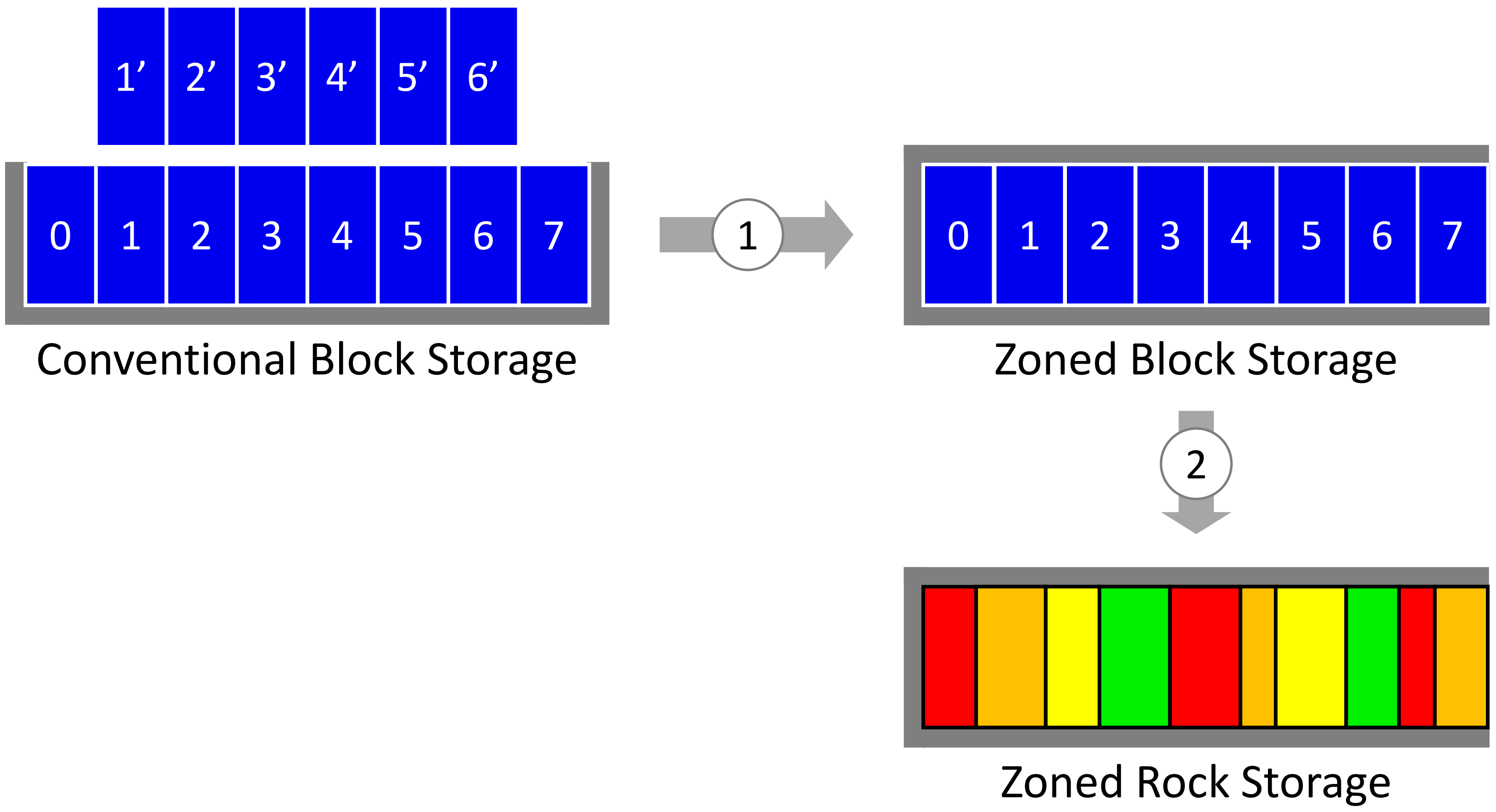
From Blocks to Rocks: A Natural Extension of Zoned Namespaces

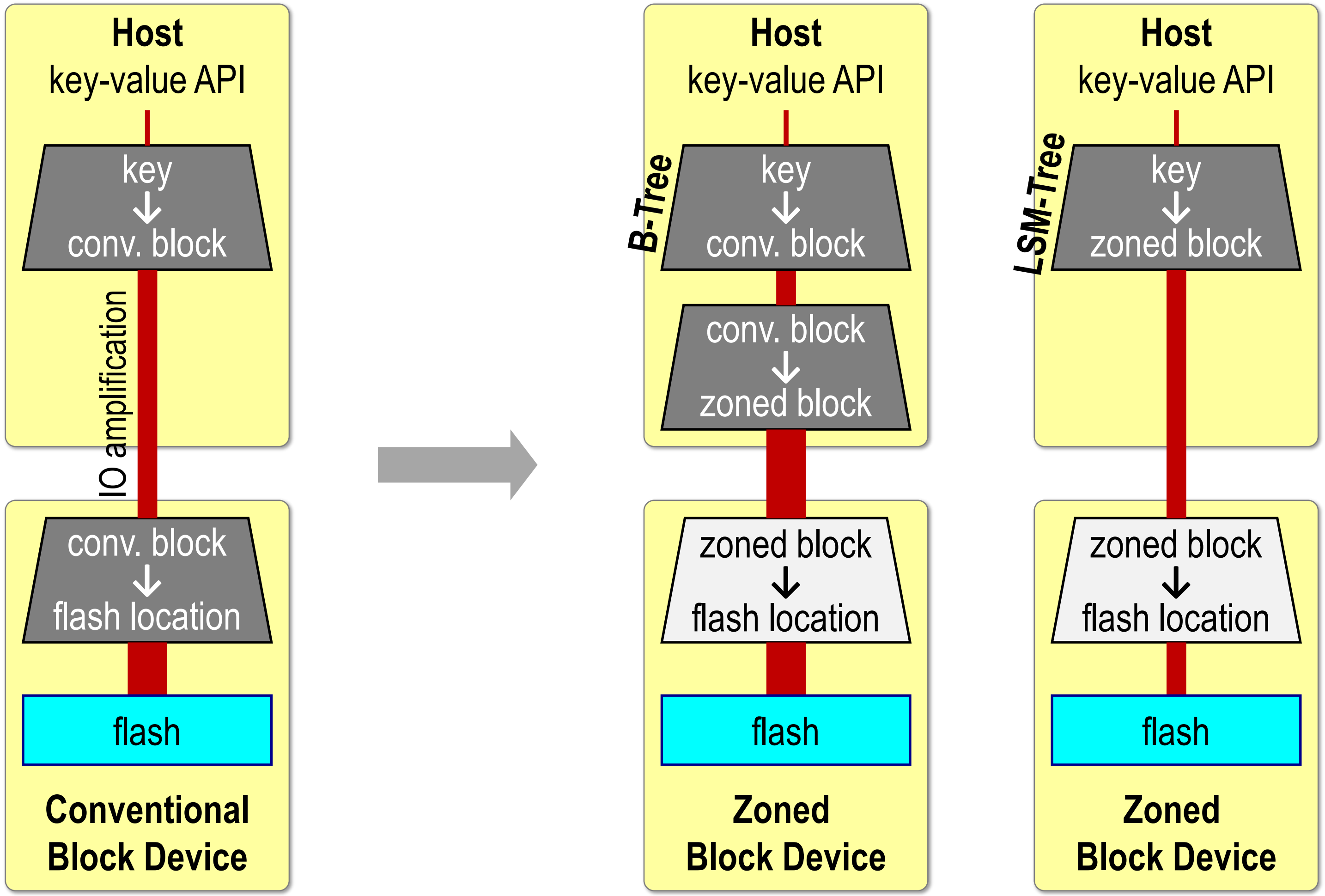
Umesh Maheshwari
Chiku Research

HotStorage 2021

Storage Abstractions



Why Zoned Storage



Why Rocks in Zoned Storage

The Downside

Add little:

- complexity in specification
- overhead in implementation

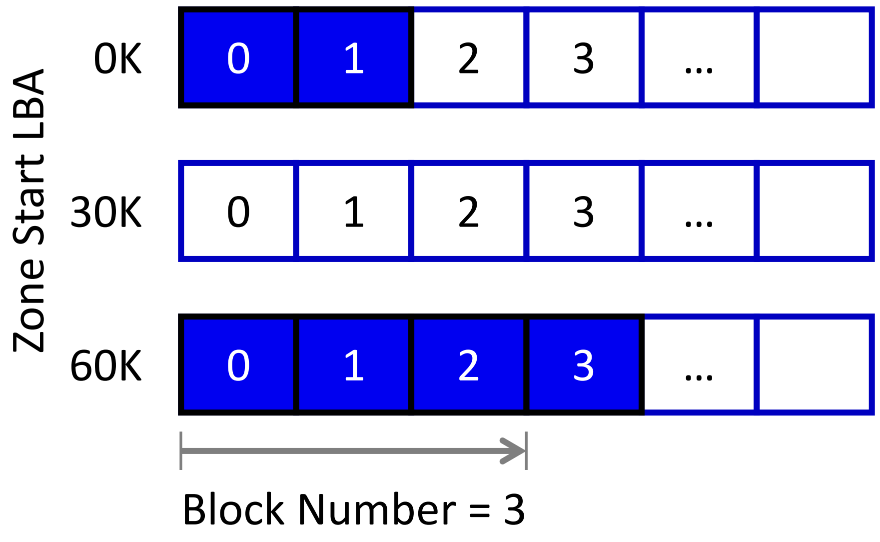
The Upside

Store small/variable-size data efficiently:

- compressed pages
- log records

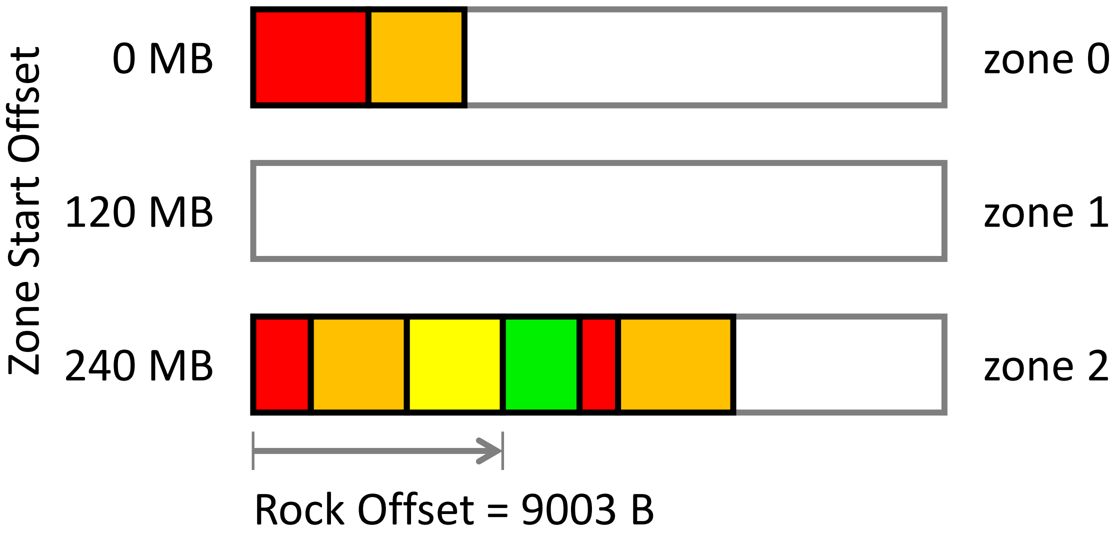
Potential Specification

Zoned Block Namespace (ZBNS)



Block Address = Zone Start LBA + Block Number = 60K+3

Zoned Rock Namespace (ZRNS)



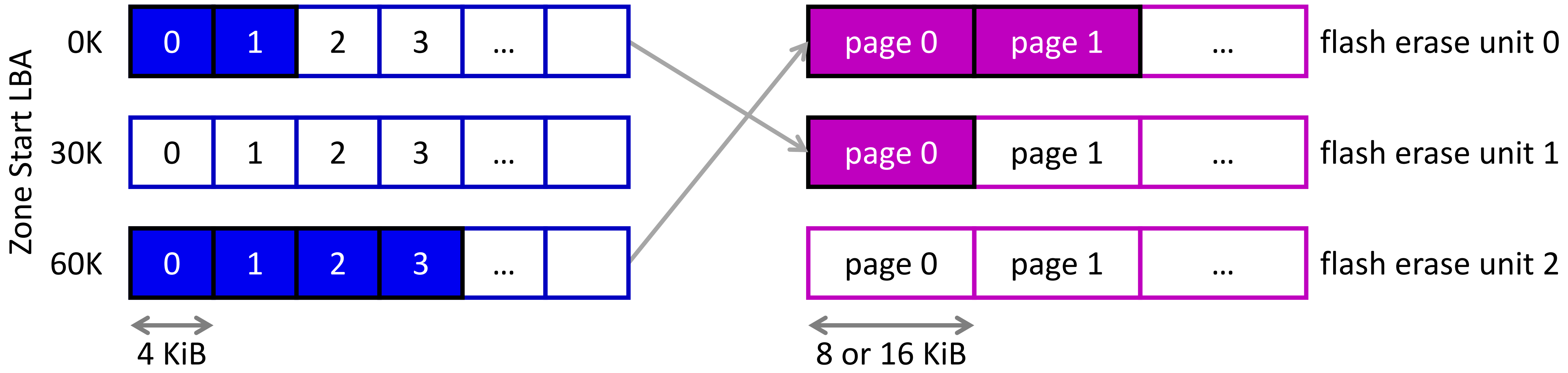
Rock Address = Zone Start Offset + Rock Offset = 240M + 9003

Rock Address = (Zone Number, Rock Offset) = (2, 9003)

Potential Implementation

Zoned Block Namespace (ZBNS)

Physical Locations in Flash

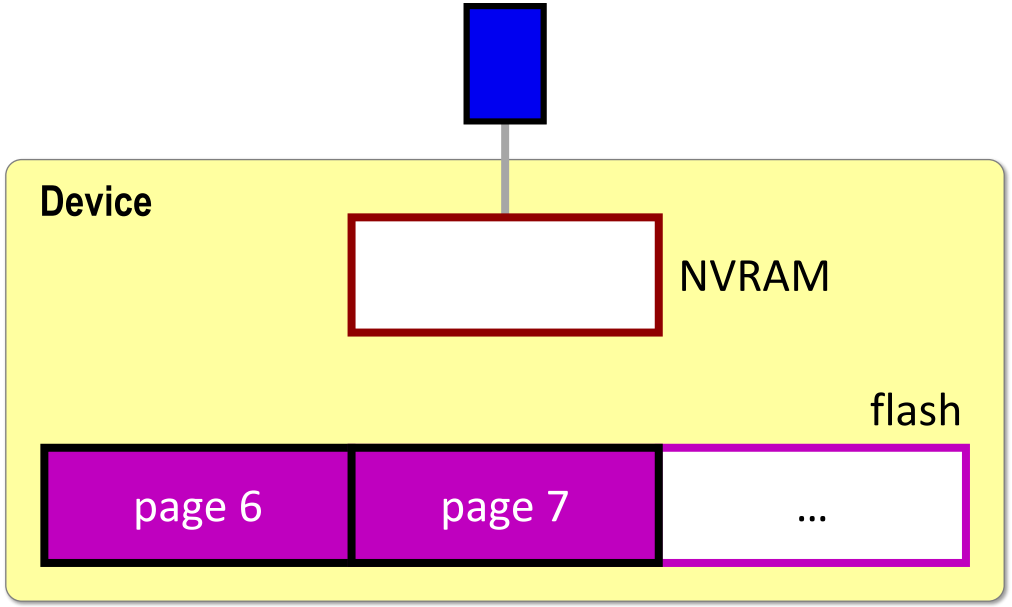


Per-zone map, no per-block map.

Logical block size < flash page size

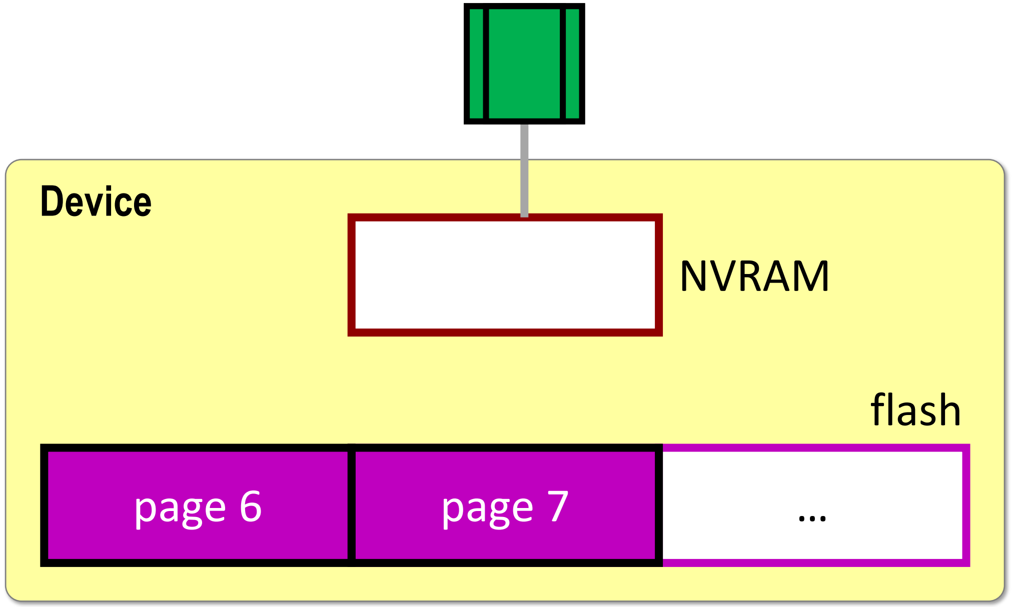
Potential Implementation

Zoned Block Namespace (ZBNS)



- One page-size NVRAM buffer per active zone
- No per-block map
- One command can read/write many blocks

Zoned Rock Namespace (ZRNS)



- One page-size NVRAM buffer per active zone
- No per-rock map
- One command can read/write many rocks
- Can support rocks as small as 16 B

Why Rocks in Zoned Storage

The Downside

Adds little:

- complexity in specification
- overhead in implementation

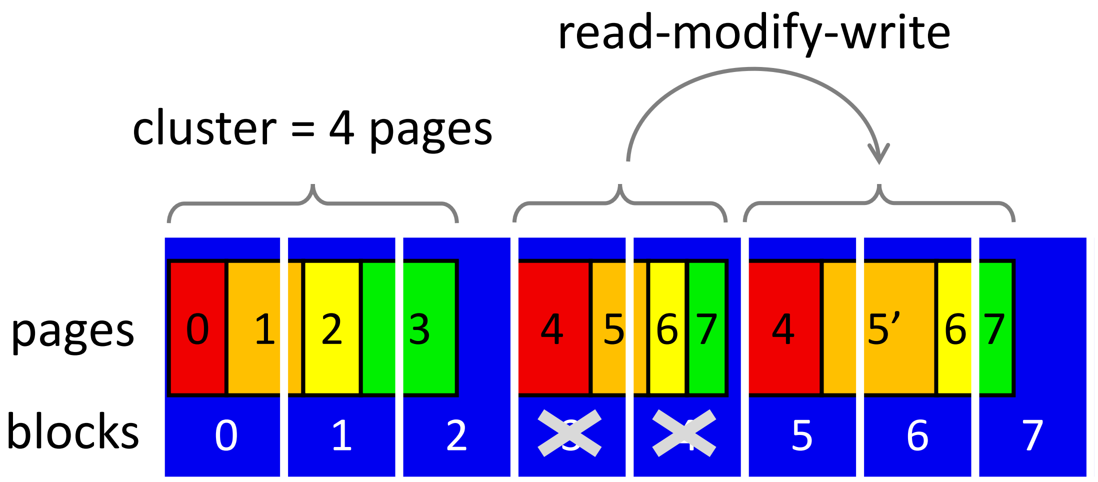
The Upside

Store small/variable-size data efficiently:

- compressed pages
- log records

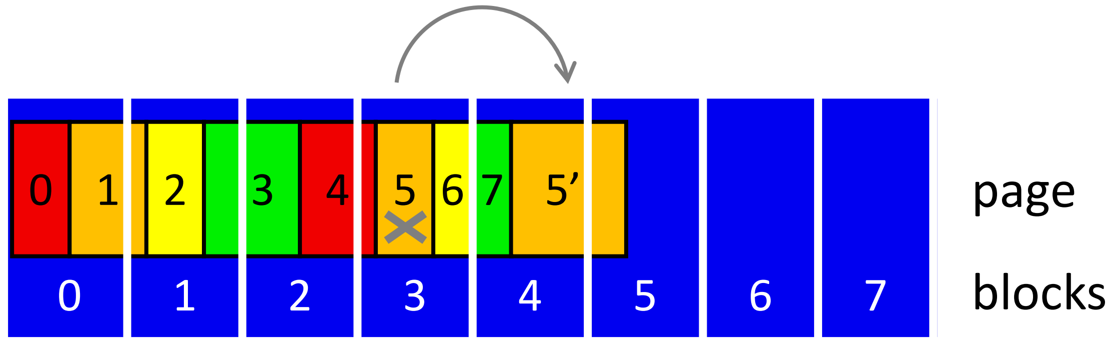
Transparent Compression

1. Block-Aligned Clusters



Unit of garbage = block
e.g., WAFL®, Btrfs, F2FS

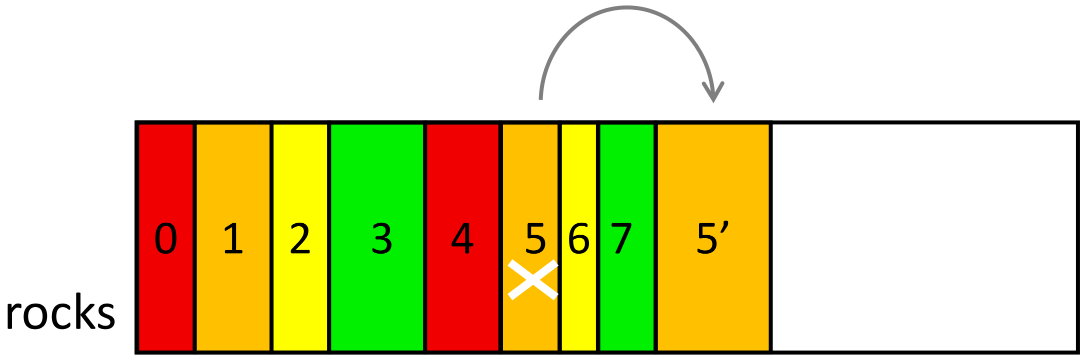
2. Soft Rocks Over Blocks



Unit of garbage = rock
e.g., CASL®

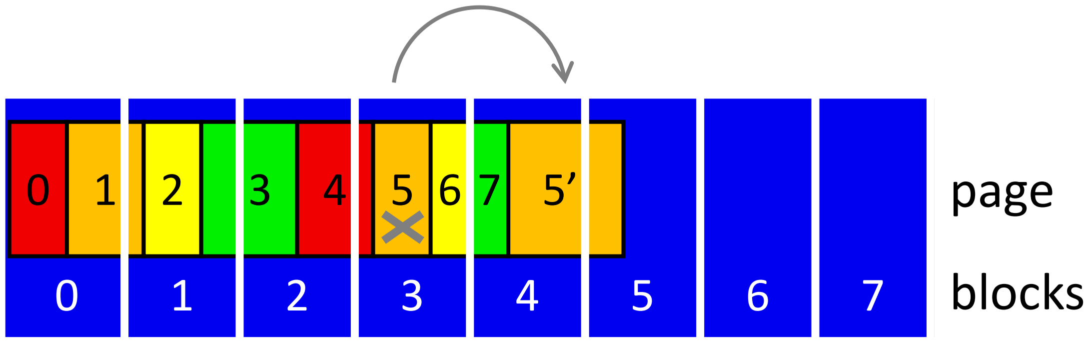
Transparent Compression

3. Device-Level Rocks



- Avoid reading extra bytes from device.
- Avoid redundant checksums on rocks and blocks.
- Offload to device: rock-level copy to optimize GC.

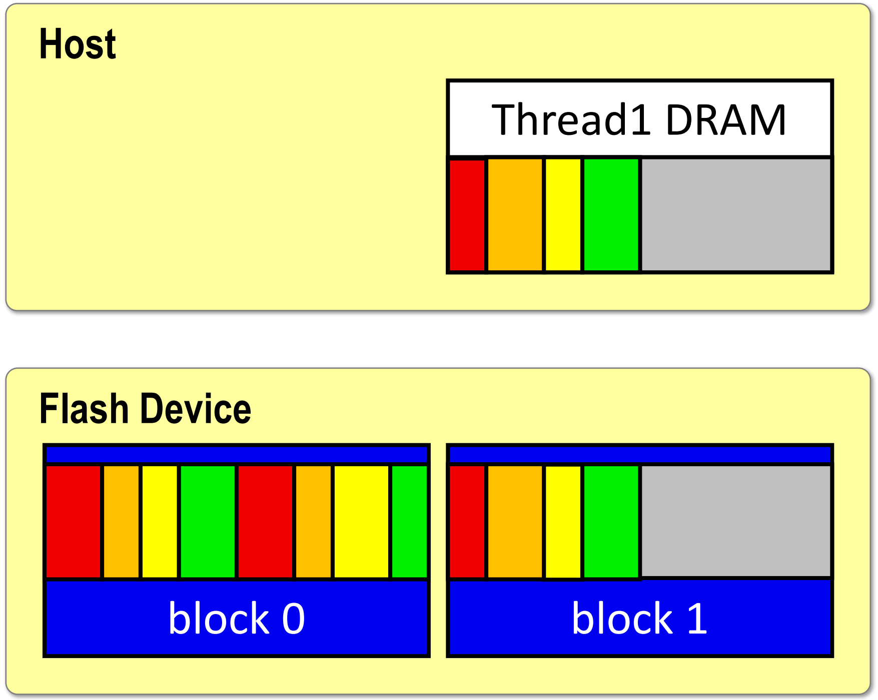
2. Soft Rocks Over Blocks



Unit of garbage = rock
e.g., CASL®

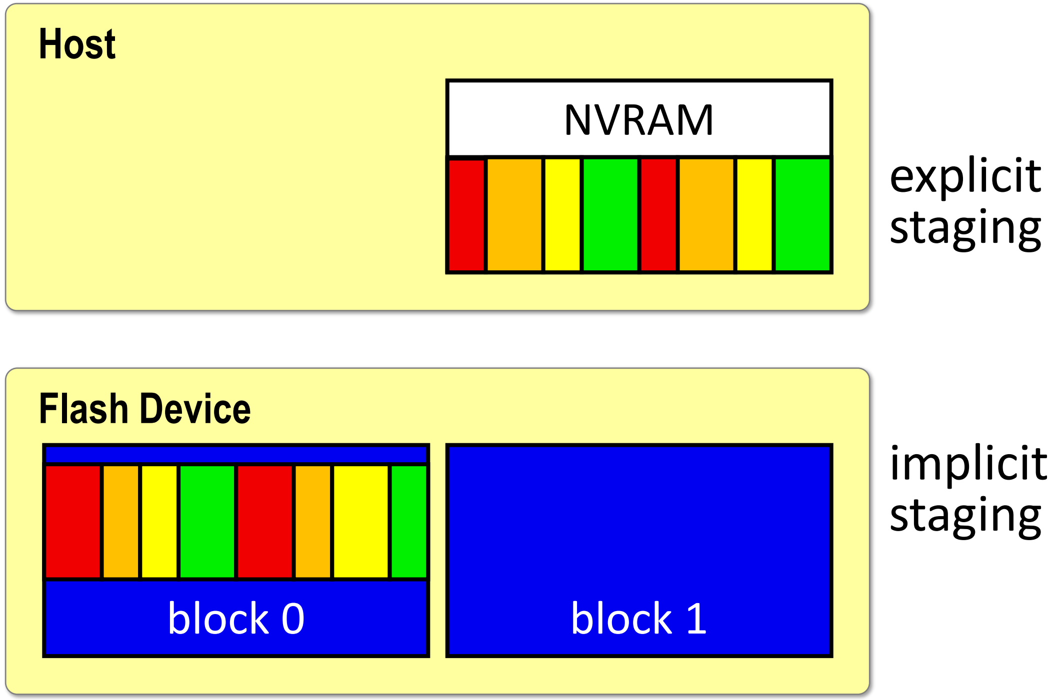
Logging Change Records

1. Rewrite Last Block



- Requires extension ZRWA, not yet standard.
- Can have at most 1 write pending to a block.
- Amplifies bytes transferred.

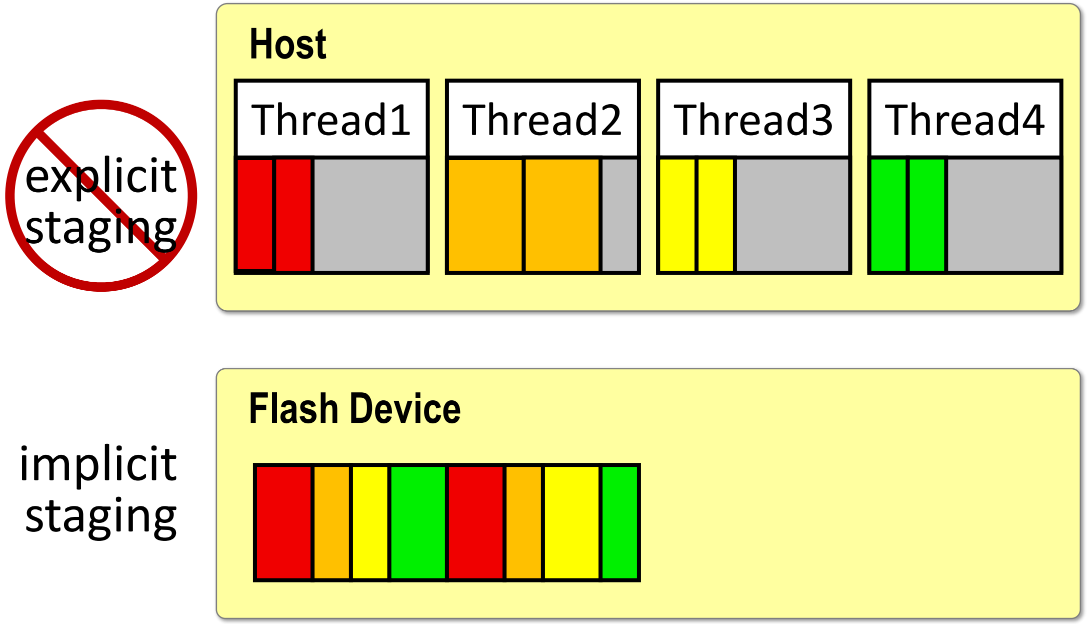
2. Stage in NVRAM



- Explicit staging adds cost and complexity.
- Need for separate replication makes it worse.

Logging Change Records

3. Device-Level Rocks

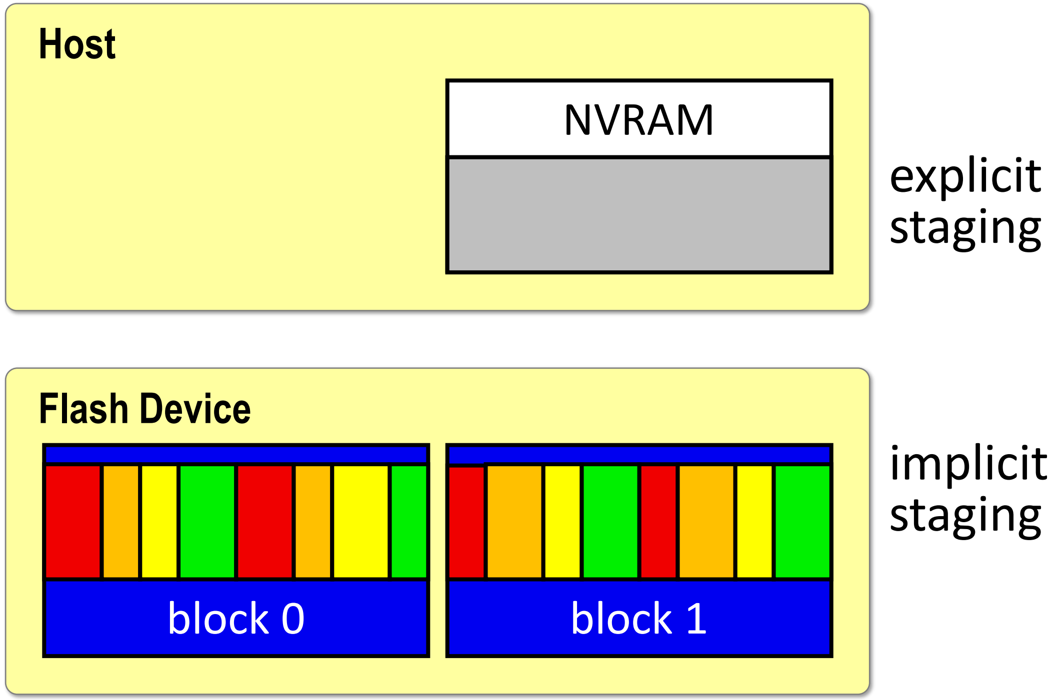


Direct logging without (explicit) staging.

Concurrent appends by multiple threads.

As fast as explicit staging in PCIe-attached NVRAM.

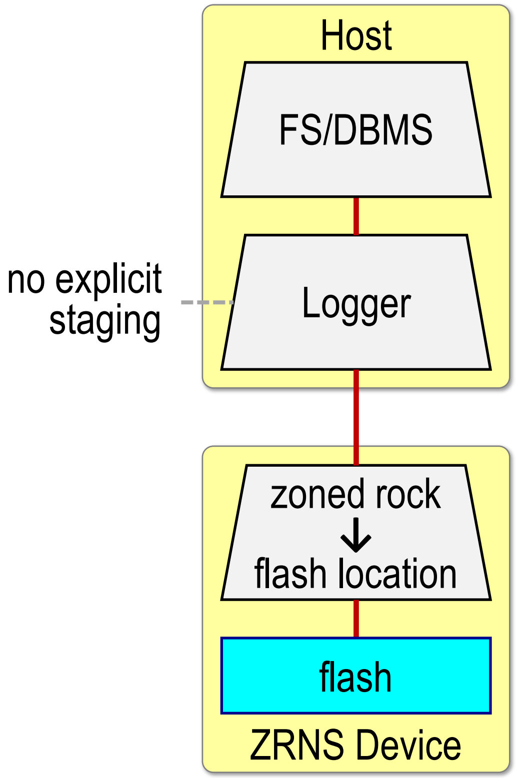
2. Stage in NVRAM



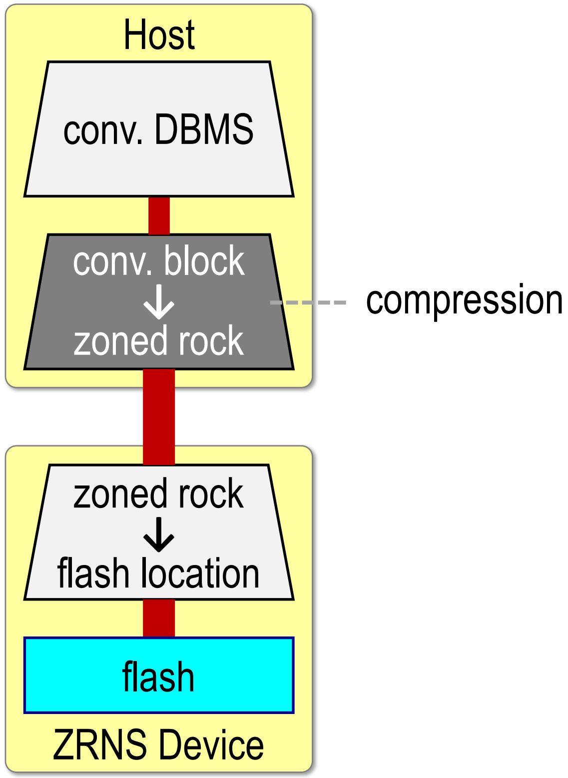
- Explicit staging adds cost and complexity.
- Need for separate replication makes it worse.

Future Directions

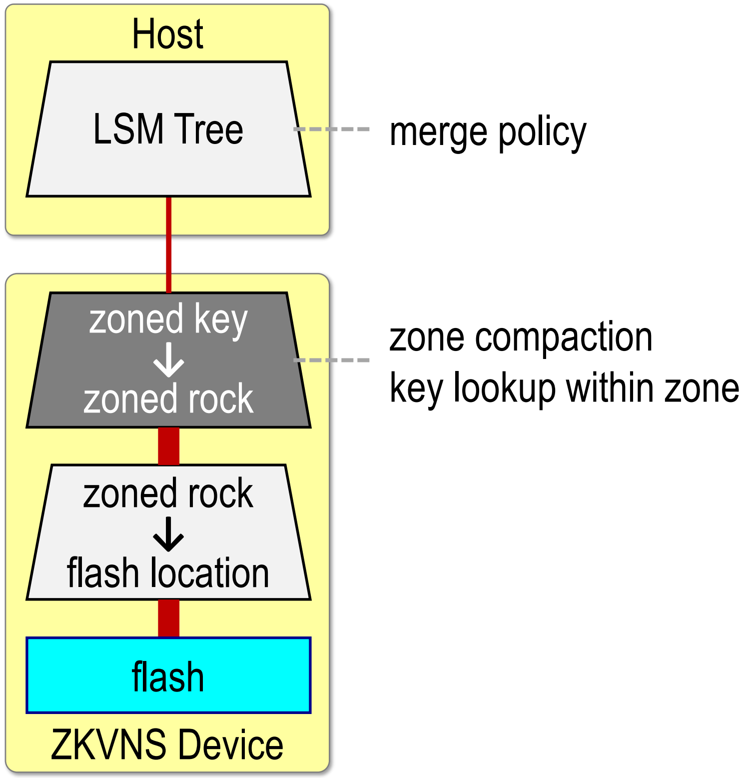
1. Direct Logging



2. Transparent Compression



3. Zoned KV Namespace (aka SSTables)



Conclusions

1. ZNS has the potential to become a dominant abstraction:
    a. Helps avoid an un-necessary translation.
    b. Supports systems with different data layouts.
2. ZNS can be extended to support rocks (ZRNS) with little cost:
    a. Specification: command set similar to blocks.
    b. Implementation: needs same (small) amount of internal NVRAM.
3. ZRNS provides significant benefits:
    a. Store small/variable size data efficiently: inodes, small files, compressed data.
    b. Append log records concurrently without explicit staging in NVRAM.
4. ZRNS enables further extensions:
    a. Zoned key-value records for offloading merging in LSM Trees.
    b. Other domain-specific formats and functions?

Please send questions/suggestions to umesh at alum.mit.edu.