SQUEEZING COMPRESSION AND ENCRYPTION INTO SPDK

Jim Harris and Chunyang Hui

AGENDA

- SPDK and Accelerators
- libreduce Compression
- bdev Crypto
- NVMe Crypto

AGENDA

- **SPDK and Accelerators**
- libreduce Compression
- bdev Crypto
- NVMe Crypto

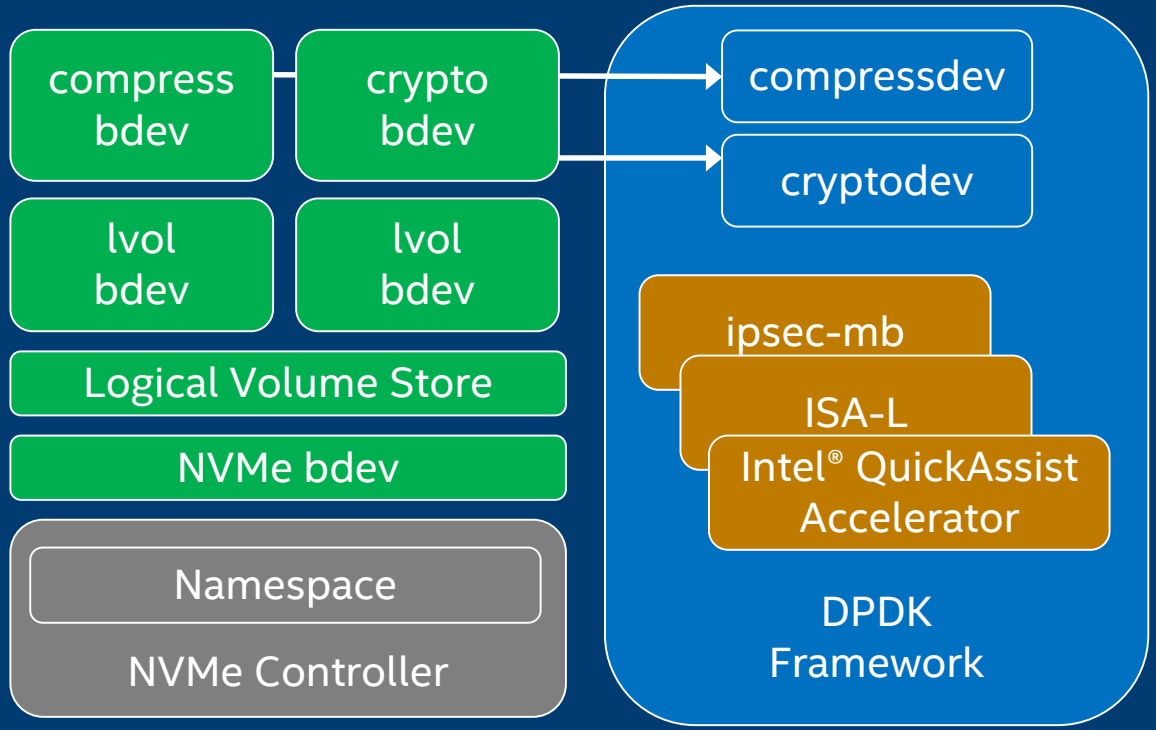
SPDK AND ACCELERATORS

Block services

- Encryption
- Compression

Accelerator Framework

- DPDK Framework
- Common API/ Multiple Backends



AGENDA

- SPDK and Accelerators
- **libreduce Compression**
- bdev Crypto
- NVMe Crypto

LIBREDUCE COMPRESSION

DPDK Framework for compression algorithms

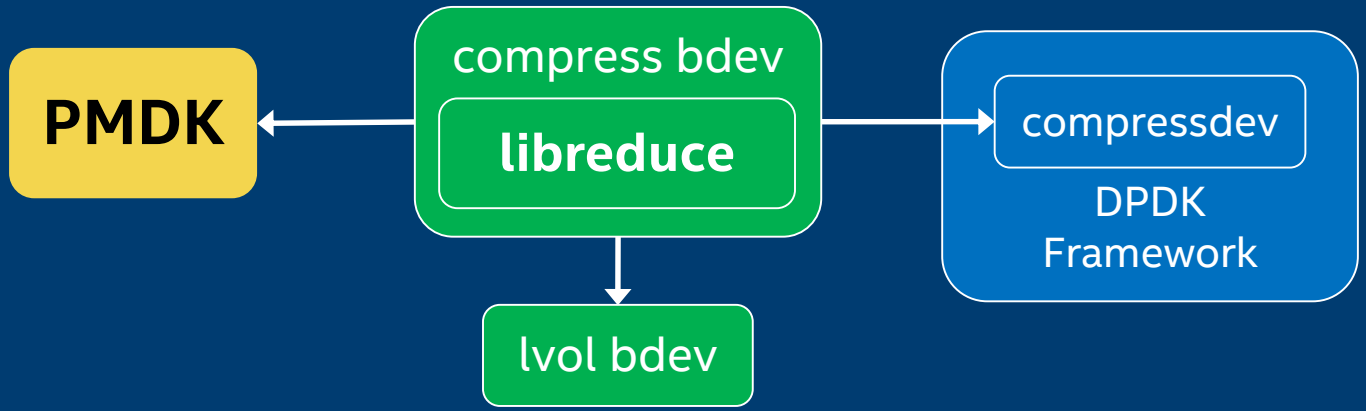
- Accelerators or Optimized SW

PMDK for persistent memory access

- For compression metadata

SPDK Logical Volumes for storage

- For realizing capacity savings



LIBREDUCE OVERVIEW

Block device for backing I/O units

- Typically thin-provisioned SPDK logical volume

Persistent memory file for mapping metadata

- Uses PMDK directly for persistent memory access

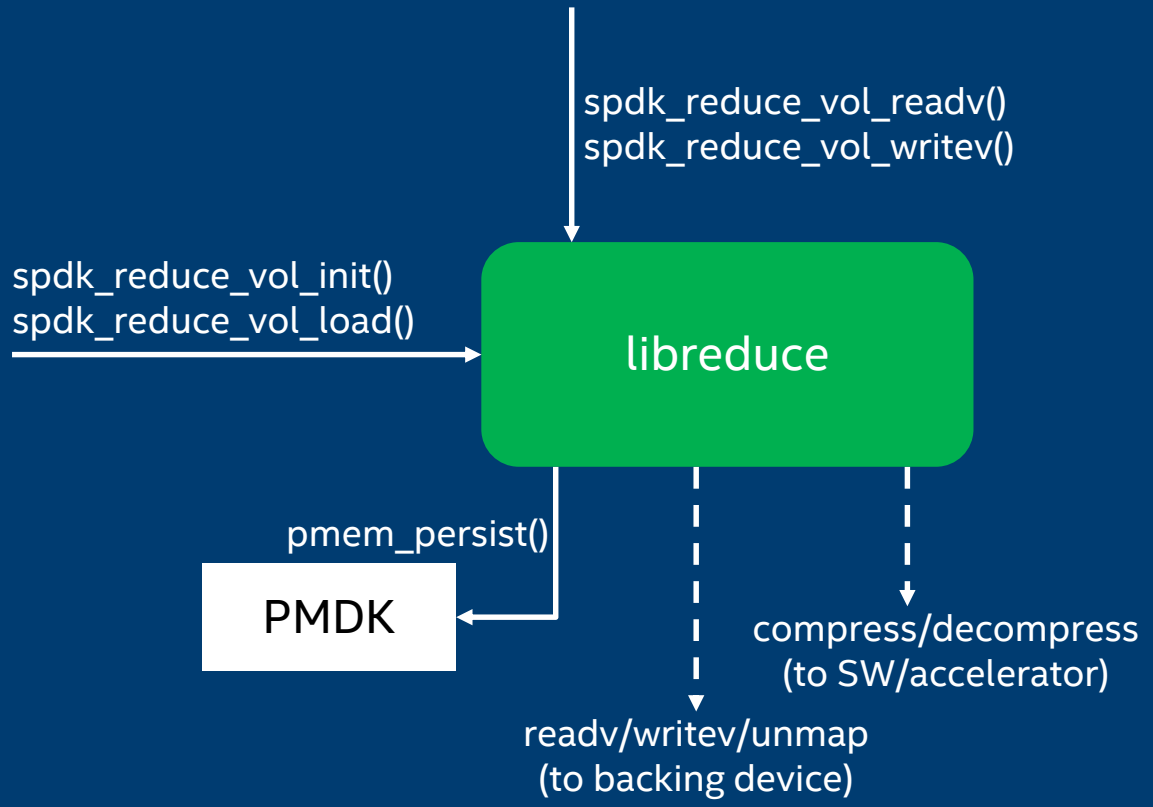
Metadata on block device

- Libreduce parameters
- Path to persistent memory file

Metadata algorithm only!

- Uses standard compression algorithms

INTEGRATION



Independent from SPDK framework and bdev layer

Caller ensures I/Os do not cross chunk boundary

Single-threaded
(per compression volume)

LAYOUTS

Persistent Memory

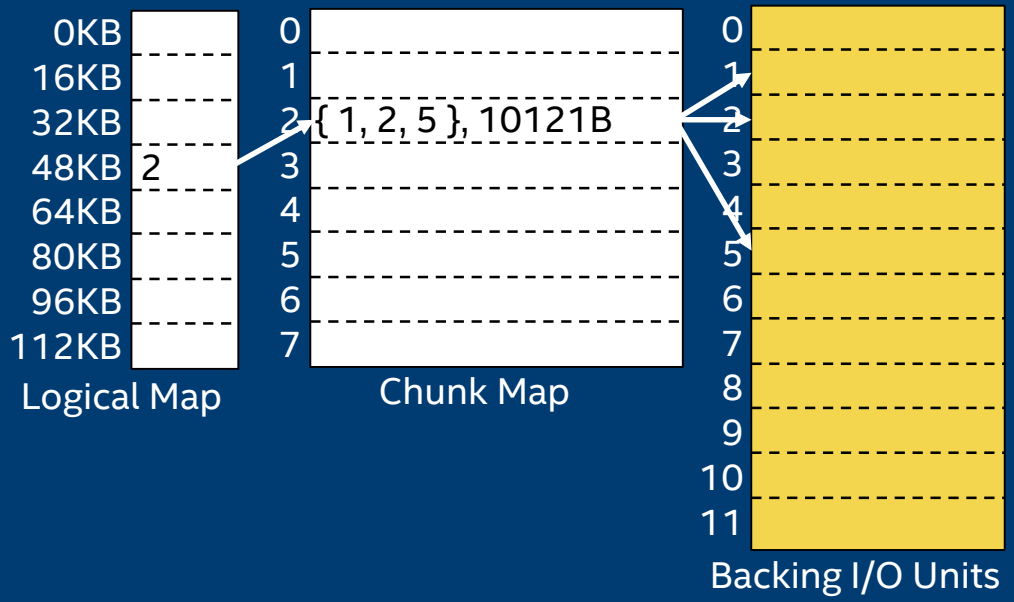
SSD

Backing Device

- Split into I/O units

Persistent Memory File

- Metadata based on chunks
- Chunk map contains chunk entries
- Chunk entry maps a logical chunk to its I/O units on disk
- Logical map contains logical map entries
- Logical map entries map a logical offset to its chunk entry



WRITE 4KB AT OFFSET 0KB

Persistent Memory

SSD

Logical Map: Lookup 0KB => empty

Allocate chunk entry in chunk map => 0

Compress chunk data

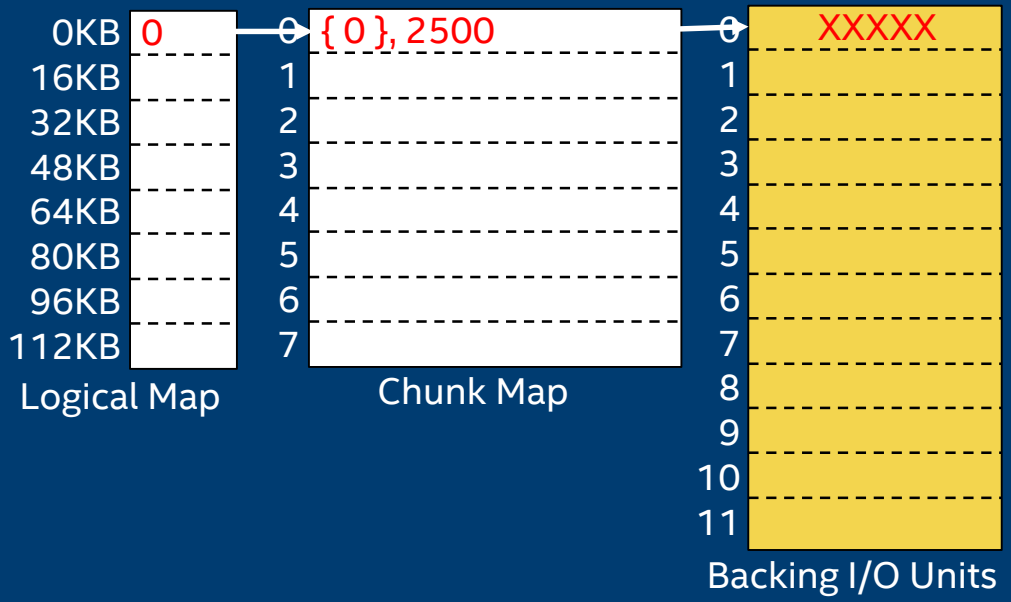
- 4KB user data + 12KB zeroes
- Compresses to 2500 bytes

Allocate 1 backing I/O unit => 0

Write compressed data to SSD

Write and persist chunk entry

Write and persist logical map entry



WRITE 16KB AT OFFSET 64KB

Persistent Memory

SSD

Logical Map: Lookup 16KB => empty

Allocate chunk entry in chunk map => 1

Compress chunk data

- 16KB user data
- Compresses to 14000 bytes

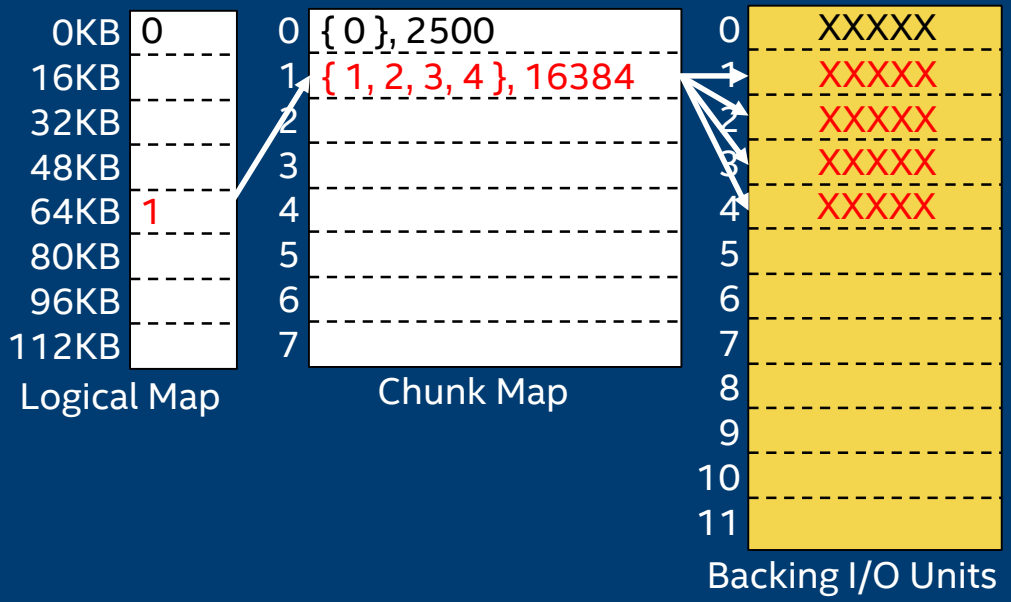
Allocate 4 backing I/O unit => 1, 2, 3, 4

Write uncompressed data to SSD

- 14000 bytes requires 4 4KB I/O units

Write and persist chunk entry

Write and persist logical map entry



WRITE 4KB AT OFFSET 4KB

Persistent Memory

SSD

Logical Map: Lookup 0KB => 0

Read I/O unit 0

Decompress 2500B => 16KB

Merge incoming 4KB

Allocate chunk entry in chunk map => 2

Compress chunk data => 5000B

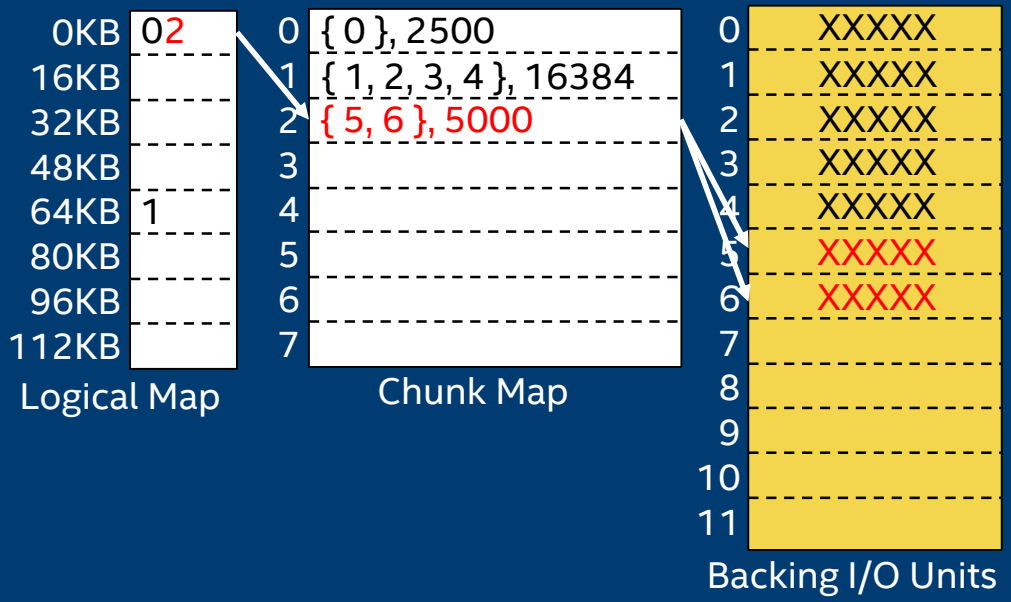
Allocate 2 backing I/O unit => 5, 6

Write compressed data to SSD

Write and persist chunk entry

Write and persist logical map entry

Release old chunk entry and I/O units



TRIM 16KB AT OFFSET 64KB

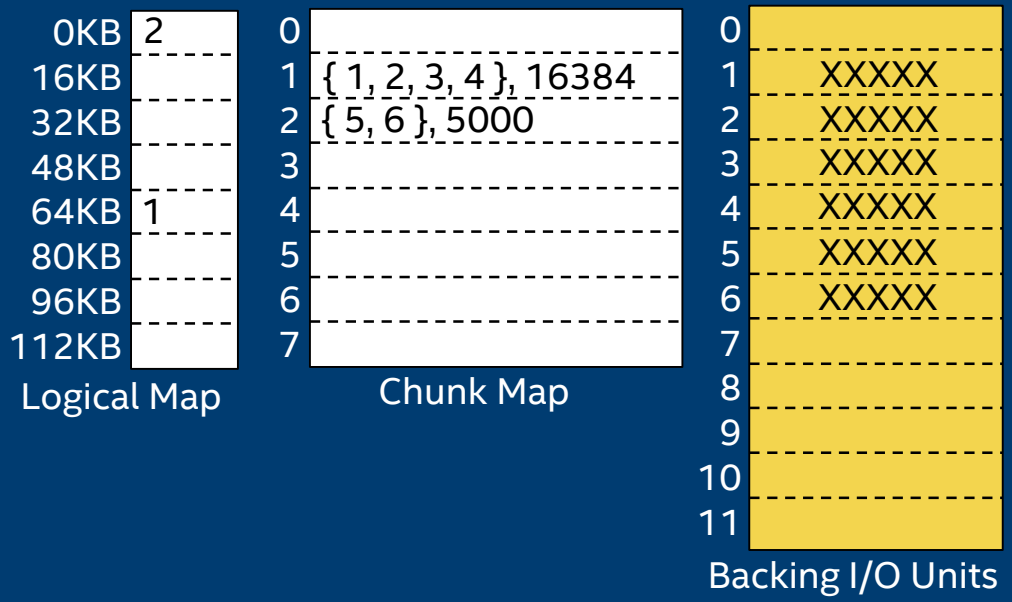
Persistent Memory

SSD

Logical Map: Lookup 64KB => 1

Clear and persist logical map entry

Release old chunk entry and I/O units



READ 4KB AT OFFSET 4KB

Persistent Memory

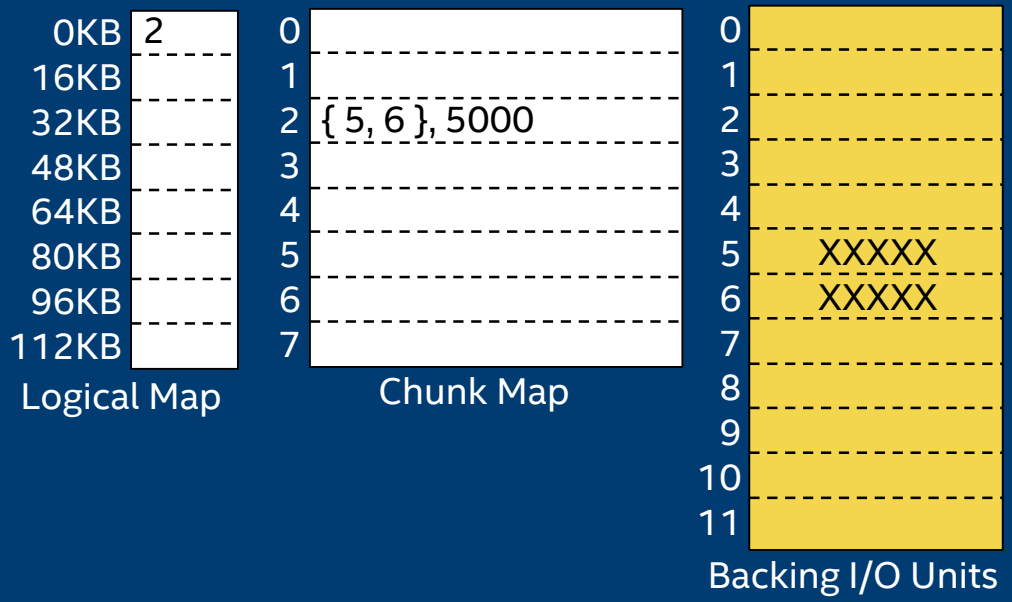
SSD

Logical Map: Lookup 0KB => 2

Read I/O units 5 and 6

Decompress 5000B => 16KB

- Target user buffer for 4KB
- Bit bucket for remaining 12KB



NEXT STEPS

spdk_reduce_vol_unmap

- With sub-chunk allocation masks

Performance data with persistent memory!

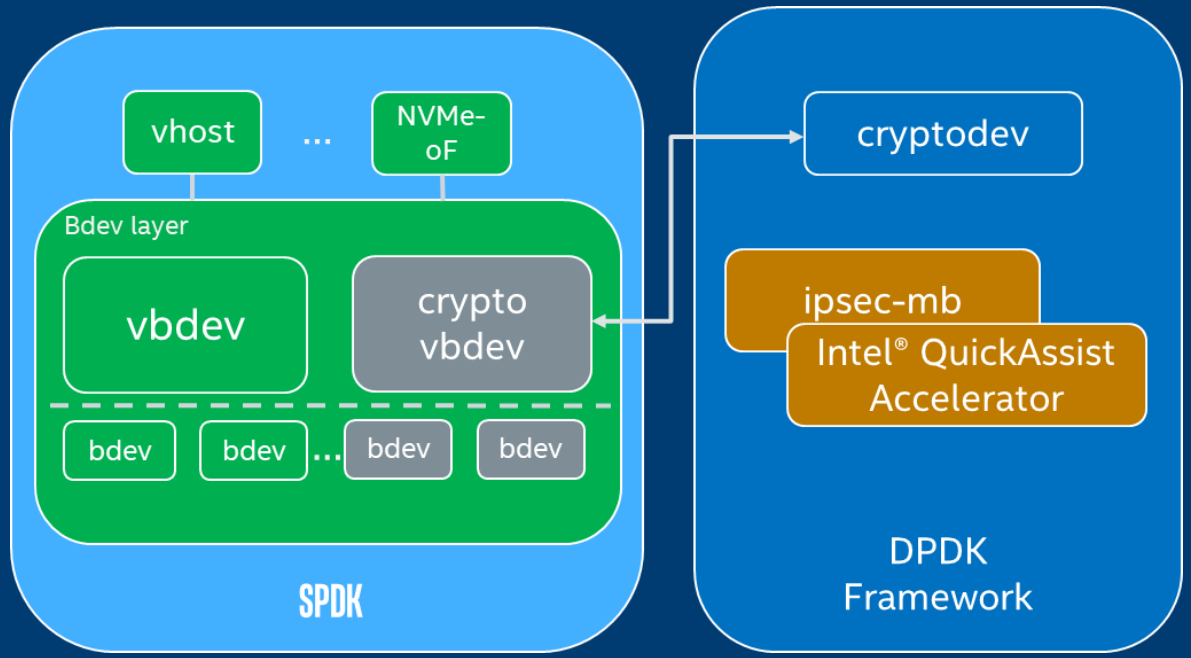
Additional on-disk metadata parameters (i.e. compression algorithm)

AGENDA

- SPDK and Accelerators
- libreduce Compression
- **bdev Crypto**
- NVMe Crypto

CRYPTO VBDEV

- Common API/ Multiple Backends
- Can be layered on any bdev/vbdev
- Ensure data on disk encrypted
- Resources overhead



CRYPTO VBDEV

- Different keys for different bdevs
- Sharing namespace with multiple clients (NVMe via Logical Volume or Split vBdev)
- SPDK doesn't provide key management service

AGENDA

- SPDK and Accelerators
- libreduce Compression
- bdev Crypto
- **NVMe Crypto**

NVME OPAL

- Opal Intro
- SPDK Integration

INTRO TO OPAL

- A set of specifications for SED (self-encrypting drives) developed by TCG (Trusted Computing Group)
- SED referring to disk with built-in FDE (full disk encryption)
- Encryption hardware within the drive

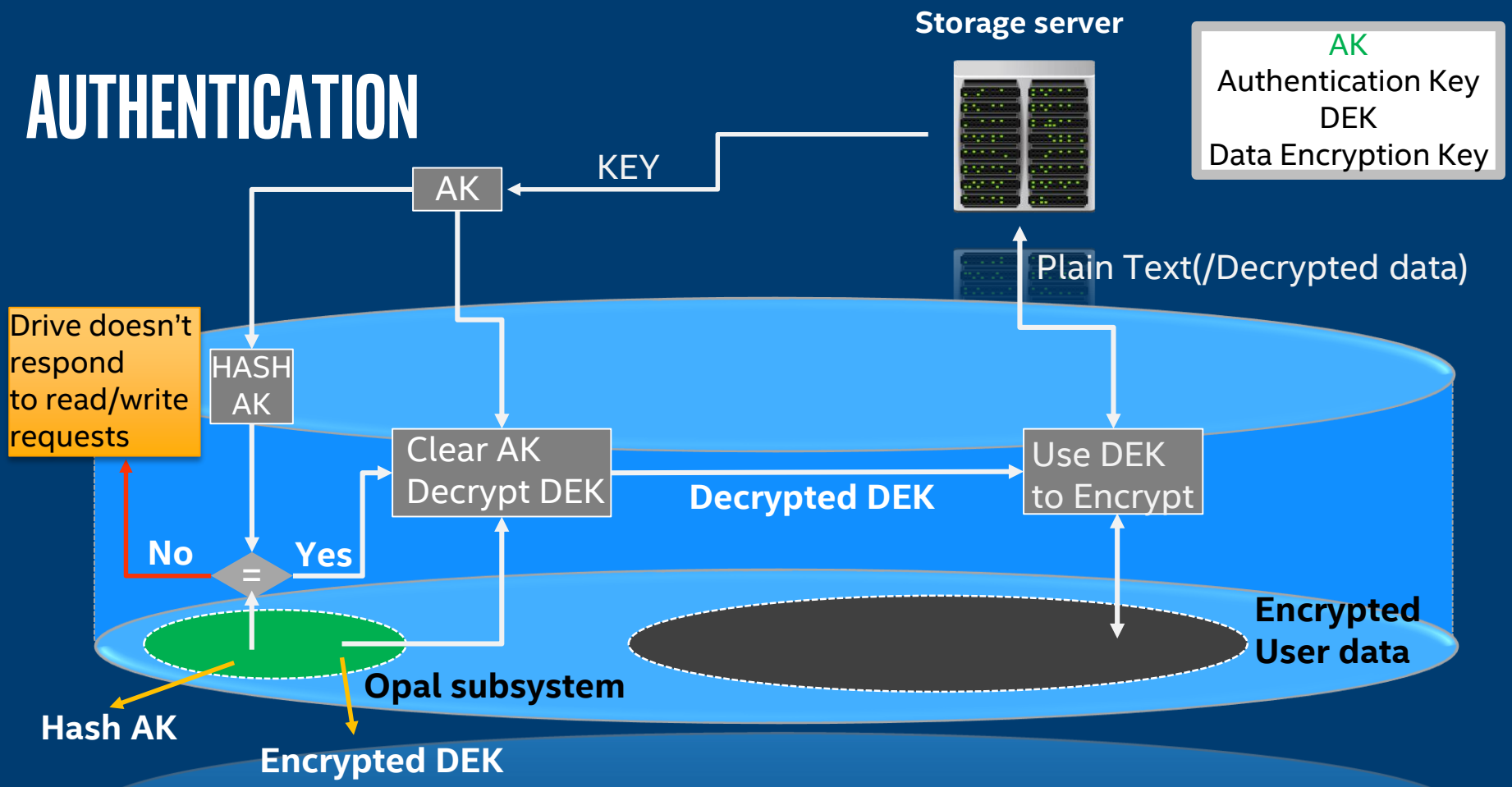
KEYS IN OPAL

- DEK (Data Encryption Key)
  - ✓ The key used to encrypt all of the user data on the drive
  - ✓ Generated randomly by hardware
  - ✓ This key never leaves the drive
  - ✓ When the DEK is changed or erased, all existing data can not be decrypted
- AK (Authentication Key)
  - ✓ The key provided by the user to unlock the drive
  - ✓ A hash of this key may be stored on the drive
  - ✓ Once confirmed, this key is used to decrypt the DEK

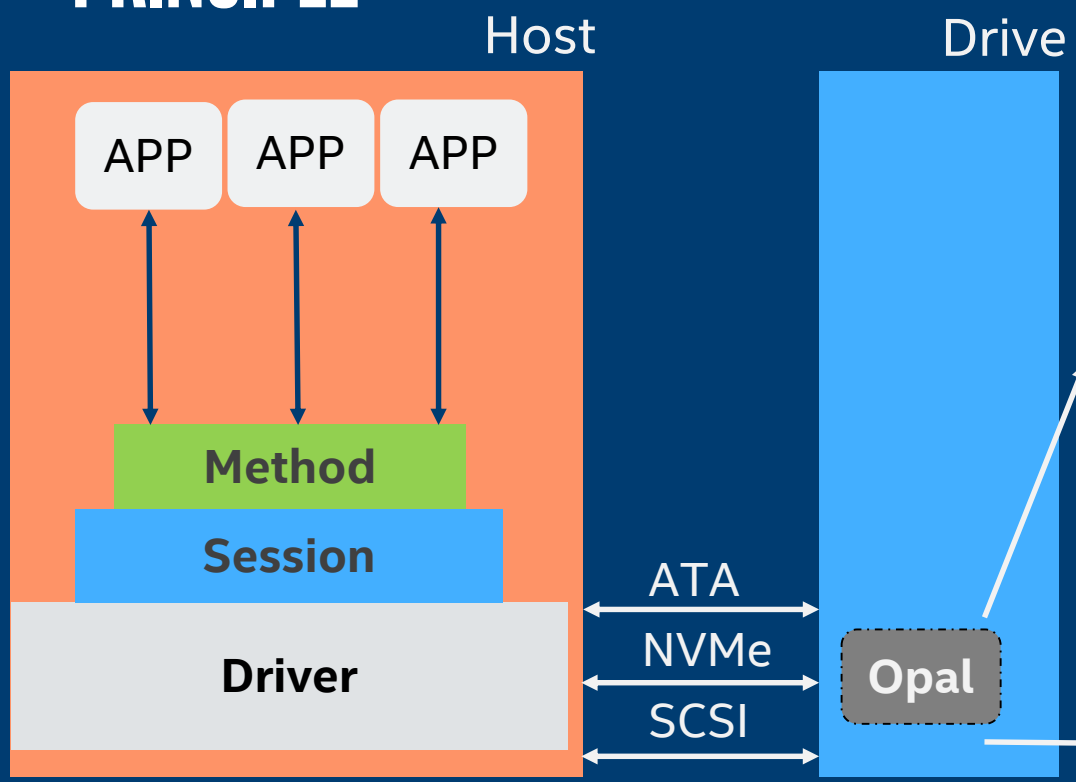
WHY OPAL

- Hardware Based Encryption
  - Operates at full drive speed
  - Scalable to large quantities
  - CPU freed up
- Stronger Security
  - Encryption always on
  - Two-step authentication, unreachable DEK(Data Encryption Key)
- Easy to Use
  - Encryption transparent to users and apps
  - Very easy to do disk sanitize / full erase / retire (only change DEK), user password change

AUTHENTICATION



PRINCIPLE



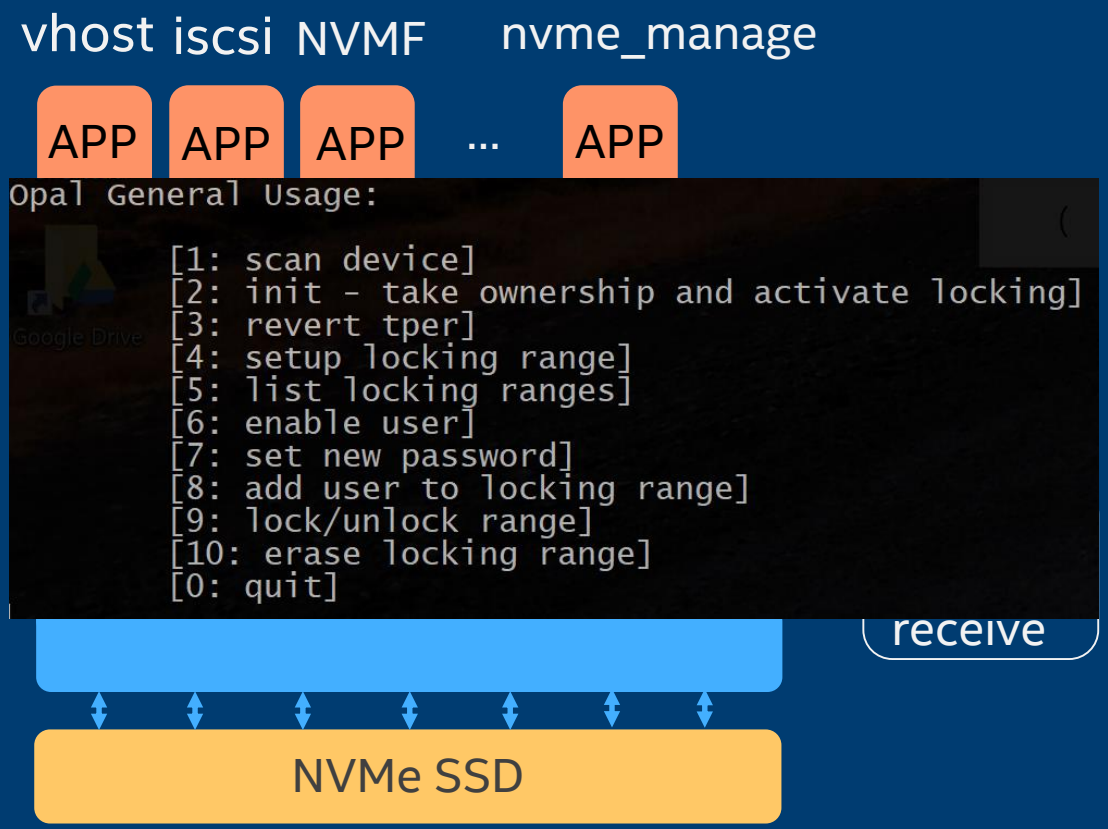
Tables

Table 209 Default Base Template Authorities

Name	UID	Common Name	IsClass	Class	Operation
Anybody	00 00 00 09 00 00 00 01	Anybody	False		Sign
Admins	00 00 00 09 00 00 00 02	Admin	True		
Makers	00 00 00 09 00 00 00 03	Maker	True		
MakerSymK	00 00 00 09 00 00 00 04	Maker	False	Makers	SymK
MakerPuK	00 00 00 09 00 00 00 05	Maker	False	Makers	Sign
SID	00 00 00 09 00 00 00 06	TPerOwner	False		Password
TPerSign	00 00 00 09 00 00 00 07	TPerSign	False		TPerSign
TPerExch	00 00 00 09 00 00 00 08	TPerExch	False		TPerExchange
AdminExch	00 00 00 09 00 00 00 09	Admin	False	Admins	Exchange

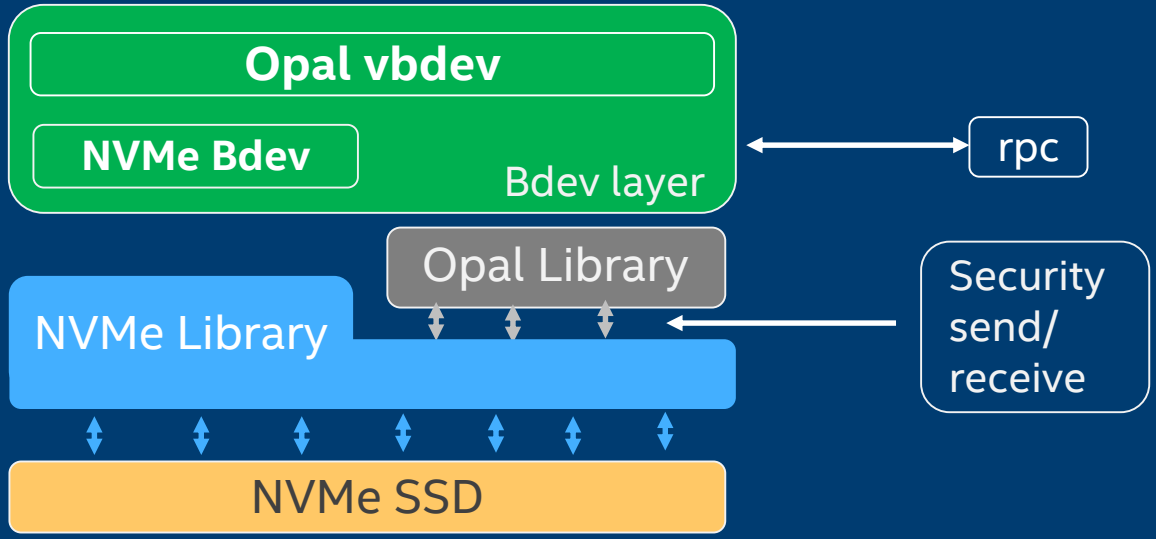
LIBRARY SUPPORT

- SPDK introduce NVMe Opal Library support in v19.04
- Triggered by nvme_manage tool
- Support most of the common operations (initialization, lock/unlock, multi-range, multi-user, revert/erase, etc.)



FUTURE WORK - BDEV SUPPORT

- Opal command directly handled by Opal library
- NVMe IO passthrough to NVMe library
- Opal vbdev layer manages multiple locking range
- Sharable namespace for multiple clients (via locking range)
- Due to hardware constrains, limited number



DRIVE SUPPORT

- Developed and tested under OPAL 2.0
- Should also work for other specs of OPAL family(some features may not be supported by the hardware)
- SPDK OPAL functionality works with any NVMe SSD that supports it
  - INTEL® OPTANE™ SSD DC P4800X SERIES
  - INTEL® SSD DC P4510 SERIES (Specific version, please contact local AE for more info)
  - OPAL supported drive from other companies

HOW TO CHOOSE

Crypto vbdev

Pros

- No specific hardware needed
- Encrypt any bdevs
- More Flexibility

Cons

- Resources overhead
- Performance degradation

Opal vbdev

Pros

- Almost no performance degradation
- No resources overhead
- More functionality

Cons

- Only NVMe bdev supported
- Hardware constraints

Q&A

intel®

BACK UP