PERSISTENT MEMORY + SUMMIT 2021 COMPUTATIONAL STORAGE

FROM DATACENTER TO EDGE : VIRTUAL EVENT
APRIL 21-22, 2021

Beyond Zoned Namespace, What Do Applications Want?

Chun Liu, Chief Architect, Futurewei Technologies

Landscape

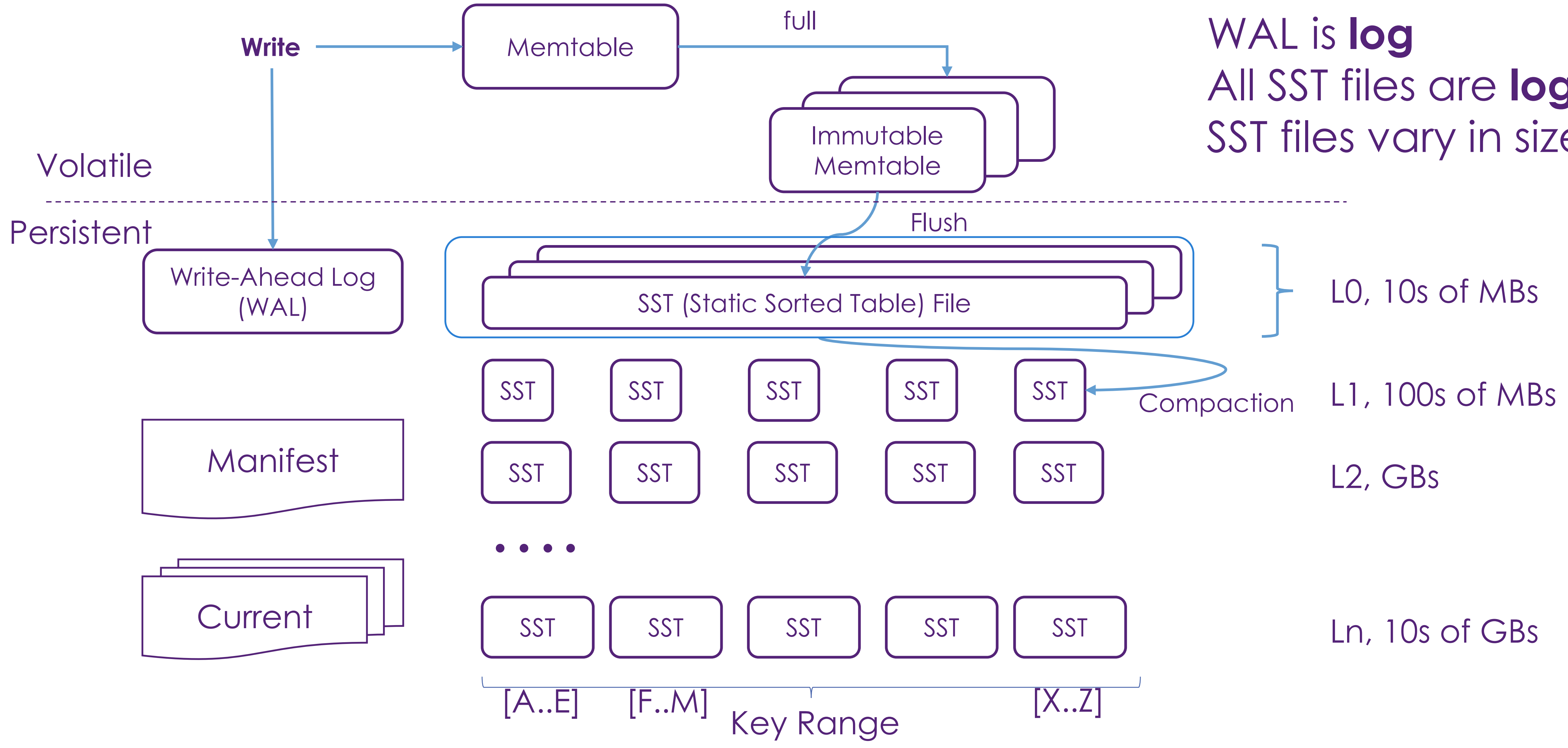
hadoop MapReduce   APACHE Spark   HIVE   APACHE STORM™   samza   Flink   HERON

Database/ DataStore   APACHE HBASE   cassandra   RocksDB   LevelDB

Distributed Storage   hadoop HDFS   kafka

Native File Systems   Ext4   XFS   BtrFS   F2FS   ZFS   …

RocksDB builds on logs

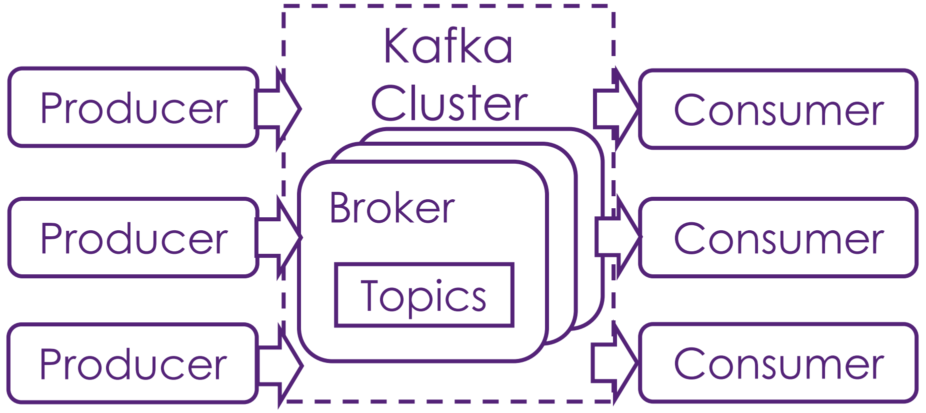


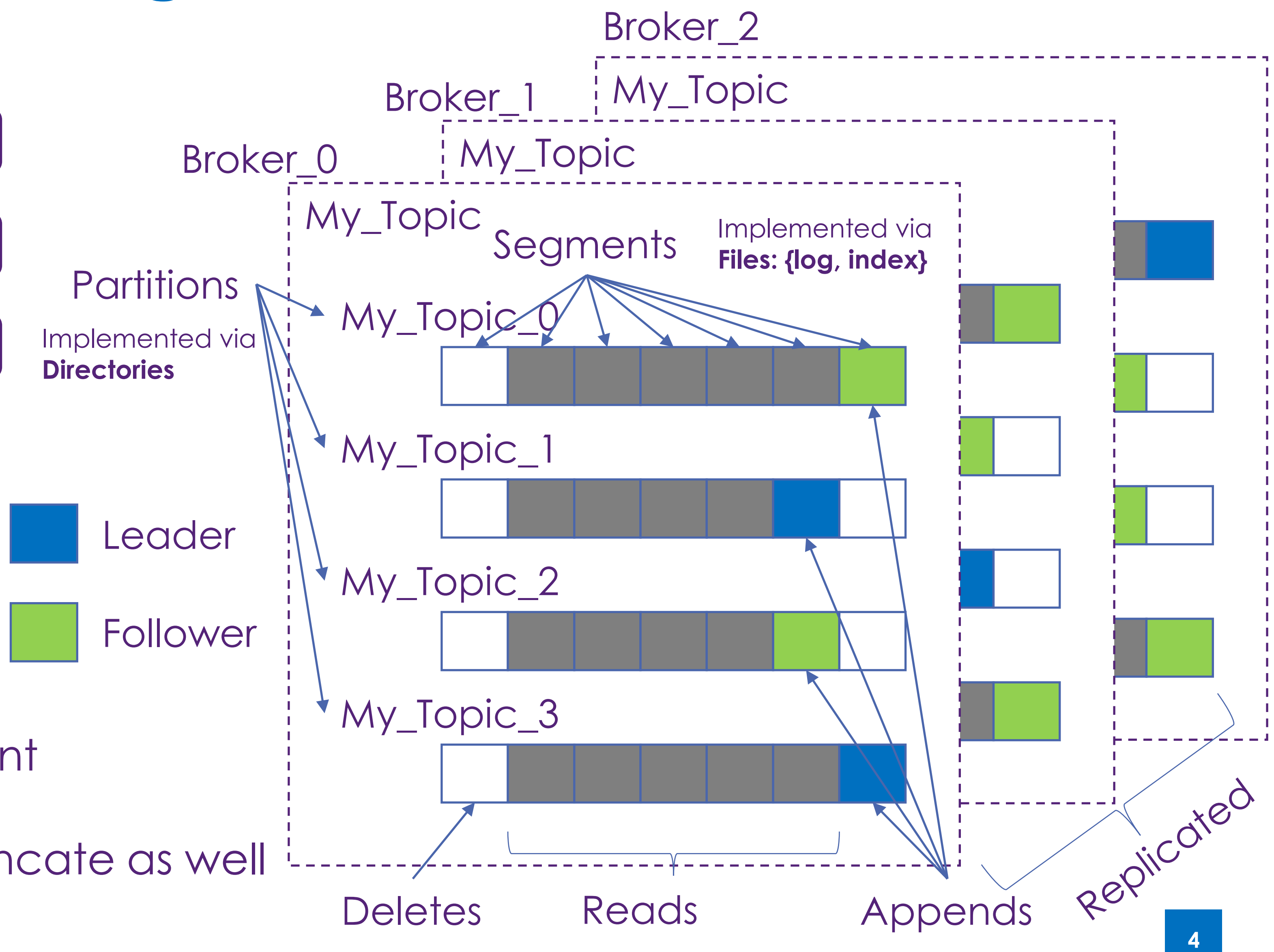
WAL is **log**

All SST files are **logs**

SST files vary in size

Kafka also builds on logs

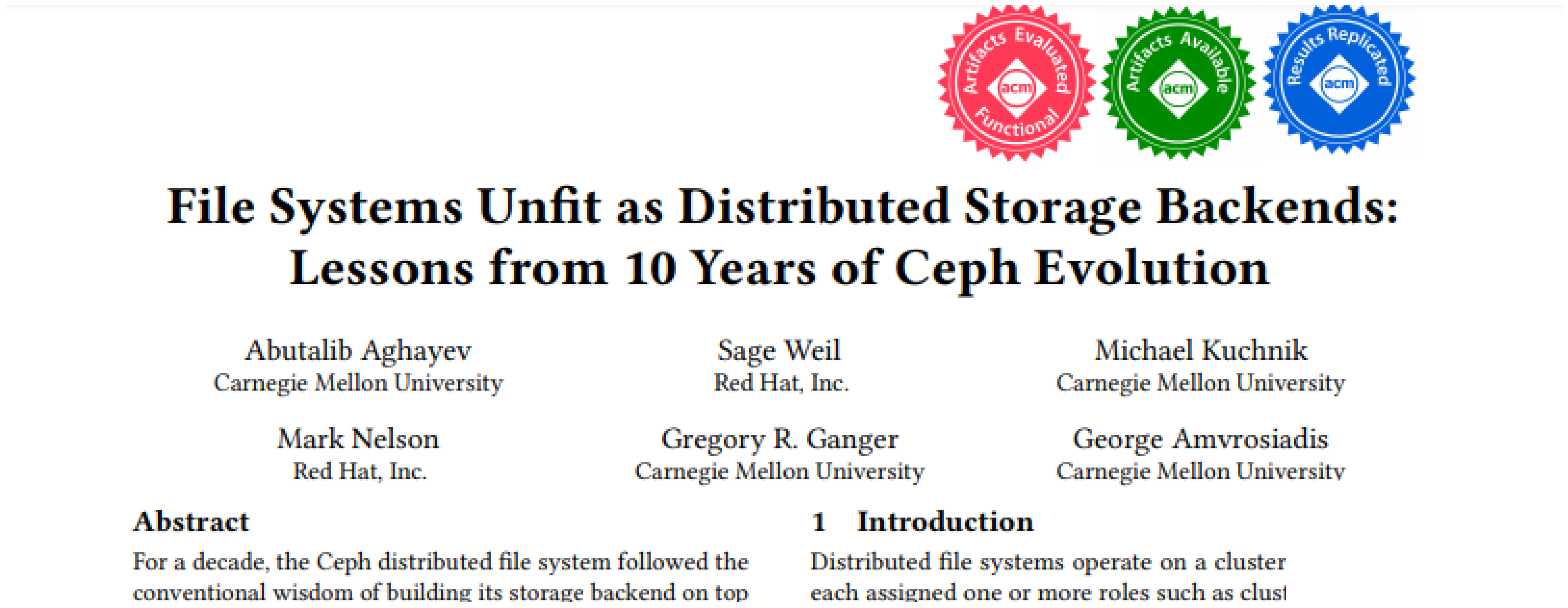




Partition is a **log**
Segments are **logs**
Partition append: create segment
Partition truncation: delete segment

HDFS is write-once + append + truncate as well

Ceph's Lessons



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

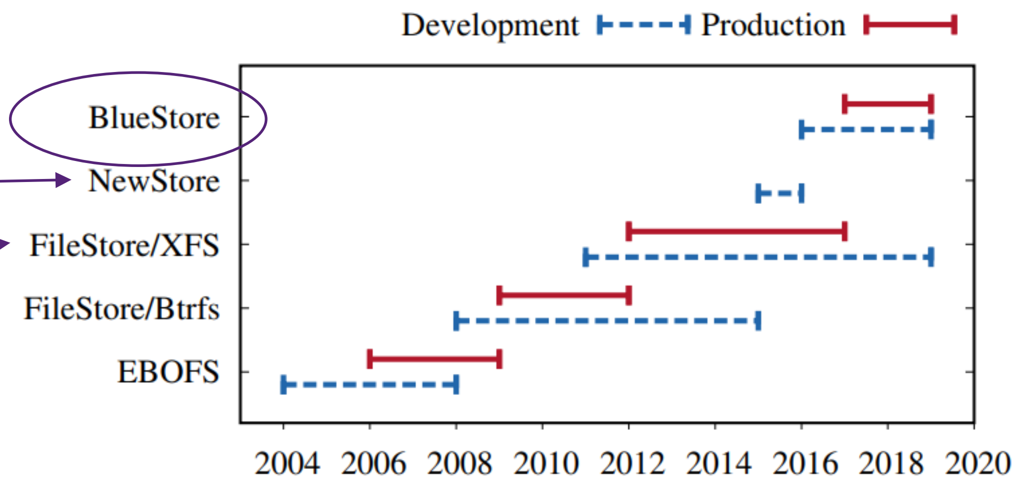
Abutalib Aghayev, Carnegie Mellon University
Sage Weil, Red Hat, Inc.
Michael Kuchnik, Carnegie Mellon University
Mark Nelson, Red Hat, Inc.
Gregory R. Ganger, Carnegie Mellon University
George Amvrosiadis, Carnegie Mellon University

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

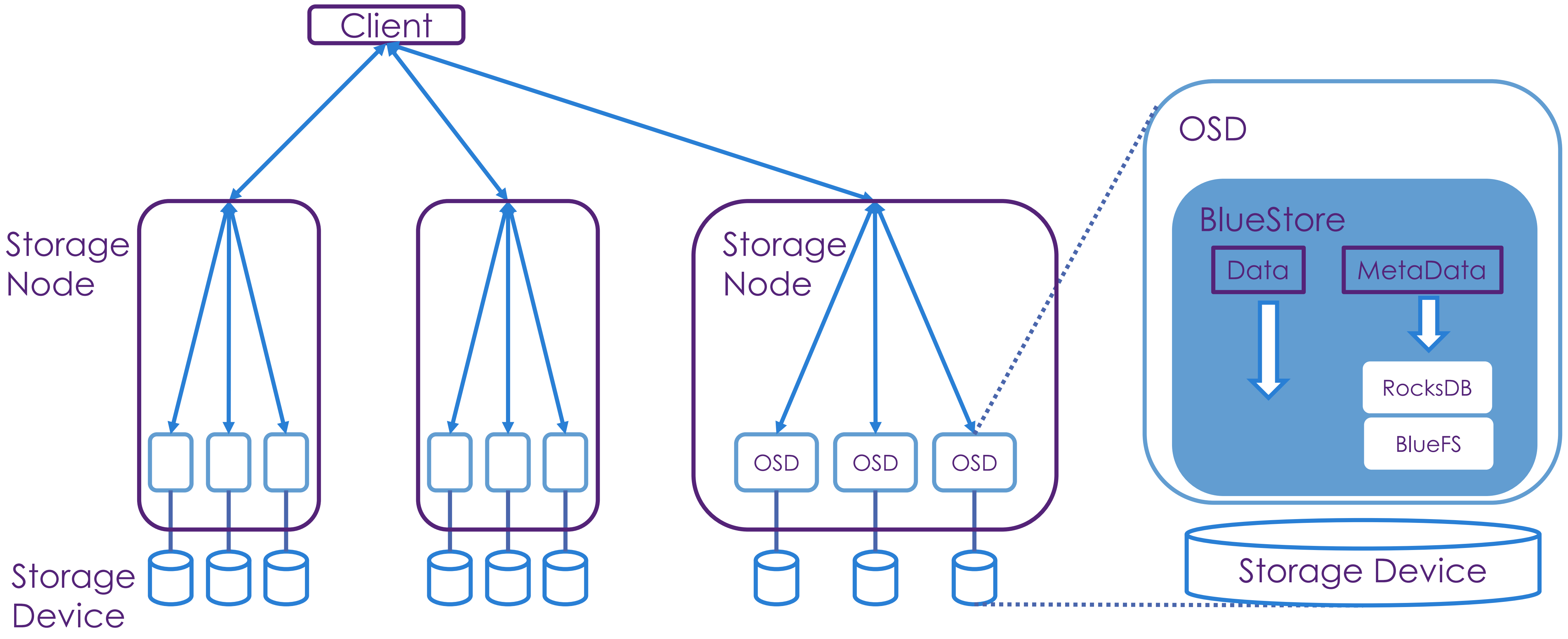


RocksDB w/ Journaling FS

User Space WAL

Hook into FS internals

Ceph's BlueStore

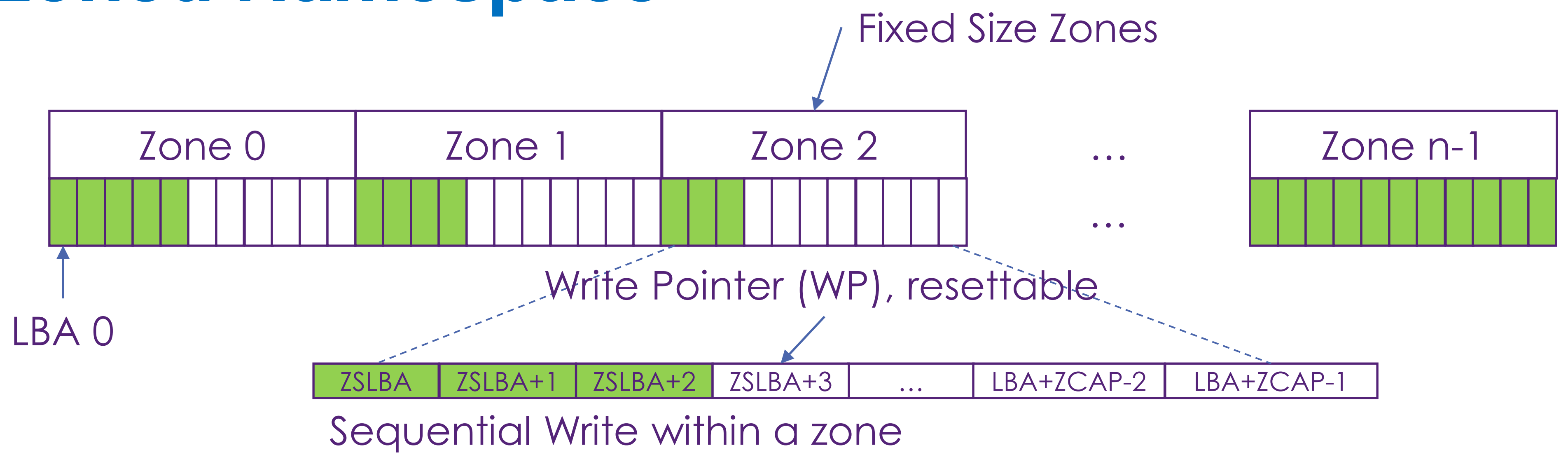


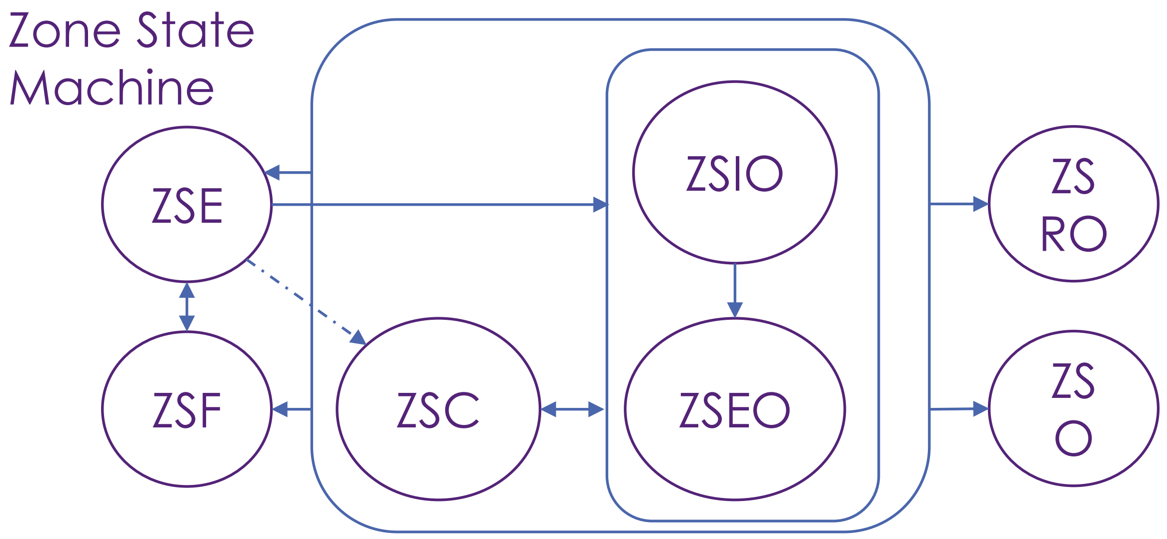
Bypassing file system, BlueFS **provides a log interface.**
BlueFS operates in user space, "**runs on raw storage device**"
Stabilized in 2 years (not 10 years), due to simplicity and limited semantics.

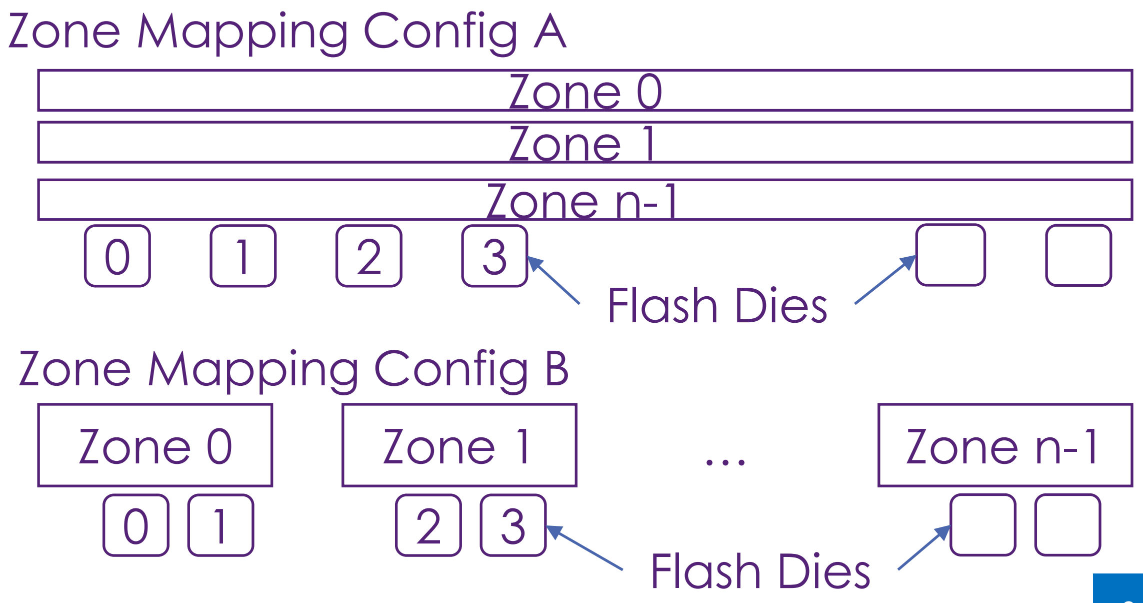
Takeaways

- “Databases”/Distributed 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.
- 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.

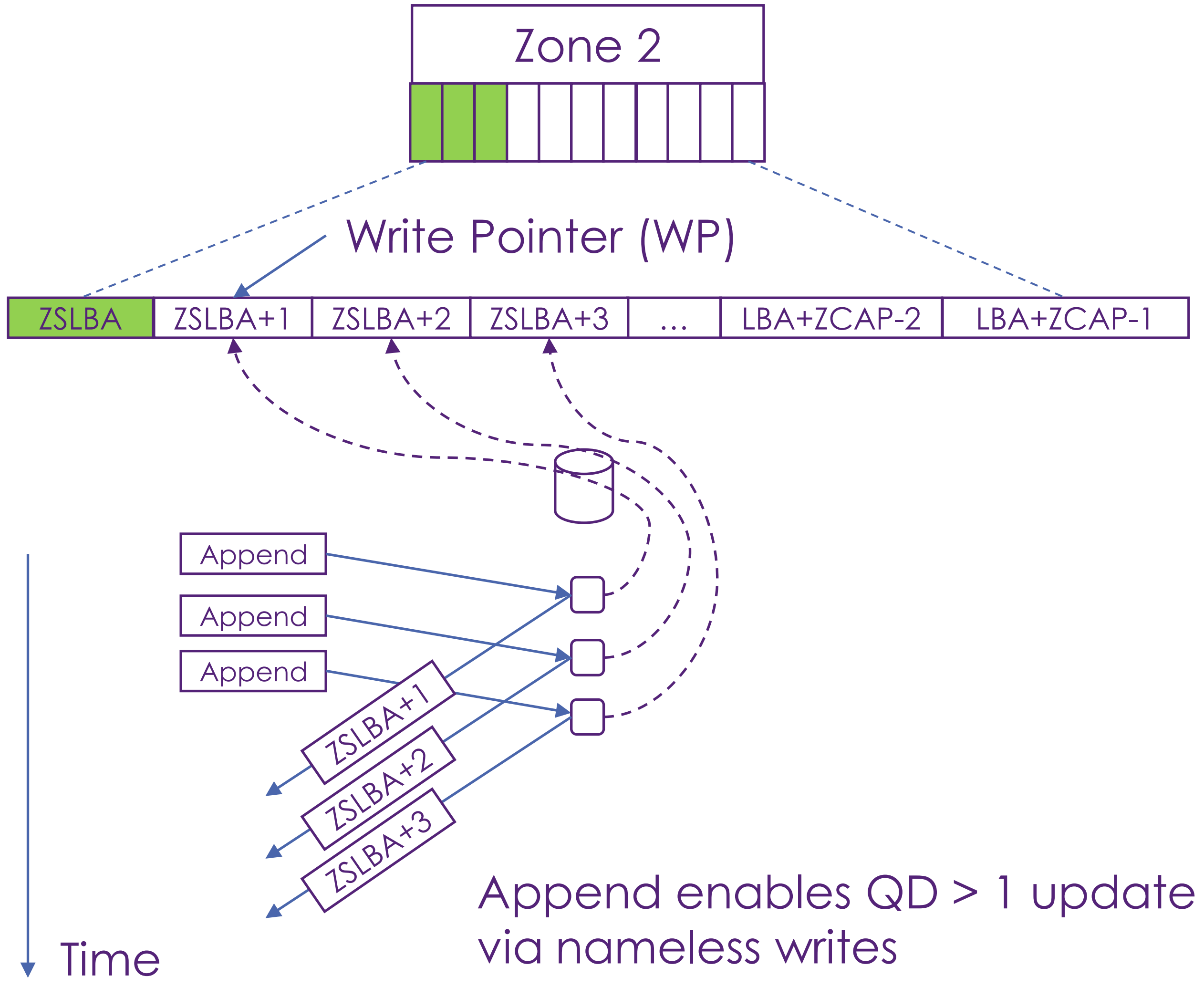
Zoned Namespace







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

SMR Drive: **Extra Capacity**.

Can we do more?
Can we do better?

Application Log in SSD

	Application Log	ZNS
Append	Yes	Yes
Immutable until delete	Yes	Yes
Size	Variable Length	Fixed Size
Update Unit	May not aligned w/ sector	Sector Aligned
Name	Directory + Filename	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 shared zone

    Still need naming to map name -> LBA

- Extend the zone to support application logs with:

    Variable Size

    Byte-level Append

    Naming

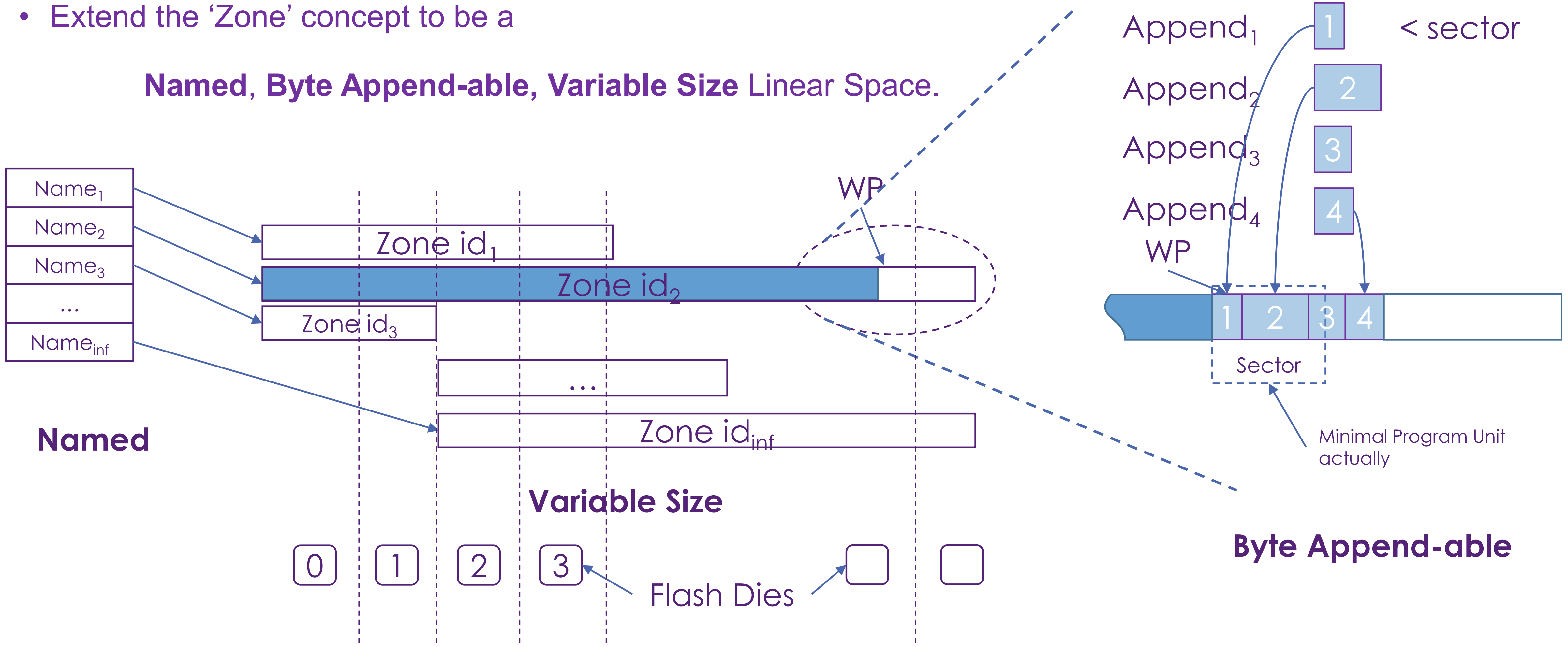
nvm EXPRESS ZNS$_{NLOG}$

ZNS Named Log Extension or ZNS$_{NLOG}$

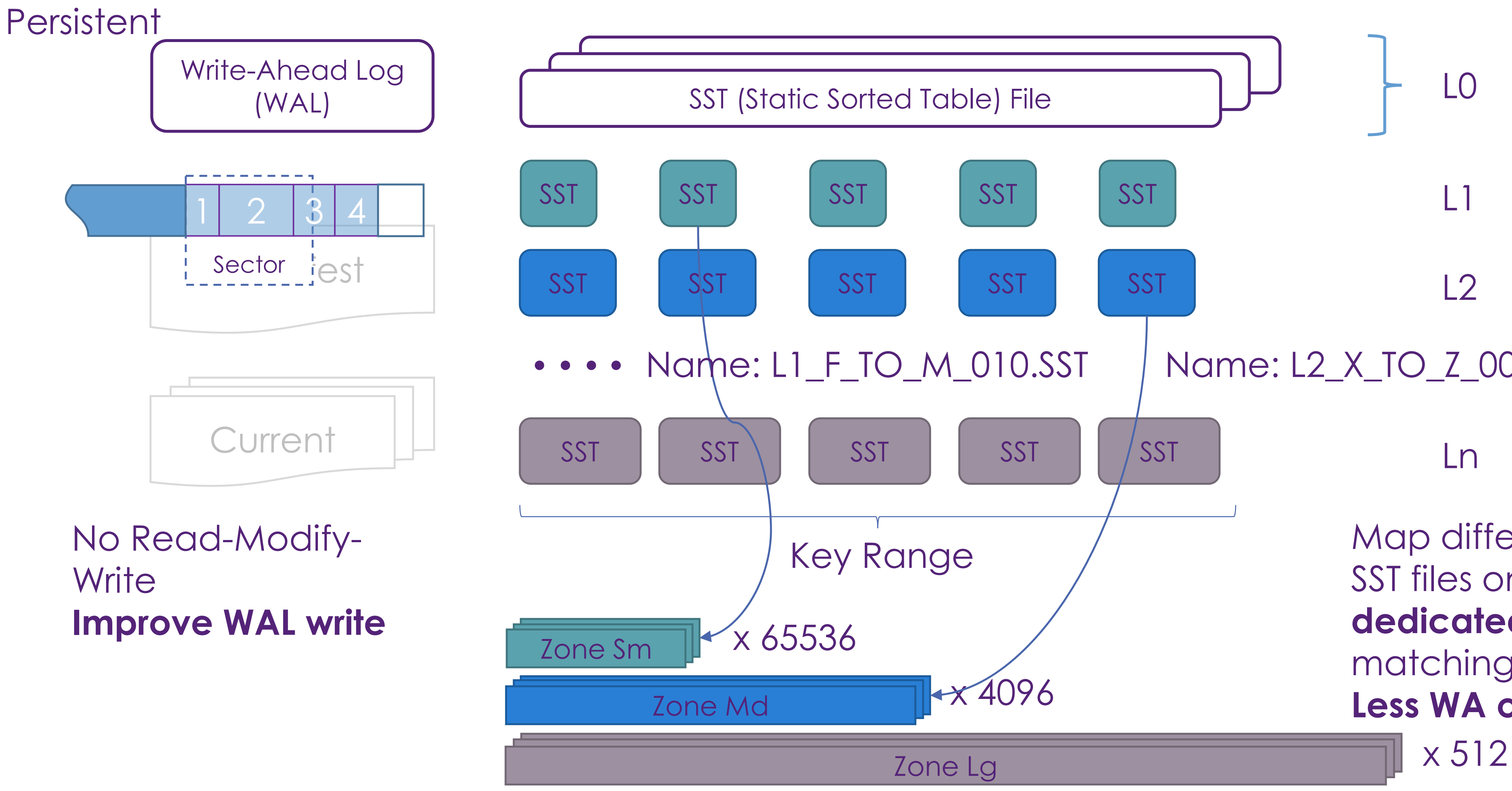
ZNS$_{NLOG}$

- Extend the ‘Zone’ concept to be a

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



RocksDB over ZNS$_{NLOG}$



Naming the Zone eliminates filename->LBA mapping, **making it robust.**

No Read-Modify-Write
Improve WAL write

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

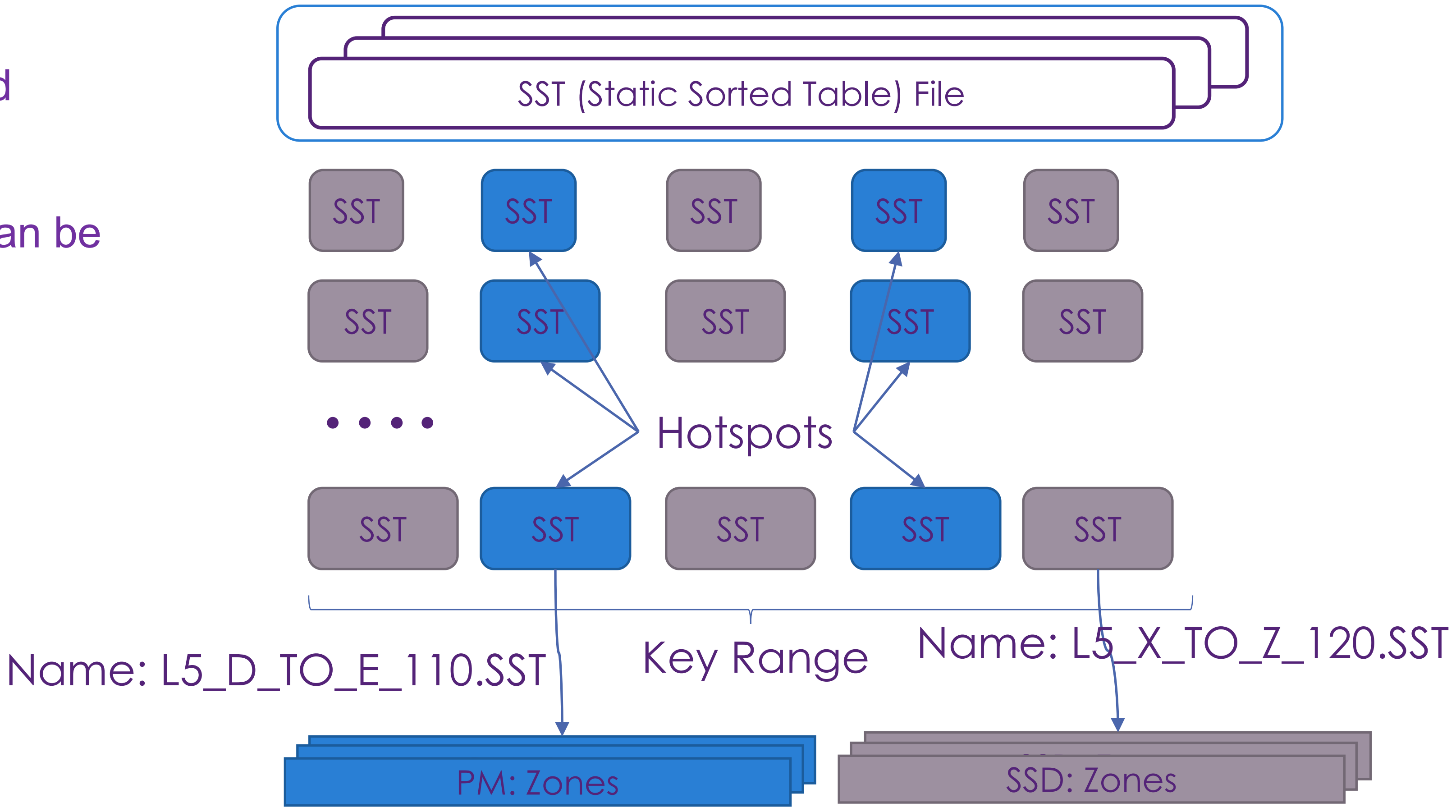
ZNS$_{NLOG}$ enables more NDP

- Transparent compression of the logs.
  - 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.

PERSISTENT MEMORY + SUMMIT 2021 COMPUTATIONAL STORAGE

ZNS$_{NLOG}$ applicable to PM

- PM is byte-addressable and memory allocator dictates the size of allocated memory.
- Adding a naming service, ZNS$_{NLOG}$ can be easily implemented on PM.



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

Conclusion

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

Thank you

Please visit www.snia.org/pm-summit for presentations