INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP / 2021 / CHICAGO / USA
acm In-Cooperation
sighpc
AUGUST 9-12, 2021

HDNH: a read-efficient and write-optimized hashing scheme for hybrid DRAM-NVM memory

Junhao Zhu[1], Kaixin Huang[2], Xiaomin Zou[3], Chenglong Huang[1], Nuo Xu[1], Liang Fang[1]

[1]National University of Defense Technology; [2]ByteDance Inc.; [3]Huazhong University of Science and Technology

国防科技大学
National university of defense technology

50th International Conference on Parallel Processing (ICPP)
August 9-12, 2021 in Virtual Chicago, IL

HPCL
State Key Laboratory of High Performance Computing

Outline

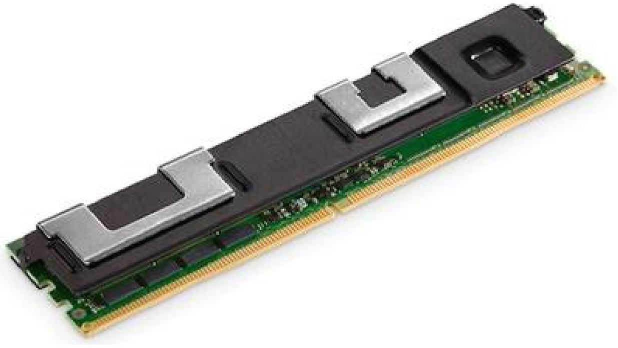
- **Background**
- Motivation
- Design of HDNH
- Evaluation
- Conclusions

Background

1. Non-volatile Memory (NVM)

- NVM features
  - Non-volatility
  - Byte-addressability
  - DRAM-scale latency
  - Large capacity

- NVM speedups storage systems
  - TB-scale memory for applications
  - Instant recovery from system failures

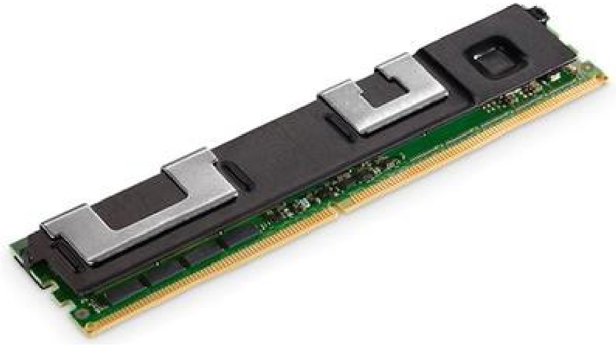


Intel Optane DC Persistent Memory (AEP)
512 GB per module at most
DIMM compatible

Background

2. Intel Optane DC Persistent Memory (AEP)

- ➢ New features of AEP (FAST '20)
  - ✓ 3x read latency and similar write latency compared with DRAM
  - ✓ Read and write bandwidth are 3x/6x lower than that of DRAM
  - ✓ The granularity disparity between CPU caches and AEP (64 vs 256 bytes)

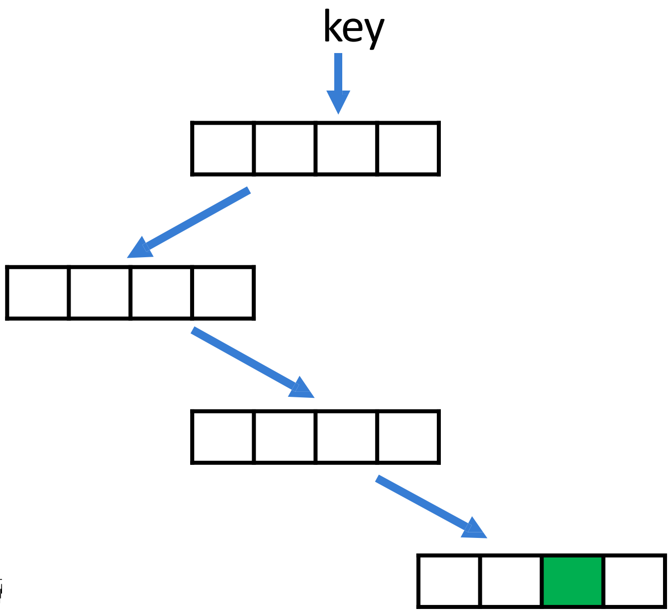


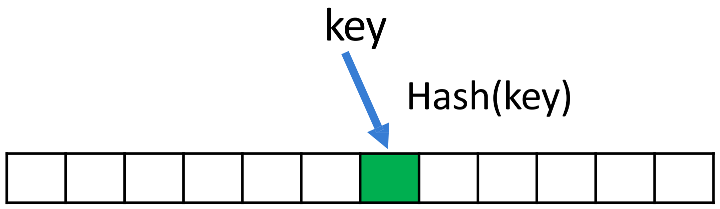
Intel Optane DC Persistent Memory (AEP)
512 GB per module at most
DIMM compatible

Background

3. NVM Index Structures

- NVM index structures are important for large-scale storage systems to provide fast queries
  - Tree-based structures
  - Hashing-based structures





✓ ***O(1) time complexity for point query***

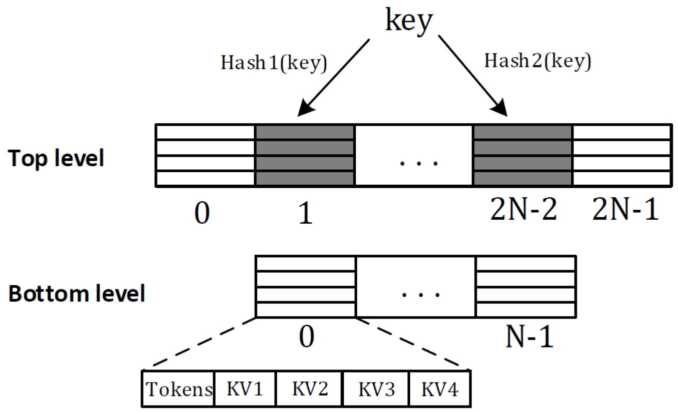
Outline

- Background
- **Motivation**
- Design of HDNH
- Evaluation
- Conclusions & Future Work

Motivation

1. Multiple accesses in NVM for probing data

- ➢ Existing hashing schemes usually use multi-slot bucket to resolve hash collisions
- ➢ Searching a bucket typically requires a linear scan of the slots
- ➢ One or multiple buckets in NVM have to be searched to find out if a key exists



Motivation

2. Hotspot issue for searching

- Alibaba observes that 50% to 90% of accesses only touch 1% of total items (FAST '20)
- It will cause longer access latency and waste NVM bandwidth for these hot data
- We can employ cache to store hot dataset in DRAM

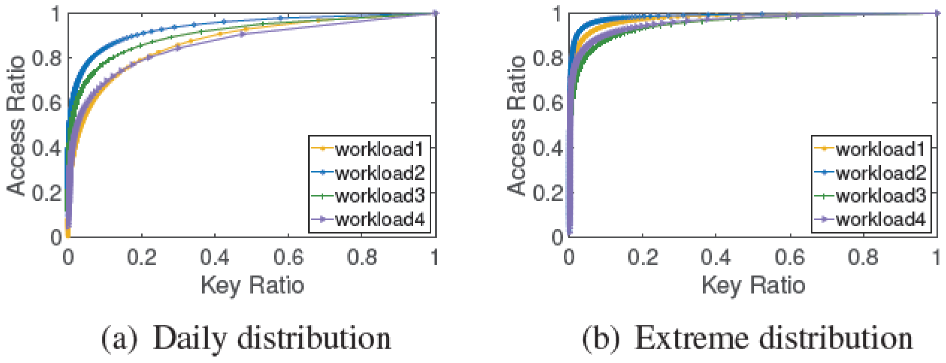


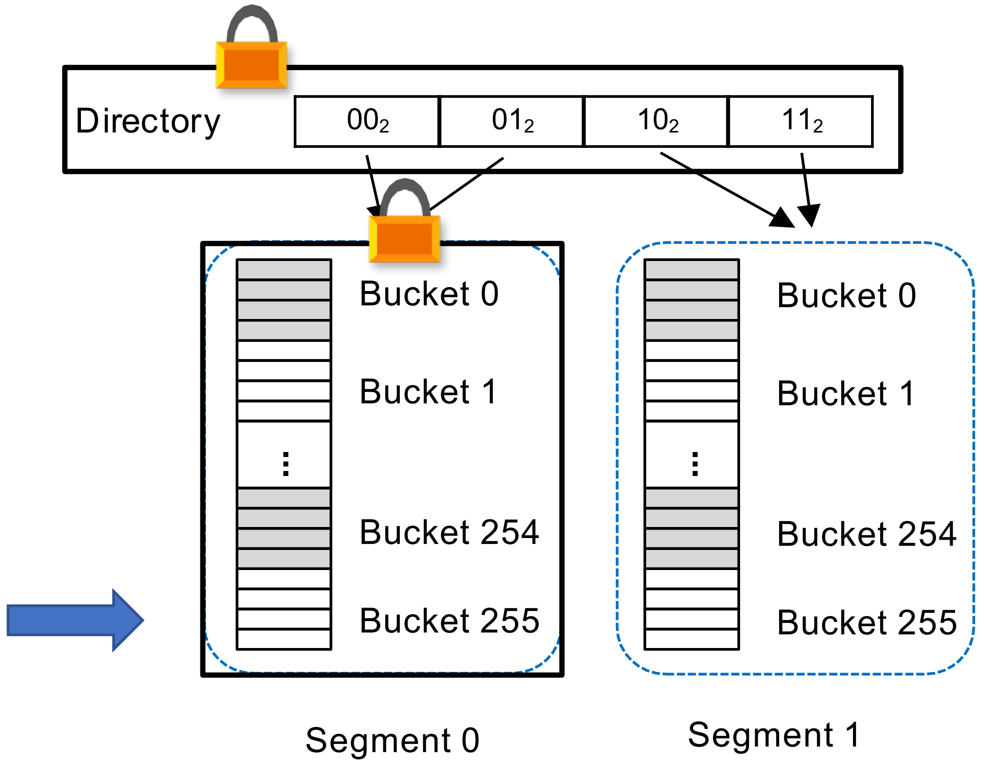
Figure 1: Access ratio of different keys.

Motivation

3. Coarse-grained lock for concurrency control

- ➢ Segment reader/writer locks for queries
- ➢ Bucket-level lock for concurrency
- ➢ Heavyweight concurrency control can easily exhaust NVM's limited bandwidth

Coarse-grained locks



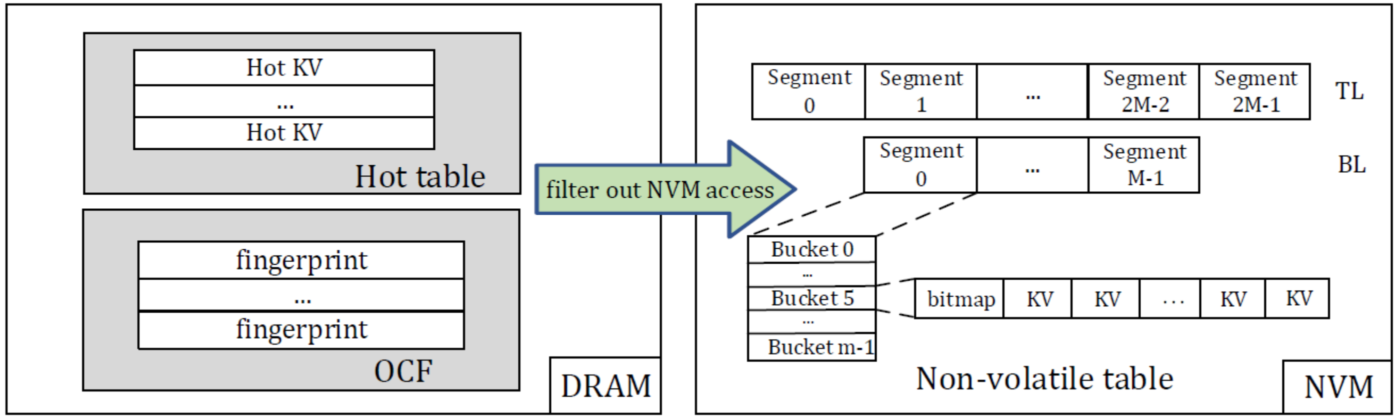
Outline

- Background
- Motivation
- **Design of HDNH**
- Evaluation
- Conclusions & Future Work

Design of HDNH

1. Overview

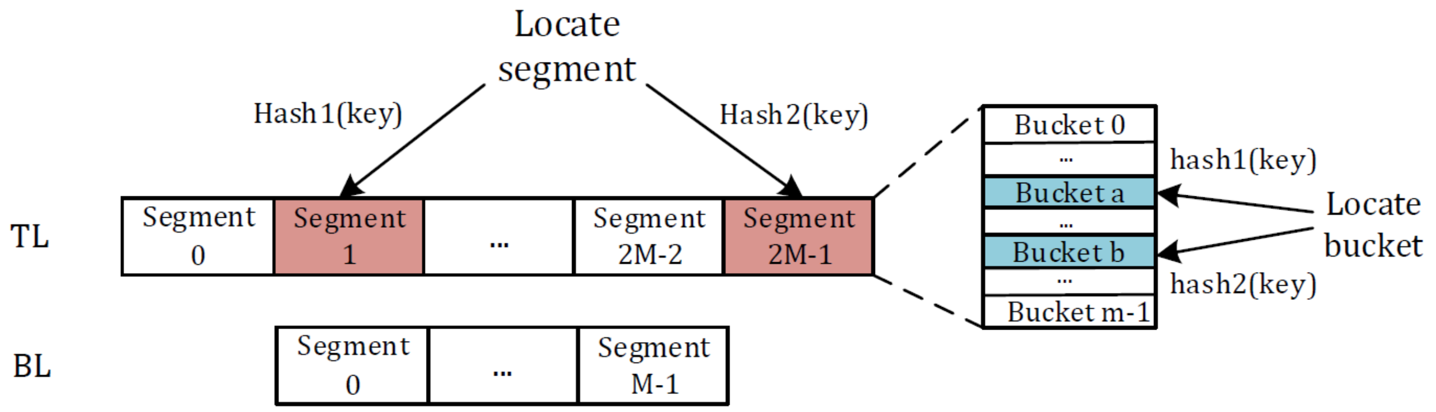
- HDNH sets up a two-level structure in NVM
- The top level has 2M segments and the bottom level has M segments
- HDNH chooses segment as the hashing unit
- HDNH places Optimistic Compression Filter (OCF) and Hot table in DRAM

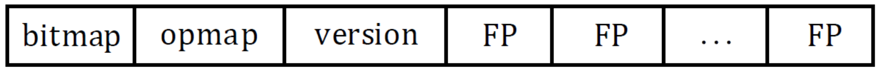


Design of HDNH

2. Optimistic Compression Filter (OCF)

- ➢ OCF reduces excessive NVM access for probing data
- ➢ OCF uses fingerprints to filter out unnecessary NVM reads in DRAM
- ➢ OCF properly configures data and metadata into hybrid memory

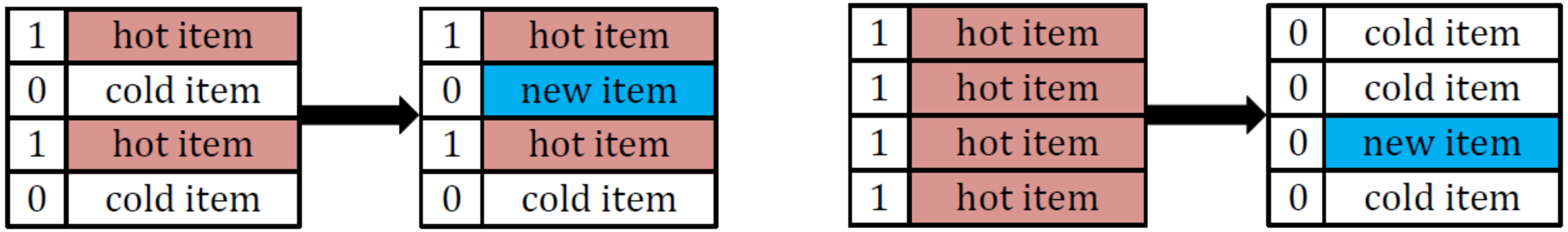




Design of HDNH

3. Hot Table

- Hot table solves the hotspot issue for searching
- Hot table places hot key-value items in DRAM to decrease the reads into NVM for skewed read workloads
- Hot table uses our proposed RAFL algorithm as its replacement strategy

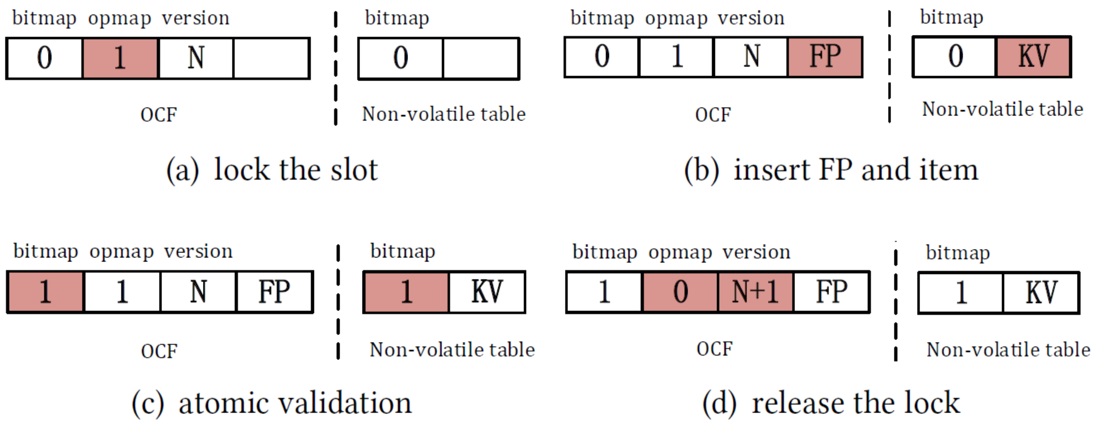


(a) Cold replacement. (b) Hot replacement.

Design of HDNH

4. Fine-grained Optimistic Concurrency

- HDNH sets opmap and version for each slot of hot table and OCF
- Opmap is used to indicate whether a slot is being written by a write thread
- Version is used to detect whether there are conflicts between read and write threads in the corresponding slot



(a) lock the slot
(b) insert FP and item
(c) atomic validation
(d) release the lock

Outline

- Background
- Motivation
- Design of HDNH
- **Evaluation**
- Conclusions

Evaluation

1. Experimental Setup

➢ Platform

- Intel Optane DC PMM configured in *App Direct* mode
- 24 threads in one NUMA node
- PMDK

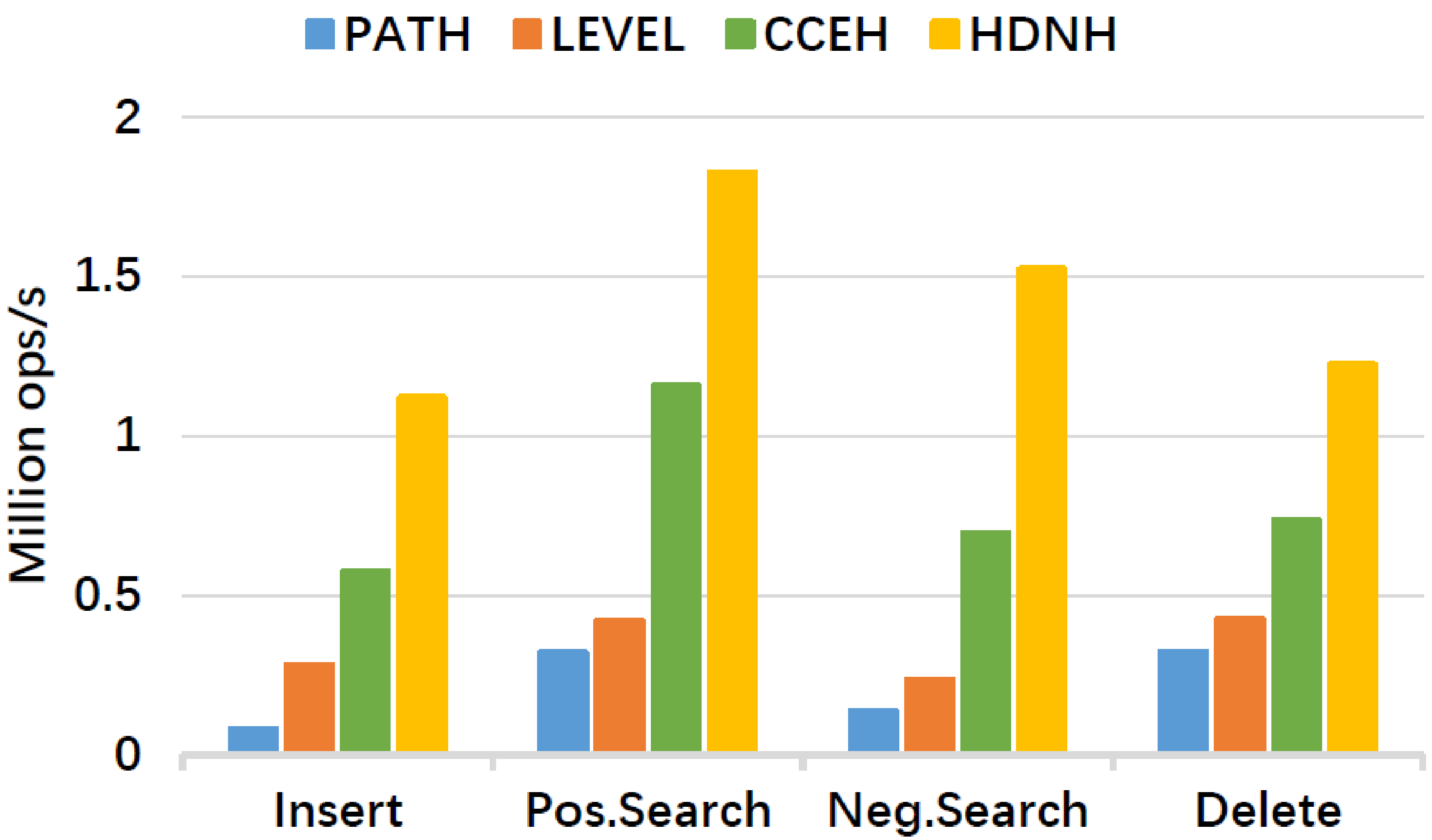
➢ Comparisons

- **PATH**: static hashing designed to reduce write accesses [MSST ’17]
- **LEVEL**: original level hashing [OSDI ’18]
- **CCEH**: lazy deletion version, default probing distance (16 slots) [FAST ’19]
- **HDNH**: our index scheme

➢ Benchmark: YCSB

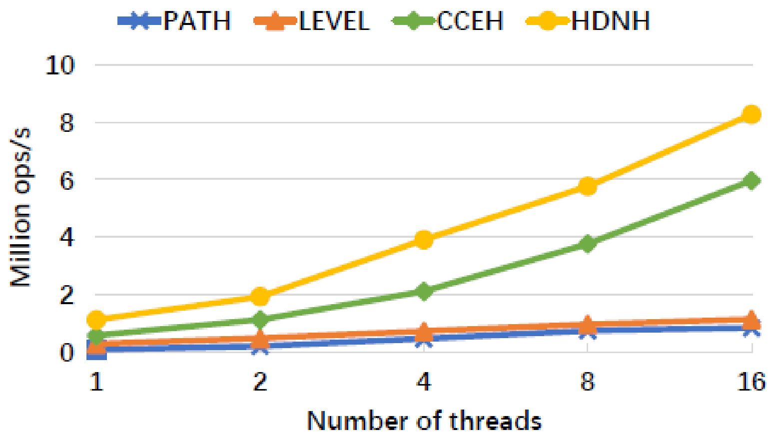
Evaluation

2. Single-thread Performance

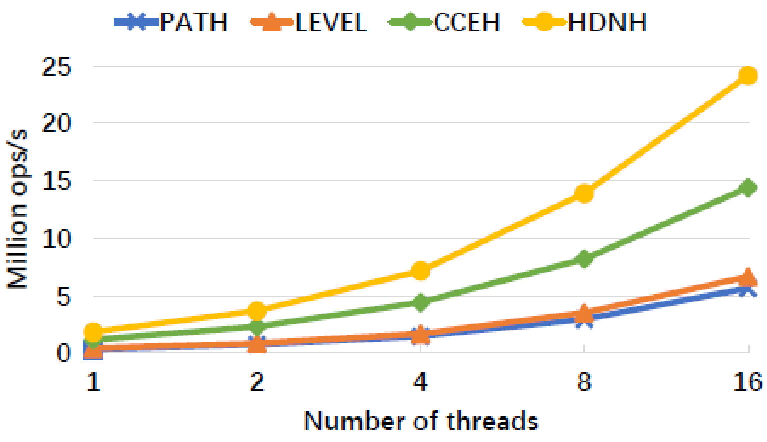


Evaluation

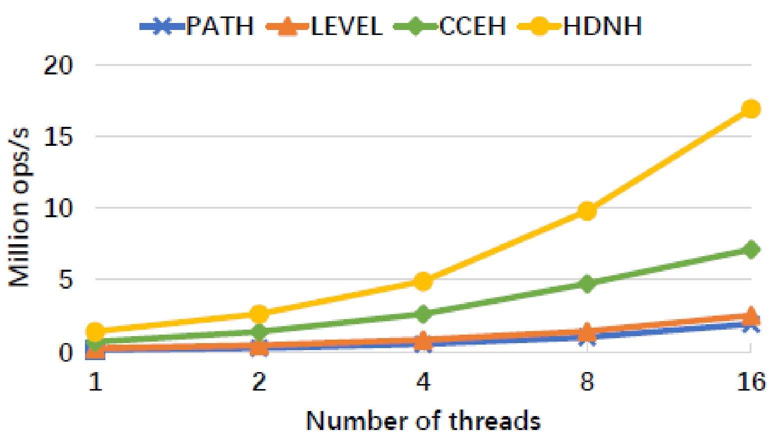
3. Concurrent Performance



(a) 100% insert workload.



(b) 100% serach workload.



(c) Mixed workload.

Evaluation

4. Recovery

Data size	2 million	20 million	200 million
OCF recovery time(ms)	0.8	9.1	60.8
Hot table recovery time(ms)	6.7	48.6	351.2
HDNH recovery time(ms)	8.3	60.5	435.1

Outline

- Background
- Motivation
- Design of HDNH
- Evaluation
- **Conclusions**

Conclusions

- HDNH persists key-value items in non-volatile table while metadatas are placed in OCF for fast access.
- HDNH uses hot table in DRAM to speed up search requests
- HDNH develops a fine-grained optimistic concurrency mechanism to enable high-performance concurrent accesses on multi-core systems
- Experimental results show that HDNH delivers superior performance and high scalability under various YCSB workloads

HDNH: a read-efficient and write-optimized hashing scheme for hybrid DRAM-NVM memory

Thank You
Q&A

国防科技大学
National university of defense technology

HPCL
State Key Laboratory of High Performance Computing