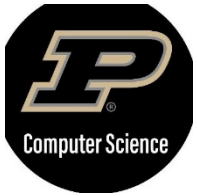
Unbounded Hardware Transactional Memory for a Hybrid DRAM/NVM Memory System

Jungi Jeong[§], Jaewan Hong[†], Seungryoul Maeng[†], Changhee Jung[§], and Youngjin Kwon[†]



[§] Purdue University

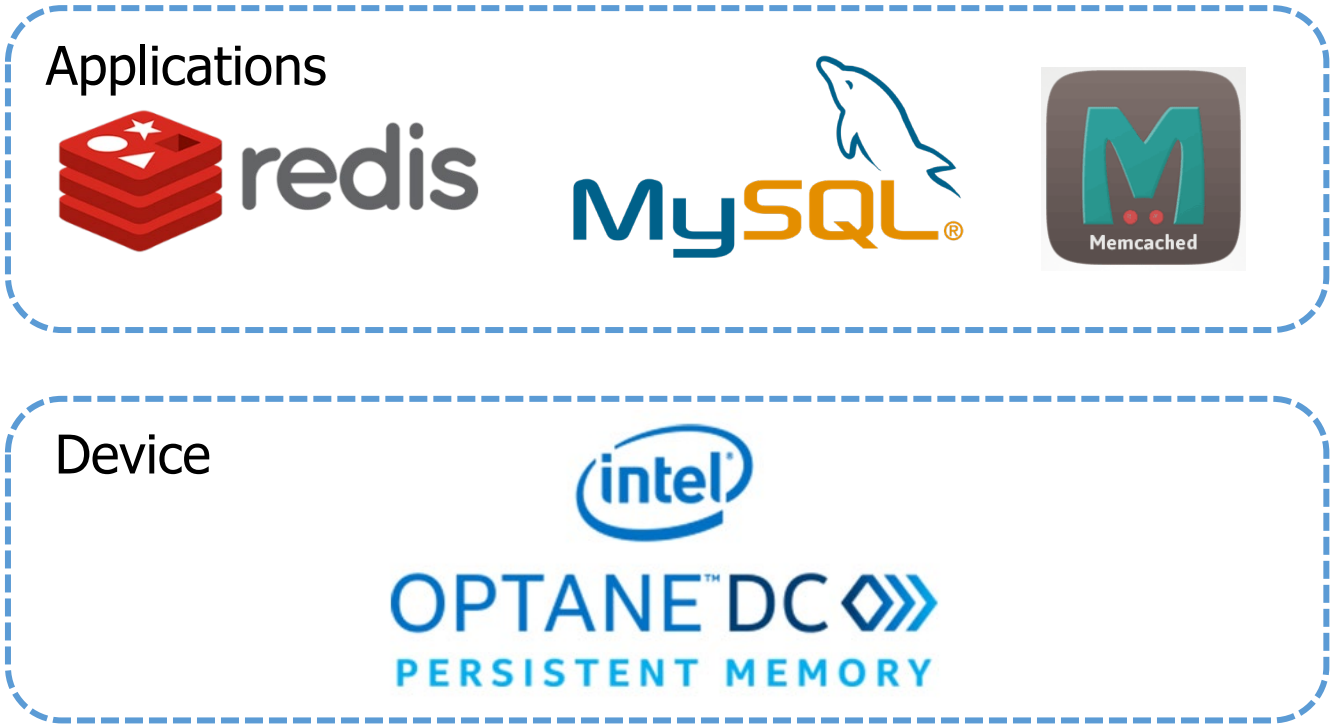
[†] KAIST

KAIST School of Computing

Session 5A: Hybrid Memory
NVMW 2021

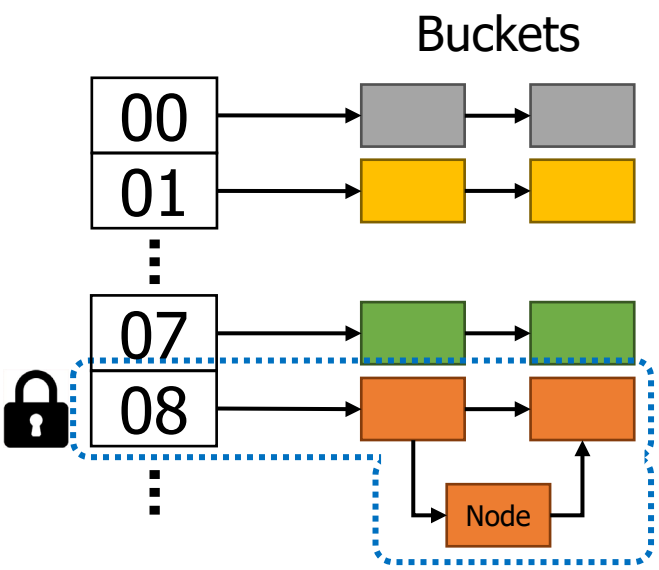
Persistent Memory (PM)

- Larger capacity than DRAM
- Non-Volatile
- Direct-Access via LD/ST instructions



What PM Programming Requires

- Ex) Persistent Hash Table[1,2]



[T0]: Add new node

```
Prev->Ptr = Node;
Node->Ptr = Next;
```

Atomicity: all or nothing
Durability: written to NVM

[T1]: Add new node

```
Prev->Ptr = Node;
Node->Ptr = Next;
```

Consistency: App-Dependent

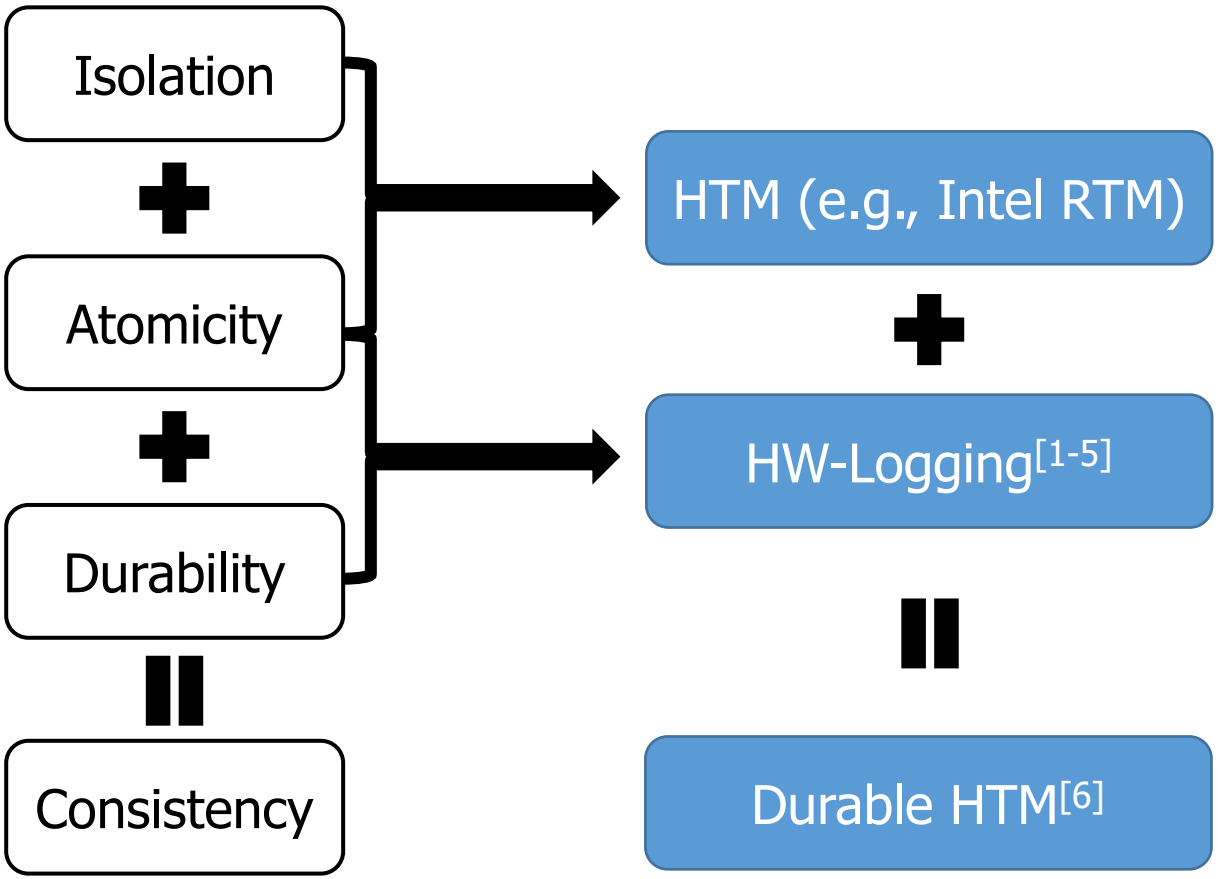
Isolation: serialize if needed

PM Programming requires ACID
➔ **Durable Transactions**

[1] Zuo et al., OSDI 2018.

[2] Nam et al., FAST 2019.

Durable Hardware Transactional Memory (HTM)

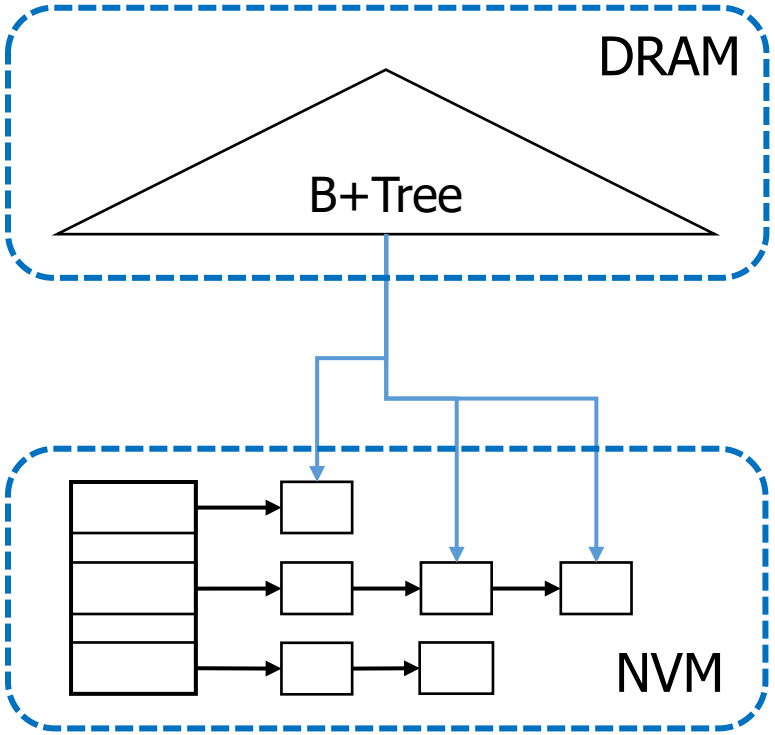


[1] Doshi et al., HPCA 2016.
[2] Joshi et al., HPCA 2017.
[3] Shin et al., MICRO 2017.
[4] Ogleari et al., HPCA 2018.
[5] Jeong et al., MICRO 2018.
[6] Joshi et al., ISCA 2018.

Limitations of Previous Work

1. NVM data only in durable transactions

- But, recent applications strive to use both DRAM/NVM
  - Ex) Index in DRAM, data in NVM[1,2]



To accelerate search

+

To simplify programming

=

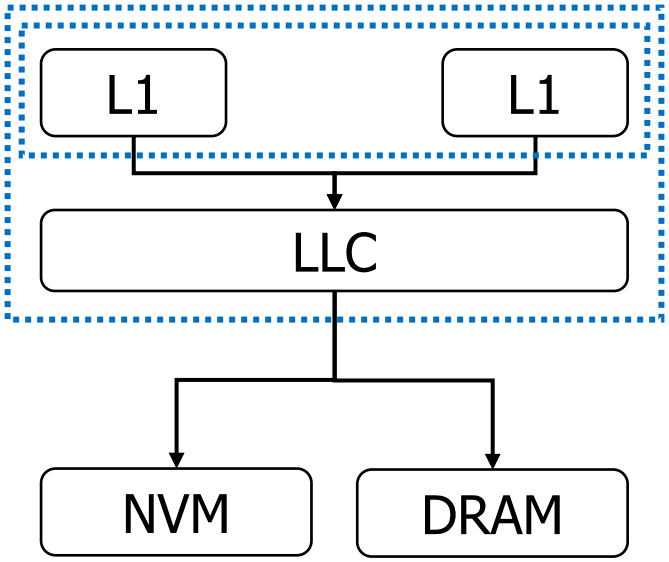
Hybrid DRAM/NVM Support

[1] Yang et al., 2015.
[2] Xia et al., 2017.

Limitations of Previous Work

2. Limited transaction boundary

- Previous works are bounded or inefficient



Commercial HTMs (e.g., Intel):
L1-bounded

State-of-the-art[1]:
LLC-bounded

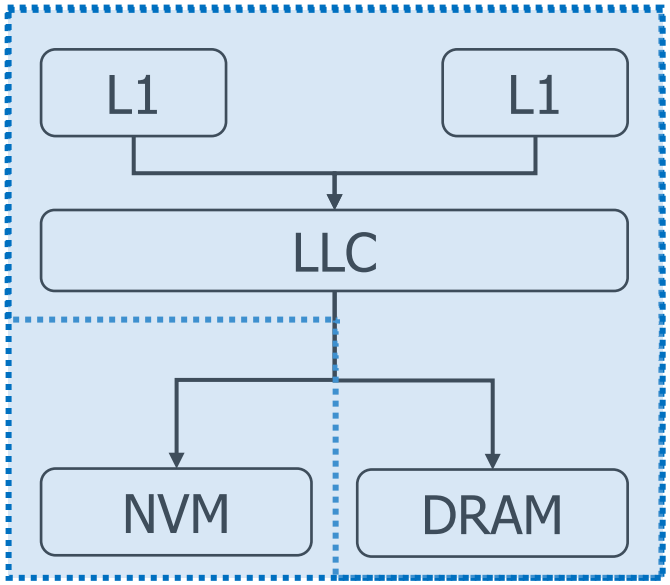
- Fallback path
  : ↑ **programming complexity**
- Serialization
  : ↓ **transaction concurrency**

[1] Joshi et al., 2018.

Limitations of Previous Work

2. Limited transaction boundary

- Previous works are bounded or inefficient



Commercial HTMs (e.g., Intel):
L1-bounded

State-of-the-art[1]:
LLC-bounded

DRAM-only unbounded HTM[2,3]:
Severe **false-positives**

Unlimited Conflict Detection

: at all memory hierarchies
: with low false-positive rates

[1] Joshi et al., 2018.
[2] L. Yen et al., 2007.
[3] A. Shriraman et al., 2008.

UHTM

- Unbounded HTM for a hybrid DRAM/NVM system

Unlimited Conflict Detection

- Detect conflicts in any memory hierarchy (both in caches and off-chip memory)

Hybrid DRAM/NVM Support

- Allow manipulating both DRAM/NVM data structures in Tx

UHTM

- Unbounded HTM for a hybrid DRAM/NVM system

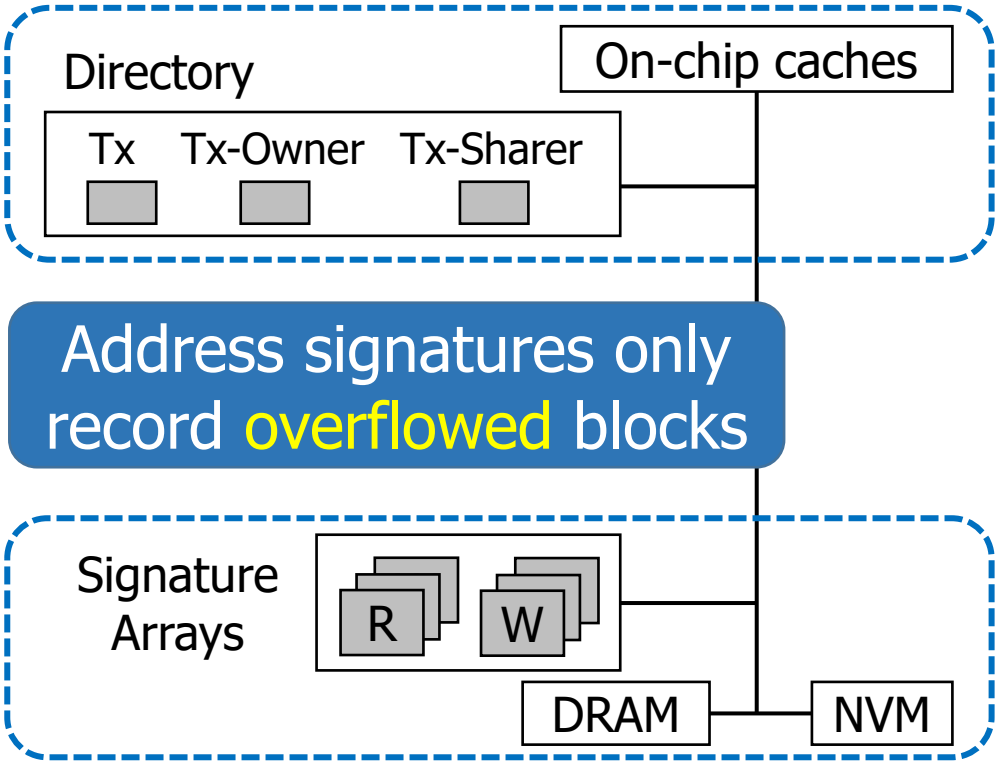
Unlimited Conflict Detection

- Challenge: very high false-positive rates of address-signatures in L1-cache
  - > 99% conflict rates due to false-positives[1,2]

- Idea: hierarchical detection
  - On-chip caches (frequently accessed)
    - ➔ **Accurate, but bounded in caches**
  - Off-chip memory (less frequently accessed)
    - ➔ **Allows false-positives, but unbounded**

[1] L. Yen et al., 2007.
[2] A. Shriraman et al., 2008.

Staged Conflict Detection



Accurate detection with cache-coherence

Address signatures only record overflowed blocks

Unlimited detection with address signatures

Reduce false-positive rate from **99%** (signature-only[1,2]) to **26%** (coherence+signature)*

[1] L. Yen et al., 2007.
[2] A. Shriraman et al., 2008.

* Both use 512-bit signature

Can we reduce false-positives further?

- Large signature (16k-bit) reduces to 9% (from 26%)
- But, too much space overheads
  - E.g., 16-core requires 64KB for signatures

Observations

- #1: conflicts by non-Tx & background processes
  - No possible conflicts, but false-positives produce false-conflicts
- #2: % false-conflicts ~= O(# signatures) ~= O(# cores)
  - Higher possibility to produce false-positives in signatures

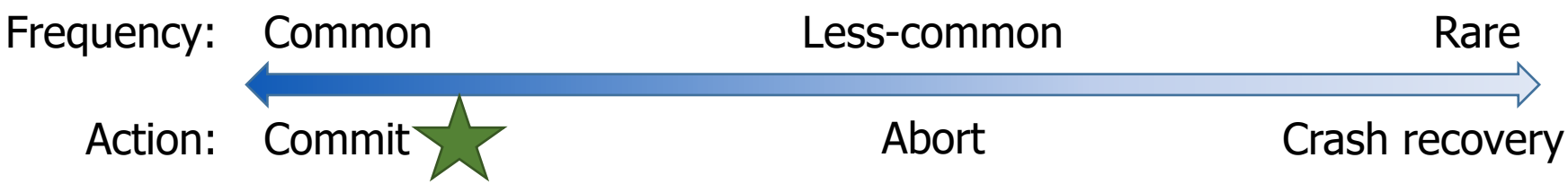
Signature Isolation

- Conflict domain
  - A group of transactions that potentially have conflicts each other
  - Assign transaction-group ID to each thread
  - Ex) multiple threads in a process

- Signature checking happens within the conflict domain
  - With the same group
  - Non-Tx & background ones not participating in conflicts checking

UHTM

Hybrid DRAM/NVM Support



- Hybrid eager/lazy-approach for FAST-commit
  - DRAM data ➔ Undo-logs (eager)
  - NVM data ➔ Redo-logs* (lazy)

Placing cache-flush to NVM out of the critical path

- Crash recovery guaranteed by NVM redo-logging

* Jeong et al., MICRO 2018.

Methodology

- Gem5 simulator

Processor	16-core, In-Order, 2GHz, x86
L1 I/D cache	Private, 32KB, 8-way
L2 cache	Shared, 32MB, 16-way
DRAM	Read/Write: 82ns
NVM	Read: 175ns, write: 94ns

- Benchmarks

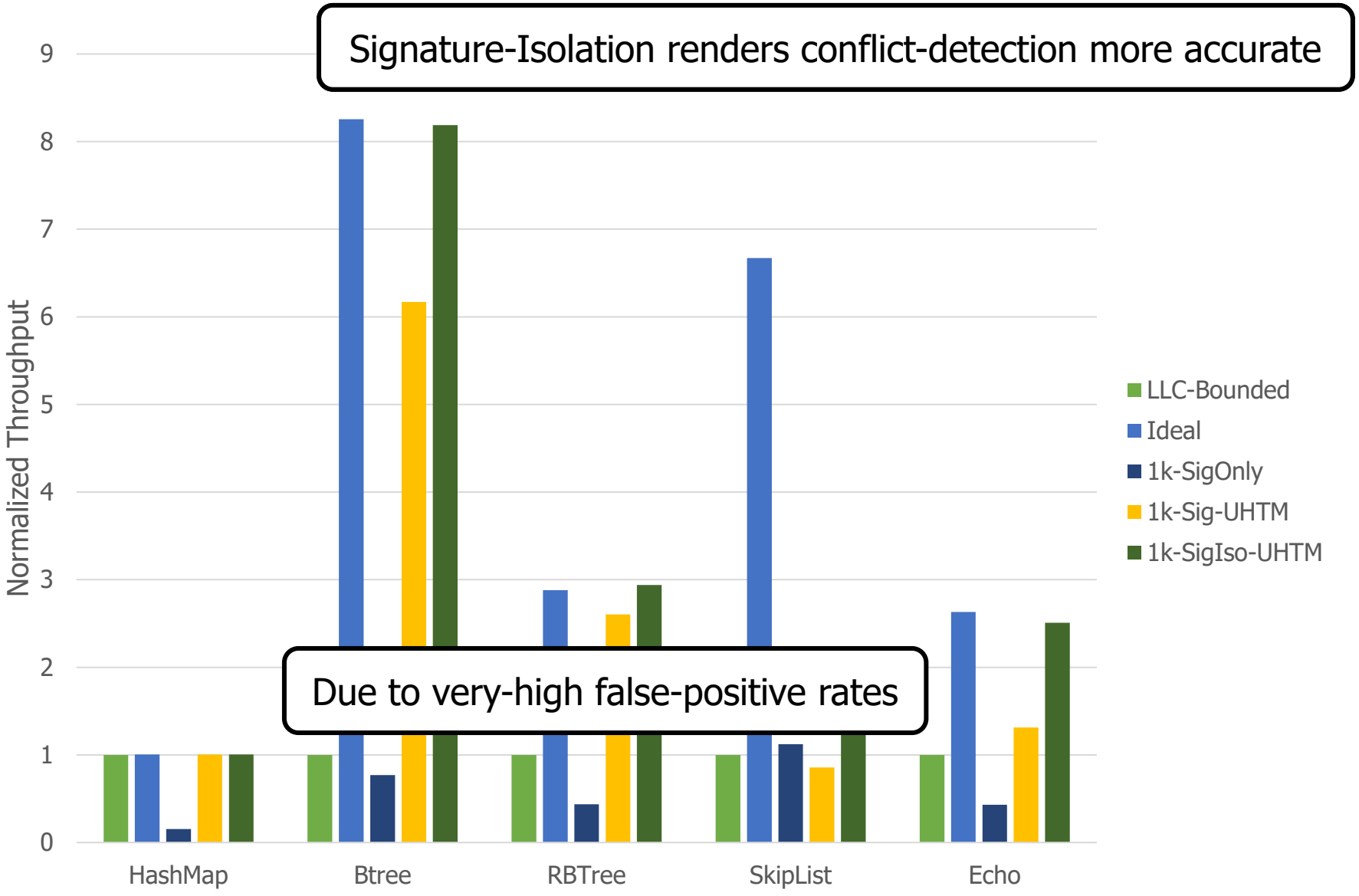
PMDK	HashMap, B-Tree, RB-Tree, SkipList
KV-Store	Hybrid-Index[1], Echo

* 4 processes with 4 threads

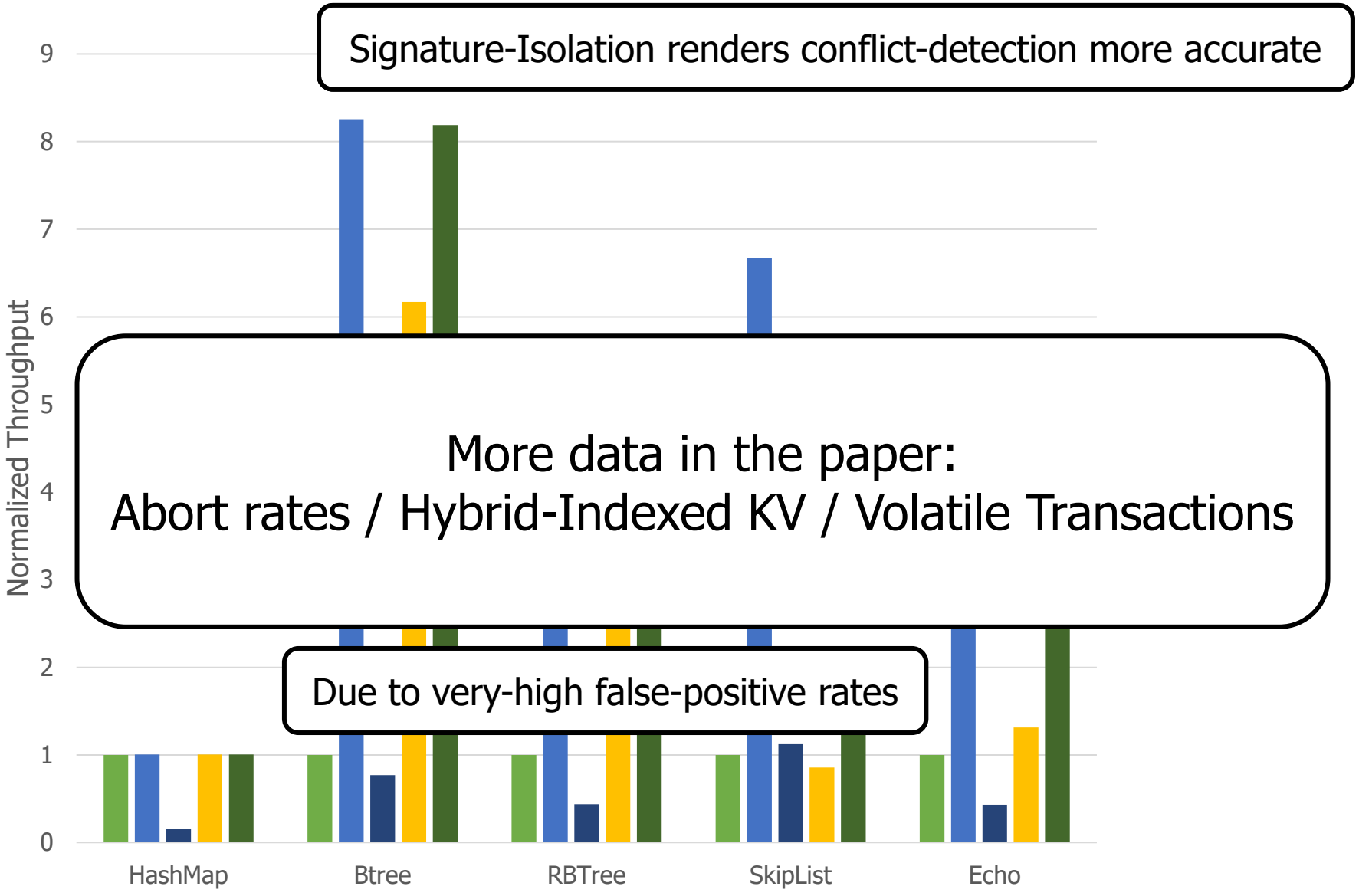
- Comparing schemes
  - LLC-Bounded[2]: Abort and restart if Tx overflows from LLC
  - SigOnly-HTM: Signature-only unbounded HTM
  - Ideal: Ideal unbounded HTM with no false-positives
  - UHTM

[1] F. Xia et al., 2017.
[2] A. Joshi et al., 2018.

Evaluation


Signature-Isolation renders conflict-detection more accurate
Due to very-high false-positive rates
Normalized Throughput
0
1
2
3
4
5
6
7
8
9
HashMap
Btree
RBTree
SkipList
Echo
LLC-Bounded
Ideal
1k-SigOnly
1k-Sig-UHTM
1k-SigIso-UHTM

Evaluation


Signature-Isolation renders conflict-detection more accurate
More data in the paper:
Abort rates / Hybrid-Indexed KV / Volatile Transactions
Due to very-high false-positive rates
Normalized Throughput
0
1
2
3
4
5
6
7
8
9
HashMap
Btree
RBTree
SkipList
Echo

Conclusion

- UHTM: Unbounded HTM for a DRAM/NVM Memory System

Unlimited Conflict Detection

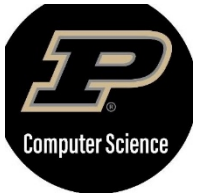
- Staged conflict detection
  - On-chip cache ➔ Cache-coherence protocol
  - Off-chip memory ➔ Address-Signatures + Isolation Technique

Hybrid DRAM/NVM Support

- Hybrid eager/lazy logging for fast COMMIT
  - DRAM data ➔ Undo-logs (eager)
  - NVM data ➔ Redo-logs (lazy)

Unbounded Hardware Transactional Memory for a Hybrid DRAM/NVM Memory System

Jungi Jeong[§], Jaewan Hong[†], Seungryoul Maeng[†], Changhee Jung[§], and Youngjin Kwon[†]



[§] Purdue University

[†] KAIST

KAIST School of Computing

Session 5A: Hybrid Memory
NVMW 2021