STORAGE DEVELOPER CONFERENCE SDC 23

BY Developers FOR Developers

KV-CSD

An Ordered, HW-Accelerated KV Store For Rapid Data Insertion and Queries

Qing Zheng, Scientist, Los Alamos National Laboratory (LANL)

LA-UR-23-30273

A Collaboration with SK hynix

Overview

Problem

Scientific data analytics often slowed down by unordered, unindexed data access

Goal

Leverage computational storage to sort and index data at rest

KV-CSD

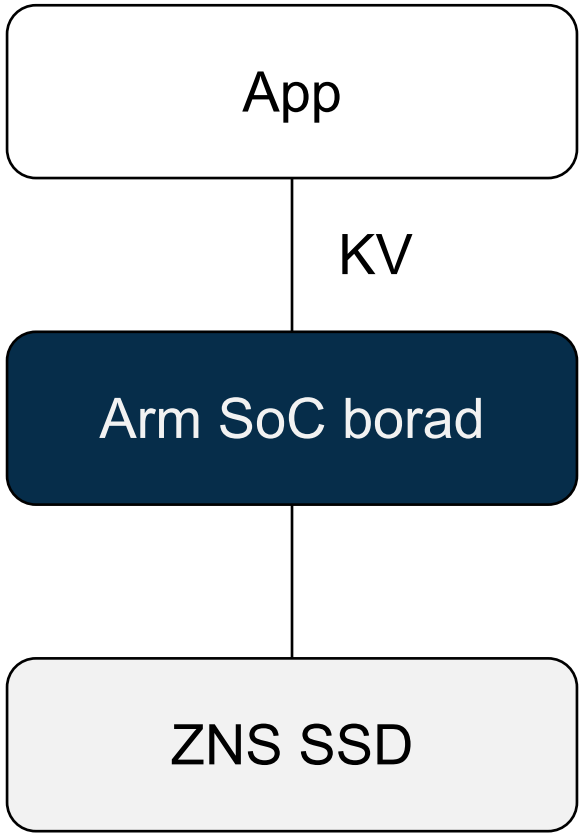
An ordered, hardware-accelerated KV store for rapid data insertion and queries

A Quick Look

Two components: (1) an arm SoC board, (2) a ZNS SSD

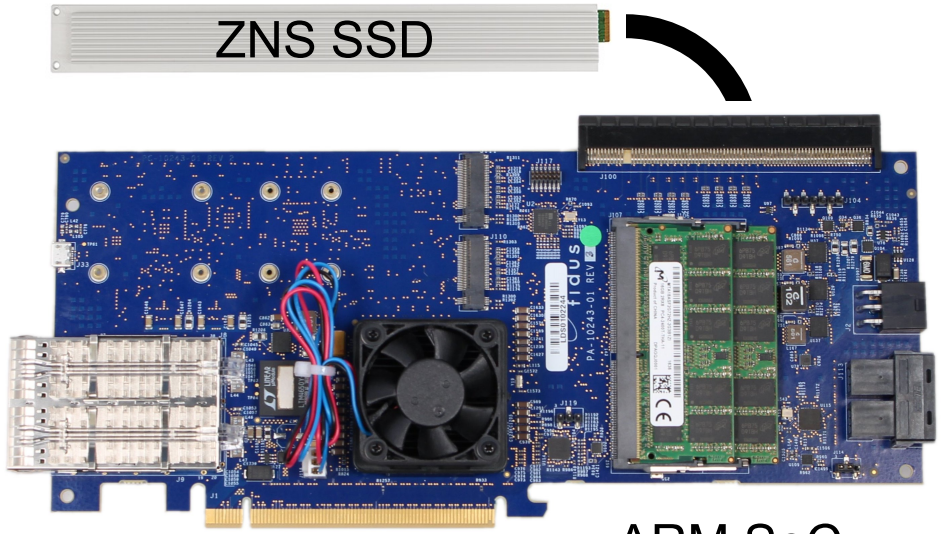
The arm board implements KV atop SSD zones

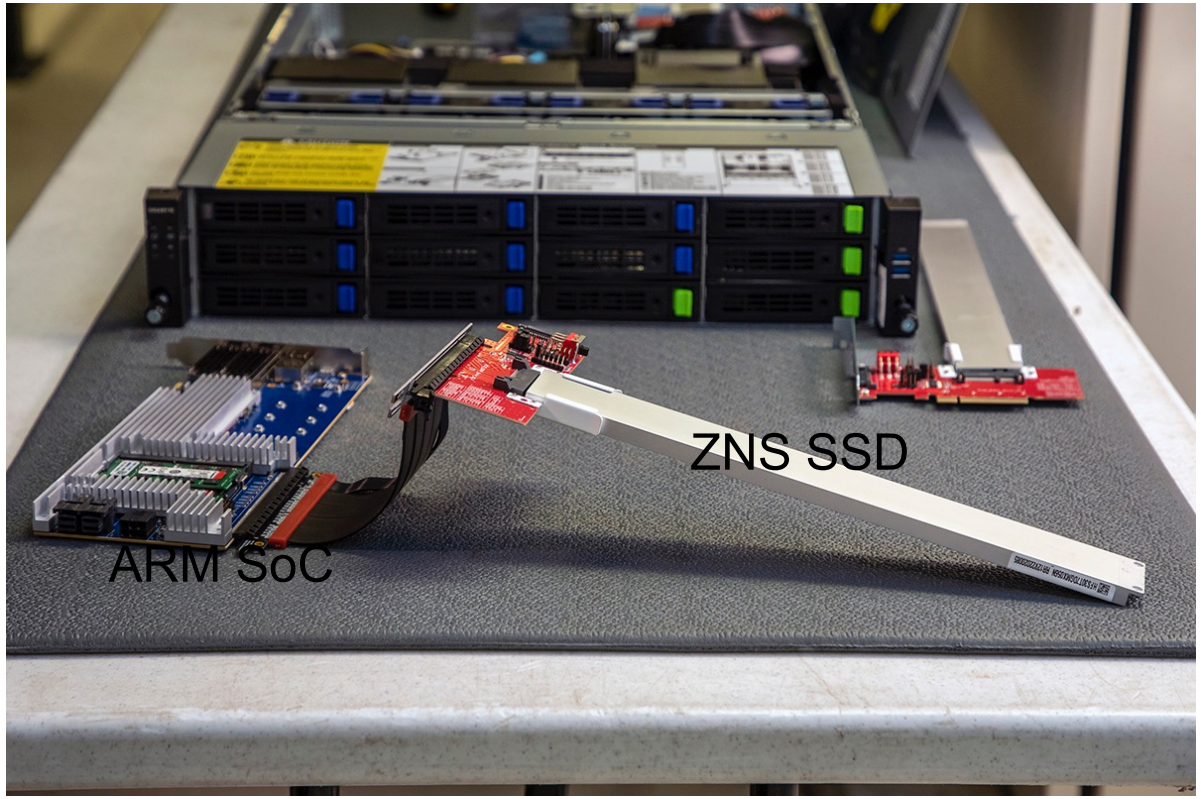
Apps use custom NVMe KV commands for bulk data insertion, index creation, and queries



KV-CSD in Real World

Current Prototype





Today's Talk

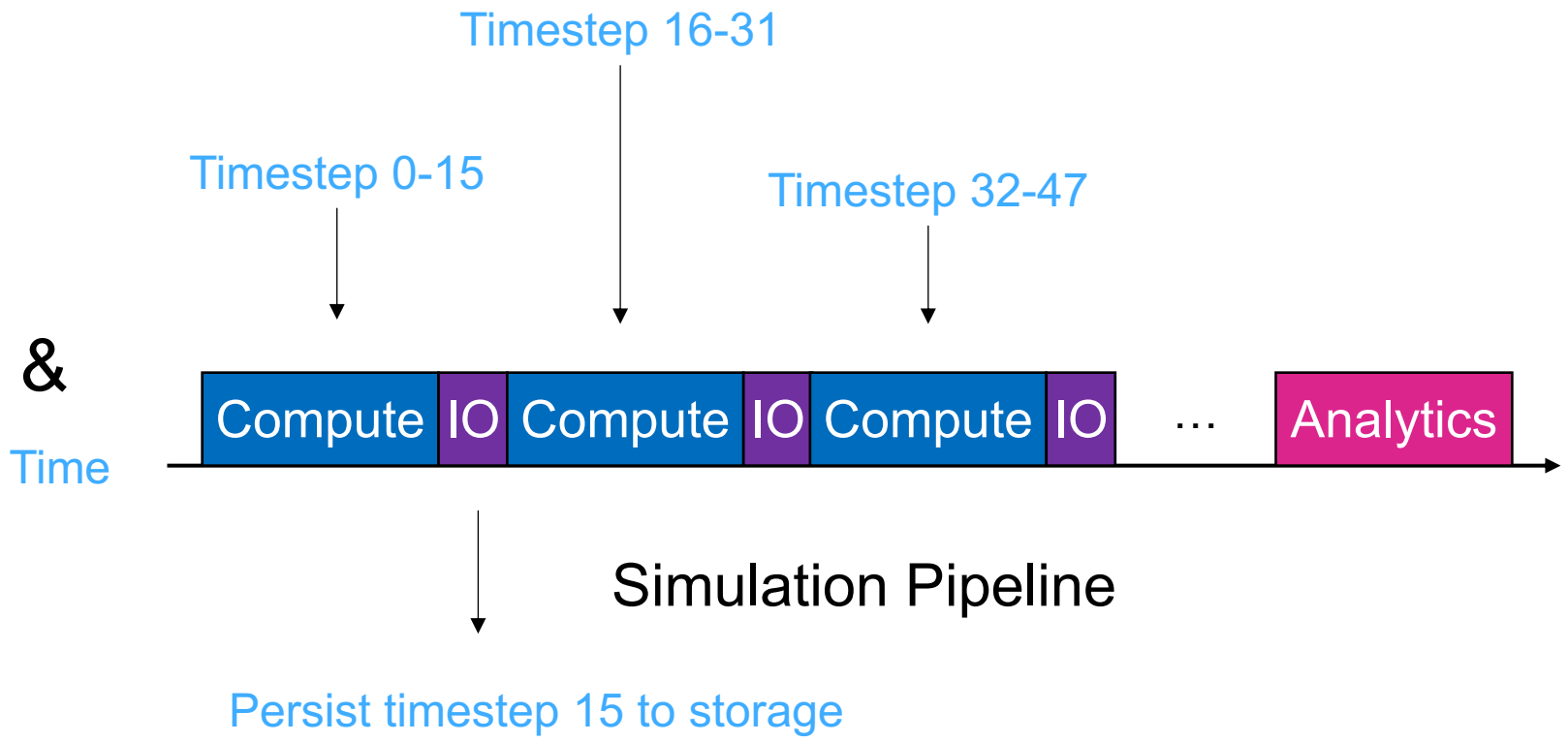
- Why ordered computational KV storage?
- How does it work?

How Scientific Simulations Run

Time based bulk-synchronous parallel programs

Iterate between compute & I/O phases

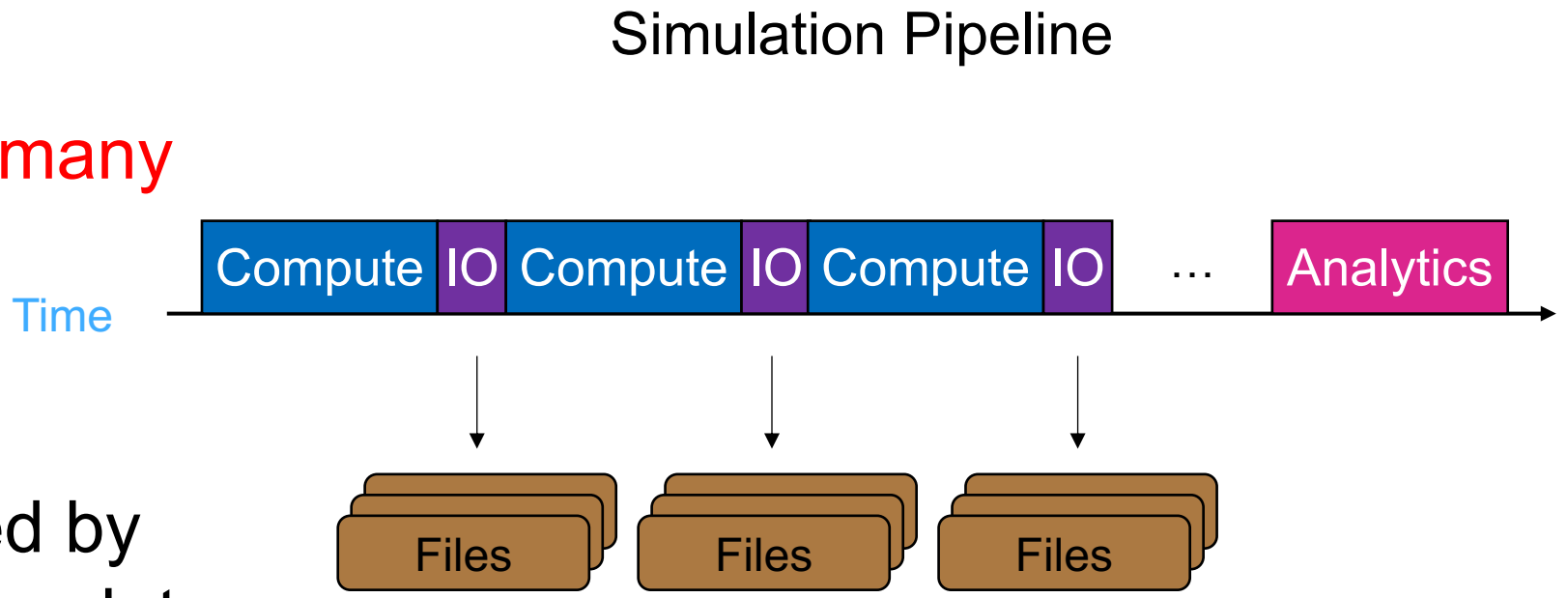
Analytics occur after simulation



How Data is Stored Today

Through filesystems

Data stored as one big or many small files per timestep



Data typically accompanied by metadata that describes the data (type, dimension, …)

Problem: Queries Often Read More Data Than Necessary

Data may not be persisted in the same order as queries, leading to full data scans

Pre-sorting data prior to queries using many compute nodes can be equally inefficient

Computational storage offers new ways of acceleration

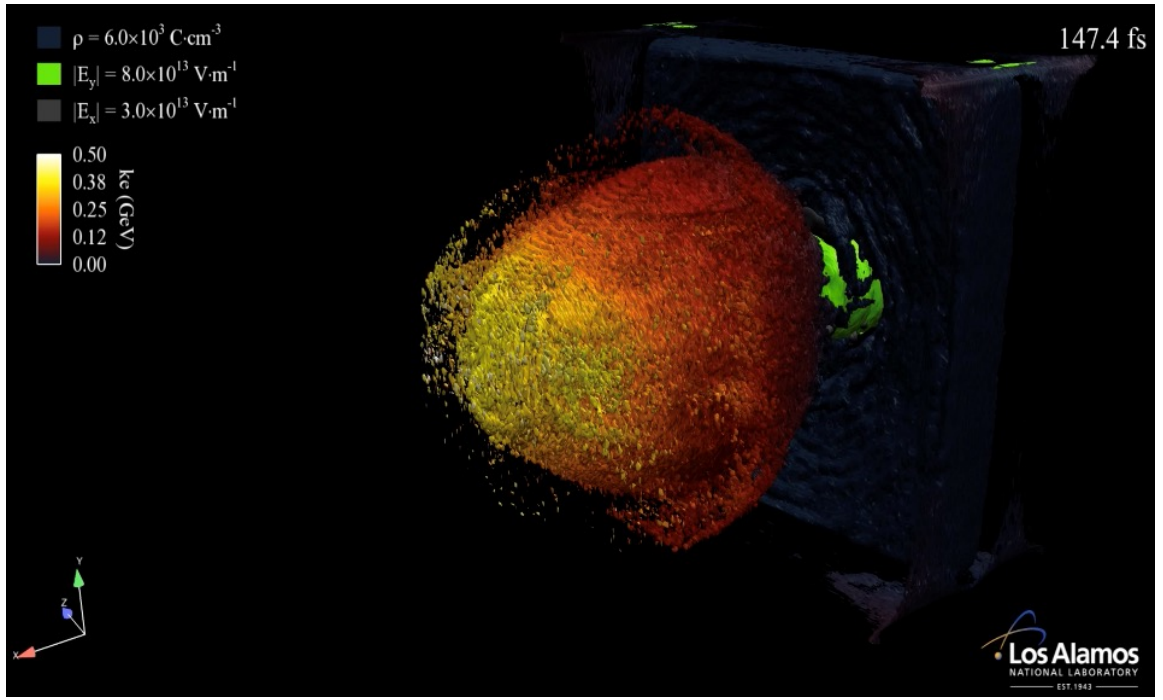


Image from LANL VPIC simulation done by L. Yin, et al at SC10

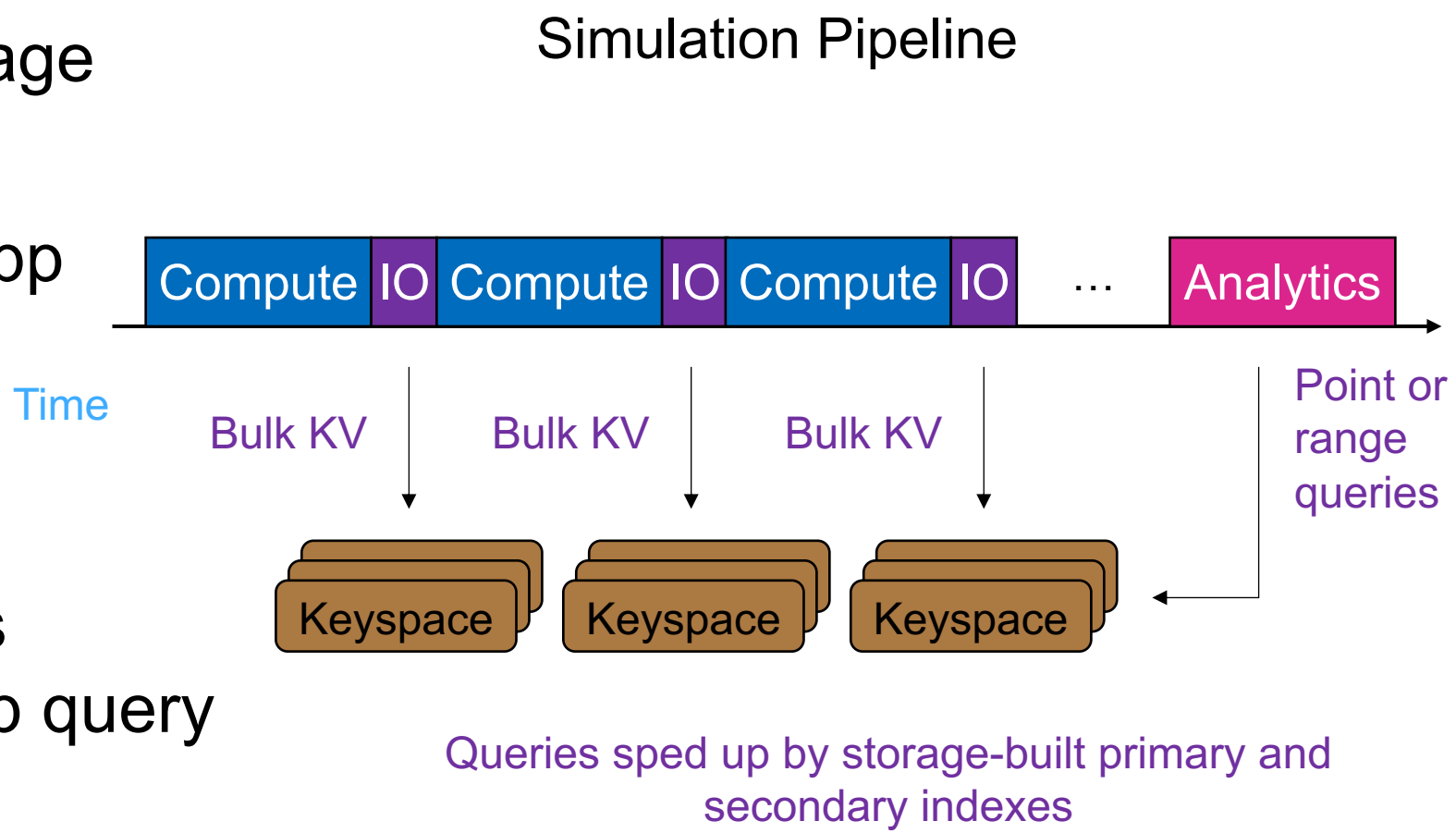
For example: a simulation may store its particles in particle ID order, but queries may target their energy levels

Toward Ordered, Computational KV Storage

App converts data to KV pairs and bulk inserts them into storage

One KV namespace per app process per timestep

Storage sorts data by key asynchronously and builds secondary indexes per app query needs



Why KV?

- Scientific data often resembles records with keys and values
- KV interface already very popular thanks to open software like RocksDB
- KV provides sufficient knowledge of data without having to resort to external metadata

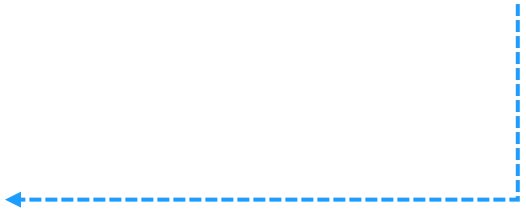
Switching from files to KV not awfully difficult

No need to map filenames to LBAs to enable offload

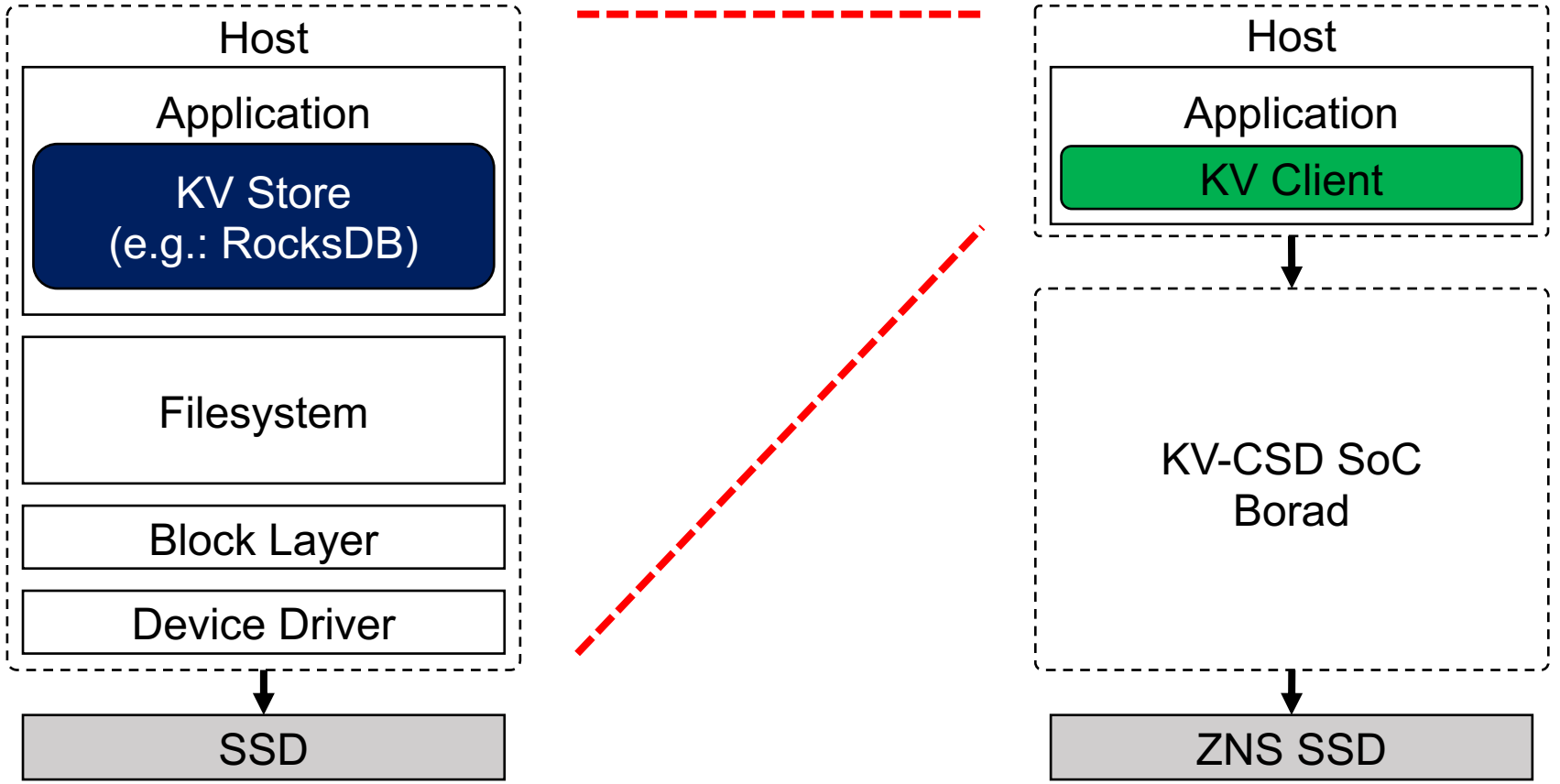
Why Hardware Acceleration?

- Software KV stores (such as RocksDB) rely on background processing to hide data sorting latency
- Insertion is suspended when background jobs cannot keep up
- Hardware acceleration allows for more aggressive latency hiding

By deferring background work until after insertion concludes and by performing it within a computational storage device



Why Hardware Acceleration?

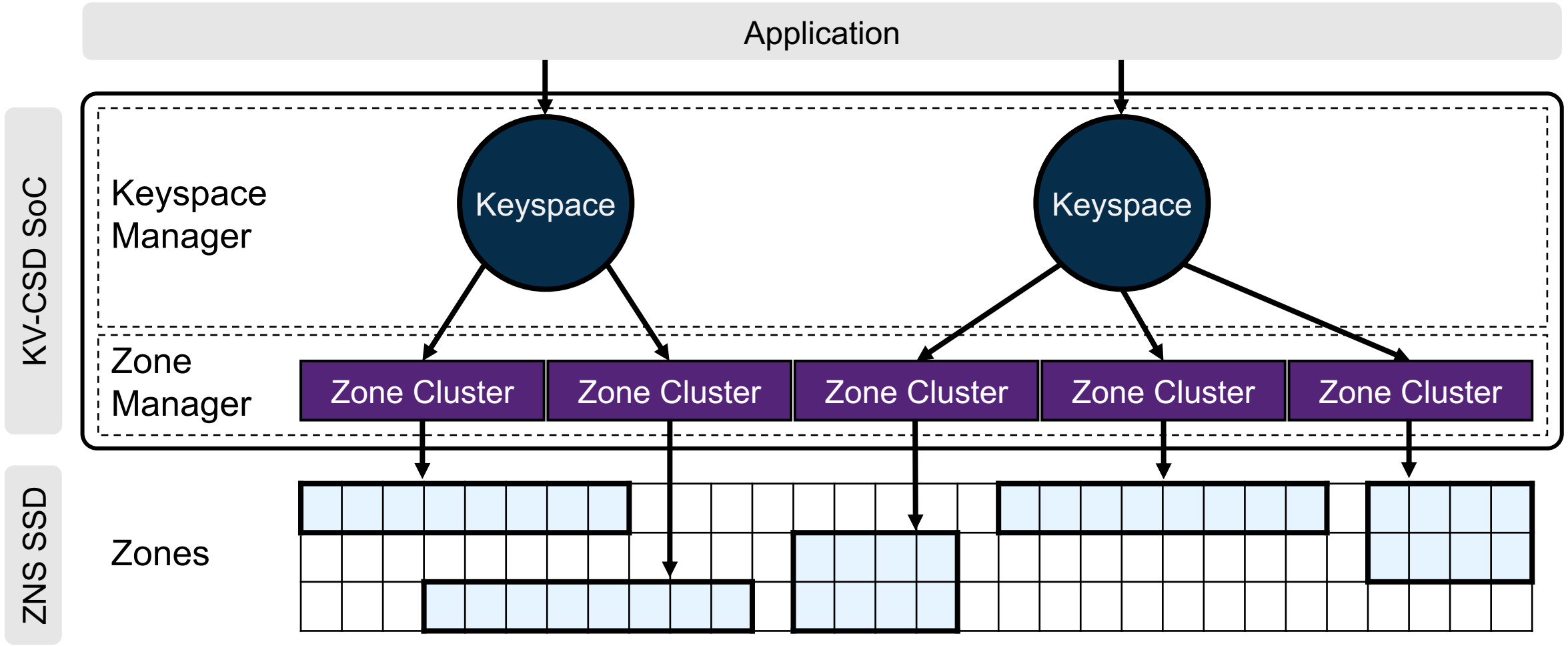


A reduction of software layers also enables higher performance

Today's Talk

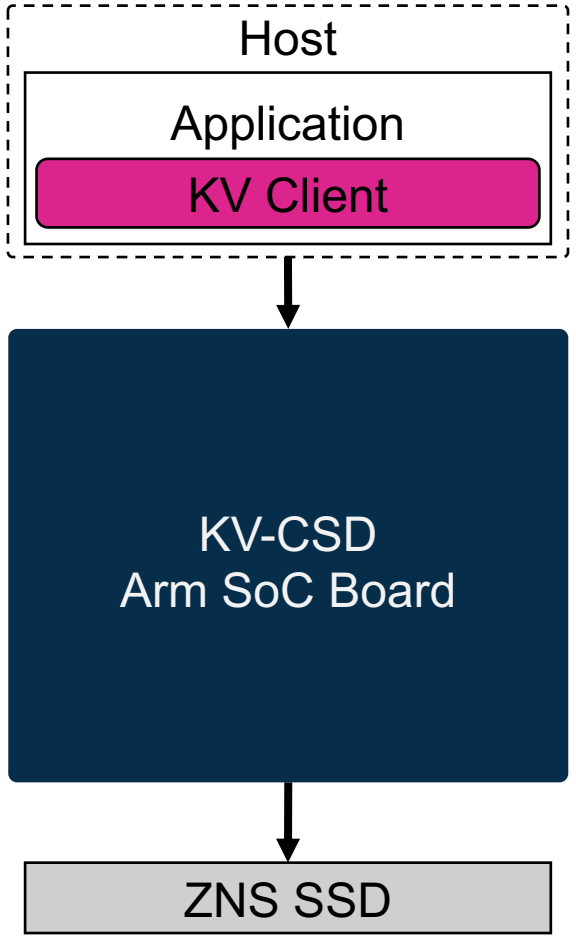
- ~~Why ordered computational KV storage?~~
- How does it work?

A Closer Look at the Device

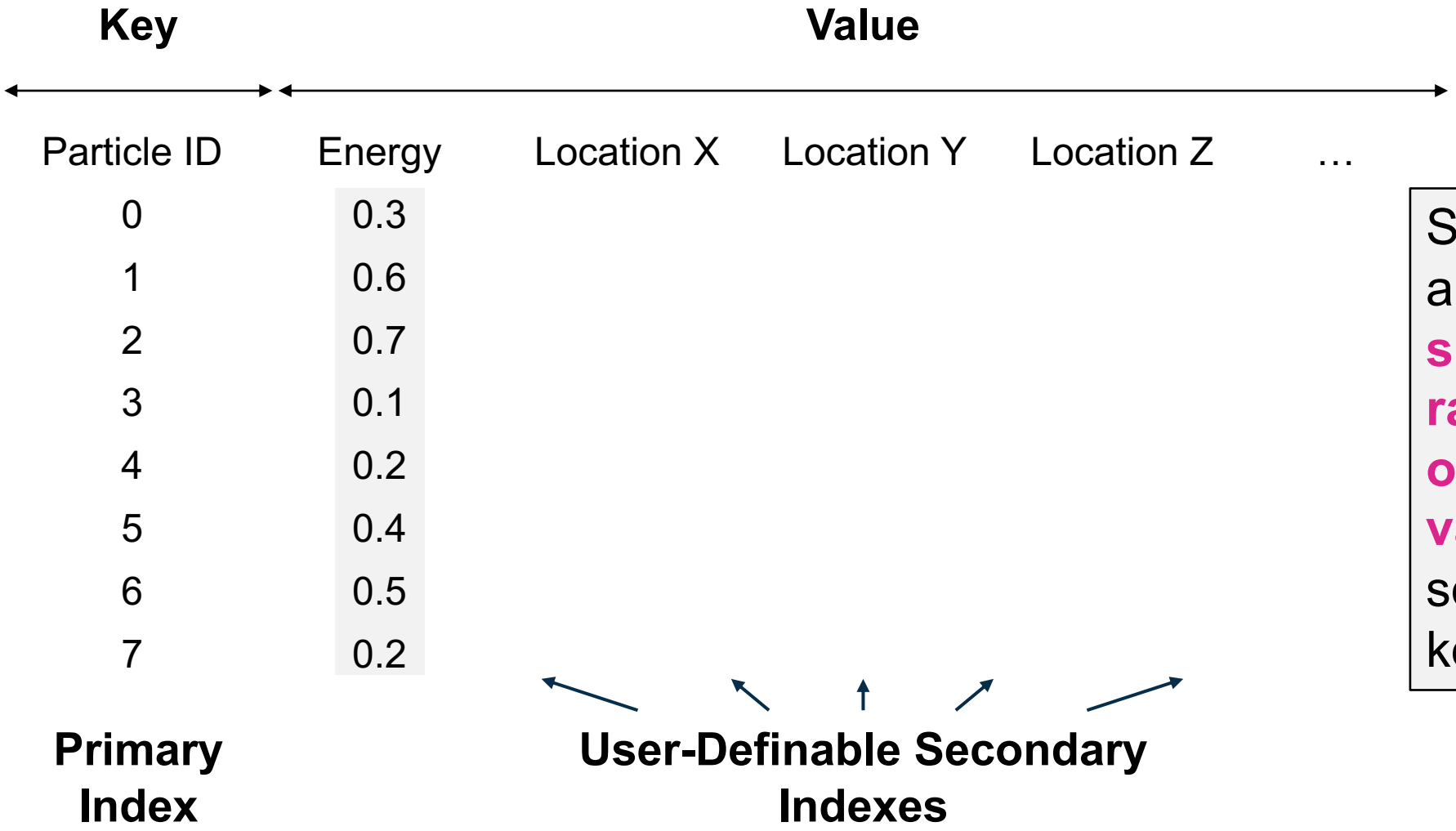


Keyspace API

	Keyspace State			
	New	Writable	Indexing	Indexed
Keyspace Info	✓	✓	✓	✓
KV insertion		✓		
Query				✓
Keyspace Deletion	✓	✓	✓	✓



Primary and Secondary Indexes



Secondary indexes are defined by **users specifying the byte range and the type of a portion of value** to serve as the secondary index keys

Evaluation Against RocksDB

Two scenarios

- Data insertion
- Range query against a secondary index

A 256-million particle dataset stored as KV pairs

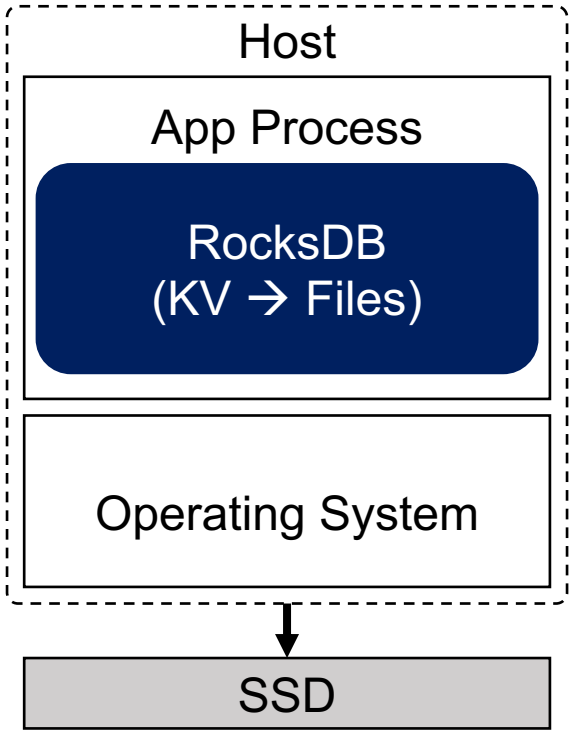
- Key: particle ID (16B)
- Value: particle payload (32B)

Analytics: range query over particle energy with varying selectivity

RocksDB vs KV-CSD Runs

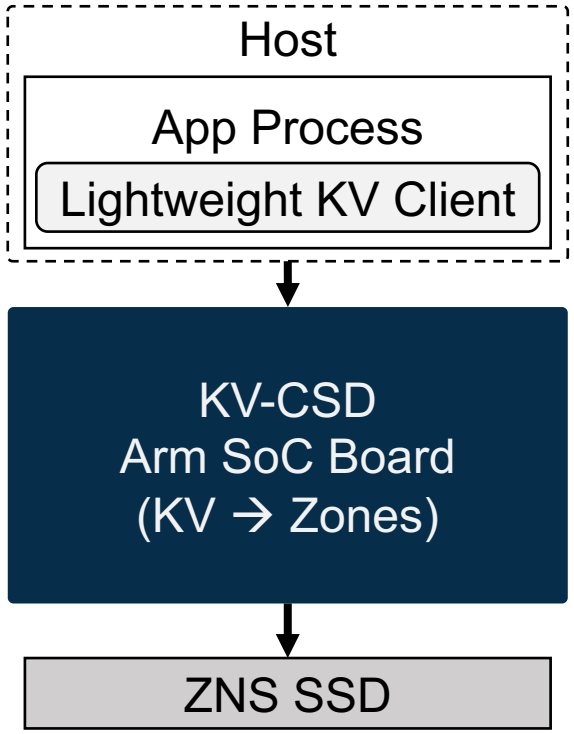
RocksDB

Full KV management (foreground & background jobs)

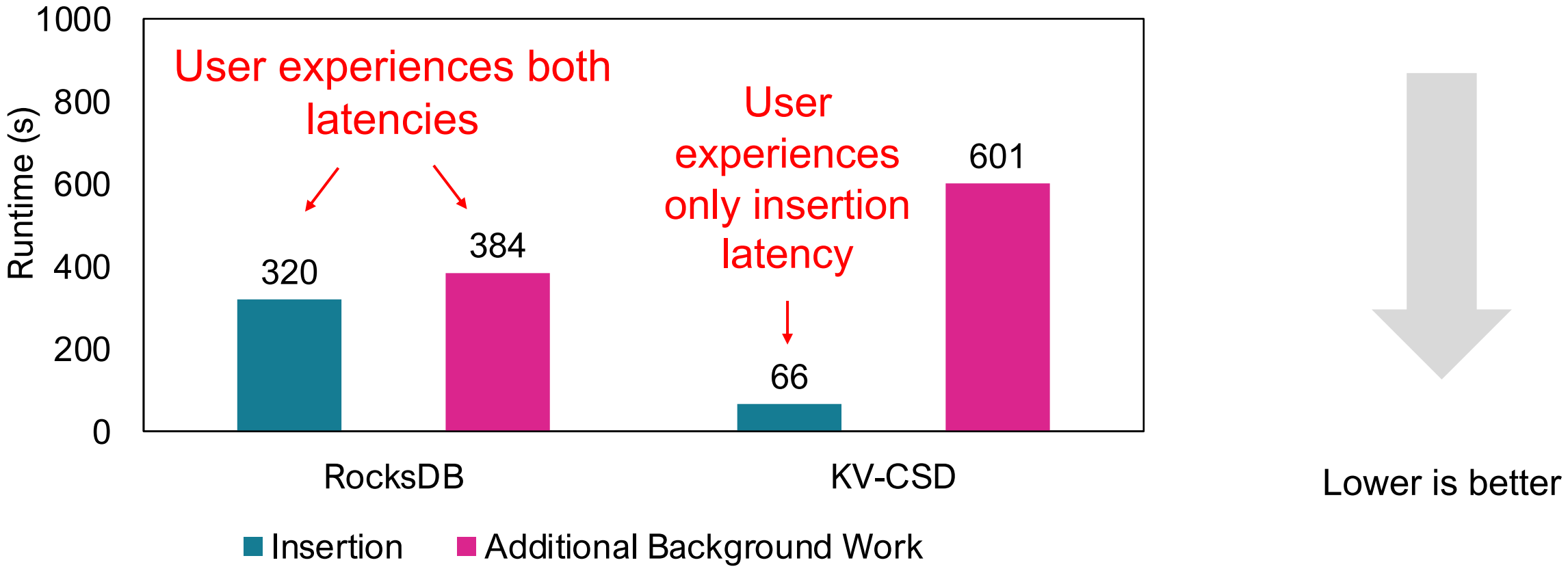


KV-CSD

NVMe KV command generation only

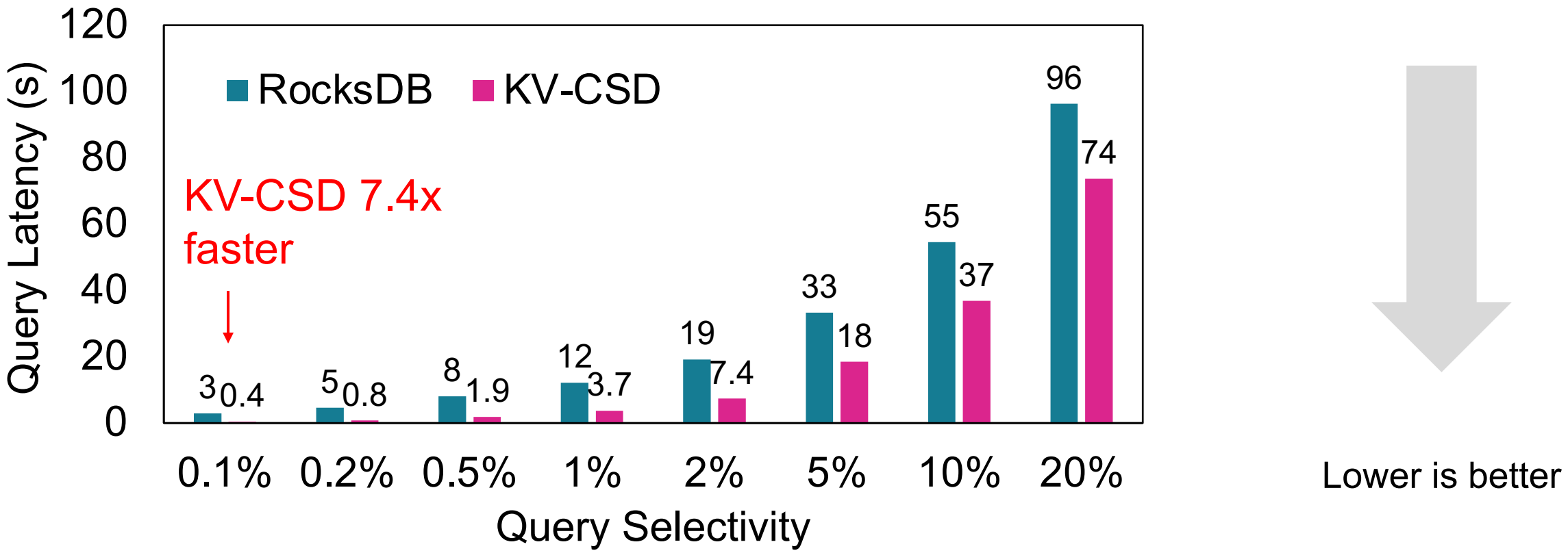


Results: Data Insertion



KV-CSD more effectively hides background work latency

Results: Range Query Against a Secondary Index



Lower is better

KV-CSD allows for more rapidly answering user queries thanks to hardware specialization

More on KV-CSD

2. KV-CSD demo at Flash Memory Summit 2023


KV-CSD: A Hardware-Accelerated
for Data-Intensive Applic

Inhyuk Park*, Qing Zheng†, Dominic Manno†, Soonyeal Yang*, Jas
Bradley Settlemyer‡, Youngjae Kim§, Woosuk Chung*,
*SK hynix, †Los Alamos National Laboratory, ‡NVIDIA, §So
{inhyuk.park, soonyeal.yang, woosuk.chung}@sk.com, {bsettlemyer}@nvidia.
{qzheng, dmanno, jasonlee, dbonnie, ggrider}@lanl.

Abstract—Popular software key-value stores such as LevelDB and RocksDB are often tailored for efficient writing. Yet, they tend to also perform well on read operations. This is because while data is initially stored in a format that favors writes, it is later transformed by the DB in the background into a format that better accommodates reads. Write-optimized key-value stores can still block writes. This happens when those background workers cannot keep up with the foreground insertion workload.

This paper advocates for a hardware-accelerated key-value store, enabling performance-critical operations, like background data reorganization and queries, to ex
instead of a host as existing key-value st
background work latency, prevents it f
writes, and improves overall I/O efficier
KV-CSD, is a key-value based comp
consisting of an NVMe SSD and a Syst
implements an ordered key-value store
offloaded processing, KV-CSD streamlir
host-device data movement for both ba
zation and query processing, and shows
times and up to 7.4× faster queries comp
of-the-art software key-value stores on

I. Introducti

Demands for storage performance
to rapidly increasing client performa
intensive applications such as simula
chine learning training, and large-sc
To keep up with these demands, many
systems — from Los Alamos' Trini
2016 to Oak Ridge's Frontier [5] and
supercomputers in 2022 — have empl
tiers to provide performance that mat
their compute tiers. In these flash-acc
remains as block devices and applica
storage using filesystems [7–10]. Su
enabled by flash allows applications t
amount of data between compute an
benefits applications such as simula
12] tha and write datasets in th
to sel eads) and do not requi
t format for efficient ret
ti [13 ertheless, applications
conversi either in the form of r
auxiliary xes alongside it — tend
processi lays. This is especiall

App
Client Library
Host

Fig. 1: Overview of a

volumes of small scien
massively parallel scie

1

2

3. R&D 100 Award

  1. KV-CSD paper at IEEE Cluster Computing Conference 2023

SDC 23

A More Complete Picture …

Tue Sep 19 | 4:05pm - 4:55pm

Salon V

Conclusion

Efficient data retrieval performance is key to scientific analytics

Computational storage opens new ways of acceleration infeasible with traditional methods

Preliminary results are very encouraging

More work/collaboration/innovation is needed for production deployment

Acknowledgement

Jason Lee (jasonlee@lanl.gov)

David Bonnie (dbonnie@lanl.gov)

Dominic Manno (dmanno@lanl.gov)

Gary Grider (ggrider@lanl.gov)

Bradley Settlemyer (bsettlemyer@nvidia.com)

Youngjae Kim (youkim@sogang.ac.kr)

Inhyuk Park (inhyuk.park@sk.com)

Soonyeal Yang (soonyeal.yang@sk.com)

Jungki Noh (jungki.noh@sk.com)

Woosuk Chung (woosuk.chung @sk.com)

Hoshik Kim (hoshik.kim@sk.com)

Pui York Wong (puiyork.wong@us.skhynix.com)

Jongryool Kim (jongryool.kim@us.skhynix.com)

Jin Lim (jin.lim@us.skhynix.com)

SDC 23

Los Alamos
NATIONAL LABORATORY

SDC 23