Building a Production Storage System in Practice

Architecture, Design, and Beyond

Wenguang Wang
Distinguished Engineer
Broadcom

April 13, 2026

Thanks to Eric Knauft for some slides I’ve borrowed from him.

About Myself

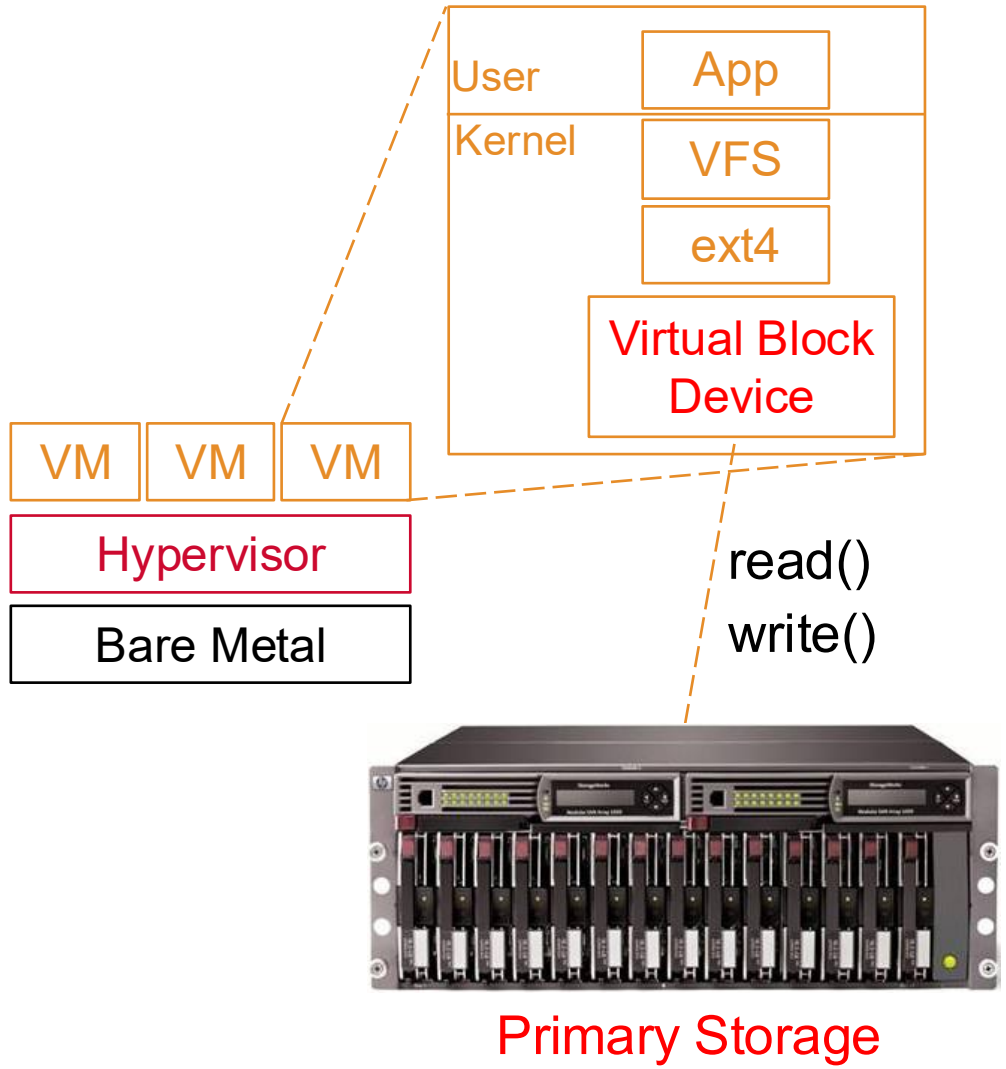
- Ph.D. in CS in 2004
- Joined Apple in 2004, led:
  - File Vault 2
  - Fusion Drive
  - File Vault Everywhere
- Joined vSAN team of VMware (now Broadcom) in 2014, led:
  - vSAN OSA End-to-end checksum
  - vSAN OSA Data-at-rest encryption
  - vSAN ESA
  - vSAN File Service
  - vSAN Object Service
  - Multiple internal productivity projects
- 178 granted patents

- In my free time
  - Exercise (Olympic weightlifting, zone 2, HIIT)
  - Books (economics, physics, math, business, health, fiction, and many more)
  - Study investment tips and share them on YT

BROADCOM

Storage in Data Center Infrastructure

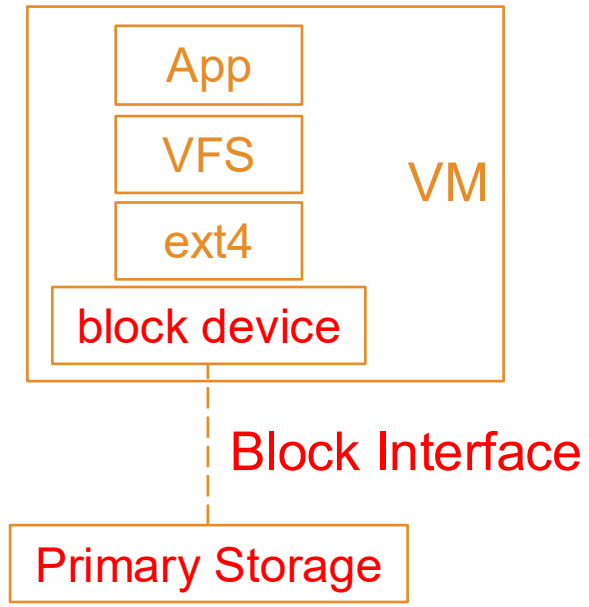
- Applications run on “computers”, which can be:
  - Physical box (bare metal)
  - Virtual machine (VM)
- VMs run in Hypervisor (ESXi, KVM, …)
- The virtual “block device” backing the file system (e.g., ext4, NTFS) is the “Primary Storage”
- It provides a Block Interface:
  - Read N blocks at a block address
  - Write N blocks at a block address

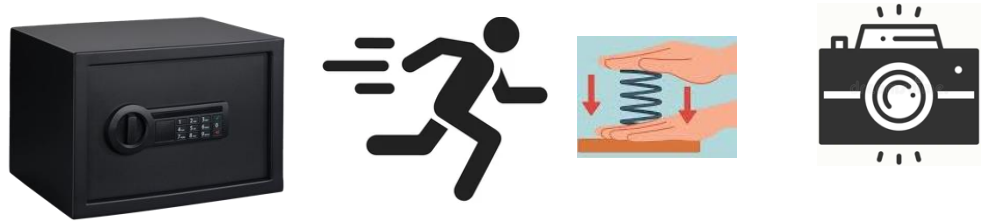


BROADCOM

Key Features of Primary Storage in Data Centers

- Safe
  - Crash resistance
  - Fault tolerance
  - Checksum verification
- Fast
  - IOPS
  - Throughput
  - Scalability
- Cheap
  - Space saving: compressing, deduplication
- Other
  - Backup/restore: snapshot
  - …



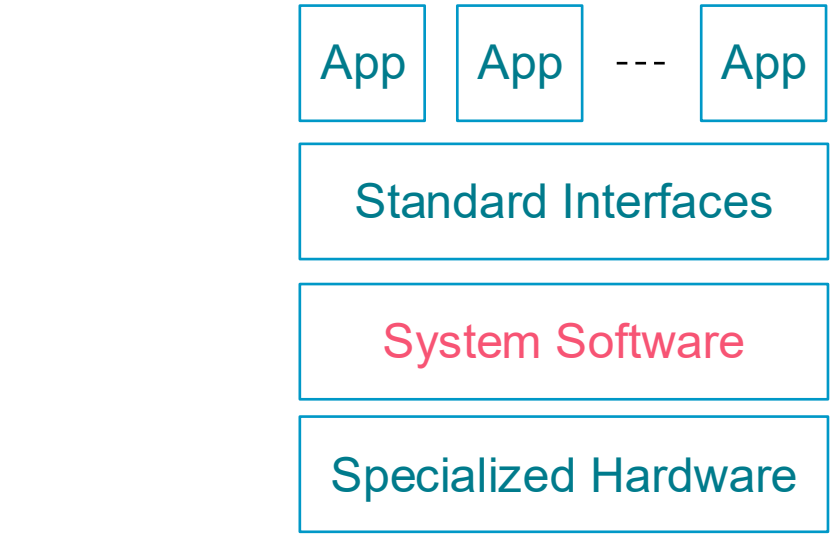


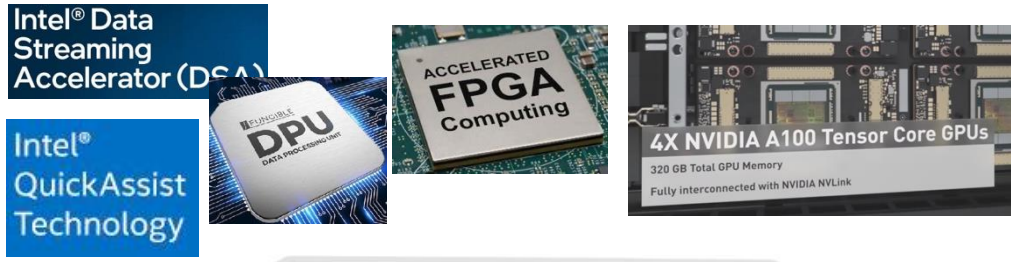
Safe Fast Space Efficient Snapshot

BROADCOM

Technology Trends for System Software

- Old hardware trend: Moore's law
  - Faster, cheaper, higher capacity, lower power consumption
- New hardware trends
  - Specialization
    - DSA, QAT, DPU (SmartNIC), FPGA, GPU, …
  - Disaggregation
    - Compute-Storage disaggregation: NVMe-oF + JBOF
    - CPU-Memory disaggregation: CXL + RDMA + Remote RAM
- Interface trends: growing but standardized
  - Block, file, object
  - MySQL (TiDB), Postgres (CockroachDB, YogabyteDB)
  - Redis (MemoryDB, KeyDB, Tair)
  - Kafka (RedPanda)
  - …
- System software hides hardware differences and provides a standardized interface





JBOF

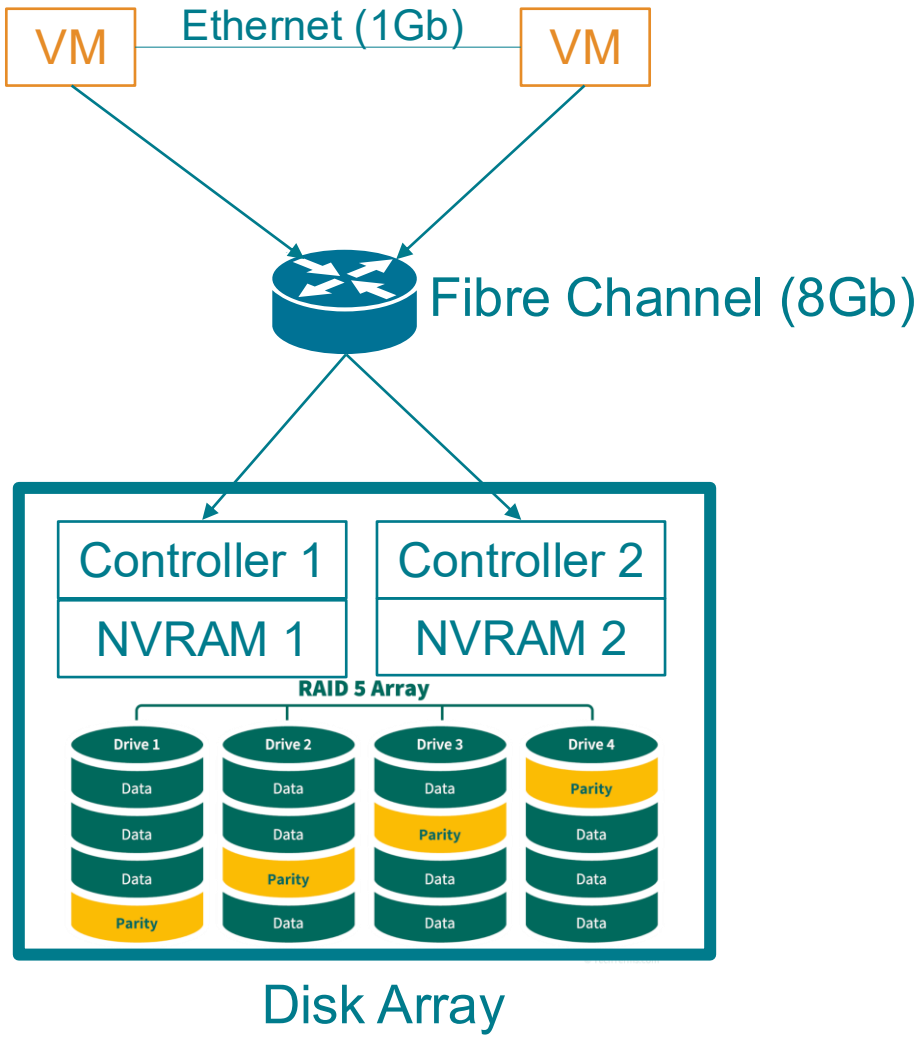
Principles of Developing System Software

- Takes a long time to develop & stabilize
  - Long term maintainability is important
  - Some over-engineering
  - Important to reuse
- New hardware changes:
  - Possible features
  - Bottlenecks
  - Architecture and design
- Performance is important
  - Higher consolidation ratio → saving money

BROADCOM

Disk Array – Storage Before vSAN

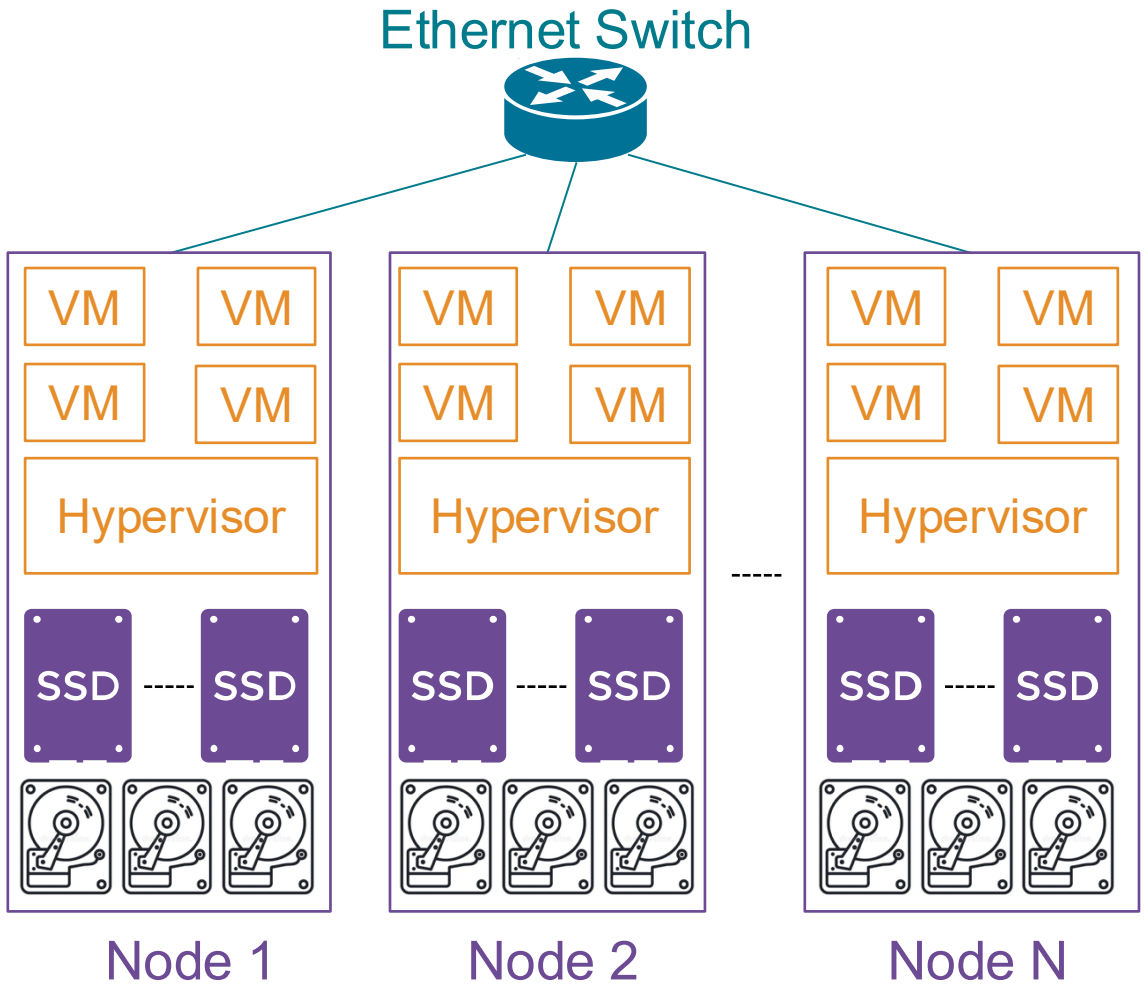
- External disk arrays
- Fault tolerance
  - Erasure coding or mirroring
  - Dual controllers
- Custom storage network
  - Fibre Channel
  - SAS
- Customized box
  - NVRAM
  - Dual controllers
- Expensive
- Hard to scale beyond one box



BROADCOM

What is vSAN and Why It's Viable?

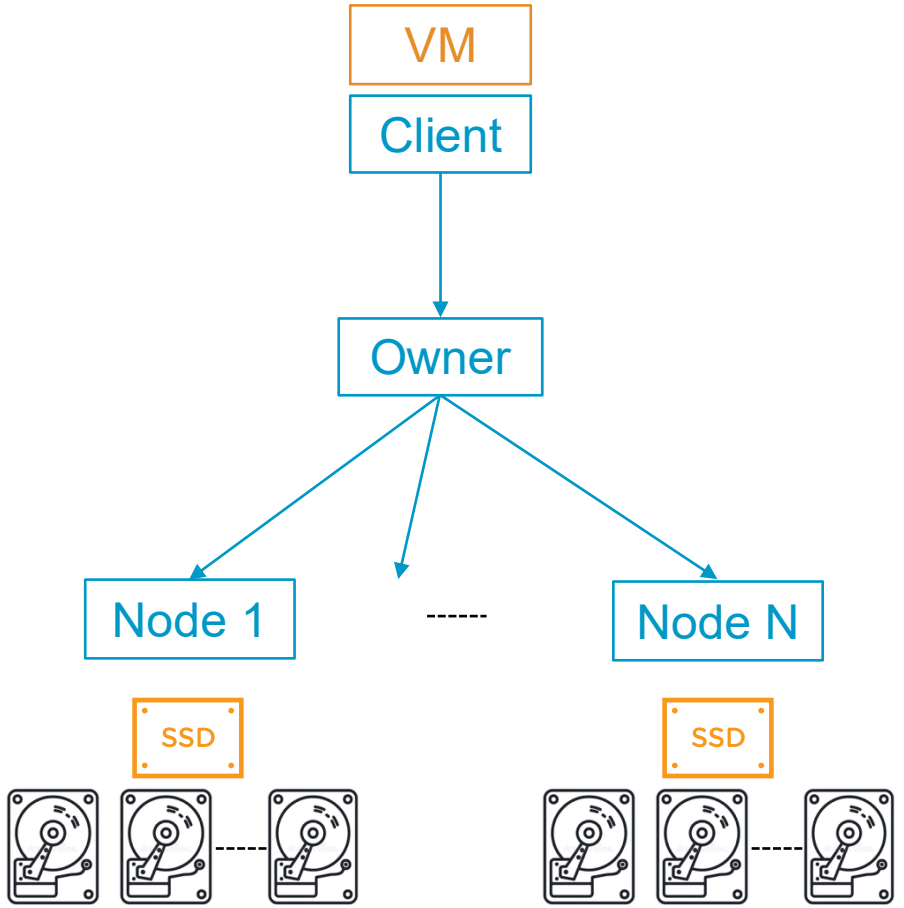
- Software-defined storage
  - Turns locally attached storage into a virtual disk array
  - Distributed erasure coding
- Benefits
  - Commodity hardware (cheap)
  - Scale-out architecture
- What hardware change made vSAN possible? In 2010
  - Fast SSDs (SATA SSD) – replaces NVRAM
  - Fast and cheap Ethernet – replaces FC
    - 10GbE caught up with Fibre Channel



BROADCOM

vSAN OSA (Original Storage Architecture) Architecture

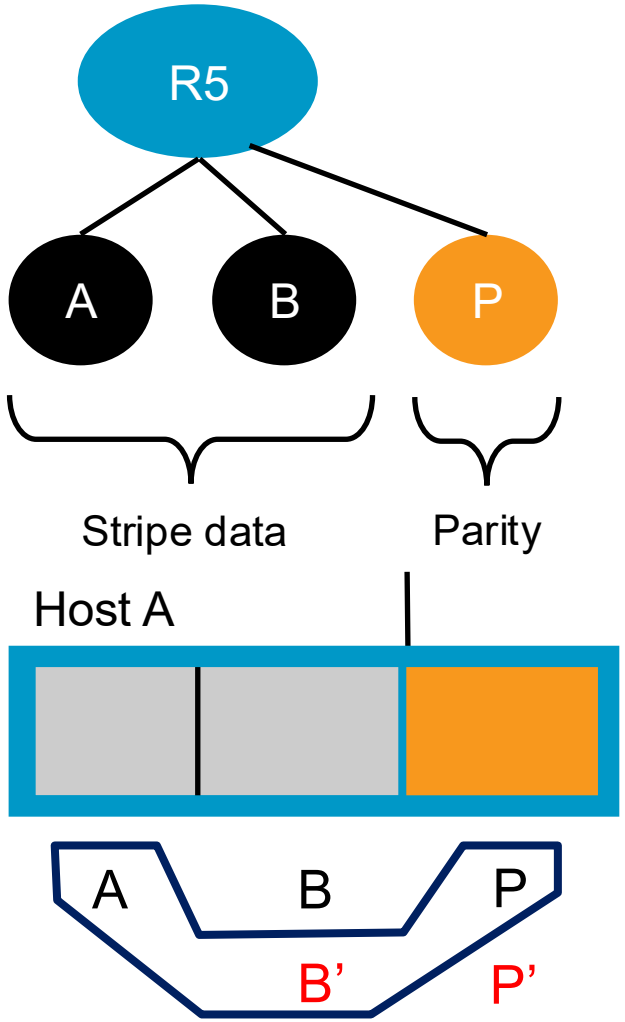
- Scale-out
  - Nodes are connected via Ethernet
- Erasure coding cross hosts
- Local file system:
  - SSD Performance Tier
  - HDD Capacity Tier



BROADCOM

Distributed Erasure Code

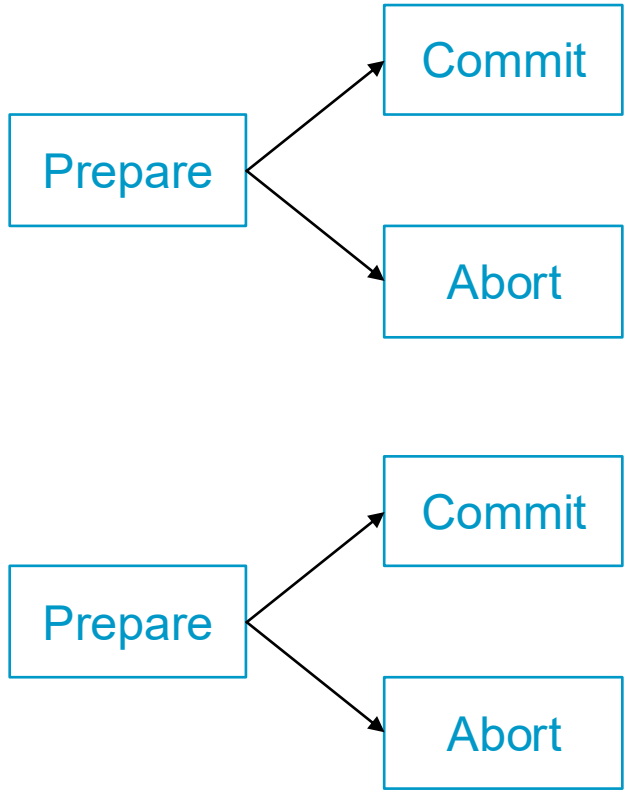
- E.g., 2 + 1 RAID-5 or 4 + 2 RAID-6
- Parity: P = A ⊕ B; P’ = A ⊕ B’;
- The “Write Hole” problem
  - Update multiple blocks on different hosts atomically
- **Question**: if it crashes before P’ is written, what happens?
- Data corruption for A & B’
- **Question**: How to solve the write hole problem?
- Distributed transaction (two-phase commit)



BROADCOM

Two-phase Commit

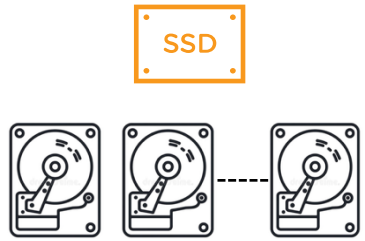
- Algorithm:
  - Ask all parities to “Prepare”
  - If any party can’t Prepare: Abort all
  - If all parties can Prepare: Commit all
- **Question**: How to reduce network round trips?
  - Use “presumed commit”
- Happy path
  - Waits for all hosts to prepare
  - ACK the write to the client
  - Batch commit all txn < N later

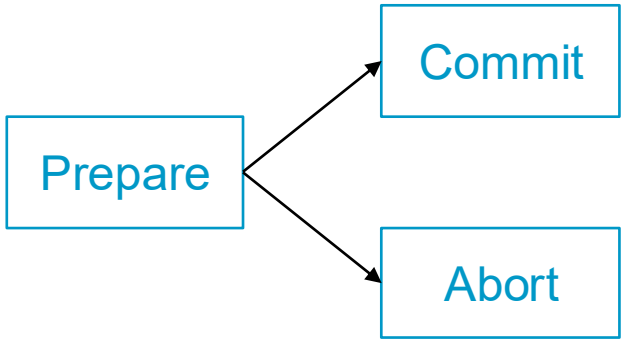


BROADCOM

The Local File System

- Manages a disk group:
  - 1 SSD and N HDDs
- Basic features
  - Large files
  - Flat namespace (no directories, no file names)
- Must support 2PC (Prepare, Commit, Abort)
- **Question**: can “Prepare” cache data in memory only?
- Prepare must persist data
  - Can commit after prepare even after a power loss
- **Question**: can conventional file systems, such as ext4 or NTFS, be used to support 2PC?
- No, because “Prepare” can’t overwrite old data
  - Can’t abort if old data is overwritten

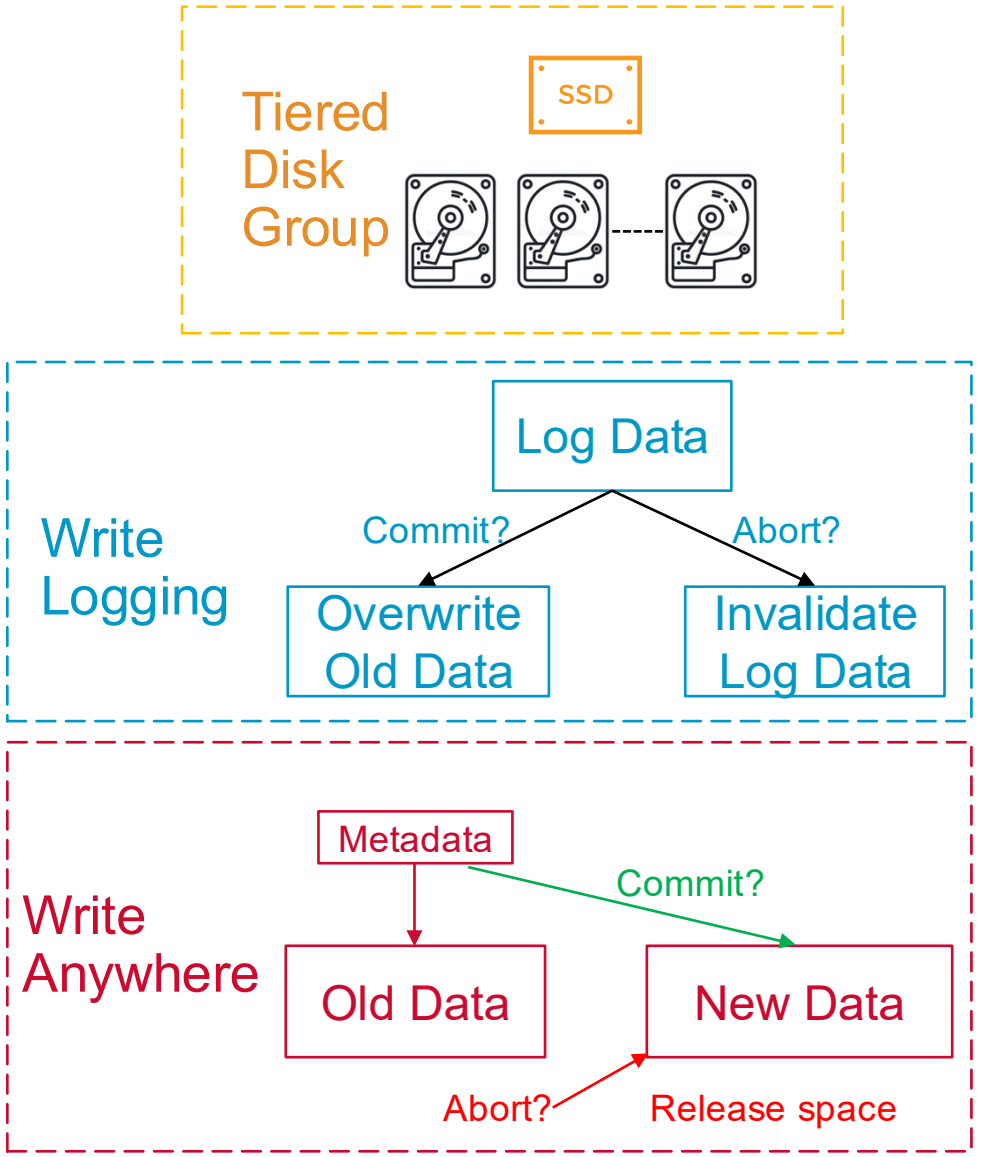




BROADCOM

Two-phase Commit in File System

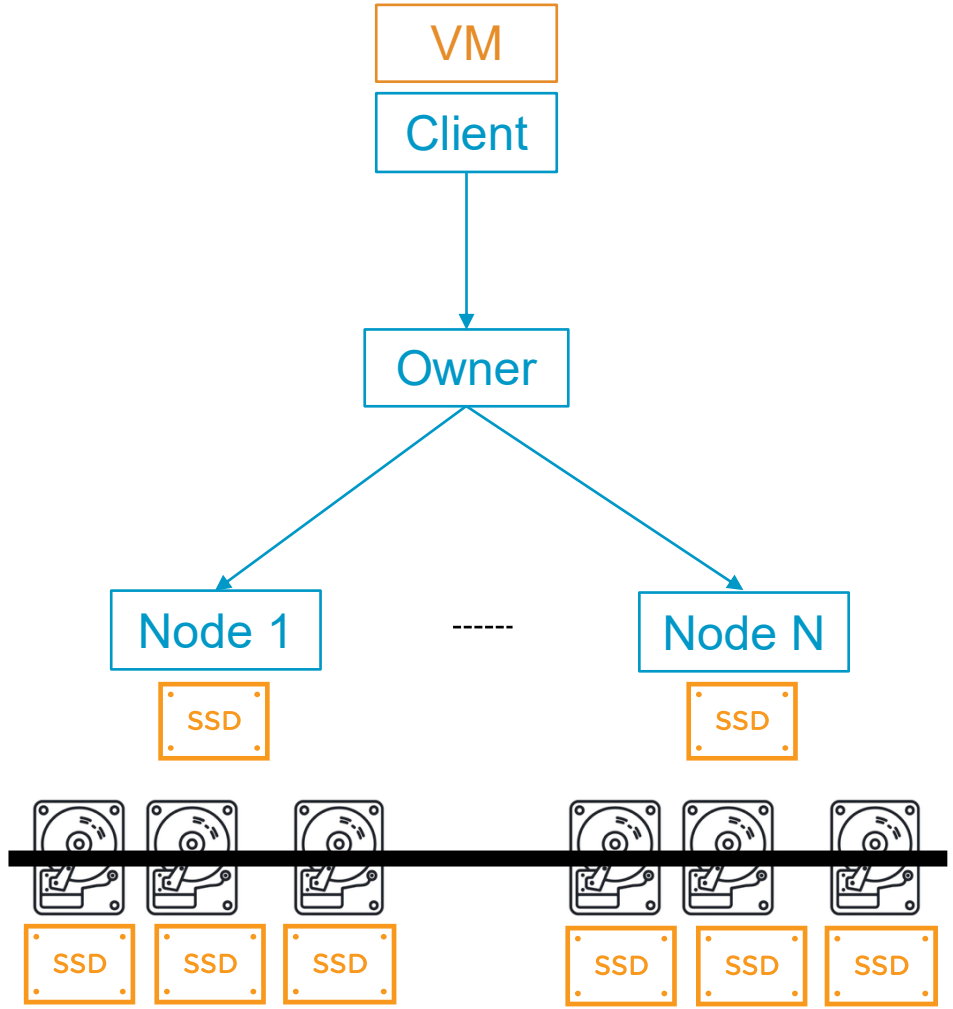
- File system has two tier: SSD and HDD
- Support Prepare, Commit or Abort
- Two choices
- Write logging
  - Log the prepare, overwrite later
  - Write twice
- Write anywhere
  - Write to new location, adjust metadata and delete old data later
  - Write once
- **Question**: which one works better?
- vSAN OSA uses write logging
  - Log to SSD and flush to HDD after commit
  - SSD also serves as a “Read Cache”



BROADCOM

Evolving vSAN OSA

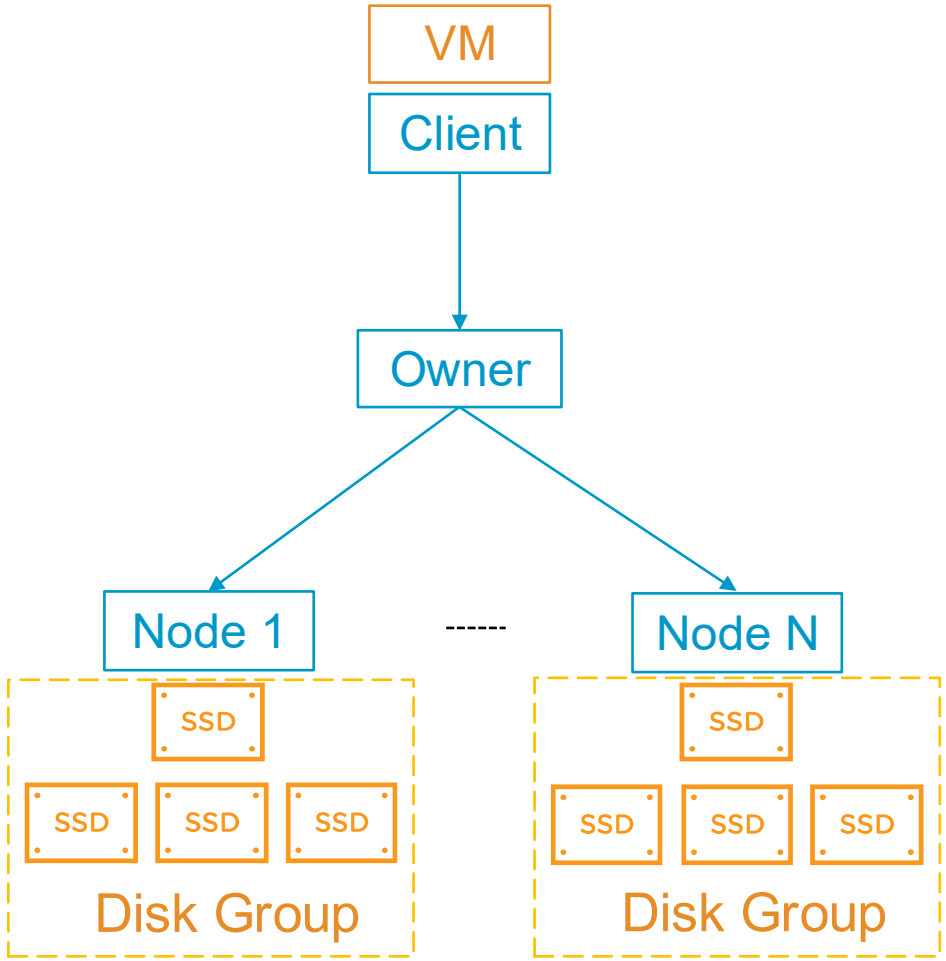
- OSA was a very fast HDD disk array
- Until
- SSDs become bigger, faster, and cheaper
  - SSD latency is more predictable than HDD latency
- Replaced HDD with SSD
- **Question**: What are potential performance problems of this change?
- Hints:
  - Performance goal: close to hardware limit
  - SSDs have similar performance



BROADCOM

vSAN OSA Problems

- New bottleneck: Performance-tier SSD
  - On reads
  - “Read Cache” on performance-tier is slow
  - **Question**: how to solve this?
  - Disable the Read Cache
- New bottleneck: Performance-tier SSD
  - On large writes
  - File system’s “write logging” is slow
- New bottleneck: CPU
  - On small I/Os
  - Lack a multi-threaded architecture
- Old problems:
  - No good snapshots nor compression
  - Small writes are slow due to **partial stripe writes** on RAID-5 and RAID-6

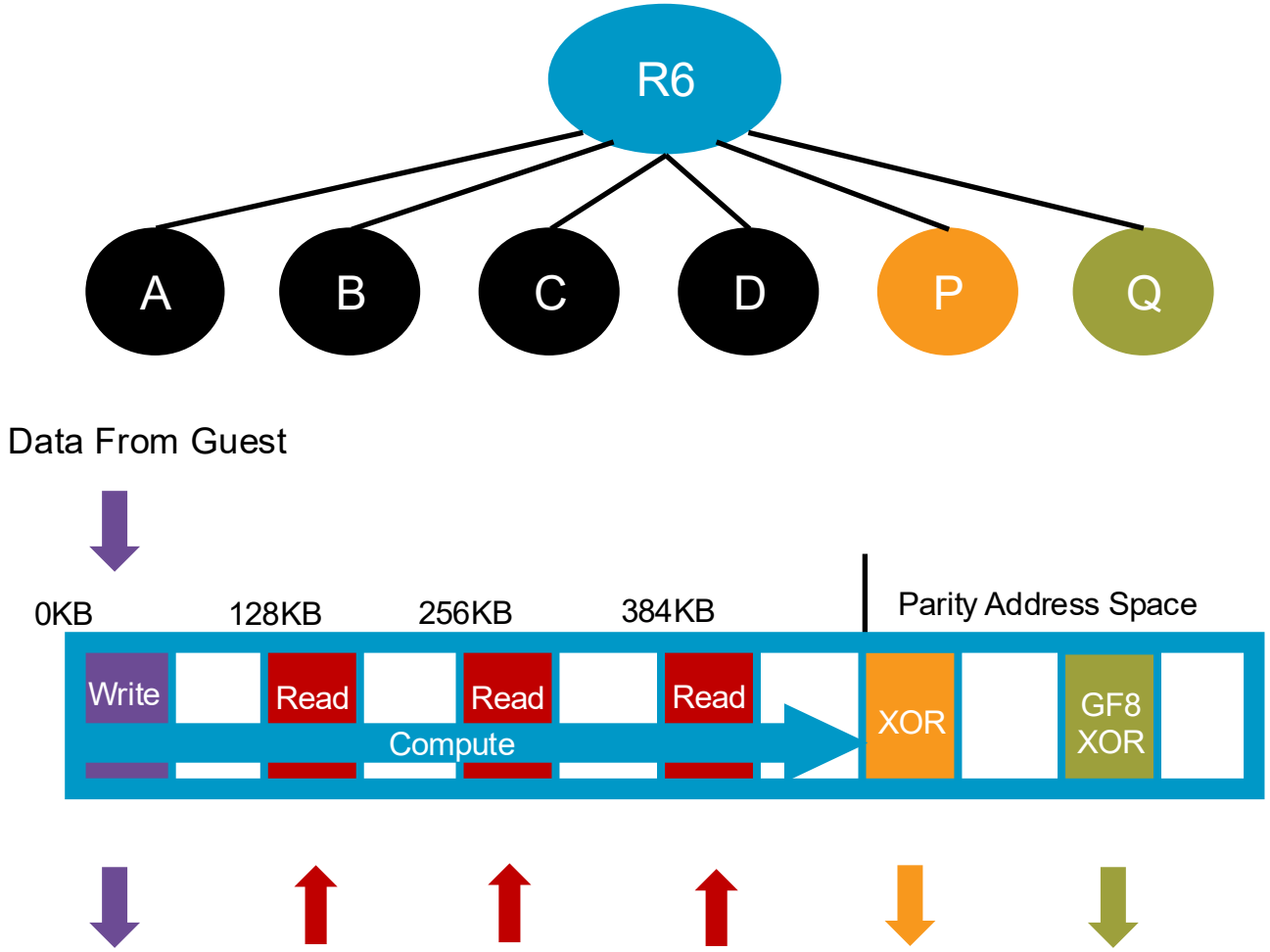


BROADCOM

Partial Stripe Write

The Difficulty With RAID6

- Write A:
  - Read B C D, verify checksum
  - Calculate P and Q
  - Write A, P and Q
  - 1 Write ➔ 3 read, 3 writes
    - 6x amplification
    - 2 network round trips

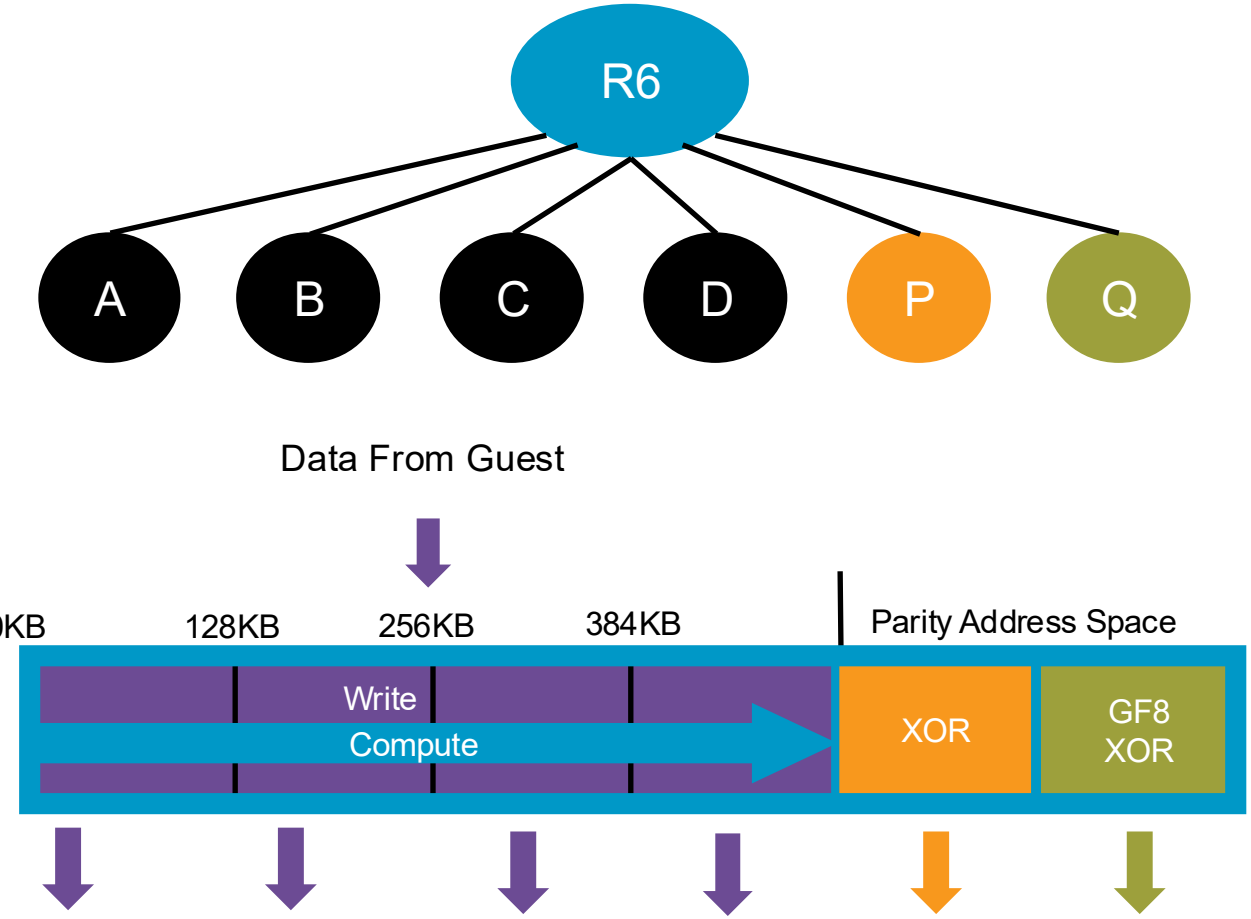


BROADCOM

Full Stripe Write

Efficient Writes With RAID6

- Write A-D
  - Calculate P and Q
  - Write A-D, P and Q
  - 4 writes ➔ 6 writes
    - 1.5x amplification
      - 4x less IO than partial stripe
    - 1 network round trip
      - 2x less latency than partial stripe

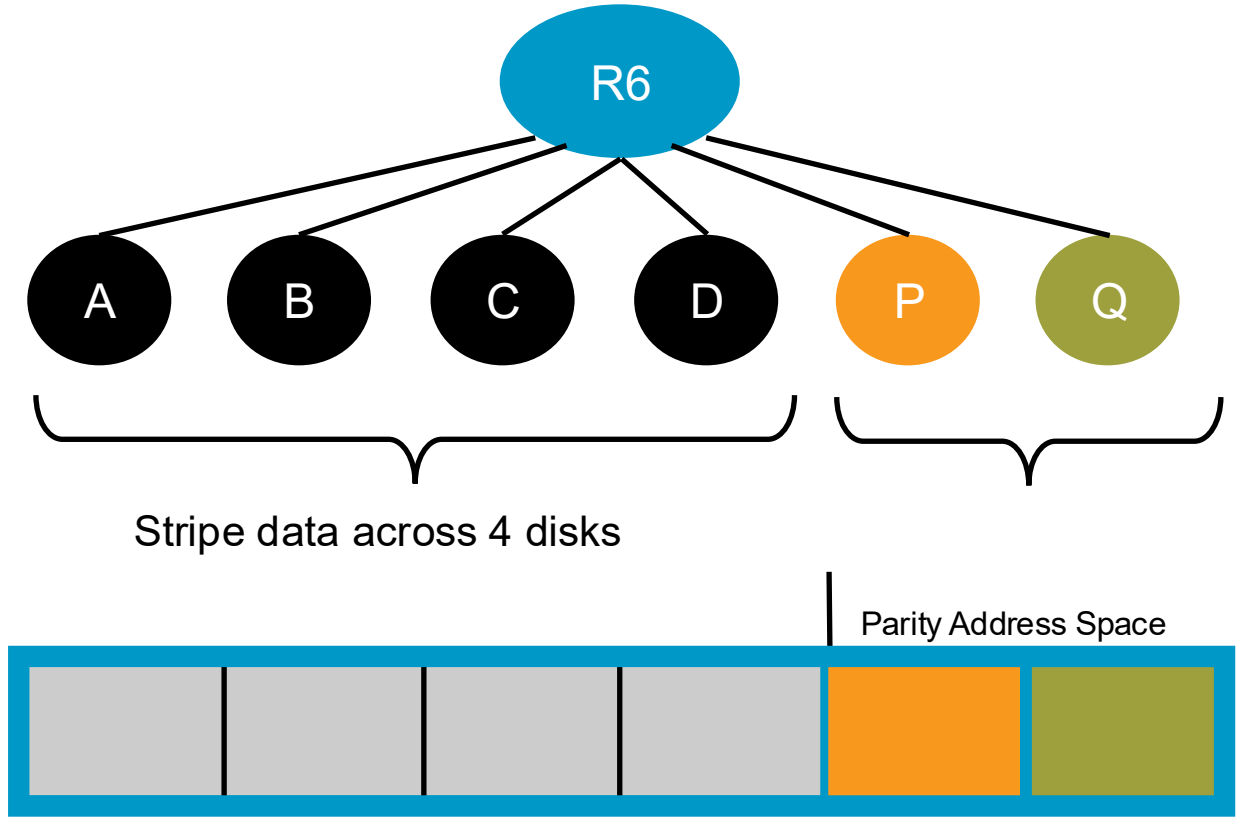


Large writes are faster

BROADCOM

How to Improve Small Writes?

- Large writes are full stripe writes
- Small writes require partial stripe writes
- Customer workloads have small writes
- **Question**: how to improve small write performance?
- Hint: how to turn partial stripe writes to full stripe writes?



BROADCOM

vSAN ESA (Express Storage Architecture)

Logs to the rescue

What can aggregate writes into big chunks?

Log-structured File System (LFS):

- Always writes to new location
- Garbage collect later

Key idea: add LFS on top of vSAN OSA

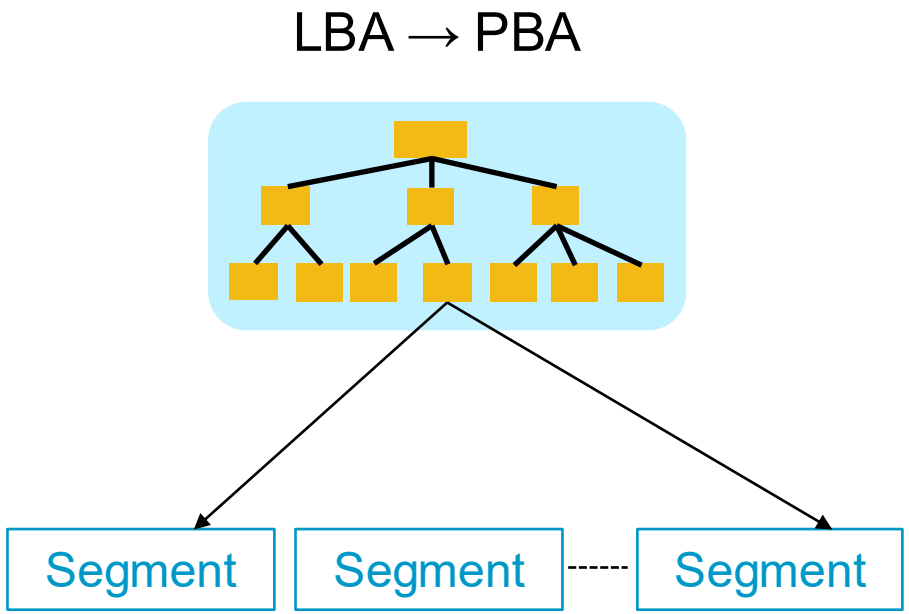
Benefits of Log-Structured Write

- Full-stripe write to RAID-5/6
- Snapshot
- Compression
- QLC (large writes)
- Reduce file system metadata size
- Save CPU on encryption & compression
- Reuse vSAN OSA

BROADCOM

Log-structured File System

- Only need one file for a virtual block device
- One metadata map (B+ Tree) LBA → PBA
- When writing data
  - Write to new address
  - Update LBA to point to new PBA
- **Question**: how to update B+ Tree?
- Need another level of indirection
  - Virtual Address Table (VAT)
  - Store VAT in checkpoint, load on bootstrap
- Problems:
  - VAT doesn't fit in memory
  - Complicated (too many levels of indirections)
- Solution: hybrid LFS

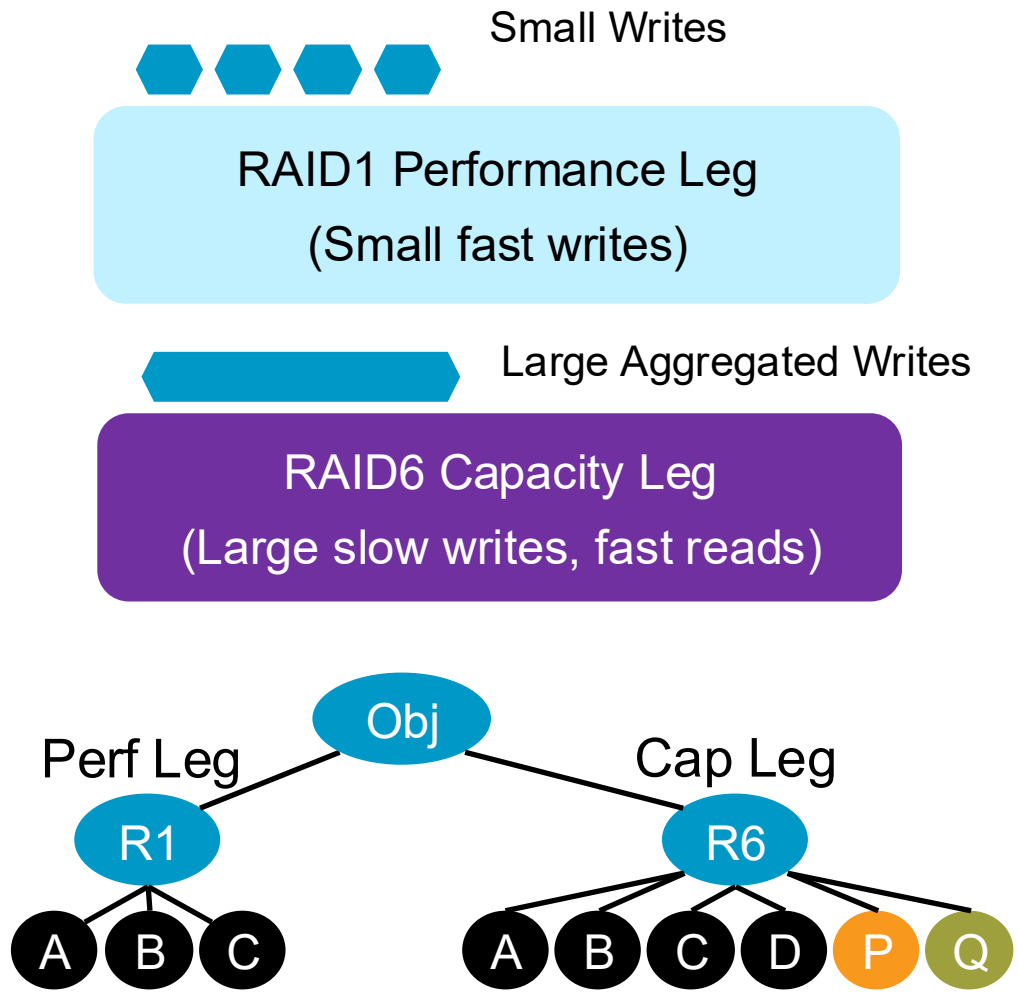


BROADCOM

Hybrid Log-structured File System

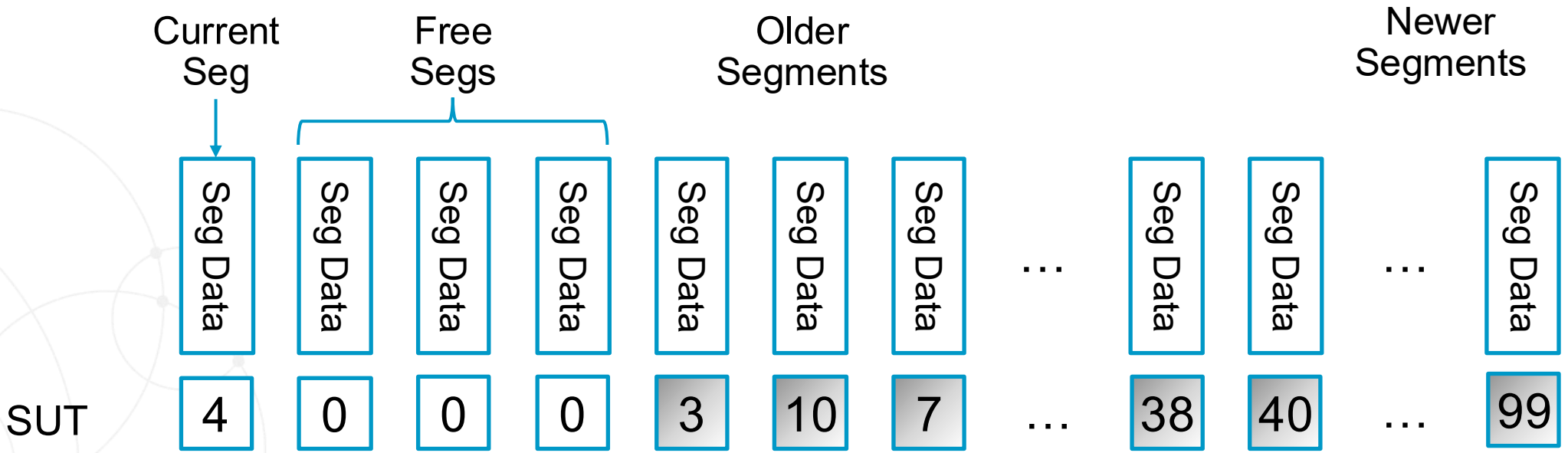
Leverage Two vSAN Policies

- Each LFS has:
  - Performance log: RAID-1 3-way mirror
    - Small writes don't cause read-modify-write
    - Stores Write-Ahead Log, B-tree pages, other metadata
  - Capacity leg: 4 + 2 RAID-6
    - Only large writes are fast
- Write small, then write big
  - Log data and metadata to aggregate
  - Data are written **twice**, but still much faster
- **Question**: why does the performance leg use a 3-way mirror, instead of a 2-way mirror?
- Both legs must tolerate 2 node failures



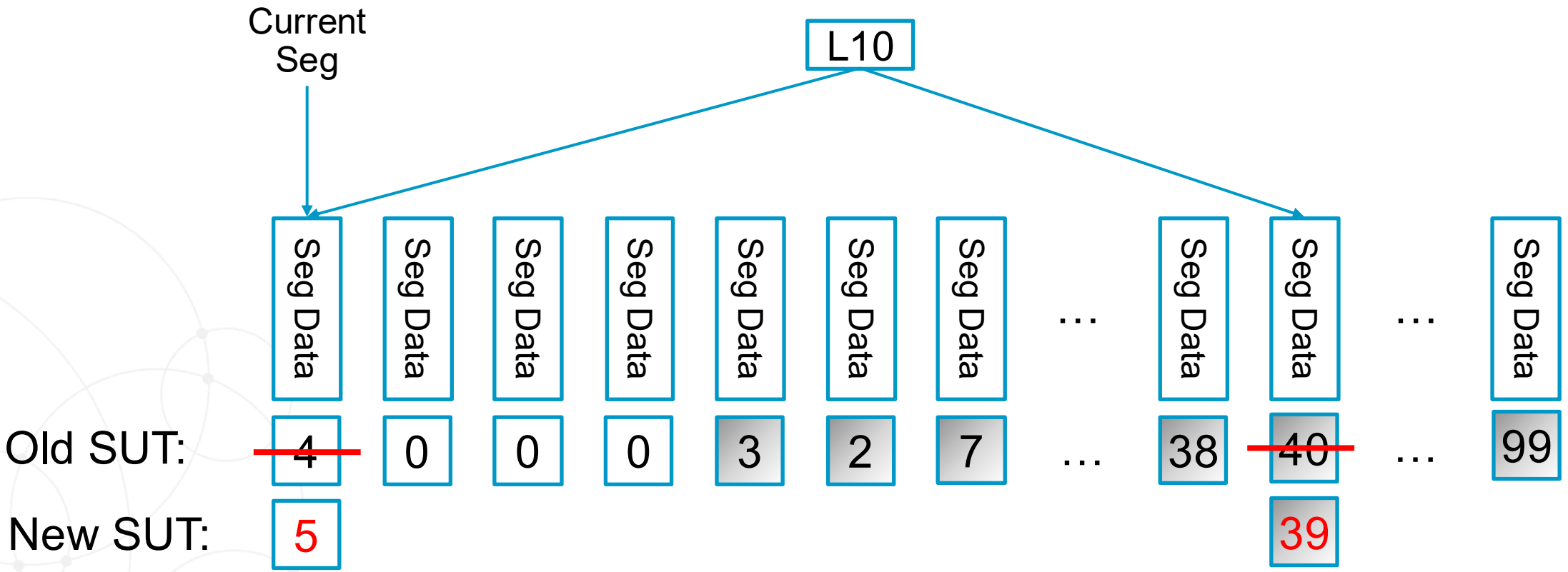
Segments and Segment Usage Table

- Free space in segments are managed by an on-disk array called “Segment Usage Table”
- Each entry in the table has:
  - numLive: number of data blocks still live



Segment Usage Table Updates

Write to Logical 10

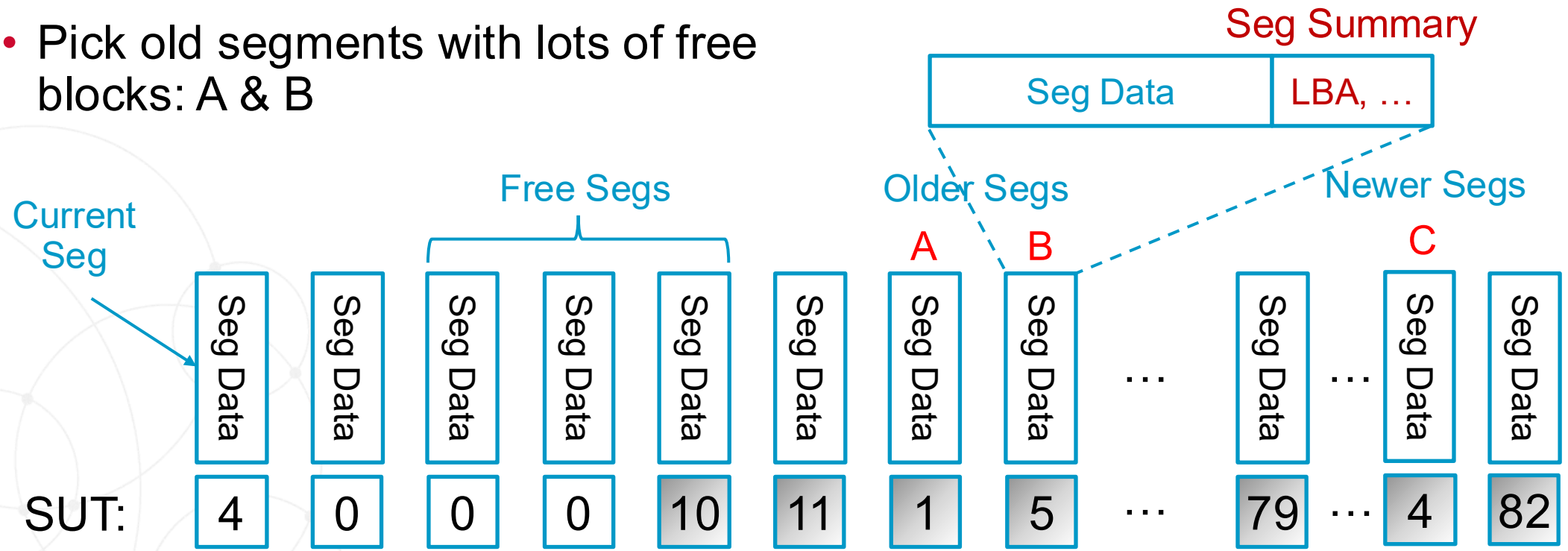


BROADCOM

Segment Cleaning (Garbage Collection)

- Check Segment Usage Table, find best segments to clean:
- **Question**: Which 2 should be cleaned: A, B, C?
- Pick old segments with lots of free blocks: A & B

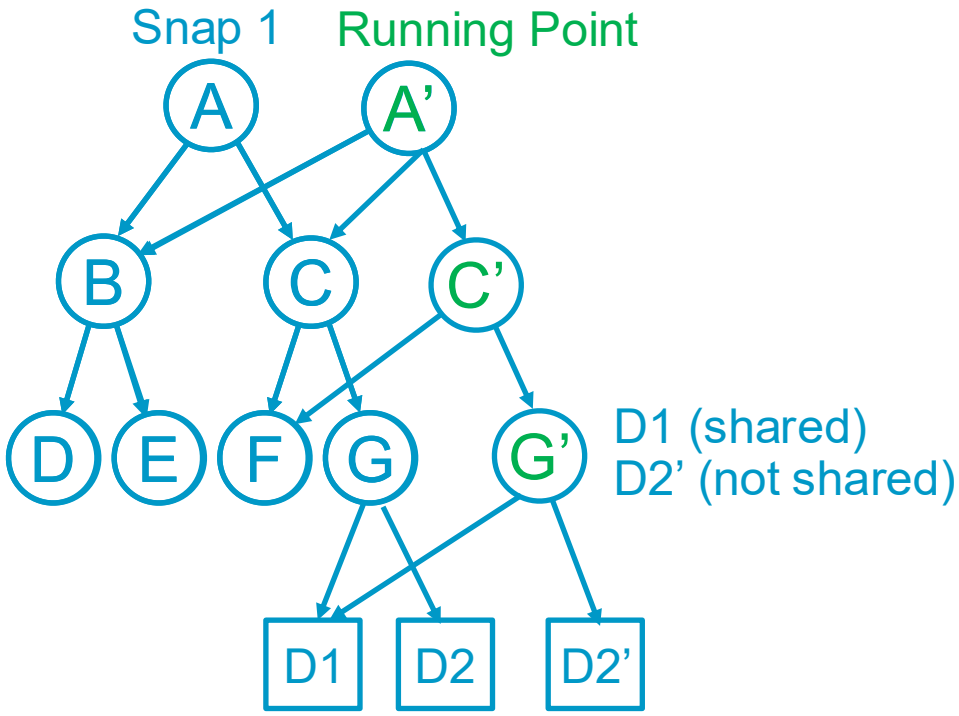
- How to clean a segment:
  - Read a segment
  - Use "Segment Summary" to find live blocks
  - Write live blocks to new location



BROADCOM

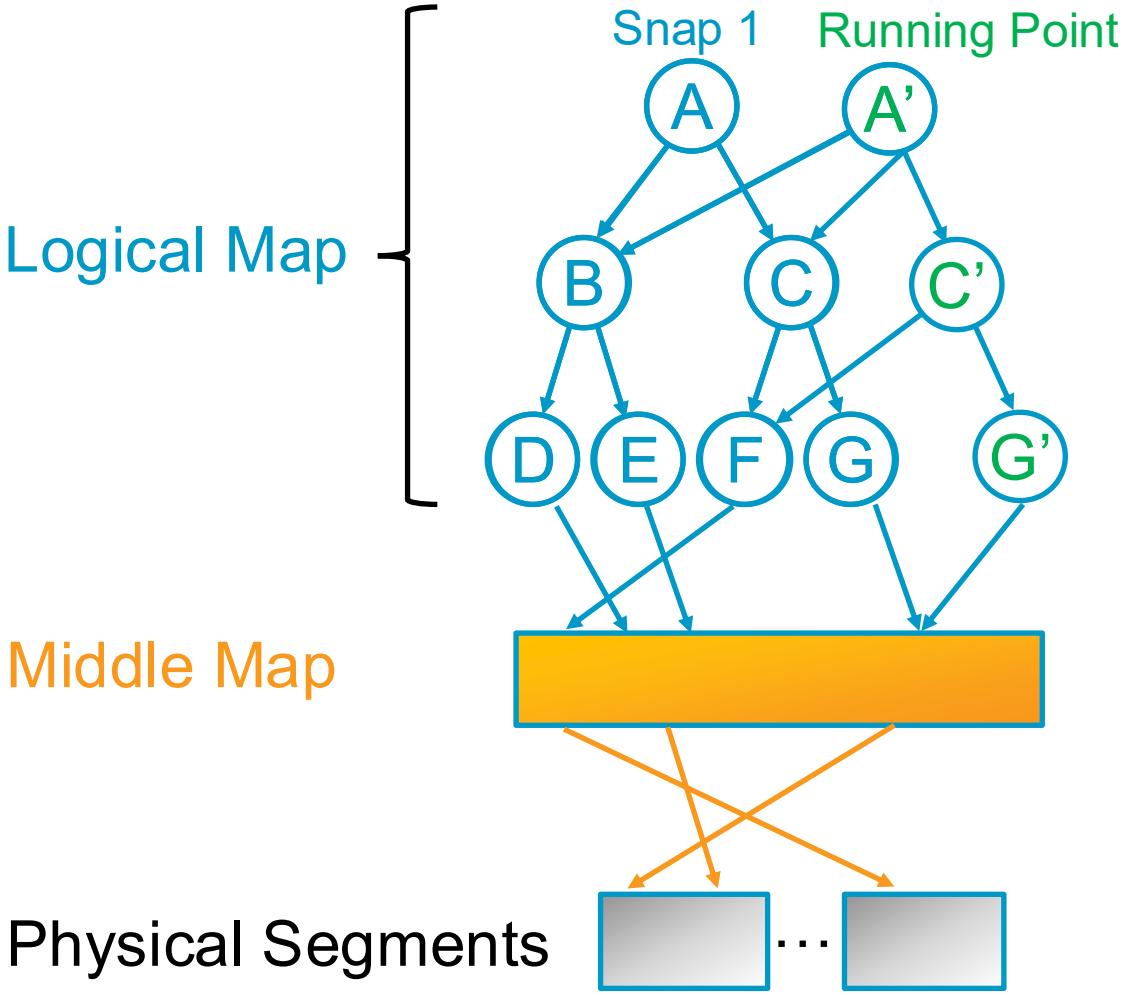
Snapshot Design – Snapshots and Clones

- Snapshot: keep old data & copy metadata
  - Keep old data: solved by LFS
  - Copy metadata: Copy-On-Write B-tree
- Snapshot creation
  - Clone a B-tree
- Snapshot deletion
  - Compare the previous and next B-trees to identify shared pages
- **Question**: How many LBAs point to a PBA and how does segment cleaner move blocks?
- **Question**: how to limit metadata updates during block moving?



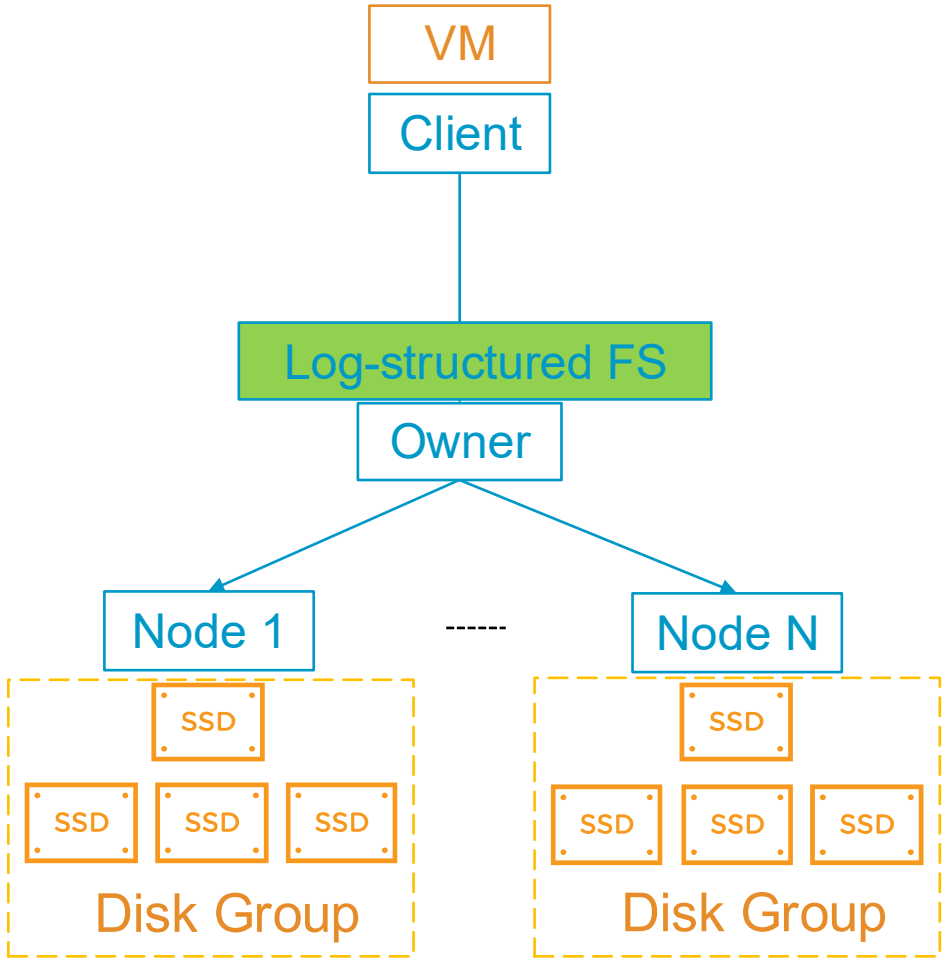
Snapshot Design – COW B-tree with Middle Map

- Logical Map B-tree
  - Key: LBA (Logical Block Address)
  - Val: MBA (Middle Block Address), NumBlocks
  - Example: L0 → M10, N20
  - LBA → MBA is N:1
- Middle Map B-tree
  - Key: MBA
  - Val: PBA (Physical Block Address), NumBlocks
  - Example: M10 → P100, N2
  - MBA → PBA is 1:1
- **Question**: now how does segment cleaner move a block?



vSAN ESA Complete Data Path

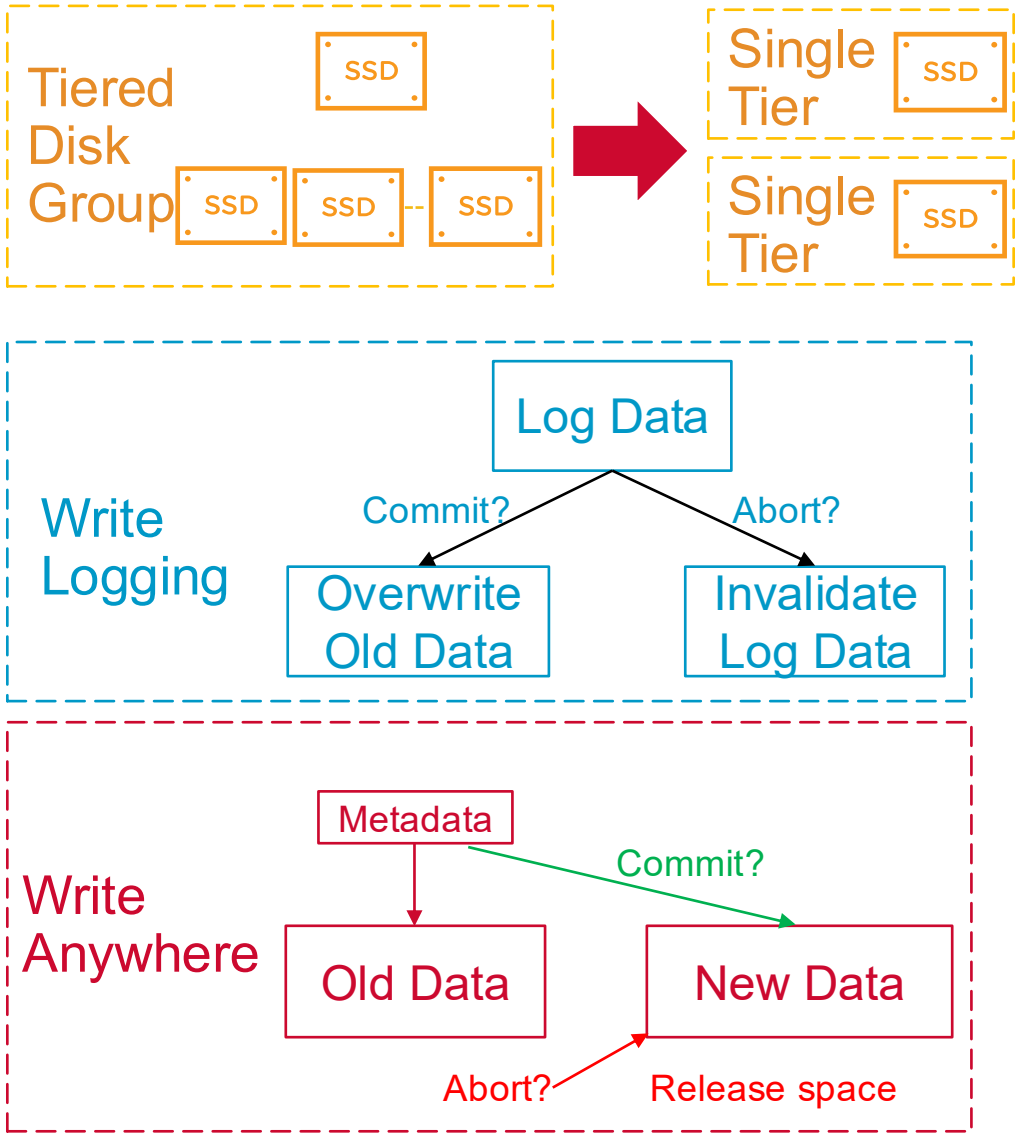
- New Log-structured File System
  - Full-stripe write
  - Compression
  - Snapshot
  - …
- New local file system to replace old disk group
  - Fix bottlenecks of performance tier SSD and CPU



BROADCOM

New Local File System

- Remove “Disk Group” concept
- Single Tier design – each SSD is an independent file system
- **Question**: How to implement 2PC, Write Logging, or Write Anywhere?
- Write Anywhere avoids double write



BROADCOM

vSAN ESA Execution and Performance Results

- Large team effort to discuss, design, and implement
- Performance tuning is an ongoing task
- Achieved much better performance
- Snapshots are > 100x faster

Workload	Throughput	CPU per IO
70/30 8k RW	5.2x	2.6x drop
8k rand read	2.6x	6% increase
Large write	3.1	1.33x drop

vSAN ESA vs. OSA
on identical hardware

BROADCOM

Conclusions and Q&A

- Principles of designing distributed storage systems
  - Design safe, fast system that fit the hardware
  - Some over-engineering to fit future hardware
  - Reuse existing system when possible
- vSAN OSA was designed for HDD
  - Extended its life into SSD
- vSAN ESA is designed for NVMe
  - Reused most of OSA
  - Added a log-structured file system on OSA
  - Achieved 5x performance on identical hardware

Thanks to Eric Knauft for some slides I've borrowed from him.

BROADCOM

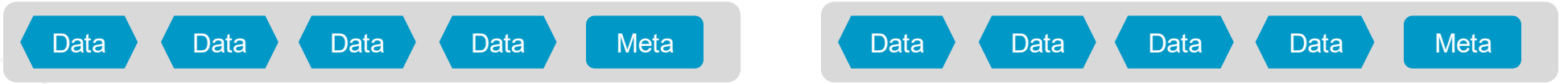
The Importance of Amortization

Adding layers of metadata can cause extra IO, led to worse performance

Offset this by amortizing the cost

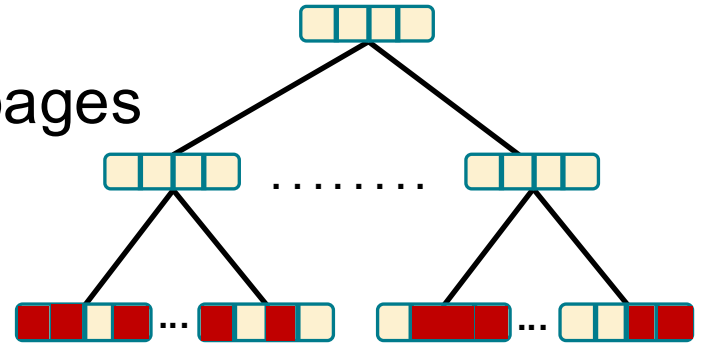
- One large IO operation is more efficient than many small IO operations

Log IOs: write all the in-flight IOs and their metadata into a single IO



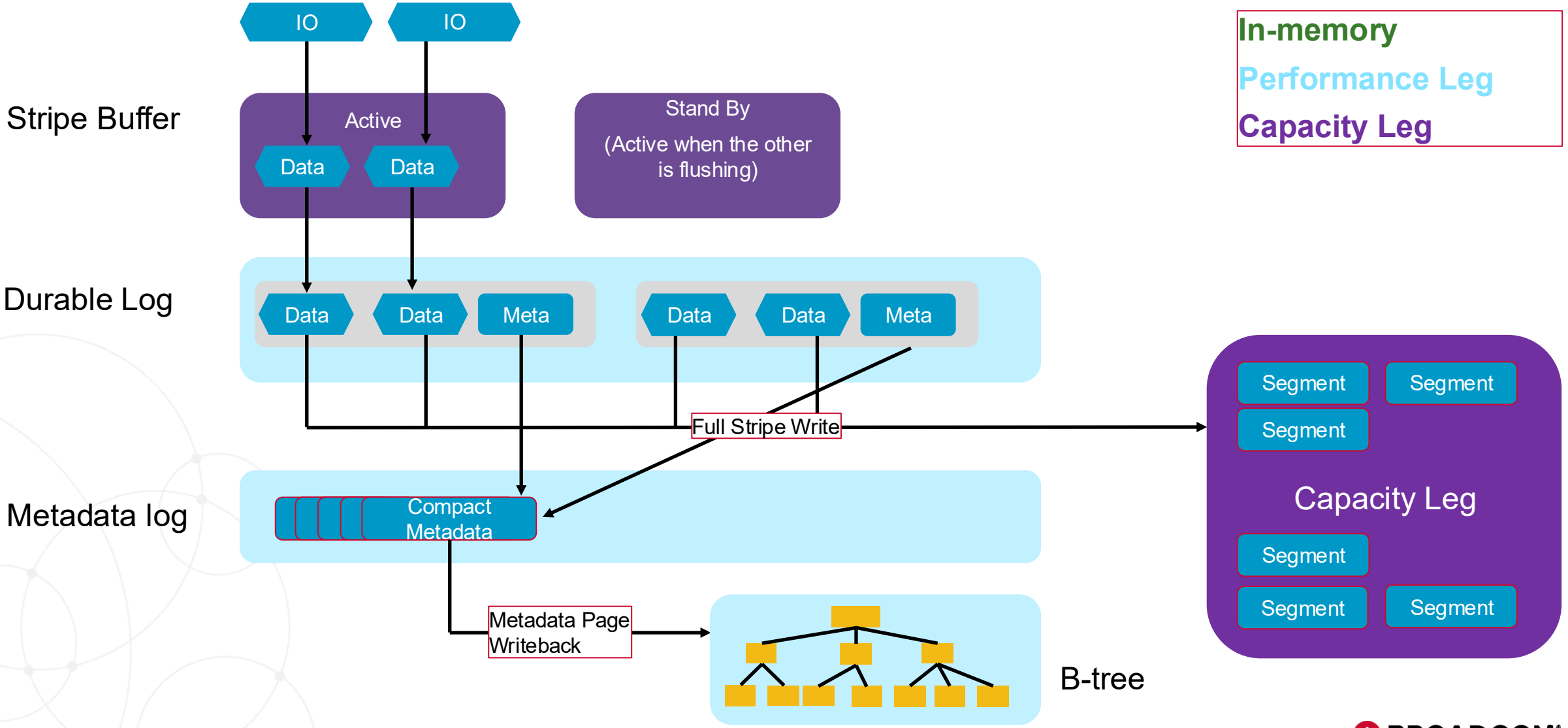
Get lots of hits in the same B-tree pages

- Requires keeping things in memory for a while



2.5 updates per page

IO Flow



Segment Cleaning with Segment Summary

- Segment summary has the LBAs of every block in the segment
- Assume segment size is 100 blocks
- PBA of segment 1 is P100 – P199
- Segment summary of segment 1: L80, L90
- Current logical map values:
  - L80 ➔ P530
  - L90 ➔ P120
- Which block is live?

Cleaning Procedure:

- Read segment 1
- Find out L90 is live
- Write L90 to new segment 3 with PBA 300
- Update logical map: L90 ➔ P300

