Why KV SSD will replace ZNS

Andy Tomlin Bio

CEO & Founder at QiStor, developing next generation storage solutions

30 Year storage veteran, over 50 patents in Flash System Design, numerous flash based products delivered to customers

Leadership and Architecture roles at Kioxia, Samsung/Stellus, WD, Skyera, Sandforce, Sandisk, Quantum, IBM

“Every problem is one more abstraction away from solving”

Storage Problem

For the last 50 years, legacy storage has used fixed sized containers (LBA's) to store data

Real objects never fit exactly, so host mapping systems have been added to manage this

This mapping is the source of significant complexity, performance & scaling problems

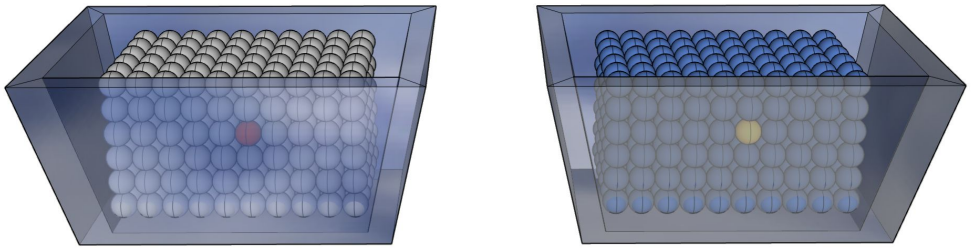
This leads to extensive costly overprovisioning of Flash & Servers as the only solution

… at the moment

Why do drives wear out and need replacement?

Tiny objects amplify wear, and tiny objects are the most common (50-200B)

To change one small red object to a yellow object, all the other objects in the same container also get rewritten, even though they never changed → Wear & Power



Meta

Facebook Kangaroo cache is a method to help manage this issue - but it is an incremental improvement and does not solve the fundamental problem

Today's Layered Architecture is Massively Inefficient

The pyramid of layers adds significant host software and management complexity

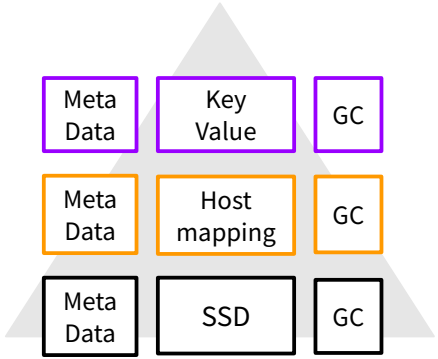
Layers ***multiply*** reads and writes

lots of CPU cores, memory, and power is spent on this work instead of Customer value (~2 cores per drive)

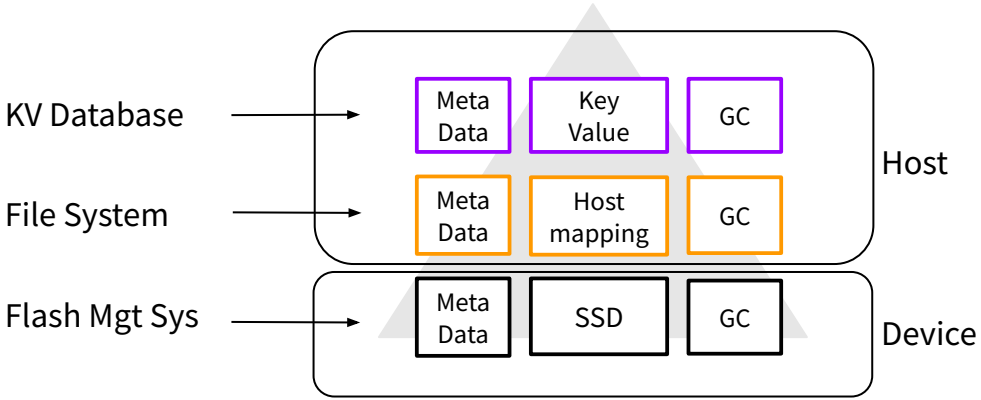
Drives wear faster, burning power and impacting performance

Due to complexity it requires skilled engineers to configure and optimize HW and database

Existing Architecture cannot scale to meet growing needs



Typical legacy block stack today



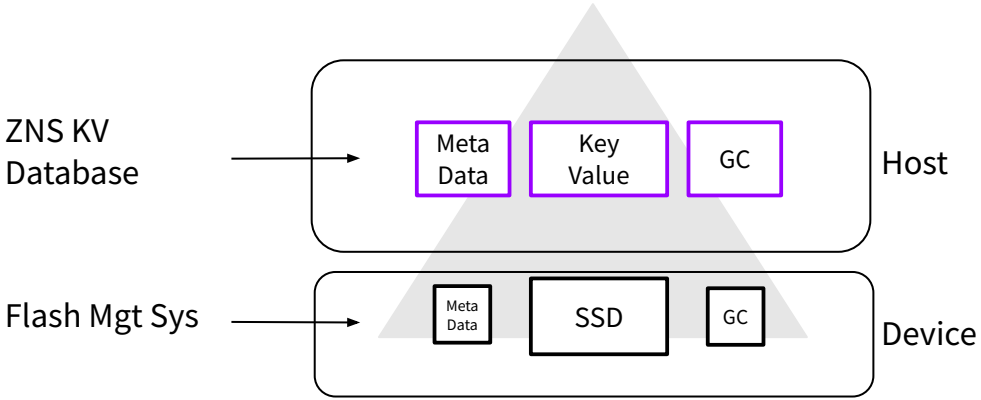
Each mapping layer performs a similar set of critical functions:

1. Allocation of space
2. Tracking of location via metadata
3. Garbage collection

As these layers are all independent of each other, write amplification is **multiplied**

The ZNS goal

Solve the legacy map stacking problem by moving everything to the host



Simplify the device Flash management system to reduce the amount of metadata

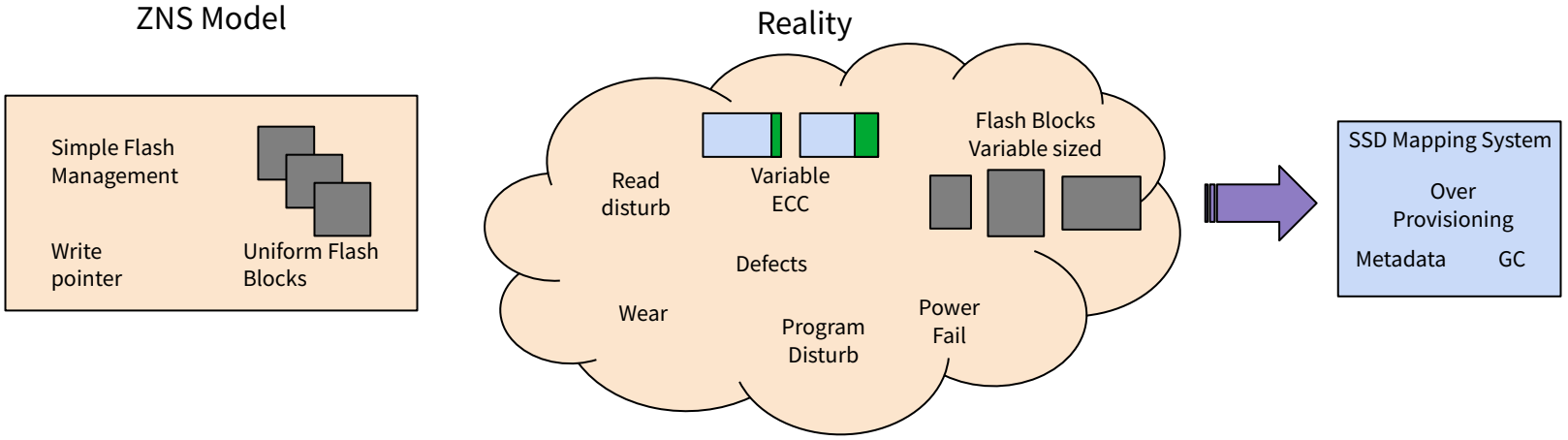
- 'Write' & Erase in bigger chunks (eg 1MB)
- Write in manner to 'eliminate' GC

Move all the mapping complexity to the host

By ***neutralizing*** the device GC the multiplication factor becomes 1

This will definitely be better than legacy…

The ZNS problem



The ‘hope’ is that SSD Mapping system GC is close to 0 and that the Overprovisioning can be minimized…

…and that you can keep balance the broomstick on your hand

ZNS is a Short Term Kludge

Pros	Cons
ZNS improves on legacy stack ✔	✘ Pulls device complexity into Host
	✘ Interface abstraction does not match the needs of the application
	✘ Poor scaling
	✘ Requires ‘well behaved’ host

FIX THE ABSTRACTION - STOP BALANCING THE BROOM

KV ensures ALL over provisioning in one place - the device

Key-Value SSD solves the fundamental problem

Removes the need for Host to track data location

Mapping layers are **eliminated**

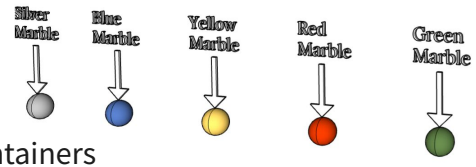
Axiomatically the **optimal solution**

- Significantly Less **Power**
- Higher **Performance & Linear Scaling**
- **No optimization required**

Lowers CAPEX
Lowers OPEX

Fixed sized containers eliminated

objects accessed by reference



KV lowers TCO compared to ZNS

	ZNS	Legacy	KV
Allocation/Tracking of Value loc	Host	Host	Device
GC of Logical space	Host	Host	Device
GC of Flash Space	Host(Mostly)	Device	
Compression	Host(SW)	Host(SW)	Device(HW)
Application WA	12-30 (LSM)	10-30 (LSM)	1
Flash WA	1.1	3	1.5-9
Total WA	13-33	30-90	1.5-9

KV is inherently the Green solution

Low Total WA is enabler for QLC/PLC flash

ZNS 'solves' Multiplied GC by *mostly* performing Flash space GC in the Host as part of logical space GC
This is an expensive solution

95% **Host CPU Performs:**
Data placement
GC
Compression
Metadata management

5% Application Functionality

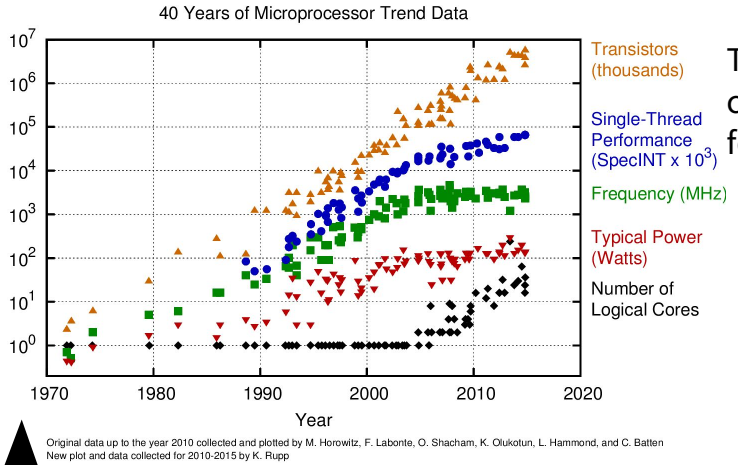
CPU limited

100% **Host CPU Performs:** Application Functionality

Scales with devices

With KV all the overprovisioning is in one place, which is the optimal place

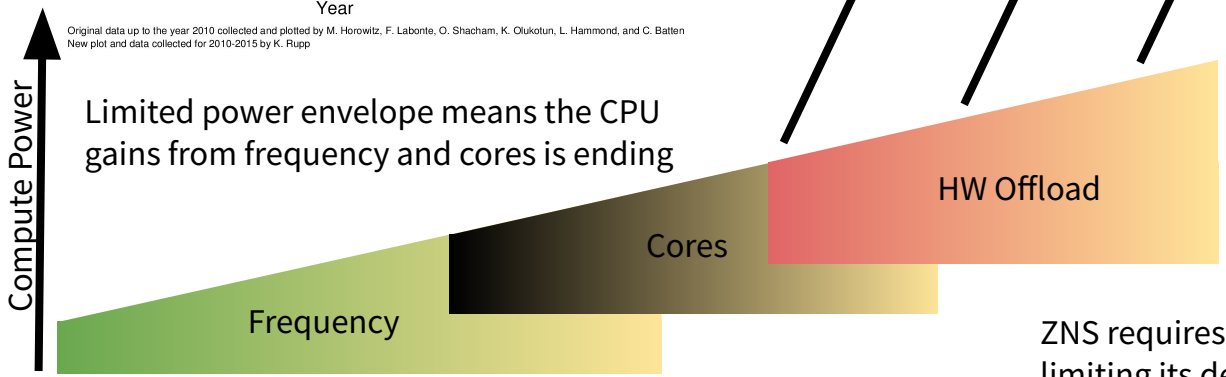
Compute Scaling - Why Host Data Placement is a Dead End



Today: Era of smart HW offloads where custom programmable HW is responsible for most of compute scaling

Storage is going in the wrong direction

ZNS and other data placement ideas are moving storage compute workload to host



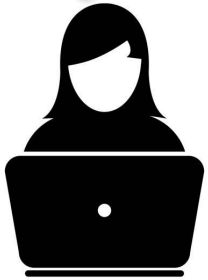
KV SSD looks to offload **Storage** tasks

ZNS is *only* a solution for top 3-4 cloud providers

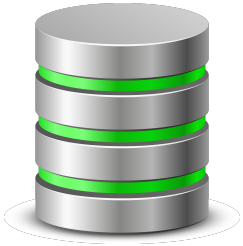
ZNS requires application support to get any benefit, limiting its deployment opportunity

The customer optimization problem

Skilled Database Engineer



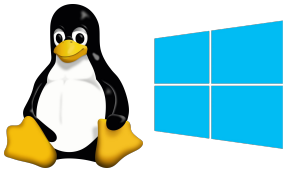
Database



Caching Indexing
Testing
Application Selection

OS

Configuration
Patching &
Optimization



File System

HW Platform

Instance Type

Sizing

Load Balancing

Memory Size

Storage Type

Most customers treat application/database optimization as a way to solve problems

Vast majority of Databases & HW platforms are run unoptimized

QiStor solution: Less HW, linear scaling, smaller software stack makes all this *much simpler*

How big a key do you need?

The main driver for the key size is the number of objects being stored

You want enough bits that the number of collisions is ‘manageable’

- Manageable can mean different things for different designs…

Collisions/collision probability is based on the Birthday Paradox

Data size	Average object size	Approx Number of Objects	Key size (bits)	Expected Number of collisions
16TB	128B	2^37	104	64
	1K	2^34	96	32
128TB	128B	2^40	112	128
	1K	2^37	104	64

How is a KV drive different from a regular Legacy SSD

Legacy drive

Data size	Compression	Metadata size
16TB	No	0.016TB
	Yes	0.052TB
128TB	No	0.144TB
	Yes	0.448TB

KV drive

Data size	Average object size	Approx Number of Objects	Key size (bits)	Metadata size
16TB	128B	2^37	104	2.5TB
	1K	2^34	96	0.3TB
128TB	128B	2^40	112	22.5TB
	1K	2^37	104	2.7TB

The key difference is all in the scale of the metadata

Implies Drive FTL must be Native KV

Remember the metadata on KV drive is replacing much of application metadata, for **overall** reduction in total metadata

Why is KV SSD so much more Power efficient

Power = **Power per op** * **Number of ops**

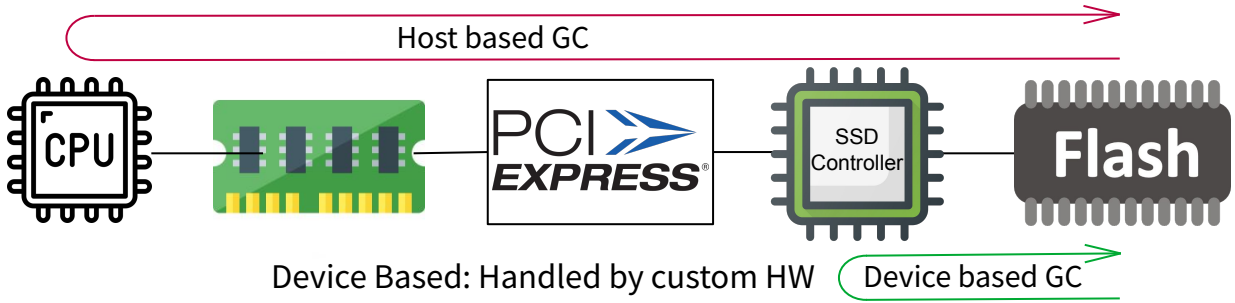
Power per op

Due to Write Amp, power is dominated by GC

Garbage collection steps

1. Identify what to GC
2. Determine if data is valid
3. Copy valid data

Host Based: Data passes over many buses, memories and requires tens of thousands of CPU instructions



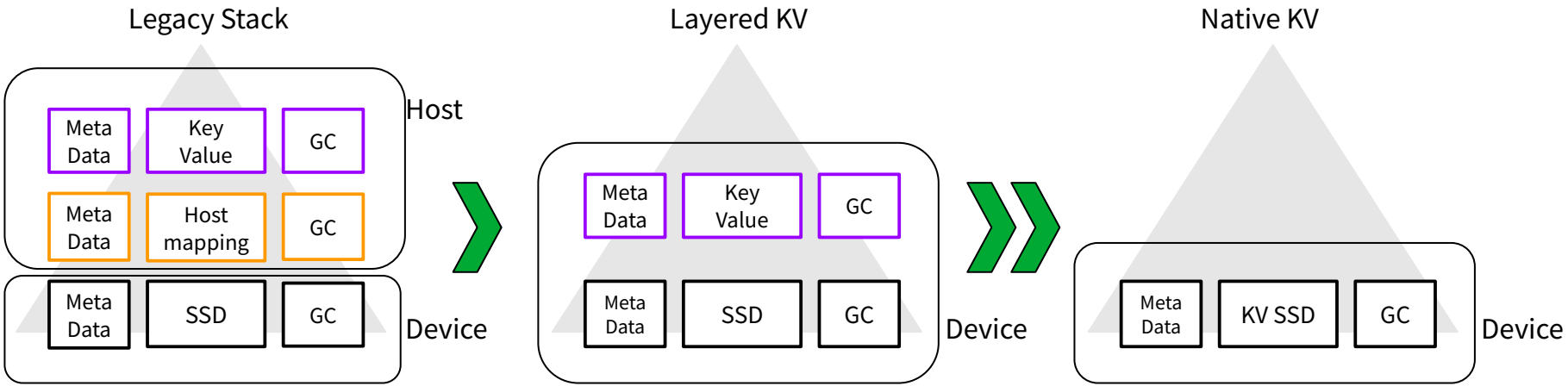
Garbage collection on host is **1-2 orders** of magnitude higher power

X

Number of ops

KV SSD does significantly less ops than legacy or ZNS type solutions due to lower write amplification

Why must a KV SSD be a native solution?



Today - maximum Metadata and GC

Layered KV - minimal reduction in Metadata and GC

Native KV - large reduction in Metadata and GC

Native KV design is necessary as the whole point is to have a **single** GC

If KV in the device is so great, why isn't everyone doing it?

KV is Hard

Requires radical rethink and innovation in how SSD mapping works to cope with significantly more mapping data (>20x legacy)

QiStor has done this innovation

Software Stack

KV is completely different to Legacy storage, and there is no software stack today

QiStor has expertise and will build this stack

Ecosystem

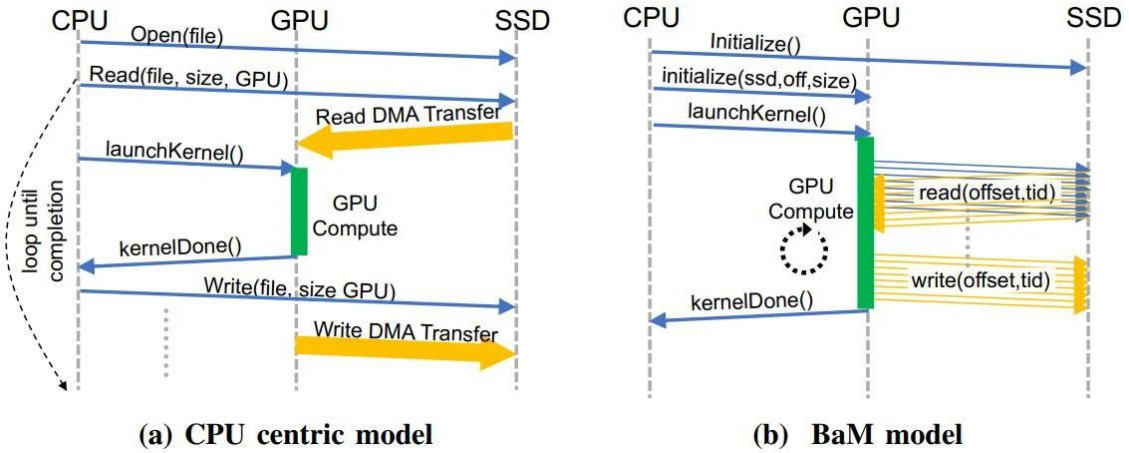
Customers want simple path to solution that solves the problem and is easy to adopt

QiStor solution will achieve this

KV enables new solutions - ML

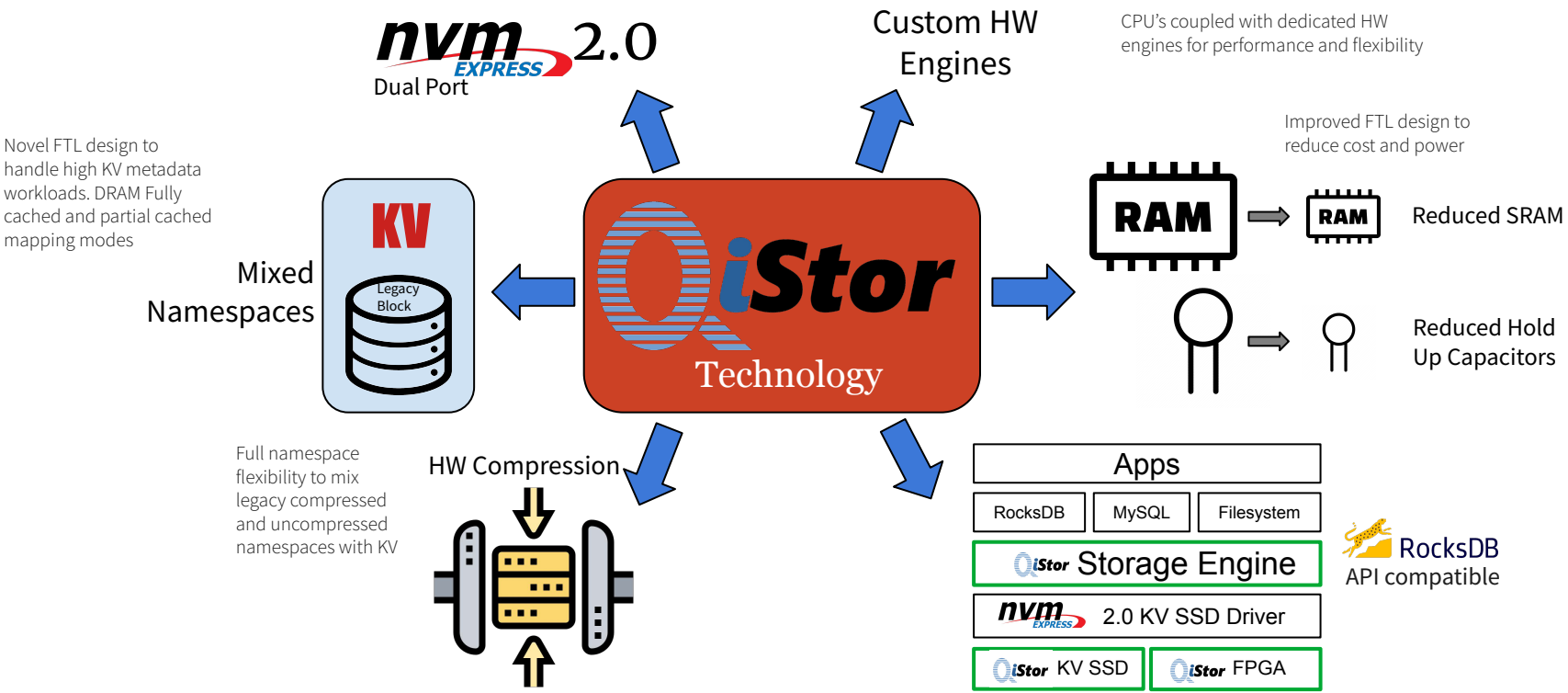
Nvidia & IBM proposed new methods of supplying data to GPU for ML without involving CPU

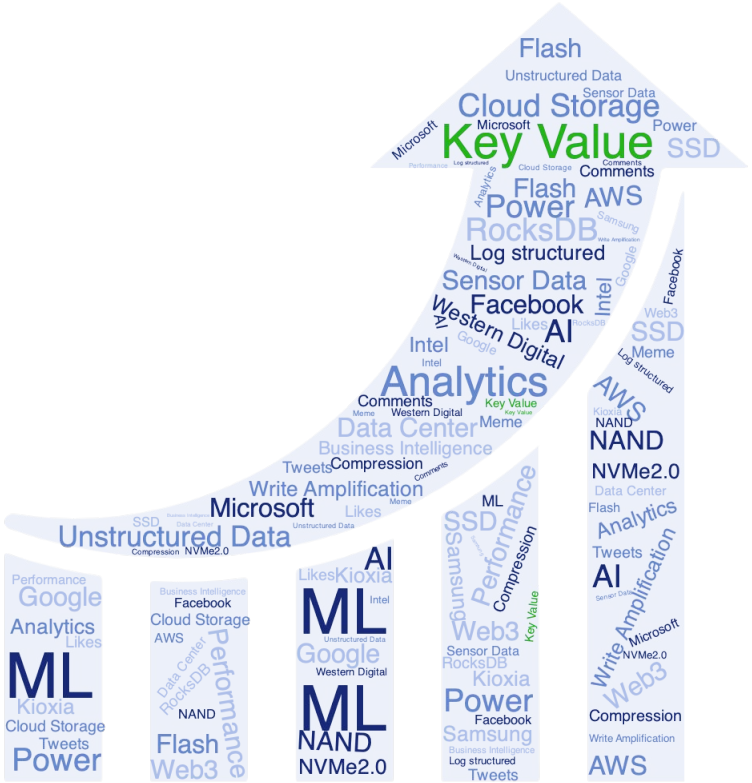
Potential ‘Killer App’ for KV SSD



Main function of CPU in the paper is to handle file system and convert into offsets. With KV none of this is necessary and just keys can be communicated for significant simplification

QiStor Technology - Application Storage Accelerated




Flash
Unstructured Data
Sensor Data
Cloud Storage
Microsoft
Key Value
Power
SSD
Comments
Flash
AWS
Power
RocksDB
Log structured
Sensor Data
Facebook
AI
Western Digital
Intel
Analytics
Comments
Key Value
Meme
Data Center
Business Intelligence
Compression
Write Amplification
Microsoft
Likes
Unstructured Data
NVMe2.0
Performance
Google
Analytics
ML
Kioxia
Cloud Storage
Tweets
Power
Facebook
Cloud Storage
AWS
Performance
NAND
Flash
Web3
AI
Kioxia
ML
Google
Western Digital
ML
NAND
NVMe2.0
ML
SSD
Samsung
Performance
Compression
Web3
Sensor Data
RocksDB
Kioxia
Power
Facebook
Samsung
Log structured
Tweets
Facebook
Web3
SSD
Meme
AWS
NAND
NVMe2.0
Data Center
Flash
Analytics
Tweets
AI
Write Amplification
Microsoft
NVMe2.0
Web3
Compression
Write Amplification
AWS