HotStorage '20
JULY 13-14, 2020

SplitKV: Splitting IO Paths for Different Sized Key-Value Items with Advanced Storage Devices

Shukai Han, Dejun Jiang, Jin Xiong
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences

Outline

- ✓ **Background & Motivation**
- Design
- Evaluation
- Conclusion

Key-Value Store

- Key-Value (KV) stores are widely deployed in data centers

RocksDB LEVELDB redis MEMCACHED

- The sizes of KV items vary from a couple of bytes to hundreds of kilobytes
  - Facebook's analysis on Memcached's workload found that more than 80% of requests are less than 500B in size[1].
  - The workload data on a typical day in Baidu: over 90% of requests are over 128KB in size[2] .

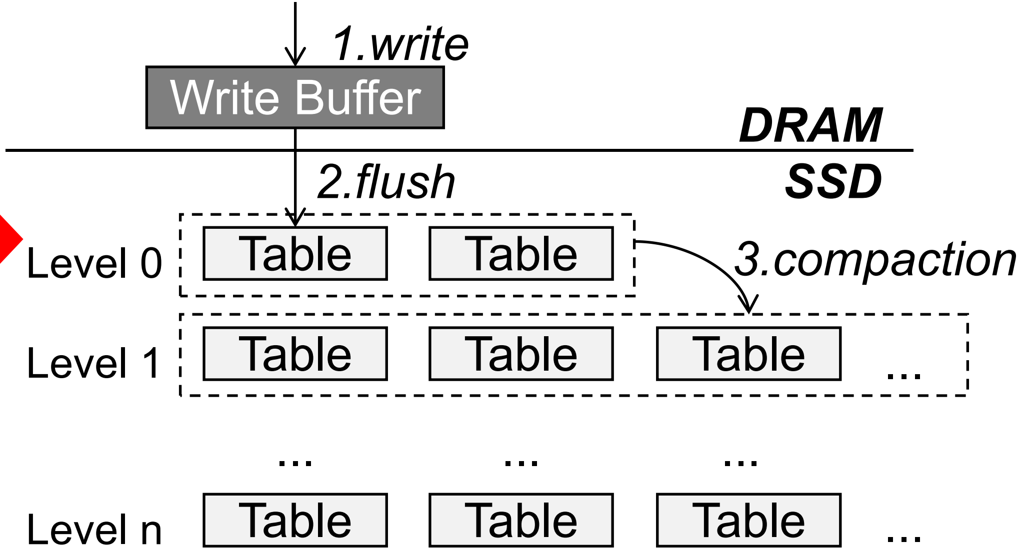
[1] Berk, SIGMETRICS '2012
[2] Lai, MSST '2015

Conventional Storage Device based KV Store

Conventional Storage Devices

- Block access
- Low random access performance

Log-Structured Merge (LSM) Tree based KV Store



Log Structured Merge Tree is widely adopted in KV stores to convert random writes to sequential writes.

Advanced Storage Device based KV Store

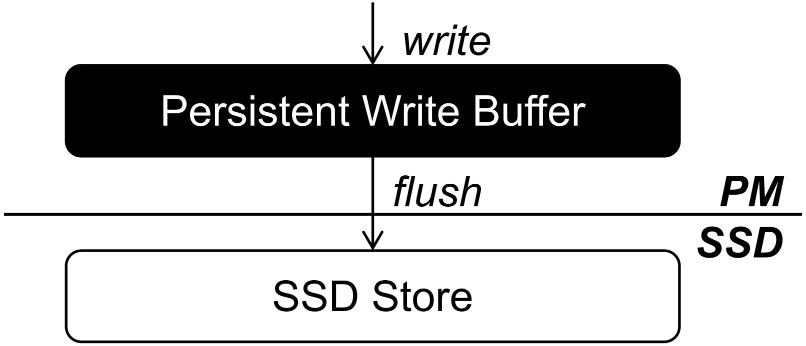
Optane SSD

Optane DC PMM

Advanced Storage Devices

- PM:Byte access
- SSD: Block access
- High random access performance

- KVell[3] builds low CPU overhead Key-Value Store Based on Modern SSDs
- Some works[4] based on the low latency characteristics of PM, in which persistent buffers are built to reduce the logging overhead.



[3] Lepers, SOSP'19
[4] Kannan, ATC'18

Motivation

Random Write	64B	256B	1KB	4KB	16KB	64KB	256KB	1MB	4MB	16MB
Optane SSD P3700	14.09	14.09	14.09	14.09	21.44	45.79	145.58	532	2091	8223
Optane DC PMM	0.18	0.20	0.43	1.05	3.90	15.50	61.88	247	1440	6840
Ratio	79.2	70.5	33.0	13.4	5.5	2.9	2.4	2.2	1.45	1.2

- PM is friendly to small KV items
- NVM based SSD is friendly to large KV items without suffering from random access cost

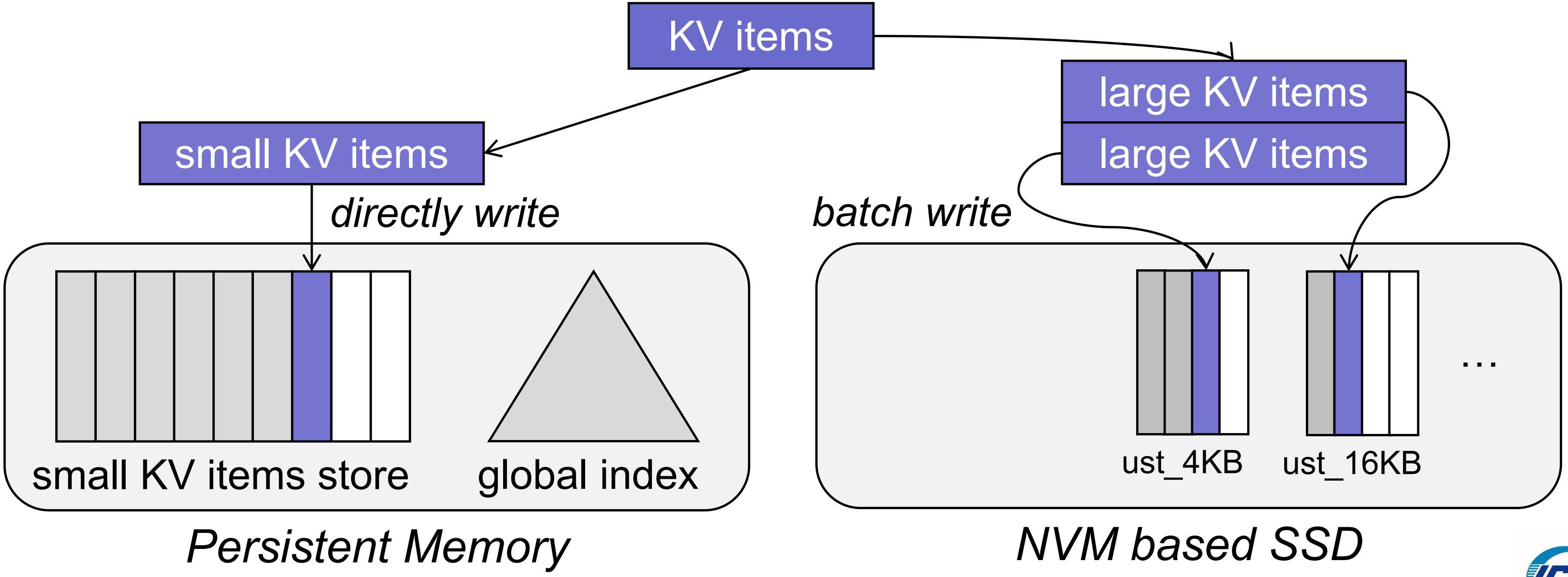
?

Outline

- Background & Motivation
- ✓ **Design**
- Evaluation
- Conclusion

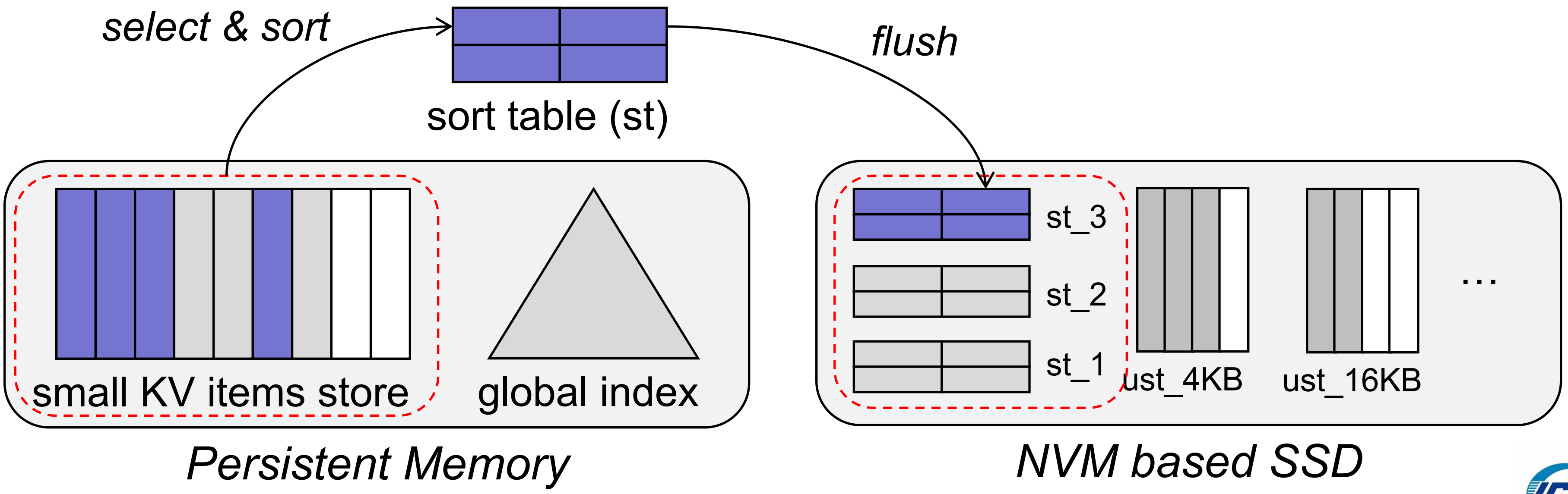
SplitKV Overview

Key idea: Splitting IO Path for small/large KV items



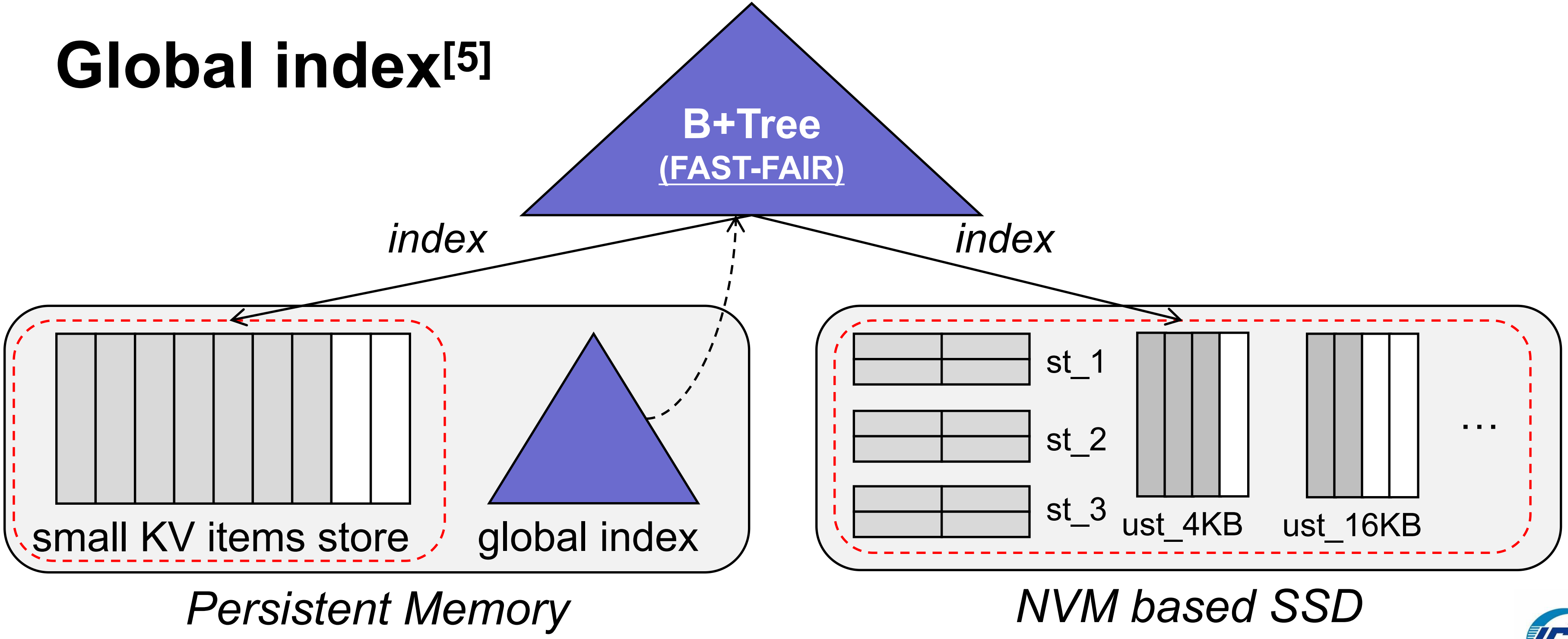
SplitKV Overview

Reclaim PM space



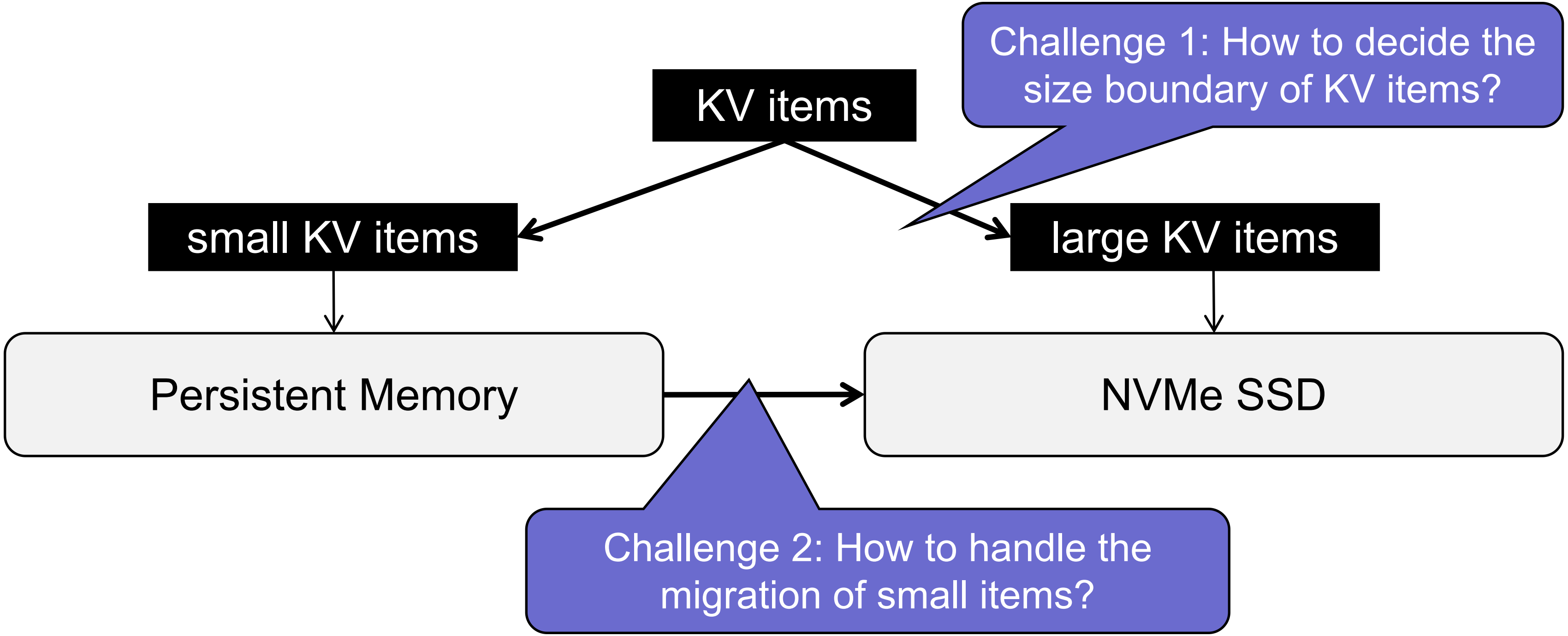
[5] Hwang, FAST'16

SplitKV Overview



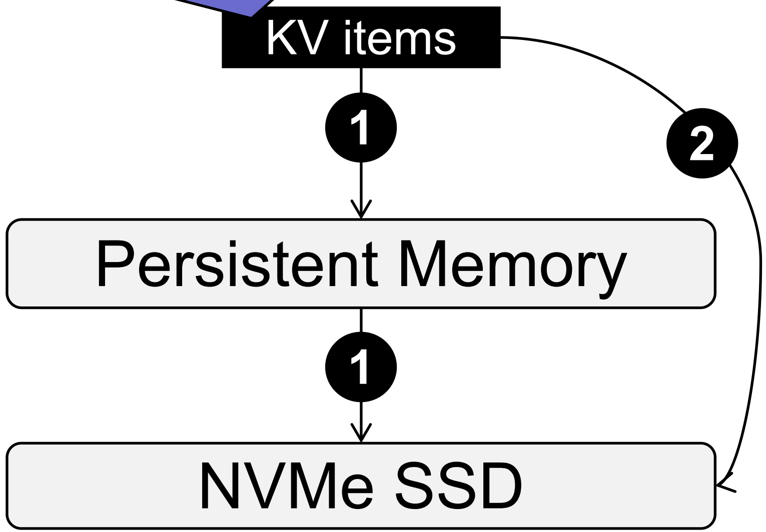
[5] Hwang, FAST'16

Design challenges



Size Boundary of KV Items

IO Path 1: KV is written to PM and then migrated to SSD through a background thread.
IO Path 2: KV is directly written to SSD.

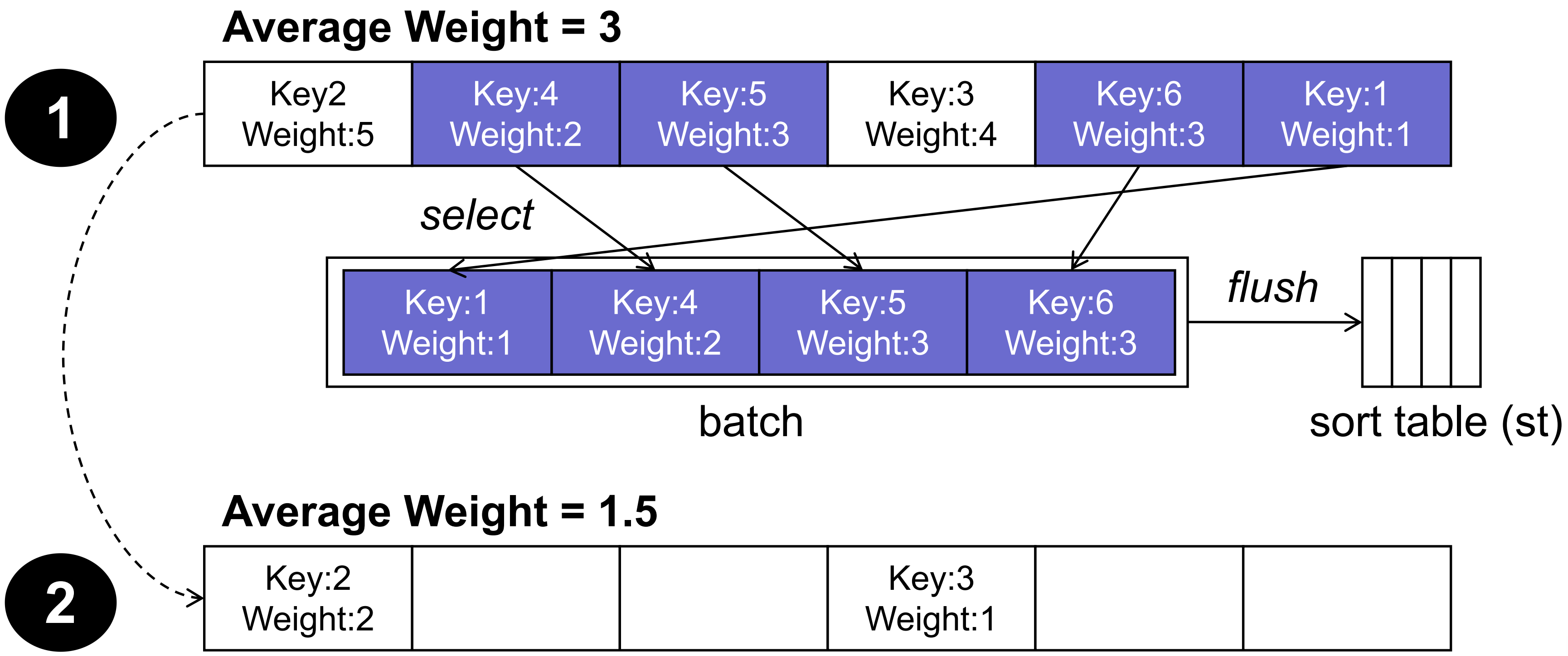


Access Size	256B	1KB	4KB	16KB
IO Path 1	1.5	4.5	15.7	27.6
IO Path 2	23.4	25.4	14.8	21.3
Ratio	15.8	5.7	0.9	0.8

Write latencies (us) of different IO path

- When the KV item size is large, the data is written directly to the SSD for better performance.
- Any KV pair whose size is equal to or greater than 4 KB is considered to be large one.

Hotness-aware KV Migration



Outline

- Background & Motivation
- Design
- ✓ **Evaluation**
- Conclusion

Experiment Setup

- **System and hardware configuration**
  - Server equipped with two Intel Xeon Gold 5215 CPU (2.5GHZ)
  - 64GB memory, one Intel Optane SSD P4800 and one Intel Optane DC PMM
  - CentOS Linux release 7.6.1810 with 4.18.8 kernel
- **Compared systems**
  - RocksDB、NoveLSM[4]、KVell[3]
- **Workload**
  - YCSB with zipfan and unifrom skew
  - Each workload handles 128 GB data set
  - 50% of the KV items are 256B/4KB in size

Workload	Description
A	50% reads and 50% updates
B	95% reads and 5% updates
C	100% reads
D	95% reads for latest keys and 5% inserts
E	95% scan and 5% inserts
F	50% reads and 50% read-modify-writes

[3] Lepers, SOSP'19
[4] Kannan, ATC'18

Average Latency with Single Thread (Zipfan)

zipfan	A	B	C	D	E	F
NoveLSM	48.35	34.89	30.52	32.28	445.83	72.57
RocksDB	17.47	21.82	21.72	21.13	497.02	35.19
KVell	11.76	8.60	8.64	9.20	609.38	14.12
SplitKV	3.81	4.65	4.56	4.56	306.65	5.05

For workloads A and F, SplitKV reduces latency by 14.4x, 6.9x, and 3.1x compared to NoveLSM, RocksDB and KVell under zipfan workloads.

Average Latency with Single Thread (Zipfan)

zipfan	A	B	C	D	E	F
NoveLSM	48.35	34.89	30.52	32.28	445.83	72.57
RocksDB	17.47	21.82	21.72	21.13	497.02	35.19
KVell	11.76	8.60	8.64	9.20	609.38	14.12
SplitKV	3.81	4.65	4.56	4.56	306.65	5.05

For read-intensive workloads B, C and D, SplitKV and KVell achieved better performance than NoveLSM and RocksDB due to the adoption of the global B+-Tree index.

Average Latency with Single Thread (Zipfan)

zipfan	A	B	C	D	E	F
NoveLSM	48.35	34.89	30.52	32.28	445.83	72.57
RocksDB	17.47	21.82	21.72	21.13	497.02	35.19
KVell	11.76	8.60	8.64	9.20	609.38	14.12
SplitKV	3.81	4.65	4.56	4.56	306.65	5.05

For workload E, KVell does not sort small KV items in SSD. This introduces read amplification to KVell when serving scan query by reading a plenty of blocks.

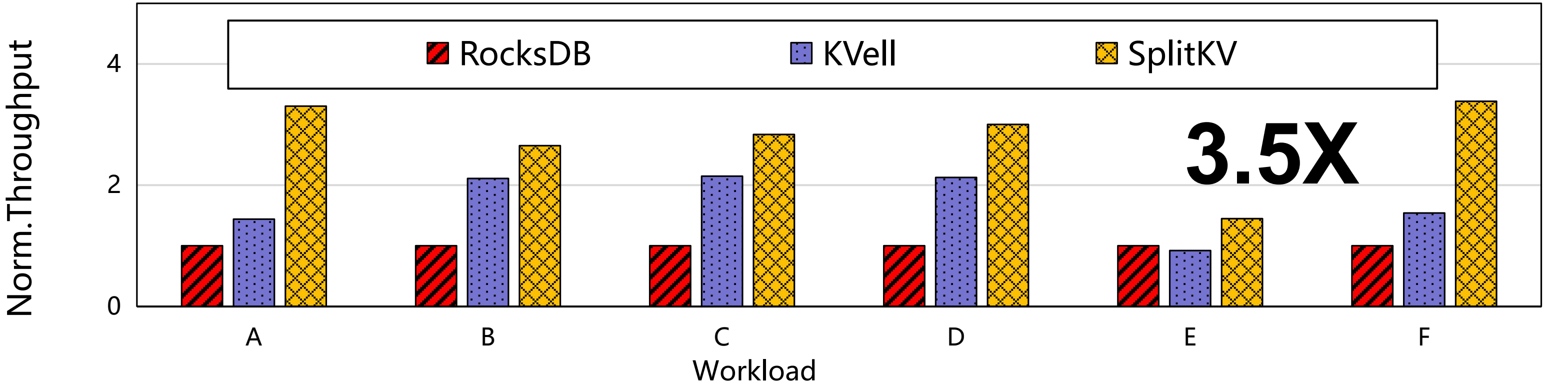
Average Latency with Single Thread (Zipfan .vs Uniform)

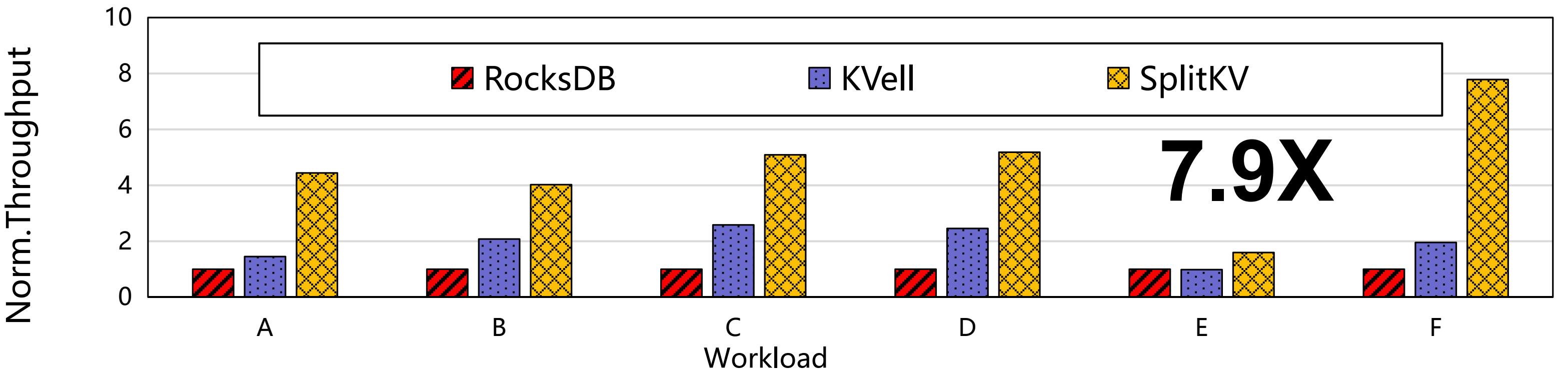
zipfan	A	B	C	D	E	F
NoveLSM	48.35	34.89	30.52	32.28	445.83	72.57
RocksDB	17.47	21.82	21.72	21.13	497.02	35.19
KVell	11.76	8.60	8.64	9.20	609.38	14.12
SplitKV	3.81	4.65	4.56	4.56	306.65	5.05

uniform	A	B	C	D	E	F
NoveLSM	96.69	69.77	61.04	64.56	476.19	145.14
RocksDB	21.11	26.13	26.08	25.89	529.10	43.27
KVell	17.86	14.02	13.31	13.80	670.69	23.09
SplitKV	8.81	12.78	12.77	9.22	346.02	13.87

Note that, the hotnessaware migration policy is difficult to figure out cold items under uniform workloads.

Throughput in YCSB with Four Threads





Outline

- Background & Motivation
- Design
- Evaluation
- ✓ Conclusion

Conclusion

- Modern NVMe SSD and persistent memory provide different access features when serving small/large data.
- We propose SplitKV to provide different IO paths for different sized KV items for building KV stores with such advanced storage devices.
- The throughput of SplitKV is up to 7.9 times that of other KV stores under zipfan load skew.

THANK YOU !

Q & A

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Author Email: hanshukai@ict.ac.cn