SuperMem: Enabling Application-transparent Secure Persistent Memory with Low Overheads

Pengfei Zuo\textsuperscript{1,2}, Yu Hua\textsuperscript{1}, Yuan Xie\textsuperscript{2}

\textsuperscript{1} Huazhong University of Science and Technology, China
\textsuperscript{2} University of California at Santa Barbara, USA

52nd IEEE/ACM International Symposium on Microarchitecture (MICRO), 2019
DRAM → Persistent Memory

Non-volatility

Low power

Large capacity

Images from Internet
Two Key Challenges for Persistent Memory

- Persistence
  - Core
  - Cache

- Security
  - username, password

Volatile: Inconsistency
Non-volatile: Physical access

Gap between persistence and security: Encryption incurs new inconsistency problem
Counter Mode Encryption

Counter Cache

AES Engine

XOR

One-time pad

Encrypted Data

CPU Cache

Write Back

Write Back

4
Counter Mode Encryption

Counter Cache

Updated Counter

Encrypted Data

CPU Cache

Write Back

Write Back
Crash Inconsistency Caused by Encryption

- Data and counter cannot reach NVM at the same time
Crash Inconsistency Caused by Encryption

**CASE 1:**
- Data and counter cannot reach NVM at the same time
CASE 2: Data and counter cannot reach NVM at the same time
Crash Inconsistency Caused by Encryption

- Data and counter cannot reach NVM at the same time
- `Clflush` and `mfence` cannot operate the counter cache
Existing Solutions (Write-back Counter Cache)

Large Battery Backup
[Awad et al., ASPLOS'16]
[Zuo et al., MICRO'18]

Software-level Modification
[Liu et al., HPCA'18]

New programming primitives
• `counter_cache_writeback()`
• `CounterAtomic`

Error Correction
[Ye et al., MICRO’18]

Expensive
Portability limitation
Long recovery time
SuperMem: Secure and Persistent Memory

- Exploit a write-through counter cache
  - No large battery backup
  - No software-level modifications
  - No need to correct counters
  - Double writes
- A counter write coalescing scheme
  - Reduce the number of write requests
- A cross-bank counter storage scheme
  - Speedup memory writes

Asynchronous DRAM refresh (ADR): cache lines reaching the write queue can be considered durable.
SuperMem: **Secure and Persistent Memory**

- **Application-transparent**
- **Write-through counter cache** *(Guarantee consistency)*
  - Counter write coalescing *(Reduce writes)*
  - Cross-bank counter storage *(Speedup writes)*

Asynchronous DRAM refresh (ADR): cache lines reaching the write queue can be considered durable.
SuperMem: Secure and Persistent Memory

- **Write-through counter cache** *(Guarantee consistency)*
- **Counter write coalescing** *(Reduce writes)*
- **Cross-bank counter storage** *(Speedup writes)*

Asynchronous DRAM refresh (ADR): cache lines reaching the write queue can be considered durable.
Write-through Counter Cache

- Ensure that data and its counter reach the write queue in the same time
  - Write through counter cache
Write-through Counter Cache

- Ensure that data and its counter reach the write queue in the same time
  - Write through counter cache
  - Add a register
SuperMem: Secure and Persistent Memory

- Asynchronous DRAM refresh (ADR): cache lines reaching the write queue can be considered durable.
- Write-through counter cache (Guarantee consistency)
- Counter write coalescing (Reduce writes)
- Cross-bank counter storage (Speedup writes)
Cross-bank Counter Storage

- **SingleBank**: Counters are stored in a continuous area in NVM [ASPLOS’15, ASPLOS’16, HPCA’18]
Cross-bank Counter Storage

- **SameBank**: Stores the counters of data into their local banks

```plaintext
Bank ID:
0  1  2  3  4  5  6  7

- $Ctr_0, Data_0$
- $Ctr_1, Data_1$
- $Ctr_2, Data_2$

Data Area

Ctr Area
```

2X write latency
Cross-bank Counter Storage

- **XBank:** Stores each data and its counter into different banks to leverage bank parallelism.

![Diagram of XBank storage](image)

- Bank ID:
  - Bank 0: Data 0, Ctr 0
  - Bank 1: Data 1, Ctr 1
  - Bank 2: Data 2, Ctr 2
  - Bank 3: Data 3, Ctr 3
  - Bank 4: Data 4, Ctr 4
  - Bank 5: Data 5, Ctr 5
  - Bank 6: Data 6, Ctr 6
  - Bank 7: Data 7, Ctr 7
SuperMem: Secure and Persistent Memory

- Write-through counter cache (Guarantee consistency)
- Counter write coalescing (Reduce writes)
- Cross-bank counter storage (Speedup writes)

Asynchronous DRAM refresh (ADR): cache lines reaching the write queue can be considered durable.
Locality-aware Counter Write Coalescing

Spatial locality of counter storage
– All counters of a page are stored in a counter line

A page: (64 lines)

A counter line: (64B)
Locality-aware Counter Write Coalescing

➤ Spatial locality of counter storage
  – All counters of a page are stored in a counter line

A page: Line₁ Line₂ Line₃ Line₄ … … … … Line₆₄
  (64 lines)

A log entry or the transaction data

➤ Spatial locality of log and data writes
Locality-aware Counter Write Coalescing

An example of writing 4 lines within a page

A page: Line$_1$ Line$_2$ Line$_3$ Line$_4$ ... ... ... ... Line$_{64}$

(64 lines)
Locality-aware Counter Write Coalescing

- An example of writing 4 lines within a page

Cache

A B C D

Write Queue

Dc D Cc C Bc B Ac A
Locality-aware Counter Write Coalescing

An example of writing 4 lines within a page

**Ac:**
```
M m₁' m₂ m₃ m₄ ... m₆₄
```

**Bc:**
```
M m₁' m₂' m₃ m₄ ... m₆₄
```

**Cc:**
```
M m₁' m₂' m₃' m₄ ... m₆₄
```

**Dc:**
```
M m₁' m₂' m₃' m₄' ... m₆₄
```

Write Queue

---

25
Locality-aware **Counter Write Coalescing**

- Coalescing counter writes in the write queue

**Ac:**

- $M \ m_1' \ m_2 \ m_3 \ m_4 \ \ldots \ \ m_{64}$

**Bc:**

- $M \ m_1' \ m_2' \ m_3 \ m_4 \ \ldots \ \ m_{64}$

**Cc:**

- $M \ m_1' \ m_2' \ m_3' \ m_4 \ \ldots \ \ m_{64}$

**Dc:**

- $M \ m_1' \ m_2' \ m_3' \ m_4' \ \ldots \ \ m_{64}$

Write Queue
Locality-aware **Counter Write Coalescing (CWC)**

- Coalescing counter writes in the write queue

**With CWC**

- Write Queue
  - Dc, D, C, B, A

**Without CWC**

- Write Queue
  - Dc, D, Cc, C, Bc, B, Ac, A
## Performance Evaluation

- Model NVM using gem5 and NVMain

<table>
<thead>
<tr>
<th>Comparisons</th>
<th>Benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsec:</strong> An un-encrypted NVM</td>
<td><strong>Array:</strong> Randomly swapping entries</td>
</tr>
<tr>
<td><strong>WB:</strong> An ideal write-back scheme</td>
<td><strong>Queue:</strong> Randomly enqueueing and dequeueing</td>
</tr>
<tr>
<td><strong>WT:</strong> A write-through scheme</td>
<td><strong>B-tree:</strong> Inserting random KVs</td>
</tr>
<tr>
<td><strong>WT+CWC:</strong> A write-through scheme with CWC</td>
<td><strong>Hash Table:</strong> Inserting random KVs</td>
</tr>
<tr>
<td><strong>WT+Xbank:</strong> A write-through scheme with XBank</td>
<td><strong>RB-tree:</strong> Inserting random KVs</td>
</tr>
<tr>
<td><strong>SuperMem</strong></td>
<td></td>
</tr>
</tbody>
</table>
SuperMem achieves the performance comparable to a secure NVM with an ideal write-back cache (WB)
SuperMem achieves the performance comparable to a secure NVM with an ideal write-back cache (WB)
The Number of Write Requests

SuperMem reduces up to 50% of write requests by using the CWC scheme
Conclusion

Problem
- Memory encryption incurs crash inconsistency issue

Existing Work
- Using a write-back counter cache
- Large battery backup, software-level modification, or error correction

Our Solution
- SuperMem: exploit a write-through counter cache
  - Large battery backup, software-level modification, error correction
  - Counter write coalescing for reducing writes
  - Cross-bank counter storage for speeding up writes
Thanks! Q&A