SMORE: A Cold Data Object Store for SMR Drives

Peter Macko, Xiongzi Ge, John Haskins Jr.*, James Kelley, David Slik, Keith A. Smith, and Maxim G. Smith

Advanced Technology Group
NetApp, Inc.

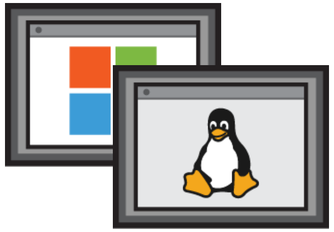
* Qualcomm (work performed while at NetApp)

Storage for large, cold files

Motivation

- Demand for storage large, cold files

Media Files



Virtual Machine Libraries

Backups

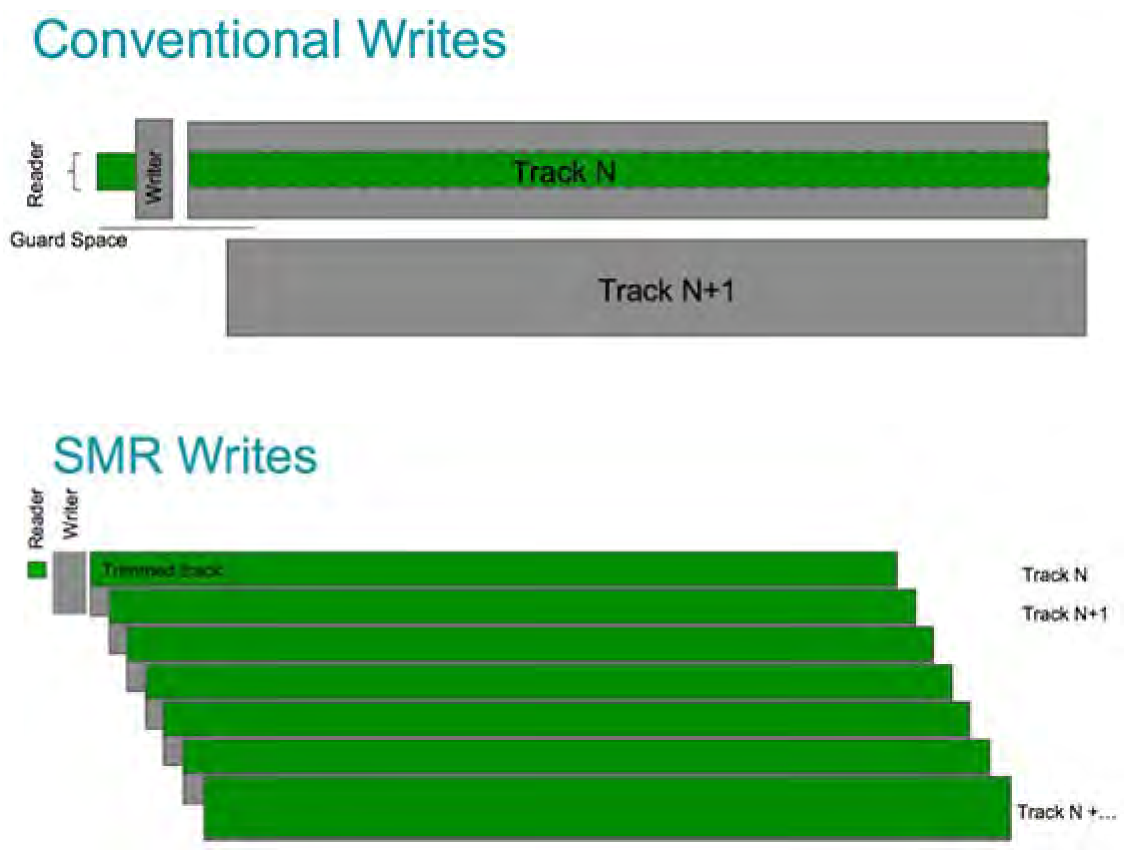
- High-capacity, low-cost storage
- Large sequential reads and writes

Good fit for SMR drives!

Media Files: https://commons.wikimedia.org/wiki/File:Video-film.svg

Shingled Magnetic Recording (SMR)

A quick review



- Conventional HDDs
  - Tracks do not overlap
  - A guard band lies between adjacent tracks
- SMR HDDs
  - Tracks partially overlap, leaving enough width for the read head (smaller than the write head)
  - No overwrite in place
- SMR in practice
  - A zone can only be written sequentially or erased (no random writes)
  - Supports random reads
  - Split into 256 MB sequential-only zones

http://www.seagate.com/tech-insights/breaking-areal-density-barriers-with-seagate-smr-master-ti/

SMORE (SMR Object Repository)

Introduction

- Object store for an array of SMR drives
  - Supports PUT, GET, and DELETE
- Optimized for large, cold objects
- Assume that frequently accessed or updated data are cached at higher layers of the storage stack
- Requirements for SMORE:
  - Ingest & read at disk speed
  - Low write amplification
  - Reliable storage

S'more: https://commons.wikimedia.org/wiki/File:Smores-Microwave.jpg

SMORE: A Cold Data Object Store for SMR Drives

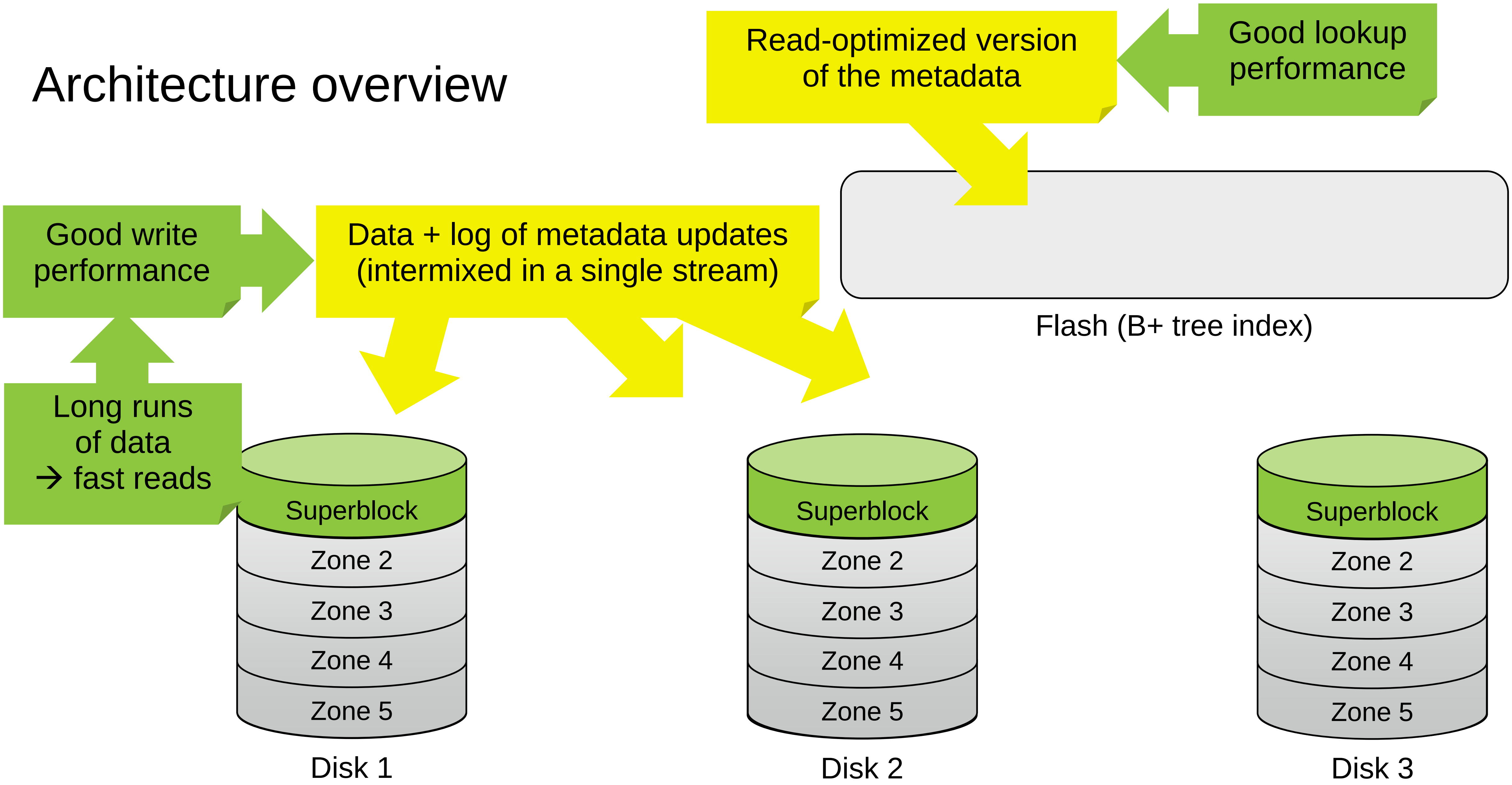
Agenda

1) Introduction
2) SMORE Architecture
3) Results
4) Conclusion

SMORE Architecture

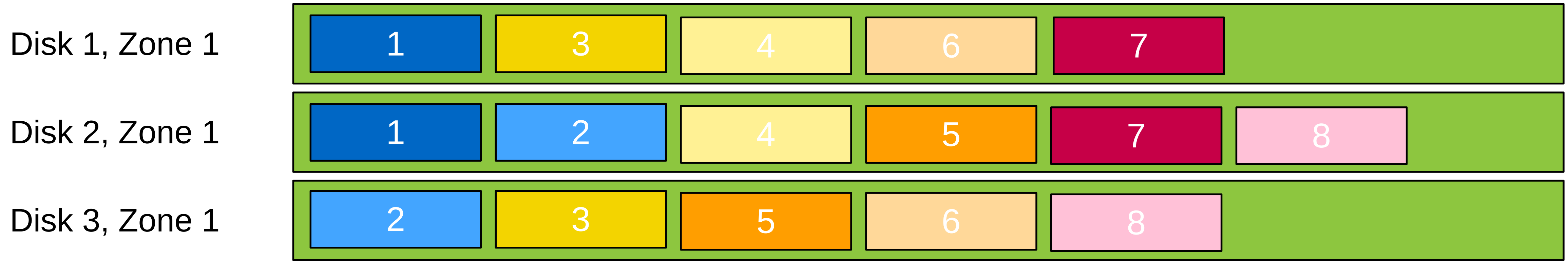
High-level overview

Architecture overview



Architecture overview

Superblock zones

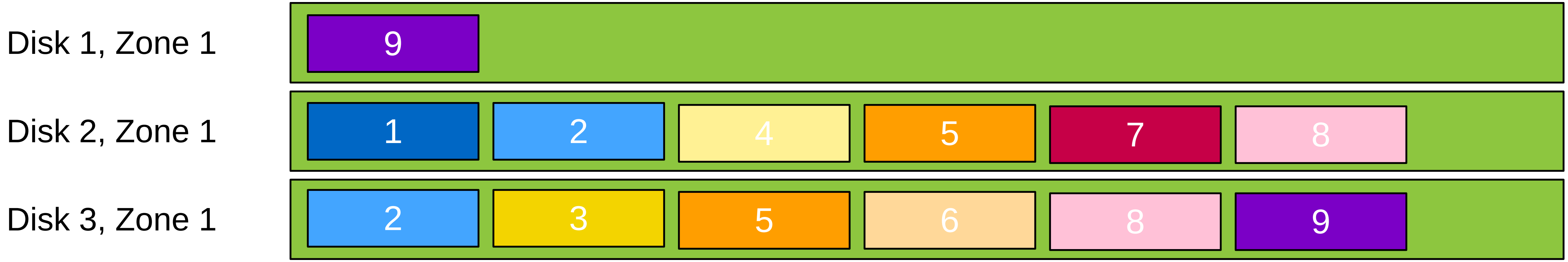


- Write the superblock in a log-structured manner
- Replicate the superblock to 2 or more drives
- Erase a zone when full, always preserving the replication count of the latest superblock

NetApp

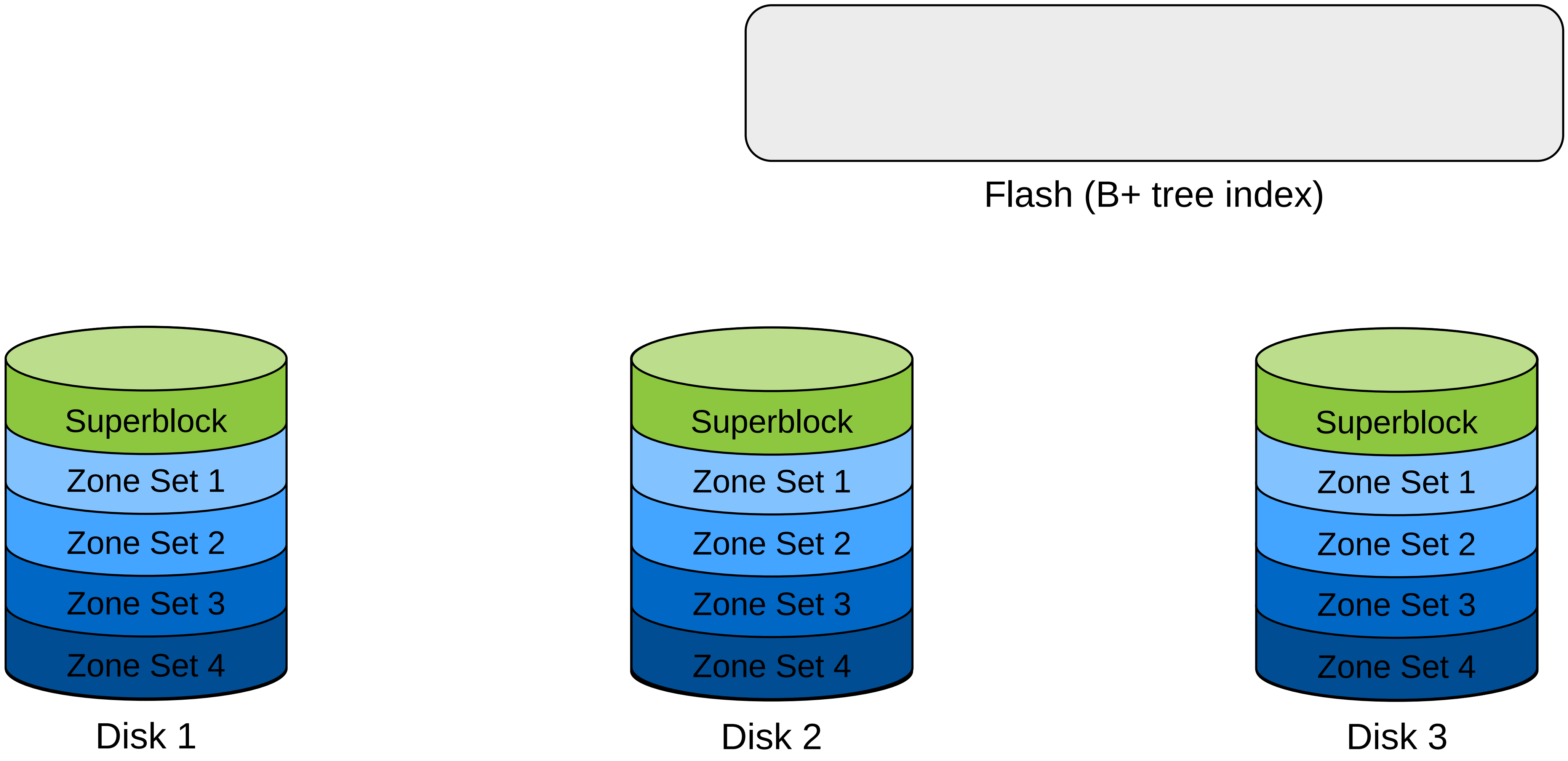
Architecture overview

Superblock zones



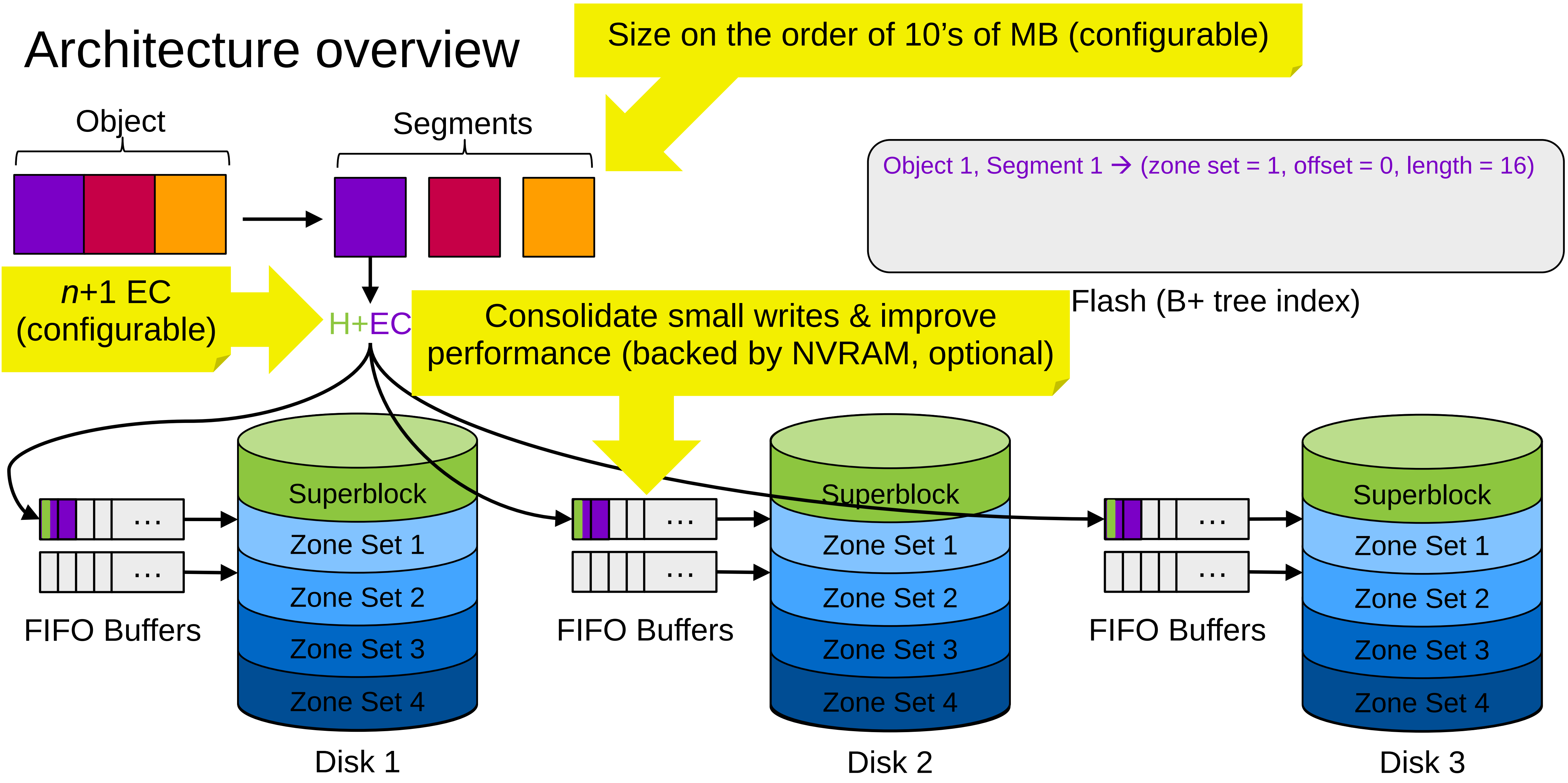
- Write the superblock in a log-structured manner
- Replicate the superblock to 2 or more drives
- Erase a zone when full, always preserving the replication count of the latest superblock

Architecture overview

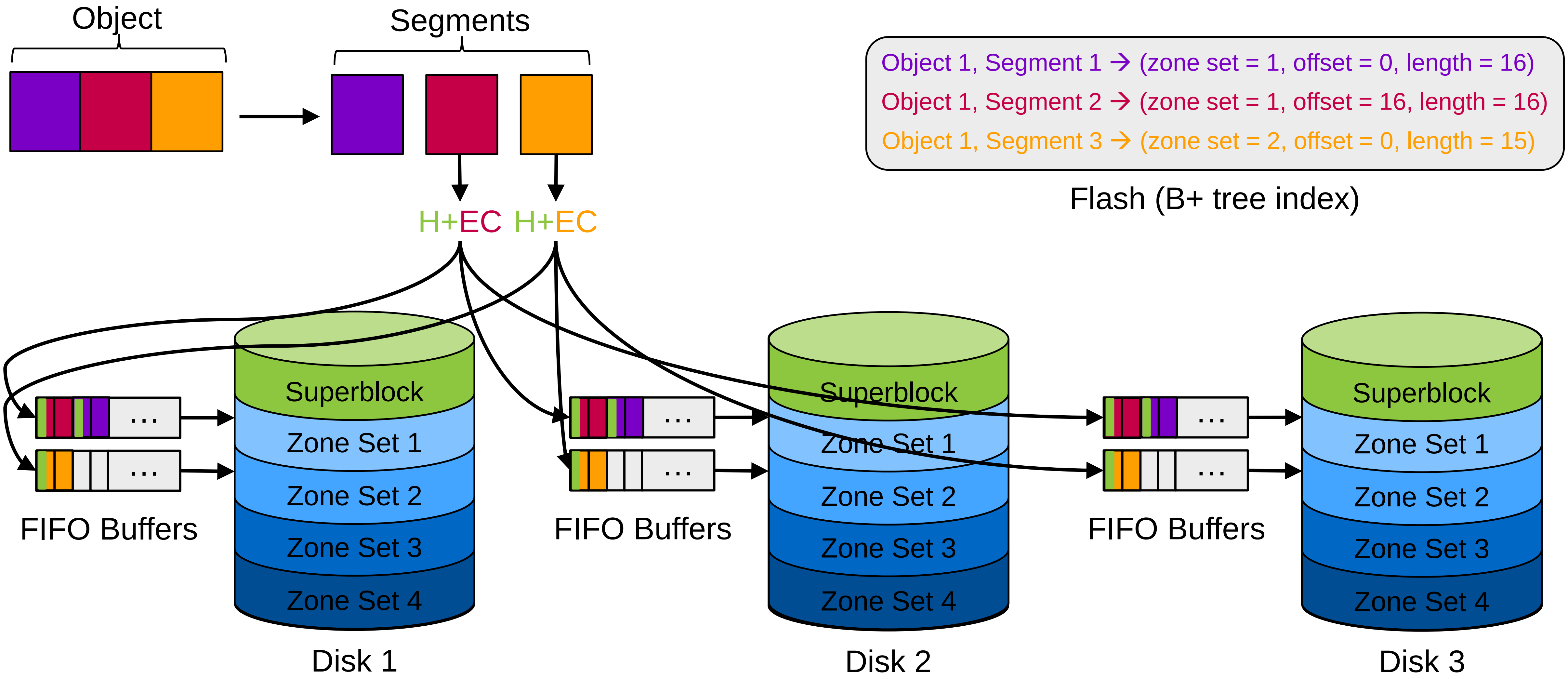


NetApp

Architecture overview



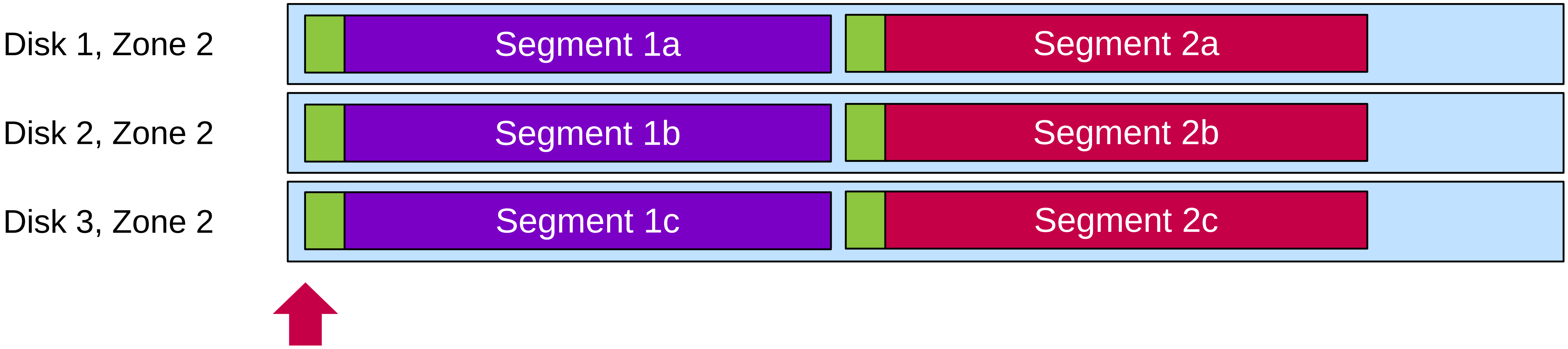
Architecture overview



NetApp

Architecture overview

Zone sets



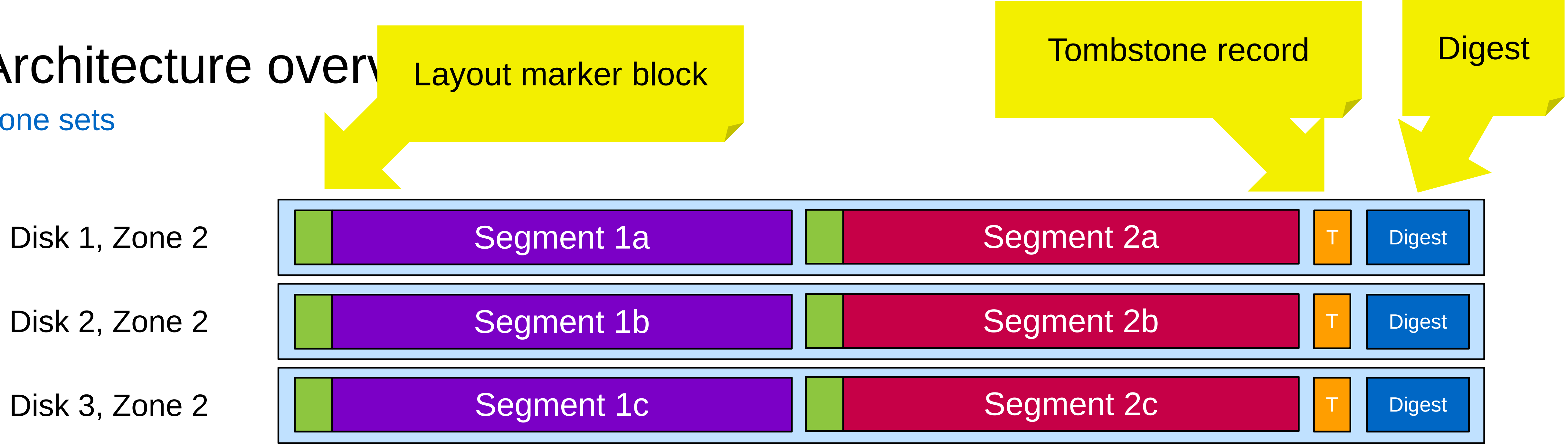
- Zone set = any *n* zones from *n different* disks
  - Written and erased together
  - Data is erasure-coded across the zones
- Write pointers always advance together

Location of a segment of data:
(zone set ID, offset)

- Decreases the size of the index
- Easily move contents of a zone

Architecture overv

Zone sets

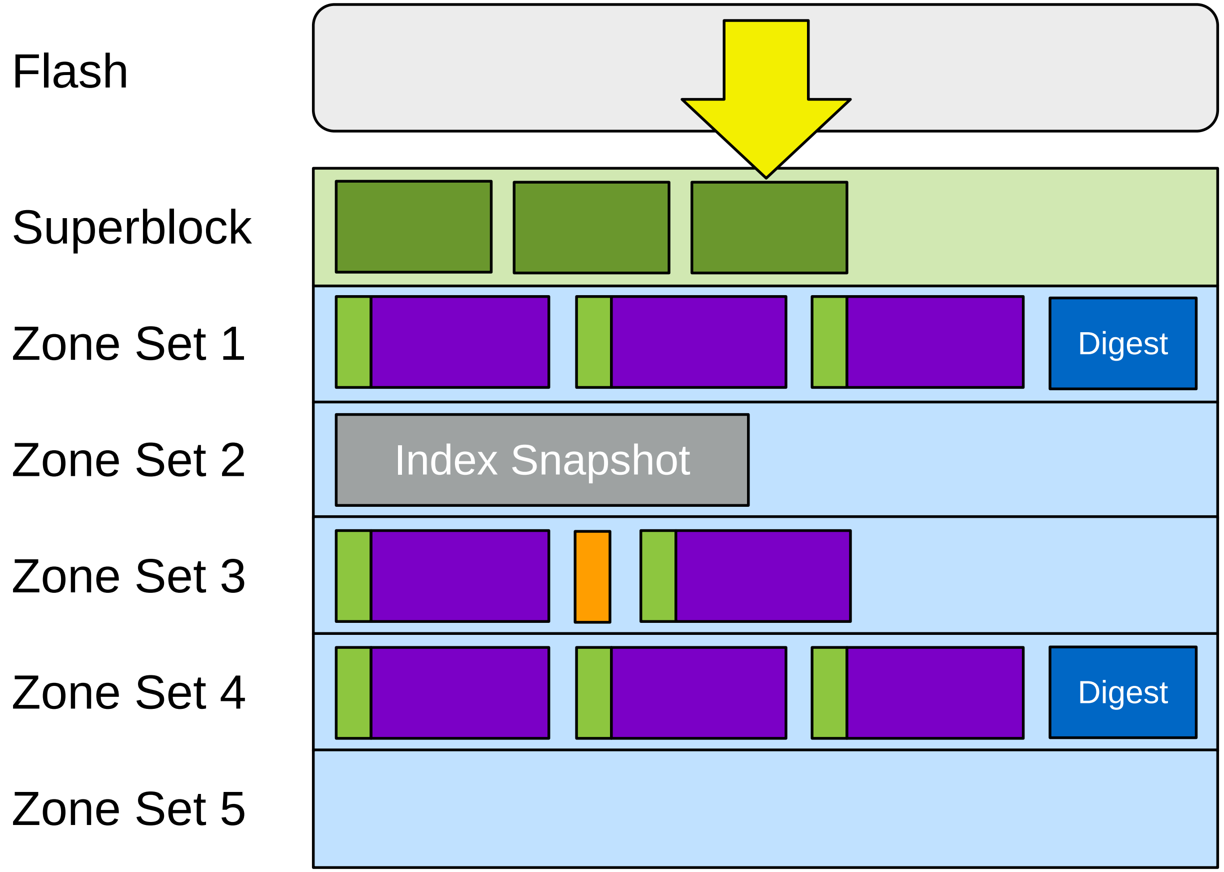


- Each segment is prefixed with a layout marker block
  - Uniquely identifies the segment and acts as a redo log entry for the index
  - Delete objects by writing tombstone records, which are special layout marker blocks
- Write a digest, a summary of all layout marker blocks, at the end of each full zone set (helps with recovery)

NetApp

Index recovery

A recovery-oriented design

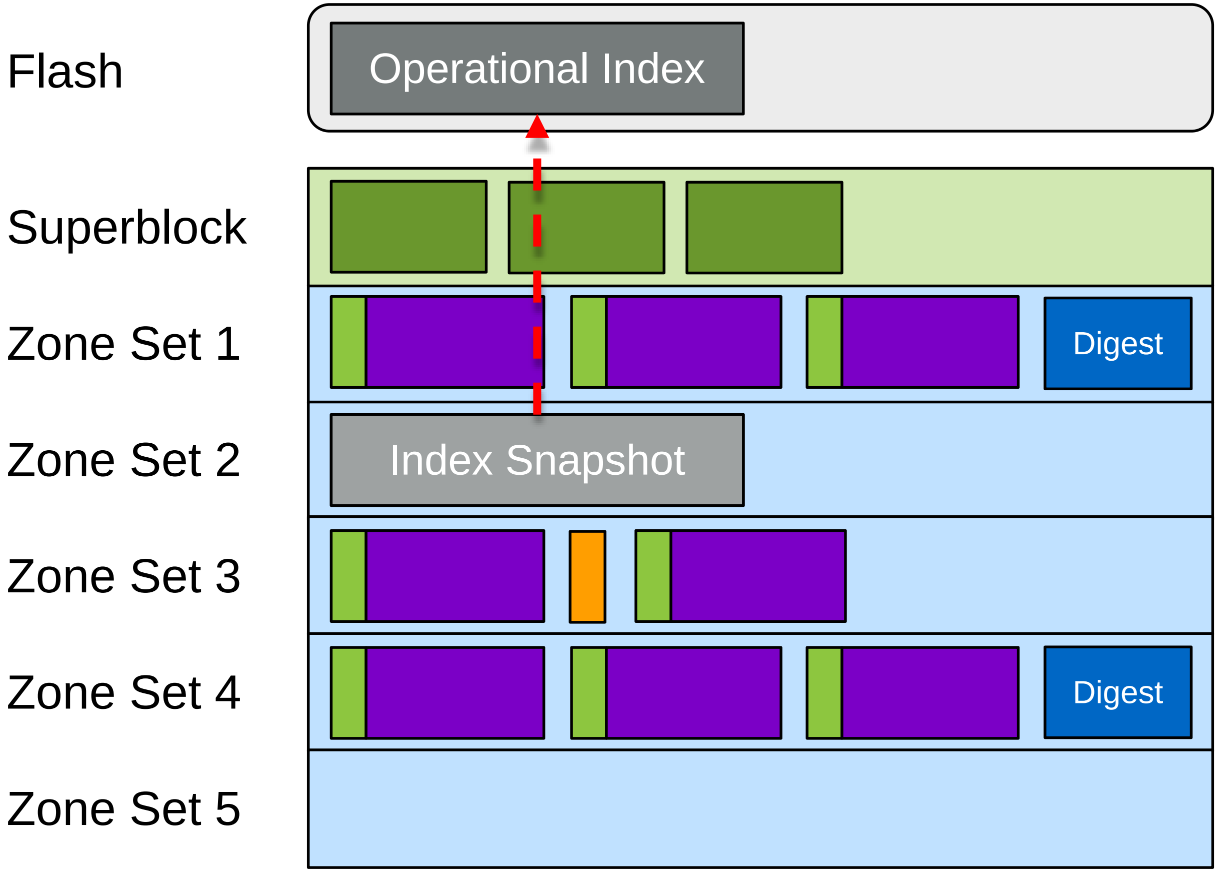


1. Read the most recent superblock
2. Restore the index from the latest consistent snapshot
3. Identify which zone sets could have changed since the snapshot was taken
4. Read and replay the digests from full zone sets and individual layout marker blocks from incomplete zone sets
5. Optionally, take an index snapshot

NetApp

Index recovery

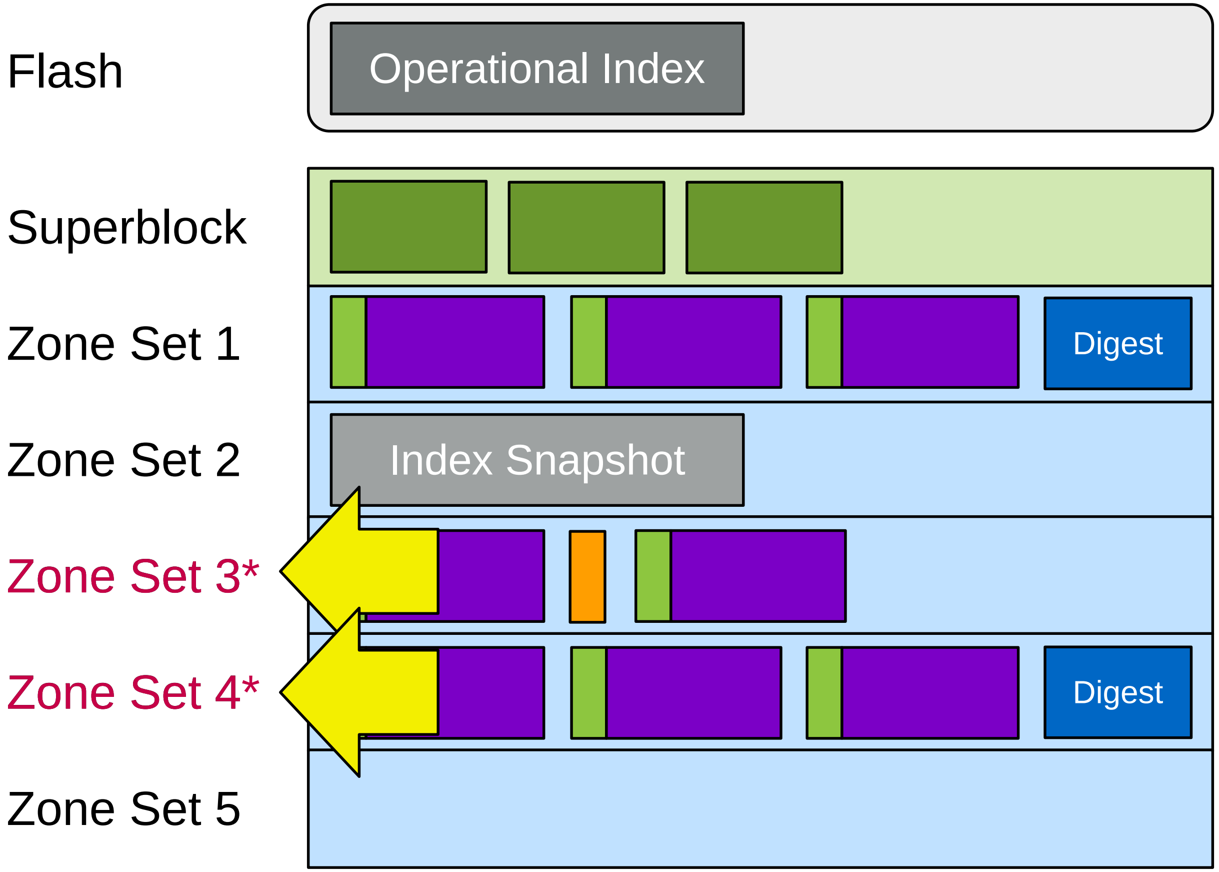
A recovery-oriented design



1. Read the most recent superblock
2. Restore the index from the latest consistent snapshot
3. Identify which zone sets could have changed since the snapshot was taken
4. Read and replay the digests from full zone sets and individual layout marker blocks from incomplete zone sets
5. Optionally, take an index snapshot

Index recovery

A recovery-oriented design

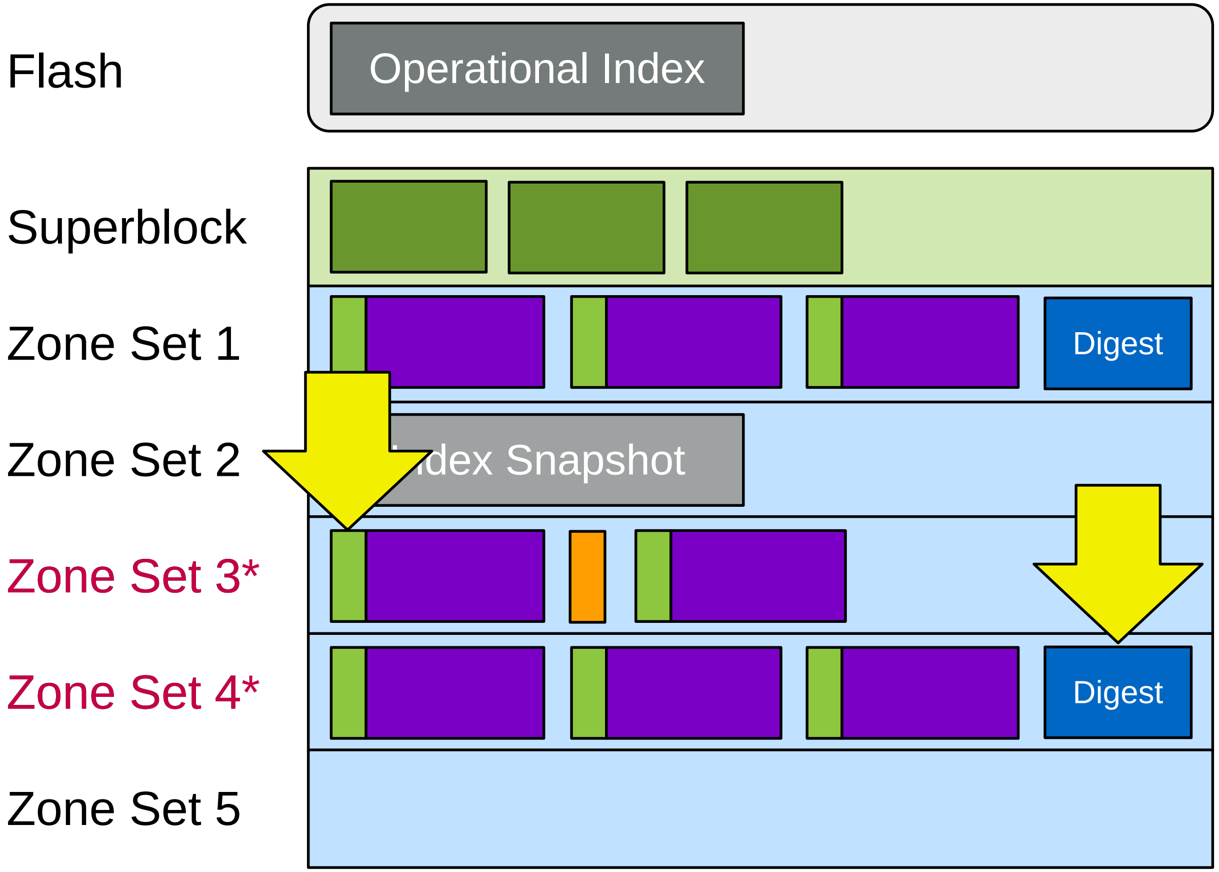


1. Read the most recent superblock
2. Restore the index from the latest consistent snapshot
3. Identify which zone sets could have changed since the snapshot was taken
4. Read and replay the digests from full zone sets and individual layout marker blocks from incomplete zone sets
5. Optionally, take an index snapshot

NetApp

Index recovery

A recovery-oriented design

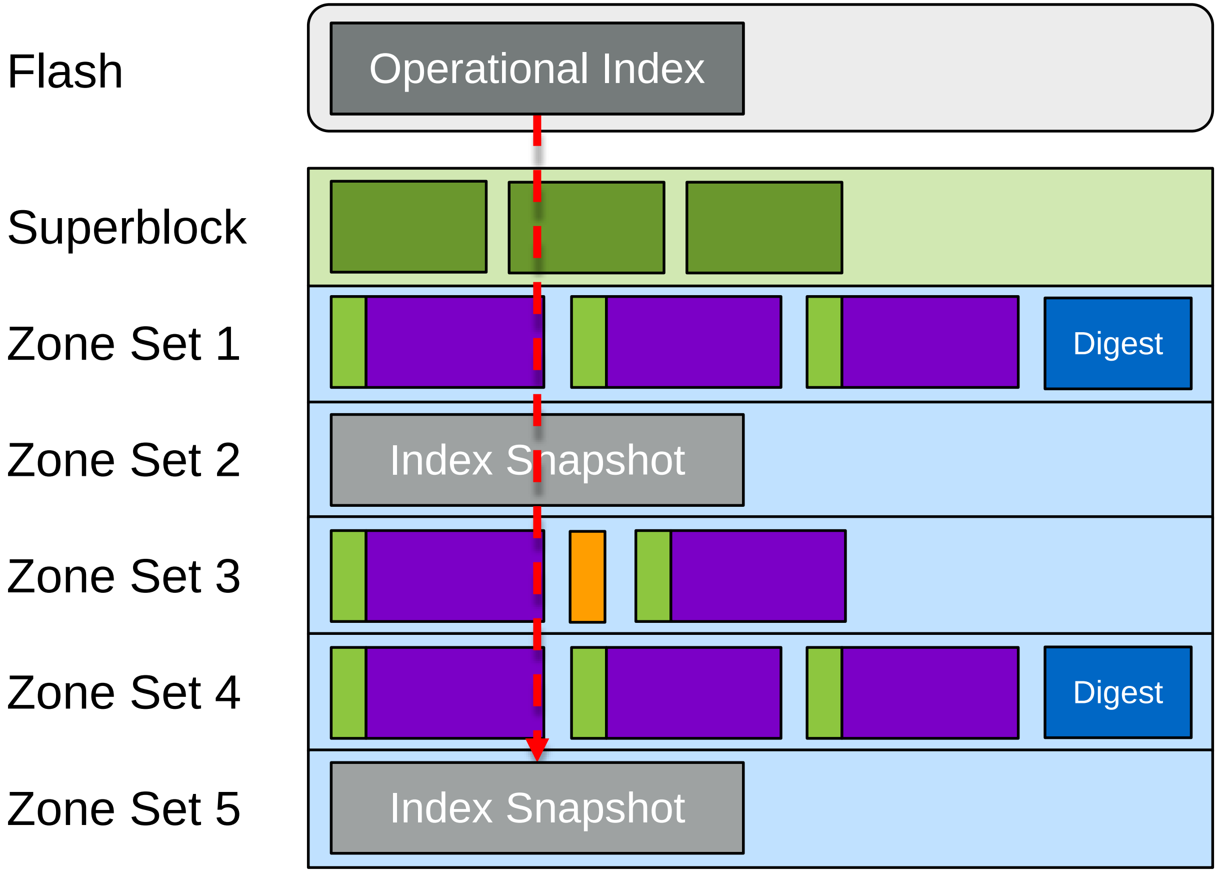


1. Read the most recent superblock
2. Restore the index from the latest consistent snapshot
3. Identify which zone sets could have changed since the snapshot was taken
4. Read and replay the digests from full zone sets and individual layout marker blocks from incomplete zone sets
5. Optionally, take an index snapshot

NetApp

Index recovery

A recovery-oriented design

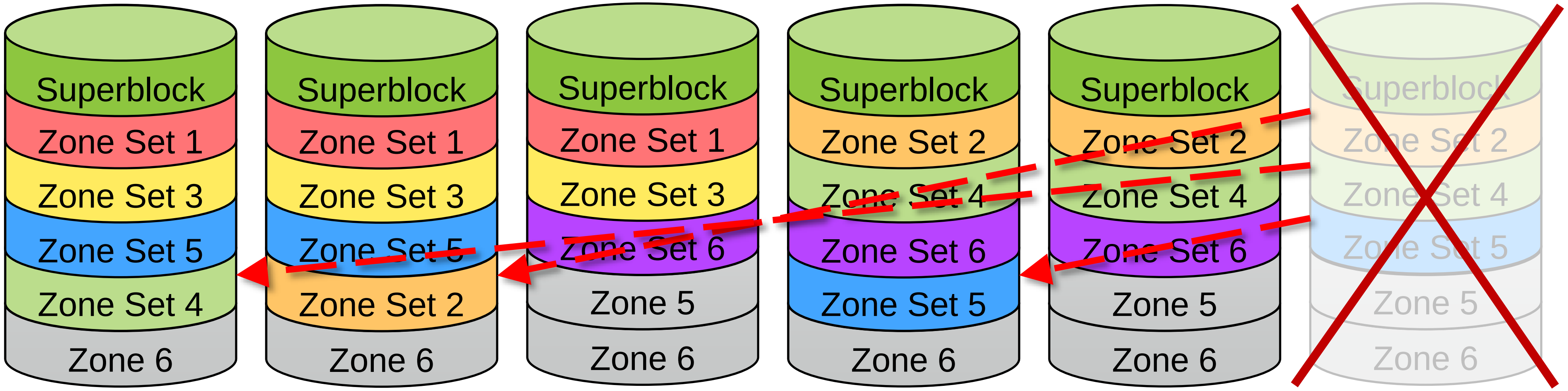


1. Read the most recent superblock
2. Restore the index from the latest consistent snapshot
3. Identify which zone sets could have changed since the snapshot was taken
4. Read and replay the digests from full zone sets and individual layout marker blocks from incomplete zone sets
5. Optionally, take an index snapshot

NetApp

Zone sets and recovery

The general case

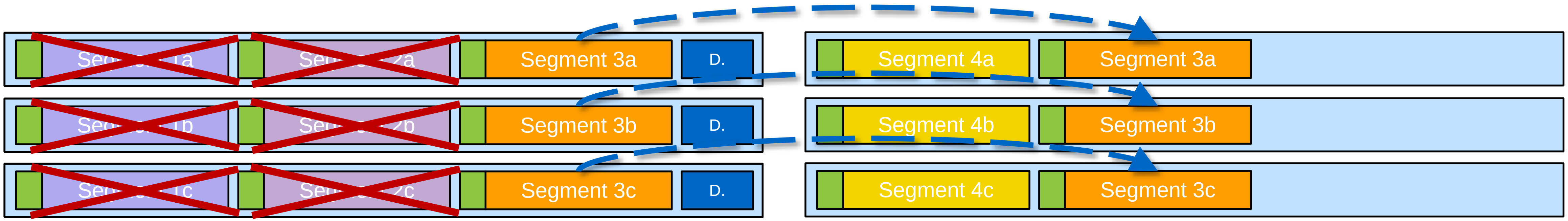


- The general case: *Any* zones from different drives can form one zone set
- Reconstruct the zones from the offline disk into the free zones on other drives
- We can do this without having to update the index!

NetApp

Garbage collection

The dead space algorithm



- Reclaim space using garbage collection
- Use the “dead space” algorithm:
  1. Maintain an accurate measure of dead space for each zone set
  2. Choose the zone set with most dead space
  3. Relocate live data to the currently open zone set
  4. Erase the zone set
- Works reasonably well for cold data (no need to maintain hot-cold separation)

Results

SMORE

Experimental setup (1/2)

Hardware Platform

- 32-core Intel Xeon 2.90 GHz, 128 GB RAM
  - The experiments used only 1.5 GB RAM
- 6x HGST Ultrastar Archive Ha10, 10 TB each
  - Used only every 60^{th} zone to reduce the capacity while preserving full seek profile
  - 5+1 parity within each zone set
  - Total system capacity after format: 766 GB
  - Confirmed that the results are representative by running selected workloads on a full 50 TB system

NetApp

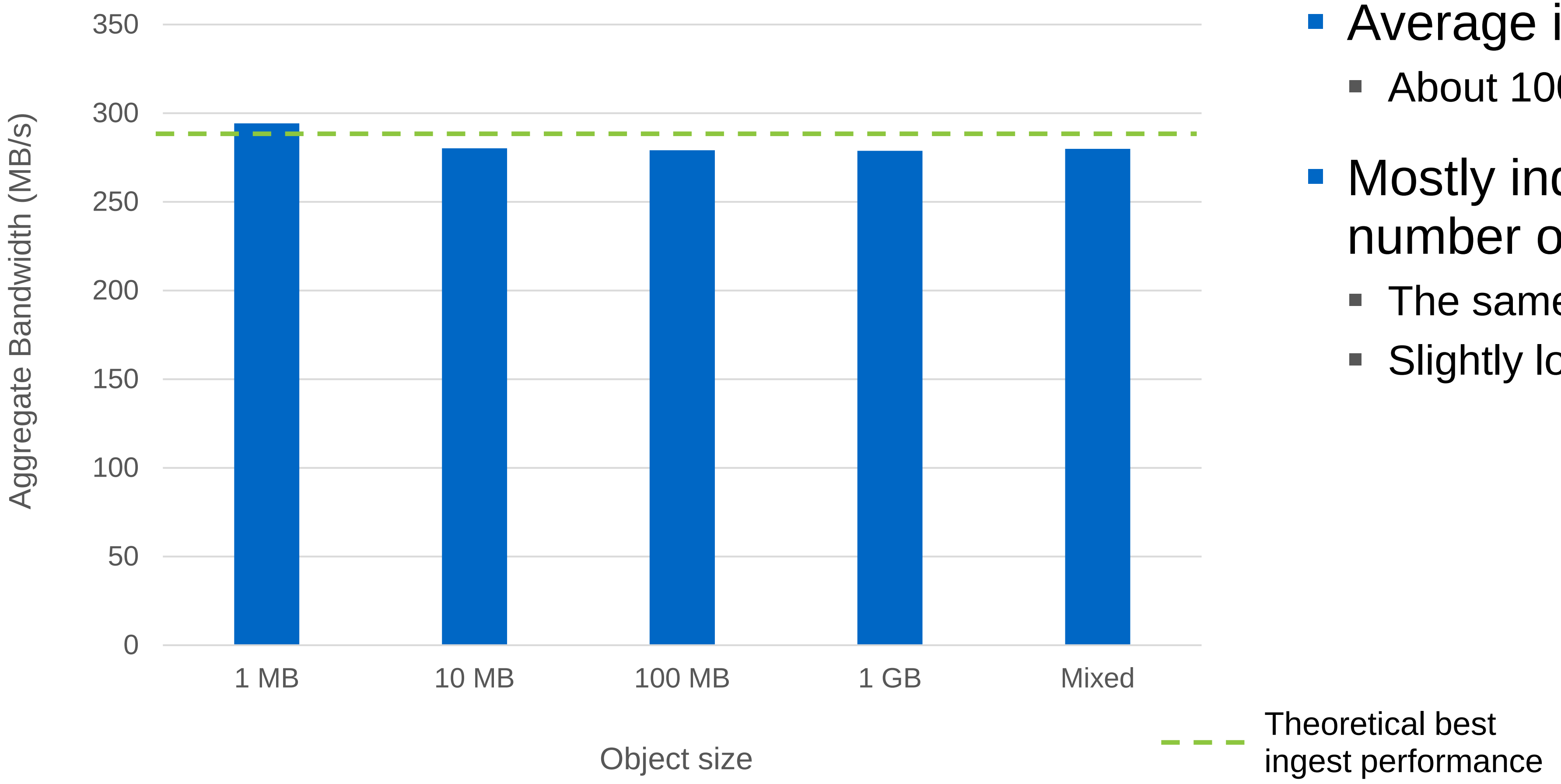
Experimental setup (2/2)

Software & Workload

- SMORE
  - Implemented as a C++ library
  - ~19k LOC
  - Custom I/O stack built on top of libzbc
- Workload generator
  - Synthetic workload generator parameterized by object size, target utilization, read-write ratio, etc.
  - Random object deletion, which produces the worst GC behavior
  - Mixed object sizes follow the distribution from the European Center for Medium-Range Weather Forecasts
- Unless stated otherwise:
  - 80% target utilization (amount of live data as a percentage of storage capacity)
  - 6 clients

Initial ingest

Per object size

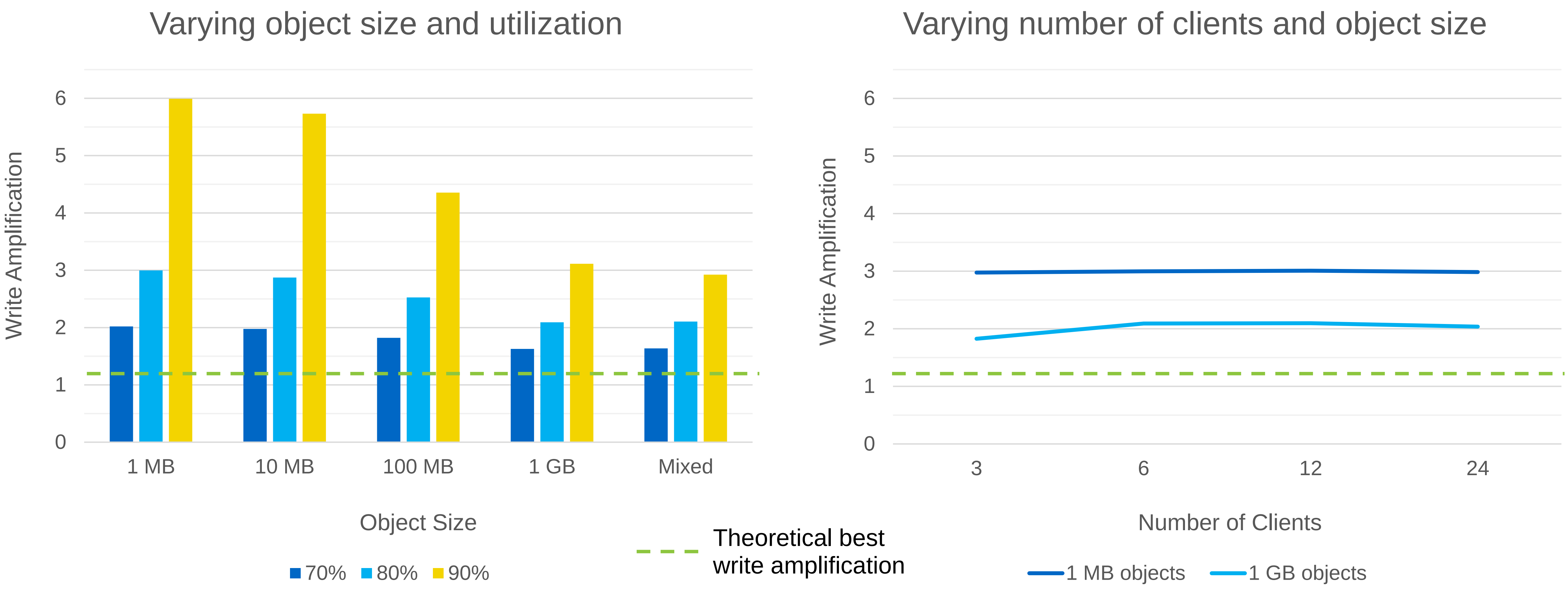


- Average ingest rate: 282 MB/s
  - About 100% of max disk bandwidth
- Mostly independent of the number of clients
  - The same results for 3-12 clients
  - Slightly lower rate for 24 clients

NetApp

Steady-state write amplification

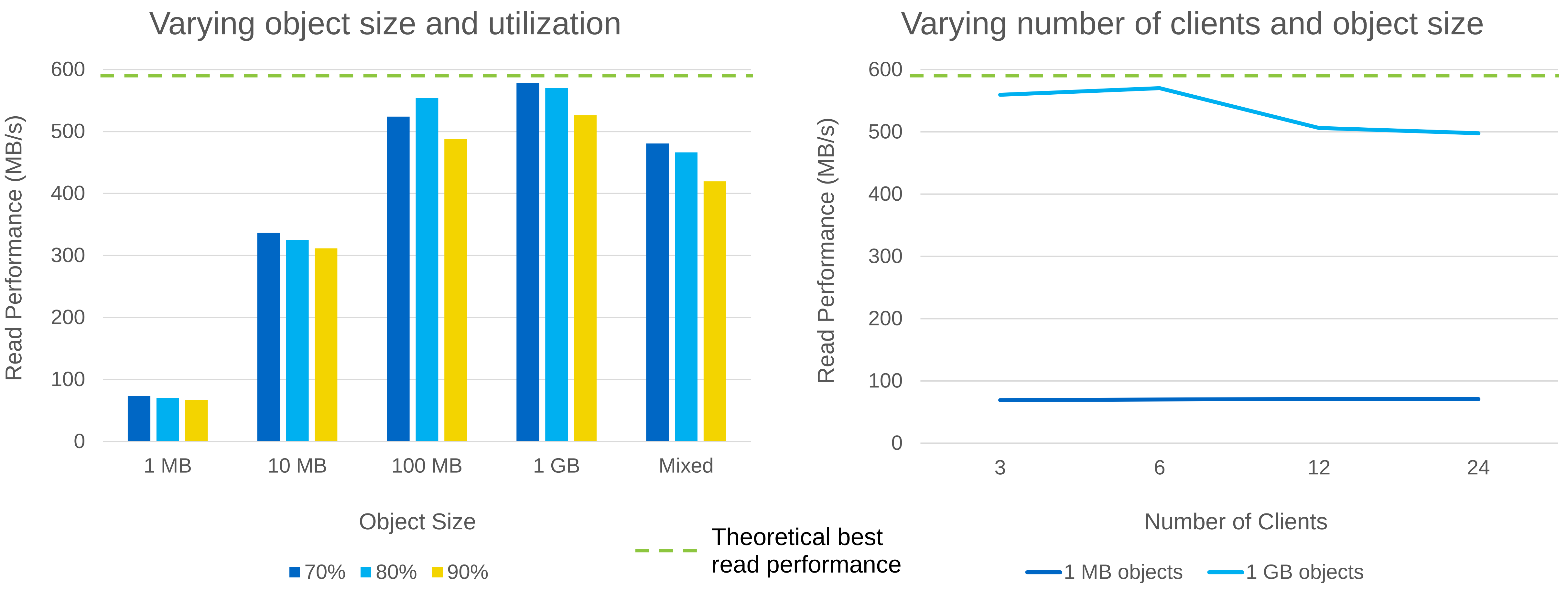
For write-only workloads, overwriting random objects (theoretical best: 1.20)



NetApp

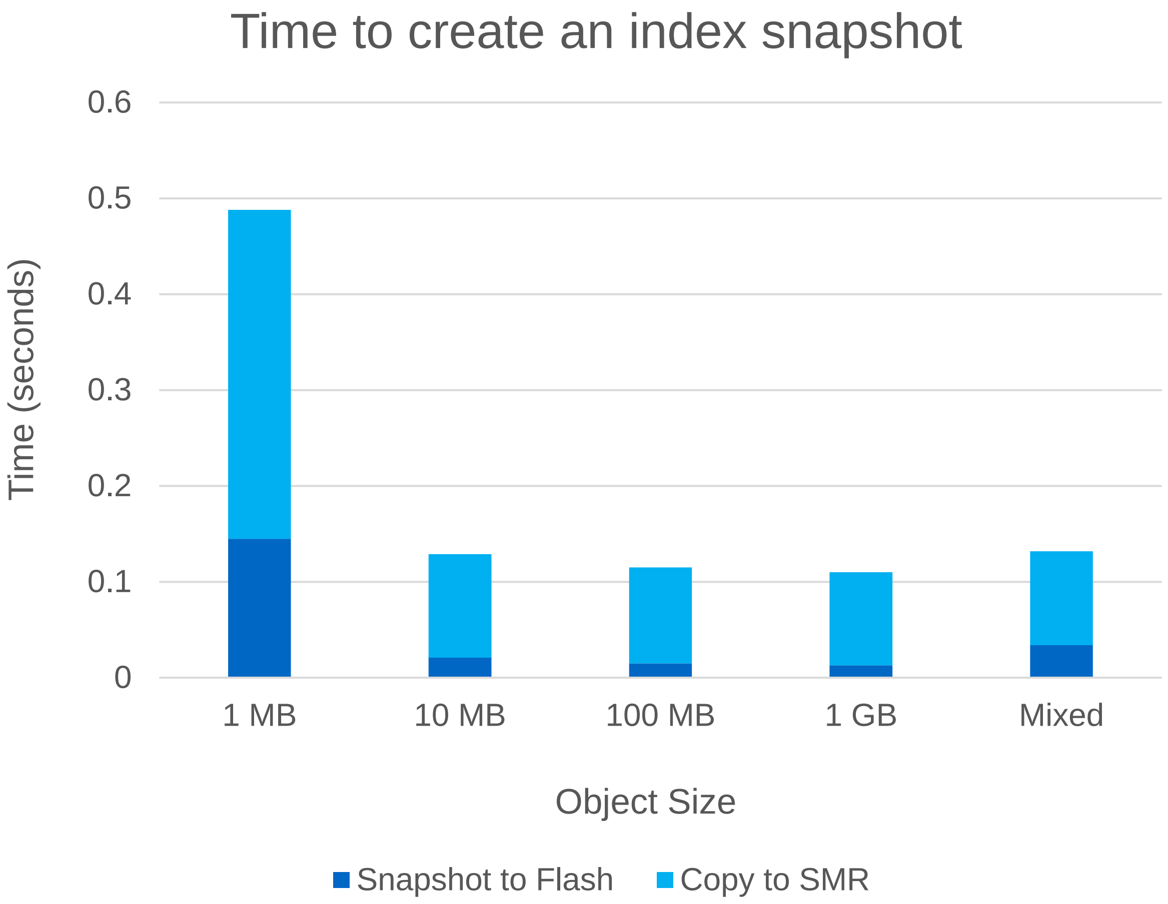
Steady-state read performance

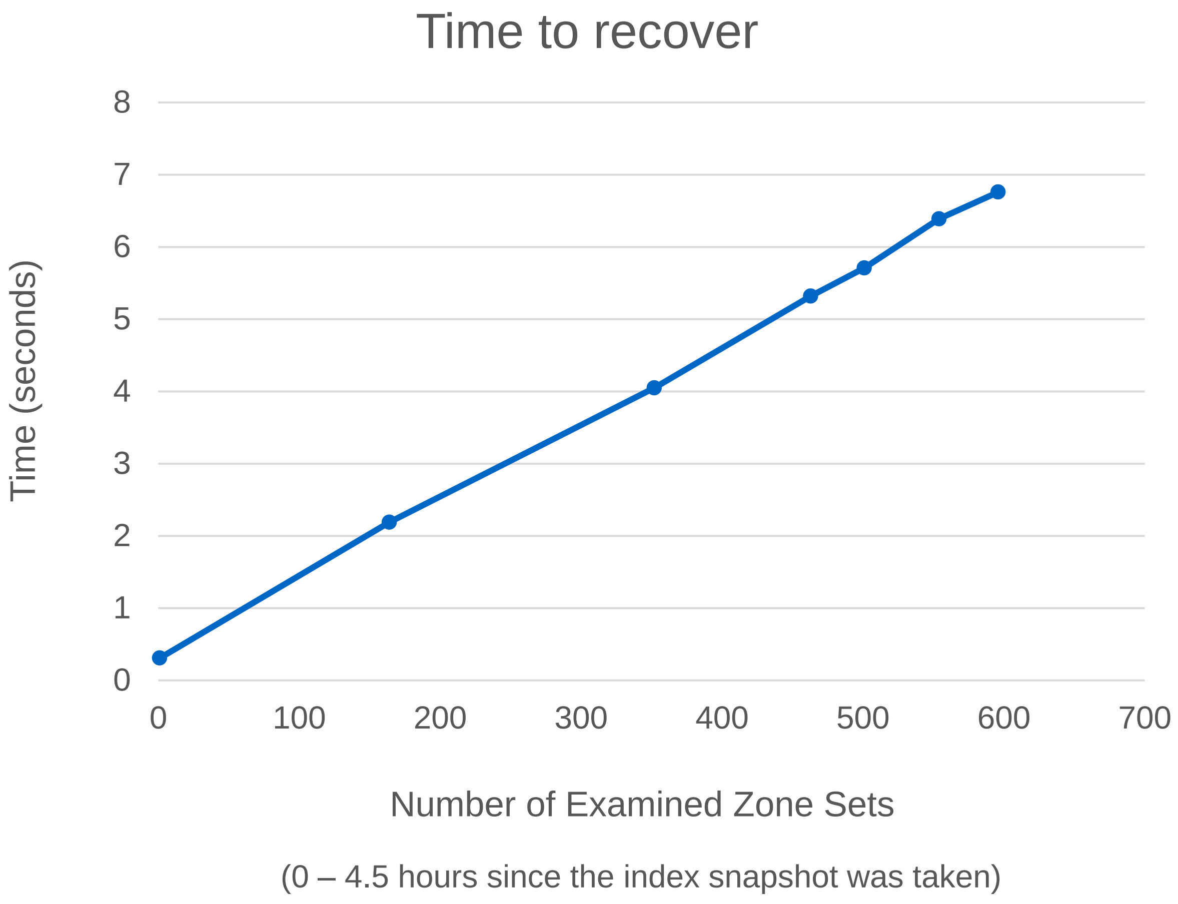
For read-only workloads, reading random objects (theoretical best: 590 MB/s)



Index recovery

Time to create and to recover an index





Conclusion

SMORE

- Fast ingest
  - Large segments of data + log of metadata updates on SMR drives
- Fast streaming reads
  - Fast lookup using a read-optimized version of the metadata index on a flash
  - Large (almost) contiguous segments of data on SMR drives
- Efficient recovery
  - Efficient index rebuild through zone set digests
  - Data relocation supported by indirection through zone sets

S'more: https://commons.wikimedia.org/wiki/File:Smores-Microwave.jpg

Thank you

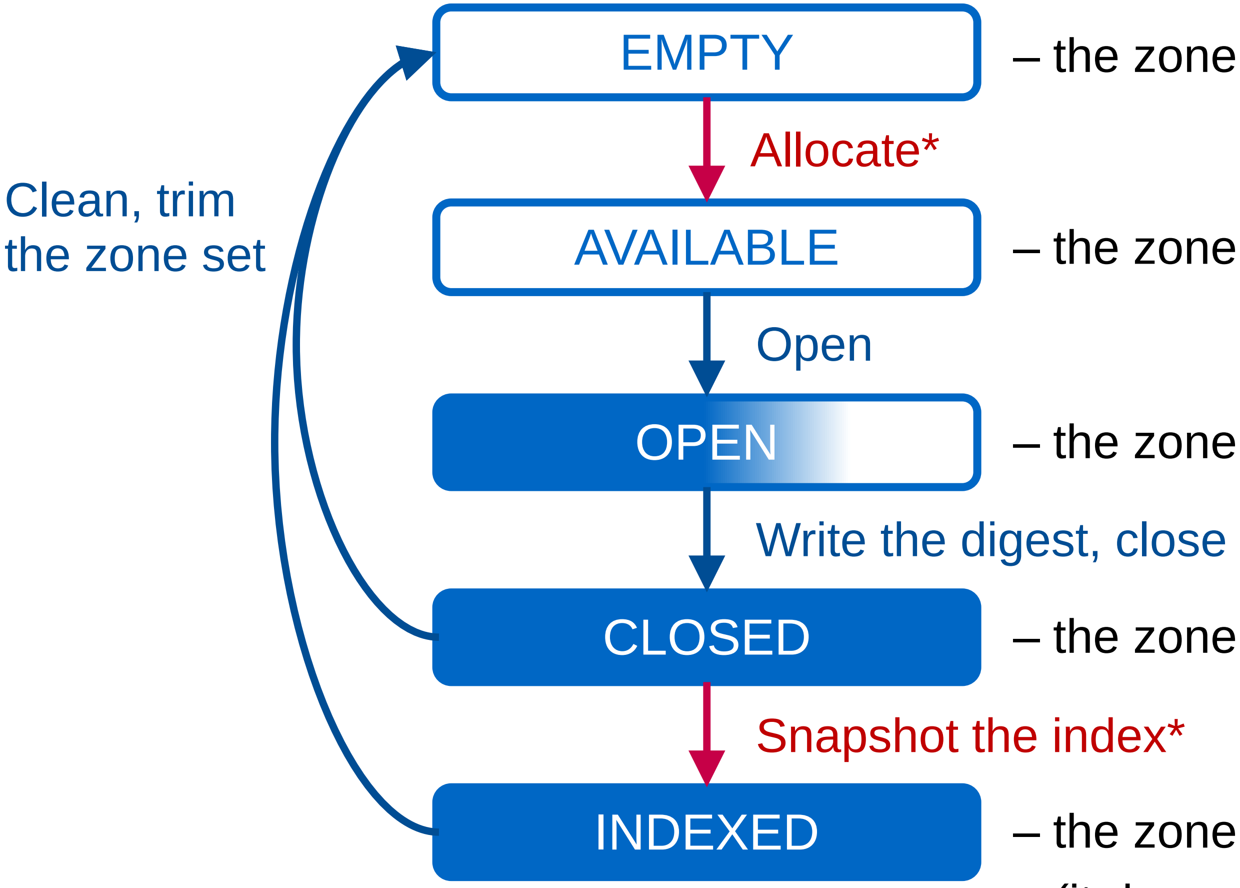
NetApp

Back up slides

Additional information

Zone sets

The state transition diagram



– the zone set is empty and not currently used by SMORE

– the zone set is empty, but is available for opening

– the zone set can receive writes

– the zone set is full (does not accept any more writes)

– the zone set was CLOSED when the index snapshot was taken (it does not need to be examined during index recovery)

* Involves writing the superblock