

STORAGE DEVELOPER CONFERENCE



*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

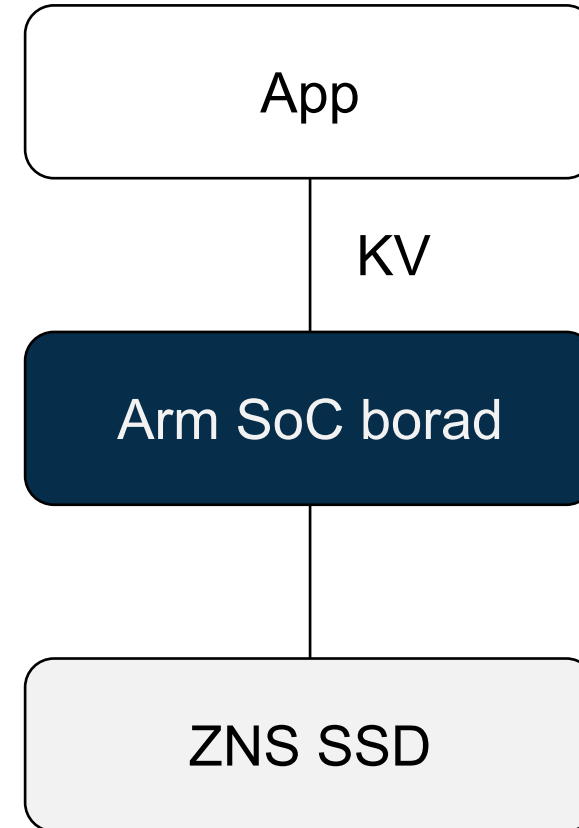
An **ordered**, hardware-accelerated **KV store** for rapid data insertion and queries

# A Quick Look

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

The arm board implements KV atop SSD zones

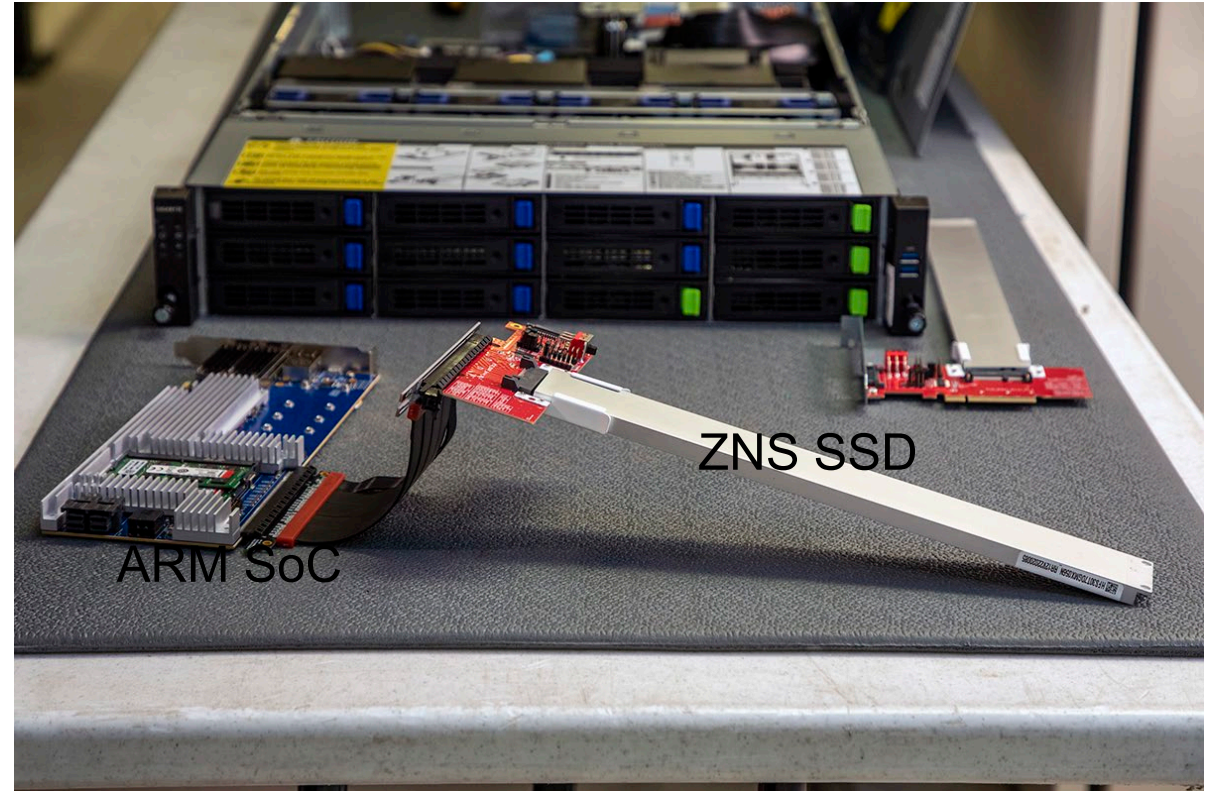
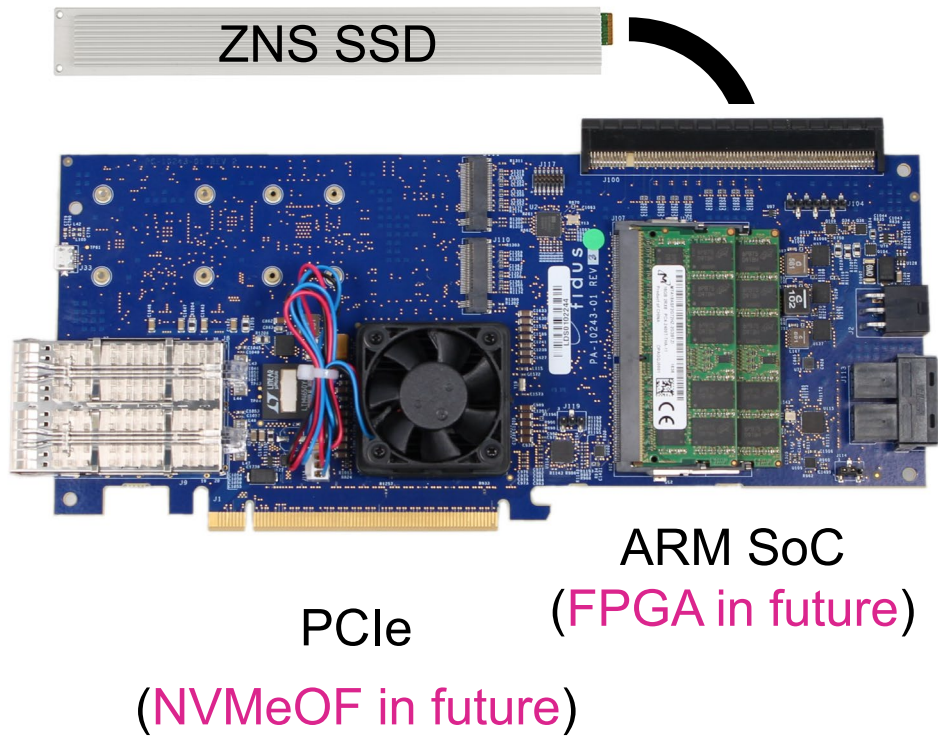
Apps use custom NVMe KV commands for bulk data insertion, index creation, and queries





# KV-CSD in Real World

Current Prototype



# Today's Talk

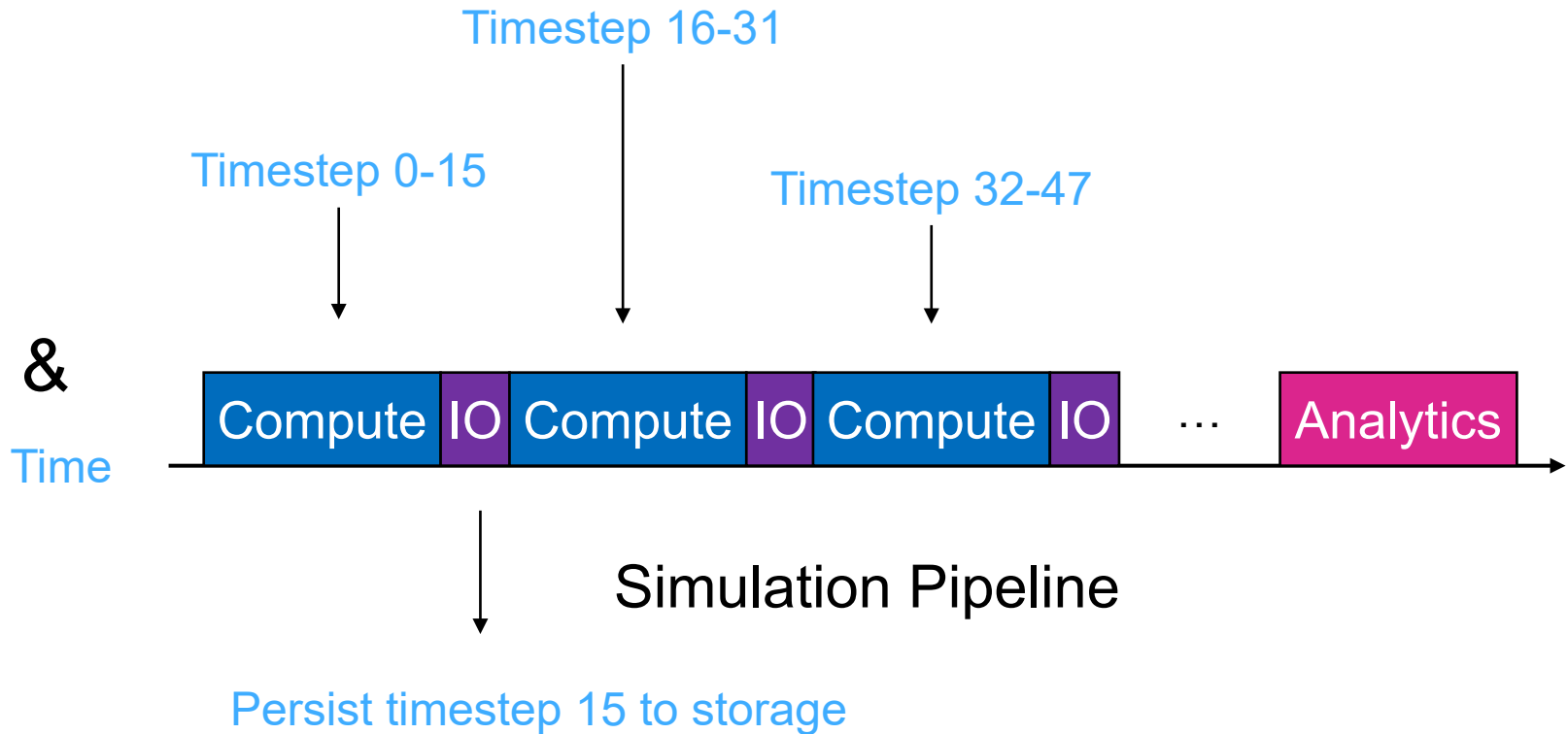
- Why ordered computational KV storage?
- How does it work?

# How Scientific Simulations Run

Time based bulk-synchronous parallel programs

Iterate between **compute** & **I/O** phases

Analytics occur after simulation



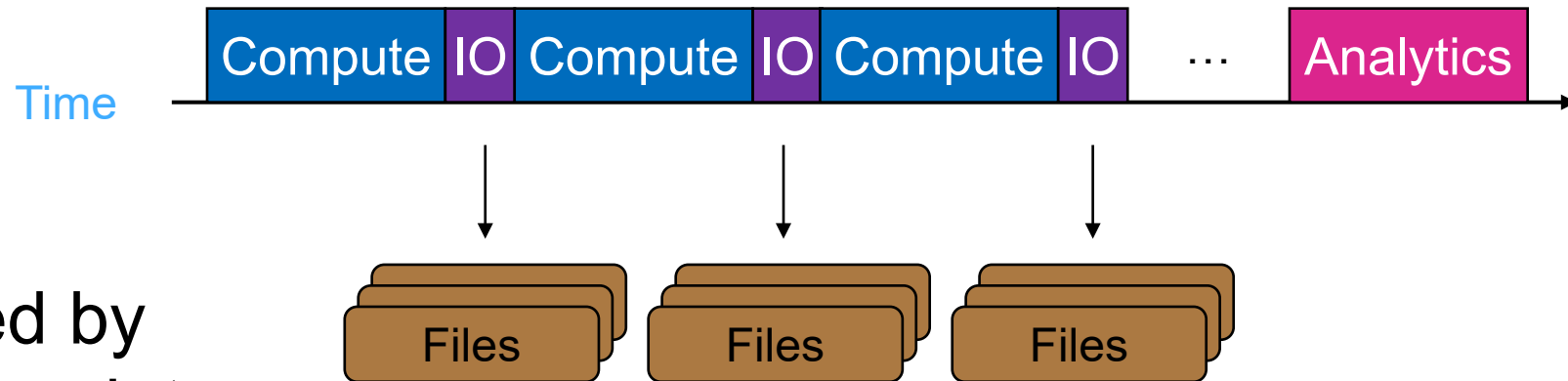
# How Data is Stored Today

Through **filesystems**

Data stored as **one big or many small files** per timestep

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

## Simulation Pipeline





# 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

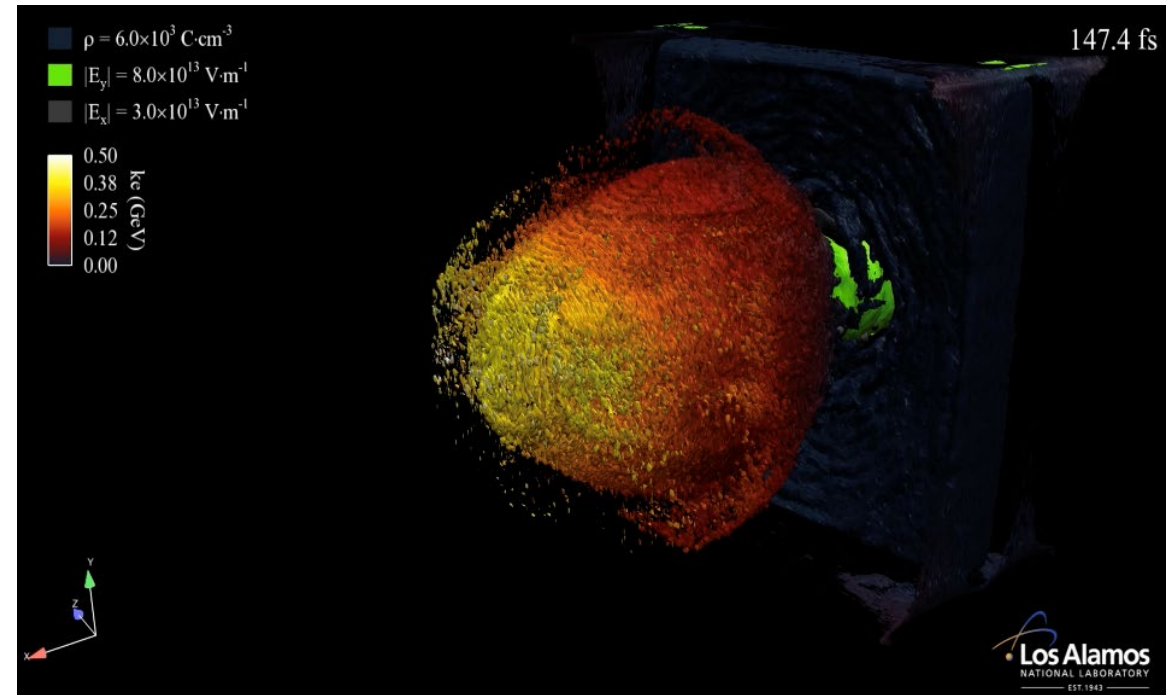


Image from LANL VPIC simulation done by L. Yin, et al at SC10

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

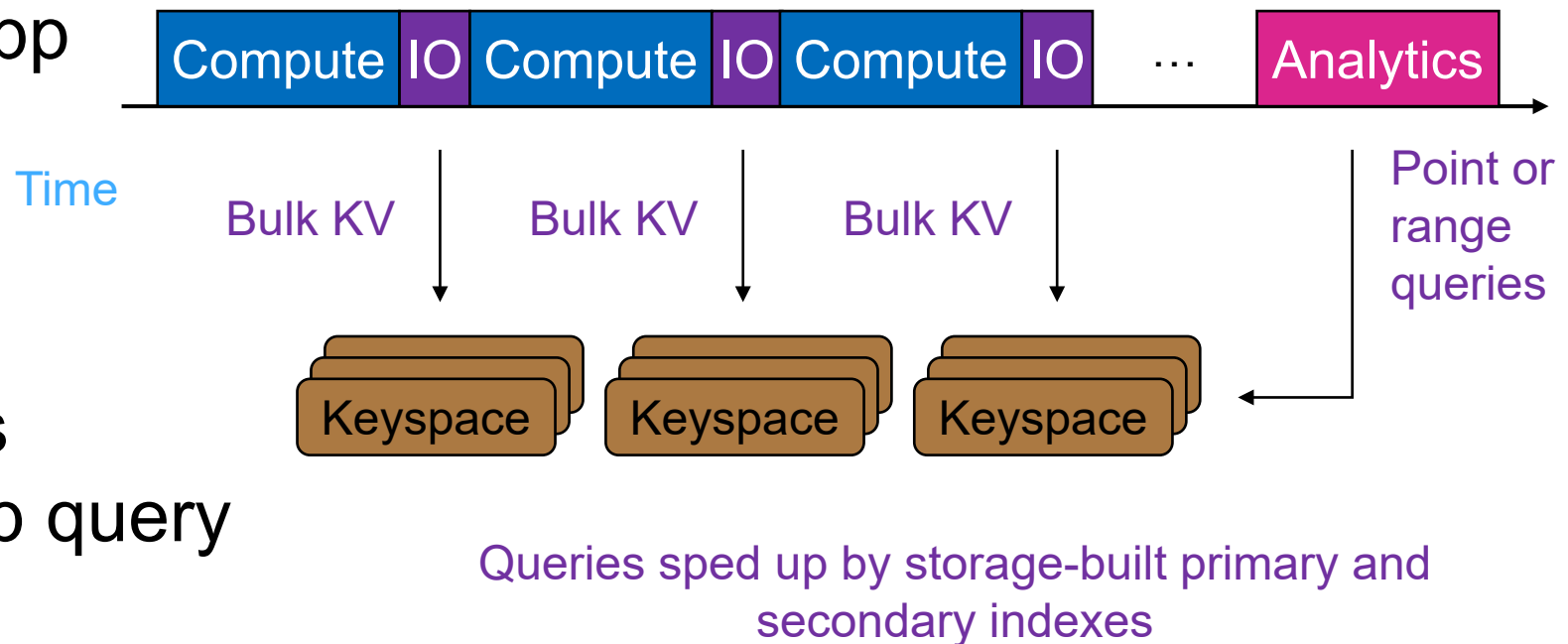
# Toward Ordered, Computational KV Storage

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

Simulation Pipeline



# 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

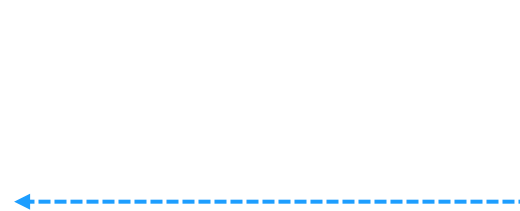
Switching from files to KV not awfully difficult

No need to map filenames to LBAs to enable offload

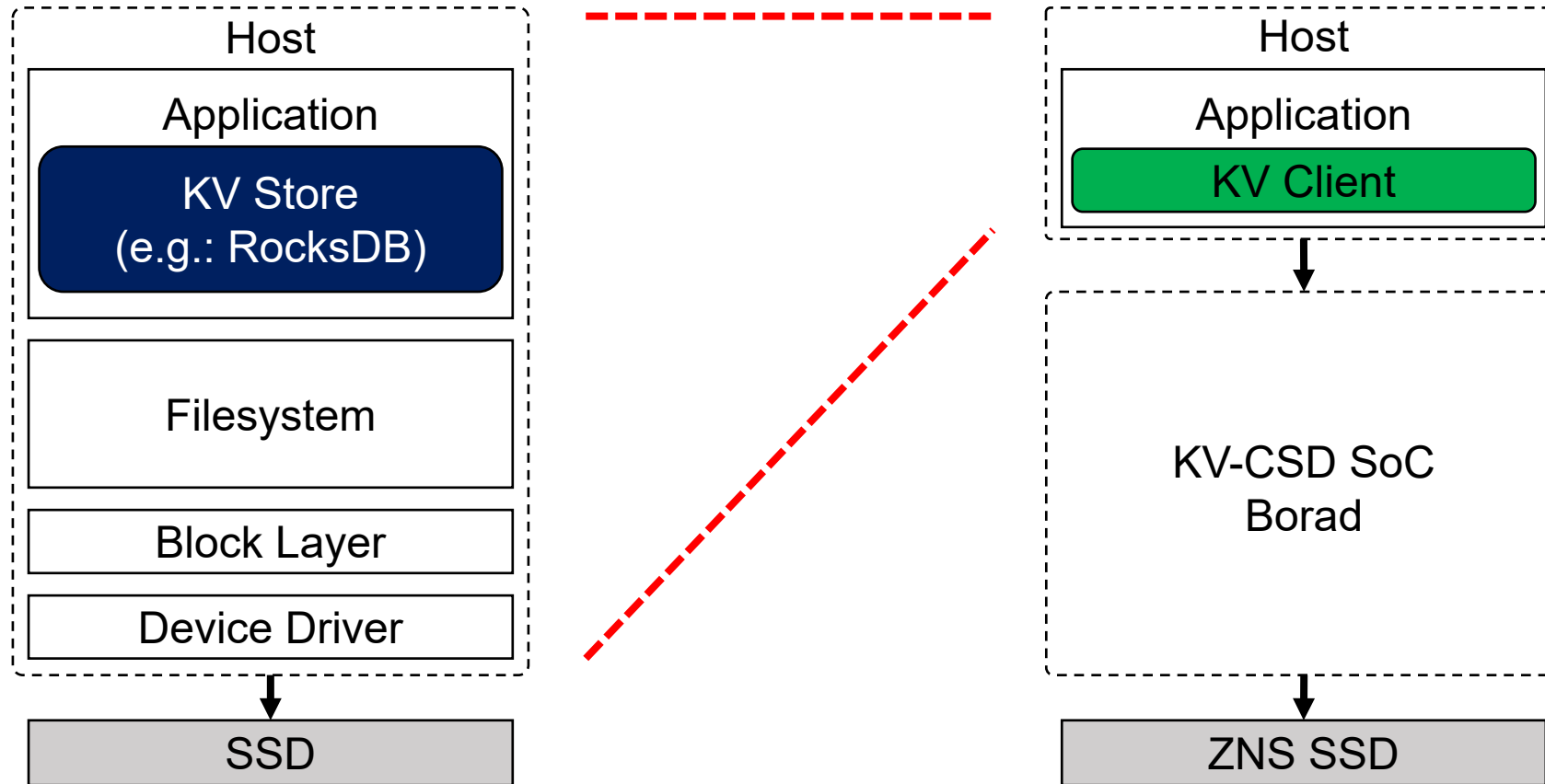
# Why Hardware Acceleration?

- 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?



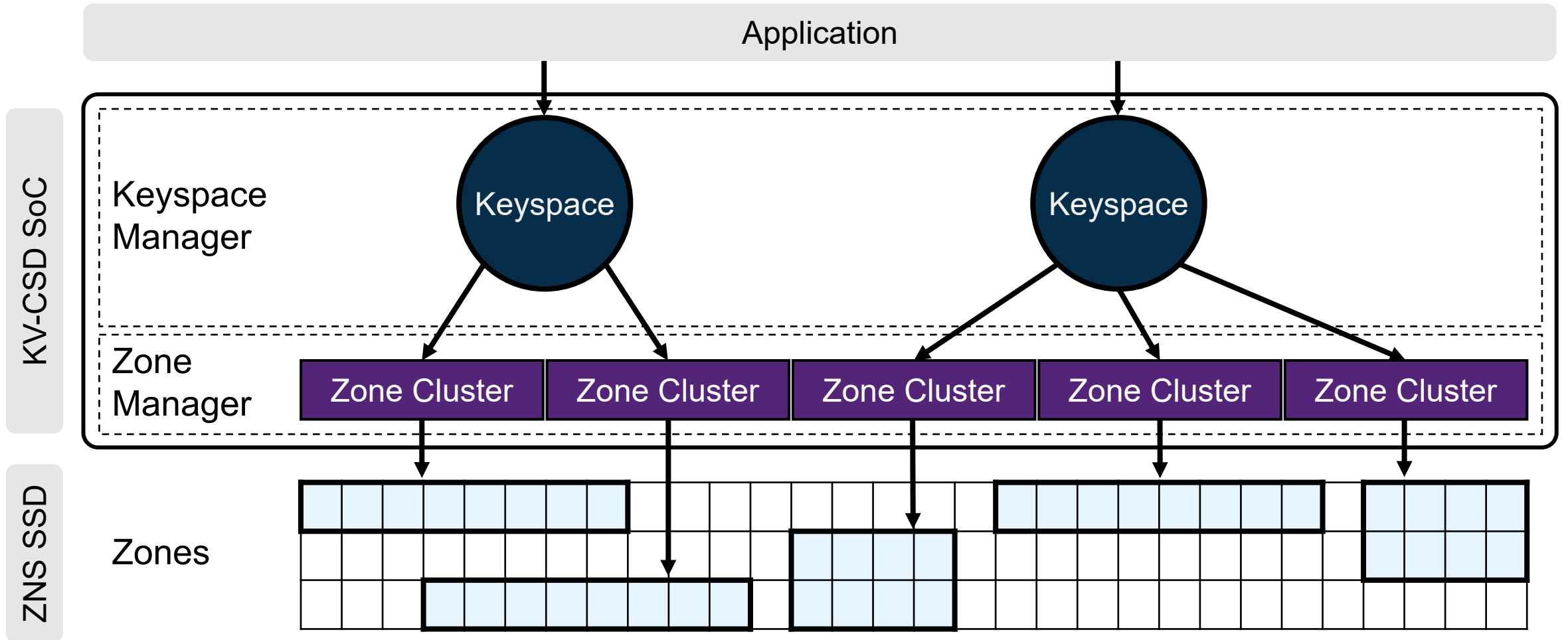
A reduction of software layers also enables higher performance

# Today's Talk

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

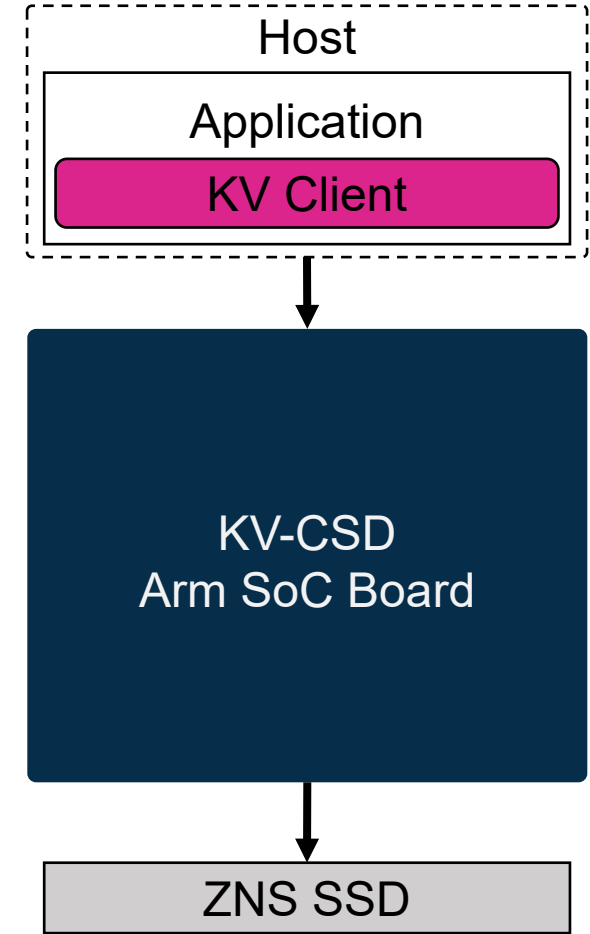


# A Closer Look at the Device

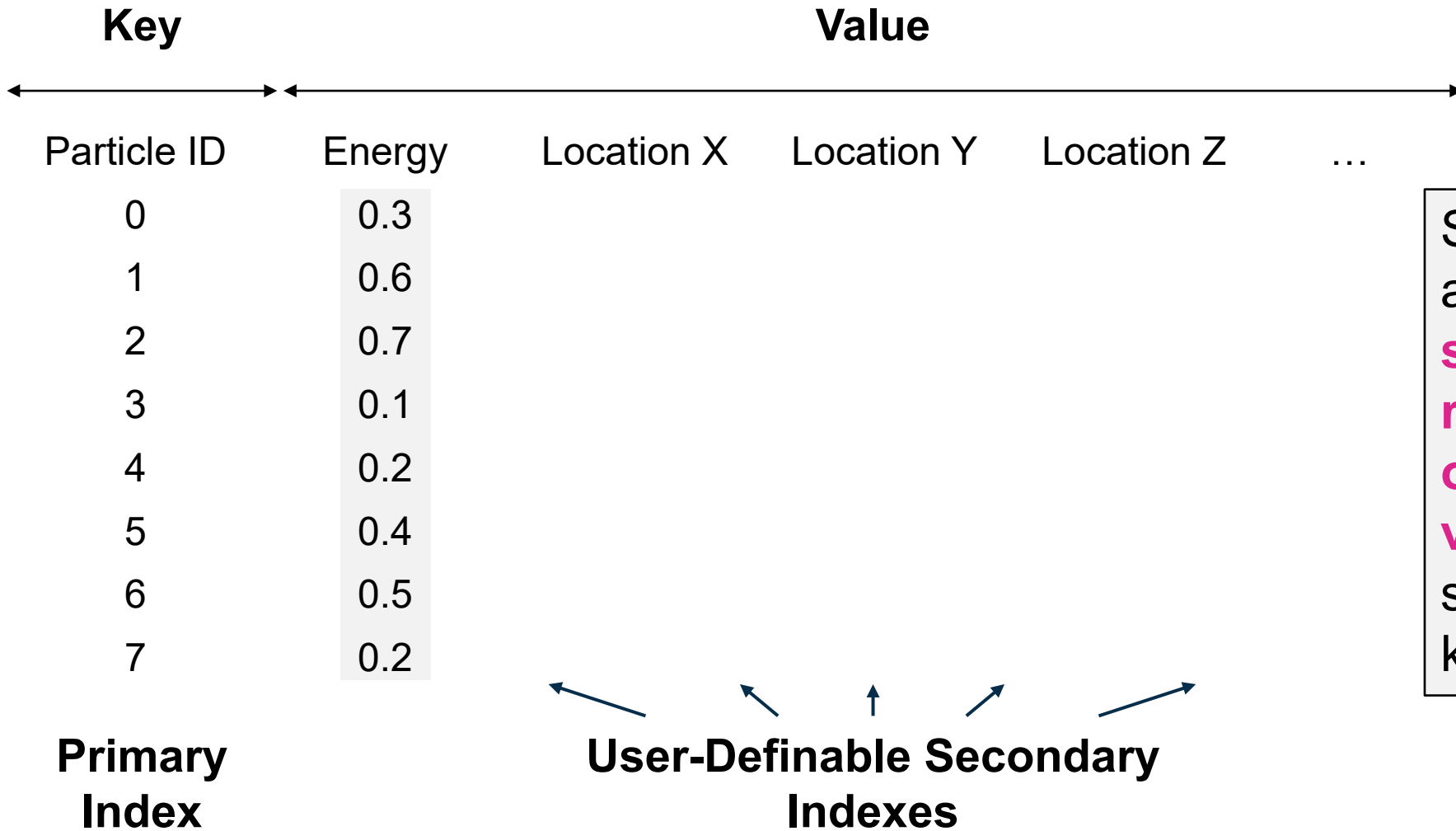


# Keyspace API

	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

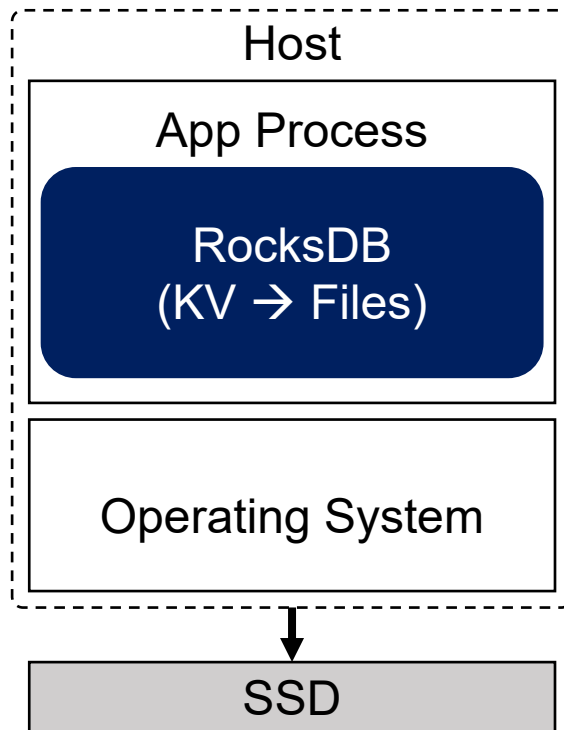
- Key: particle ID (16B)
- Value: particle payload (32B)

**Analytics:** range query over particle energy with varying selectivity

# RocksDB vs KV-CSD Runs

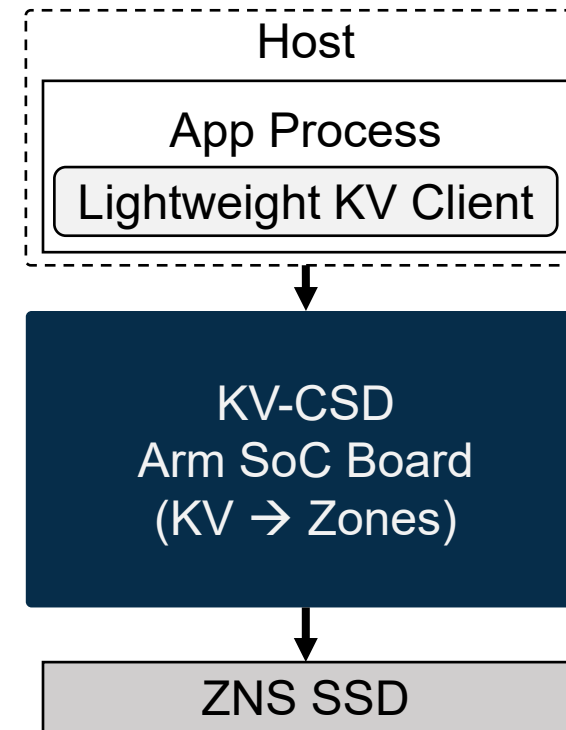
## RocksDB

Full KV management (foreground & background jobs)

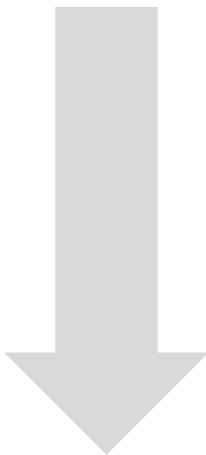
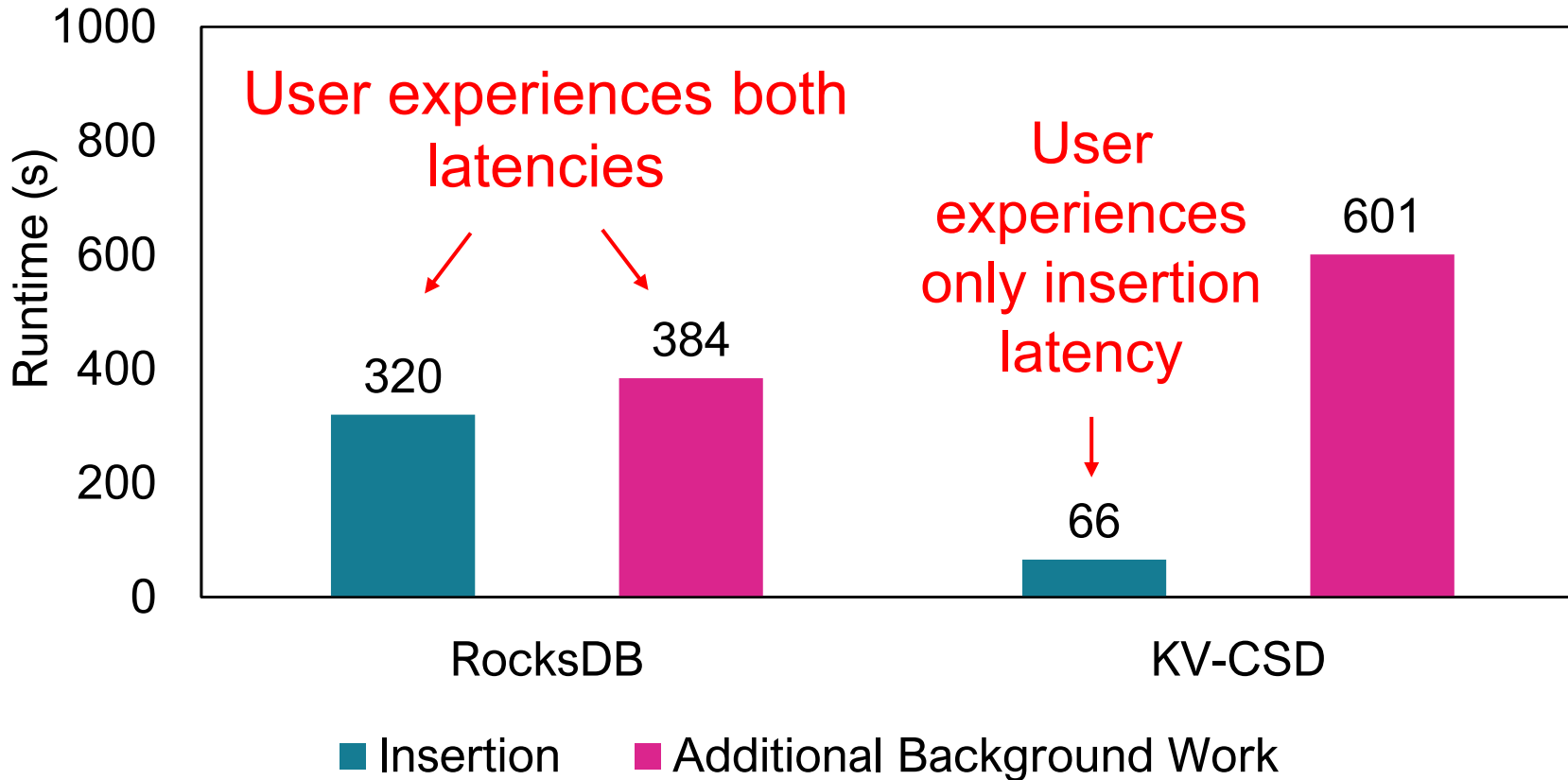


NVMe KV command generation only

## KV-CSD



# Results: Data Insertion

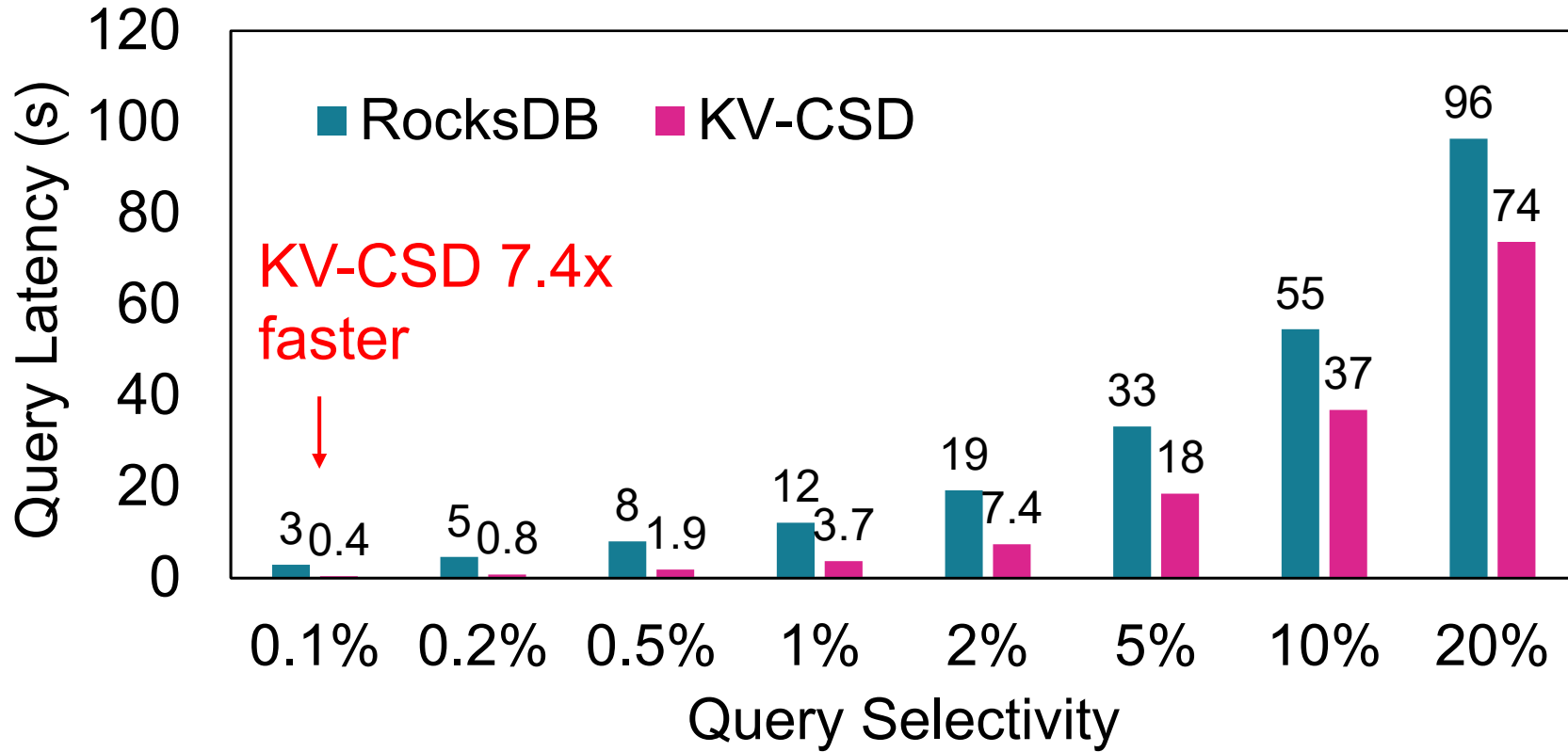


Lower is better

**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 Applications

Inhyuk Park\*, Qing Zheng<sup>1</sup>, Dominic Manno<sup>1</sup>, Soonyeal Yang\*, Ja Bradley Settlemyer<sup>1</sup>, Youngjae Kim<sup>3</sup>, Woosuk Chung\*,<sup>1</sup> SK hynix, <sup>1</sup>Los Alamos National Laboratory, <sup>2</sup>NVIDIA, <sup>3</sup>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 write 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 occur in parallel with foreground work. Instead of a host as existing key-value stores, KV-CSD is implemented on a hardware-accelerated storage device consisting of an NVMe SSD and a System-on-Chip (SoC). KV-CSD streamlines host-device data movement for both background and query processing, and shows up to 7.4x faster queries compared to software key-value stores on a host.

#### 1. INTRODUCTION

Demands for storage performance to rapidly increasing client performance-intensive applications such as machine learning training, and large-scale systems — from Los Alamos' Tritium 2016 to Oak Ridge's Frontera [5] and supercomputers in 2022 — have empires to provide performance that matches their compute tiers. In these flash-accelerated storage devices [7–10], storage using file systems [7–10]. Enabled by flash allows applications to process amount of data between compute and storage benefits applications such as simulation [2] that read and write datasets in the form of secondary keys (reads) and do not require reformat for efficient retrieval [13]. Nevertheless, applications converted to either in the form of primary keys alongside it — tend to process in layers. This is especially



Fig. 1: Overview of a massive parallel system.



1

2

## 3. R&D 100 Award

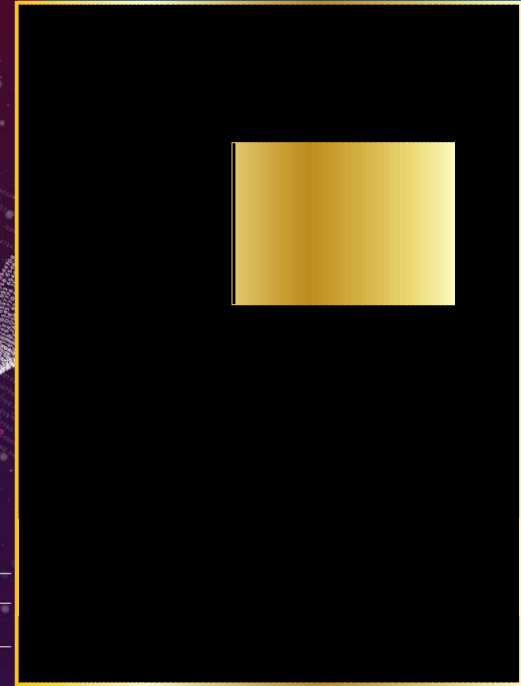
2023 R&D 100 JOINT ENTRY

# ORDERED KEY-VALUE COMPUTATIONAL STORAGE DEVICE (KV-CSD)

Revolutionary information storage hardware efficiently records, sorts, and indexes supercomputer simulation output to streamline big-data analysis

- PRODUCES** a billion-fold reduction in unnecessary data movement
- SLASHES** the time to scientific insight
- PROVIDES** needle-in-a-haystack identification of phenomena of interest
- INDEXES** multitrillion-particle and other petascale datasets
- SUPPORTS** multidimensional point and range queries

Los Alamos NATIONAL LABORATORY SK hynix



# A More Complete Picture ...



**STORAGE DEVELOPER CONFERENCE**  
SDC<sup>23</sup> | FREMONT MARRIOTT SILICON VALLEY  
SEPTEMBER 18-21, 2023  
*BY Developers FOR Developers*

Computational Storage for  
Simulation Science Storage  
System Design

**Dominic Manno**, Senior Scientist, **Los Alamos National Laboratory**

[www.storagedeveloper.org](http://www.storagedeveloper.org)

A SNIA<sup>®</sup> Event



**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)







# Los Alamos

NATIONAL LABORATORY