KVAccel: A Novel Write Accelerator for LSM-Tree-Based KV Stores with Host-SSD Collaboration

Kihwan Kim¹, Hyunsun Chung¹, Seonghoon Ahn¹, Junhyeok Park¹, Safdar Jamil¹, Hongsu Byun¹, Myungcheol Lee², Jinchun Choi², Youngjae Kim¹







The 39th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Milan, Italy, June 3-7, 2025



- Background
- Motivation
- Design
- Evaluation
- Conclusion



Background

LSM-tree based Key-Value Stores



- Log-Structured Merge-Tree(LSM-tree)
 - Designed for write-intensive workloads
 - Optimized for large-scale data
 - Out-of-place updates
 - Sequential batch operations







RocksDB [1]

^{[1]:} Facebook, "RocksDB" https://rocksdb.org, 2012

^{[2]:} Google, "LevelDB" https://github.com/google/leveldb, 2017

^{[3]:} Meta, "ZippvDB" https://engineering.fb.com/2021/08/06/core-infra/zippvdb/, 2021

LSM-tree based Key-Value Stores



- LSM KVS(e.g. RocksDB) stores data in an append-only manner in the active MemTable
- Data in MemTable is moved to and managed on disk through background jobs(Flush, Compaction)

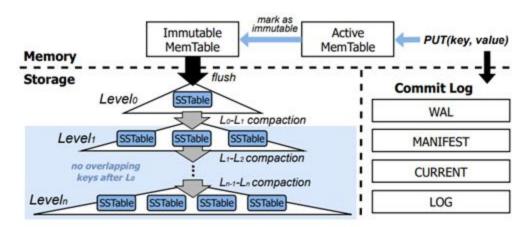


Fig. 1: An architecture of LSM-tree.

Write Stall Problem



- Write Stall: write operation blocked, due to bottlenecks in Flush,
 Compaction
- In RocksDB, Write stall occurs under these 3 scenarios_{[4][5]}
 - Incoming Writes > Flush
 - Flush > Level 0 to Level 1 Compaction
 - Pending deep level compaction size becomes heavier

^{[4]:} SILK: Preventing Latency Spikes in Log-Structured Merge Key-Value Stores, Oana Balmau et al., USENIX ATC'19

^{[5]:} ADOC: Automatically Harmonizing Dataflow Between Components in Log-Structured Key-Value Stores for Improved Performance, Jinghuan Yu et al. (USENIX FAST'23)

Background **Evaluation** Conclusion Motivation Design

Existing Work: ADOC_[5]



- In three types of overflow scenarios, ADOC alleviates write stalls by adjusting two tuning knobs
- Two tuning knobs: # of Compaction threads, MemTable size

	# of Compaction Threads	MemTable Size
Incoming Writes > Flush	↓	↑
Flush > Level 0 to Level 1 Compaction	1	
Pending deep level compaction size becomes heavier	1	—

Background Design **Evaluation** Conclusion Motivation

Existing Work: ADOC_[5]



- In three types of overflow scenarios, ADOC alleviates write stalls by adjusting two tuning knobs
- Two tuning knobs: # of Compaction threads, MemTable size

- 1. Not an immediate remedy → Write stalls still occur
- 2. Requires Slowdown methods while accelerating compaction







Motivation

Observation 1. Slowdowns_[6]: The Inefficient Write Stall Solution



- RocksDB uses the slowdown_[6] method to prevent user writes from becoming completely blocked.
- The state of the art solution ADOC_[5] also uses slowdowns.

→ Both RocksDB and ADOC_[5] ultimately fall back to using slowdown to avoid a write stall.

^{[5]:} ADOC: Automatically Harmonizing Dataflow Between Components in Log-Structured Key-Value Stores for Improved Performance, Jinghuan Yu et al. (USENIX FAST'23)

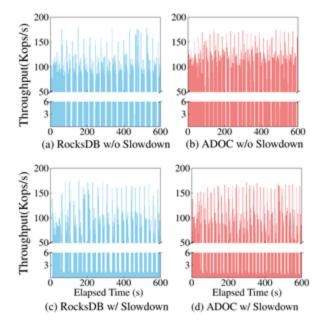
^{[6]:} https://github.com/facebook/rocksdb/wiki/Write-Stalls

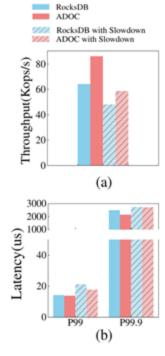
Observation 1. Slowdowns_[6]: The Inefficient Write Stall Solution



Slowdowns, while preventing a complete write stall from occurring,

harms overall performance.

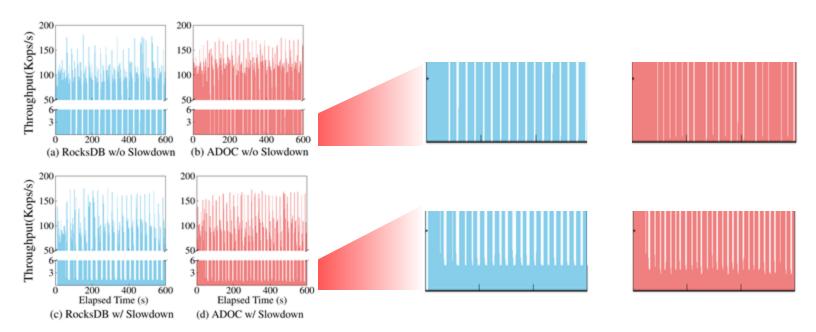




Observation 1. Slowdowns_[6]: The Inefficient Write Stall Solution



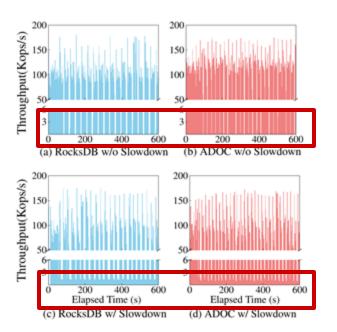
• *Slowdowns*, while preventing a complete write stall from occurring, harms overall performance.



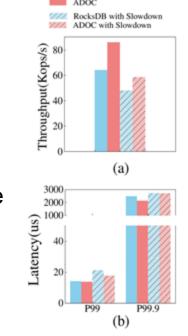
Observation 1. Slowdowns_[6]: The Inefficient Write Stall Solution



 Slowdowns, while preventing a complete write stall from occurring, harms overall performance.



I/O service is uninterrupted thanks to slowdowns preventing write stalls...



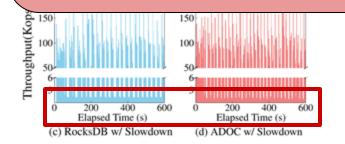
...At the cost of overall throughput and latency.

Observation 1. Slowdowns_[6]: The Inefficient Write Stall Solution

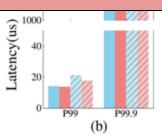


• *Slowdowns*, while preventing a complete write stall from occurring, harms overall performance.

Both state-of-the-art and industry-standard solutions employ write slowdowns to prevent write stalls, which can sharply degrade over throughput and significantly increase tail latency.



stalls...

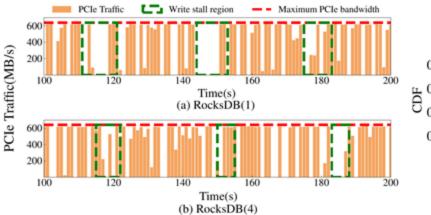


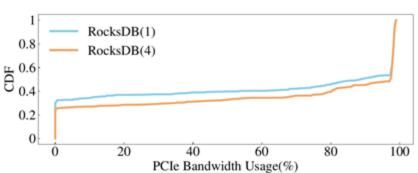
RocksDB with Slowdown

Observation 2. Under-utilization of PCIe Bandwidth



 PCIe Traffic drop sharply during a write stall, implying inefficient device resource usage.

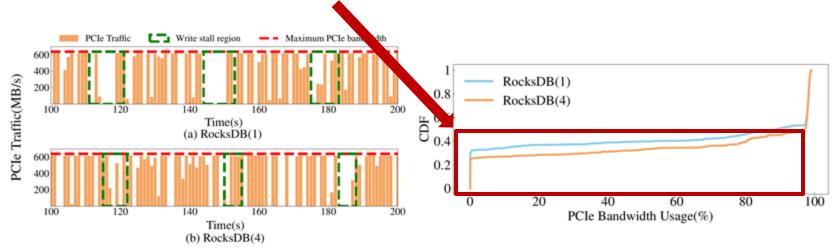




Observation 2. Under-utilization of PCIe Bandwidth



- PCIe Traffic drop sharply during a write stall, implying inefficient device resource usage.
 - RocksDB is shown to leave up to 90% of available PCIe bandwidth around 50% of the time during a write stall.

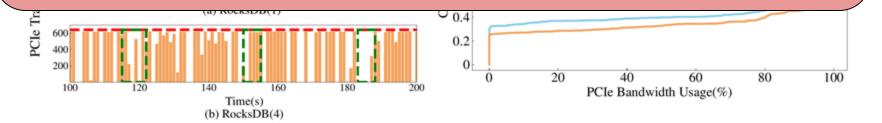


Observation 2. Under-utilization of PCIe Bandwidth



 PCIe Traffic drop sharply during a write stall, implying inefficient device resource usage.

PCIe bandwidth is under-utilized during write stalls in industry standard LSM-KVS due to the compaction operation blocking device I/O.



The status quo



• Observation 1. ultimately leads to the following options for write stalls.

Slowdowns

VS

Allowing Write Stalls

- Maintains I/O service at all times
- Overall throughput and latency penalty due to said slowdowns

- Overall throughput and latency conserved
- Complete interrupts in I/O service as write stalls are allowed to occur.
- **Observation 2.** reveals an unexploited resource to help mitigate write stalls and increase performance without sacrificing system resources: underutilized PCIe and device bandwidth during write stalls.

The status quo



• Observation 1. ultimately leads to the following options for write stalls.

Slowdowns

VS

Allowing Write Stalls

Maintaina I/O coming at all Overall throughout and

Can write stalls be mitigated without sacrificing system resources by leveraging underutilized PCIe and device bandwidth during write stalls?

and increase performance without sacrificing system resources: underutilized PCIe and device bandwidth during write stalls.



Proposed Solution: KVAccel

Proposed Solution: KVAccel



 KVAccel's design is based on two key factors: Disaggregation and Aggregation.

Disaggregation

- Division of SSD into hybrid interface (block and keyvalue) and its required I/O paths
- Maintenance of each interface's separate LSM-Tree

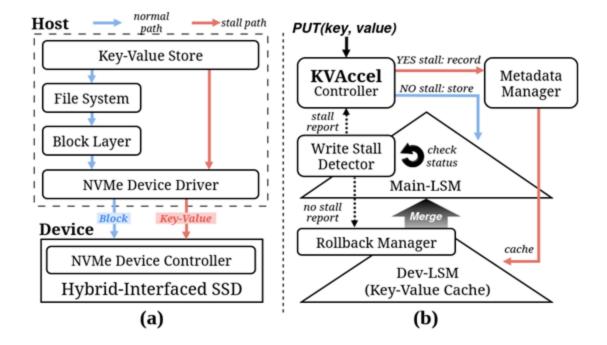
Aggregation

- Manage data from each interface as if it was one database instance
- Unify separate I/O commands and database state with rollback

Overview of **KVAccel**



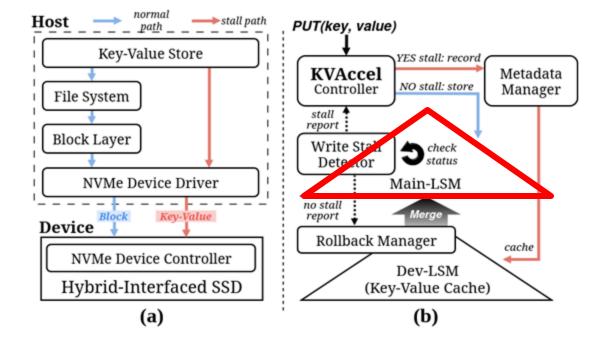
- Co-Design of Hardware & Software provides 2 I/O paths
- Different I/O paths taken based on the presence of a write stall



Overview of **KVAccel**



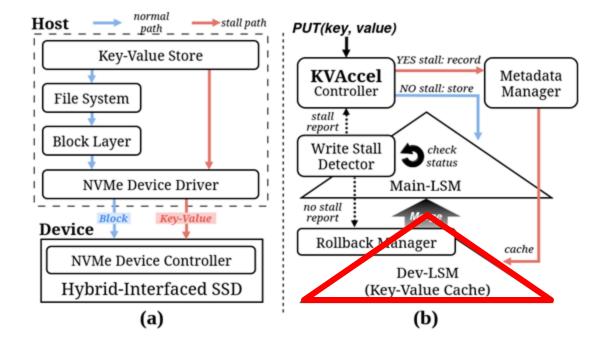
- Co-Design of Hardware & Software provides 2 I/O paths
- Different I/O paths taken based on the presence of a write stall



Overview of **KVAccel**



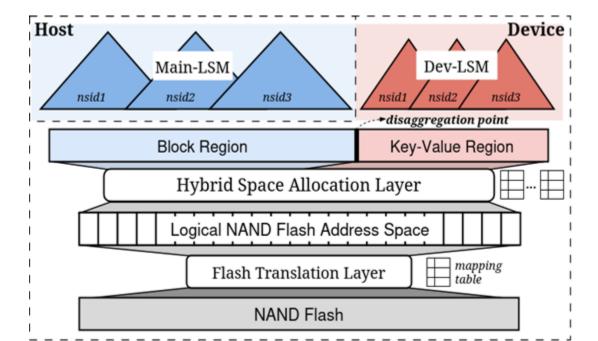
- Co-Design of Hardware & Software provides 2 I/O paths
- Different I/O paths taken based on the presence of a write stall



Hybrid Dual-Interface SSD

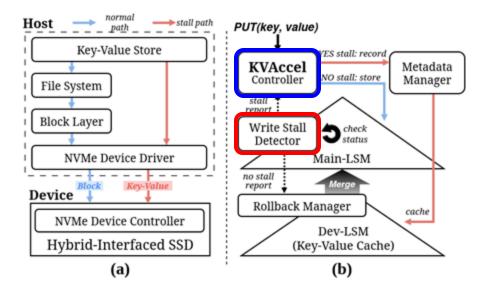


- Hybrid interface SSD achieved by logical NAND flash address disaggregation via a specified address boundary
 - SSD issues different commands for each interface



Software Modules(1)





Detector

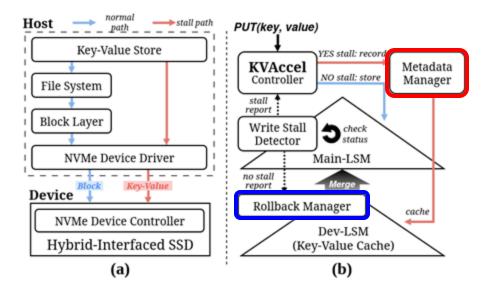
- Detects write stalls checking3 components
 - # of Level 0 SSTs
 - Memtable size
 - Pending compaction size

Controller

 Directs I/O commands to the correct interface based on the Detector's output.

Software Modules(2)





Metadata Manager

 Keeps track of KV pairs located in Dev-LSM via a hash table for membership testing

Rollback Manager

 Initiates and performs the rollback operation based on the rollback scheduling policy and the Detector's output.

Rollback Operation: Scheduling



- Rollback refers to return the KV pairs in Dev-LSM back to Main-LSM into one LSM-KVS instance.
- Rollback operation can be scheduled eagerly or lazily based on workload characteristics.

Eager Rollback

- Perform rollback as soon as there are enough resources available (by using L₀ file count threshold)
- Ideal for a read orientated workload to avoid slow Dev-LSM read operations

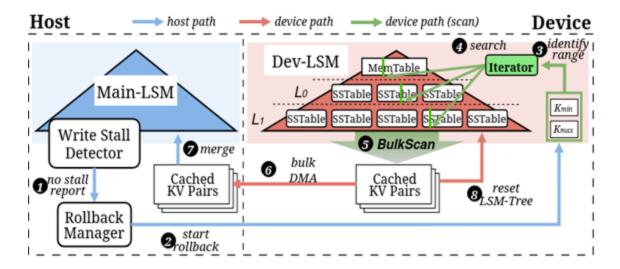
Lazy Rollback

- Delay rollback until the current write workload is completely finished
- Ideal for a write intensive workload to lower interference of rollback with write operations

Rollback Operation



- To accelerate rollback, KV pairs are read in bulk using a range scan operation.
- Iterator reads Dev-LSM in its entirety and serializes the KV pairs.
- KV pairs are then sent to the host by performing DMA multiple times.



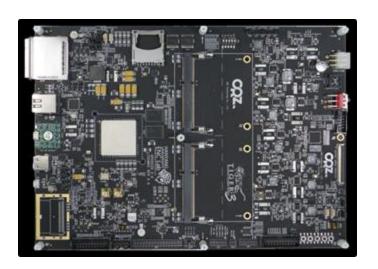


Evaluation

Evaluation Setup

Testbed:

KV-SSD on Cosmos+ OpenSSD Platform_[7]



oiscos 🚭

TABLE I: Specifications of the OpenSSD platform.

SoC	Xilinx Zynq-7000 with ARM Cortex-A9 Core
NAND Module	1TB, 4 Channel & 8 Way
Interconnect	PCIe Gen2 ×8 End-Points

TABLE II: Specifications of the host system.

CPU Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz (32 co		
Memory	384GB DDR4	
OS	Ubuntu 22.04.4, Linux Kernel 6.6.31	

LSM-KVS and Benchmark Configurations



TABLE III: LSM-KVS configurations. For all figures, the numbers next to each LSM-KVS refer to compaction thread count. For KVACCEL, the settings refer to the Main-LSM.

LSM-KVS	Compaction Threads (n)	MT Size
KVACCEL(n)	1	
	2	
	4	
RocksDB(n)	1	
	2	128 MB
	4	
ADOC(n)	1	
	2	
	4	

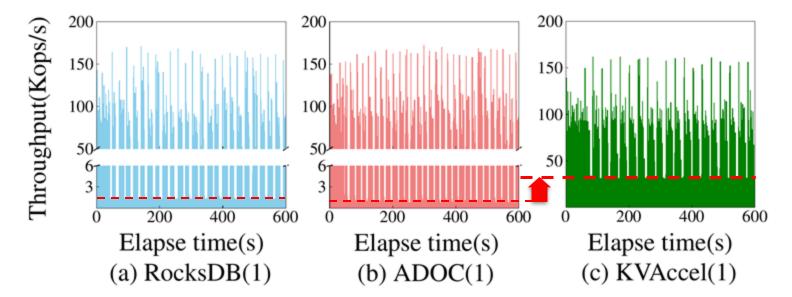
TABLE IV: db_bench^{8} workload configurations. Each benchmark was run with a 4 B key and 4 KB value size. Workload A,B,C were run for 600 seconds, and Workload D performed 60K read operations.

Name	Type	Characteristics	Notes (write/read ratio)
A	fillrandom	1 write thread	No write limit
В	readwhilewriting	1 write thread	9:1
С	readwiniewriting	+ 1 read thread	8:2
D	seekrandom	1 range query thread	Run after initial
		(Seek + 1024 Next)	20GB fillrandom

Write Stall Avoidance



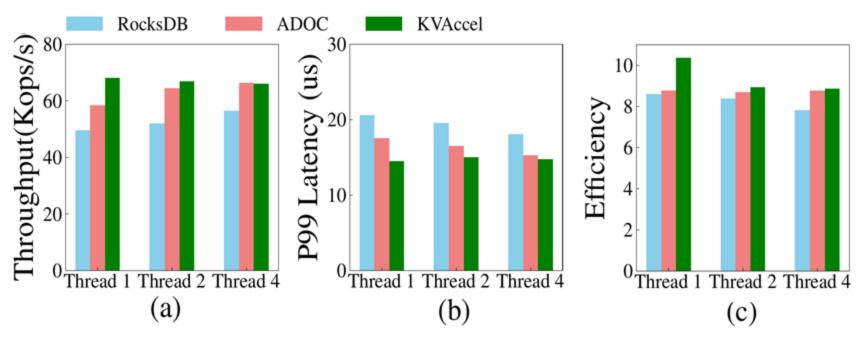
• Throughput minimum values greatly increased, as *KVAccel* is designed to allow as much throughput as the SSD and system allows without slowdowns.



Performance Evaluation



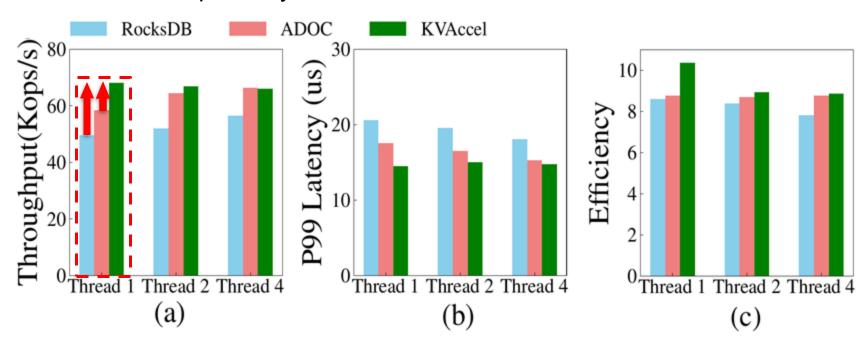
• (a) Throughput, (b) P99 Latency, (c) Efficiency



Performance Evaluation



- (a) Throughput
- KVAccel shows at most a 37% and 17% improvement over than RocksDB and ADOC, respectively.

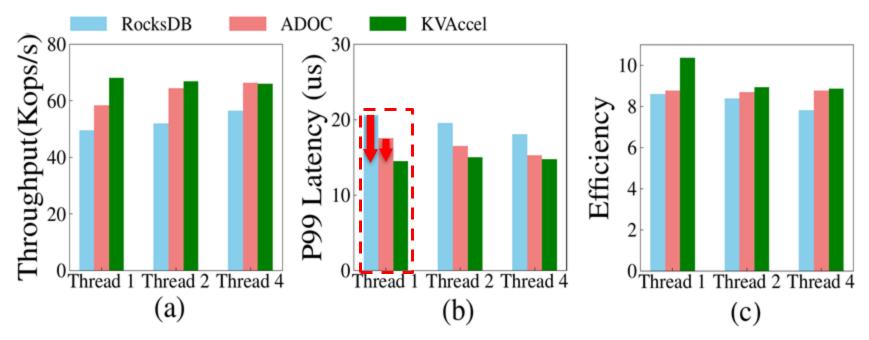


Performance Evaluation

DISCOS &

(b) Throughput

 Maximum of 30% and 20% decrease in latency was also observed between KVAccel and RocksDB, ADOC, respectively.



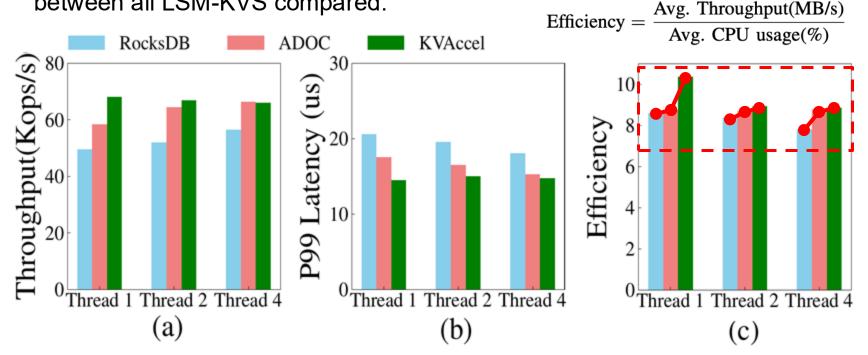
Performance Evaluation



(c) Efficiency

KVAccel maintains the better efficiencies in host machine's resources

between all LSM-KVS compared.

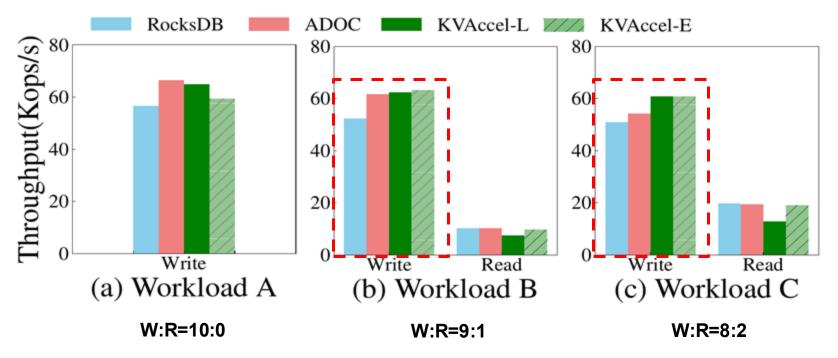


Rollback Policies Evaluation

DISCOS

Eager vs Lazy Rollback analysis

 From (b) and (c), we observe that it still outperforms RocksDB and ADOC under read-oriented workloads

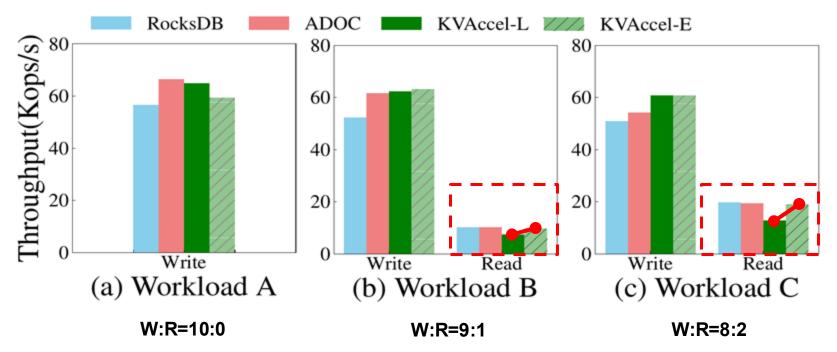


Rollback Policies Evaluation

DISCOS 🚭

Eager vs Lazy Rollback analysis

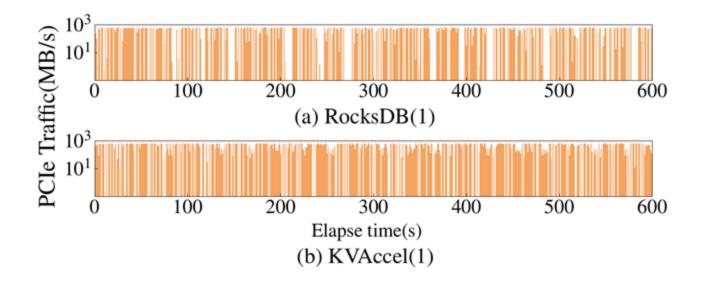
 As the read ratio increases, Eager Rollback becomes increasingly advantageous



PCIe Traffic Usage



- More available PCle traffic exploited
- KVAccel takes advantage of its dual interface and demonstrate higher PCIe utilization over RocksDB.







Conclusion



- Prior work addresses write stalls to a limited extent
 - Hardware and software are treated in isolation
- **KVAccel** achieved a 17% improvement in throughput and a 20% reduction in latency compared to ADOC.
- KVAccel demonstrates the effectiveness of hardware-software codesign
 - Alleviates write stalls by utilizing:
 - Under-used PCIe bandwidth
 - Computational capabilities within SSDs



Thank you!

Contact

- Kihwan Kim / lewis461@sogang.ac.kr
- Hyunsun Chung / hchung1652@sogang.ac.kr
- Seonghoon Ahn / ok10p@sogang.ac.kr
- Data-Intensive Computing & Al Systems Laboratory https://discos.sogang.ac.kr/



<Camera-ready paper> Can be found on Google Scholar

KVACCEL: A Novel Write Accelerator for LSM-Tree-Based KV Stores with Host-SSD Collaboration

Kilmus Kim¹⁴, Hyumus Chang¹⁴, Snoughous Alm¹⁴, Joshyook Park¹, Saltar Sanif Hongra Byer¹, Myong-Serd Lee², Finches Clot², Youngise Kim^{1,2} Tour. of Conquery Science and Engineering, Sugary University, Senal, Republic of Kinna

Advisors dup threshold Marys (LOM) tree based Key Nobe: comparine formin during a sole standard, foreity telluling those AVNs; are widely adopted for their high performance in comparine duration. However, AXXX: increases loss CVV. wite talegate contraspose, but they after their performance dependents due to write stade during compaction. Felor solutions, such an expliciting I/O traffic or using analogic compaction thinks, can come assumpted dright in thoughput at themps:
good, Province Microry (PM)-based drogs in thoughput at themps:
that CPC stops, while hardness-speed approaches using EPGs, Justice votes in PSI before floating than to the LSM-see.
USFs, and SPC stated at reflecting comparison formers before before the SPCs, but an accessment CPC-1-CL, CPC (SPC SPC-1-CL), and Signal ANDS finds ages to support both thesis and key-sides interfaces, using the key-sides interface as a foregoing writer of CCS and fit systems contribute. Although these agreements lattle alternal seven made. The meaning significantly relation with rattle, agriculture insome sugge, and among containing lattices this, agriculture insome sugge, and among containing lattices the late and firsts by highwarding as in effects MM dated enti-liabilities with an investment and range over a section. One outside black with an investment and range over a section. One outsiders reductes dues that for with intensive medicals, \$750.00. say performance degradation due to manages performe or orkhols, bols dimenstrate comparable performance. Judio Termo-Key-Yalise State, Log-Structured Morge Tree, hald from Error, Wyder Stall Milligation

Log-Structured Marger (LEM) tree-based Key-Yalon Street (KVS) sciences, such as Books DB [1] and LevelDB [1], are commonly used in with introduce applications that to their ability to handle high-throughput wittes efficiently. However, LSM hased KVIn (LSM-KVIn) often experience performance department for exercised for some delay comparison? So (1). These write sight block incoming write operations, resulting in a eigenfusion enducions in throughput and an increase in tall latency, which undermose system reliability in timeentities workleads.

To alleriate note stalls, many software based solutions have hore replaced and dephysiol. Burkel & [1], one of the most wishrly used LSM-KVS, respirements a mechanism known as directives (%). This showdown standardon anticipates promise ente staffs and progetredy reduces the wire presence on the LSM-EVS. White absolutes can provent write stalls, it may amountarily domain the throughput of BooksCR by Leating. the write pressure directed to the LIMISTS. Additionally, the state of the set solution ADOC [1] estigates write stalls by dynamically increasing bank sizes and the number of

"This are this or patient and have constituted equally.

stitution by employing multiple compaction decads.

Abernatively, bandware based solutions have been presi

apartersa ABOC to up to 17% in turns of throughput and . increased host CPU usage, while hardware solutions require obsesses to CPC officiality efficiency for extent read with additional hardware majors come in the mark, an arrangement a grandworking approach that aready write mally without compromising KVS performance, minimizes here CPU utilization, and requires an additional bandware costs. Our method expression a new paradigm that in fundamentally different from entering approaches, by actively treeraging afte resources in staining storage devices to aread write stafts while minimizing Sout CPU involvement.

In this paper, we present EVS/VIII. a second beheld bandware software producing framework that irretages a new shall interface KIID architecture to militario write stalls and optimis the activation of energy bandwidth. EVACCO, to built on the observation that sharing boot-oals with malls, the and/of/reg stance Arriva's probable IRS highwidth streams understricts dragate to princeted to bands additional DO operations. ENh: (11) then incorporates a dynamic I/O redirection mechanic that monitors the status of boot-outs LSM-RVS and, open detecting a writer staff, aboth worken from the LSM AVVS to the deployments have realized senter further

KYNCKE, process a disapprepares of the SMYs Inguis-NANC Suit abless space total two regions, one for the traditional block interface, which is essenged by the look-side LSM KVS, and another for the key rules interface improved by like KY-SSD, which serves as a temporary write halfor to serve pending sinte request by bypassing the traditional LDM hazed sheri math sharing malfu-

To maintain consisency between the main LDM on the have and the write hafter on the device, KNRCCO, introduces