FlexGroup Volumes:

A Distributed WAFL File System

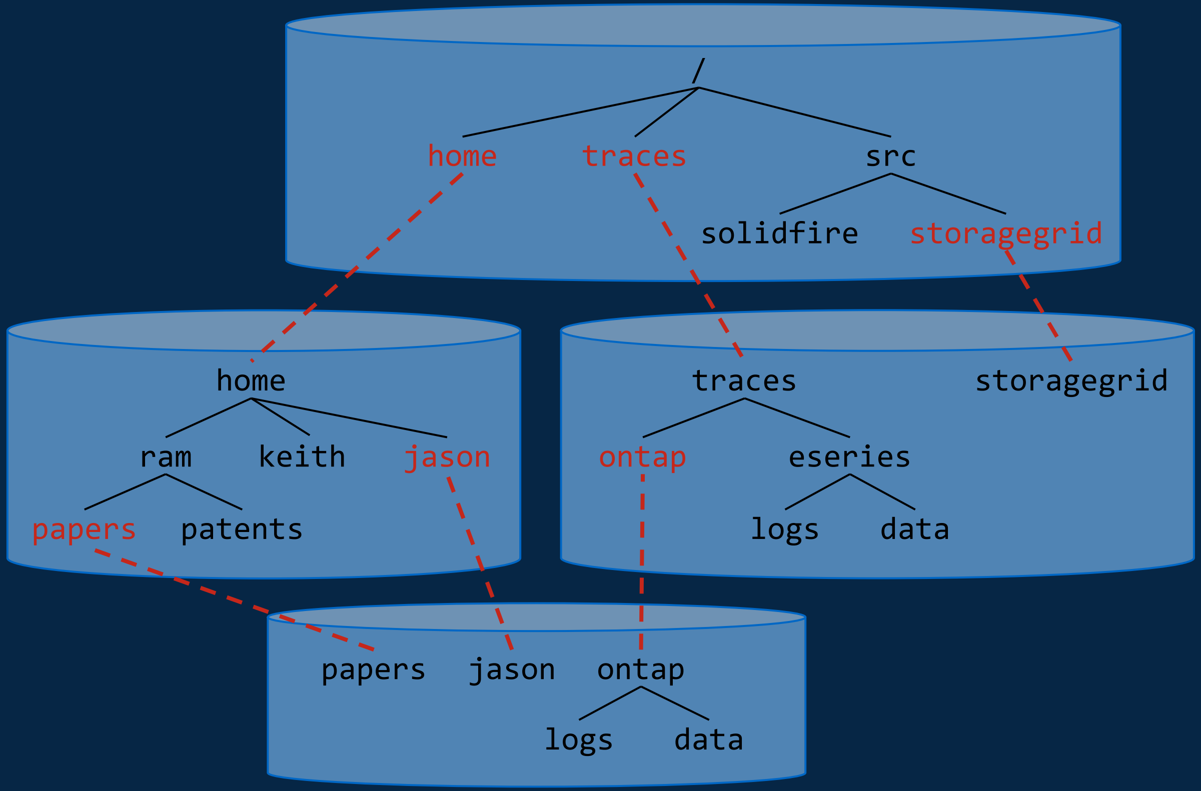
Ram Kesavan, Google; ***Jason Hennessey***, Richard Jernigan,
Peter Macko, Keith A. Smith, Daniel Tennant, and Bharadwaj V. R., NetApp

30 second overview

- Wanted an automatically balanced, simple performant distributed filesystem
- Reuse our existing filesystem technology
  - WAFL filesystem, ONTAP clusters
  - Highly reliable local nodes

30 second overview

- Wanted an automatically balanced, simple performant distributed filesystem
- Reuse our existing filesystem technology
  - WAFL filesystem, ONTAP clusters
  - Highly reliable local nodes
- FlexGroups: a distributed filesystem that seamlessly fuses WAFL volumes with automatic placement:
  - Remote links stich together filesystems from multiple nodes
  - Heuristics to keep them balanced

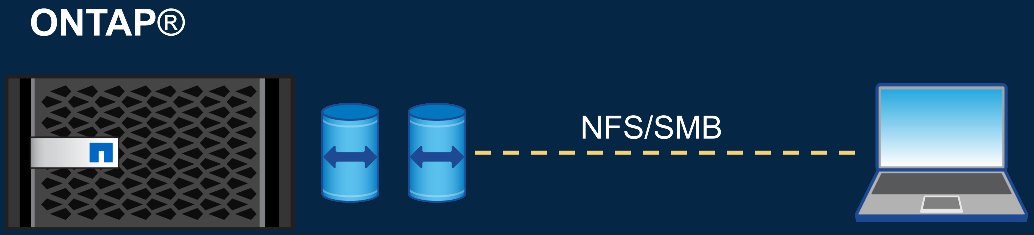


Agenda Slide

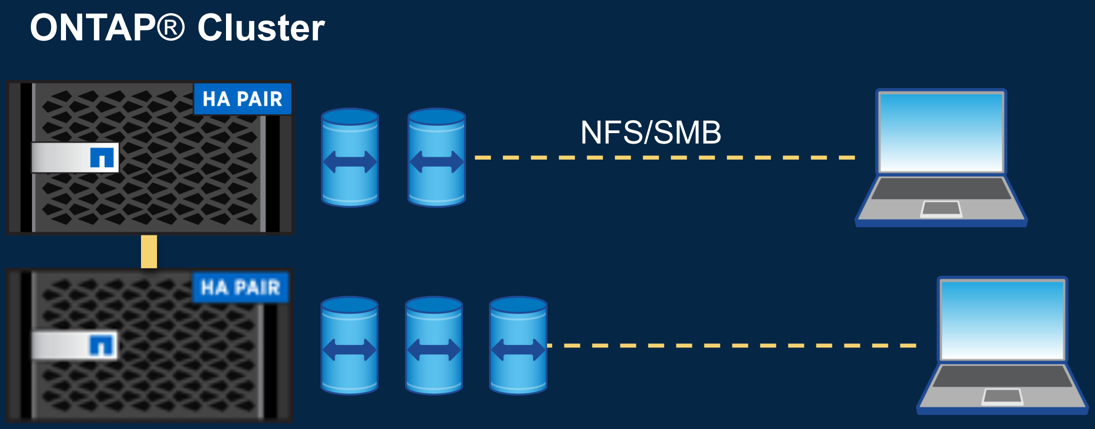
1) Introduction:
   - Background, Problems, Requirements
2) FlexGroups Design
3) Evaluation
   - Micro/macro benchmarks + customer experience
4) Conclusions + refs

Introduction

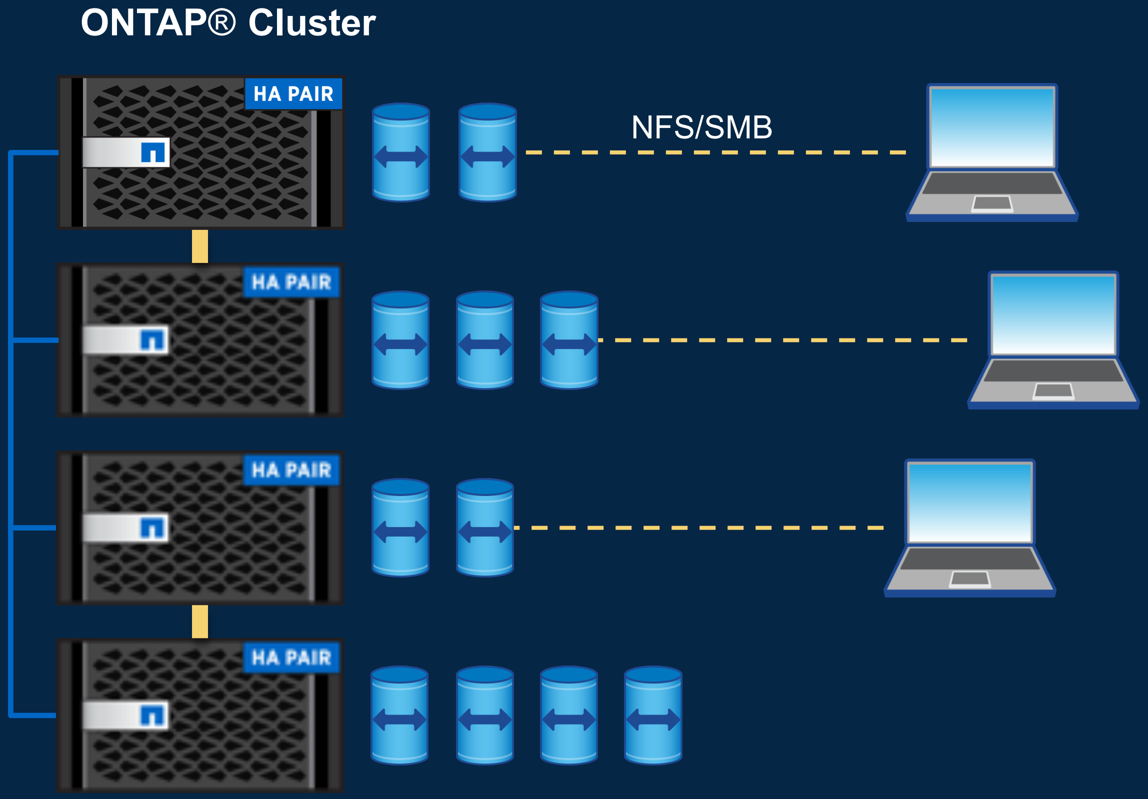
NetApp brief history



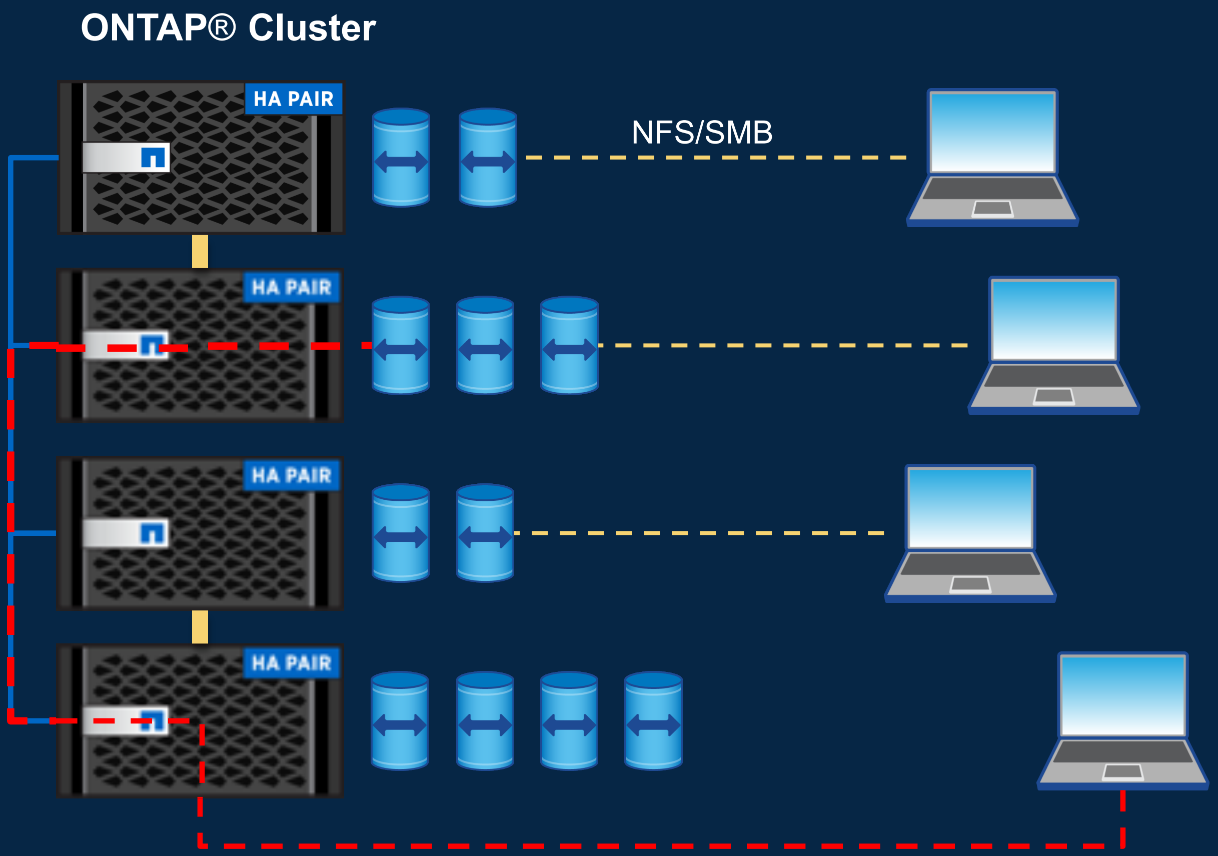
NetApp brief history



NetApp brief history



NetApp brief history



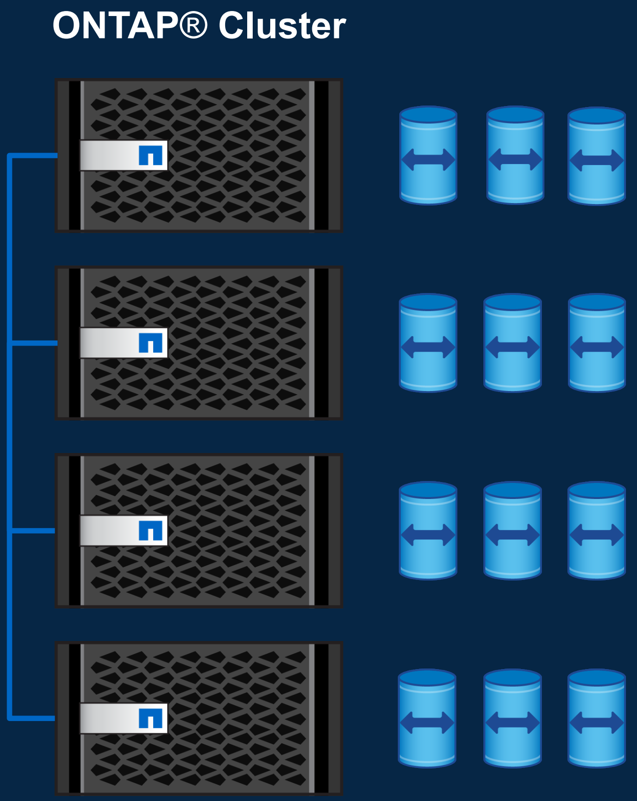
Problem

- Load balancing was manual
- Want to simplify/automate data placement
- Have fast, reliable, local technologies (like WAFL filesystem & ONTAP Clusters) with a lot of customer experience

Requirements

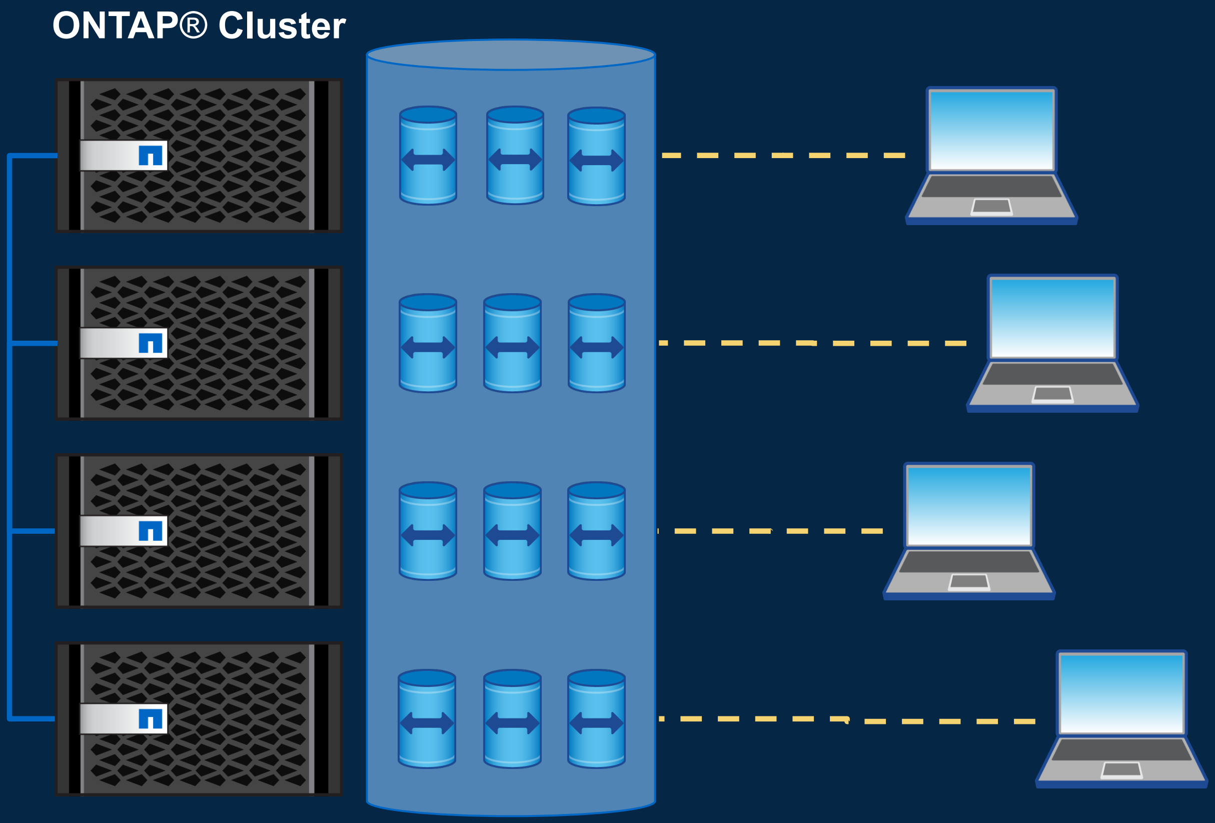
- Dynamic load balancing
- Ease of management
- Low latency
- Metadata scales with cluster

Solution: FlexGroups



- FlexGroup internally composed of automatically distributed individual WAFL volumes
- Simple for:
  - **admins**: automatic load balance
  - **clients**: single volume
- FlexGroups are implemented with:
  - Data linked across volumes with remote link
  - Heuristics at ingest balance across nodes

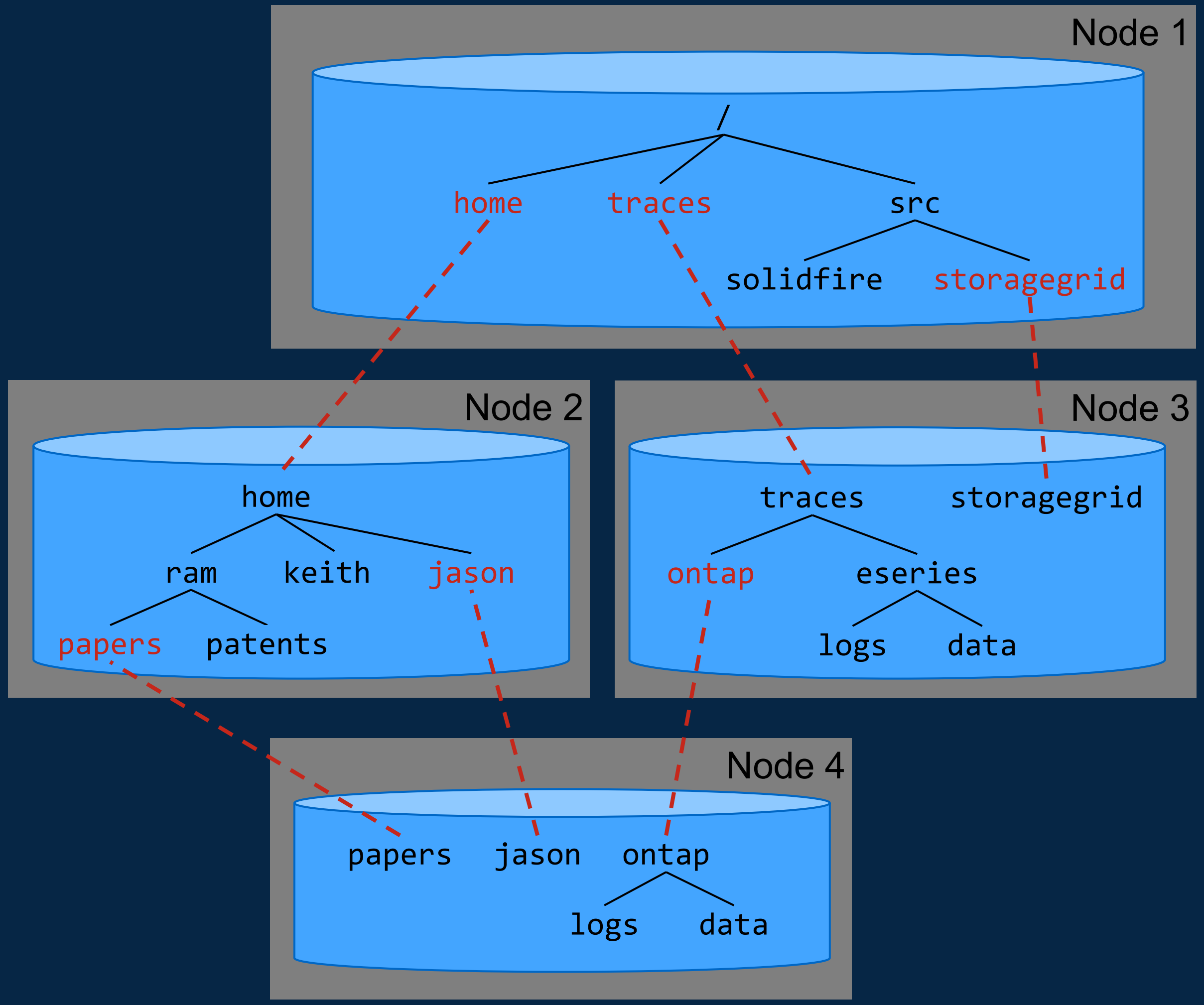
Solution: FlexGroups



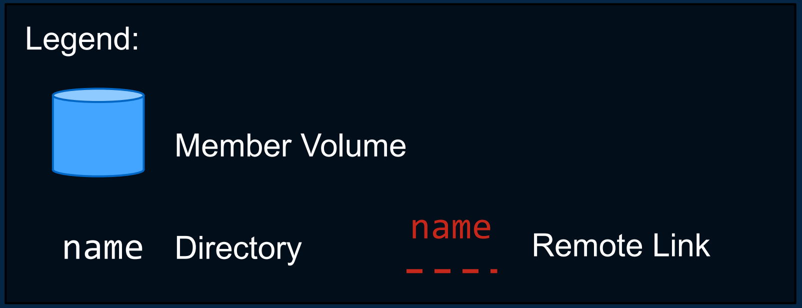
- FlexGroup internally composed of automatically distributed individual WAFL volumes
- Simple for:
  - **admins**: automatic load balance
  - **clients**: single volume
- FlexGroups are implemented with:
  - Data linked across volumes with remote link
  - Heuristics at ingest balance across nodes

Design of FlexGroups

Overview



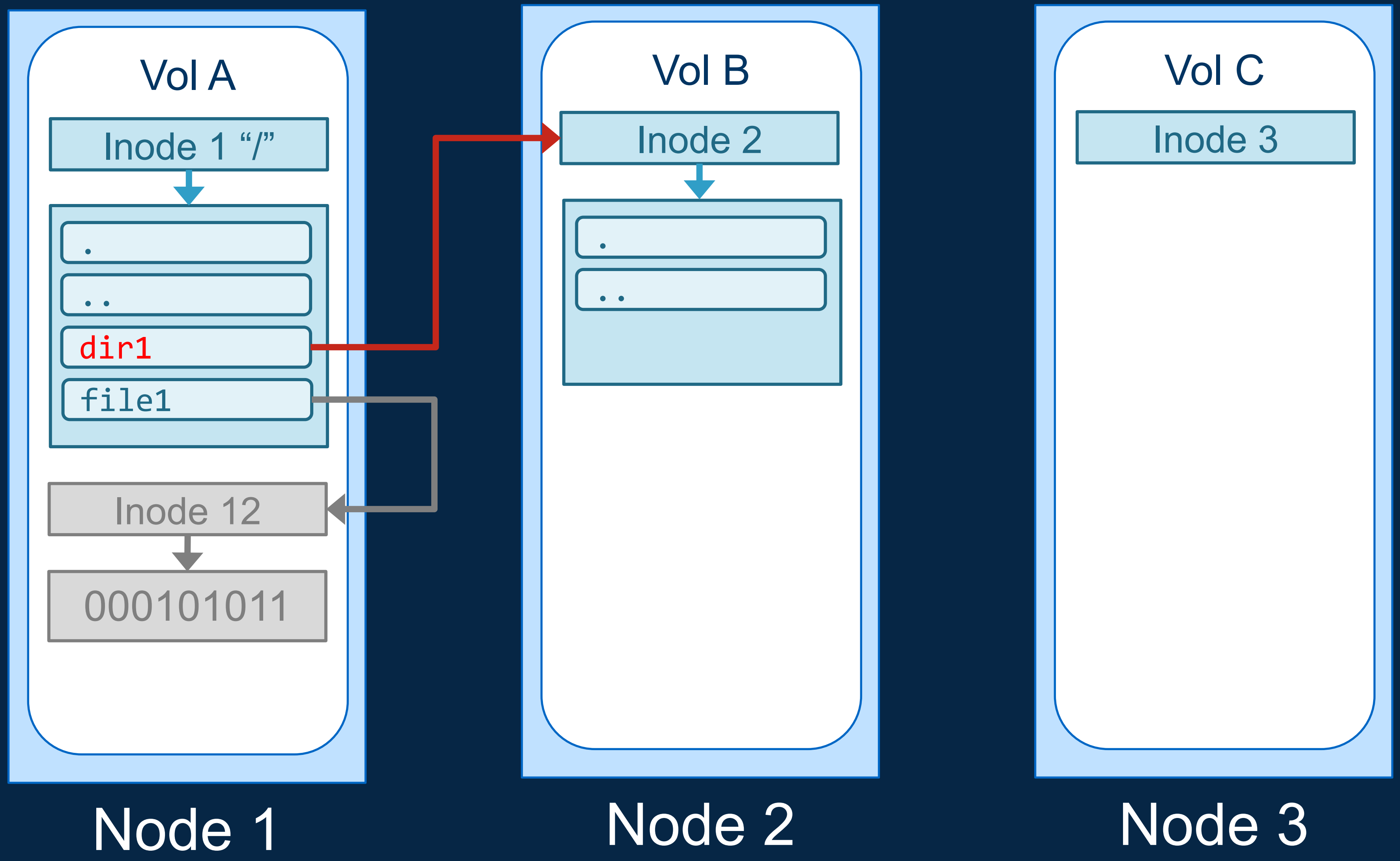
- Remote links + Remote Access Layer (RAL)
- Heuristics



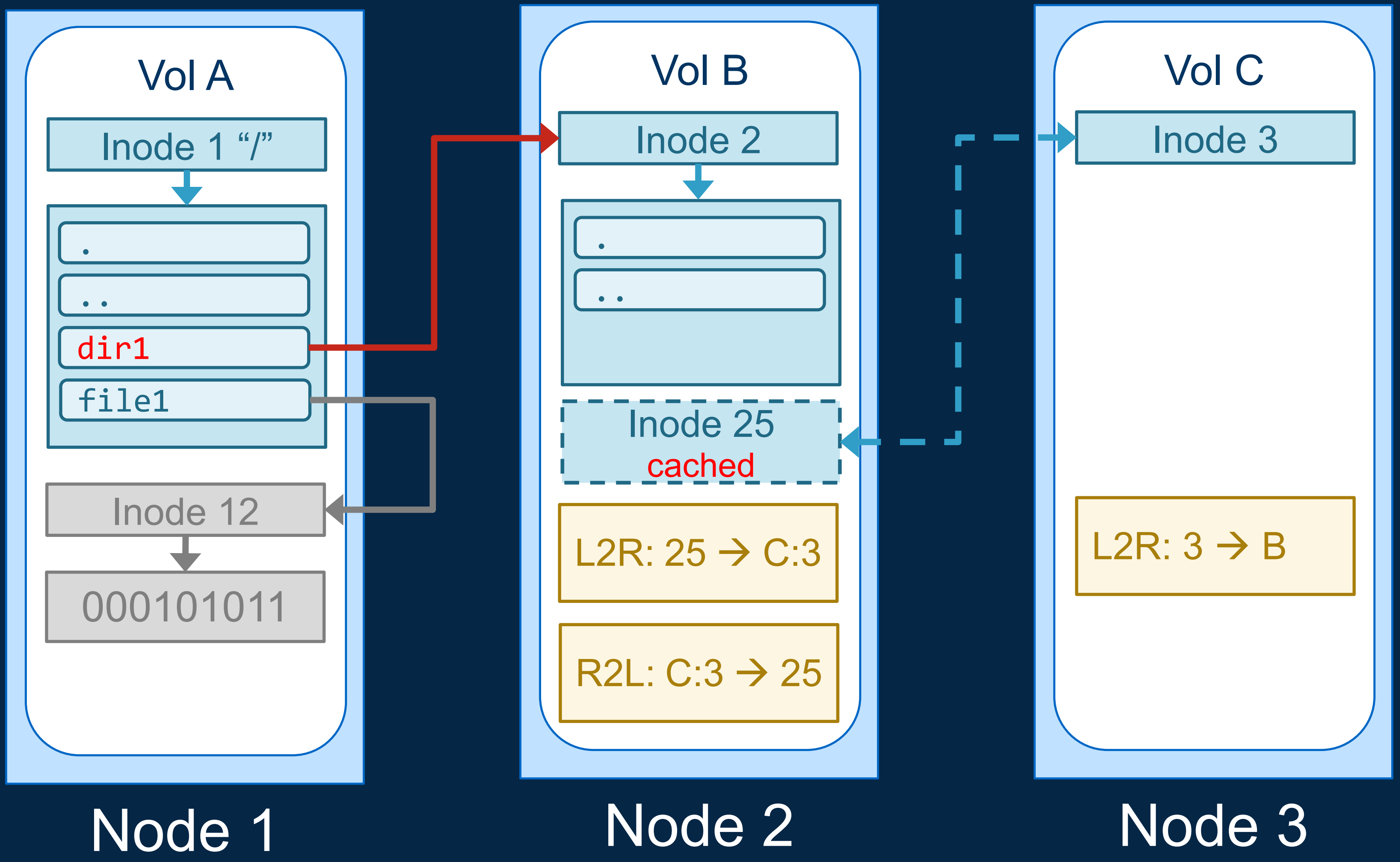
Remote Access Layer (RAL)

- Problem: How to coordinate operations across nodes?
- Answer: Delegations!
- RAL Cache: remote inodes are cached in a local inode
  - Cached inodes stored persistently and crash consistently in local filesystem
  - Writeback cache with delegations
  - Takes advantage of existing failure filesystem recovery

Remote Access Layer (RAL) Cache: New remote link to dir on Vol C
Step 1: create remote inode

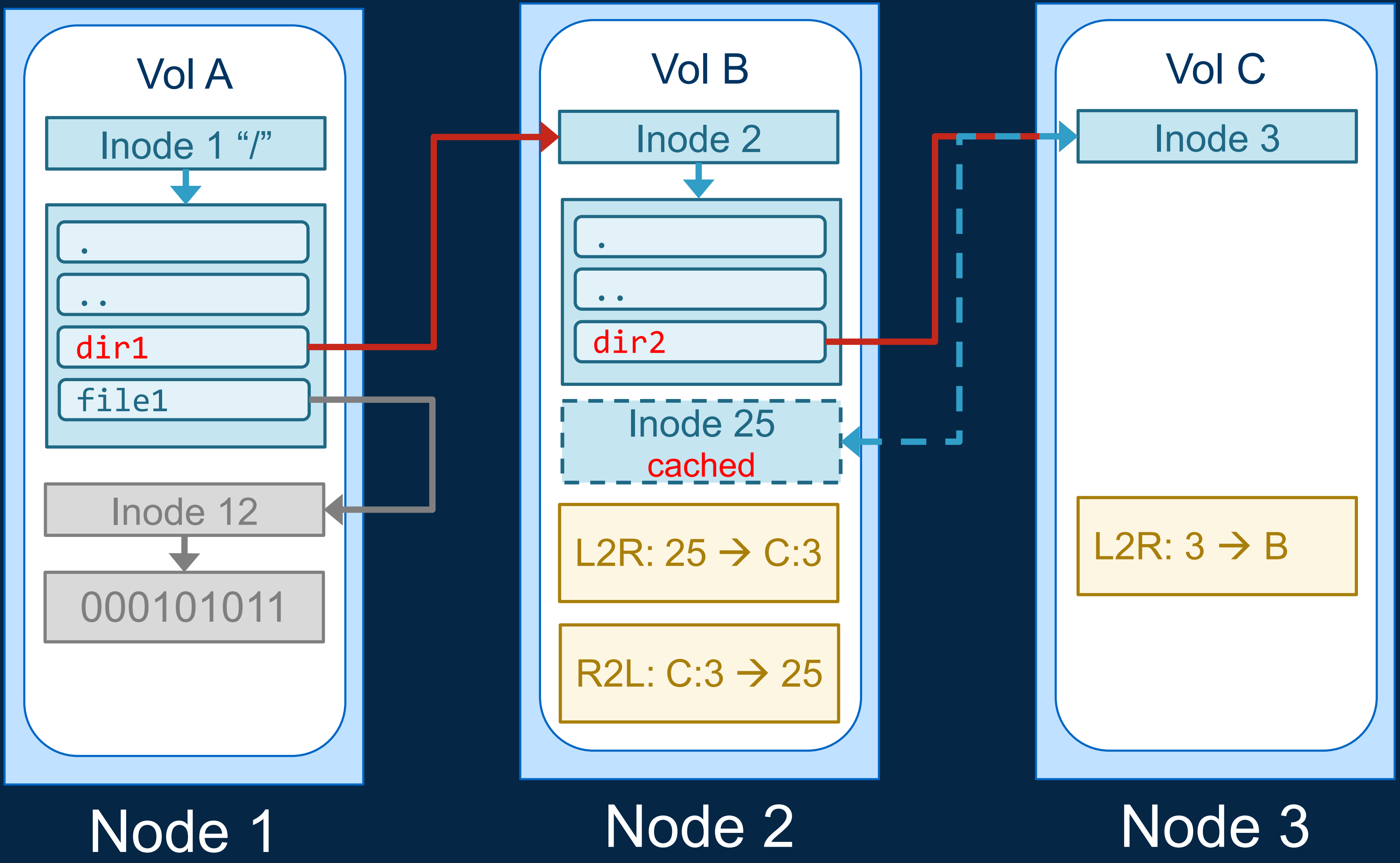


Remote Access Layer (RAL) Cache
Step 2: delegate to Vol B

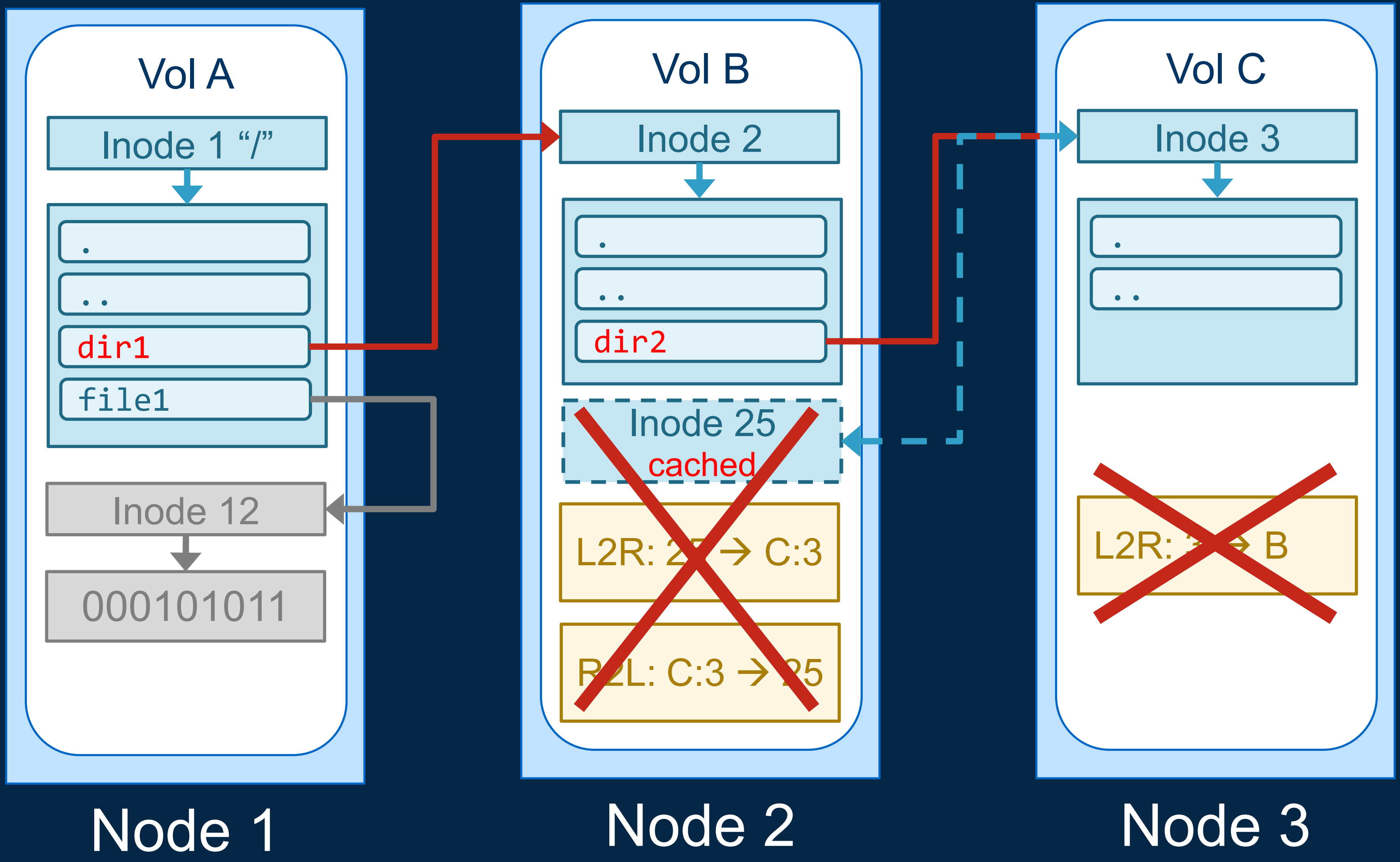


NetApp

Remote Access Layer (RAL) Cache
Step 3: Create remote link



Remote Access Layer (RAL) Cache
Step 4: Flush cached state back to origin



Heuristics for data placement

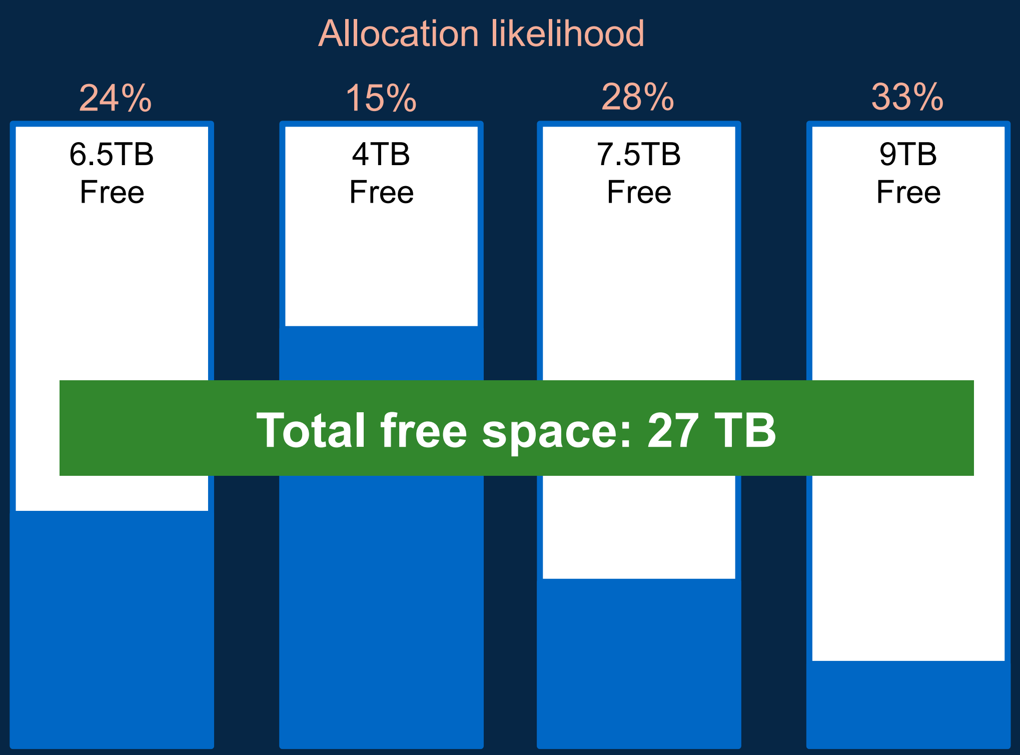
- Problems: When to allocate remote? If so, where?
- Goal: just enough remote links to balance space & current load
  - Calculated on block usage, inode usage and recent ingest load
- Decisions made at ingest only
- Each node makes independent placement decisions
- More aggressive as volume space fills

Remote allocation

Playing the percentages

- Probability of remote allocation relative to free space
- Simplified example
- Allocation proportional to free space

Total capacity: 40TB
Member capacity: 10 TB



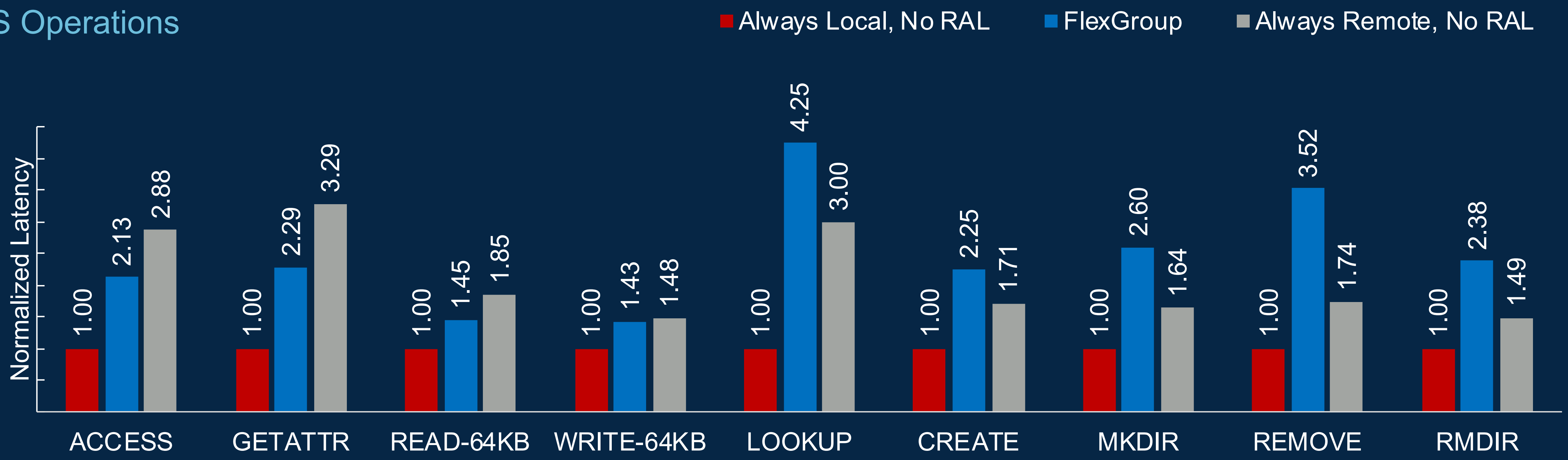
Evaluation

Evaluation

- Goals:
  - Quantify Remote Access Layer overhead
  - How well do heuristics work?
- Micro-benchmarks (mdtest)
- Macro-benchmarks (SFS)
- Customer experiences: space balancing in the real world

Microbenchmarks

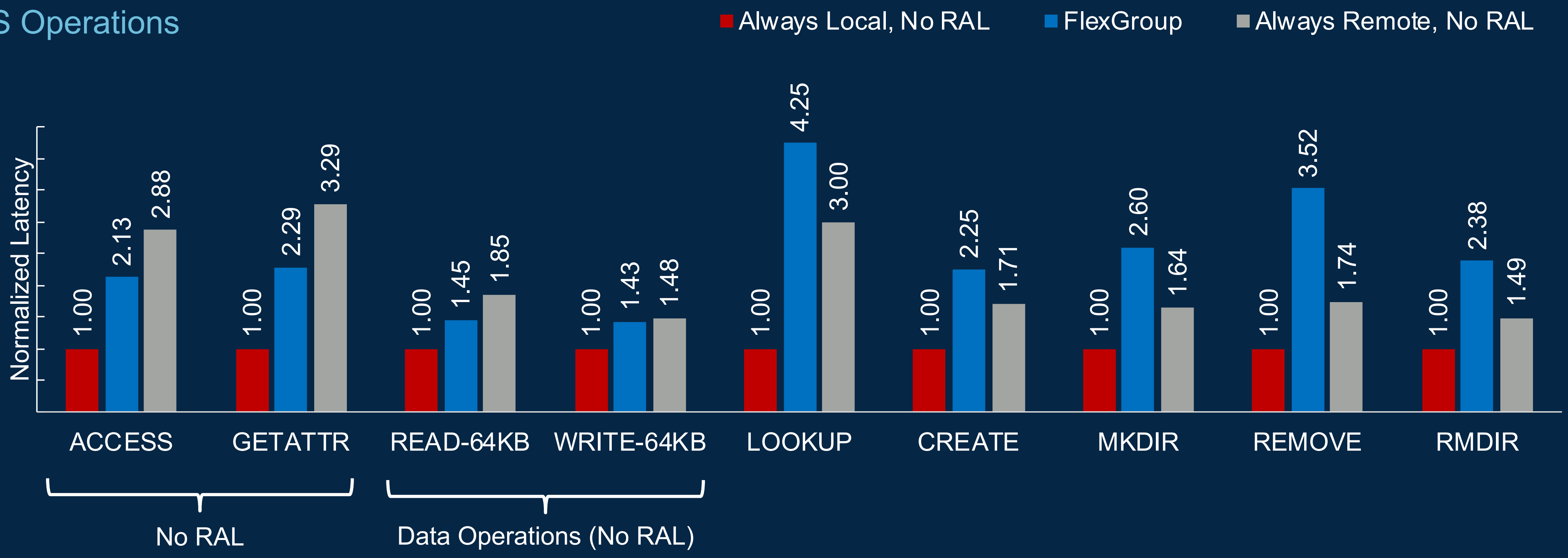
NFS Operations



- RAL overhead when creating RO or RW caches
- No RAL overhead otherwise

Microbenchmarks

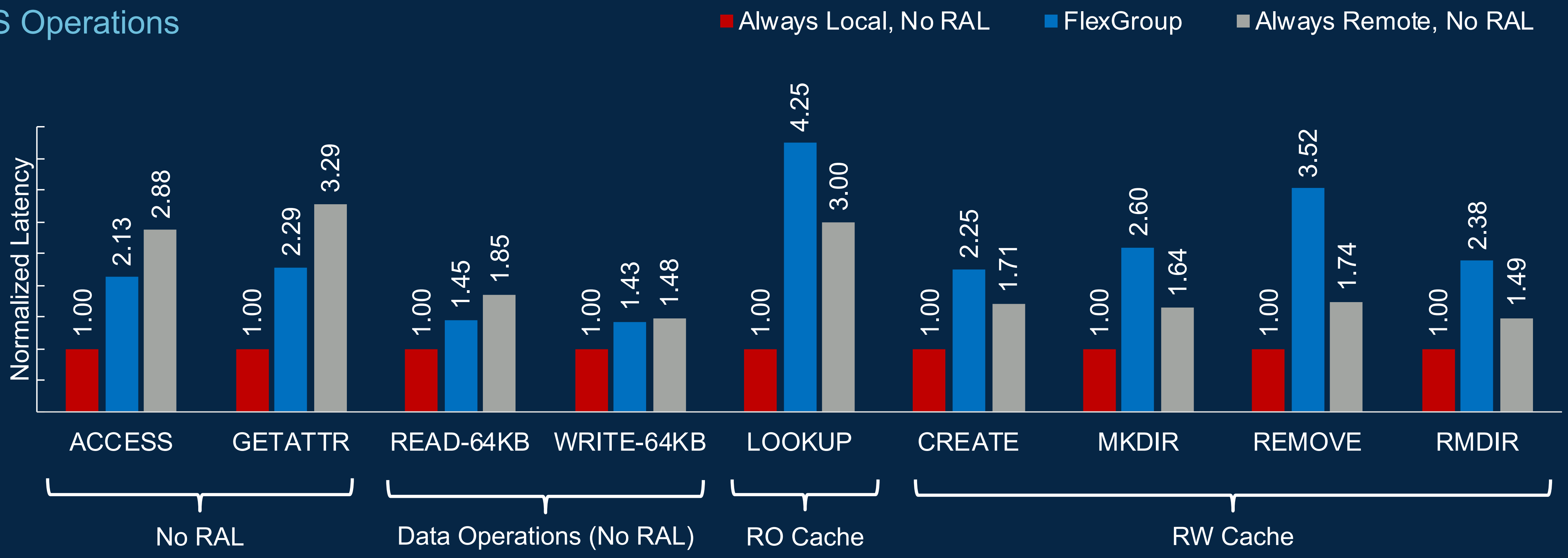
NFS Operations



- RAL overhead when creating RO or RW caches
- No RAL overhead otherwise

Microbenchmarks

NFS Operations

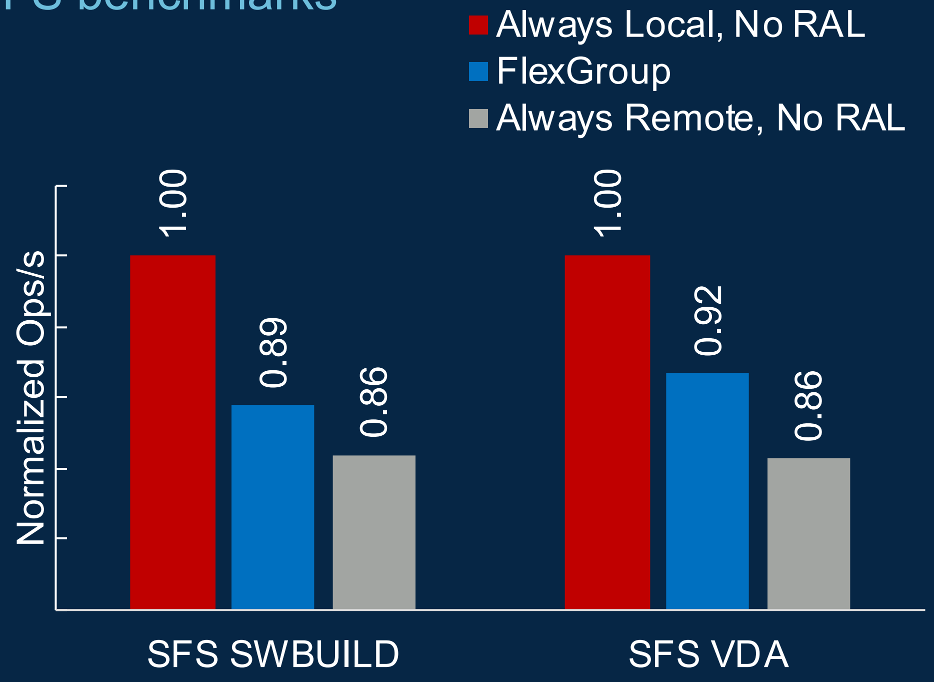


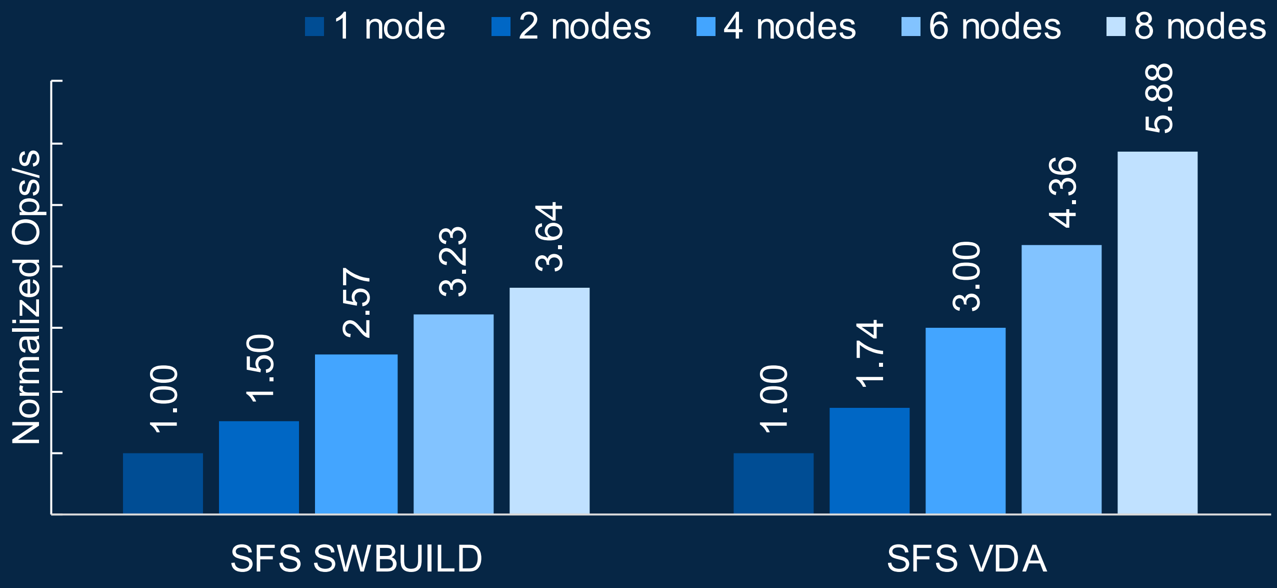
- RAL overhead when creating RO or RW caches
- No RAL overhead otherwise

NetApp

Macrobenchmarks

SFS benchmarks





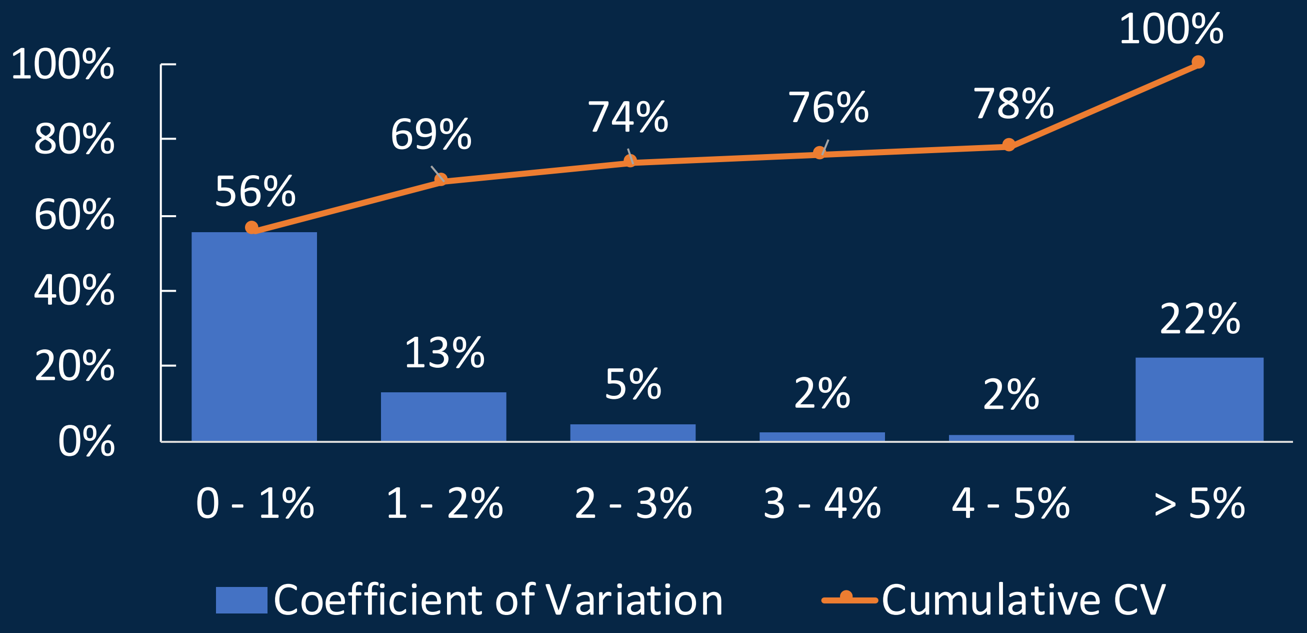
- Effective balancing of load, minimal RAL overhead

- Scalability by cluster size

Customer experiences

Achieved balancing

$$COV = \frac{stdev}{mean}$$

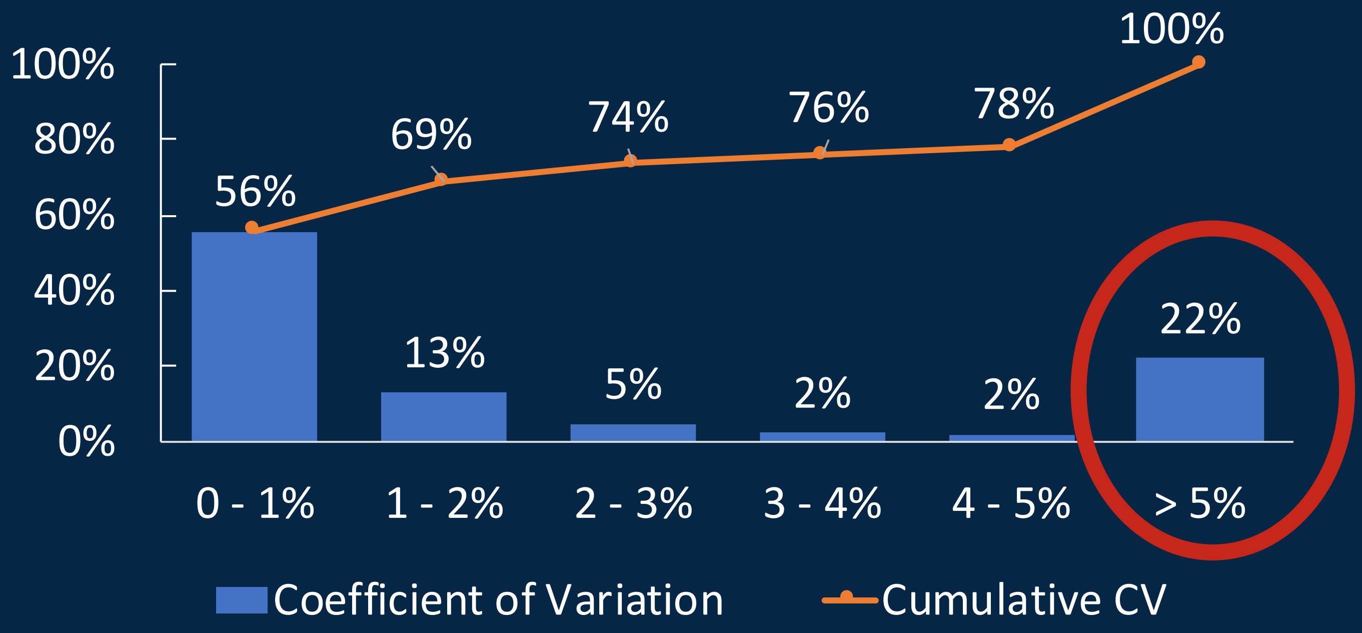


- 1000s of FlexGroups, 100s of PB deployed in the field
- Data well-balanced for most customers, with two exceptions:
  - Adding new member volumes to increase capacity
  - FlexGroups with small numbers of very large files

Customer experiences

Achieved balancing

$$COV = \frac{stdev}{mean}$$



- 1000s of FlexGroups, 100s of PB deployed in the field
- Data well-balanced for most customers, with two exceptions:
  - Adding new member volumes to increase capacity
  - FlexGroups with small numbers of very large files

Conclusions

- Created distributed filesystem from existing technologies
- A reliable local filesystem changes design
- More papers on ONTAP/WAFL: https://atg.netapp.com/?tag=ontap