persistent Memory Systems

- Persistent Memory
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

- Crash Consistency
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

“Ensuring failure atomicity for all this computation without failure-atomic transactions is practically infeasible, if not impossible.”

Marathe et al. [HotStorage’17]
Persistent Memory Systems

- Persistent Memory
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data
- Crash Consistency
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

“Ensuring failure atomicity for all this computation without failure-atomic transactions is practically infeasible, if not impossible.”

Marathe et al. [HotStorage’17]

How fast can we support ACID?
ACID Transactions

L1

LLC

Persistent Memory
ACID Transactions

Atomic Visibility

L1

LLC

Persistent Memory
ACID Transactions

Persistent Memory

Atomic Durability

Atomic Visibility
ACID Transactions

Persistent Memory

L1

LLC

L1

Atomic Visibility

Locks

STM

HTM

Atomic Durability
ACID Transactions

Persistent Memory

L1

LLC

L1

Atomic Visibility

Locks

STM

HTM

Checkpointing

S/W Logging

H/W Logging

Atomic Durability
ACID Transactions

Persistent Memory

L1 → LLC → L1

Atomic Visibility
- Locks
- STM
- HTM

Atomic Durability
- Checkpointing
- S/W Logging
- H/W Logging

H/W Logging
Atomic Visibility: HTM
Atomic Visibility: HTM

- Commercial HTMs [Intel, IBM]

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>B = 20</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Atomic Visibility: HTM

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
Atomic Visibility: HTM

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
Atomic Visibility: HTM

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
  - **Commit**: make updates non-speculative
Atomic Visibility: HTM

- Commercial HTMs [Intel, IBM]
  - Version Management: read/write sets in L1 cache
  - Conflict Detection: piggy back on the coherence protocol
  - Commit: make updates non-speculative
  - Abort: invalidate write set
• **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
  - **Commit**: make updates non-speculative
  - **Abort**: invalidate write set

> Write-sets in commercial HTMs limited by the size of the L1 cache.
Atomic Durability: Logging
Atomic Durability: Logging

- Logging for durability [Doshi’16, Joshi’17, Shin’17, Ogleari’18]

### Persistent Memory

<table>
<thead>
<tr>
<th>In-place Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 10</td>
</tr>
<tr>
<td>B = 20</td>
</tr>
<tr>
<td>C = 30</td>
</tr>
</tbody>
</table>
Atomic Durability: Logging

- **Logging for durability** [Doshi’16, Joshi’17, Shin’17, Ogleari’18]
  - Write a log entry for every update
Atomic Durability: Logging

- **Logging for durability** [Doshi’16, Joshi’17, Shin’17, Ogleari’18]
  - Write a log entry for every update
  - **Commit**: Update the values in-place

---

<table>
<thead>
<tr>
<th>Persistent Memory</th>
<th>In-place Values</th>
<th>Transaction Log</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A = 15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B = 25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C = 30</td>
<td></td>
</tr>
</tbody>
</table>
Atomic Durability: Logging

- **Logging for durability** [Doshi’16, Joshi’17, Shin’17, Ogleari’18]
  - Write a log entry for every update
  - **Commit**: Update the values in-place
  - **Abort**: Undo any in-place updates
Atomic Durability: Logging

- **Logging for durability** [Doshi’16, Joshi’17, Shin’17, Ogleari’18]
  - Write a log entry for every update
  - **Commit**: Update the values in-place
  - **Abort**: Undo any in-place updates

In-place updates in the critical path of commit
High memory write bandwidth requirement
ACID = HTM + Logging

Goals:
- Support fast commits
- Minimise memory bandwidth consumption
- Extend the supported transaction size
- Maintain the simplicity of commercial HTMs
DHTM: Durable Hardware Transactional Memory
DHTM: Durable Hardware Transactional Memory

Commercial HTM + Hardware Redo Log

Persistence Memory
DHTM: Durable Hardware Transactional Memory

Commercial HTM + Hardware Redo Log

- H/W Redo Log + Log Buffer
  - Reduced memory bandwidth
  - Fast commits

Diagram showing memory hierarchy with L1, LLC, and Persistent Memory.
DHTM: Durable Hardware Transactional Memory

- **Commercial HTM + Hardware Redo Log**
  - H/W Redo Log + Log Buffer
    - Reduced memory bandwidth
  - H/W Log + Sticky State
    - Extended transaction size to the LLC
    - Simplicity of commercial HTM
DHTM: Log Buffer

L1

LLC

Log Writes

Persistent Memory
DHTM: Log Buffer

- Redo Log Bandwidth Problem
DHTM: Log Buffer

- **Redo Log Bandwidth Problem**
  - write a log entry for every store
DHTM: Log Buffer

- Redo Log Bandwidth Problem
  - write a log entry for every store
  - multiple stores create multiple log entries

Persistent Memory
DHTM: Log Buffer

- Redo Log Bandwidth Problem
  - write a log entry for every store
  - multiple stores create multiple log entries

- Solution: Log Buffer
DHTM: Log Buffer

- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries

- **Solution: Log Buffer**
  - track cache lines being modified
DHTM: Log Buffer

- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries

- **Solution: Log Buffer**
  - track cache lines being modified
  - multiple writes coalesced in a log entry
• **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries

• **Solution: Log Buffer**
  - track cache lines being modified
  - multiple writes coalesced in a log entry
  - log entry written to persistent memory on eviction from log buffer
DHTM: Transaction States
DHTM: Transaction States

Begin Transaction → Active
DHTM: Transaction States

- Begin Transaction
- Active
- End Transaction & Log Records Persisted
- Commit
DHTM: Transaction States

Begin Transaction → Active → End Transaction & Log Records Persisted → Commit → In-place Data Persisted → Commit Complete
DHTM: Transaction States

- Active
  - Begin Transaction
  - End Transaction & Log Records Persisted
  - Conflict

- Commit
  - In-place Data Persisted

- Commit Complete
DHTM: Commit Example

**L1 Cache**

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**State**

<table>
<thead>
<tr>
<th>Log Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Persistent Memory**

<table>
<thead>
<tr>
<th>In-place Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 10</td>
</tr>
<tr>
<td>B = 20</td>
</tr>
<tr>
<td>C = 30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transaction Log</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Begin_Transaction**

Write (A=15)
Read (B)
Write (B=25)

**End_Transaction**
DHTM: Commit Example

L1 Cache

Cache Line  R  W

State

Active

Log Buffer

Persistent Memory

In-place Values

Transaction Log

A = 15
B = 25
C = 30

Begin_Transaction
Write (A=15)
Read (B)
Write (B=25)
End_Transaction
DHTM: Commit Example

L1 Cache

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

State

Active

Log Buffer

A

Persistent Memory

In-place Values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 10</td>
<td></td>
</tr>
<tr>
<td>B = 20</td>
<td></td>
</tr>
<tr>
<td>C = 30</td>
<td></td>
</tr>
</tbody>
</table>

Transaction Log

Begin_Transaction

Write (A=15)

Read (B)

Write (B=25)

End_Transaction
DHTM: Commit Example

L1 Cache

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B = 20</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

State

Active

Log Buffer

A

Persistent Memory

In-place Values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 10</td>
<td></td>
</tr>
<tr>
<td>B = 20</td>
<td></td>
</tr>
<tr>
<td>C = 30</td>
<td></td>
</tr>
</tbody>
</table>

Transaction Log

Begin Transaction

Write (A=15)

Read (B)

Write (B=25)

End Transaction
DHTM: Commit Example

L1 Cache

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B = 25</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

State

Active

Log Buffer

B

Persistent Memory

Transaction Log

A = 15

In-place Values

A = 10
B = 20
C = 30

Begin_Transaction

- Write (A=15)
- Read (B)
- Write (B=25)

End_Transaction
DHTM: Commit Example

**L1 Cache**

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>B = 25</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**Persistent Memory**

**In-place Values**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 10</td>
<td>B = 20</td>
<td>C = 30</td>
</tr>
</tbody>
</table>

**Transaction Log**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td></td>
</tr>
<tr>
<td>B = 25</td>
<td>Commit</td>
</tr>
</tbody>
</table>

**State**

**Log Buffer**

**Commit**

**Begin_Transaction**

Write (A=15)

Read (B)

Write (B=25)

**End_Transaction**
DHTM: Commit Example

L1 Cache

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

State

Commit

Complete

Log Buffer

Persistent Memory

In-place Values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td></td>
</tr>
<tr>
<td>B = 25</td>
<td></td>
</tr>
<tr>
<td>C = 30</td>
<td></td>
</tr>
</tbody>
</table>

Transaction Log

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 15</td>
<td></td>
</tr>
<tr>
<td>B = 25</td>
<td></td>
</tr>
<tr>
<td>Commit</td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td></td>
</tr>
</tbody>
</table>

Begin_Transaction

Write (A=15)

Read (B)

Write (B=25)

End_Transaction
DHTM: Supporting Overflow
DHTM: Supporting Overflow

• Problems with Overflow:
DHTM: Supporting Overflow

- Problems with Overflow:
  - Version Management:
    - global operation on write-set on a commit/abort
    - overhead infeasible in larger caches (beyond L1)
DHTM: Supporting Overflow

• Problems with Overflow:
  - Version Management:
    - global operation on write-set on a commit/abort
    - overhead infeasible in larger caches (beyond L1)
  - Conflict Detection:
    - additional metadata to detect conflicts
    - increased complexity due to NACK based protocols
DHTM: Supporting Overflow
DHTM: Supporting Overflow

• Solution
DHTM: Supporting Overflow

• Solution
  - Version Management:
  - Overflow List
DHTM: Supporting Overflow

• Solution
  - Version Management:
    - Overflow List
DHTM: Supporting Overflow

• Solution
  - Version Management:
  - Overflow List
DHTM: Supporting Overflow

• Solution
  - Version Management:
    - Overflow List
  - Conflict Detection:
    - maintain sticky state on overflow (similar to LogTM)
    - avoid NACK by restricting overflow to LLC
Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Atomic Visibility</th>
<th>Atomic Durability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>Locks</td>
<td>Hardware Undo Log</td>
</tr>
<tr>
<td>LogTM+ATOM</td>
<td>HTM (LogTM)</td>
<td>Hardware Undo Log</td>
</tr>
<tr>
<td>DHTM</td>
<td>HTM</td>
<td>Hardware Redo Log (Log Buffer)</td>
</tr>
</tbody>
</table>

• System Configuration
  - We evaluate an 8-core machine with a 2-level cache hierarchy
  - HTM’s implement (first) writer wins conflict resolution policy
Evaluation
Evaluation

- Atom
- LogTM+ATOM
- DHTM

Comparison of different data structures:
- Queue
- Hash
- SDG
- SPS
- BST
- RBT
- GMean
Evaluation

- queue
- hash
- sdg
- sps
- btree
- rbtree
- gmean

![Bar chart showing comparisons between ATOM, LogTM+ATOM, and DHTM for various data structures. The chart includes bars for each structure with values corresponding to their performance metrics.]
Evaluation

The diagram shows the performance comparison of different data structures: queue, hash, sdg, sps, btree, rbtree, and gmean. The performance metrics are evaluated using ATOM, LogTM+ATOM, and DHTM. The graph indicates that DHTM outperforms ATOM and LogTM+ATOM by 26% in terms of efficiency.

Key data structures:
- Queue
- Hash
- SDG
- SPS
- B-tree
- RB-tree
- Gmean
Conclusion

• Persistent memory systems require crash consistency
• ACID Transactions: widely understood crash consistency mechanism
• DHTM: ACID transactions in hardware
  - Atomic Visibility: commercial HTM
  - Atomic Durability: bandwidth optimized hardware redo log
  - Leverage hardware logging to extend transaction size unto LLC
Hardware Support for ACID Transactions in Persistent Memory

Arpit Joshi, Vijay Nagarajan, Marcelo Cintra, Stratis Viglas

NVMW 2019