SPHT: Scalable Persistent Hardware Transactions

Daniel Castro\textsuperscript{1}, Alexandro Baldassin\textsuperscript{2}, Paolo Romano\textsuperscript{1}, João Barreto\textsuperscript{1}

\textsuperscript{1}Instituto Superior Técnico & INESC-ID
\textsuperscript{2}UNESP - Universidade Estadual Paulista
Introduction

- Multithread synchronization
- Transactional Memory
- Hardware Implementations
- Byte-addressable Persistent Memories
- Memory Mapped Files
- Better Performance
- Durability

Simplify Programming

+ Better Performance
Hardware Transactional Memory

Core
Private Cache

Core
Private Cache

Core
Private Cache

Core
Private Cache

Hardware level Conflict Detection

A=1, B=2, C=3
Shared Memory

Hardware Transactional Memory

Persistent Memory

DRAM
A=1, B=2, C=3
volatile

PM
A=1, B=2, C=3
persistent

Core
Private Cache

Core
Private Cache

Core
Private Cache

Core
Private Cache

Last Level of Cache

Volatile

Cache evictions

CLWB, CLFLUSH, WBINVD, ...

PM

Persistent
How to combine HTM and PM?

htm_begin
A ← 1
B ← 2
persist?
htm_end

I’ll focus on this

Existing solutions:
• Shadow memory
• Non-destructive undo log

Better Performance

Hardware Implementations

Byte-addressable Persistent Memories

DudeTM[ASPLOS’17]
c-c-HTM[ISMM’17]
NV-HTM[IPDPS’18]
Crafty[PLDI’20]
Combining HTM and PM

- **Volatile Working Snapshot**
  - HTM transactions
  - recover

- **Persistent Heap**
  - apply

- **Persistent Logs**
  - generate
  - repeat writes

**Commit Logic**

- $T_0$ starts
- HTM exec. read TS
- $T_0$ returns time

**DudeTM** [ASPLOS'17]

**cc-HTM** [ISMM'17]

**NV-HTM** [IPDPS'18]

**SPHT (this paper)**
Durability semantics

- DBMS guarantee ACID: externalized commits are durable
  - what about PTM systems?

**Immediate Durability:**

The application is guaranteed that a transaction \( T \) is durable after the commit logic completes.

All transactions that \( T \) may depend upon must also be durable.
### Scalability limitations of the SoTA

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Clock updated by TXs</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Extended TX vulnerability window</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Sequential mechanism to ensure durability</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Sequential Log Replay</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

#### Limitations at the level of log replay

#### Limitations at the level of transaction processing

- **Workaround:** relax durability semantics ...
  ... not always applicable & more complex
Scalability limitations of the SoTA

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Clock updated by TXs</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Extended TX vulnerability window</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Sequential mechanism to ensure durability</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Sequential Log Replay</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

**Limitations at the level of log replay**

**Limitations at the level of transaction processing**

➢ **Workaround:** relax durability semantics ...

... not always applicable & more complex
SPHT: contributions

• Evaluate performance of various PTMs in **commodity hardware**:
  • DudeTM, cc-HTM, NV-HTM, Crafty, PSTM (Mnemosyne), SPHT
  • Previous solution evaluated in emulated PM

• Can Immediate Durability scale with commodity HTM+PM?
  • Yes! Using our novel SPHT design.

• Novel **commit logic**

• Novel **log replay techniques**

<table>
<thead>
<tr>
<th></th>
<th>DudeTM</th>
<th>Cc-HTM</th>
<th>NV-HTM</th>
<th>Crafty</th>
<th>SPHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Clock updated by TXs</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Extended TX vulnerability window</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Sequential mechanism to ensure durability</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Sequential Log Replay</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
Immediate Durability: NV-HTM

Execution of the transaction in HTM
Commit Logic

What if $T_3$ handled the commit markers?
Immediate Durability: SPHT
SPHT: log replay

• Background process replays logs in PM:
  • Prunes logs during normal execution, recovers application state on bootstrap

• Performance is critical!
  • System availability;
  • Frees log space.

• SPHT log replayer overview:
  • Linked log: avoid searching for the next transaction in the log;
  • Parallel log replay (via sharding of the persistent heap);
  • NUMA-aware;
  • Exploit WBINVD to avoid tracking which addresses to flush.

Details and optimizations in the paper
SPHT: linked log

• We propose SPHT and two variants using linking

• Without hints in the per-thread logs:
  • Log Replayer has to scan logs to order transaction

• **Linked log**: avoid search the next transaction in the log:
  • **Key idea**: log contains hints of where the next transaction is;
  • Pays a **relatively low cost** in transaction processing:
    • Exploit the commit logic to **pinpoint the predecessor** OR **successor**

Check pseudo-code in the paper
Evaluation

• Dual-socket Intel Xeon Gold 5218 CPU (16C/32T – HTM enabled)
• 128GB DRAM / 512GB PM (4 DIMMs)
• Code available\textsuperscript{1}:
  • SPHT-NL, SPHT-FL, SPHT-BL, NV-HTM, DudeTM, Crafty, cc-HTM and PSTM;
  • All implementations guarantee immediate durability.
• Tested in:
  • \textbf{STAMP} benchmark suit (check all benchmarks \textit{in the paper})
  • \textbf{TPC-C}: \textit{in the paper}

\textsuperscript{1} - \url{https://bitbucket.org/daniel_castro1993/spht}
STAMP

- SPHT and its variants scale up to 64 threads.
- Other solutions bottleneck around 1.5 MTX/s.
- Gains over NV-HTM of 2.6x.
Log Replay

Logs generated by **SPHT-NL**

Linking approaches

Gains 2.1x

Gains 1.3x
Key takeaways

• HTM in CPUs since 2013 and PM available since April 2018:
  • Many HTM+PM solutions proposed:
    • DudeTM \cite{ASPLOS'17}, cc-HTM \cite{ISMM'17}, NV-HTM \cite{IPDPS'18}, Crafty \cite{PLDI'20}
  • Little focus on immediate durability;
  • First experimental study on real hardware that combine these systems.

• Existing solutions have scalability limitations:
  • During transaction processing:
    • SPHT introduces a novel group commit approach;
    • \textbf{2.6x better throughput} at high thread count (64 threads).

  • In the replay phase:
    • Parallel log replay (scales up to \textbf{16 threads}), linked log (\textbf{up to 2.1x speedup})

Thank you! Send follow-up questions to: daniel.castro@ist.utl.pt