中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY ,CAS

USENIX ATC'17, Jul. 12-14

HiKV: A Hybrid Index Key-Value Store for DRAM-NVM Memory Systems

Fei Xia[1,2], Dejun Jiang[1], Jin Xiong[1], Ninghui Sun[1]

1. Institute of Computing Technology, CAS
2. University of Chinese Academy of Sciences

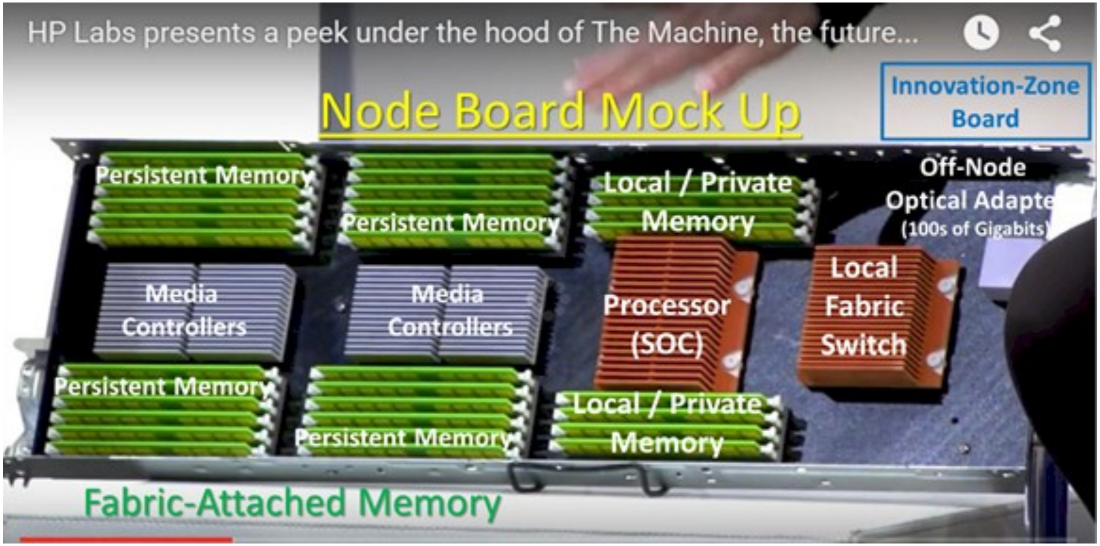
Non-Volatile Memory

- Emerging Non-Volatile Memories (NVMs)
  - PCM, ReRAM, STT-MRAM
- Characteristics of memory technologies
  [Xia+,JCST'2015, Yang+, FAST'2015,Chi+,ISCA'2016]

Categories	Volatility	Density	Read Latency	Write Latency	Write Endurance
DRAM	Yes	Low	60ns	60ns	10^{16}
PCM	No	High	50~70ns	150~1000ns	10^{9}
ReRAM	No	High	25ns	300ns	10^{12}
NAND Flash	No	High	35us	350us	10^{5}

Hybrid Memory (DRAM+NVM)

- DRAM： volatile, low latency, low capacity
- NVM： non-volatile, high latency, high capacity
- Hybrid DRAM and NVM memory is a promising solution.
  - Example: The machine



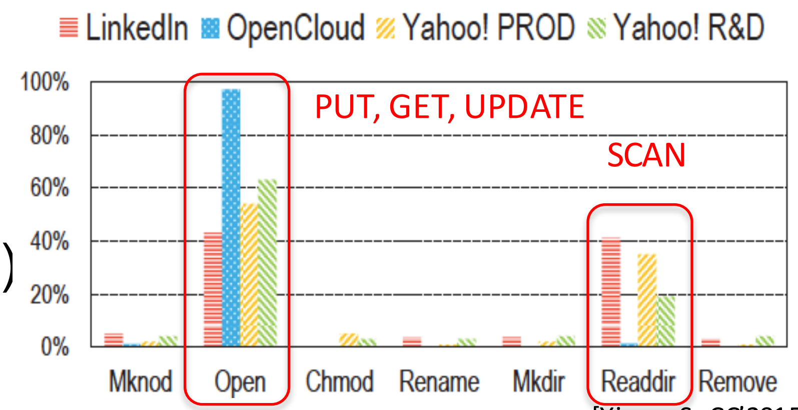
"The Machine" **[Source: HP Discover 2015]**

Key-Value Store

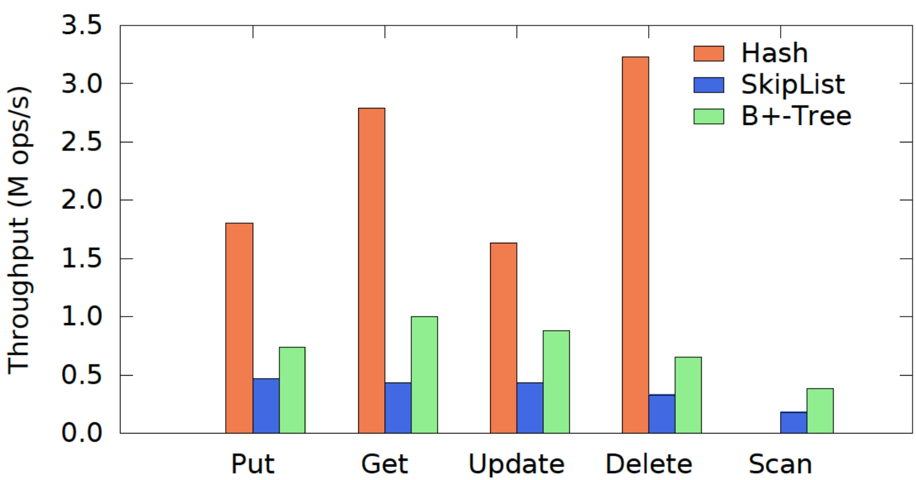
- Key-Value Store Systems (KV Store) have become an storage infrastructure of datacenters
  - Google LevelDB, Facebook RocksDB
  - Facebook, Twitter, Amazon et al. Memcached cluster
- Local file system and distributed file system use KV store to store metadata
  - Local file system： TableFS[Ren+, ATC'2013], BetrFS[Jannen+, FAST'2015]
  - Distributed file system： CephFS[Weil+, OSDI'2006]， HDFS[HDFS summit, 2015]
- Relational databases use KV as the storage engine
  - Facebook has replaced the InnoDB with MyRocks (KV store) in MySQL

Motivation

- Rich KV operations：
  - Put/Get/Delete/Update
  - Range Scan/Query(Scan)



Distribution of file system operations [Xiao+, SoCC'2015]



Neither hash nor sorted indexing can efficiently support different KV operations.

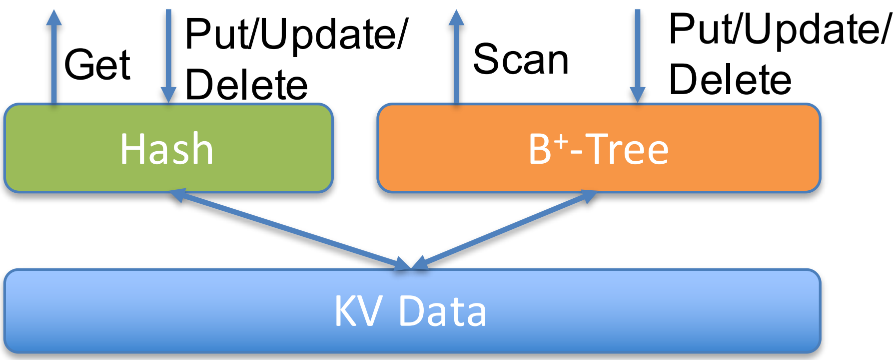
Related work

- Echo [Bailey+, INFLOW'2013]
  - Hybrid memory, Hash index
- NVStore [Yang+, FAST'2015]
  - NVM, Optimized B^+-Tree index:
    - unsorted leaf nodes
- FPTree [Oukid+, SIGMOD'2016]
  - Hybrid memory, Optimized B^+-Tree index:
    - Unsorted leaf nodes
    - Bitmap and fingerprints

All these NVM-based systems use a single index.

Hybrid index Key-Value Store (HiKV)

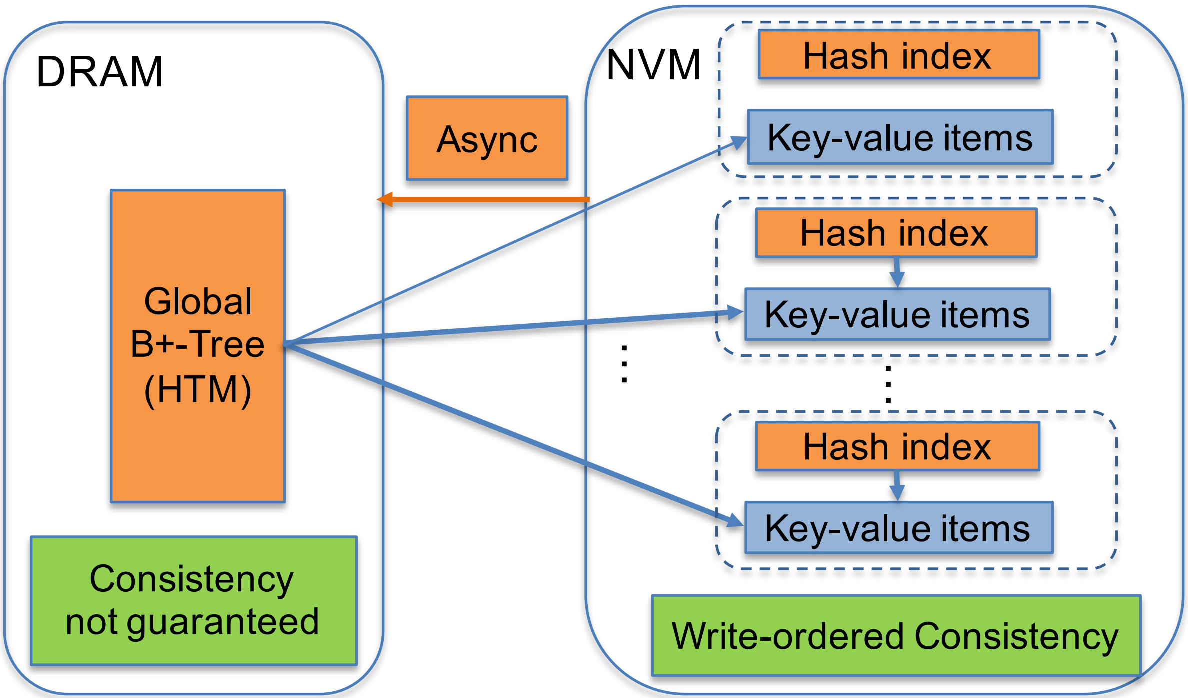
- Key idea of HiKV:
  - *Hybrid index*: Hash and B^+-Tree



- Challenges of hybrid index:
  - Latency: How to reduce the latency of Put/Update/Delete?
  - Concurrency: How to control the concurrency of hybrid index?
  - Consistency: How to guarantee crash consistency with low performance overhead?

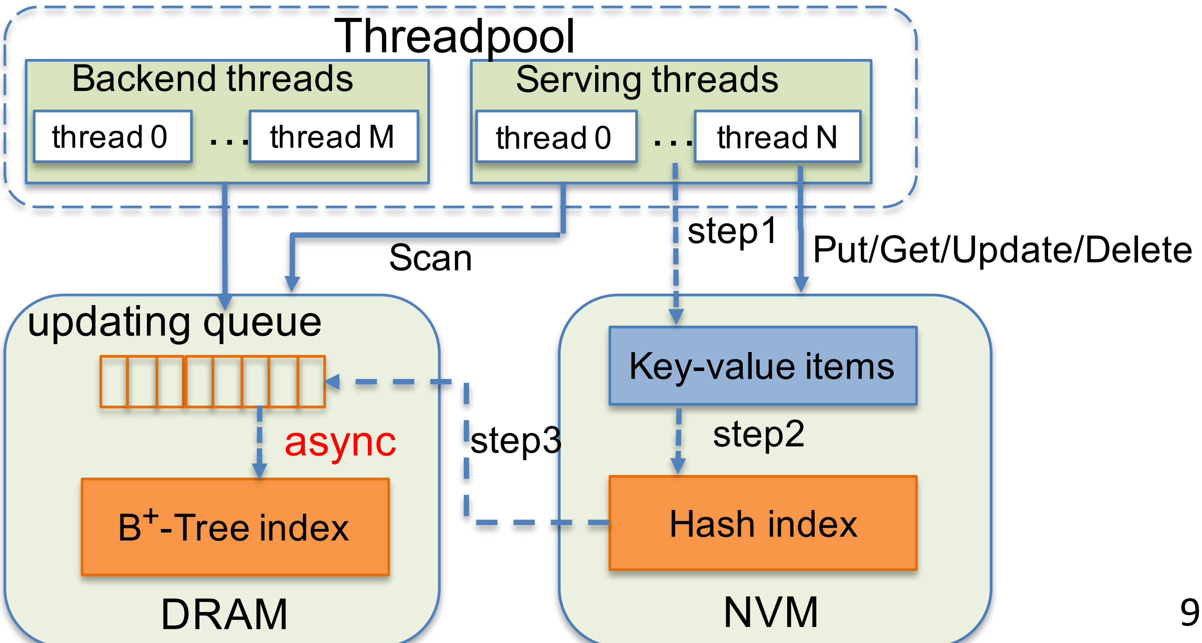
HiKV overview

- **Techniques：**
  - Asynchronous index updating
  - Differential concurrency control
  - Write-ordered consistency



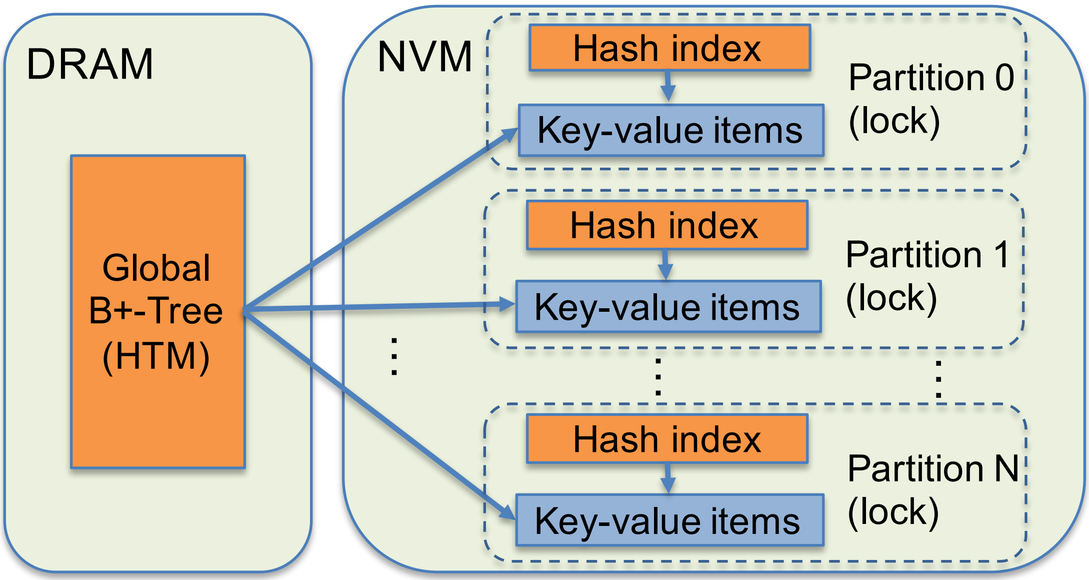
Asynchronous index updating

- Index Placement
  - Placing hash in slow NVM and B$^+$-Tree in fast DRAM
- Index Updating
  - Updating kv_item and hash index synchronously
  - Updating B$^+$-Tree asynchronously in the backend



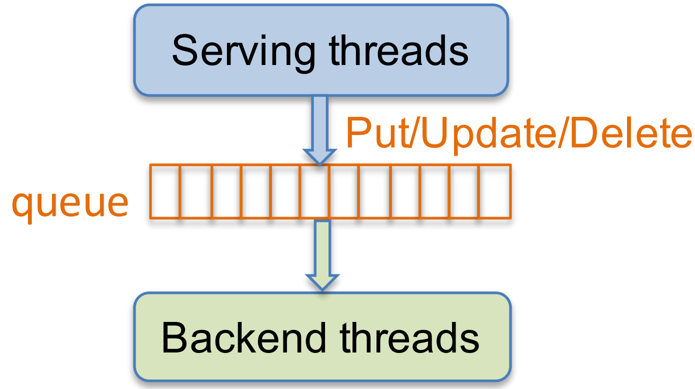
Differential concurrency control

- Hash index and KV items
  - Partitioning, fine-grained lock in partition
- Global B+-Tree index
  - Hardware Transactional Memory (HTM)



Dynamic threads adaption

- **Challenge**
  - Performance degradation in multithreaded execution

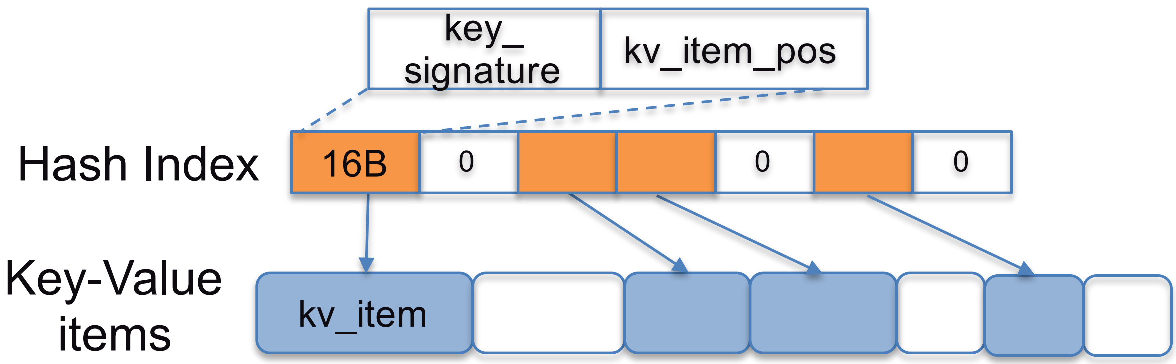


- **Solution**
  - Sample # of KV ops and their latencies
  - Dynamically adjust # of serving threads (N_{sthd}) and backend threads (N_{bthd})

$$\begin{cases} \underbrace{(N_{pd} \cdot L_{spd} + N_g \cdot L_{sg} + N_u \cdot L_{su} + N_s \cdot L_{ss})/N_{sthd}}_{\text{Filling rate}} = \underbrace{N_{pd} \cdot L_{bpd}/N_{bthd}}_{\text{Processing rate}} \\ N_{sthd} + N_{bthd} = N_{thd} \end{cases}$$

Write-ordered consistency

- Does not guarantee consistency of B^+-Tree index to reduce NVM write.
- Write-ordered consistency
  - First, update a kv item out-of-place
  - Then, update the index entry atomically

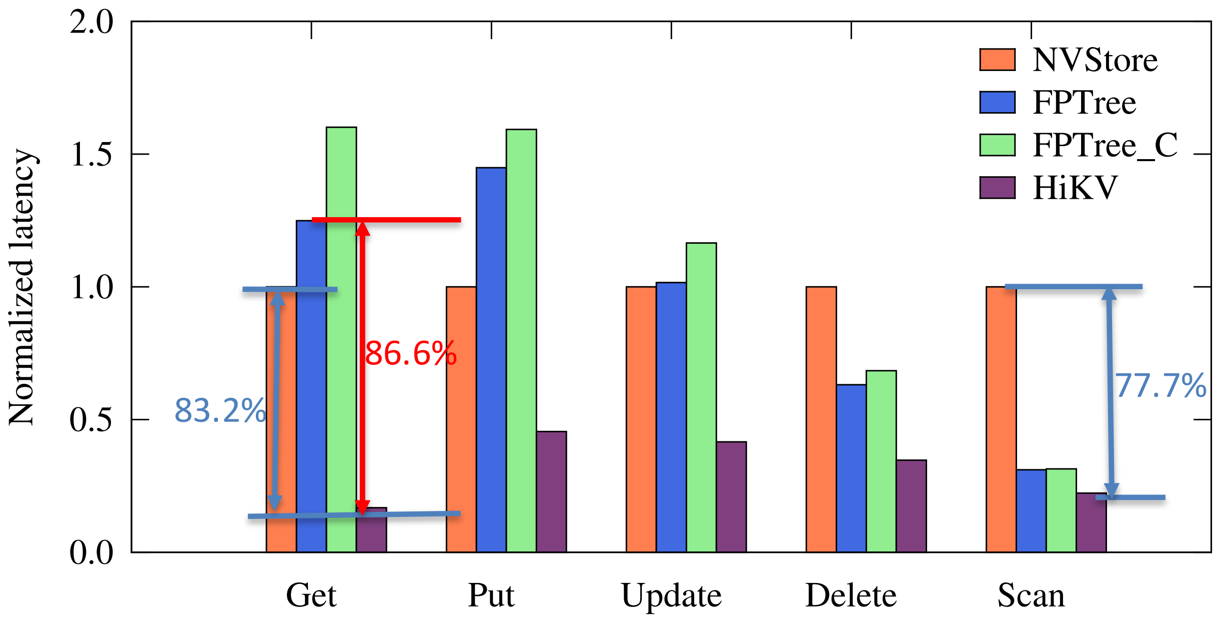


Evaluation methodology

- Platform：
  - Server：Intel Xeon E5-2620 v4
  - Emulating NVM using DRAM by adding write latency in software (600ns)
- Workloads：
  - Micro-benchmarks: Put/Get/Update/Delete/Scan
  - YCSB[Cooper+,SOCC'10]
  - 16B key, 256B value, 50M key-value items
- Compared systems：
  - NVStore[Yang+,FAST'15]
  - FPTree[Oukid+,SIGMOD'16]
  - FPTree-C: using DRAM as Cache

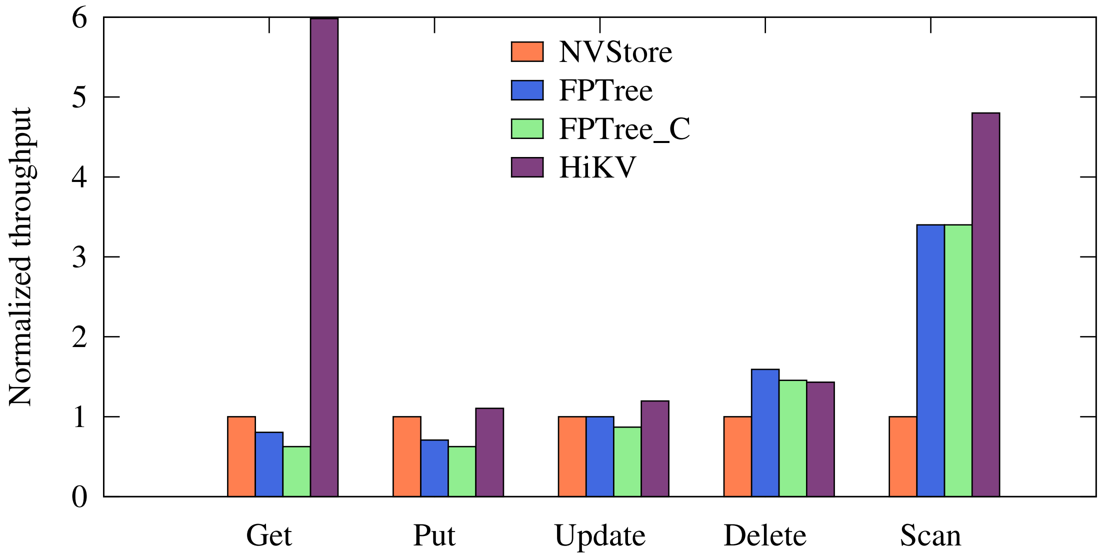
Single-threaded performance

- Latency reduction



Single-threaded performance

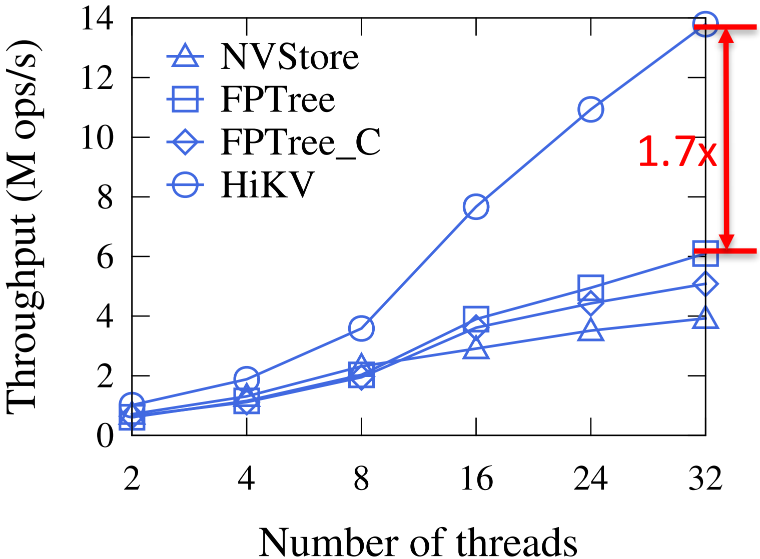
- Throughput improvement



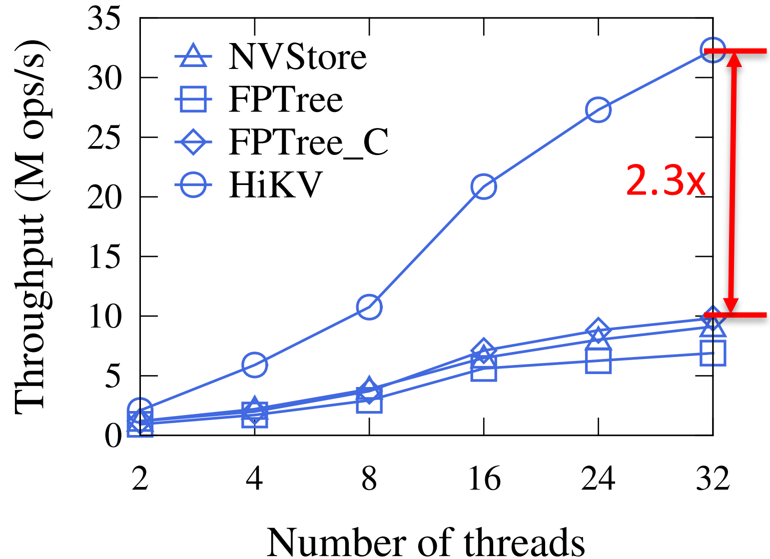
- For Get, HiKV can improve throughput by 5.0x and 6.4x than NVStore and FPTree.
- For Delete, HiKV is 10.0% lower than FPTree due to one serving thread.

Scalability

- Throughput of YCSB-A/B

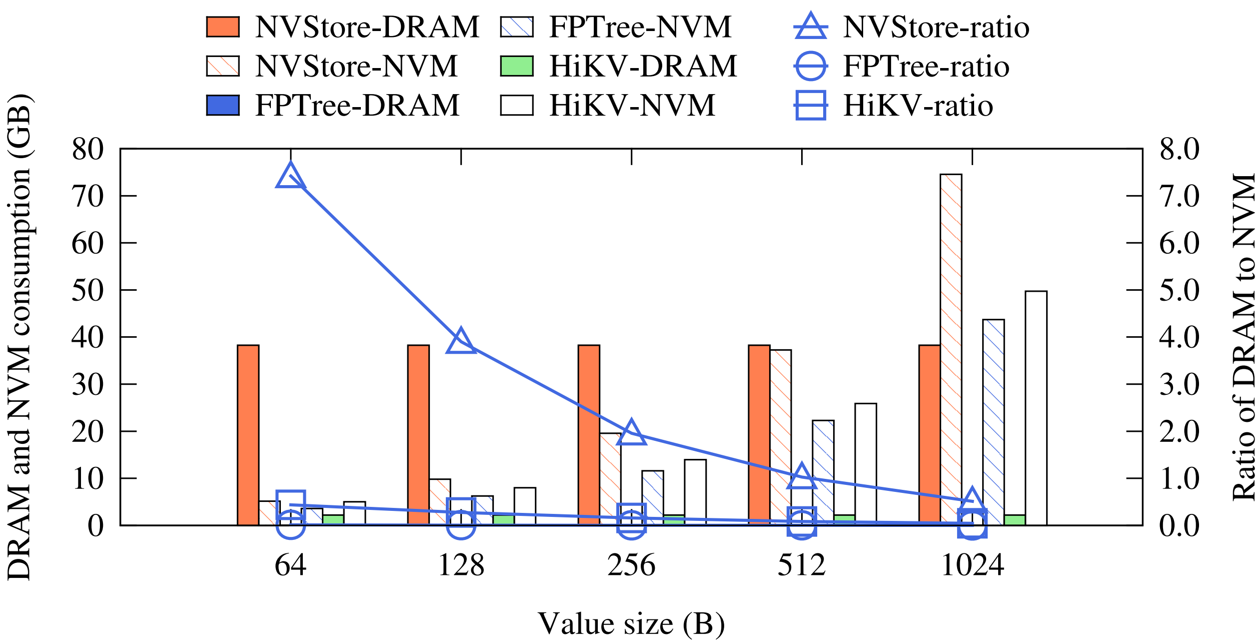


YCSB-A: 50%Get-50%Update



YCSB-B: 95%Get-5%Update

DRAM Consumption



- For 256B value, HiKV-ratio is 15.8%, while FPTree-ratio is 0.4%.
- Reducing the DRAM consumption is our future work.

Recovery time

- Recovering 50M key-values

system	Time (s)
NVStore	11.0s
FPTree	1.7s
HiKV-1thread	88.2s
HiKV-4thread	23.1s
HiKV-16thread	6.3s

- HiKV takes longer recovery time than NVStore and FPTree due to unsorted hash index.

Summary

- Hybrid DRAM and NVM memory is a promising solution for future storage system.
- A single index employed in existing NVM-based KV stores can not efficiently support all KV operations.
- This work proposes a *hybrid index for hybrid memory systems* to serve different KV operations.
- *HiKV* based on hybrid index outperforms the start-of-art NVM-based KV stores.

Thanks for your listening!

Q & A

中国科学院计算技术研究所

INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES