An Analysis of Persistent Memory Use with WHISPER

Sanketh Nalli, Swapnil Haria, Michael M. Swift, Mark D. Hill, Haris Volos*, Kimberly Keeton*

University of Wisconsin-Madison & *Hewlett-Packard Labs

Executive Summary
Facilitate better system support for Persistent Memory (PM)

Wisconsin-HP Labs Suite for Persistence, a benchmark suite for PM
- 4% accesses to PM, 96% accesses to DRAM
- 5-50 epochs/tx, contributed by memory allocation & logging
- 75% of epochs are small, update just one PM cacheline
- Re-referencing PM cachelines:
  Common in a thread, rare across threads

Hands Off Persistence System (HOPS) optimizes PM transactions

WHISPER: research.cs.wisc.edu/multifacet/whisper

Outline

WHISPER: Wisconsin-HP Labs Suite for Persistence
WHISPER Analysis
HOPS: Hands-Off Persistence System

Persistent Memory is coming soon

PM = NVM attached to CPU on memory bus
Offers low latency reads and persistent writes
Allows user-level, byte-addressable loads and stores
What guarantees after failure?

- **Durability** = Data survives failure
- **Consistency** = Data is usable

1. Data update followed by pointer update in cache
2. Pointer is evicted from cache to PM
3. Data lost on failure, dangling pointer persists

Achieving consistency

1. Store data update in cache
2. Flush data update to PM
3. Store pointer update in cache
4. Flush pointer update to PM

- **Ordering** = Useful building block of consistency mechanisms
- **Epoch** = Set of writes to PM guaranteed to be durable before ANY subsequent writes become durable

Ordering primitives: SFENCE on x86-64

PM systems for consistency

- **Native**
  Application-specific optimizations
- **Persistent library**
  Atomic allocations, transactions
- **PM-aware Filesystems**
  POSIX interface

What’s the problem?

- Lack of standard workloads slows research
- Micro-benchmarks not very representative
- Partial understanding of how applications use PM
**WHISPER benchmark suite**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Type</th>
<th>Brief description</th>
<th>(*Adapted to PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo*</td>
<td>KV store</td>
<td>Scalable, multi-version key-value store</td>
<td></td>
</tr>
<tr>
<td>N-store*</td>
<td>Database</td>
<td>Fast, in-memory relational DB</td>
<td></td>
</tr>
<tr>
<td>Redis</td>
<td>NVML</td>
<td>Remote Dictionary Service</td>
<td></td>
</tr>
<tr>
<td>C-tree</td>
<td>NVML</td>
<td>Microbenchmarks for simulations</td>
<td></td>
</tr>
<tr>
<td>Hashmap</td>
<td>NVML</td>
<td>Microbenchmarks for simulations</td>
<td></td>
</tr>
<tr>
<td>Vacation*</td>
<td>Mnemosyne</td>
<td>Online travel reservation system</td>
<td></td>
</tr>
<tr>
<td>Memcached*</td>
<td>Mnemosyne</td>
<td>In-memory key-value store</td>
<td></td>
</tr>
<tr>
<td>NFS</td>
<td>PMFS</td>
<td>Linux server/client for remote file access</td>
<td></td>
</tr>
<tr>
<td>Exim</td>
<td>PMFS</td>
<td>Mail server; stores mails in per-user file</td>
<td></td>
</tr>
<tr>
<td>MySQL</td>
<td>PMFS</td>
<td>Widely used RDBMS for OLTP</td>
<td></td>
</tr>
</tbody>
</table>

**Outline**

✔ WHISPER: Wisconsin-HP Labs Suite for Persistence

→ WHISPER Analysis

HOPS: Hands-Off Persistence System

**How many accesses to PM?**

- Total number of accesses in a WHISPER application
  - 4% Accesses to PM
  - 96% Accesses to DRAM

**Suggestion:** Do not impede volatile accesses

**How many epochs/transaction?**

Durability after every epoch impedes execution

**Expectation:** 3 epochs/TX = log + data + commit

**Reality:** 5 to 50 epochs/TX

**Suggestion:** Enforce durability only at the end of a transaction
What contributes to epochs?

Log entries
  • **Undo log**: Alternating epochs of log and data
  • **Redo log**: 1 Log epoch + 1 data epoch

Persistent memory allocation
  • 1 to 5 epochs

**Suggestion:** Use redo logs and reduce epochs from memory allocator

How large are epochs?

- **Determines amount of state buffered per epoch**
  - Small epochs are abundant

  - 75% update single cacheline

  - Large epochs in PMFS

**Suggestion:** Consider optimizing for small epochs

What are epoch dependencies?

Self-dependency: B → D
Cross-dependency: 2 → C

Why do they matter?

- Dependency can stall execution

Measured dependencies in 50 microsec window

**Suggestion:** Design multi-versioned buffers
  OR avoid updating same cacheline across epochs
Outline

✔ WHISPER: Wisconsin-HP Labs Suite for Persistence
✔ WHISPER Analysis
→ HOPS: Hands-Off Persistence System

ACID Transactions in HOPS

WHISPER

4% accesses to PM, 96% to DRAM
5-50 epochs/transaction
Self-dependencies common
Cross-dependencies rare

HOPS

Volatile memory hierarchy (almost) unchanged by PBs
Order epochs without flushing
Allows multiple copies of same cacheline in PB via timestamps
Correct, conservative method using coherence & timestamps
**Summary**

- Persistent Memory (PM) is coming soon
- Progress is slowed by ad-hoc micro-benchmarks
- We contributed **WHISPER**, open-source benchmark suite
- **HOPS** design, based on WHISPER analysis
- We hope for more similar analysis in the future!

[research.cs.wisc.edu/multifacet/whisper/](research.cs.wisc.edu/multifacet/whisper/)

**Extra**

- **WHISPER**: Wisconsin-HP Labs Suite for Persistence
- 4% accesses to PM, 96% accesses to DRAM
- 5-50 epochs/TX, primarily small in size
- Cross-dependencies rare, self-dependencies common
- **HOPS** improves PM app performance by 24%
- More results in ASPLOS’17 paper and code at:

[research.cs.wisc.edu/multifacet/whisper/](research.cs.wisc.edu/multifacet/whisper/)
A Simple Transaction using Epochs

```c
TM_BEGIN();
pobj.data = 42;
pobj.init = True;
TM_END();
```

Transaction Begin:
- Log entries stored & persisted.
- Variables stored & persisted.

Epoch 1
- Log entries
- Stored & persisted.
- Variables
- Stored & persisted.

Epoch 2
- PMFS
- Mnemosyne
- Logs every PM write
- PMFS
- NVML
- Clears log
- Auxiliary structures
- < 5% writes to PM
- Non-temporal writes
- Mnemosyne logs
- PMFS user-data