Key Value Standardization

Bill Martin
Samsung
Agenda

- Key-Value architecture
- Why is Key-Value Important
- Standardization
  - NVMe Standardization
  - SNIA Standardization
- Open Source Libraries
Key Value Architecture

Where does Key Value fit in your architecture
## Block Storage vs Key-Value

<table>
<thead>
<tr>
<th>Block Storage</th>
<th>Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data stored in blocks of a fixed size</td>
<td>Data is stored as unstructured data</td>
</tr>
<tr>
<td>Data addressed by a Logical Block Address (LBA)</td>
<td>Data addressed by a key</td>
</tr>
<tr>
<td>LBA is a fixed number of bytes</td>
<td>Key is variable length</td>
</tr>
<tr>
<td>Storage space allocated in integer multiples of block size</td>
<td>Storage space is allocated in increments of bytes</td>
</tr>
<tr>
<td>Logical Blocks are associated 1:1 with physical blocks</td>
<td>Value is associated with amount of physical storage necessary</td>
</tr>
</tbody>
</table>
Key-Value vs Object Storage

**Key Value**
- Data stored based on a key on native KV device
- Key is variable length
- Storage provides mapping of Key to Value
- Key-Value storage is device level only
- No Metadata associated with value
- Value is complete on the Key-Value device

**Block Storage**
- Data stored based on an object identifier on Block storage device
- Object Identifier is fixed length
- Protocol provides mapping of Identifier to Object
- Object Storage may be split across multiple levels
- Metadata is associated with object
- Object may be split across devices
Characteristics of Key-Value Storage

- **Key**
  - Variable length
  - From 1 byte to 32 bytes or more
  - Unique across Key-Value device

- **Value**
  - Variable length
  - From 1 byte to Megabytes or more
Key-Value Operations

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

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

- Deleting
  - Key-Value pair may be deleted

- Listing
  - Able to list all Keys stored on the device
Why is Key-Value Important
Block Architecture vs KV Architecture

Datacenter S/W Infra

Storage Plugin Interface

Key Value Glue Logic

Key Value API

Index

S/W Key Value Store

Log

POSIX API

Block Map

File System

Journal

Block Interface

Block Device Driver

Command Protocol

Map

Block Device

Log

VS

TX/s

↑

WAF, Latency

↓

Datacenter S/W Infra

Storage Plugin Interface

Key Value Glue Logic

Key Value API

Thin KV Library

Index

KV Device

Log

KV Device Driver

Command Protocol

Index
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

- 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
Standardization

NVMe Key-Value Command Set

SNIA Key-Value API
NVMe Standardization

- NVMe Key-Value Command Set
- Unique specification within NVMe
- Each Namespace is 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
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
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 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
Standardization

NVMe Key-Value Command Set

SNIA Key-Value API
SNIA KV API General concepts

- Aligned with the NVMe KV Command Set
- Open device
- Retrieve device information
  - Capacity, max key length, max value length, etc.
- Create/Delete Key space
  - Equivalent of an NVMe Namespace associated with the NVMe KV Command Set
- Retrieve Key Value Pair information
- Store
- Retrieve
- Delete
- List
SNIA KV API

- Provides synchronous and asynchronous functions
- Grouping
  - Done in the user library (not currently supported in NVMe KV Command Set)
  - Fixed length keys
  - Allows a portion of the key to be used to group keys together
  - Requires to create a group
    - Walks tree to put keys into group
    - Puts keys into group when storing
  - Functions that are enabled by group
    - List within the group
    - Delete entire group
SNIA KV APIs

- Open device
- Retrieve device information
  - Capacity, max key length, max value length, etc.
- Create/Delete Key space
  - Equivalent of an NVMe Namespace associated with the NVMe KV Command Set
- Retrieve Key Value Pair information
- Store
- Retrieve
- Delete
- List
- Delete Group
Open Source Libraries
Open Source code

- SNIA KV API, Kernel driver, and emulator:
  - Public github: https://github.com/OpenMPDK/KVSSD
  - KV userspace driver: https://github.com/OpenMPDK/uNVMe
- KV Ceph: Ceph object storage designed for Samsung Key-Value SSD
  - https://github.com/OpenMPDK/KVCeph
- Network KV: APIs at host software level abstracting multiple direct attached or remote KV SSDs (NVMeoF coming soon)
  - https://github.com/OpenMPDK/NKV
- KVRocks: RocksDB compatible key value store and MyRocks compatible storage engine designed for KV SSD
  - https://github.com/OpenMPDK/KVRocks
Please take a moment to rate this session.

Your feedback matters to us.