New NVMe® Command Sets
Zoned Namespace (ZNS) & Key Value (KV)

Matias Bjorling, Distinguished Engineer and Country Manager, R&D, Western Digital

Bill Martin, SSD I/O Standards, Samsung

Sponsored by NVM Express organization, the owner of NVMe specifications

Agenda

- Overview of NVMe® Command Sets
- ZNS Command Set
- KV Command Set

NVMe® Command Sets Overview

- NVMe 2.0 specifications included multiple Command Sets
- Each Namespace
  - Associated with a single NVMe Command Set
  - Utilizes the current NVMe base specification
    - Administrative commands
- Queue definitions
- Log pages
- Asynchronous Event Notification
- NVMe over PCIe® or NVMe over Fabrics

Why Zoned Namespaces (ZNS)?

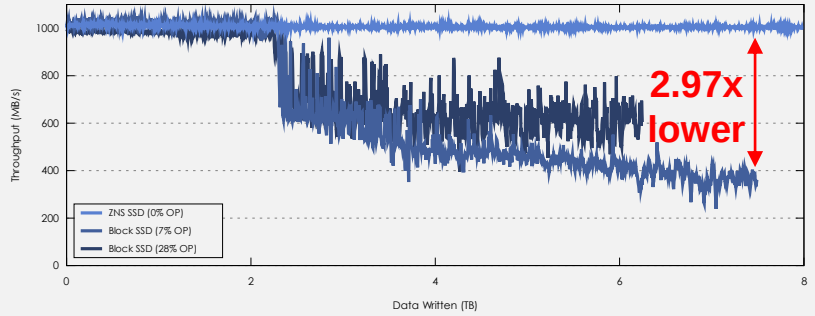
Motivation

- SSDs internal garbage collection (GC) and its write amplification (WA)
  - Inherent mismatch between the block interface and SSDs NAND media
  - Host writes mixed onto the same media, increases GC burden
  - **Lowers SSDs performance** and **increases cost at scale**

"To achieve these levels of device-level write amplification (1.1x & 1.4x), flash is typically overprovisioned by 50% (…) but reducing flash overprovisioning while maintaining the current level of performance is an open challenge at Facebook."

Source: The CacheLib Caching Engine: Design and Experiences at Scale. USENIX OSDI 2020

Throughput



Source: ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. USENIX ATC 2021

Cost

	General		CacheLib (7.68TB workload)	
	SSD	SSD /w ZNS	SSD	SSD /w ZNS
SSD Capacity	7.68T	8T	15.36T	8T
NAND Usable	$584	$584	$584	$584
NAND Over-Provisioning	$39	$0	$661	$0
DRAM	$40	$40	$80	$40
Controller	$6	$6	$6	$6
Other	$10	$10	$10	$10
Total Drive Cost	$679	$640	$1341	$640

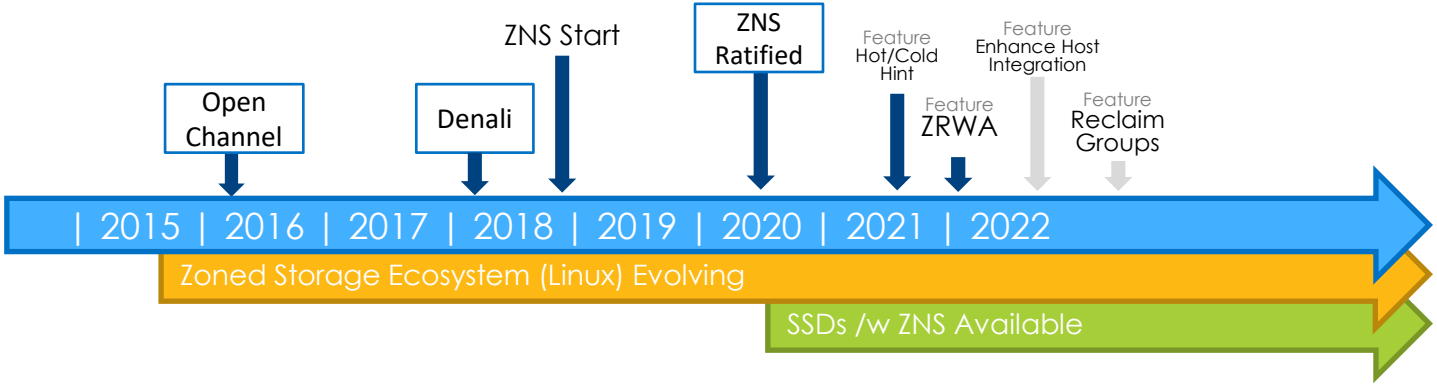
Performance Parity
2x Cost!

Source: https://www.soothsawyer.com/best-online-ssd-cost-calculator

Zoned Namespaces

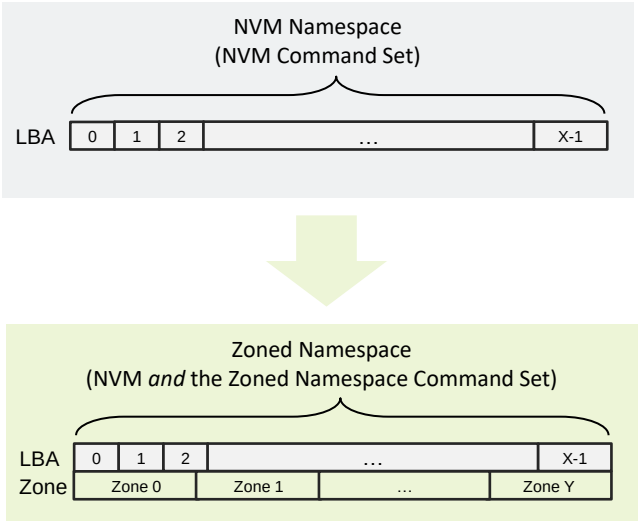
Development

- Industry need for a standardized approach to direct data placement aligned to SSD's media characteristics
- ZNS Task Group was formed to work on what became the Zoned Namespace Command Set
  - TP work began late 2018 and was ratified June 2020
  - Zoned Namespace Command Set 1.1 specification was release June 2021
- ZNS support in Linux since June 2020, and SSDs with ZNS support announced shortly after
- New features being developed
  - ZRWA, Zone Data Hot/Cold hint, namespace improvements, and reclaim groups

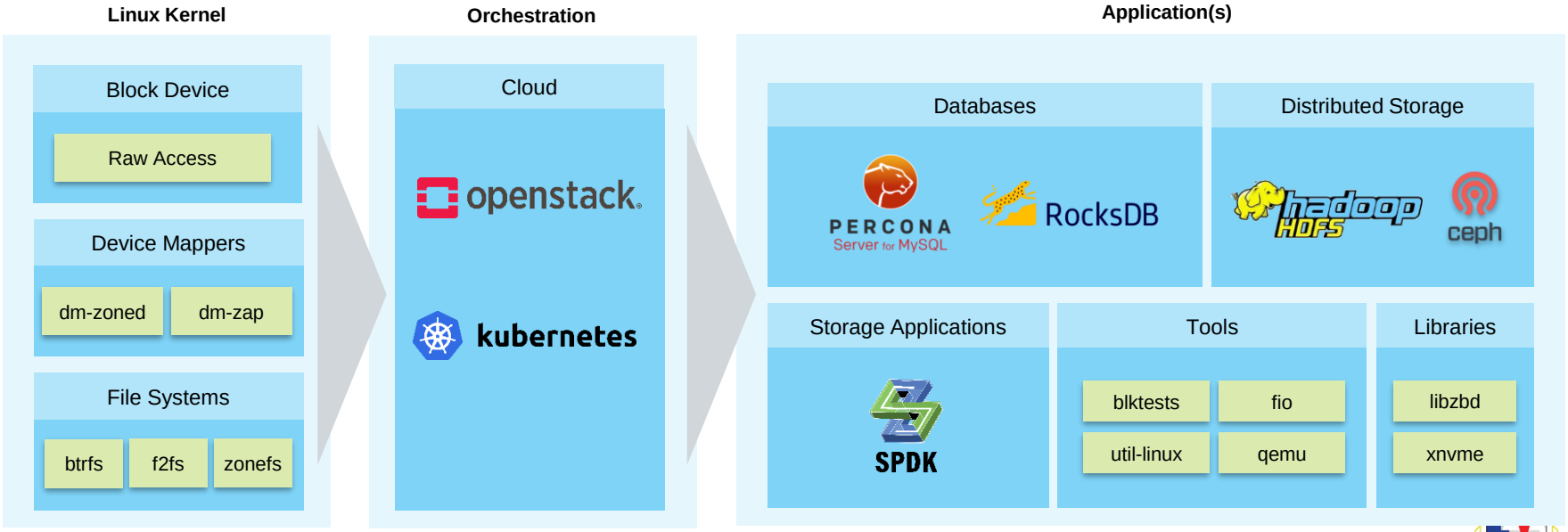


Architecture Overview

- What is a Zoned Namespace?
  - A namespace that supports the common NVM Command Set, and extended with the concept of zones
  - Logical blocks are divided into fixed-sized zones which are then utilized for data placement by the host
  - Mimics the ZAC/ZBC models for host-managed SMR HDDs to take advantage of its existing software ecosystem
- Inherits the functionality of the NVM Command Set
  - Logical blocks, addressing, I/O Commands (e.g., Read and Write), Admin Commands, Log Pages, …
  - Adds three new I/O Commands and one new log page.
    - Zone Management Send/Review and Zone Append.

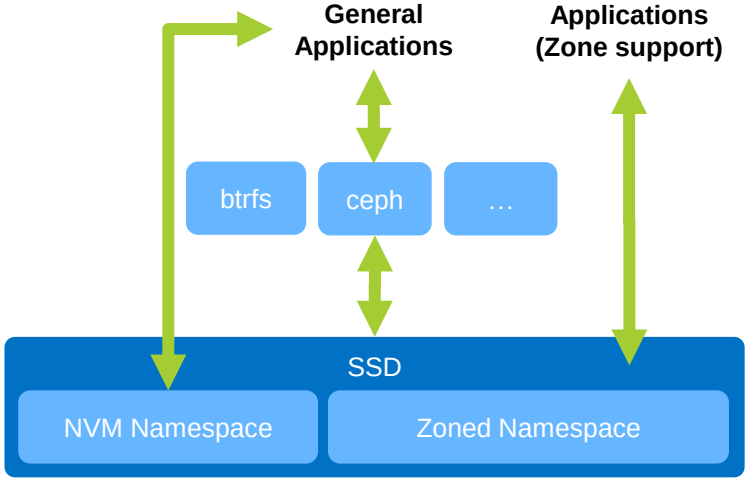


Zoned Storage Software Ecosystem



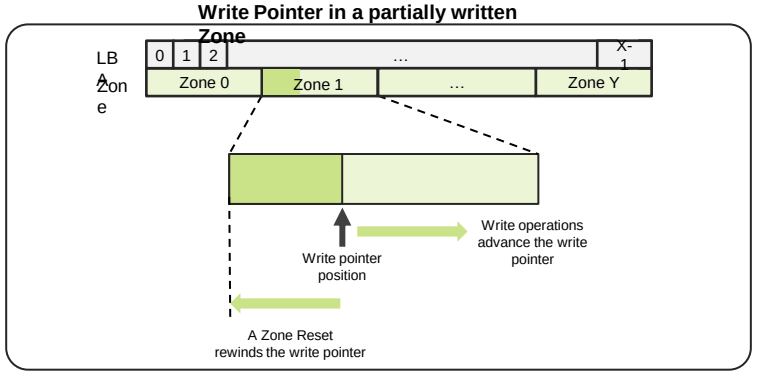
Adopting ZNS at scale

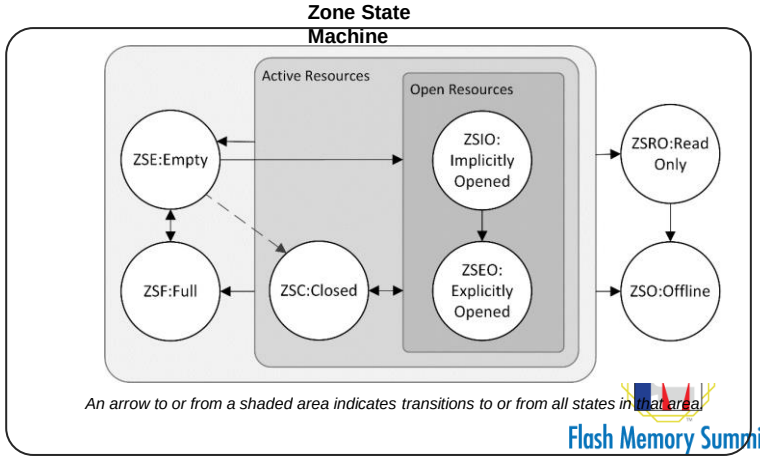
- Raw block device
  - SSD with support for **both NVM & ZNS namespaces**
  - User creates the necessary namespace to be used for the current deployment
  - Easy roll out of new applications and use-cases
- File system with zone support
  - SSD with a ZNS namespace with btrfs, ceph, hdfs, … on top
  - Applications works as usual with files
  - Integrates easily into existing deployments
- I/O Heavy Applications
  - File-systems with zone support
  - Applications with zone support
  - Highest performance, but requires specific application support



The Zone Storage Model

- “Sequential Write Required”
  - Write operations must be issued in order to a zone.
- A zone has a write pointer, that communicates where the next write must be issued.
- A zone has a state machine associated:
  - It controls how a zone is accessed. e.g.,
    - Empty or Open -> writes operations are allowed.
    - Full -> write operations fails.
- State machine and other zone attributes are maintained in Zone Descriptors. The Zone Descriptors are accessed using the Zone Management Receive command.
  - Active Resources and Open Resources restrict how many zones can be in specific state.
- A zone’s state can be manipulated by the host by using the Zone Management Send command
  - E.g., Open Zone, Close Zone, Finish Zone, Reset Zone, …





Why Zoned Namespaces (ZNS)?

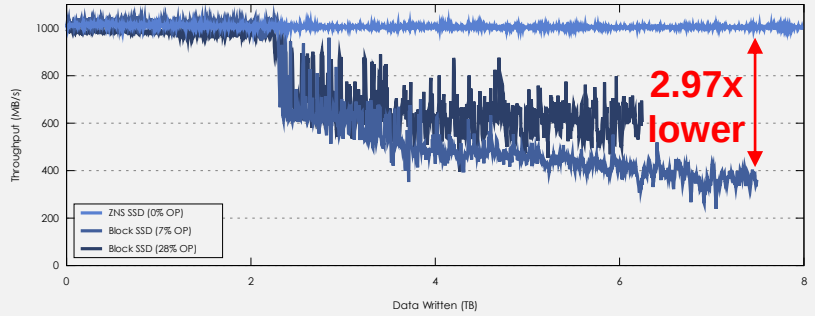
Motivation

- SSDs internal garbage collection (GC) and its write amplification (WA)
  - Inherent mismatch between the block interface and SSDs NAND media
  - Host writes mixed onto the same media, increases GC burden
  - **Lowers SSDs performance** and **increases cost at scale**

"To achieve these levels of device-level write amplification (1.1x & 1.4x), flash is typically overprovisioned by 50% (…) but reducing flash overprovisioning while maintaining the current level of performance is an open challenge at Facebook."

Source: The CacheLib Caching Engine: Design and Experiences at Scale. USENIX OSDI 2020

Throughput



Source: ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. USENIX ATC 2021

Cost

	General		CacheLib (7.68TB workload)	
	SSD	SSD /w ZNS	SSD	SSD /w ZNS
SSD Capacity	**7.68T**	**8T**	**15.36T**	**8T**
NAND Usable	**$584**	**$584**	**$584**	**$584**
NAND Over-Provisioning	**$39**	**$0**	**$661**	**$0**
DRAM	$40	$40	$80	$40
Controller	$6	$6	$6	$6
Other	$10	$10	$10	$10
Total Drive Cost	**$679**	**$640**	**$1341**	**$640**

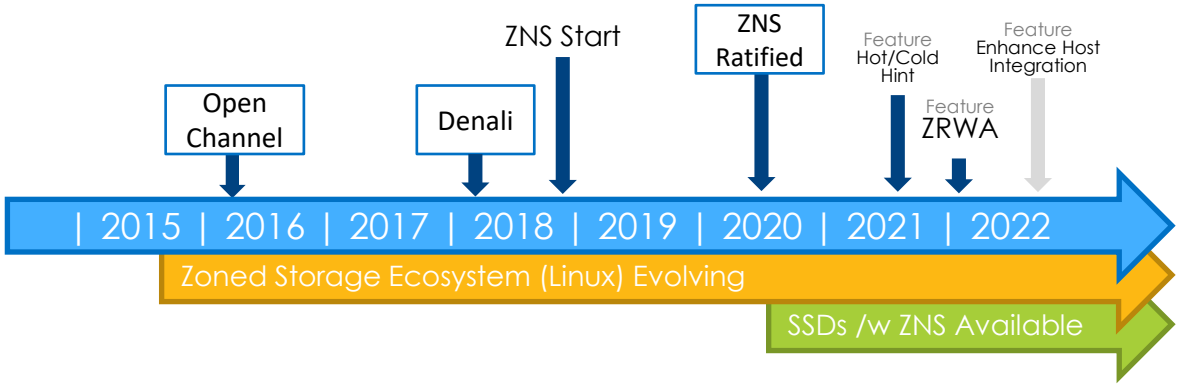
Performance Parity
2x Cost!

Source: https://www.soothsawyer.com/best-online-ssd-cost-calculator

Zoned Namespaces

Motivation & Timeline

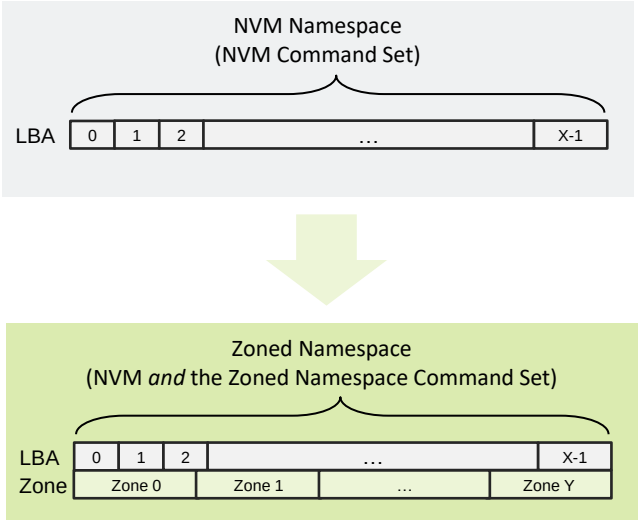
- Industry need for a standardized approach to direct data placement aligned to SSD's media characteristics
- ZNS Task Group was formed to work on what became the Zoned Namespace Command Set
  - TP work began late 2018 and was ratified June 2020
  - Zoned Namespace Command Set 1.1 specification was release June 2021
- ZNS support in Linux since June 2020, and SSDs with ZNS support announced shortly after
- New features added after initial revision
  - ZRWA, Zone hot/cold data placement hint, and namespace improvements



Architecture Overview

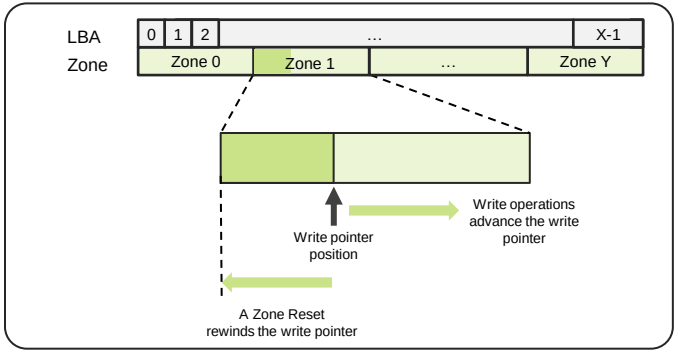
What is a Zoned Namespace?

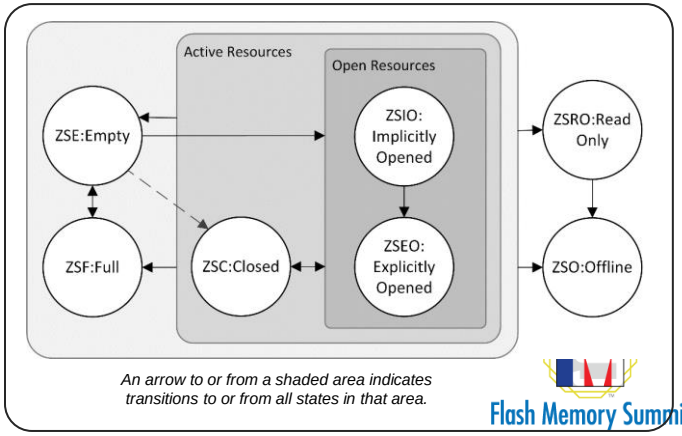
- A namespace that supports the common NVM Command Set, and extended with the concept of zones
  - Builds upon the existing concepts of logical blocks, LBAs, I/O Commands (e.g., Read and Write commands), Admin Commands, Log Pages, …
  - Adds three new I/O Commands
    - Zone Management Send/Received and Zone Append
- Logical blocks are divided into fixed-sized zones which are then utilized for data placement by the host
- Mimics the ZAC/ZBC models for host-managed SMR HDDs to take advantage of its existing software ecosystem



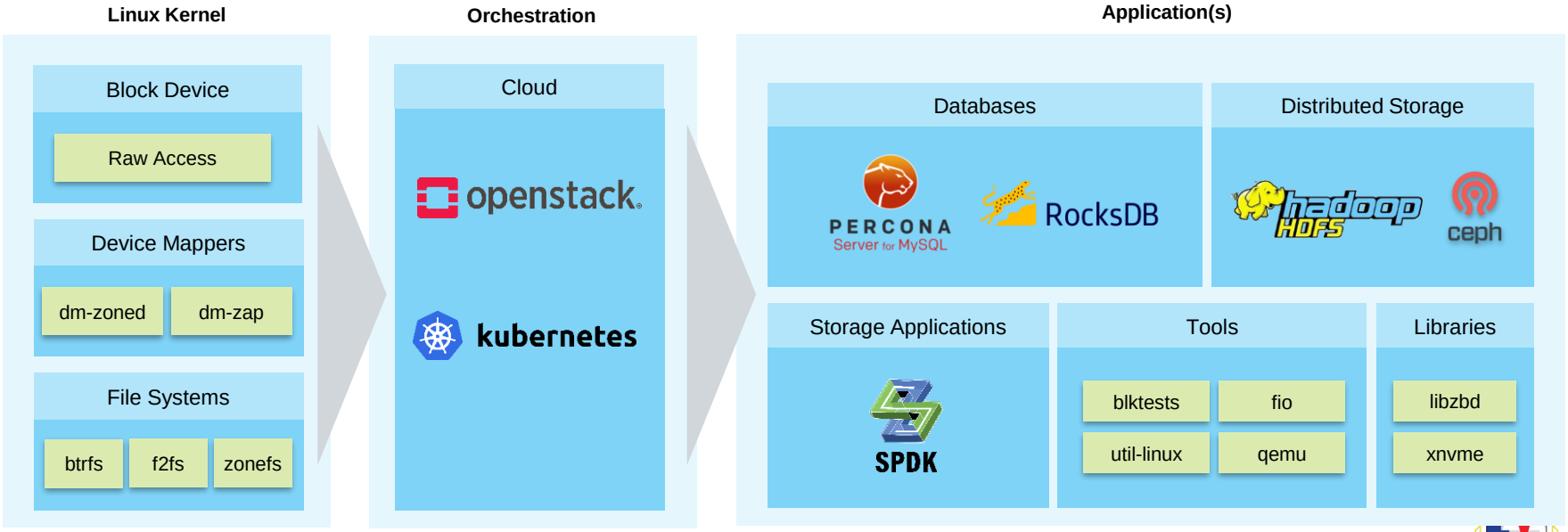
The Zone Storage Model

- Enables an NVMe® device to expose zones such that they align with the write unit of its storage media
- Zoned Namespace defines the zone type “Sequential Write Required”
  - Aligns with the inherent characteristics of NAND flash.
  - Each zone has a state machine associated:
    - It controls how a zone is accessed. e.g., empty, writeable, full.
  - Also has a write pointer, that communicates next write
- A zone’s state can be manipulated by the host by using the Zone Management Send command
  - For example by resetting a full zone.





Zoned Storage Software Ecosystem



Why Key Value (KV)?

Motivation

- SSDs perform a mapping from a logical address to a physical address
  - If that logical address is a key, the host does not have to translate from the key to one or more LBAs
    - Reduces host compute cost for calculating mapping
    - Reduces host memory space for mapping table
    - Improves throughput by eliminating a translation step
- If data is stored as key value pairs on the SSD, then accessing a particular object for performing compute on that object is much easier
  - Pass the key as part of your compute request and the entire value is available to perform the compute on
    - Host does not need to generate a list of LBAs and length that describe the object
    - The list of LBAs does not have to be passed across the IO bus to the SSD
    - Results in:
      - Decrease in CPU utilization
      - Decrease in bus utilization
      - Increase in performance
- The value can be stored in flash in such a way that Garbage Collection is reduced
  - The value is stored in one erase unit on the flash potentially without other data in that erase unit
  - If the value is all that is stored in one erase unit, then after a delete that entire unit may be erased

NVMe® KV basic constructs

Key

Specified in the command

32 bytes maximum length

1 byte minimum length

1 byte granularity

Length specified in the command

- allows 255 bytes

An n-byte key does NOT match a m-byte key

- 00BEh does NOT match BEh

Value

- Length specified in the command
- Up to 4 Gigabytes
- May be zero length

Key-Value Operations

Store

- Data is stored as a single value associated with a key
  - Not updatable in place
  - Not extendable in place
  - Complete value

Retrieve

- Data is retrieved as a single value associated with a key
  - Could be portion of value

Delete

- Key-Value pair may be deleted

List

- Able to list all Keys stored on the device

Store/Retrieve Command

Store Command

Provides ability to store a Key-Value pair

Options

- Compress/no compress
- Do not overwrite
- Do not create

Retrieve Command

Provides ability to retrieve value associated with Key

Options

- Decompress/raw data

Size of value returned in the completion queue entry

- Returns the amount of the value that fits into the specified host buffer
- Cannot return data starting at an index
  - The host must provide a buffer large enough to retrieve the entire value

Exist/List Command

Exist Command

- Takes a Key as an input
- Returns a status of 00h if the Key-Value pair exists
- Returns a status of Key Does Not exist if the Key-Value pair does not exist

List Command

- Returns a list of Keys that exist on the device
- Starts from the Key provided in the command
- NOT in sorted order
- Idempotent if there are no intervening Store or Delete commands
- Does Not return value length associated with each key

Use Cases

Storing photos or videos as a single addressable object

Storing records associated with a unique identifier

- Medical record
- Employment record

Personal profiles

Benefits of Key-Value Reviewed

Removes a translation layer (performance benefit)

Allows storage device to manipulate data based on content

- Search values for a particular pattern
- Perform encoding on value

Removes provisioning overhead

- No pre-assigned mapping of logical to physical association
- Limit to the address range is not based on size of physical storage

Key may be unique across multiple devices

Questions?

Architected for Performance