STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers
Virtual Conference
September 28-29, 2021
A SNIA Event

Beyond Zoned Namespace

ZNS_{NLOG} bridging the semantic gap

Presented by Chun Liu, Chief Architect, Futurewei Technologies

Landscape

Database/
DataStore

RocksDB LevelDB

Distributed
Storage

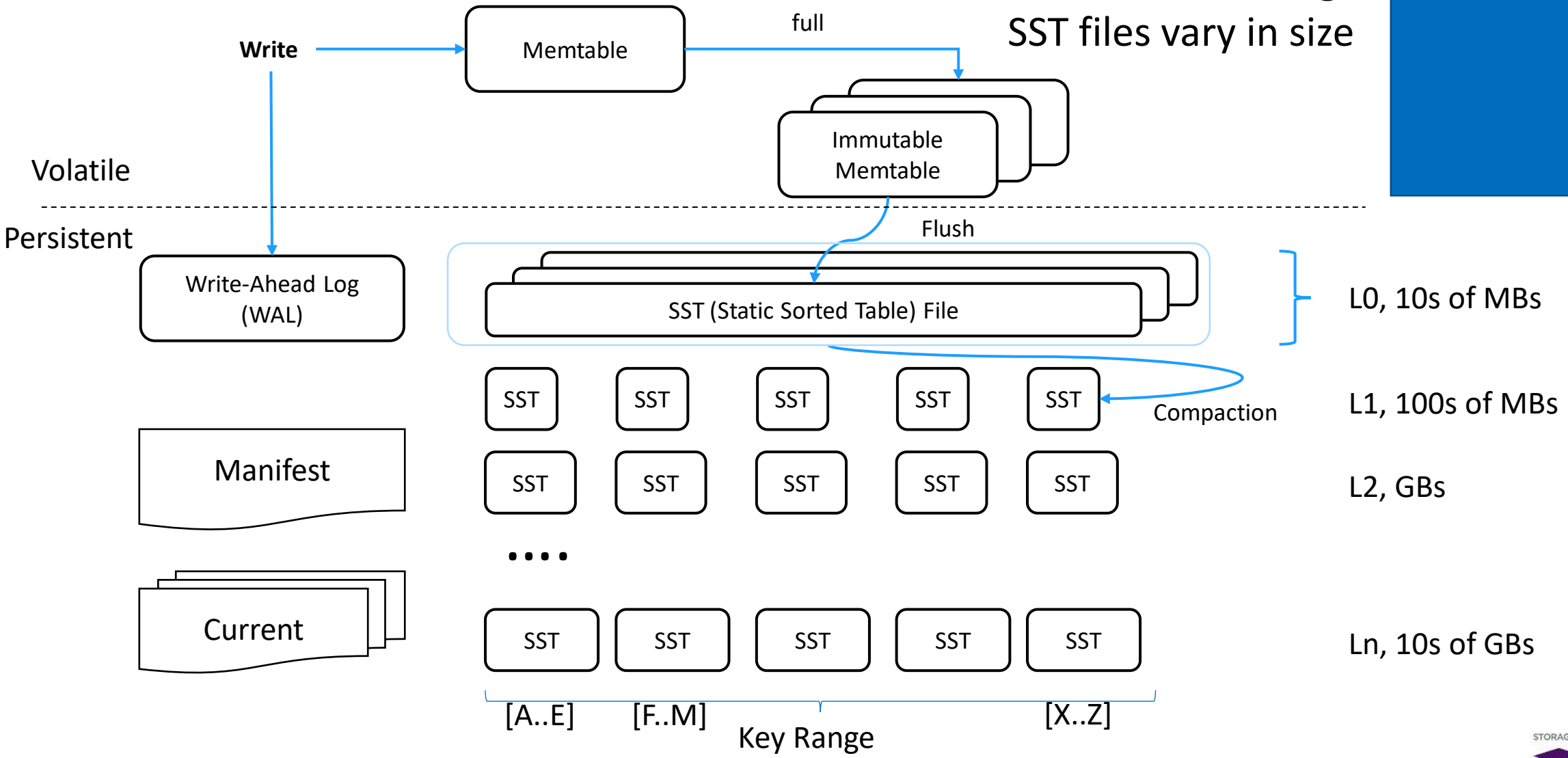
Apache
BookKeeper

Native File Systems Ext4 XFS BtrFS F2FS ZFS …

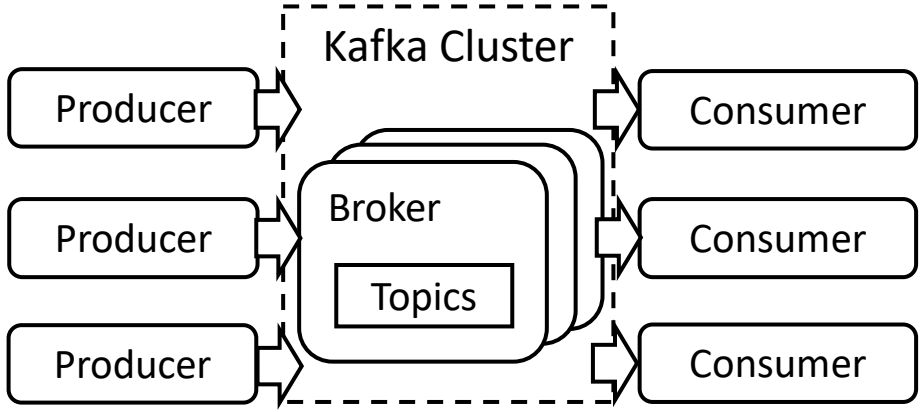
Block Devices NVMe (w/ ZNS) NVMeOF SATA/SAS

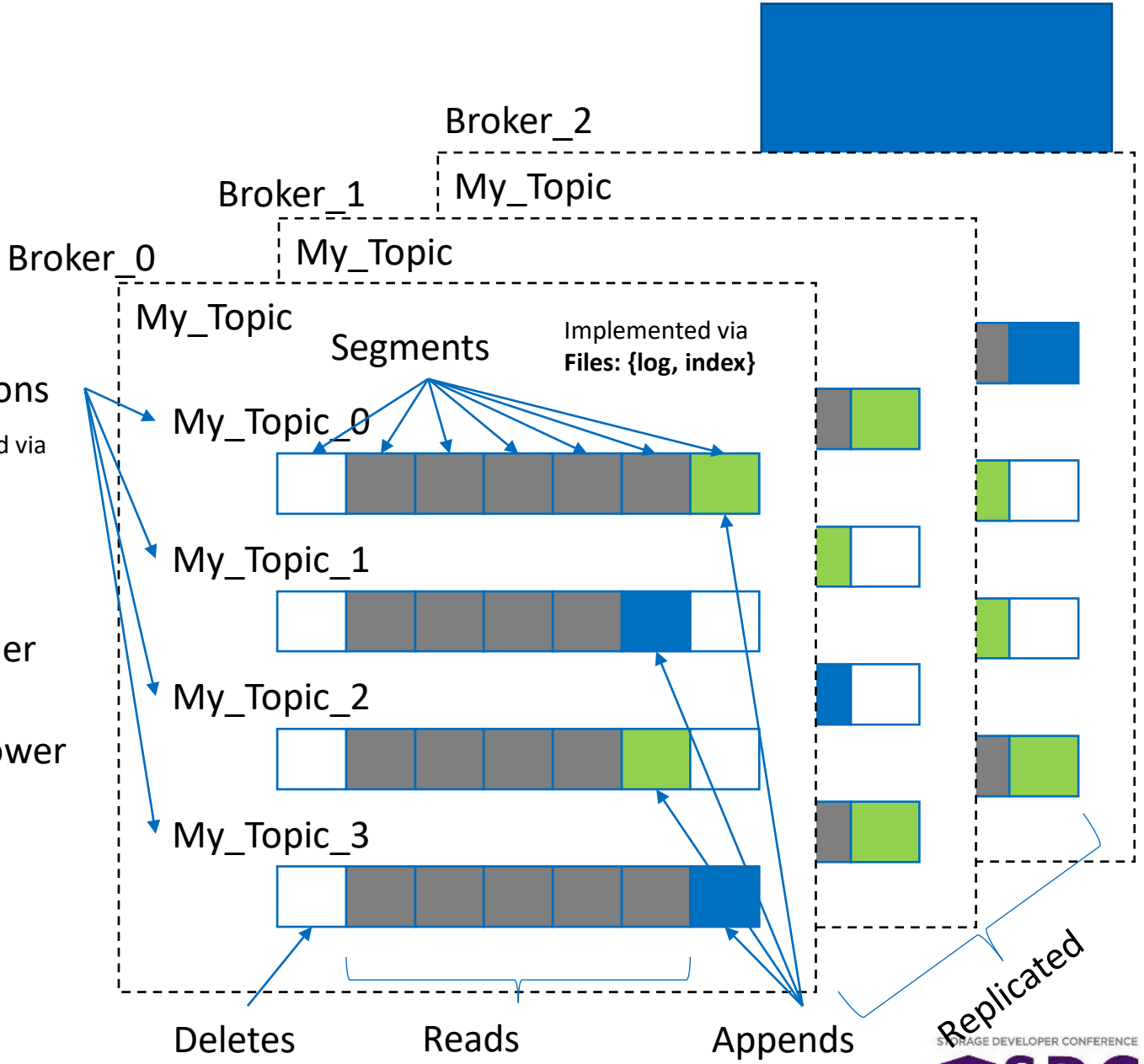
RocksDB builds on logs

WAL is **log**
All SST files are **logs**
SST files vary in size



Kafka also builds on logs

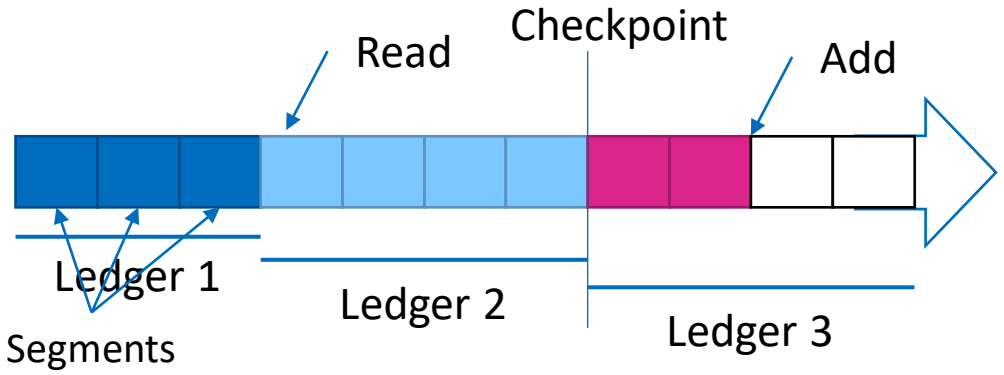


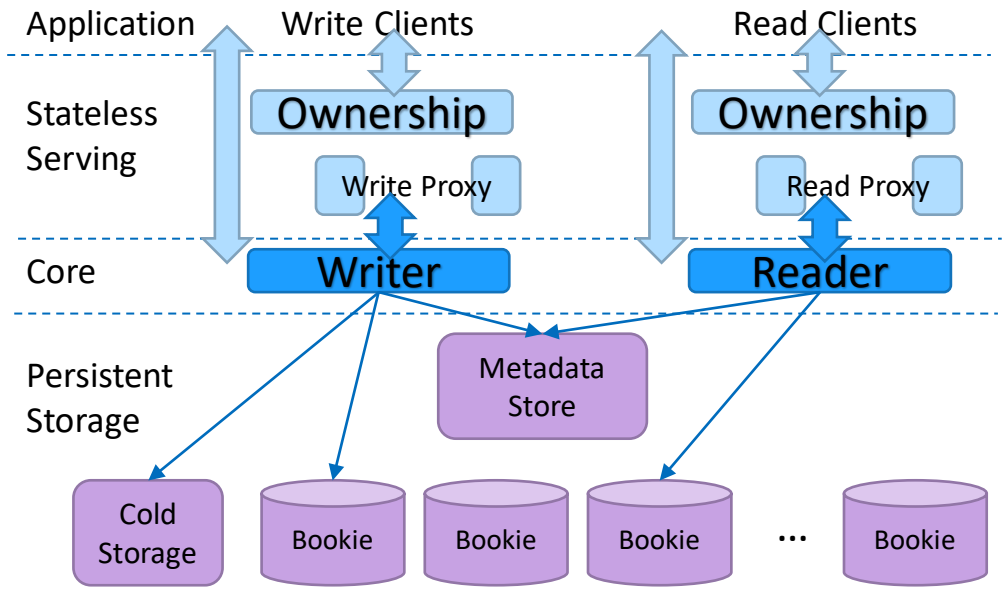


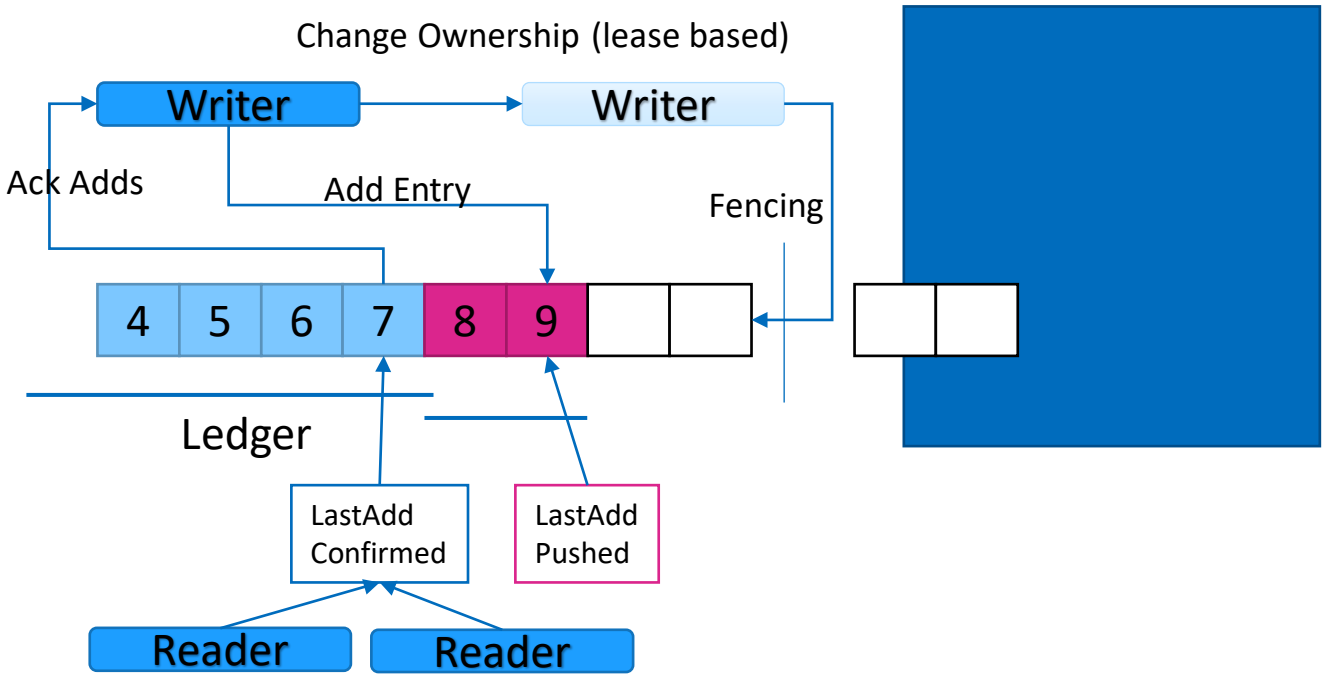
Partition is a **log**
Segments are **logs**
Partition append: create segment
Partition truncation: delete segment

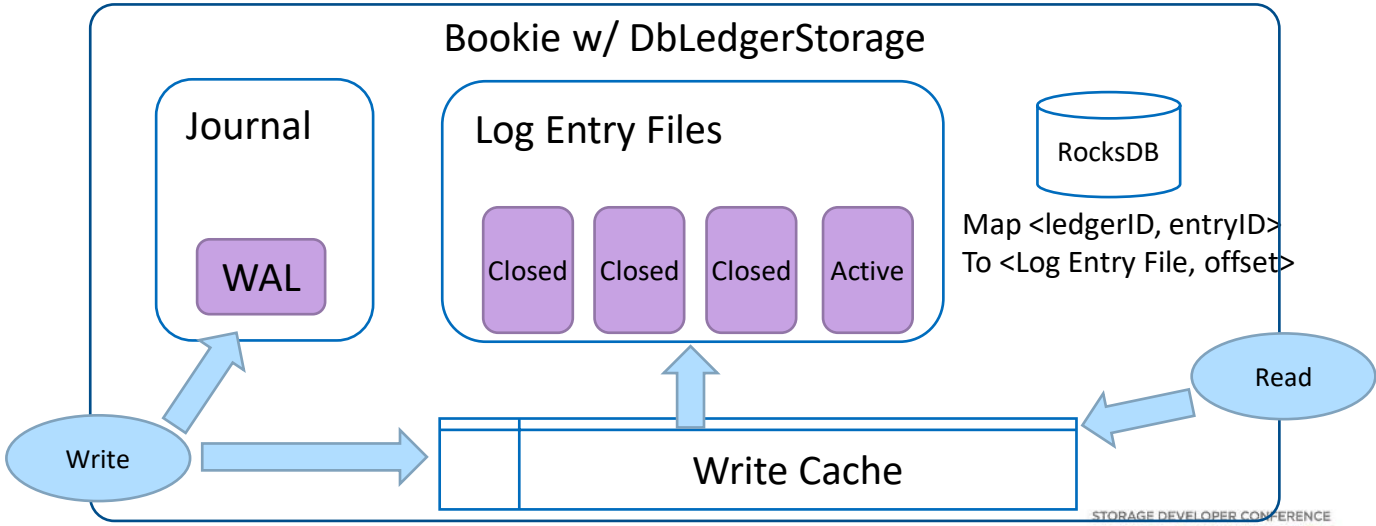
HDFS is write-once + append + truncate as well

BookKeeper



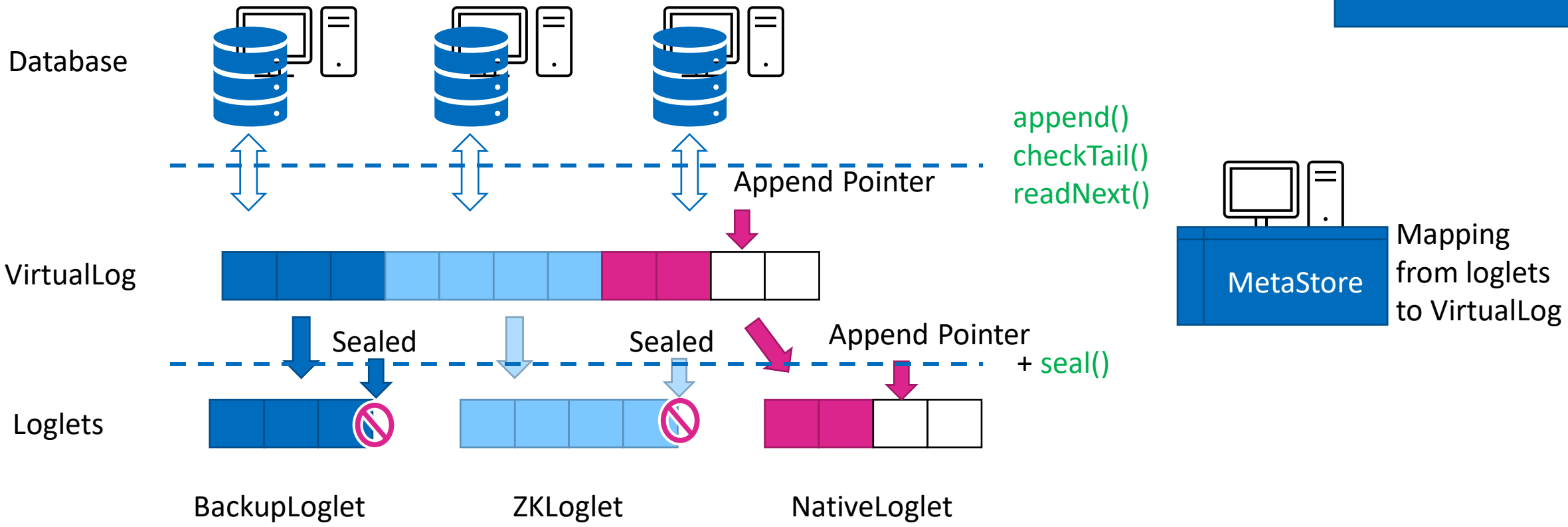






Facebook's Delos

- Delos: Replicated Storage for Control Plane
- VirtualLog enables online storage engine swapping.

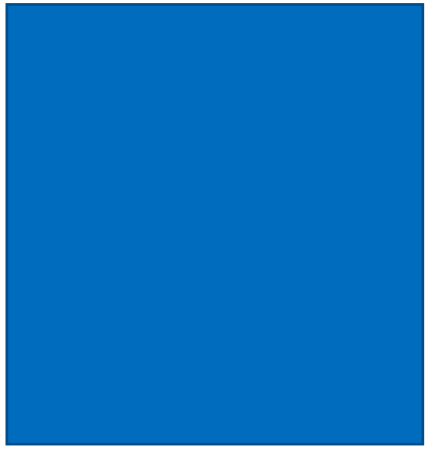


~~Log is (one kind of) data~~

DATA IS LOG!

Semantics

DATA IS LOG!



- Append to a (partitioned) logical log (Why?). **append()**
- Logical log consists of sequence of individual segments (size varies). **Stored in metadata store**
- Delete whole segment. **delete()**
- Find out where to write next, individually or collectively. **tail()**
- Fence/Seal, make segments read-only. **seal()**

Q: Can native file system **efficiently** support those?

ef·fi·cient

/əˈfiSHənt/

adjective

(especially of a system or machine) achieving maximum productivity with minimum wasted effort or expense.

Ceph's Lessons

File Systems Unfit as Distributed Storage Backends: Lessons from 10 Years of Ceph Evolution

Abutalib Aghayev
Carnegie Mellon University

Sage Weil
Red Hat, Inc.

Michael Kuchnik
Carnegie Mellon University

Mark Nelson
Red Hat, Inc.

Gregory R. Ganger
Carnegie Mellon University

George Amvrosiadis
Carnegie Mellon University

Abstract

For a decade, the Ceph distributed file system followed the conventional wisdom of building its storage backend on top

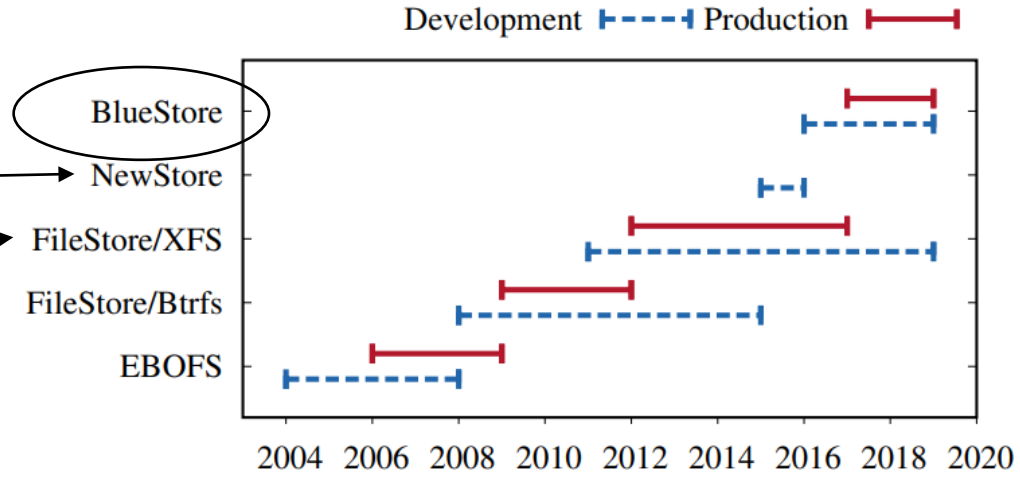
1 Introduction

Distributed file systems operate on a cluster each assigned one or more roles such as clust

RocksDB w/ Journaling FS

User Space WAL

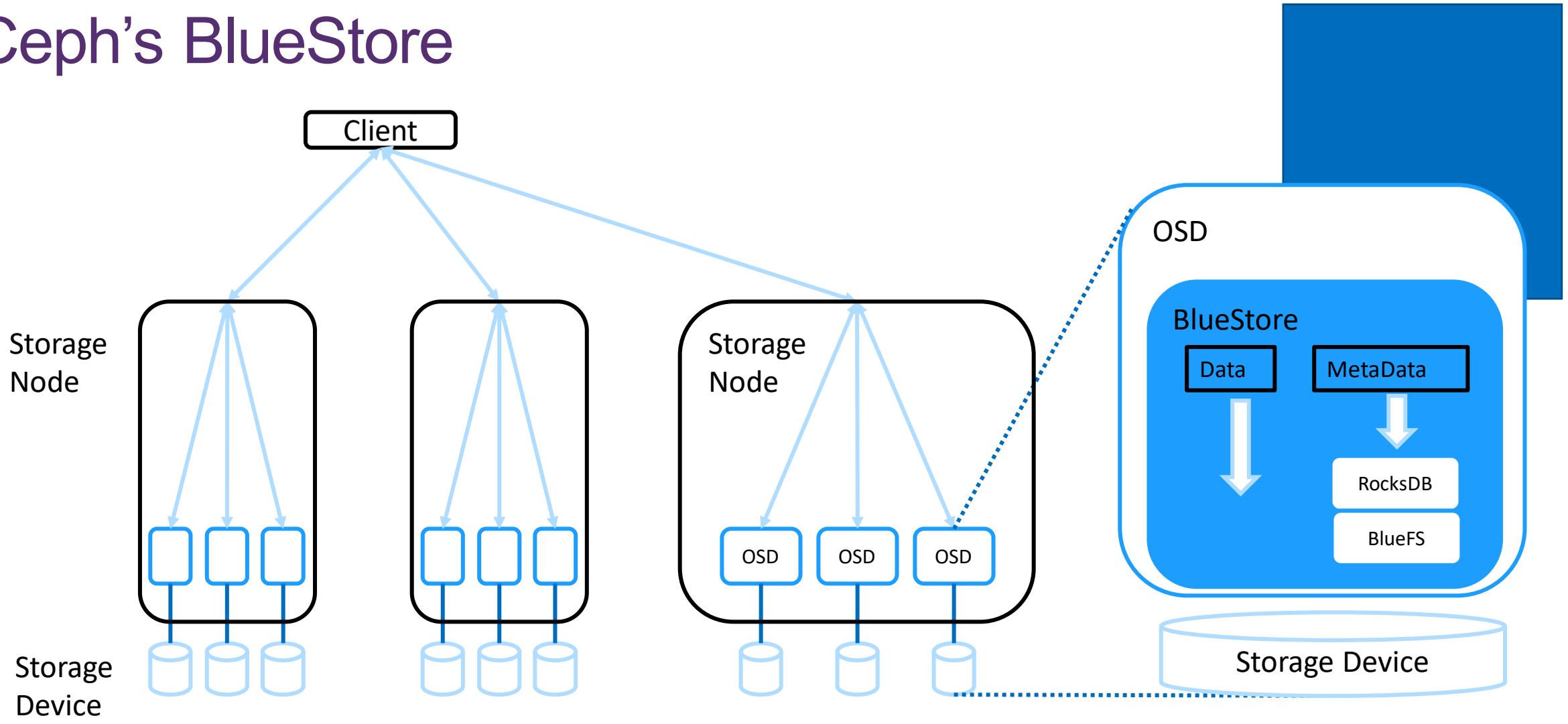
Hook into FS internals



Symposium on Operating Systems Principles, SOSP'19

NCE SDC 21

Ceph's BlueStore

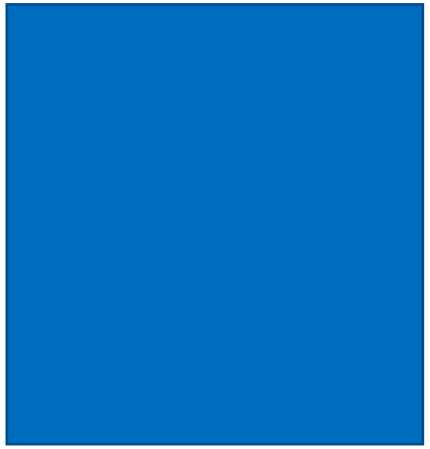


Bypassing file system, BlueFS **provides a log interface.**

BlueFS operates in user space, "**runs on raw storage device**"

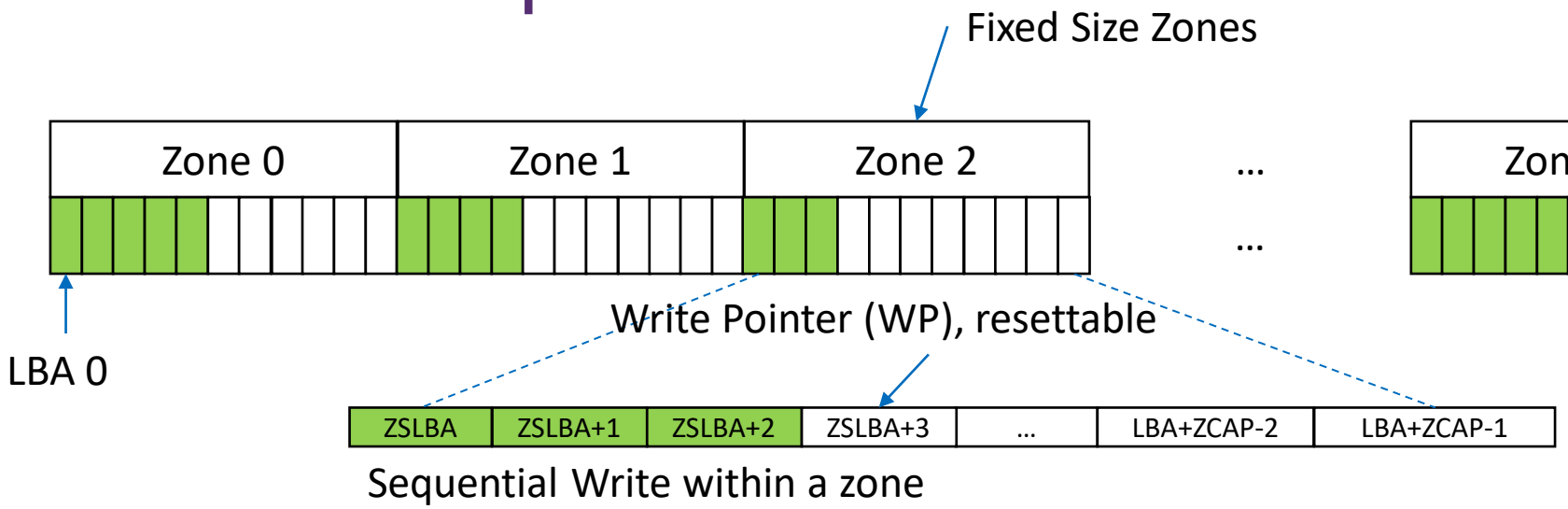
Stabilized in 2 years (not 10 years), due to simplicity and limited semantics.

Takeaways

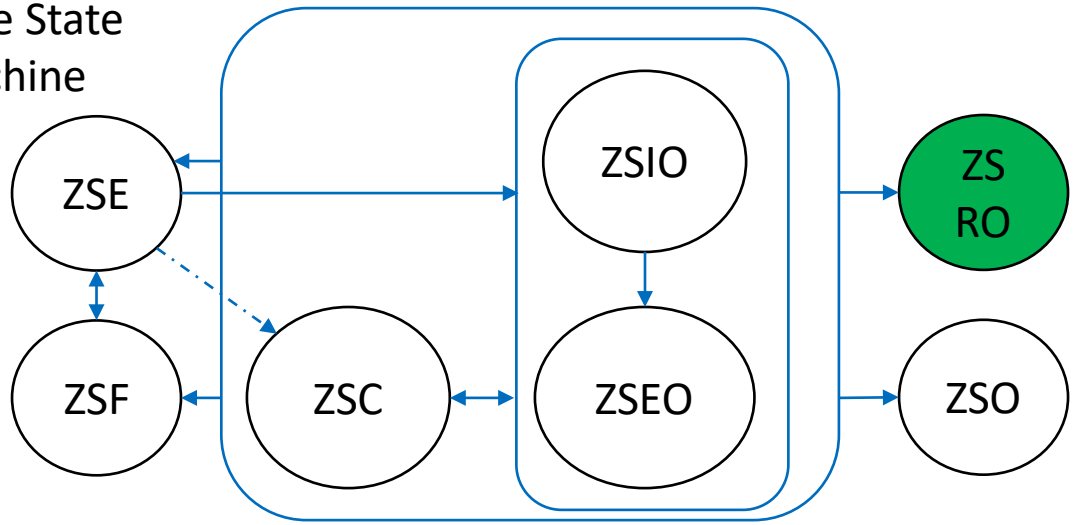


- Distributed “Databases”/Storages are using log-structured approach to manage data.
  - Log approach: append-only, immutable, delete-as-a-whole.
  - Logs vary in size, from several MBs to hundreds of GBs.
  - Limited Semantics!
- Most existing data processing frameworks are still using native file system, which is demonstrated to be “unfit”
  - Slow read-modify-write
  - Double writes
  - Slow to adopt new hardware like Zoned Namespace, which natively supports operations like append(), tail(), and seal().

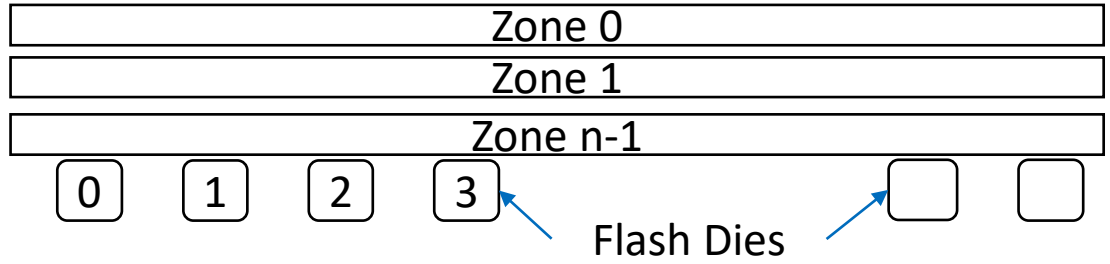
Zoned Namespace



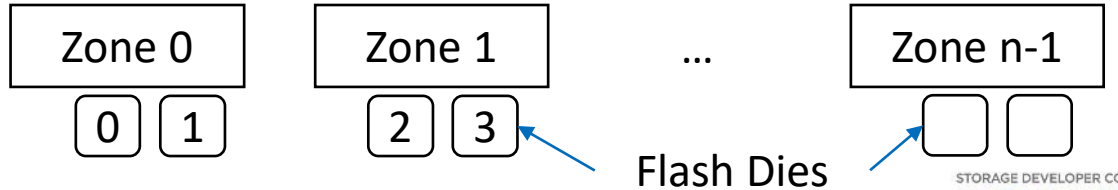
Zone State Machine



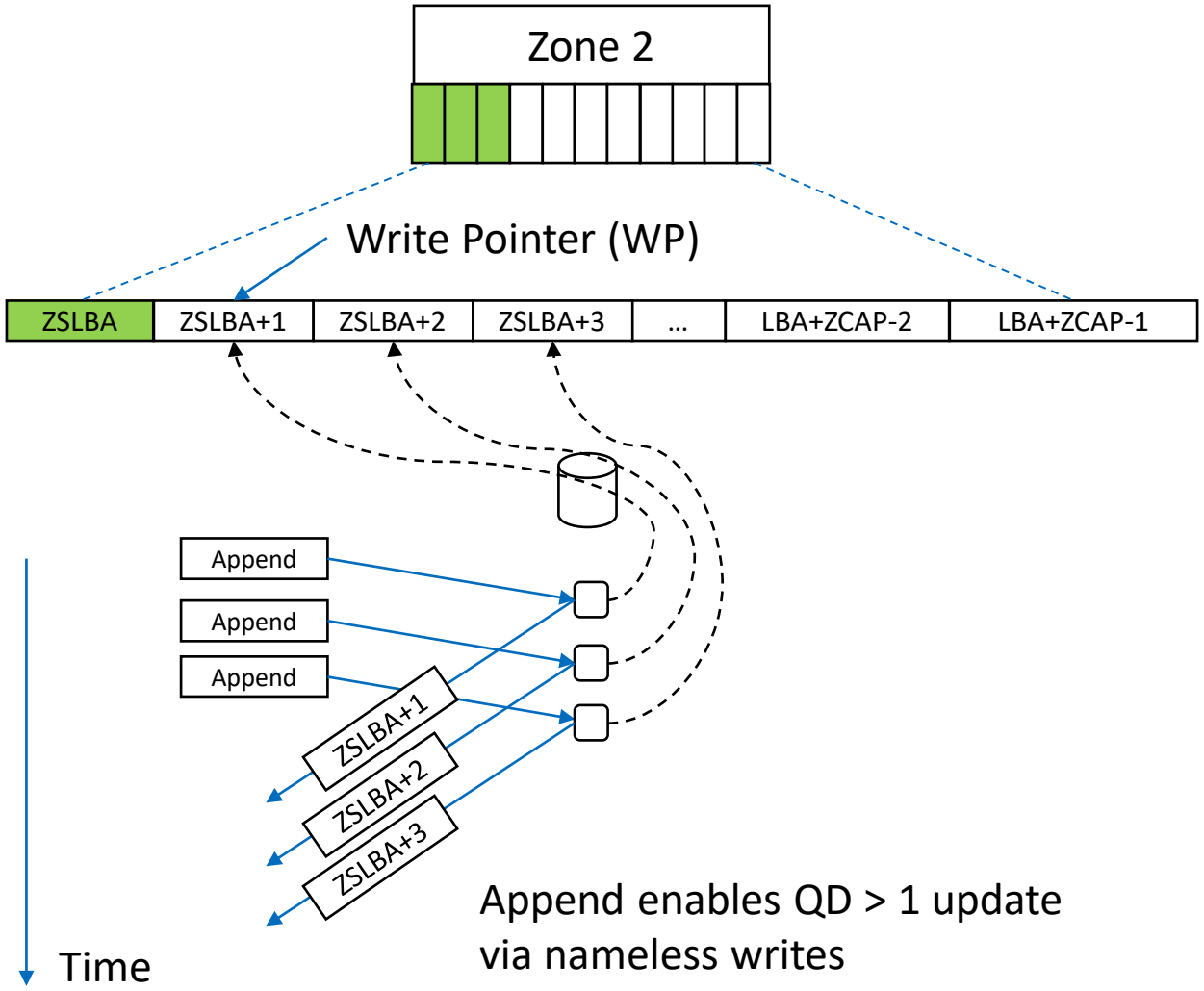
Zone Mapping Config A



Zone Mapping Config B



Zoned Namespace (contd.)



Append enables QD > 1 update via nameless writes

Improved Performance

- **Bandwidth** w/ reduced WAF
- **Tail latency** w/ isolation and reduced GC

Reduce **TCO**

- Less OP, DRAM, WAF and **QLC** adoption

Can we do more?
Can we do better?

Application Log in SSD

	Application Log	Zone in ZNS
Append	Yes	Yes
Immutable until delete	Yes	Yes
seal() for fencing	Needed to handle takeovers	Yes (Make Zone Read-Only)
tail()	Needed to synchronize writers	Yes (Query Zone WP, need to make it fast though)
Size	Variable Length	Fixed Size
Update Unit	May not aligned w/ sector	Sector Aligned
Name	Directory + Filename, or ObjectID	ZSLBA (requires FS to map name -> LBA)

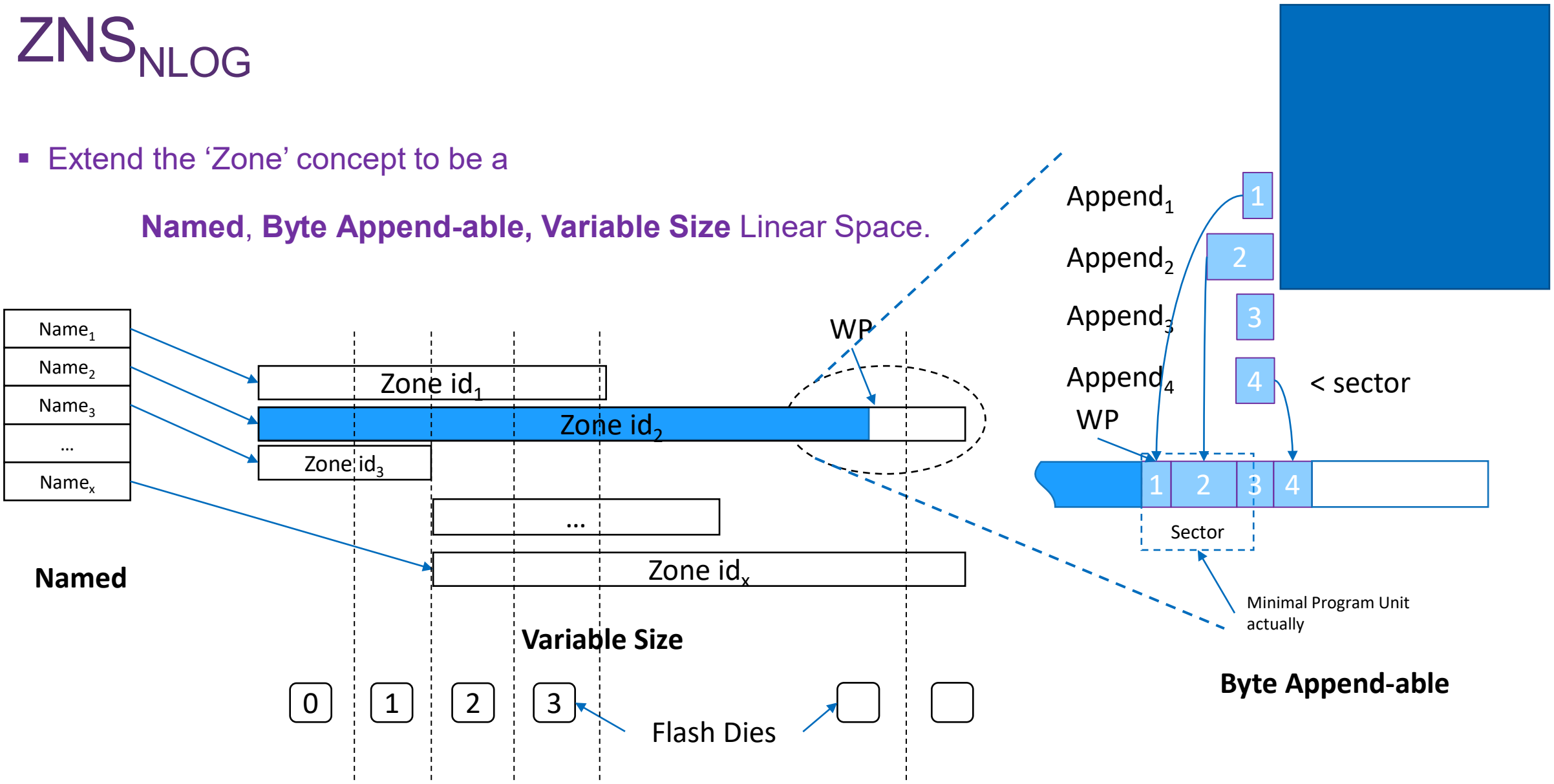
Map application logs natively onto ZNS? Two approaches:

- Map logs into zones… (multiple logs in one zone, one log span multiple zones)
  - Internal fragmentation
  - Garbage collection due to share a zone
  - Still need naming to map name -> LBA
- Extend the zone to support application logs with:
  - Variable Size
  - Byte-level Append
  - Naming

ZNS$_{NLOG}$

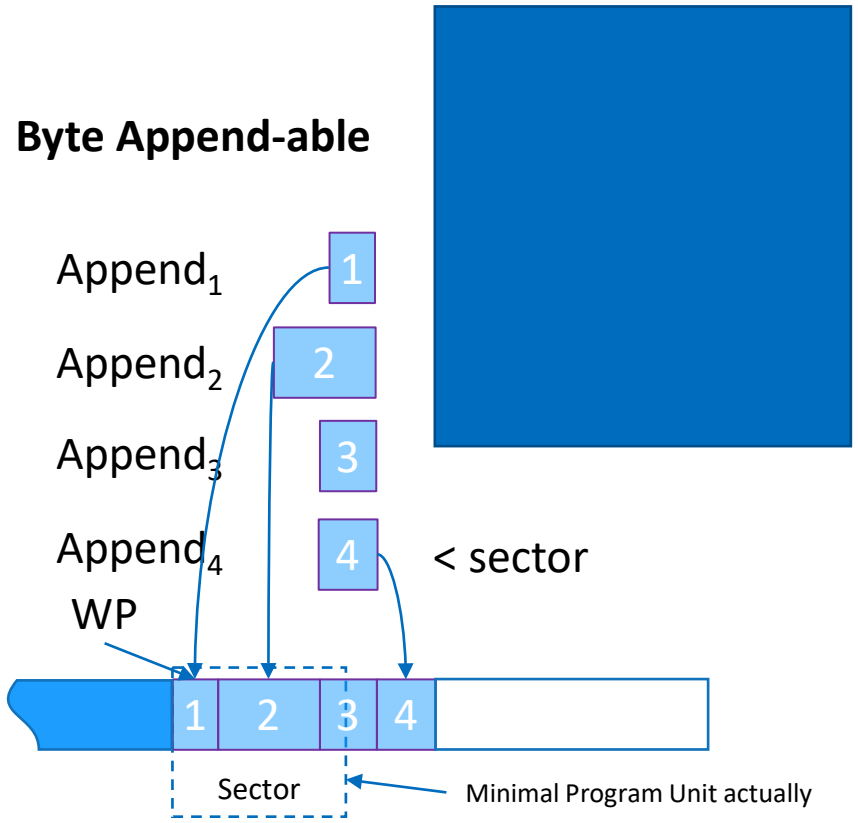
- Extend the ‘Zone’ concept to be a

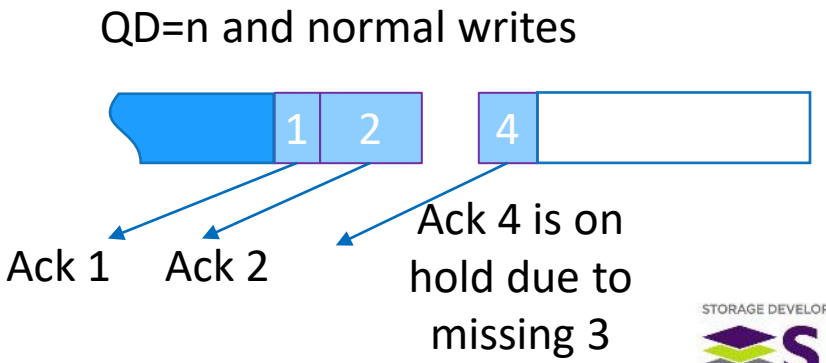
Named, **Byte Append-able, Variable Size** Linear Space.



Byte Append

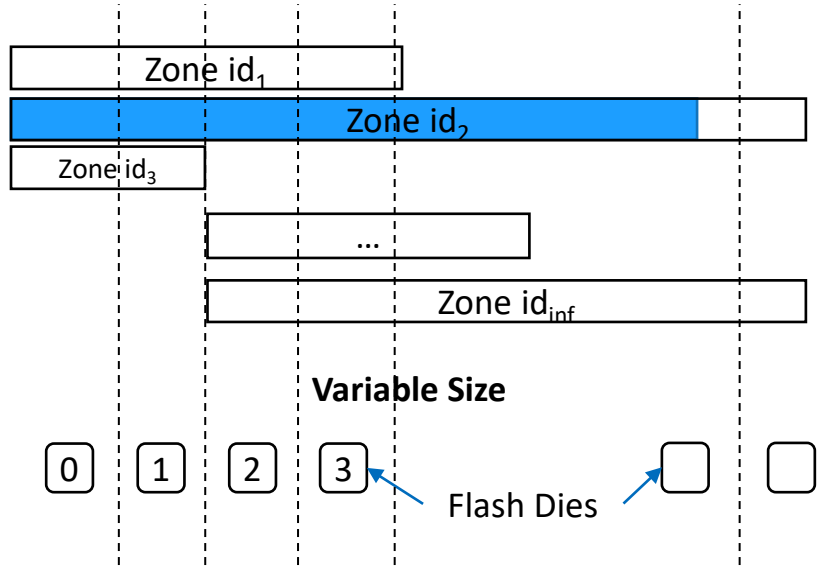
- Can be implemented efficiently
  - Works well with QD=1 and QD=n for nameless writes (appends) as well as normal writes.
  - QD=n normal writes can be re-ordered at the device
    - “Ack On Hold” for missing write
    - Throughput as good as QD=n nameless writes
    - Ordering for replicated NLOGs.
- Benefits to applications
  - Partial sector read-modify-write eliminated.
  - Greatly reduce the bandwidth requirement for remote storage nodes.
    - Up to several folds of reduction for normal WAL in RocksDB.





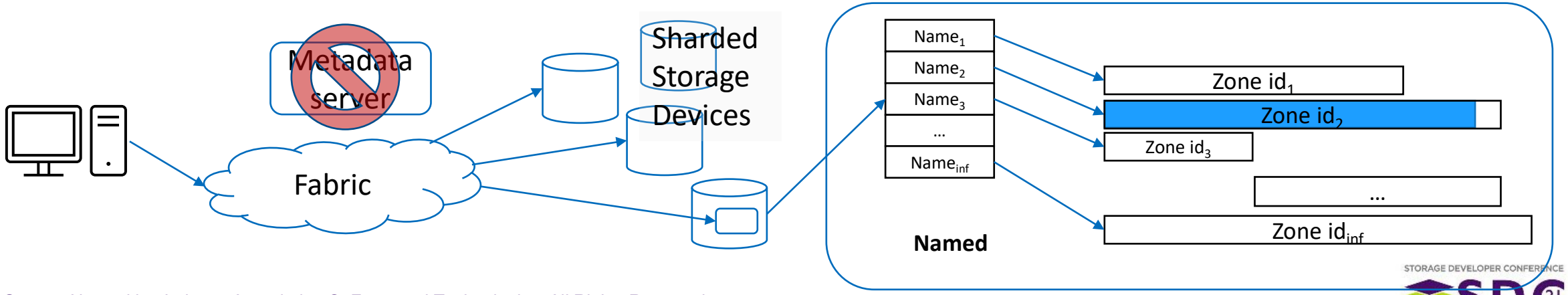
Variable Size Zones

- Q: Why ZNS device can achieve WA=1.0?
- Can we keep WA = 1.0 when segment is smaller than a zone?
  - Probably, if we can group all segments to be deleted together to the same zone, and pray...
- Accommodate both small segments and huge segments?
  - Zone's write throughput is determined by the # of dies it spans.
- Best of both?
  - Key insight: minimal unit is the flash block size of each die.
  - Zone spans 1) one die or 2) all dies or 3) any where in between.
  - User can demand a zone with different performance characteristics.



Named Zones

- Variable size zone: impossible to name zone using their ZSLBA.
- Name the zone using the “filename” or “object name”!
  - Bypass the file system’s naming service. Eliminates filename->LBA mapping.
- Works over fabric as well.
  - Directly send to sharded storage device with global name.
  - Eliminated the mapping which maps global name to a storage device and its local filename. Reducing remote access latency, and metadata server work.



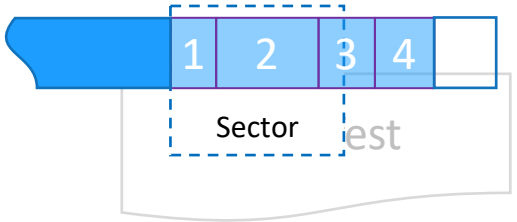
Bridge Semantical Gap

	Application Log	Zone in ZNS_{NLOG}
Append	Yes	Yes
Immutable until delete	Yes	Yes
seal() for fencing	Needed to handle takeovers	Yes (Make Zone Read-Only)
tail()	Needed to synchronize writers	Yes (Query Zone WP, need to make it fast though)
Update Unit	May not aligned w/ sector	Sector and Byte-Append (great for WAL)
Size	Variable Length	Variable Size (not arbitrary size)
Name	Directory + Filename, or ObjectID	Global Names (no FS or Meta Server mapping)

RocksDB over ZNS$_{NLOG}$

Persistent

Write-Ahead Log (WAL)



Current

No Read-Modify-Write
Improve WAL write

SST (Static Sorted Table) File

L0

SST SST SST SST SST L1

SST SST SST SST SST L2

Name: L1_F_TO_M_010.SST

Name: L2_X_TO_Z_005.SST

SST SST SST SST SST Ln

Key Range

Zone Sm x 65536

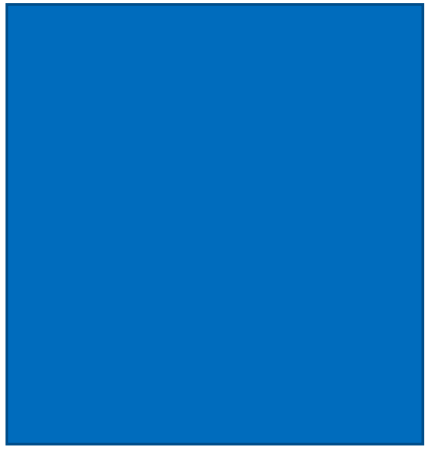
Zone Md x 4096

Zone Lg x 512

Naming the Zone eliminates filename->LBA mapping (file system), **making it fast and robust.**

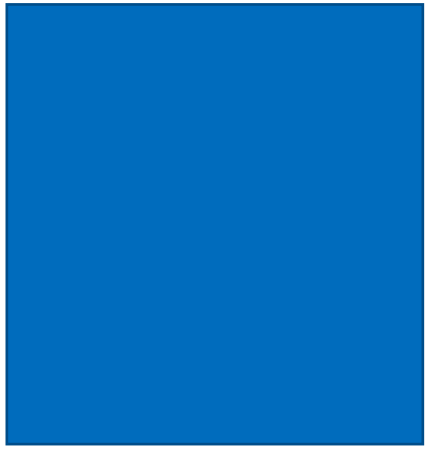
Map different size SST files onto **dedicated** size-matching zones.
Less WA overall.

ZNS_{NLOG} enables more NDP



- Transparent compression of the logs/zones.
  - Much larger size = better compression ratio.
  - Maintain original logical offset.
- Offload RocksDB's operations:
  - Compaction of SST files (merge-sort).
    - One zone is one SST file, no more native file system indirection.
    - Compaction can be offloaded to the SSD to leverage internal SSD bandwidth.
  - Search multiple SST files on the device.
  - Wildcard search, not supported by the current prefix or normal bloomfilter.
- Offload Kafka's matching operations.

ZNS_{NLOG} helps building distributed storage



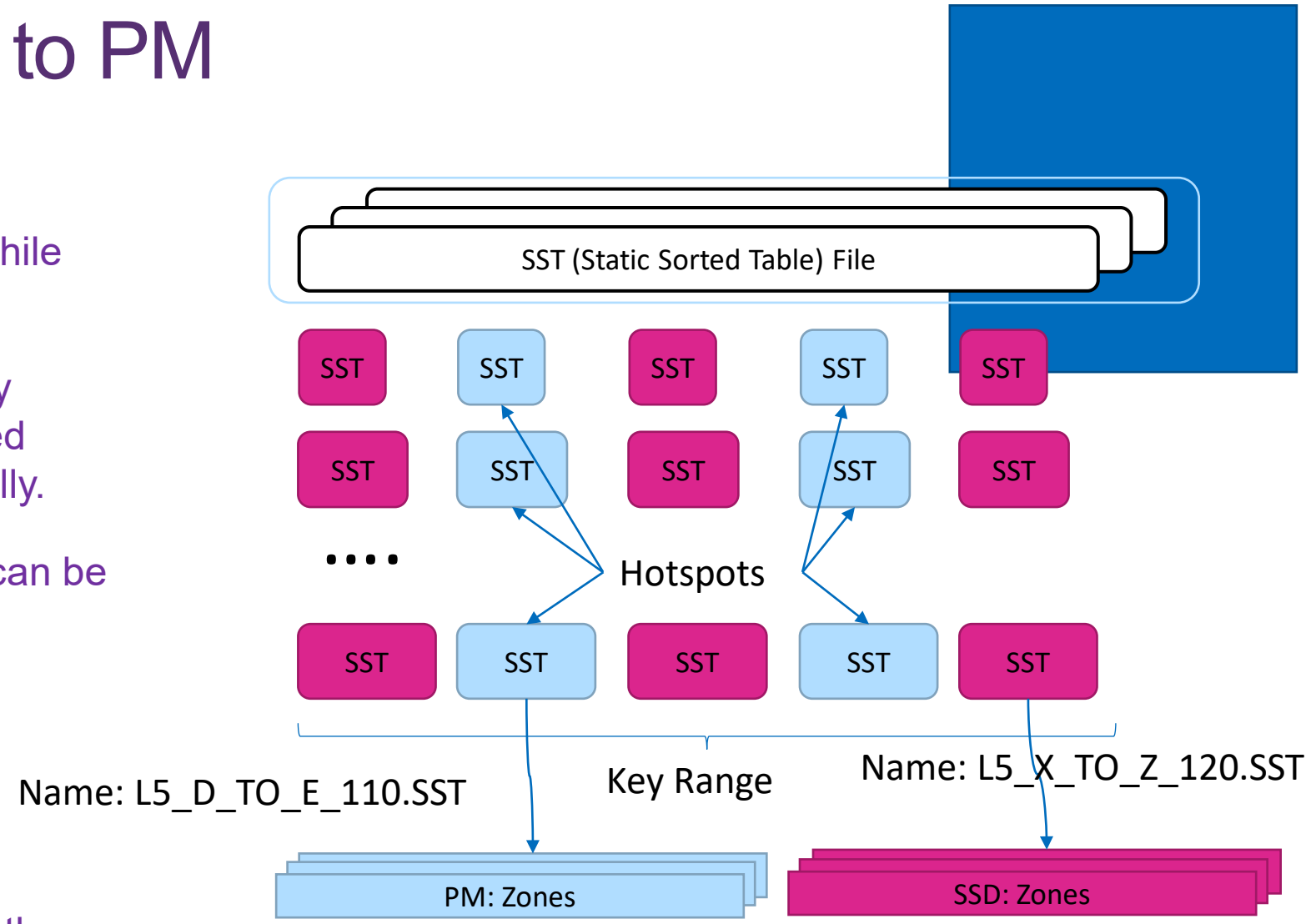
- Replicated Storage
  - Writes can be replicated to multiple ZNS_{NLOG} devices, since the write addresses dictates the ordering.
  - Named LOG eliminates some mapping tables on metadata server.
    - Single Log can be named directly using global file or object name.
    - Replicated Log can be named LOG.R1, LOG.R2, LOG.R3 and so on.
    - Erasure coded Log can be named LOG.1/3, LOG.2/3, LOG.3/3, LOG.P/3, LOG.Q/3 for 3+2

ZNS_{NLOG} applicable to PM

Our experience shows logging is an excellent way to use PM as storage, while write-in-place is excellent for caching.

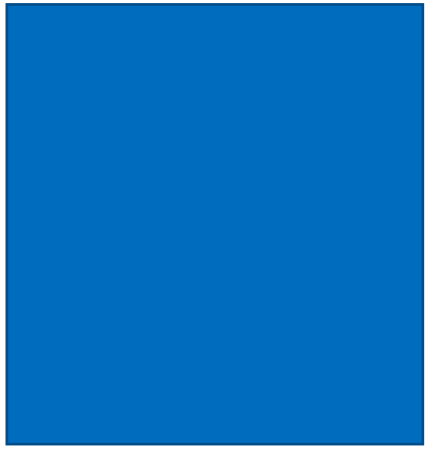
- PM is byte-addressable and memory allocator dictates the size of allocated memory, variable size comes naturally.
- Adding a naming service, ZNS_{NLOG} can be easily implemented on PM.

This way we can provide the same semantics on both PM and SSD, unify the two.



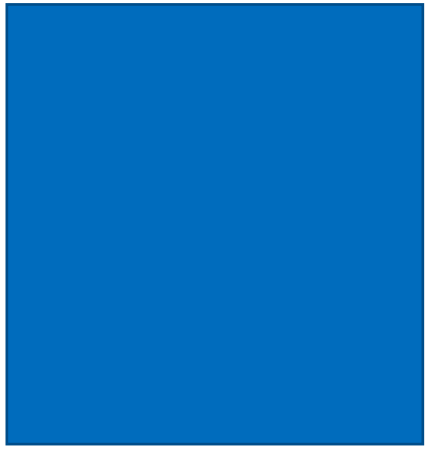
RocksDB w/ SST on tiered zones: PM Zones and SSD Zones

Futurewei Plan



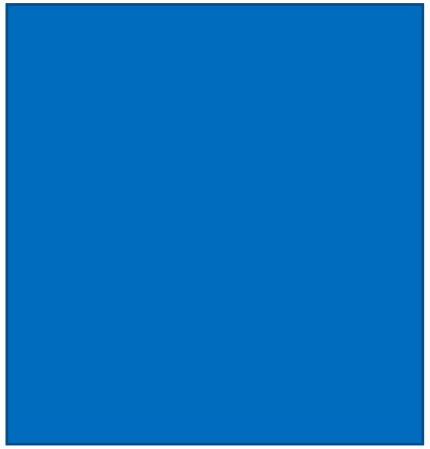
- Semantically equivalent prototype by Huawei.
- Available to partners now, remote access.
- Plan to propose ZNS_{NLOG} extension to NVM Express.
  - Validated by several partners.
  - Mature enough.

Call for Collaborations



- Futurewei is sponsoring:
  - ZNS_{NLOG} support in FEMU
  - RocksDB/LevelDB over ZNS_{NLOG}
- Need partners on:
  - Improving the specification
  - Alternative implementations
  - Adapt more log-based applications like Kafka, BookKeeper
- Contact chun.liu@futurewei.com or nelson.liao@futurewei.com

Conclusion



- ZNS_{NLOG} can bridge the semantical gap between applications and SSD, which traditionally was blurred by file systems.
  - Naming, Byte Append-able, Variable Size.
- ZNS_{NLOG} enables less write amplification, more log write performance, and provides more functionality for distributed system.
- ZNS_{NLOG} lowers the technical barrier for near data processing.
- ZNS_{NLOG} concept is applicable to Persistent Memory.
- Collaborations!

Please take a moment to rate this session.

Your feedback is important to us.

