Assise: Performance and Availability via Client-local NVM in a Distributed File System

Thomas Anderson, Marco Canini, Jongyul Kim, Dejan Kostić, Youngjin Kwon, Simon Peter, **Waleed Reda**, Henry N. Schuh, and Emmett Witchel

Commercial adoption of NVM

intel. OPTANE™

Non-Volatile Memory

- **Fast IO with byte-addressability**
- **Only 40% cost of DRAM**
- **Persistent across failures**

	DRAM		NVM
Write Latency	82 ns	~	94 ns
Cost ($/GB)	9.77	>>	3.83

Application

Memory

Commercial adoption of NVM

intel. OPTANE™

Non-Volatile Memory

- **Fast IO with byte-addressability**
- **Only 40% cost of DRAM**
- **Persistent across failures**

	DRAM		**NVM**
Write Latency	82 ns	~	94 ns
Cost ($/GB)	9.77	>>	3.83

Application

Memory

Commercial adoption of NVM

intel OPTANE

Non-Volatile Memory

- **Fast IO with byte-addressability**
- **Only 40% cost of DRAM**
- **Persistent across failures**

	DRAM		NVM
Write Latency	82 ns	~	94 ns
Cost ($/GB)	9.77	>>	3.83

Application

Memory

Commercial adoption of NVM

Non-Volatile Memory

- **Fast IO with byte-addressability**
- **Only 40% cost of DRAM**
- **Persistent across failures**

intel OPTANE

	DRAM		NVM
Write Latency	82 ns	~	94 ns
Cost ($/GB)	9.77	>>	3.83

Application	
DRAM	NVM

Application	
DRAM	NVM

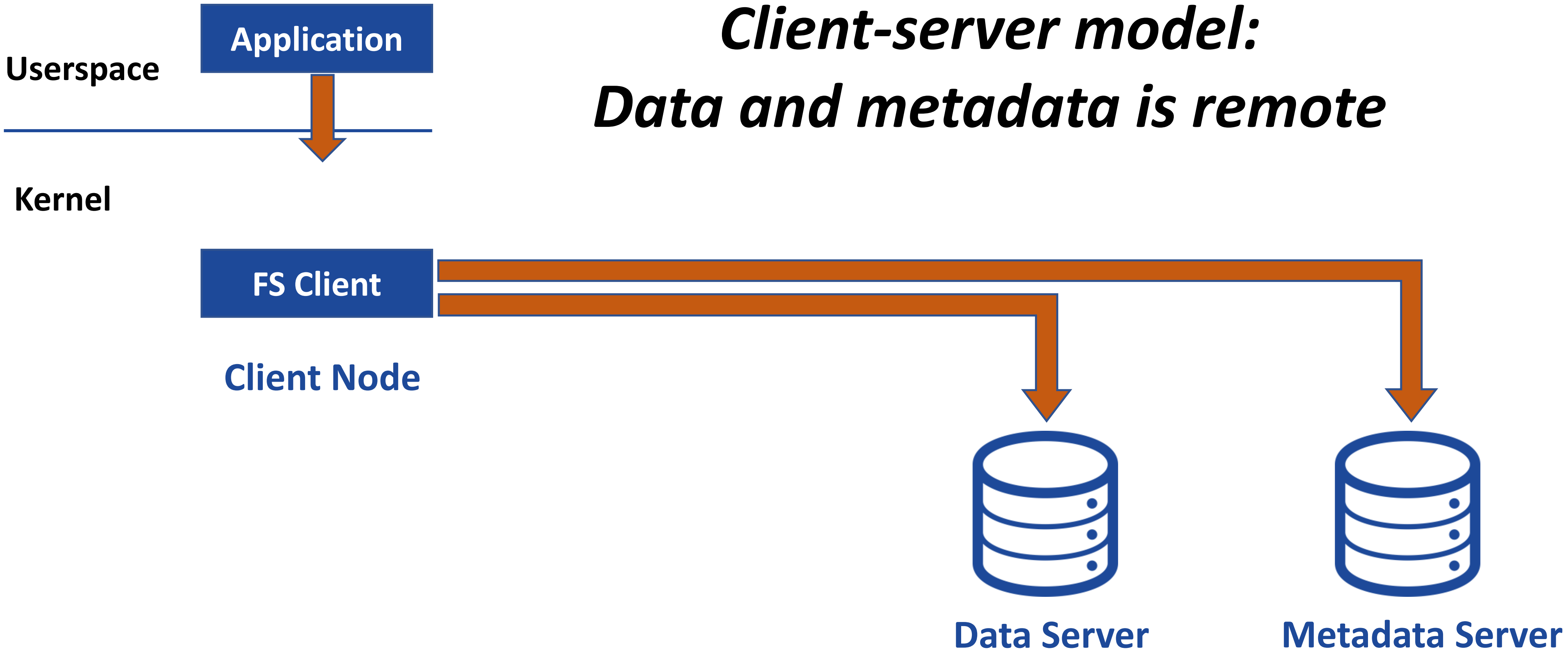
Application	
DRAM	NVM

NVM adoption at scale

Distributed filesystems are integrating NVM

e.g. Ceph, NFS, …

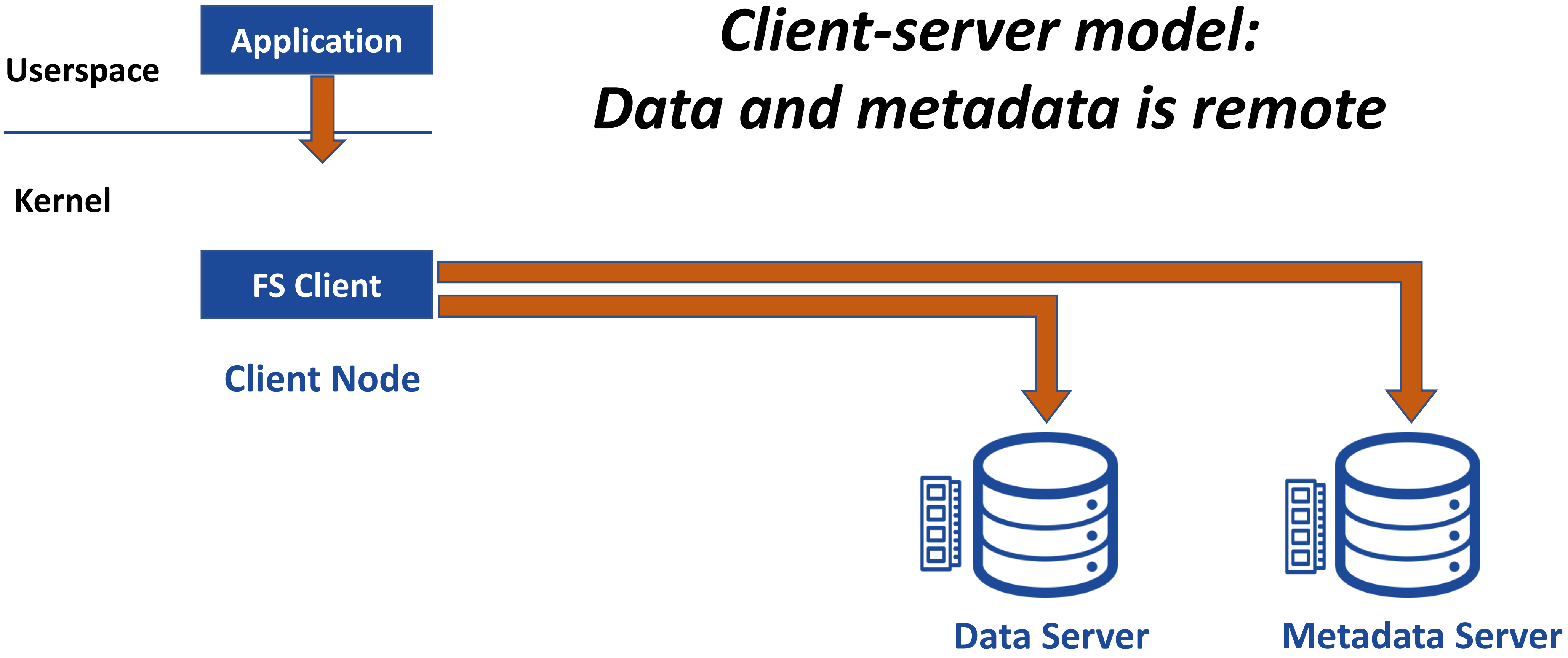
Client-server model:
Data and metadata is remote



Distributed filesystems are integrating NVM

e.g. Ceph, NFS, …

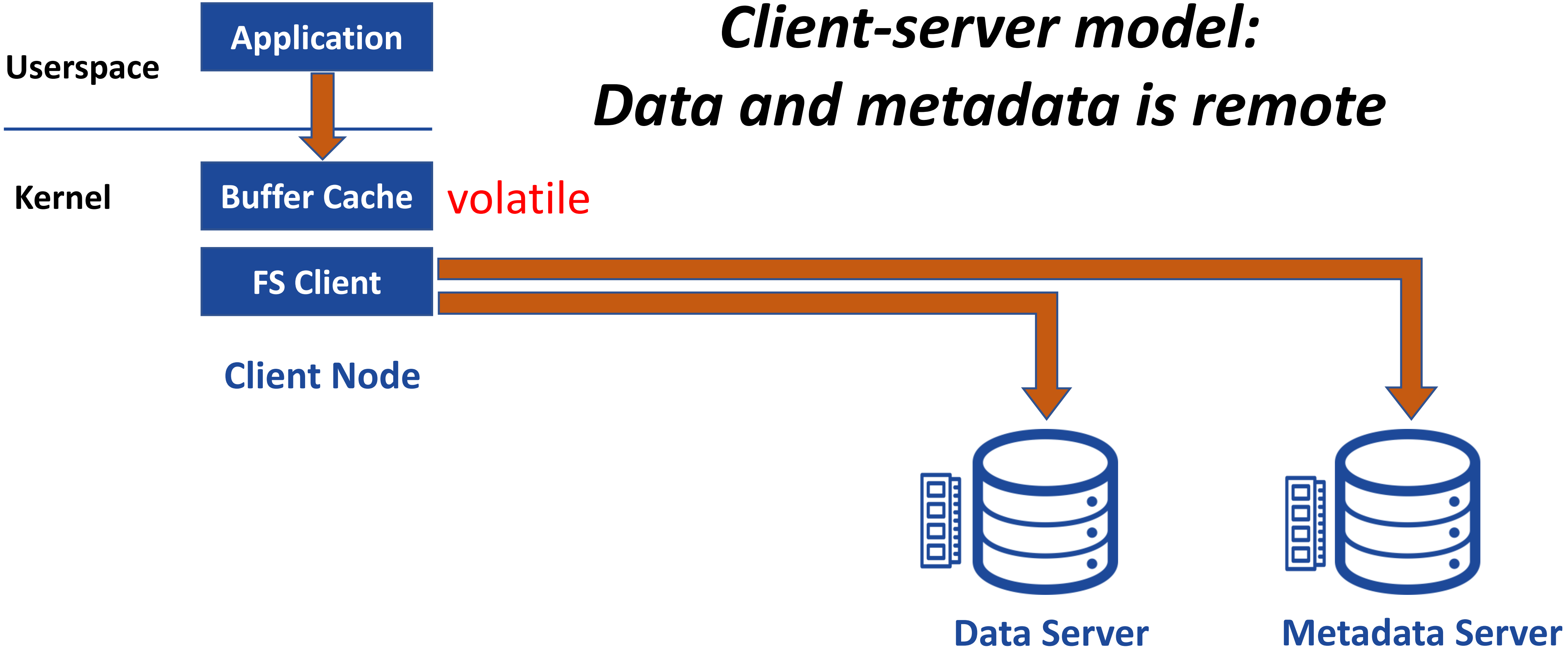
Client-server model:
Data and metadata is remote



Distributed filesystems are integrating NVM

e.g. Ceph, NFS, …

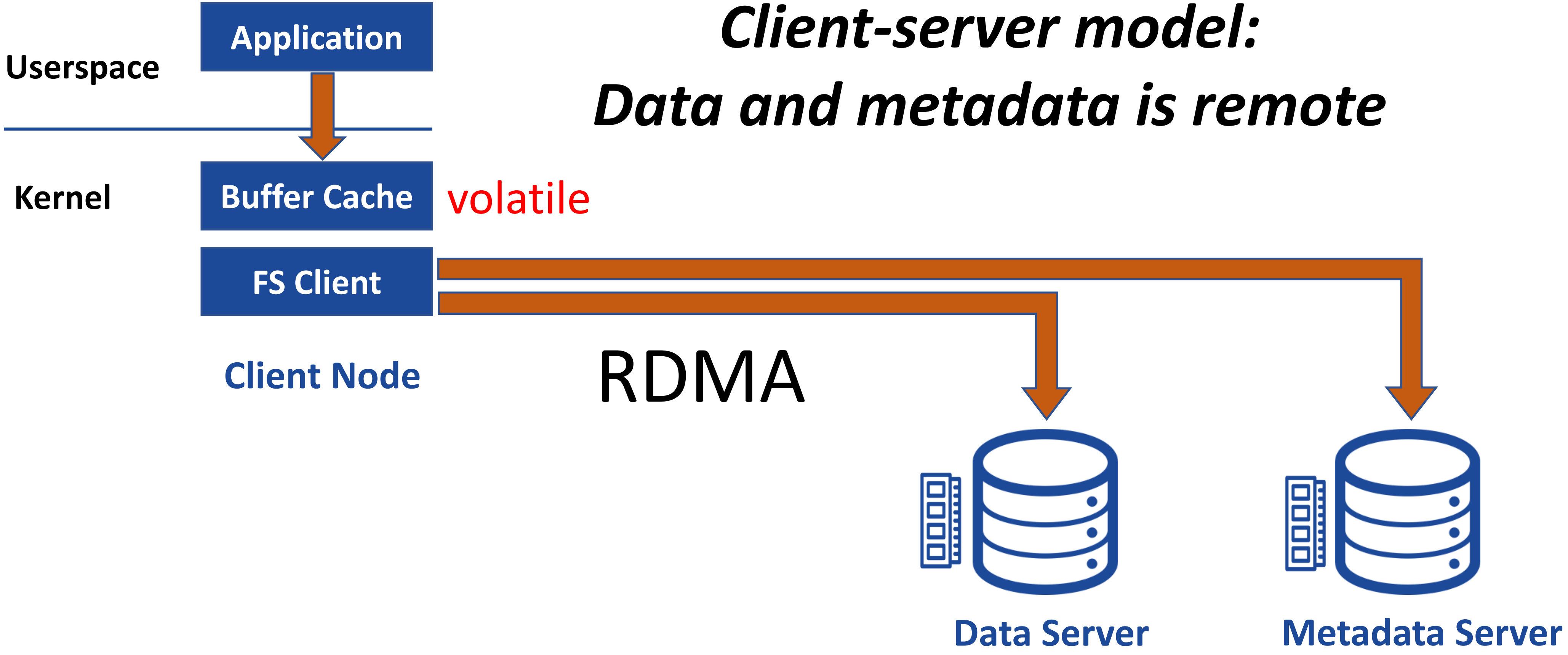
Client-server model:
Data and metadata is remote



Distributed filesystems are integrating NVM

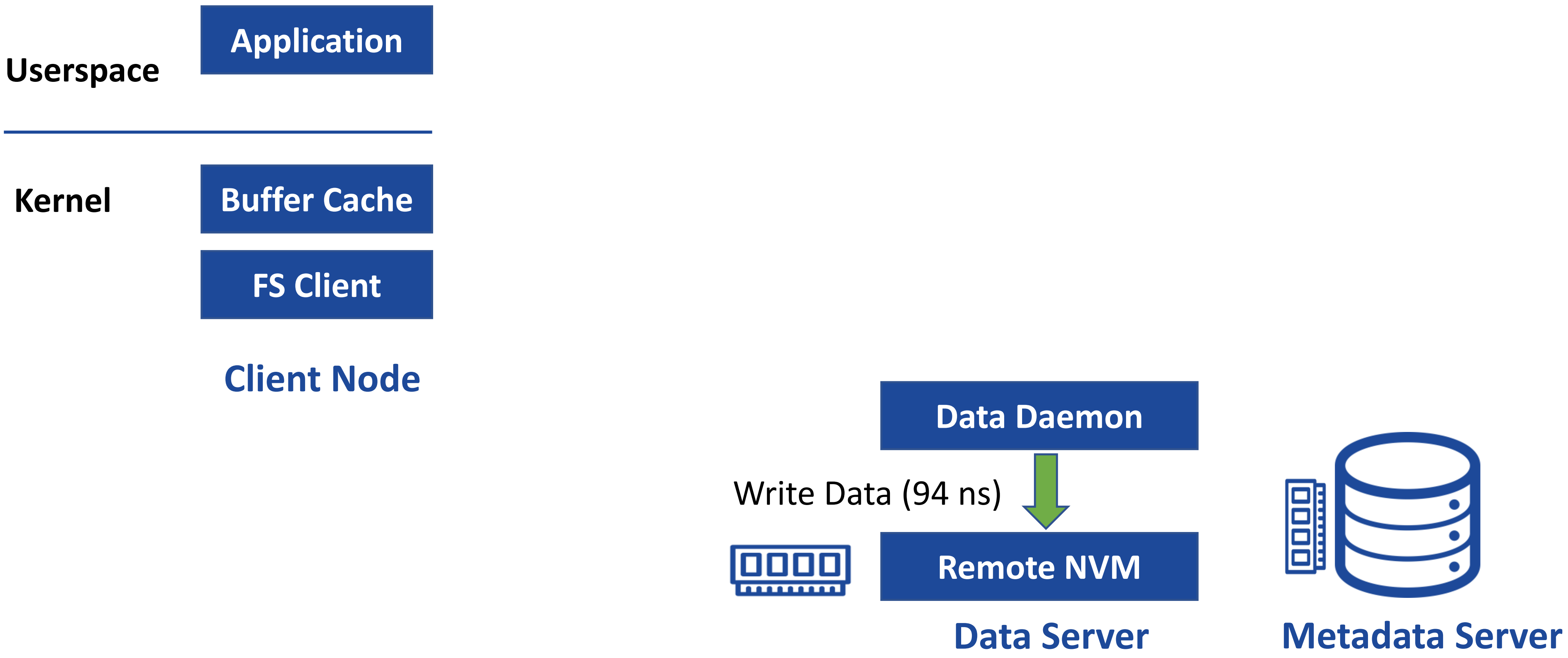
e.g. Ceph, NFS, …

Client-server model:
Data and metadata is remote



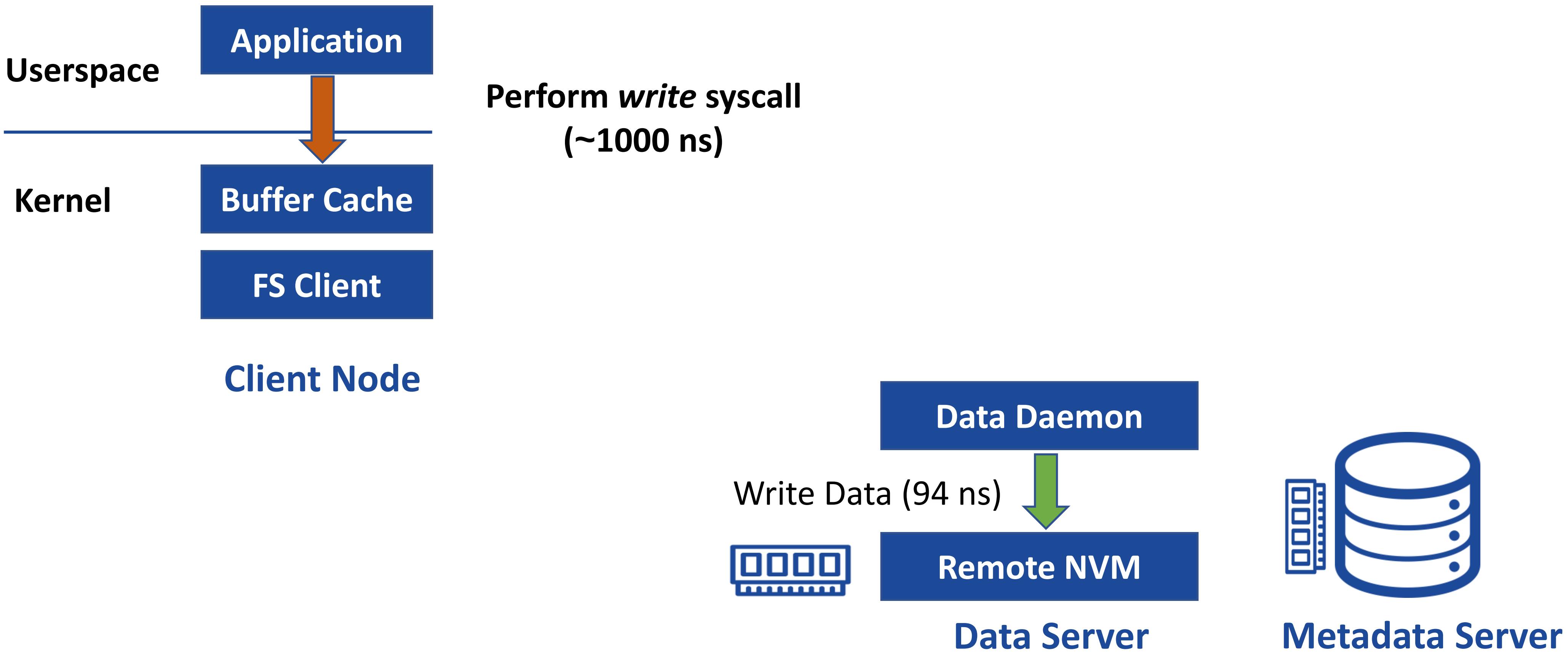
Example: Write 256B durably to a file

Challenge 1: Network/kernel latencies dominate NVM IO latency



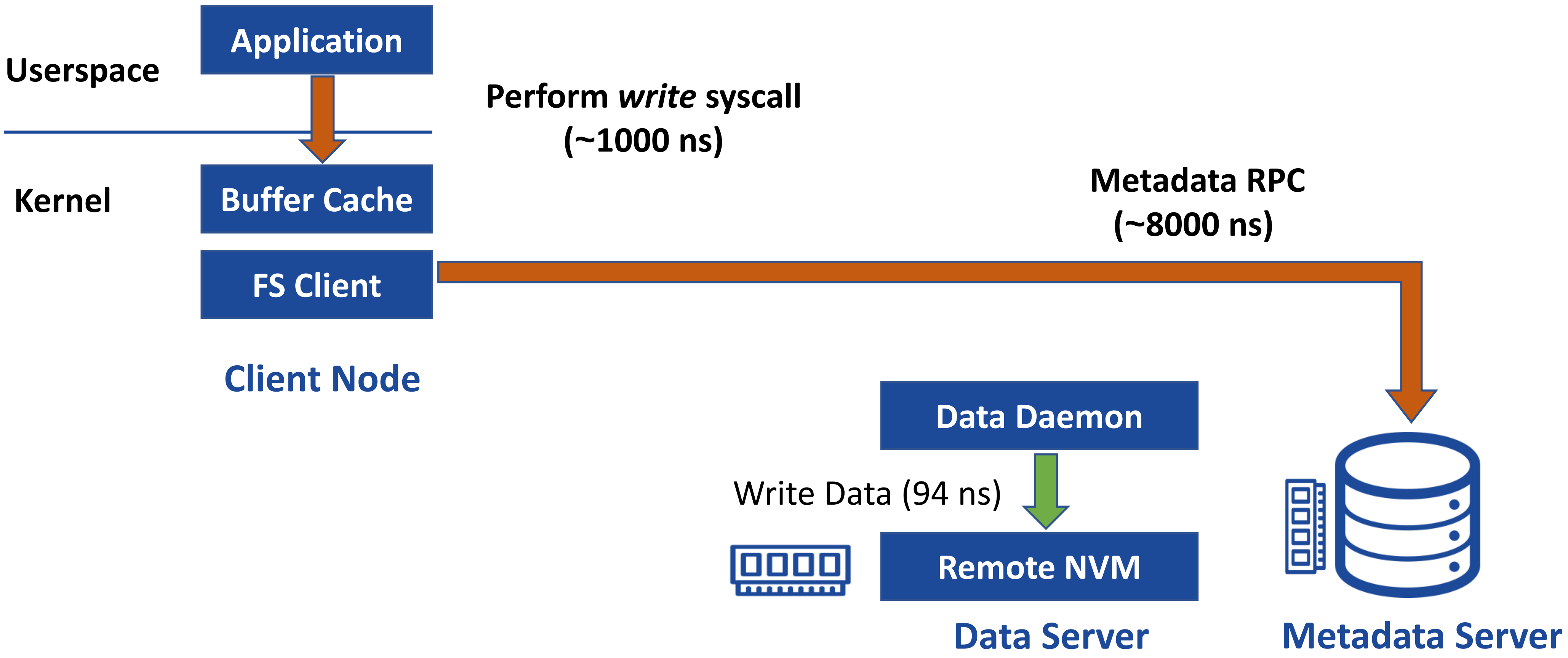
Example: Write 256B durably to a file

Challenge 1: Network/kernel latencies dominate NVM IO latency



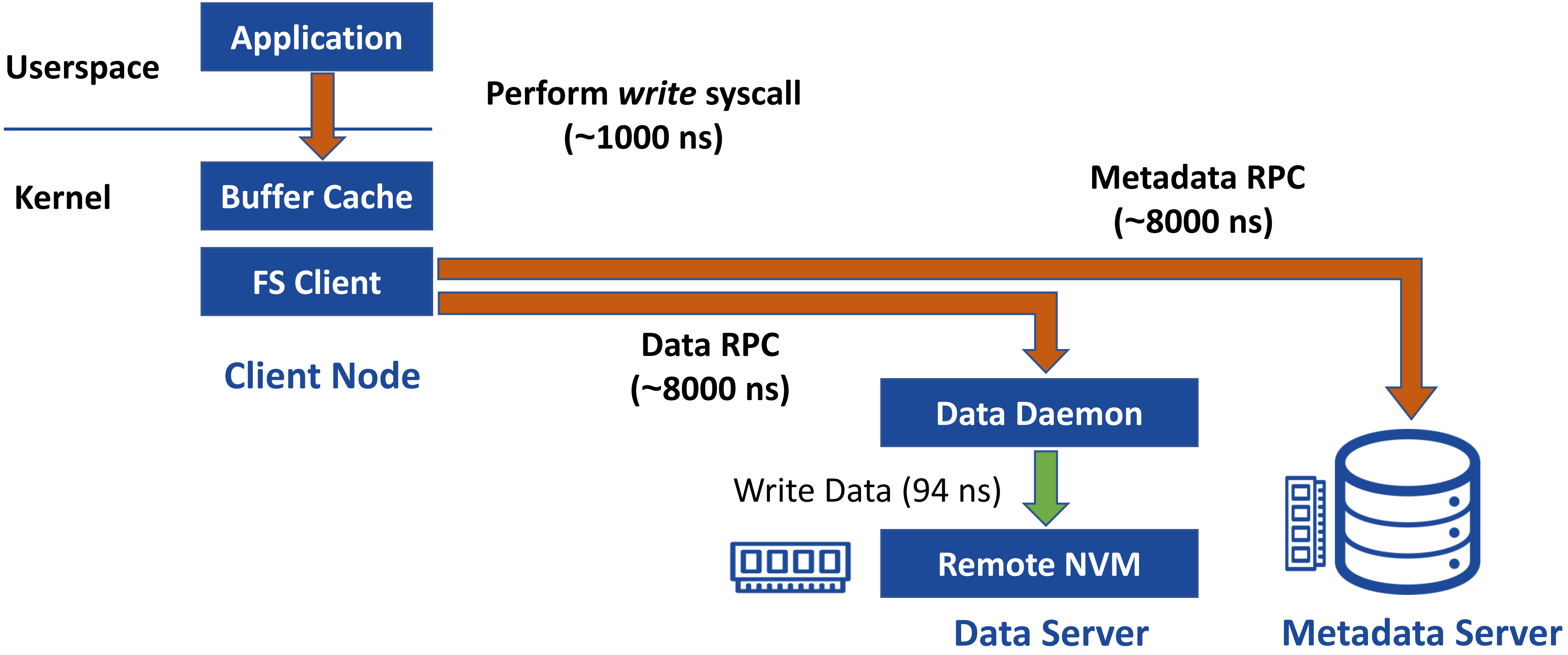
Example: Write 256B durably to a file

Challenge 1: Network/kernel latencies dominate NVM IO latency



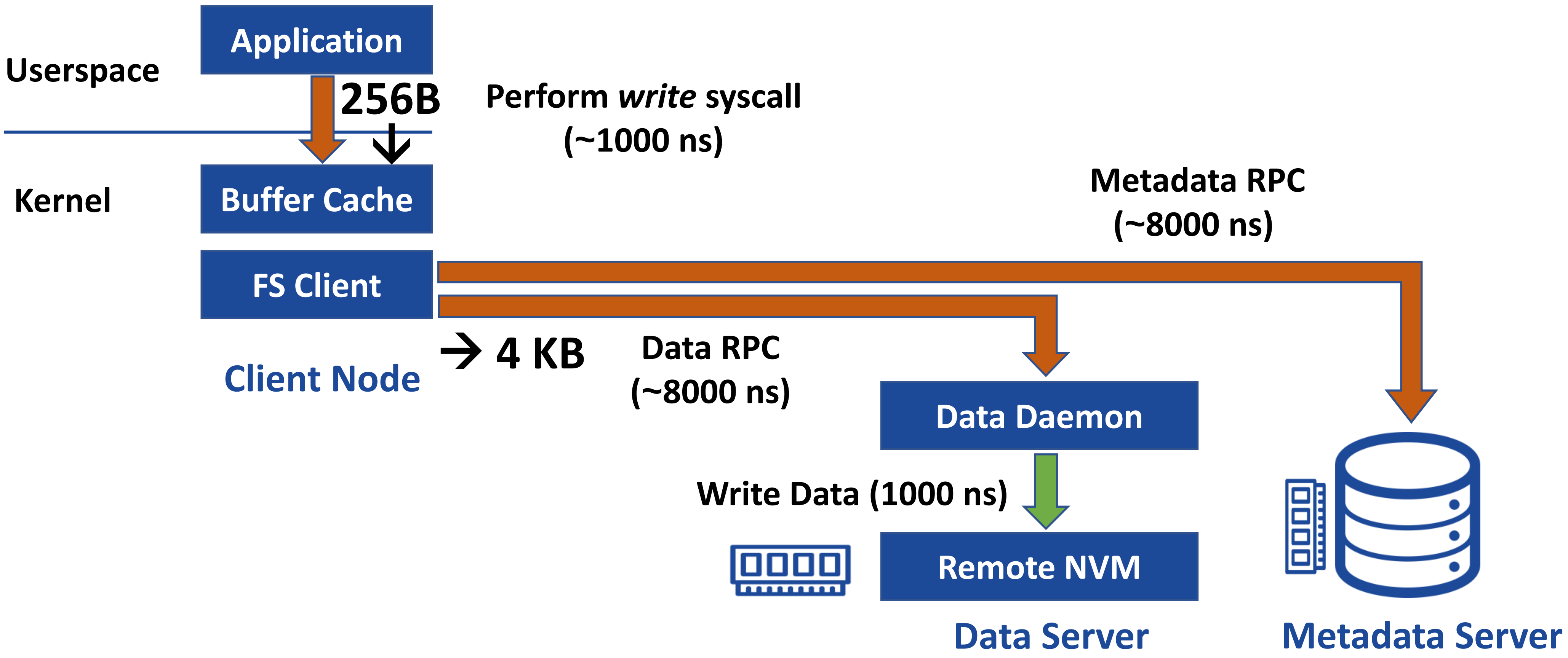
Example: Write 256B durably to a file

Challenge 1: Network/kernel latencies dominate NVM IO latency



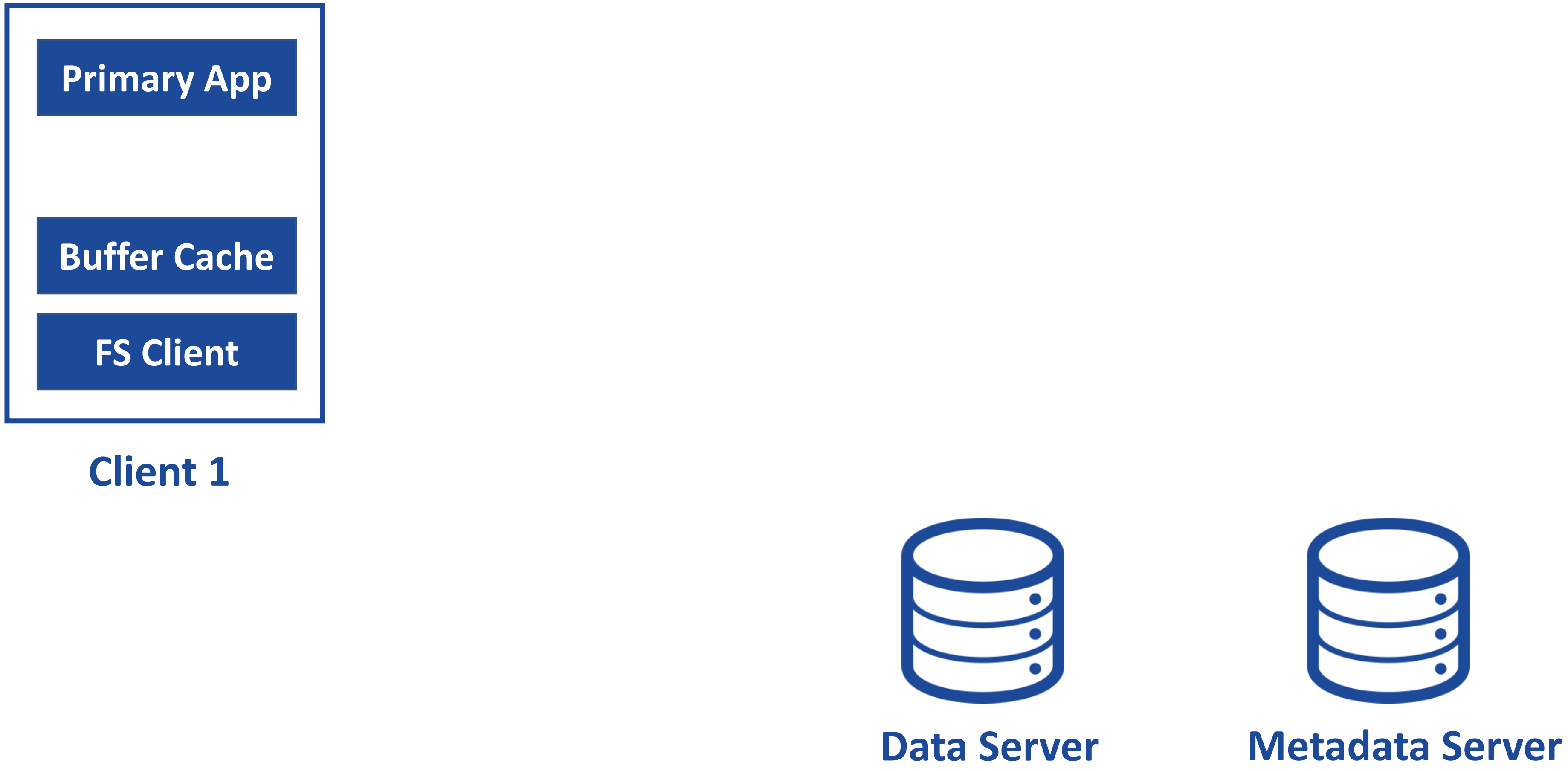
Example: Write 256B durably to a file

Challenge 2: Block-sized IO exacerbates network & storage latencies



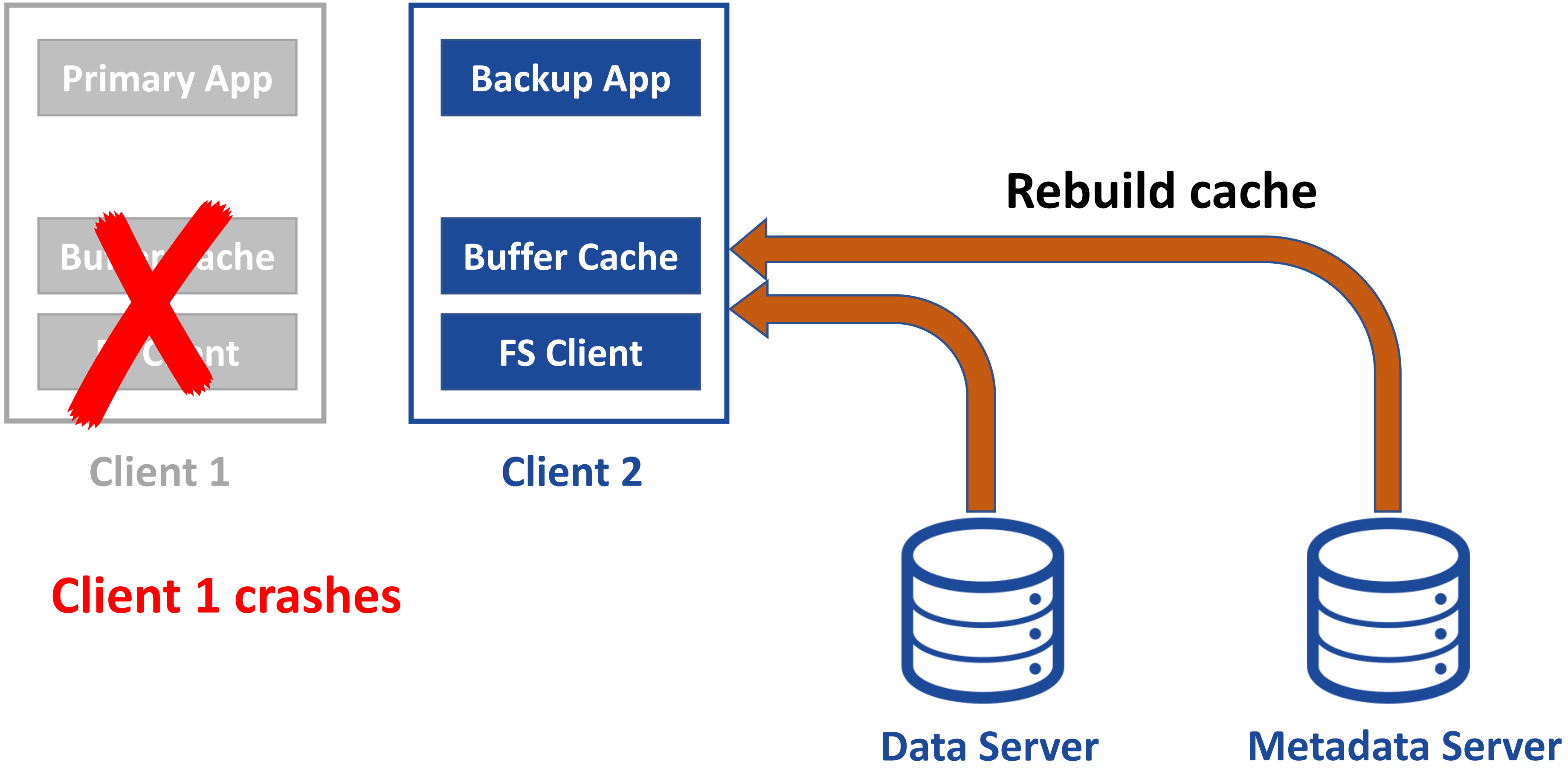
Client fail-over & recovery

Challenge 3: File system must rebuild caches of failed clients



Client fail-over & recovery

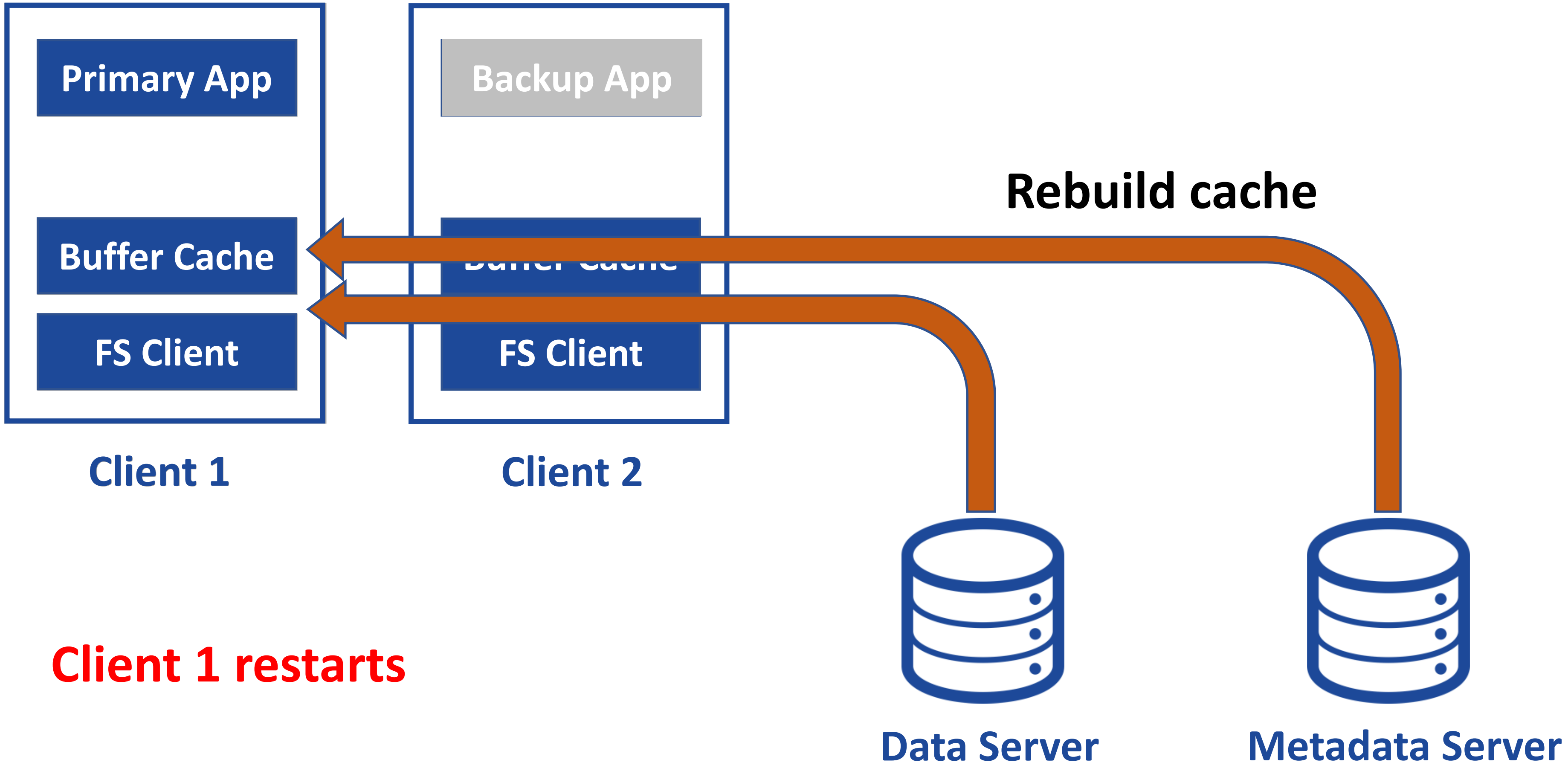
Challenge 3: File system must rebuild caches of failed clients



Client 1 crashes

Client fail-over & recovery

Challenge 3: File system must rebuild caches of failed clients

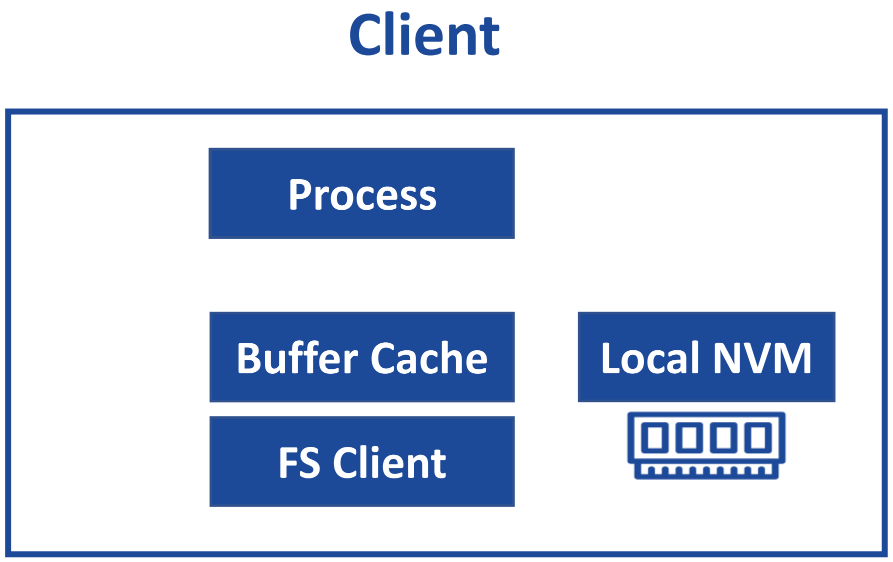


Assise: High performance & availability

Main principle: **Maximize use of client-local NVM**

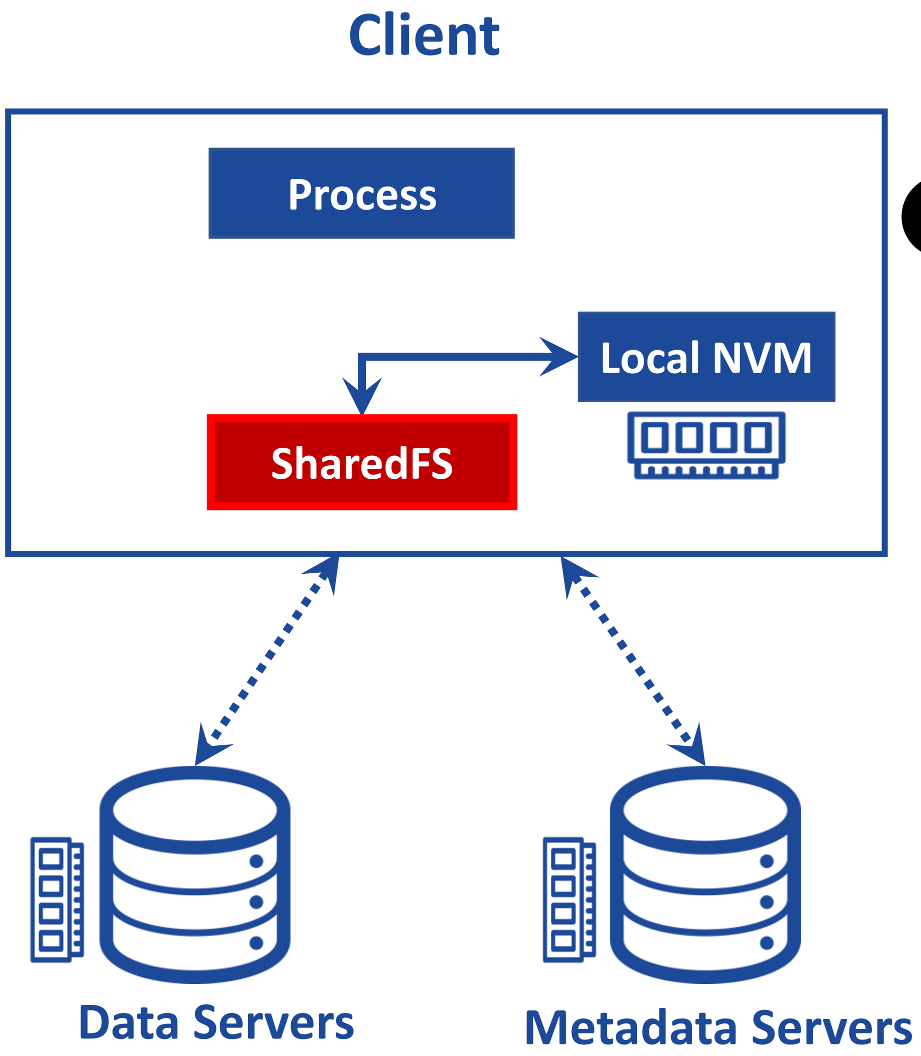
1. **Low latency IO**: Perform process-local & client-local IO
2. **Linearizability and data crash consistency**
   CC-NVM distributed client-side NVM coherence protocol
3. **High availability**
   Fail-over to cache-hot client replicas, recover client NVM caches

Client-local IO with Assise



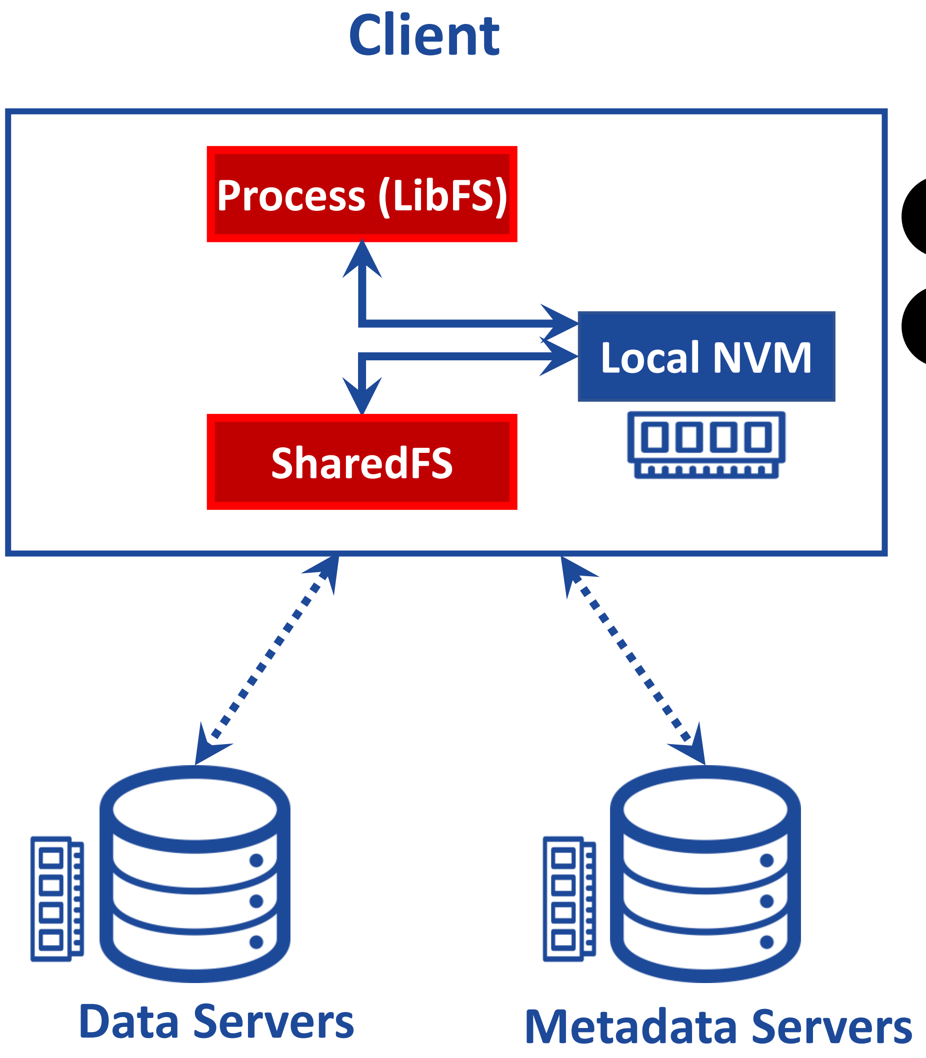
❶ **Store (meta-)data in client-local NVM**

Client-local IO with Assise



❶ **Store (meta-)data in client-local NVM**

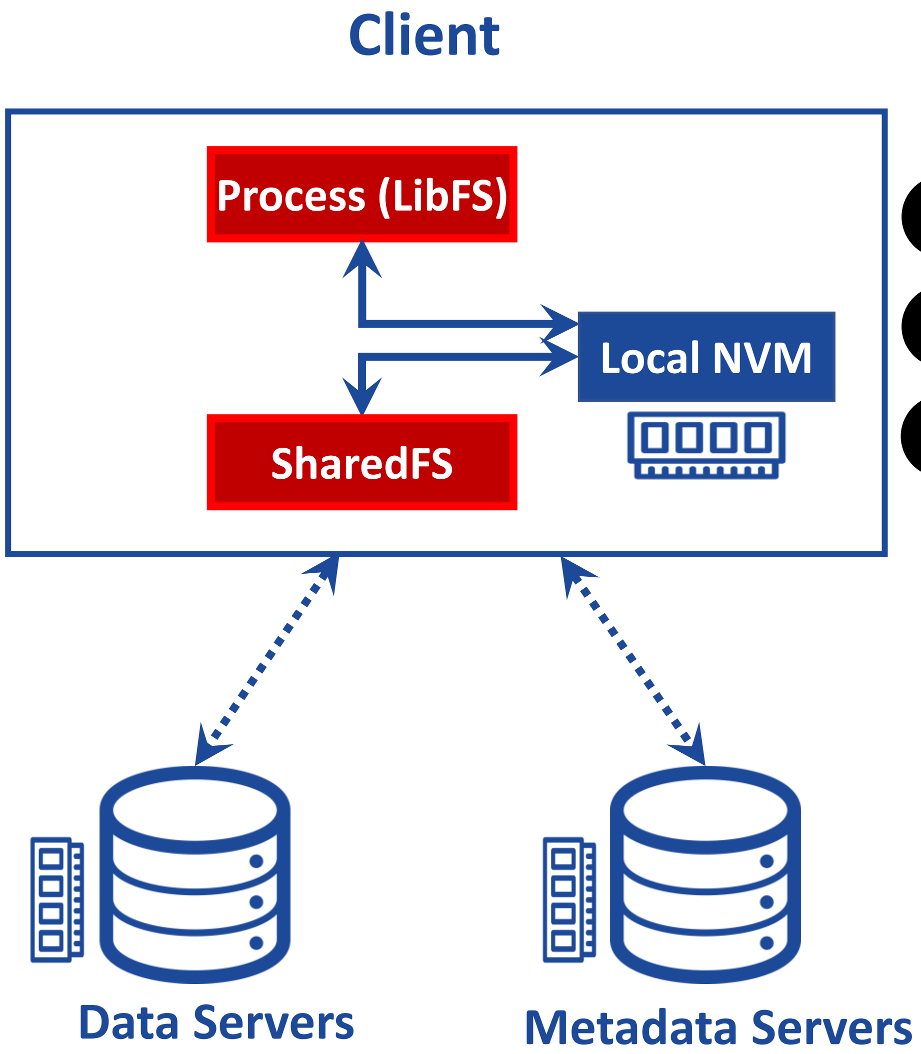
Client-local IO with Assise



❶ **Store (meta-)data in client-local NVM**

❷ **Userspace access via LibFS**

Client-local IO with Assise

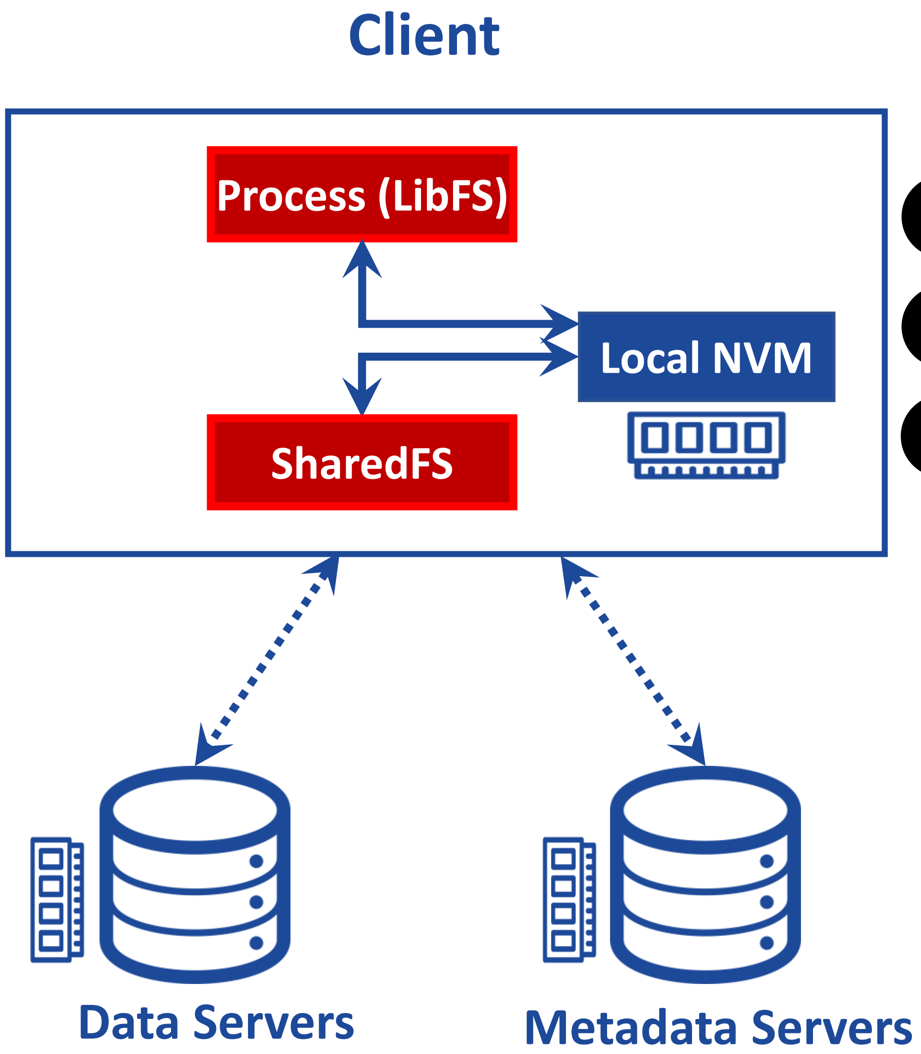


❶ Store (meta-)data in client-local NVM

❷ Userspace access via LibFS

❸ Log writes at operation granularity

Client-local IO with Assise



❶ **Store (meta-)data in client-local NVM**

❷ **Userspace access via LibFS**

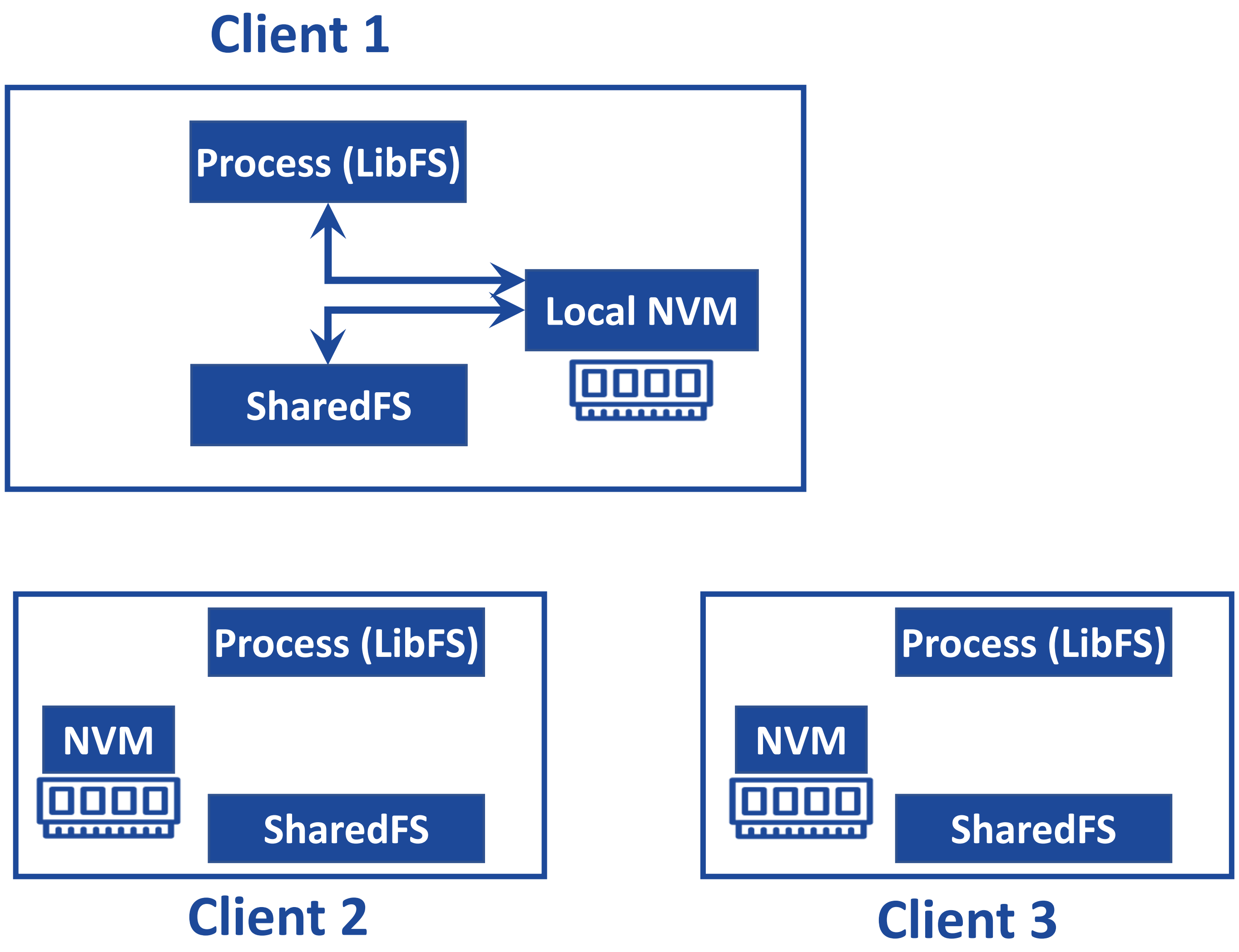
❸ **Log writes at operation granularity**

Local userspace data access (= low latency IO)

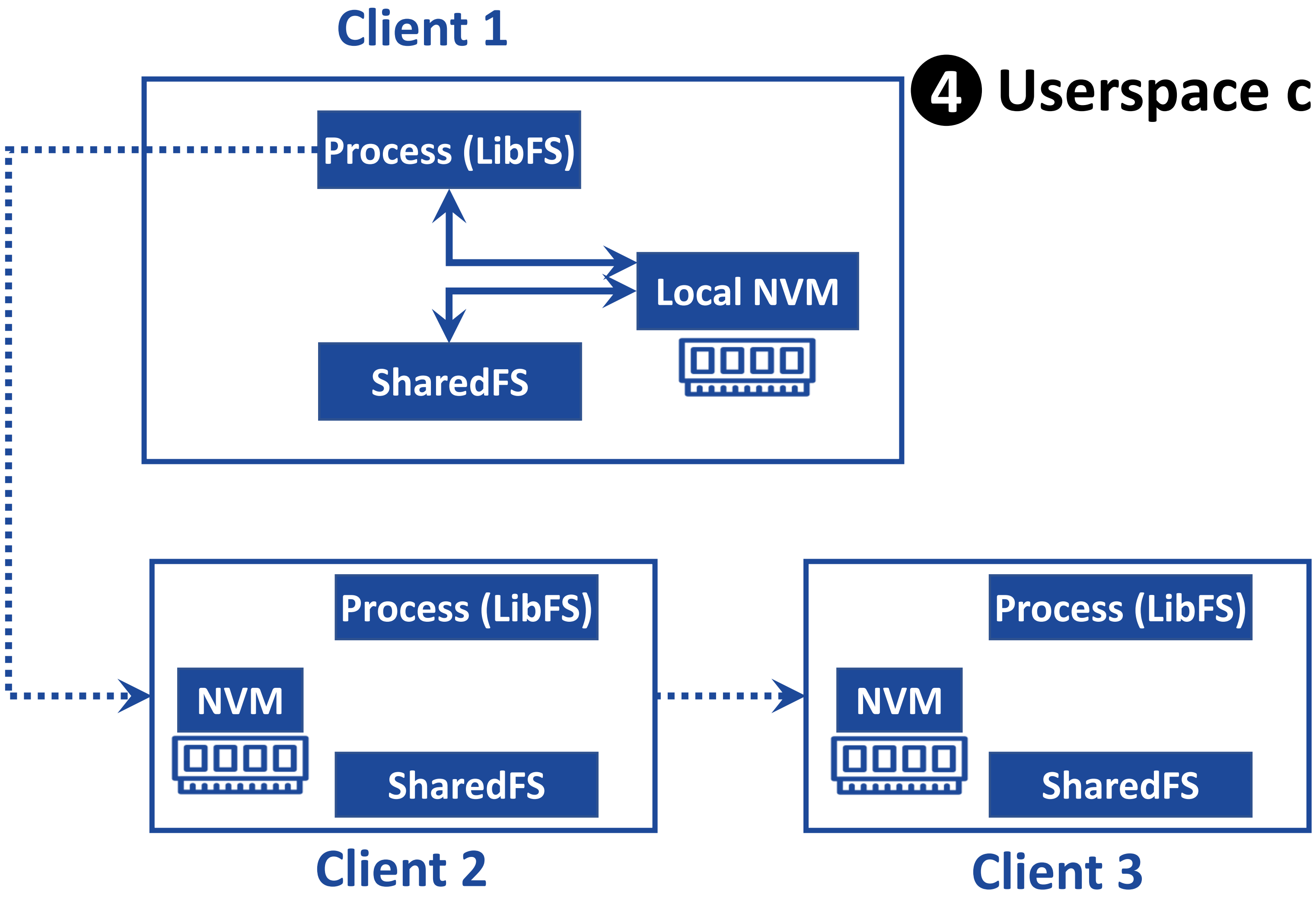
No block amplification (= efficient IO)

Local durability (= fast, local recovery)

Remote IO with Assise

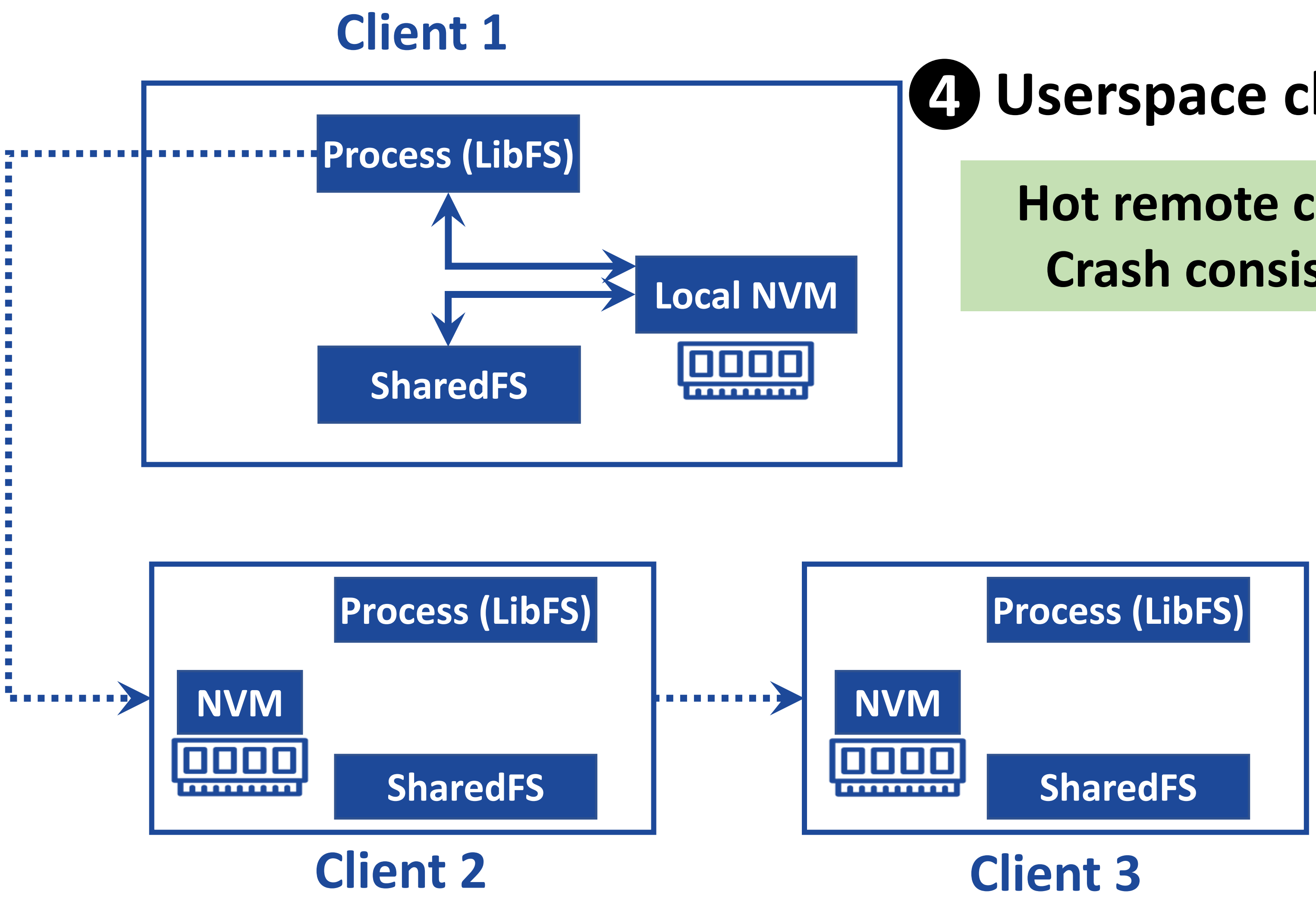


Remote IO with Assise



❹ **Userspace chain-replication via RDMA**

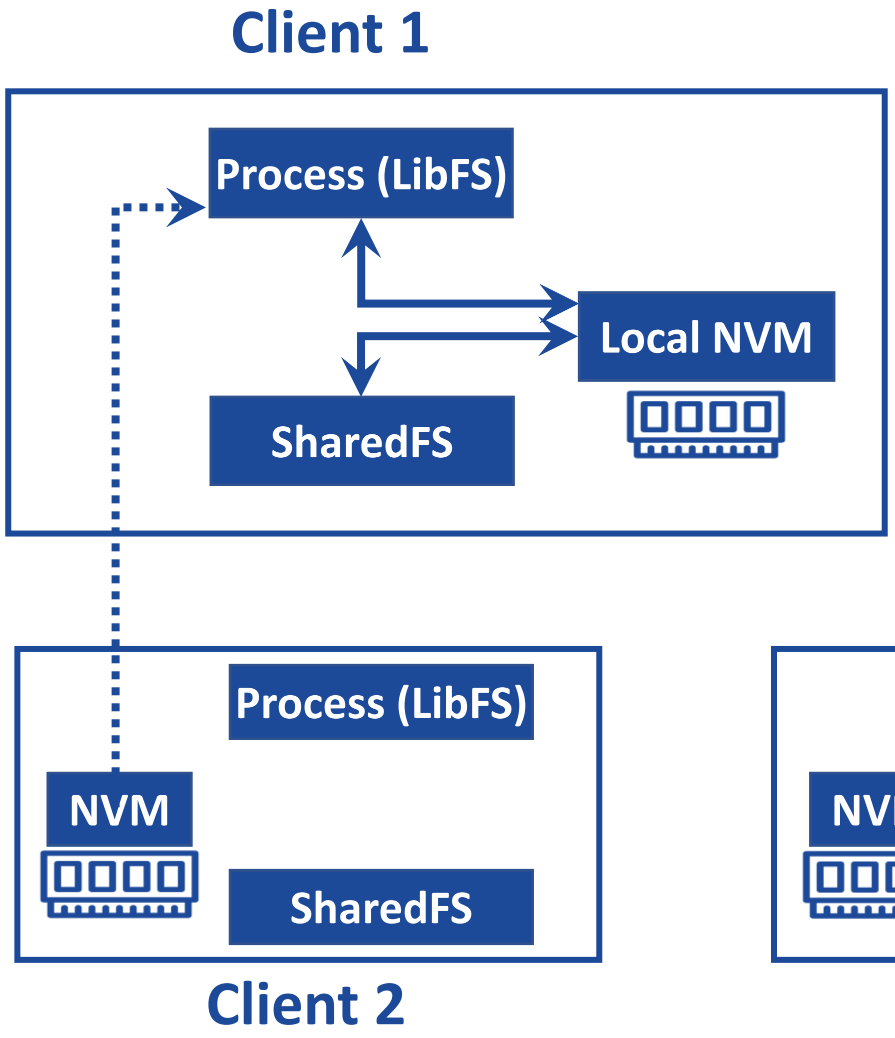
Remote IO with Assise



❹ Userspace chain-replication via RDMA

Hot remote client caches (= fast fail-over)
Crash consistent via RDMA write order

Remote IO with Assise

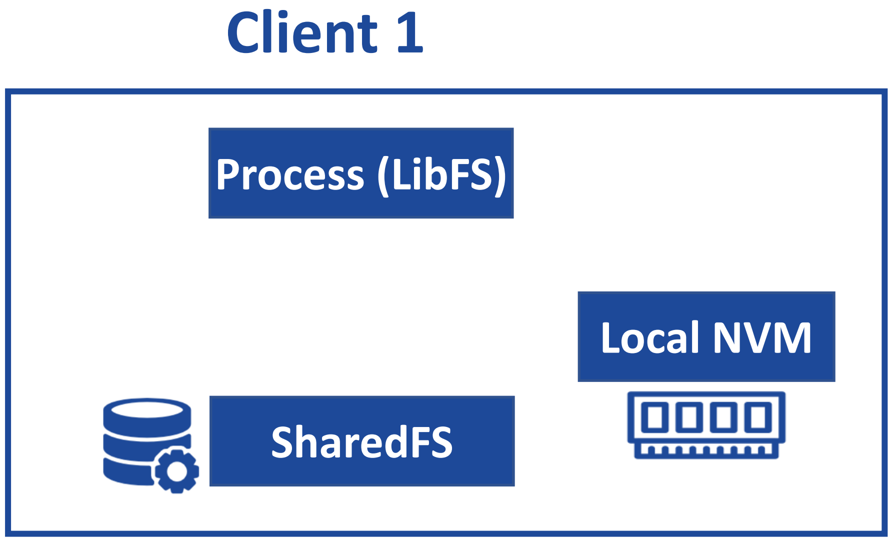


❹ **Userspace chain-replication via RDMA**

Hot remote client caches (= fast fail-over)
Crash consistent via RDMA write order

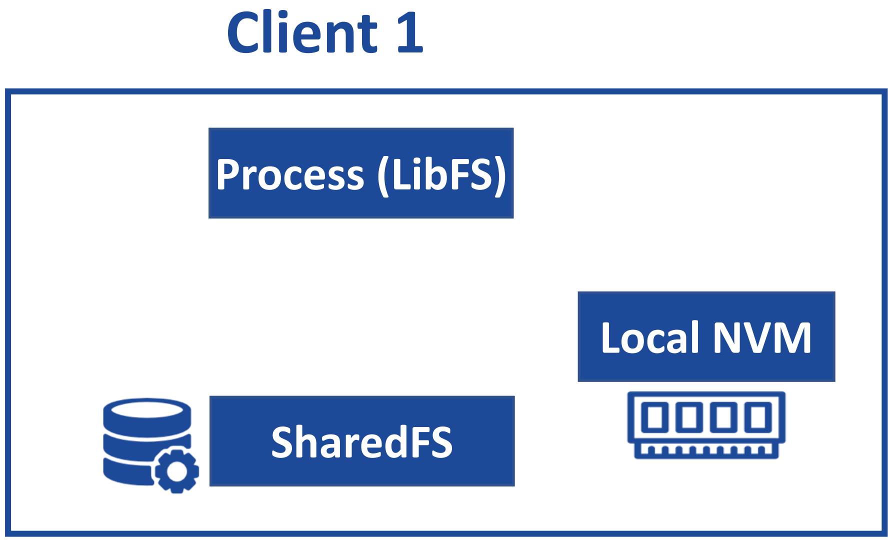
❺ **RPC via RDMA for read misses in client-local caches**

CC-NVM: Linearizability with client-local IO



❻ **CC-NVM provides linearizability by granting leases to processes**

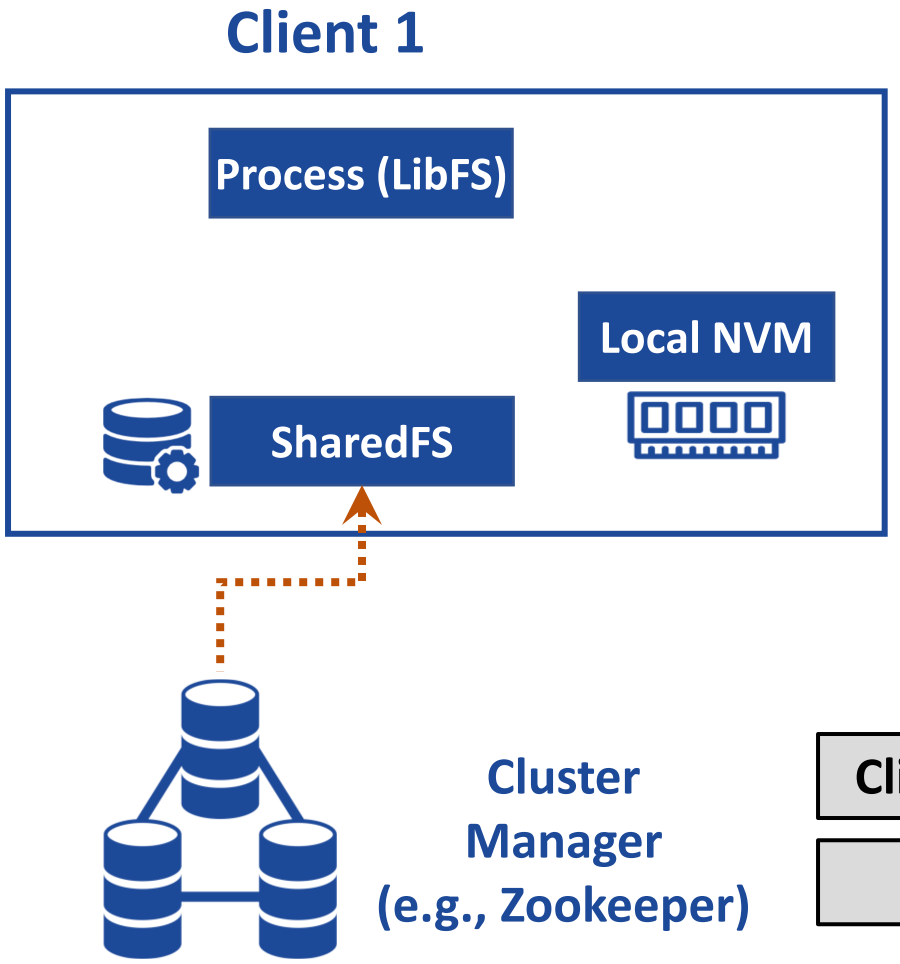
CC-NVM: Linearizability with client-local IO



❻ CC-NVM provides linearizability by granting leases to processes

Example: **Create file /home/alice/mail1**

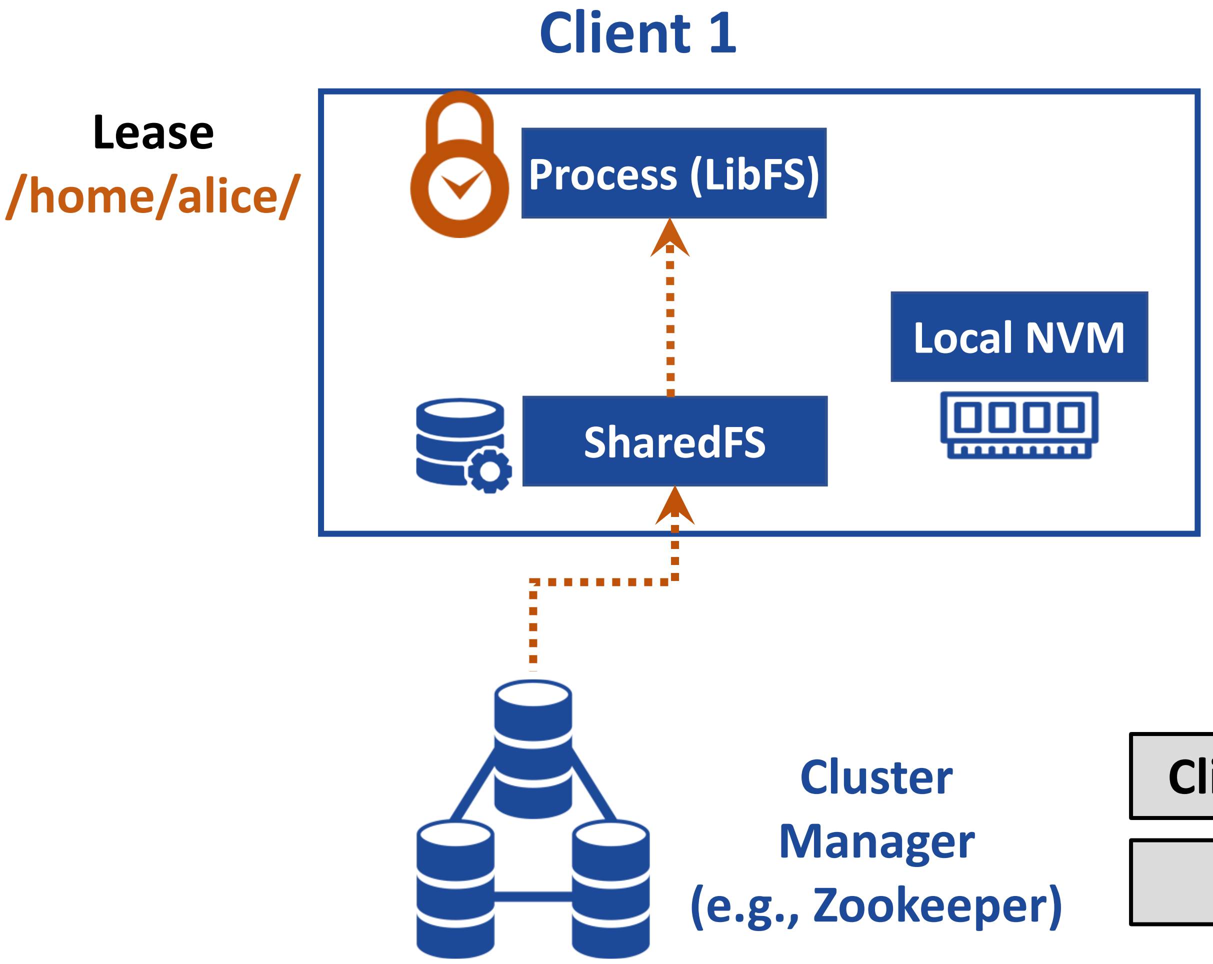
CC-NVM: Linearizability with client-local IO



❻ **CC-NVM provides linearizability by granting leases to processes**

Example: Create file /home/alice/mail1

CC-NVM: Linearizability with client-local IO



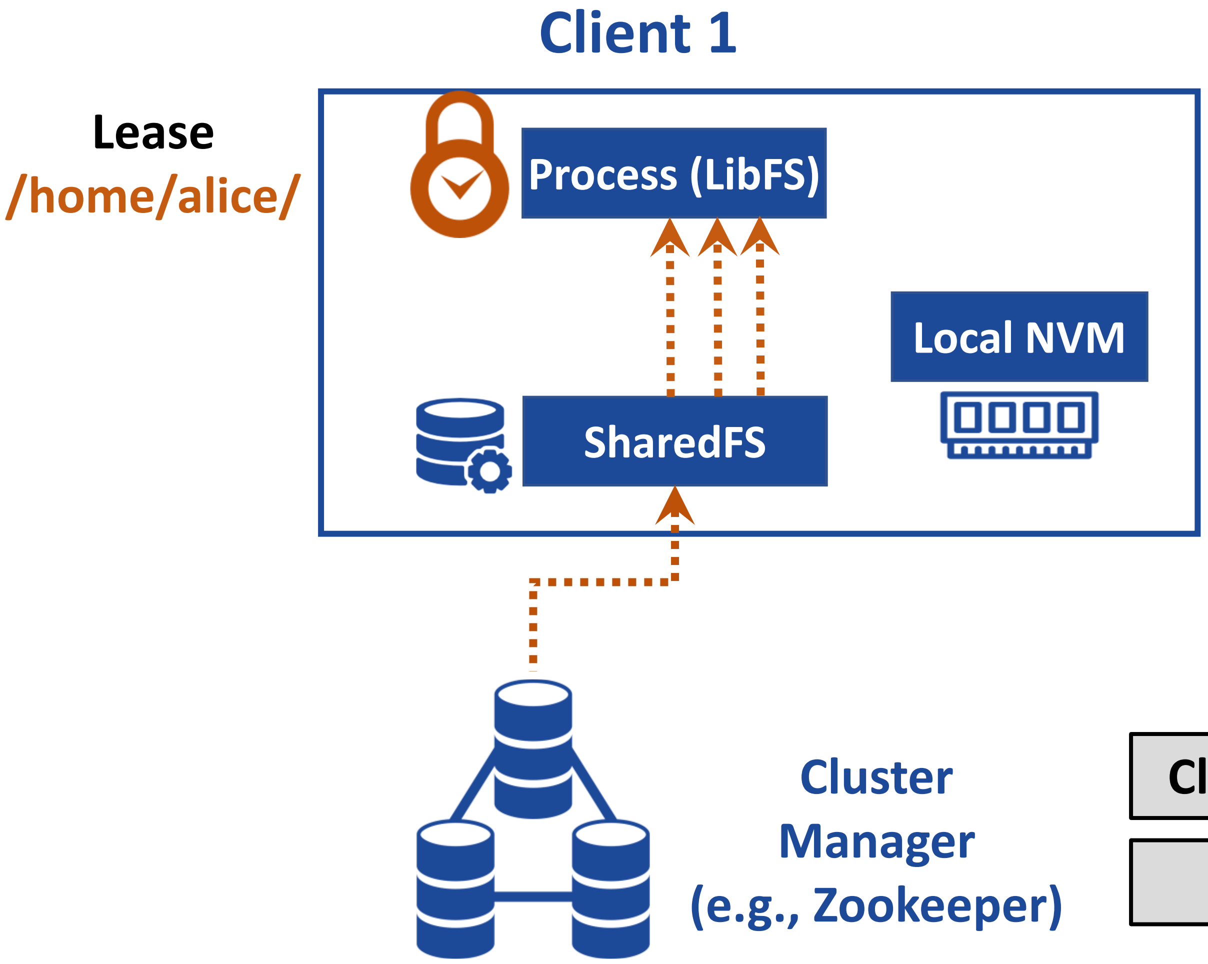
❻ **CC-NVM provides linearizability by granting leases to processes**

Example: Create file /home/alice/mail1

Client membership & data location services

Lease management

CC-NVM: Linearizability with client-local IO

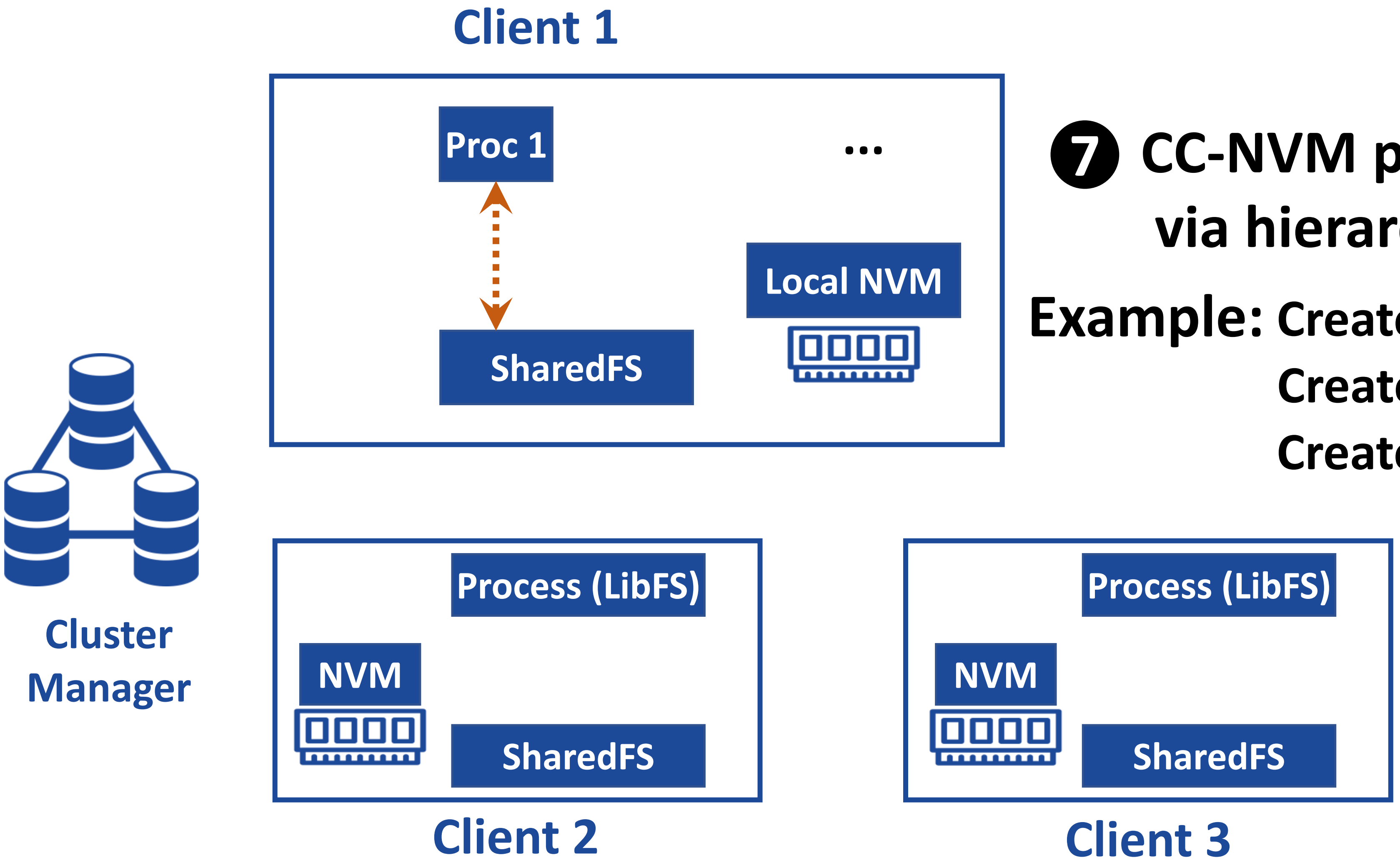


❻ CC-NVM provides linearizability by granting leases to processes

Example: Create file /home/alice/mail1
Create file /home/alice/mail2
Create file /home/alice/mail3

Client membership & data location services
Lease management

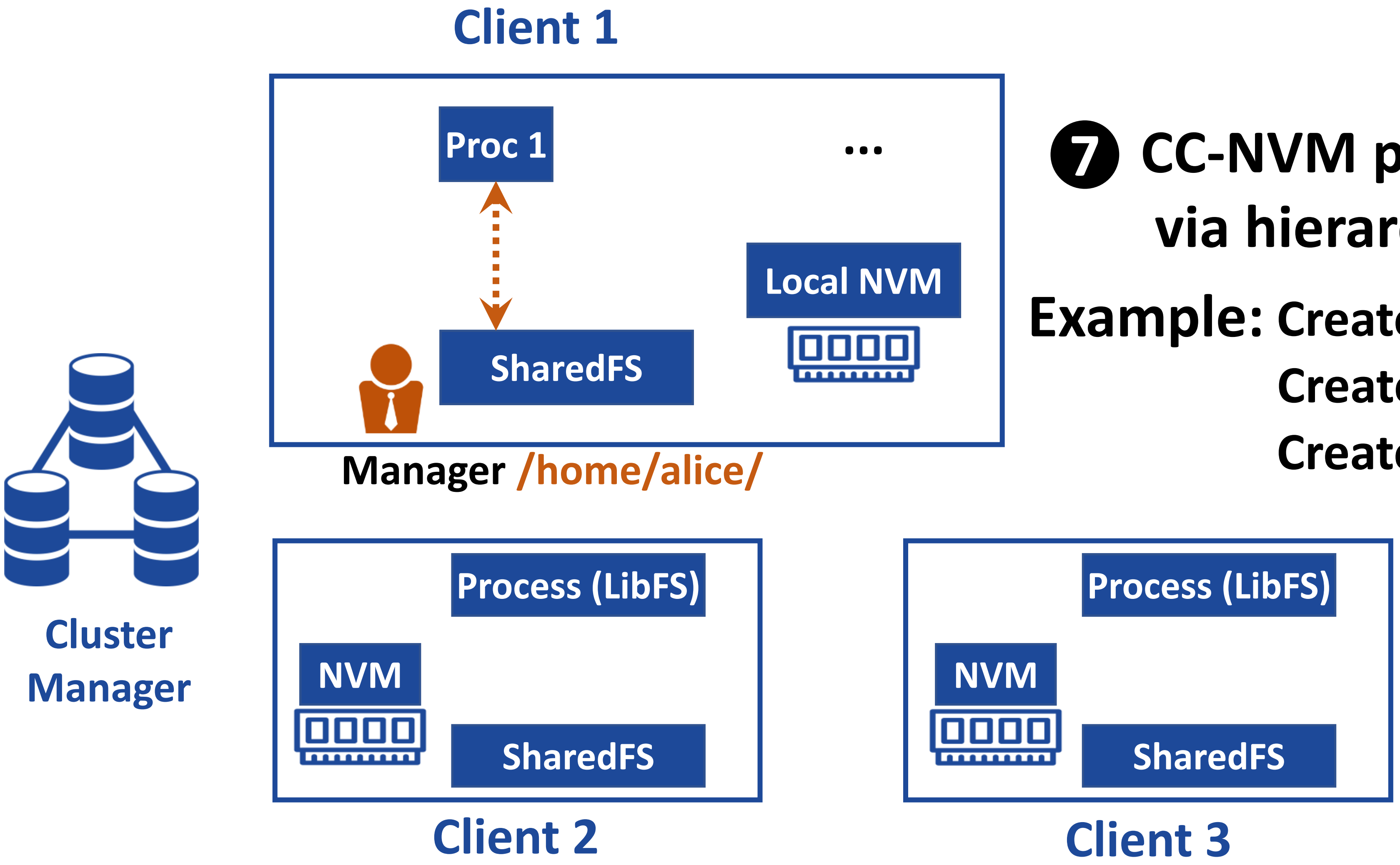
CC-NVM: Scalable shared file system access



❼ CC-NVM provides scalable access via hierarchical lease delegation

Example: Create file /home/alice/mail1
Create file /home/alice/mail2
Create file /home/alice/mail3

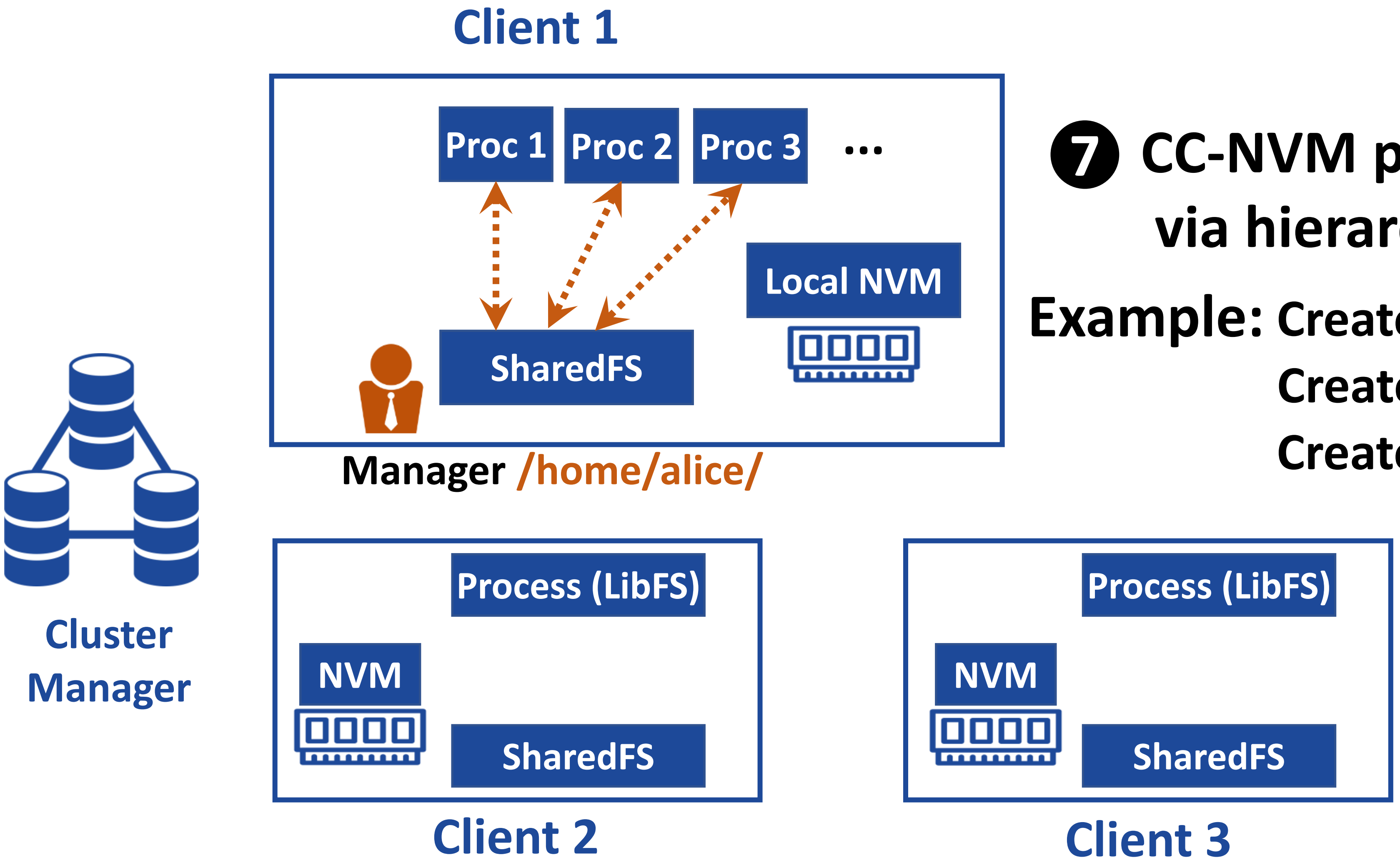
CC-NVM: Scalable shared file system access



❼ CC-NVM provides scalable access via hierarchical lease delegation

Example: Create file /home/alice/mail1
Create file /home/alice/mail2
Create file /home/alice/mail3

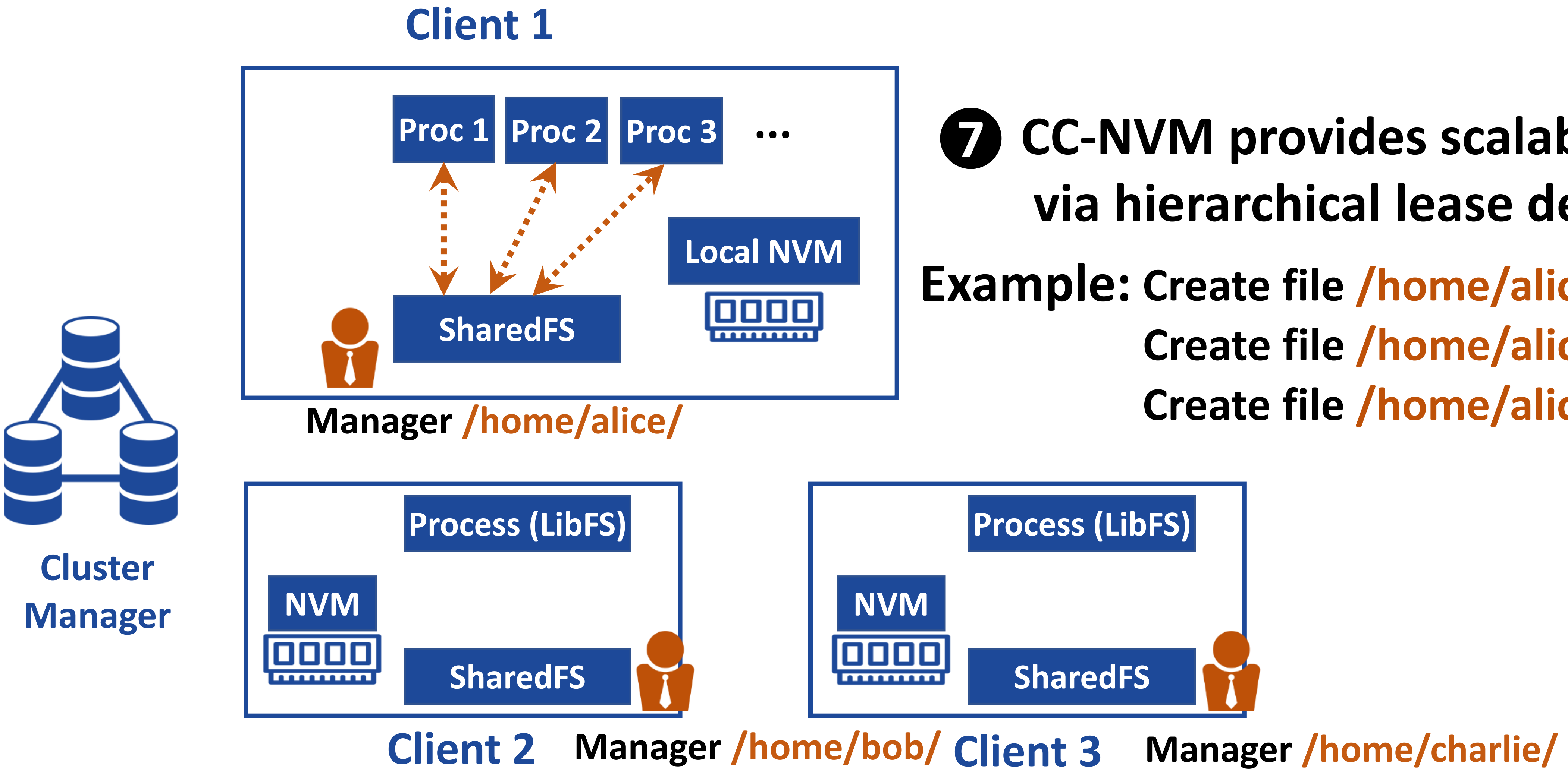
CC-NVM: Scalable shared file system access



❼ CC-NVM provides scalable access via hierarchical lease delegation

Example: Create file /home/alice/mail1
Create file /home/alice/mail2
Create file /home/alice/mail3

CC-NVM: Scalable shared file system access



❼ **CC-NVM provides scalable access via hierarchical lease delegation**

Example: Create file /home/alice/mail1
Create file /home/alice/mail2
Create file /home/alice/mail3

Assise: Design summary

Main principle: **Maximize use of client-local NVM**

1. **Low latency IO**: Perform process-local & client-local IO
2. **Linearizability and data crash consistency**
   CC-NVM distributed client-side NVM coherence protocol
3. **High availability**
   Fail-over to cache-hot client replicas, recover client NVM caches
4. Exploit NUMA locality
5. Efficient NVM and network bandwidth use
6. Low-cost cold storage

Assise: Design summary

Main principle: **Maximize use of client-local NVM**

1. **Low latency IO**: Perform process-local & client-local IO
2. **Linearizability and data crash consistency**
   CC-NVM distributed client-side NVM coherence protocol
3. **High availability**
   Fail-over to cache-hot client replicas, recover client NVM caches
4. Exploit NUMA locality
5. Efficient NVM and network bandwidth use
6. Low-cost cold storage

Evaluation

Evaluation questions

- **Latency:** How close is Assise to raw, replicated NVM IO latency? How much faster is Assise than the state-of-the-art?

- **Scalability:** By how much can CC-NVM improve multi-process and multi-node scalability?

- **Availability:** How quickly can Assise fail-over?

- More results in the paper
  - Large-scale distributed storage, cloud application benchmarks, re-establishing the replication factor, …

Experimental Setup

- **5× dual-socket** Cascade Lake-SP servers **with 48 cores @ 2.2GHz**
  - All nodes connect to an InfiniBand switch via **40 Gbps ConnectX-3 NICs**
- Each node has **6 TB of Optane DC NVM** and **384 GB of DRAM**
  - 6x DIMMs of NVM and DRAM per NUMA node

Experimental baselines – Properties

				ATC’17	FAST’19
Feature	**Assise**	**CephFS**	**NFS**	**Octopus**	**Orion**
Local consistency	✓				
Kernel bypass	✓				
Linearizability	✓				✓
Byte-oriented	✓			✓	✓
Replication	✓	✓			✓
RDMA	✓	✓	✓	✓	✓

Experimental baselines – Properties

Feature	Assise	CephFS	NFS	Octopus (ATC'17)	Orion (FAST'19)
Local consistency	✓				
Kernel bypass	✓				
Linearizability	✓				✓
Byte-oriented	✓			✓	✓
Replication	✓	✓			✓
RDMA	✓	✓	✓	✓	✓

Experimental baselines – Properties

Feature	Assise	CephFS	NFS	Octopus (ATC'17)	Orion (FAST'19)
Local consistency	✓				
Kernel bypass	✓				
Linearizability	✓				✓
Byte-oriented	✓			✓	✓
Replication	✓	✓			✓
RDMA	✓	✓	✓	✓	✓

Experimental baselines – Properties

				ATC’17	FAST’19
Feature	**Assise**	**CephFS**	**NFS**	**Octopus**	**Orion**
Local consistency	✓				
Kernel bypass	✓				
Linearizability	✓				✓
Byte-oriented	✓			✓	✓
Replication	✓	✓			✓
RDMA	✓	✓	✓	✓	✓

Experimental baselines – Properties

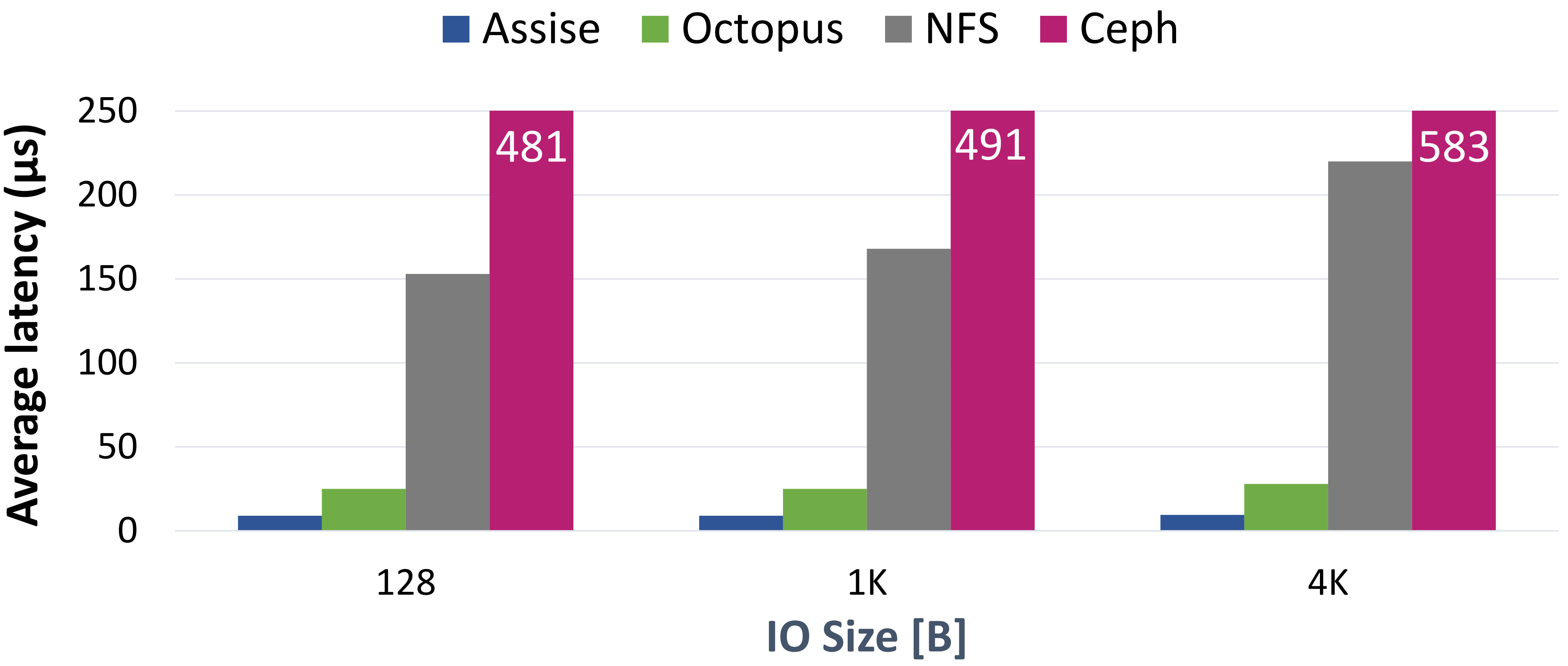
Feature	Assise	CephFS	NFS	Octopus ATC'17	Orion FAST'19
Local consistency	✓				
Kernel bypass	✓				
Linearizability	✓				✓
Byte-oriented	✓			✓	✓
Replication	✓	✓			✓
RDMA	✓	✓	✓	✓	✓

Experimental baselines – Properties

Feature	Assise	CephFS	NFS	Octopus (ATC'17)	Orion (FAST'19)
Local consistency	✓				
Kernel bypass	✓				
Linearizability	✓				✓
Byte-oriented	✓			✓	✓
Replication	✓	✓			✓
RDMA	✓	✓	✓	✓	✓

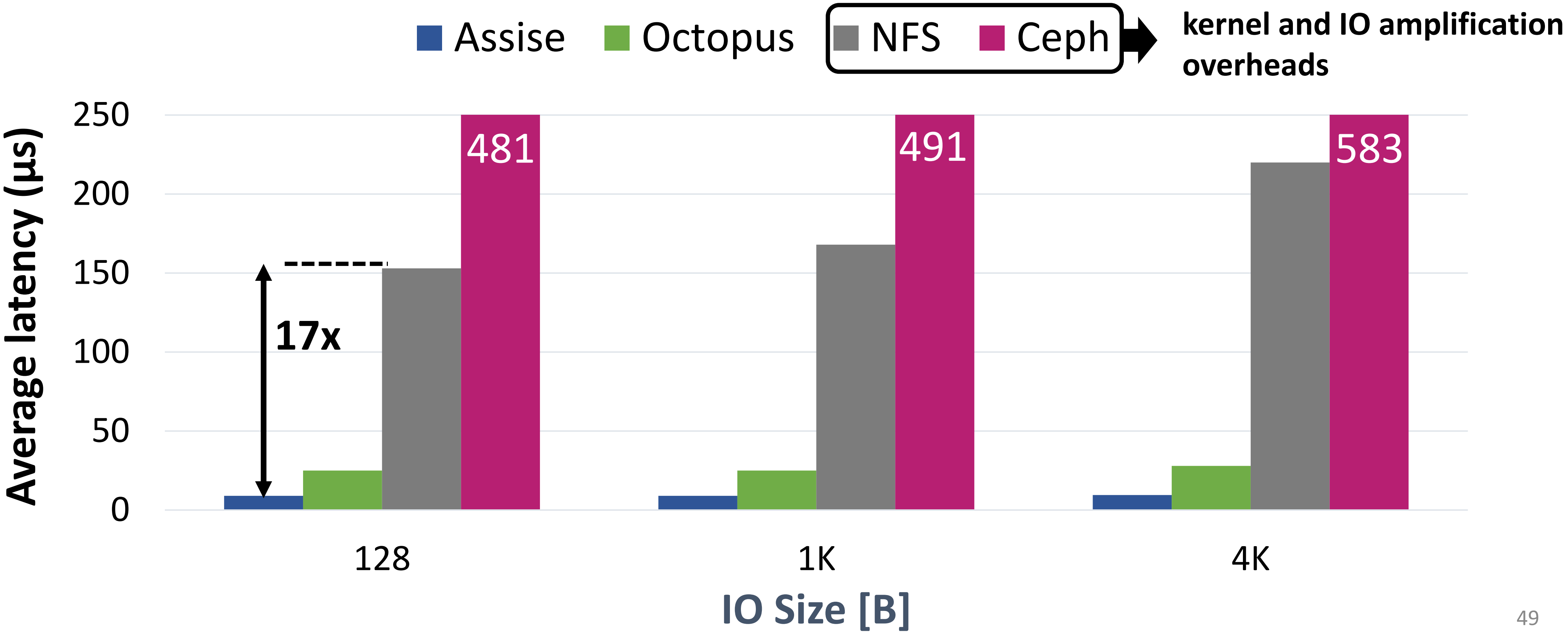
Microbenchmark: Write Latency

- 1GB burst of synchronous writes to a single file
  - Assise and Ceph replicate to an additional node



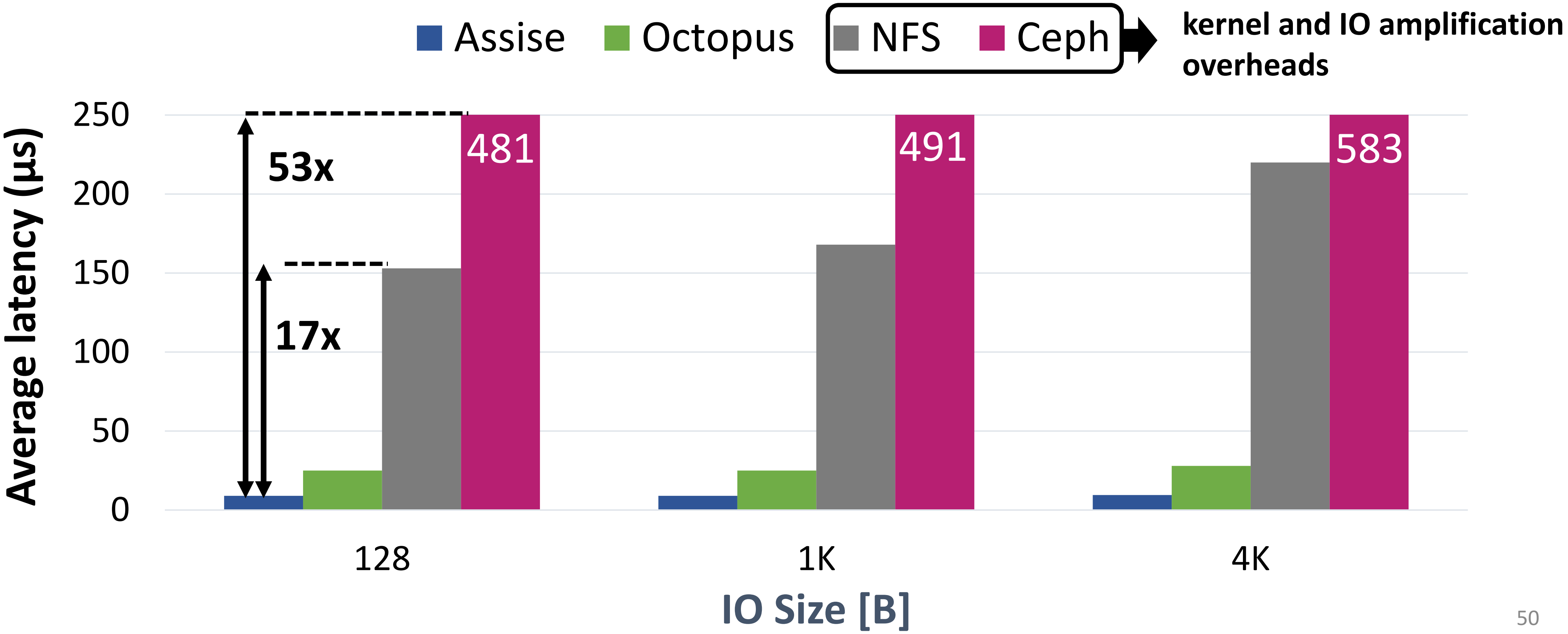
Microbenchmark: Write Latency

- 1GB burst of synchronous writes to a single file
  - Assise and Ceph replicate to an additional node



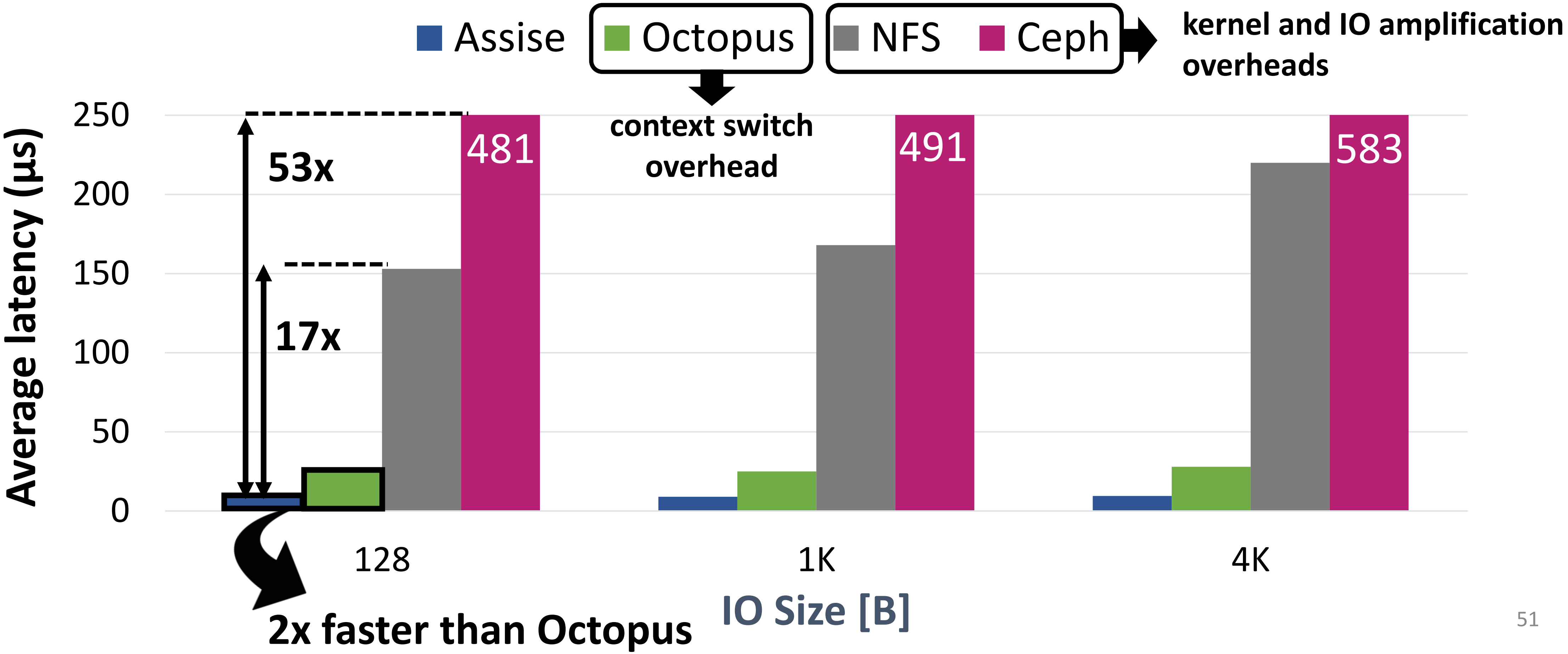
Microbenchmark: Write Latency

- 1GB burst of synchronous writes to a single file
  - Assise and Ceph replicate to an additional node



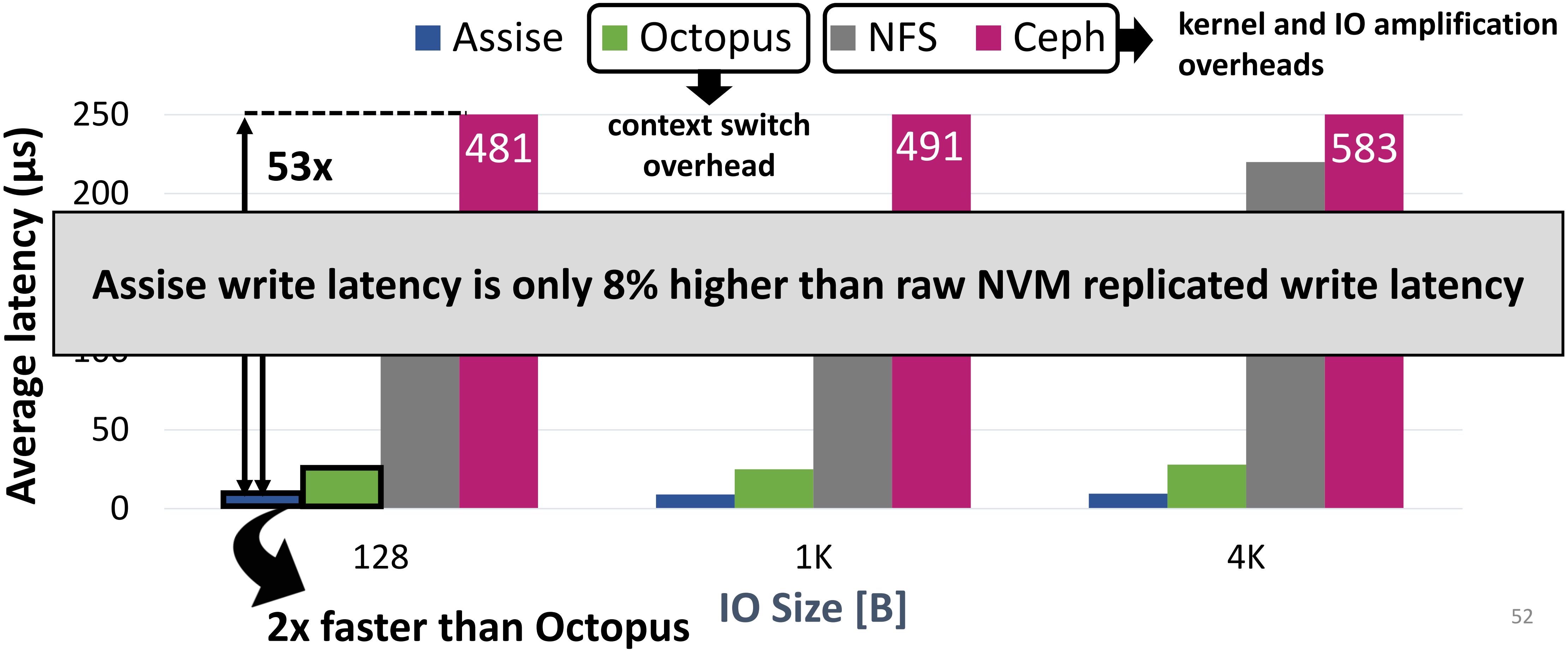
Microbenchmark: Write Latency

- 1GB burst of synchronous writes to a single file
  - Assise and Ceph replicate to an additional node



Microbenchmark: Write Latency

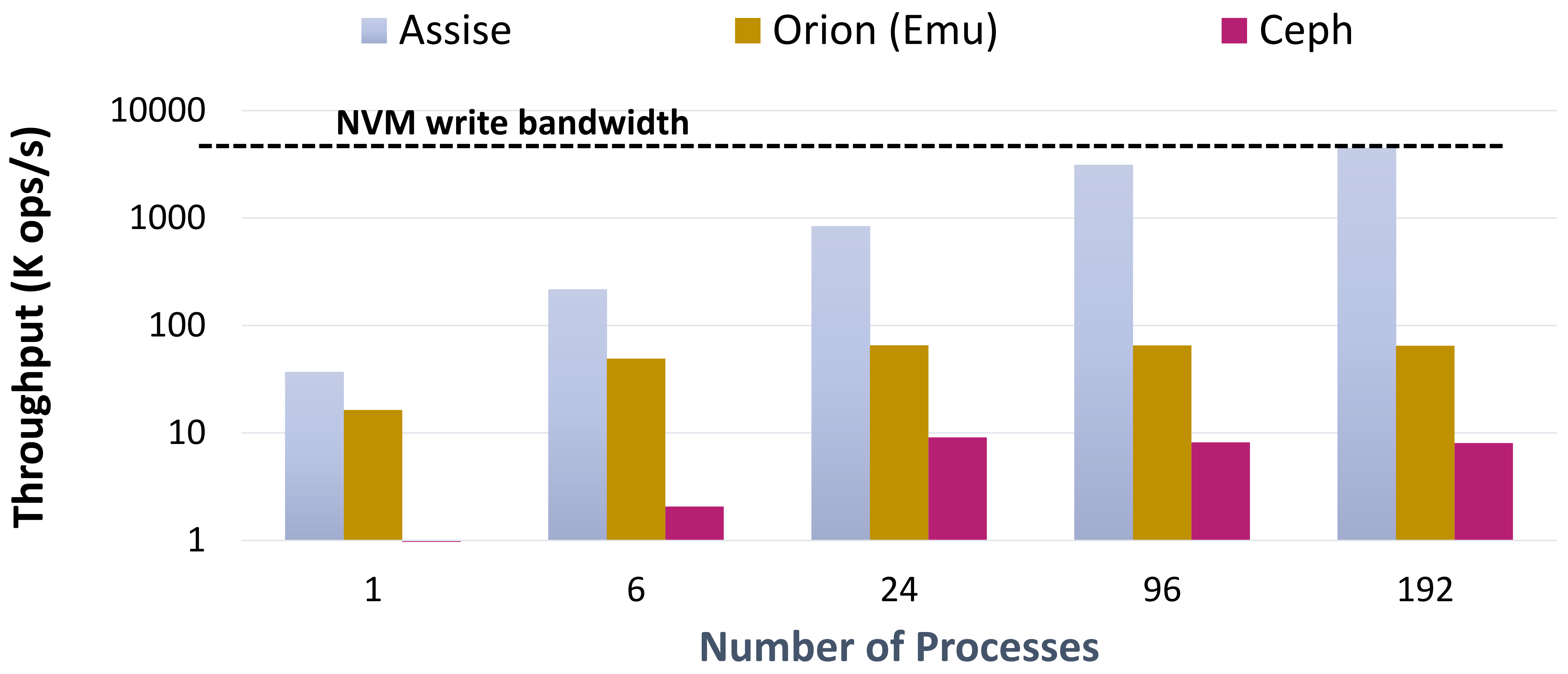
- 1GB burst of synchronous writes to a single file
  - Assise and Ceph replicate to an additional node



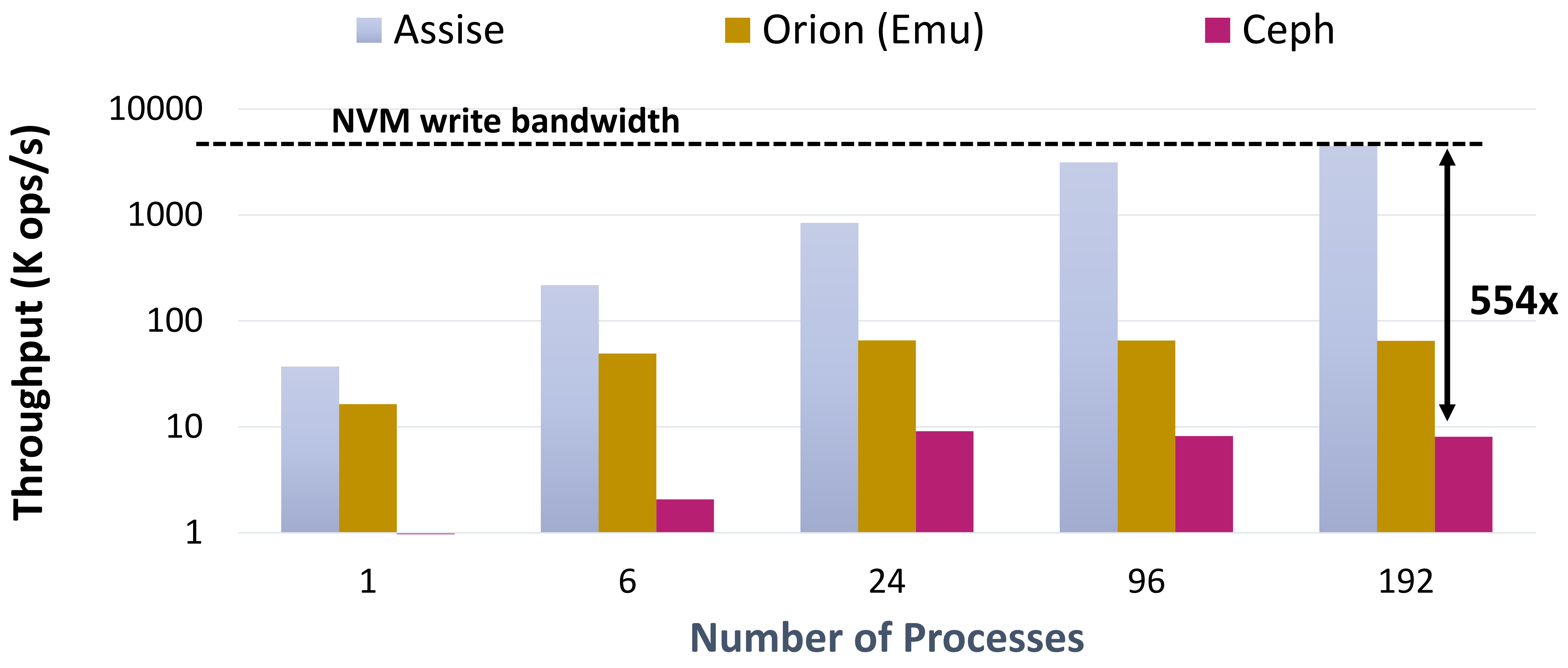
Microbenchmark: Scalability

- Multi-process benchmark to test scalability
  - Data & metadata operations (create file, write 4K, rename file)
  - **Embarrassingly parallel**: Processes write to their own private sub-directories
  - Replication is disabled: Eliminate network bottlenecks of our testbed
- Benchmark runs on 3 testbed nodes
  - Balanced deployment of processes over nodes

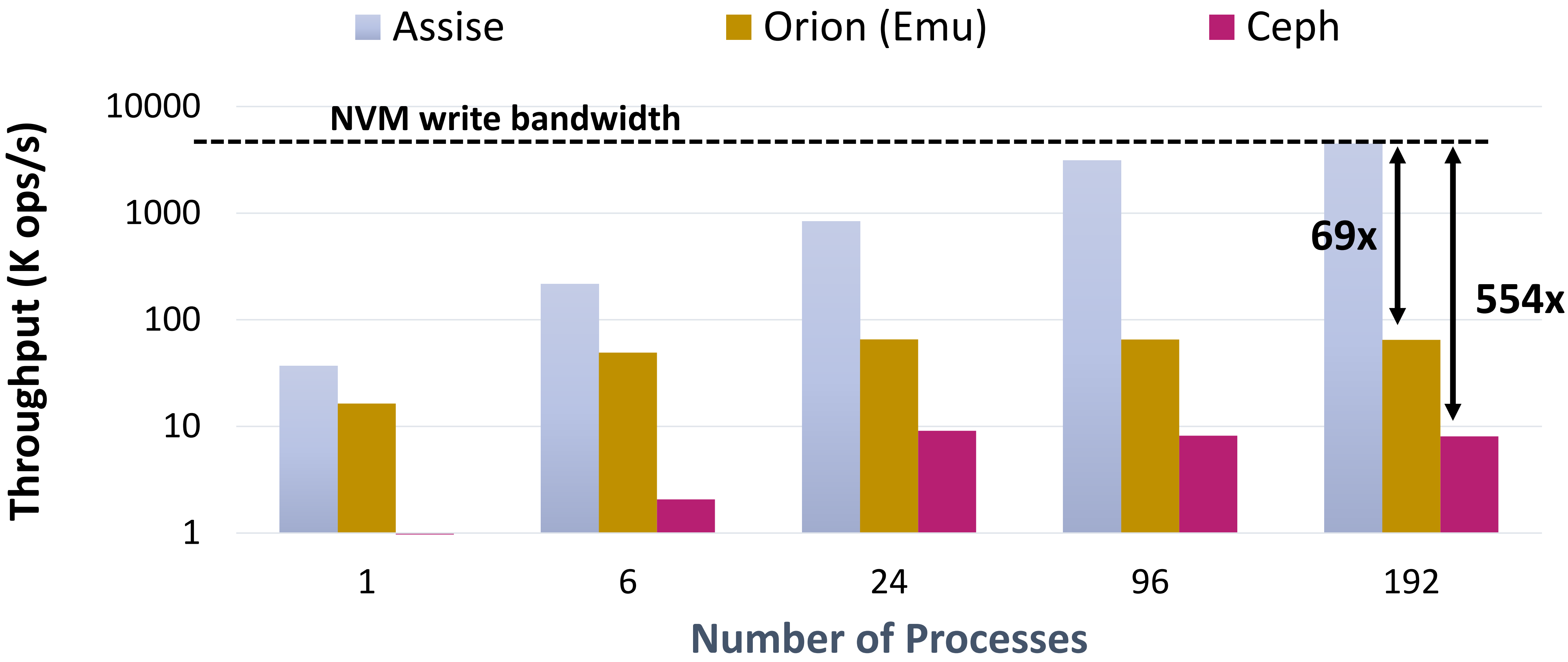
Microbenchmark: Scalability



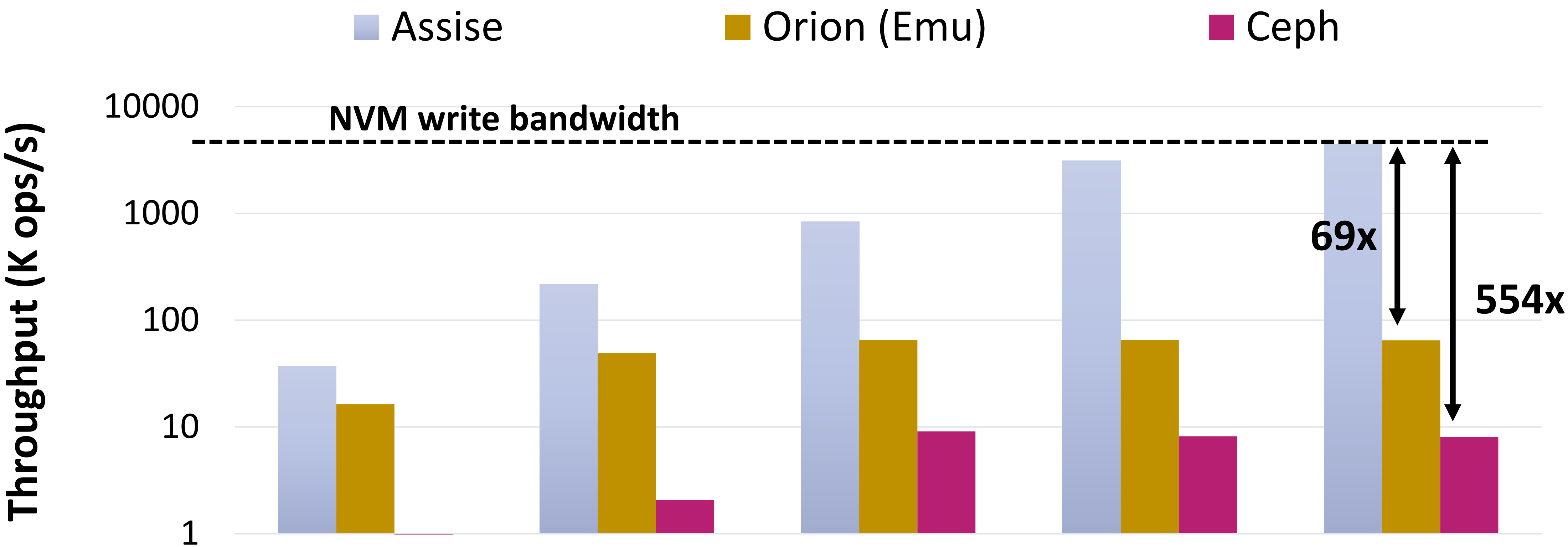
Microbenchmark: Scalability



Microbenchmark: Scalability



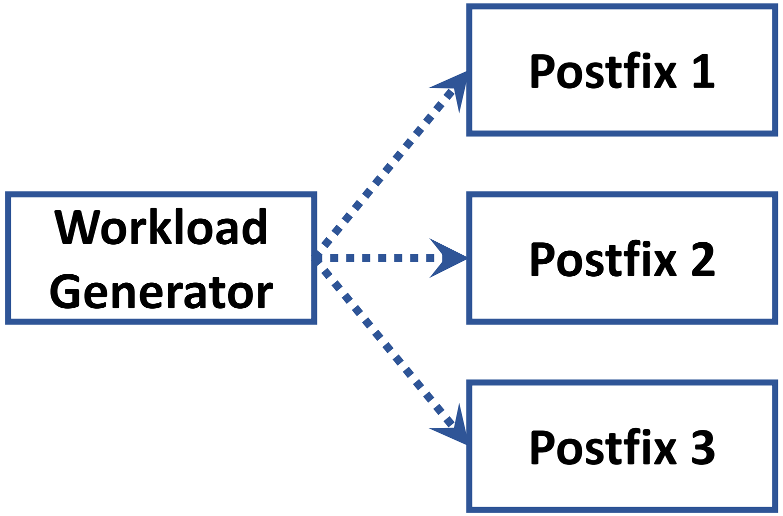
Microbenchmark: Scalability



Assise scales because of client-local synchronization, provided by CC-NVM

Postfix: Scalable Mail Delivery

- We replay 70GB worth of emails to a pool of Postfix Mail Delivery Agents
  - Deliver to Maildir: Each inbox is a directory, contains 1 file per mail message
  - Real email trace; average mail size is 200 KB
- We use 3 nodes for mail delivery and 1 node for workload generation
  - Delivery processes are distributed uniformly over nodes
- Mail delivery load balancing
  1. Round robin
  2. Sharded across clients by recipient

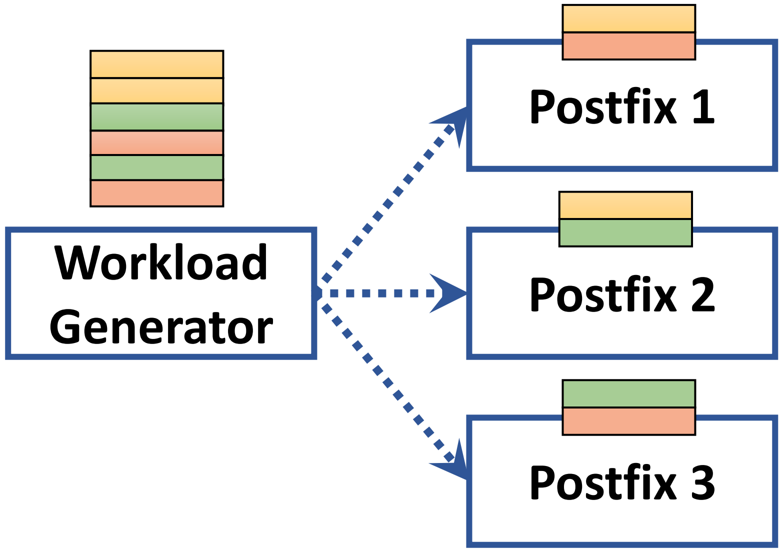


Postfix: Scalable Mail Delivery

- We replay 70GB worth of emails to a pool of Postfix Mail Delivery Agents
  - Deliver to Maildir: Each inbox is a directory, contains 1 file per mail message
  - Real email trace; average mail size is 200 KB

- We use 3 nodes for mail delivery and 1 node for workload generation
  - Delivery processes are distributed uniformly over nodes

- Mail delivery load balancing
  1. Round robin
  2. Sharded across clients by recipient

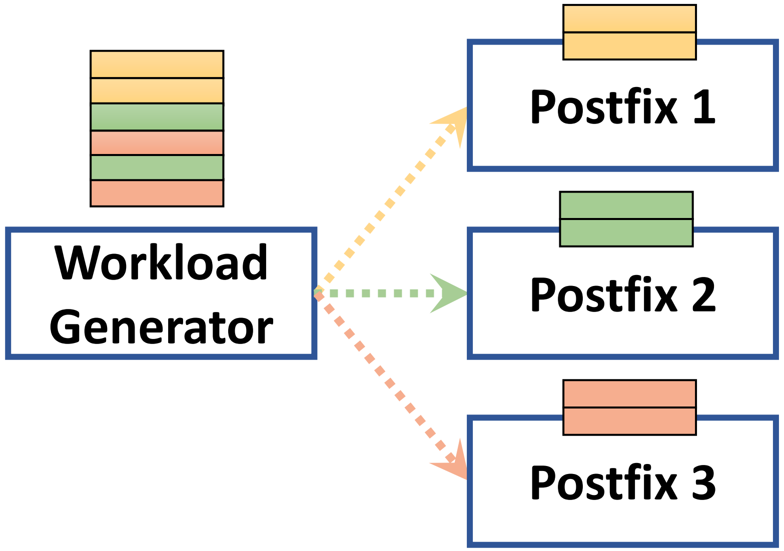


Postfix: Scalable Mail Delivery

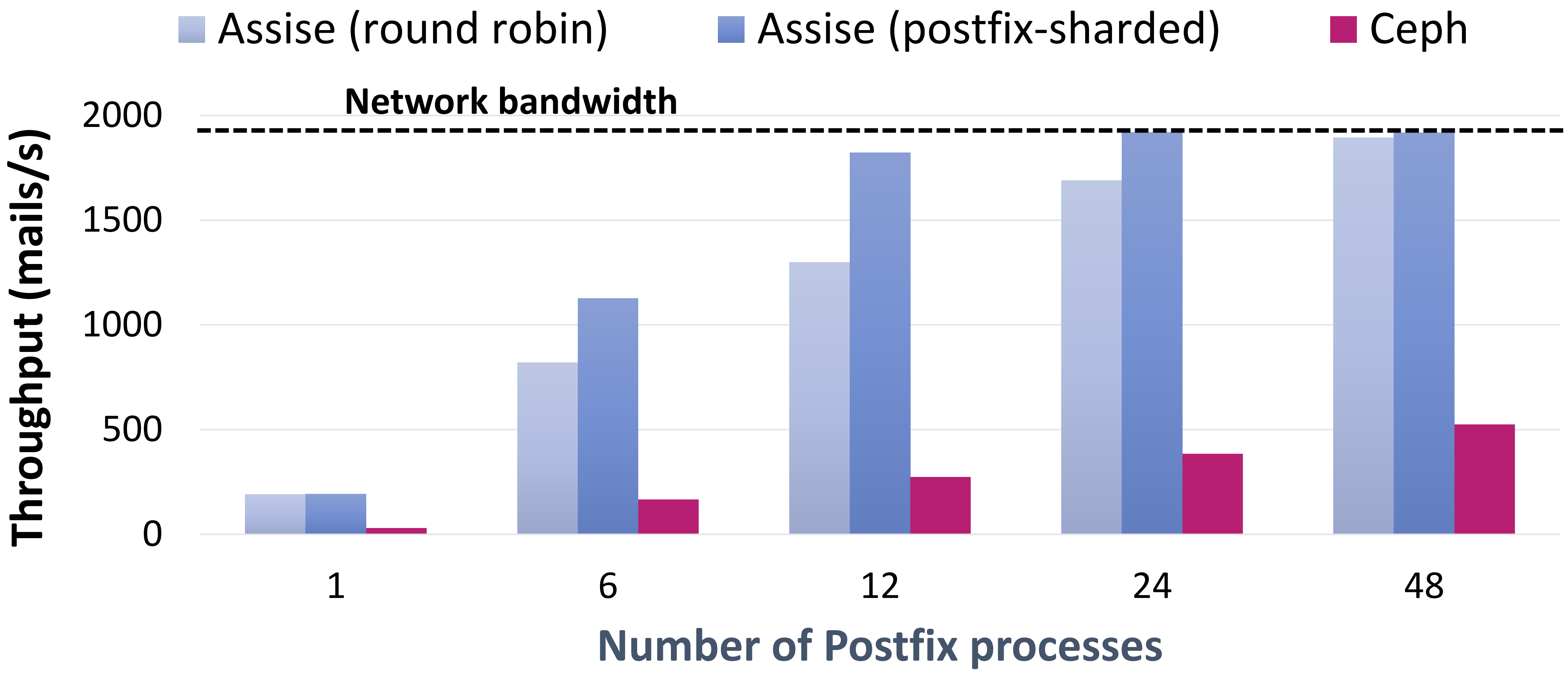
- We replay 70GB worth of emails to a pool of Postfix Mail Delivery Agents
  - Deliver to Maildir: Each inbox is a directory, contains 1 file per mail message
  - Real email trace; average mail size is 200 KB

- We use 3 nodes for mail delivery and 1 node for workload generation
  - Delivery processes are distributed uniformly over nodes

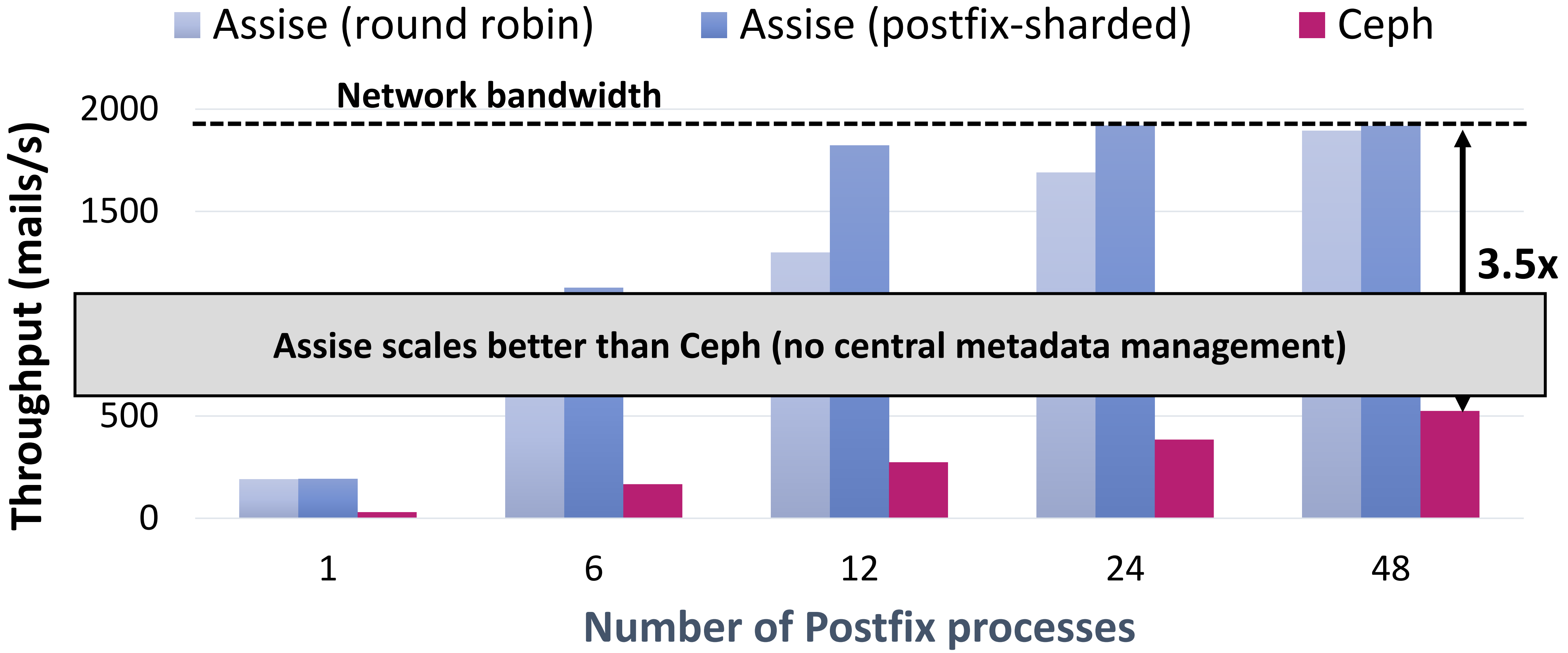
- Mail delivery load balancing
  1. Round robin
  2. Sharded across clients by recipient



Postfix: Scalable Mail Delivery

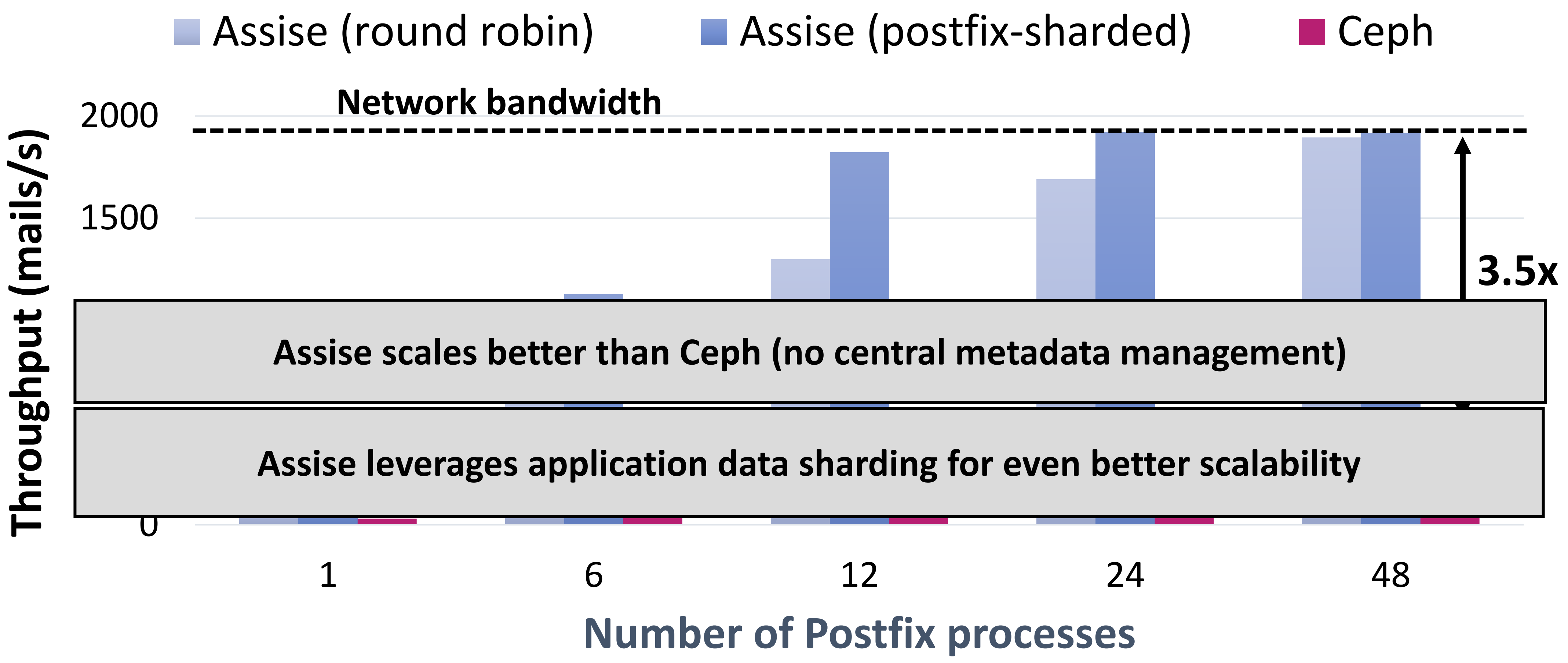


Postfix: Scalable Mail Delivery



Assise scales better than Ceph (no central metadata management)

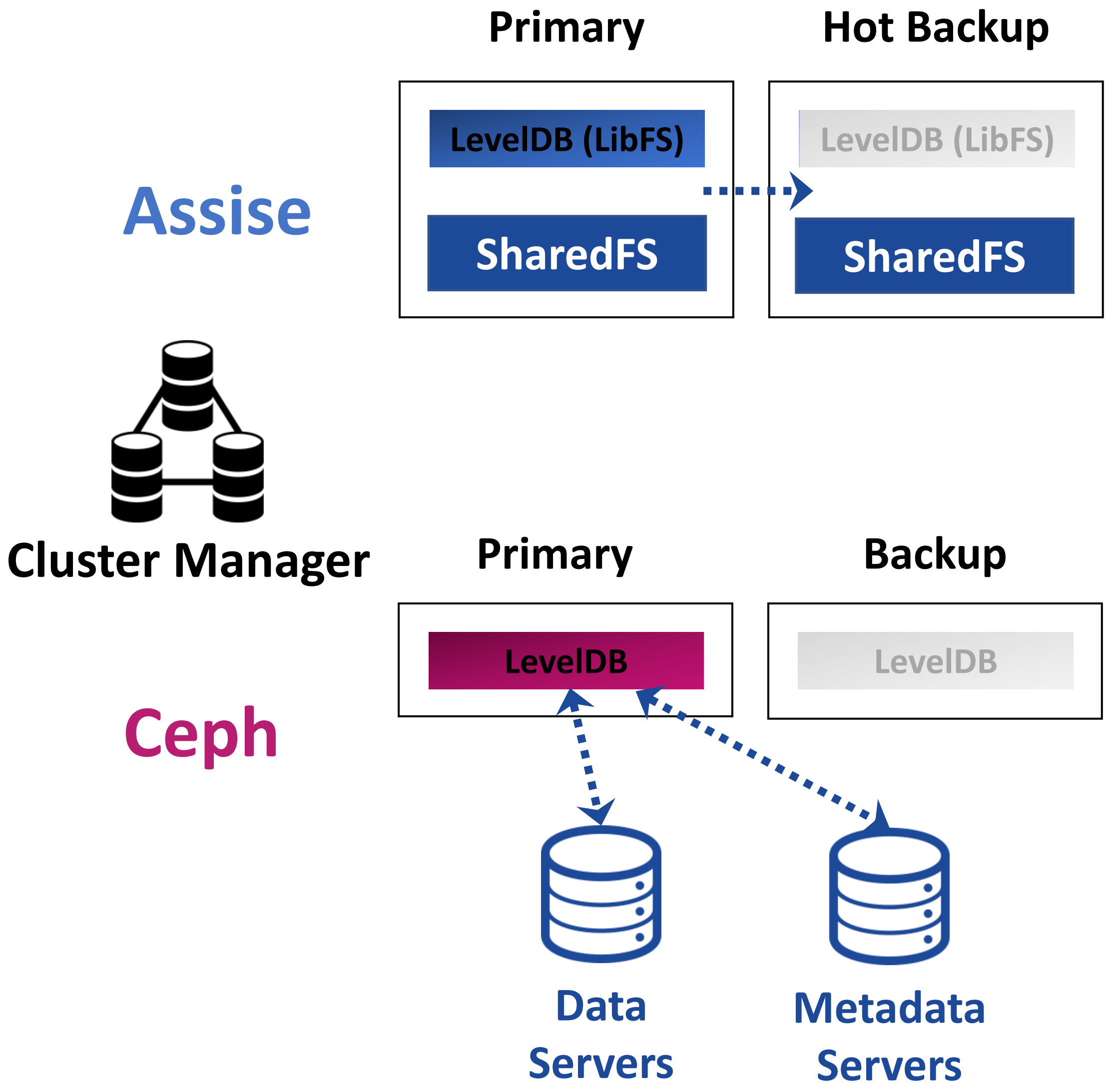
Postfix: Scalable Mail Delivery



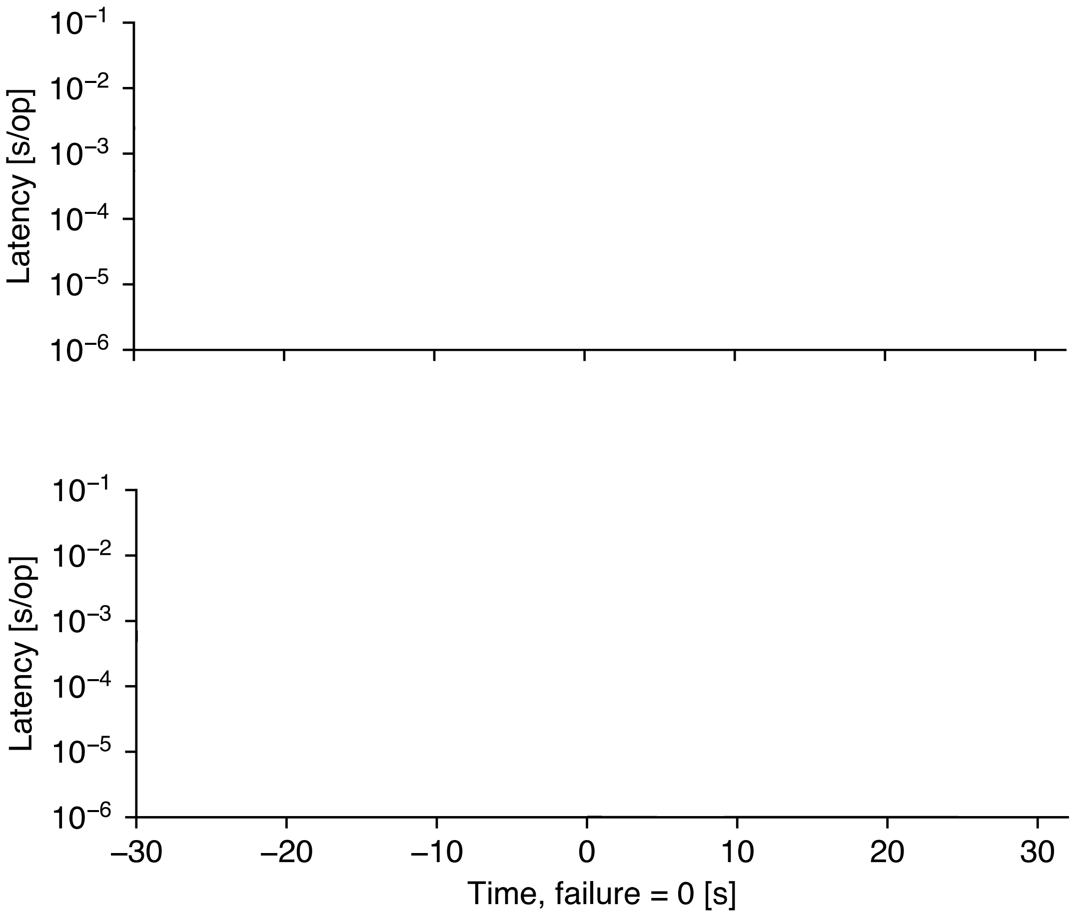
Assise scales better than Ceph (no central metadata management)

Assise leverages application data sharding for even better scalability

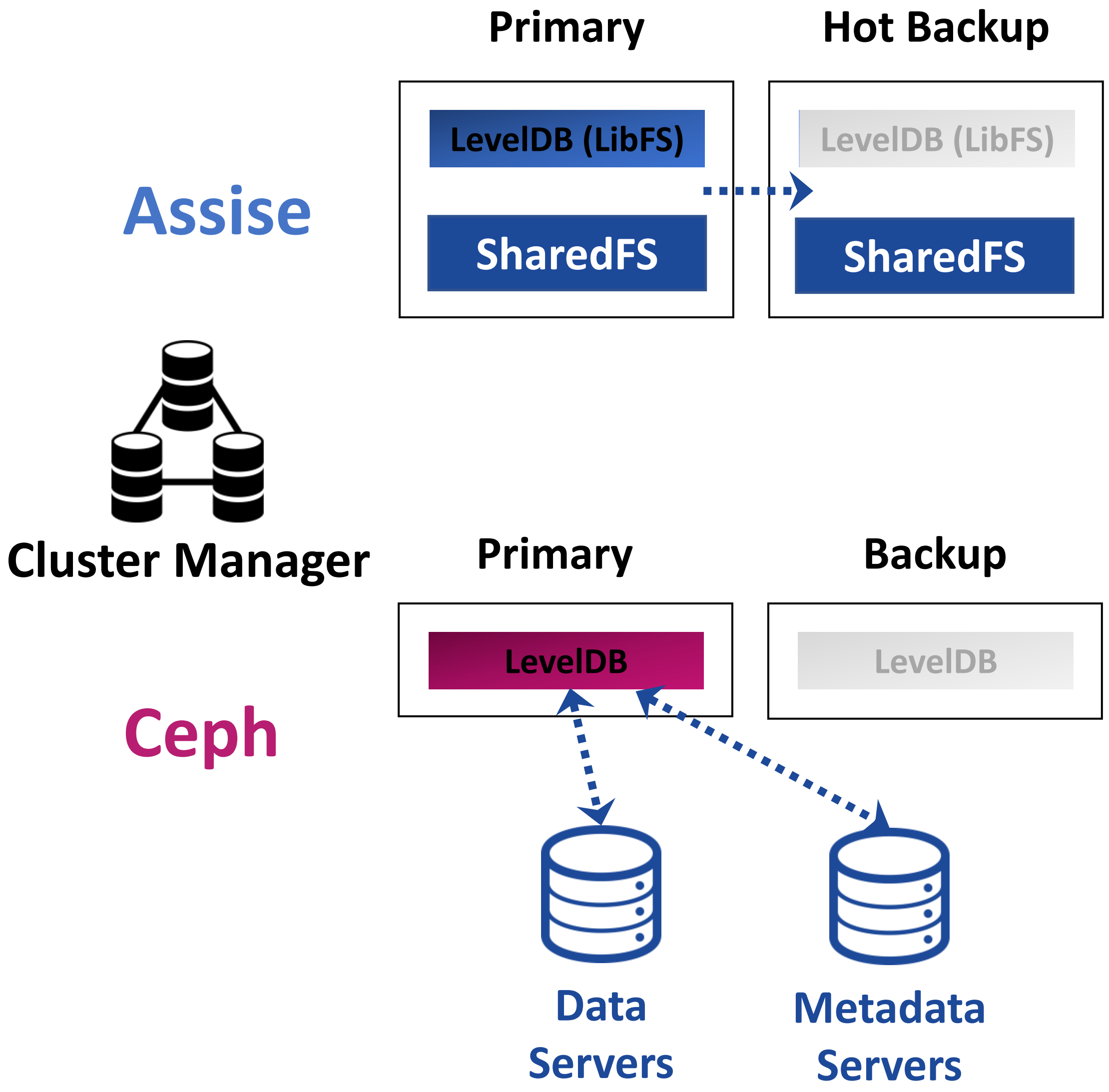
LevelDB: Fail-over to backup

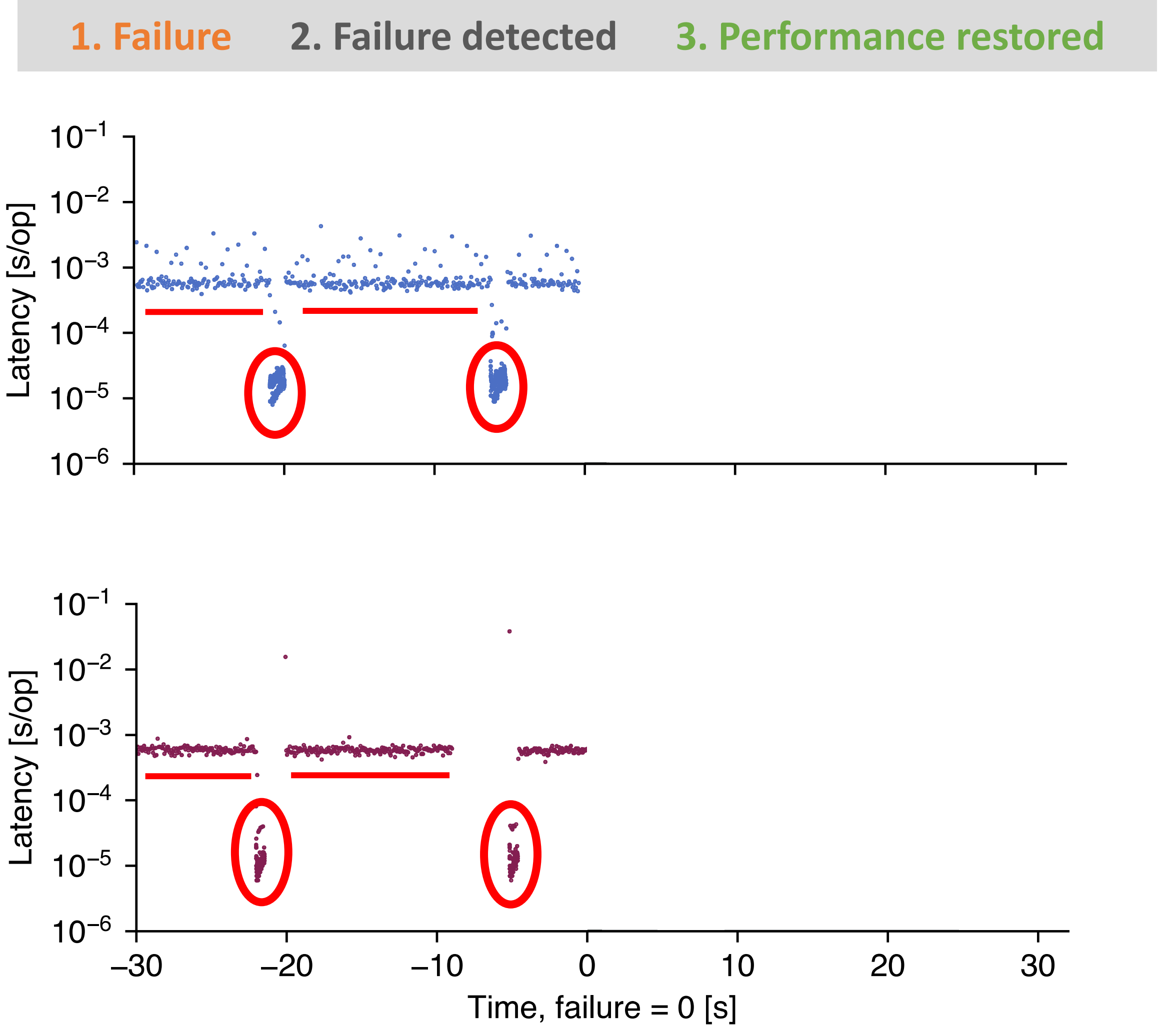


1. Failure 2. Failure detected 3. Performance restored

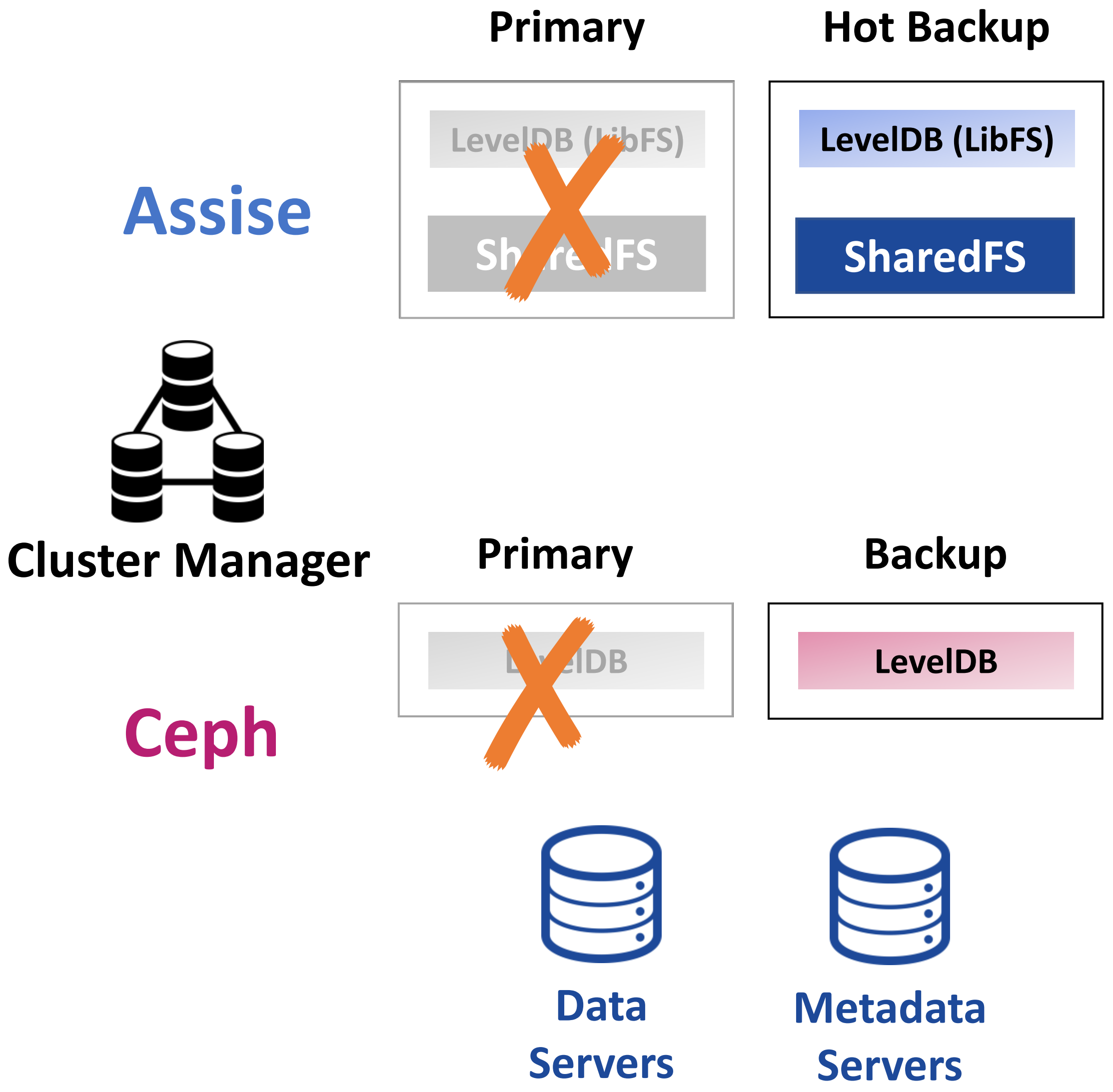


LevelDB: Fail-over to backup

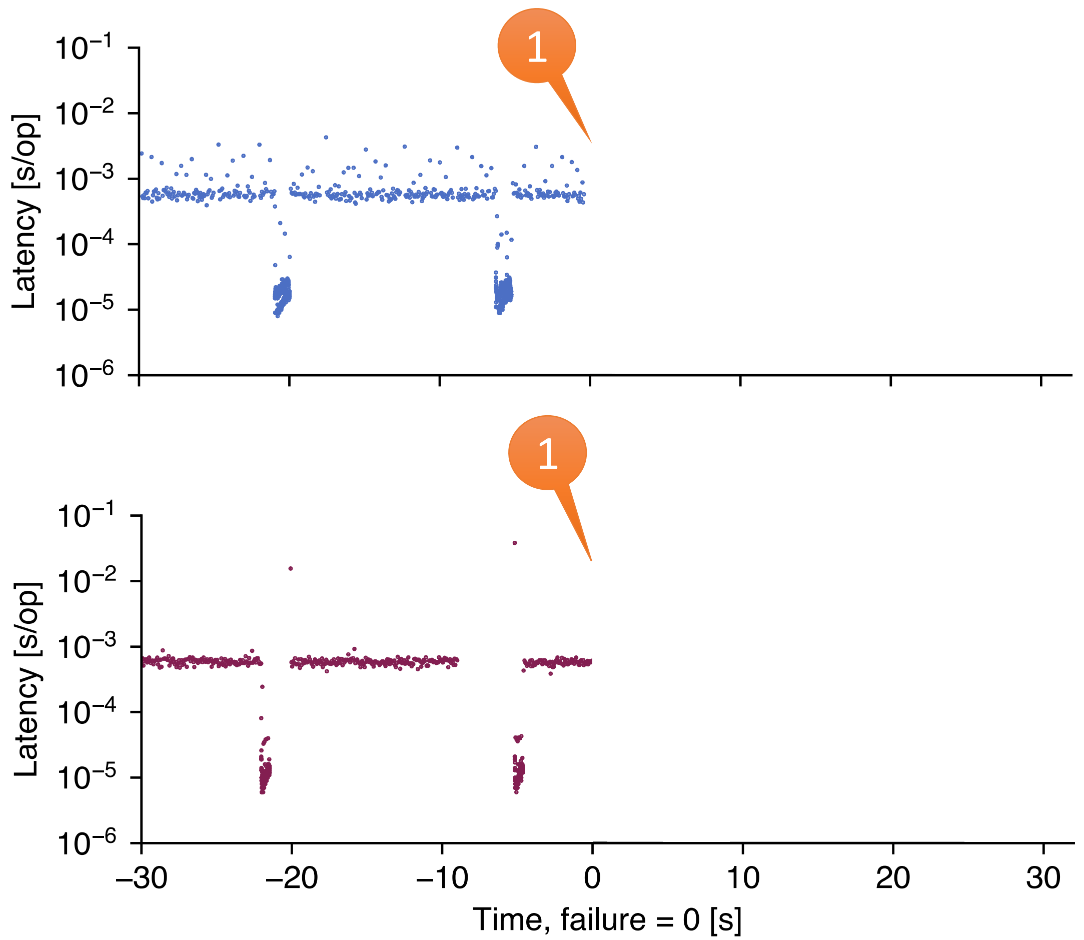




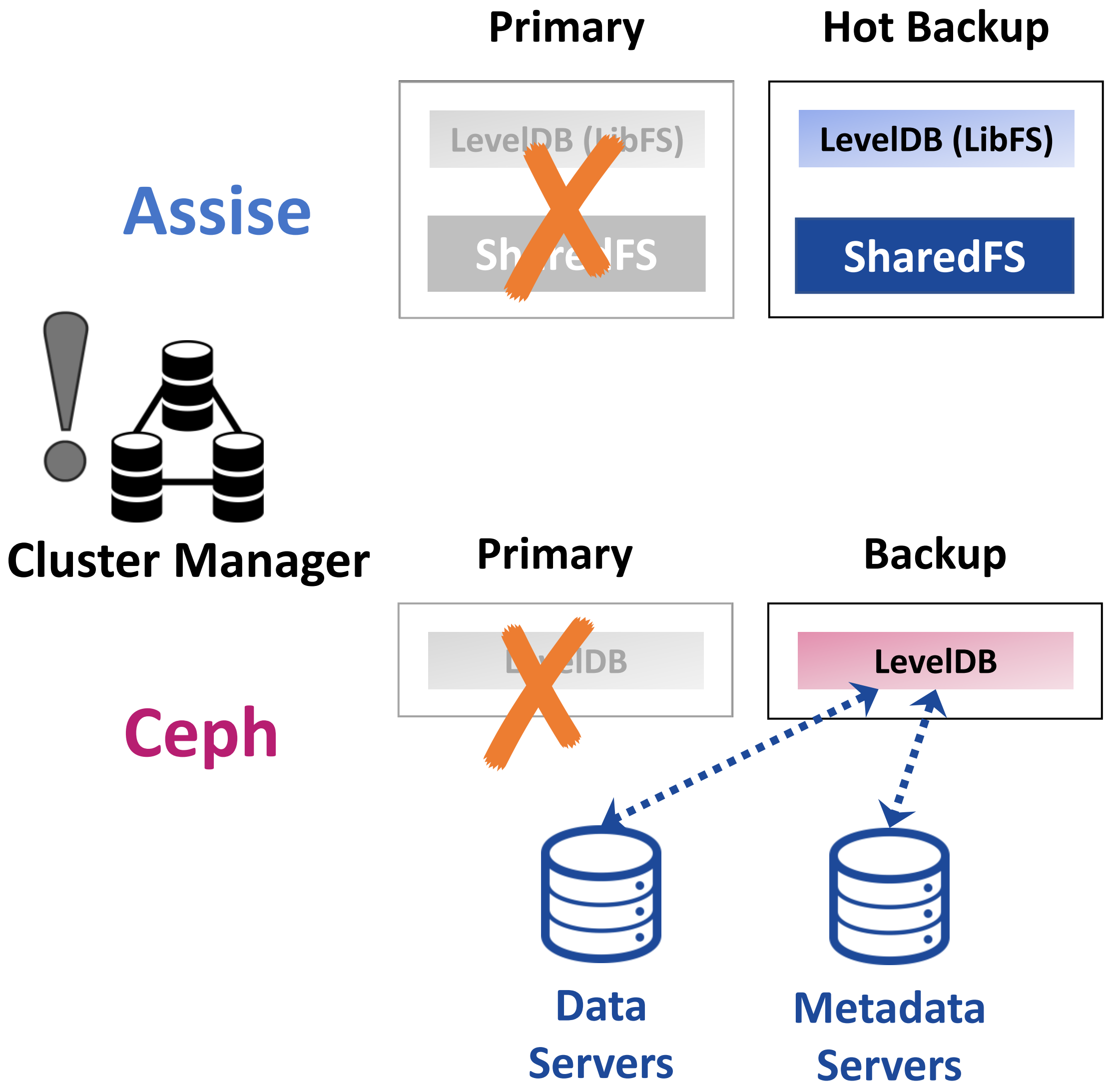
LevelDB: Fail-over to backup

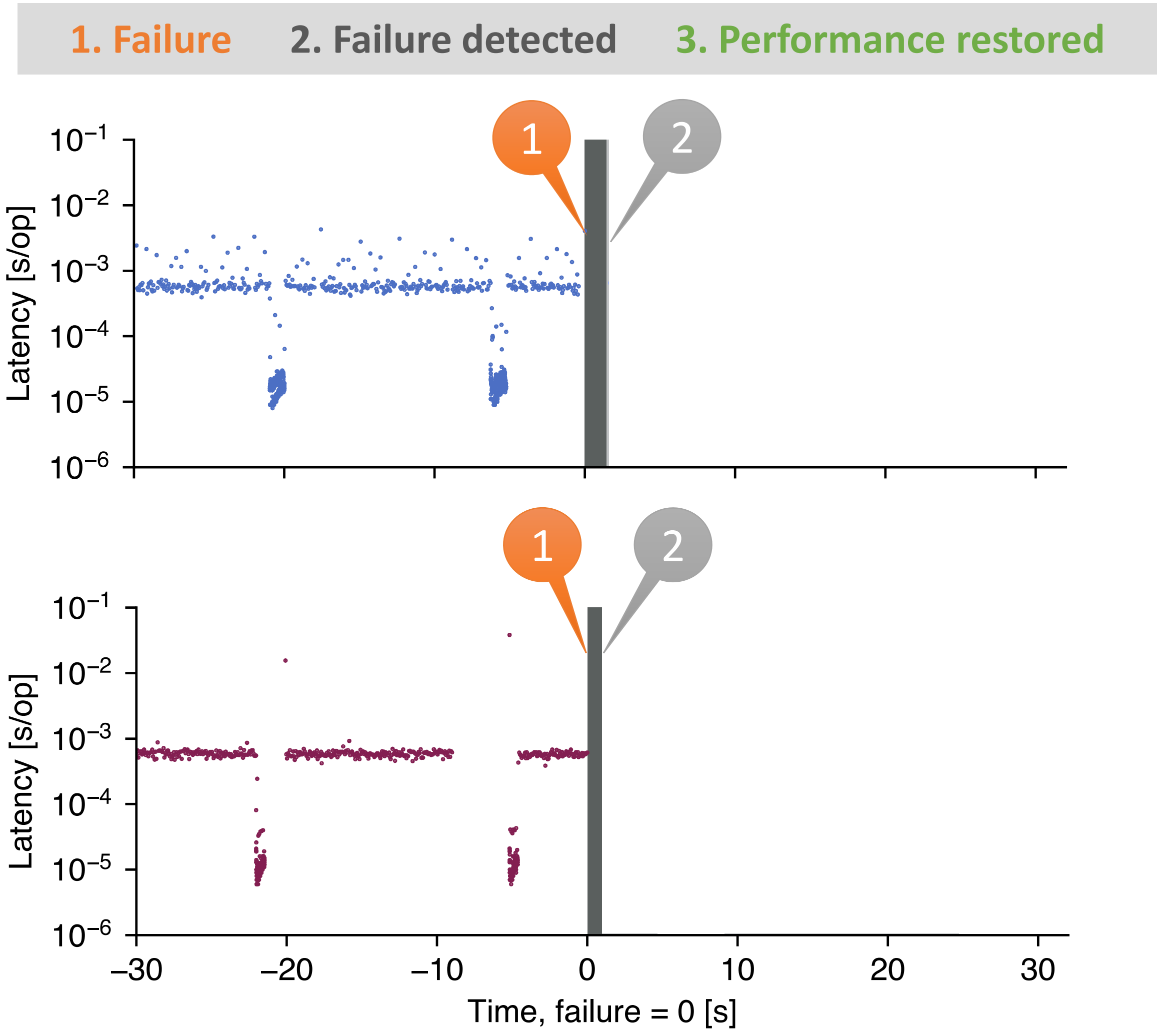


1. Failure 2. Failure detected 3. Performance restored

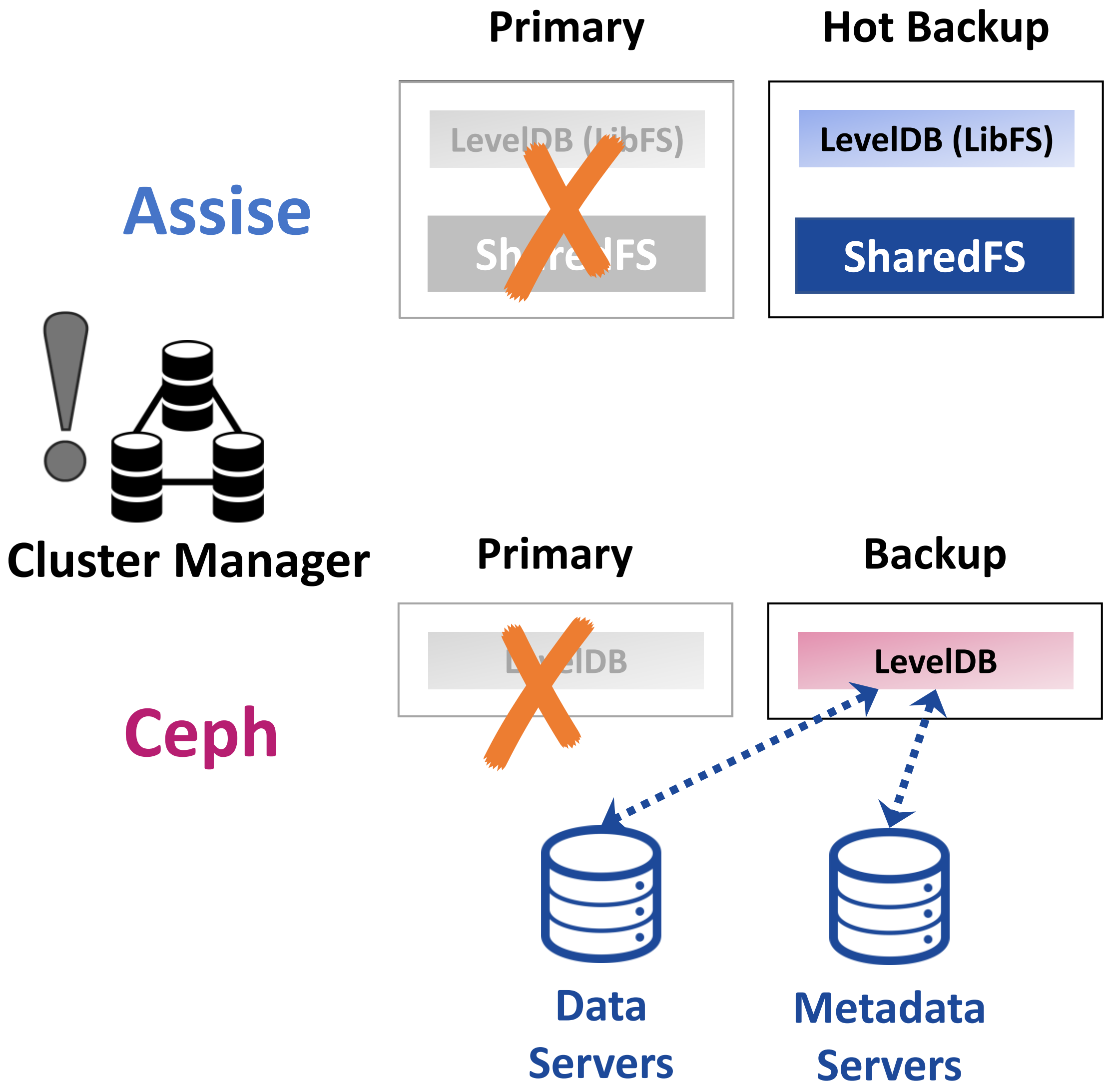


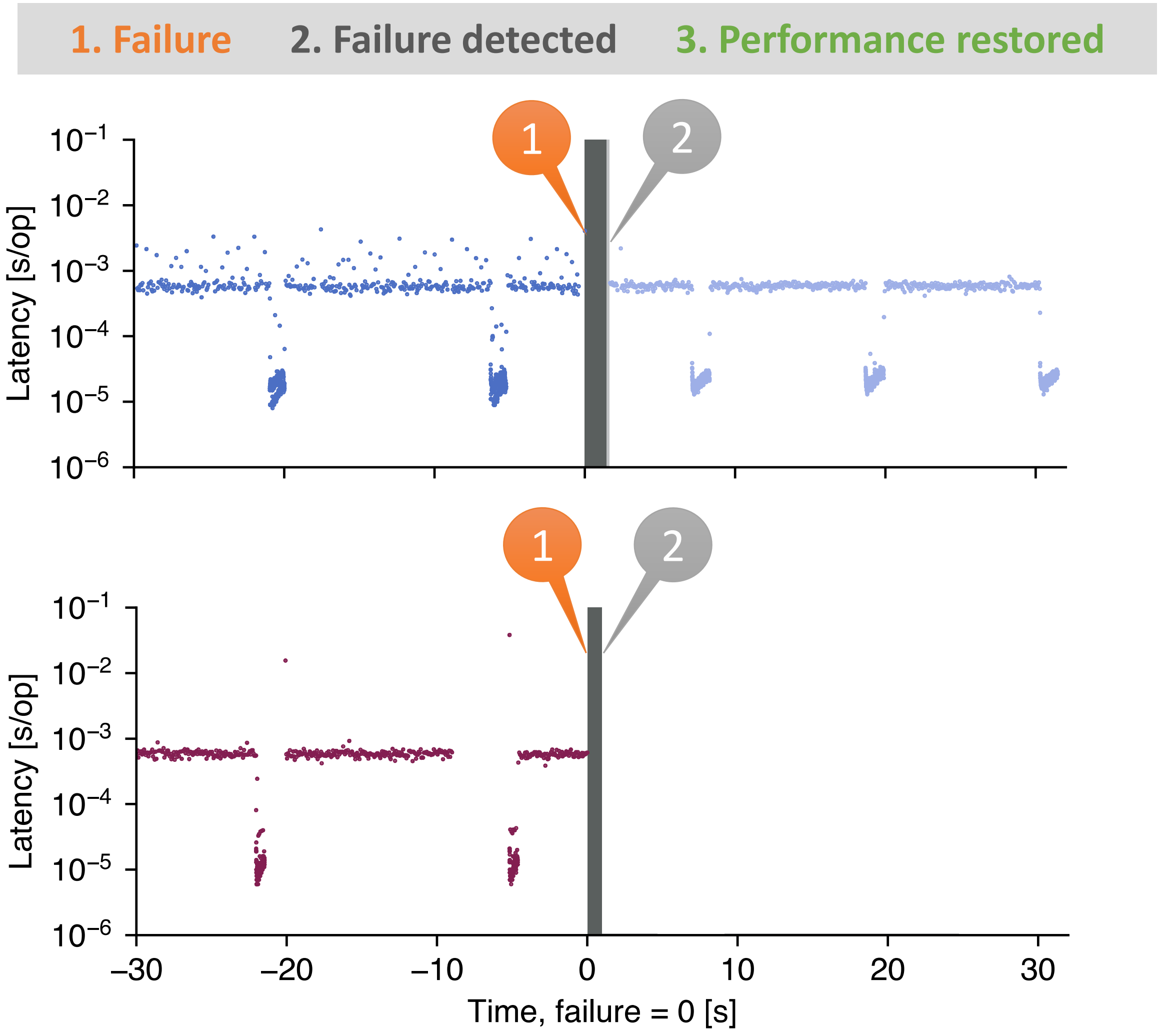
LevelDB: Fail-over to backup



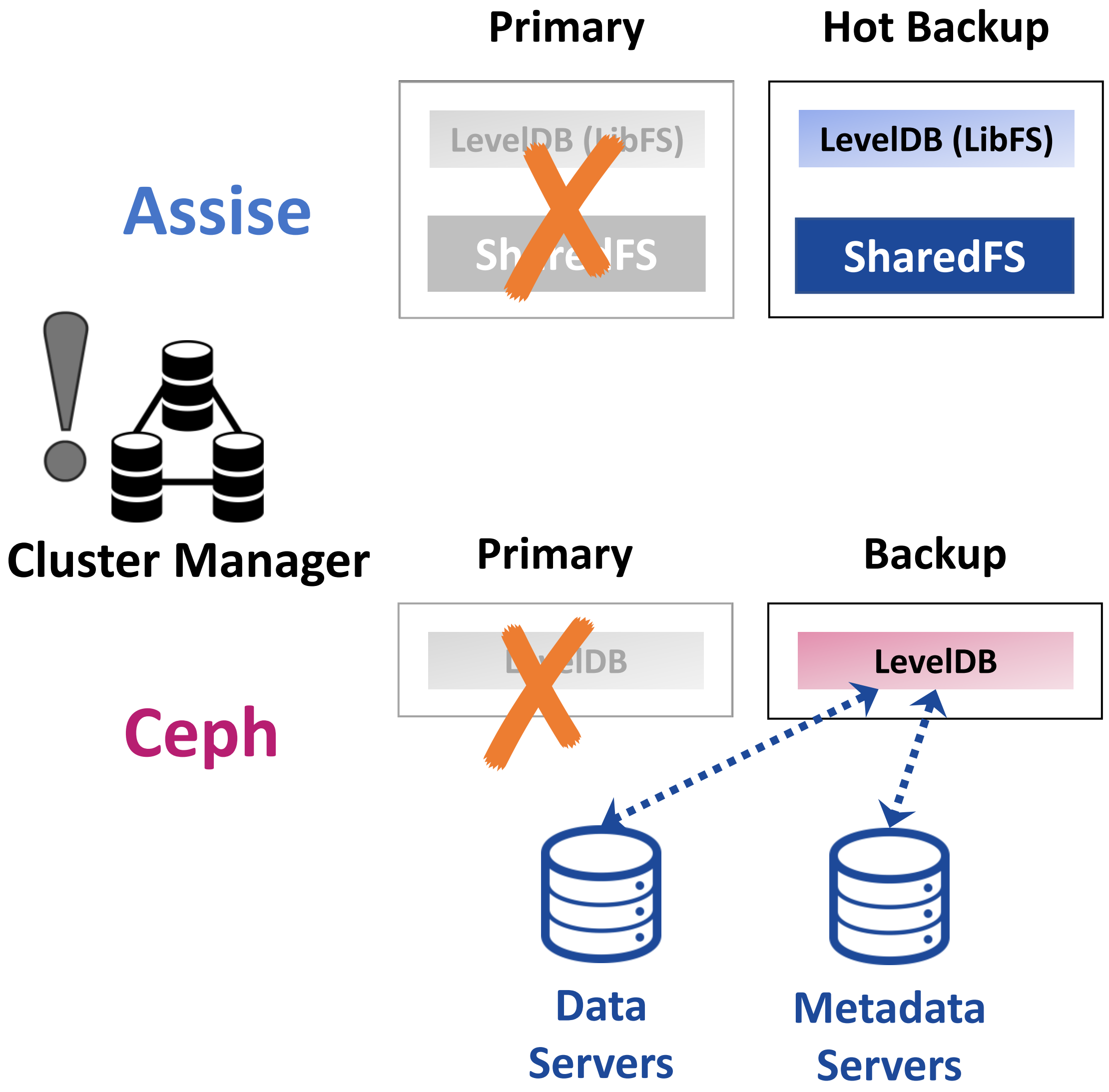


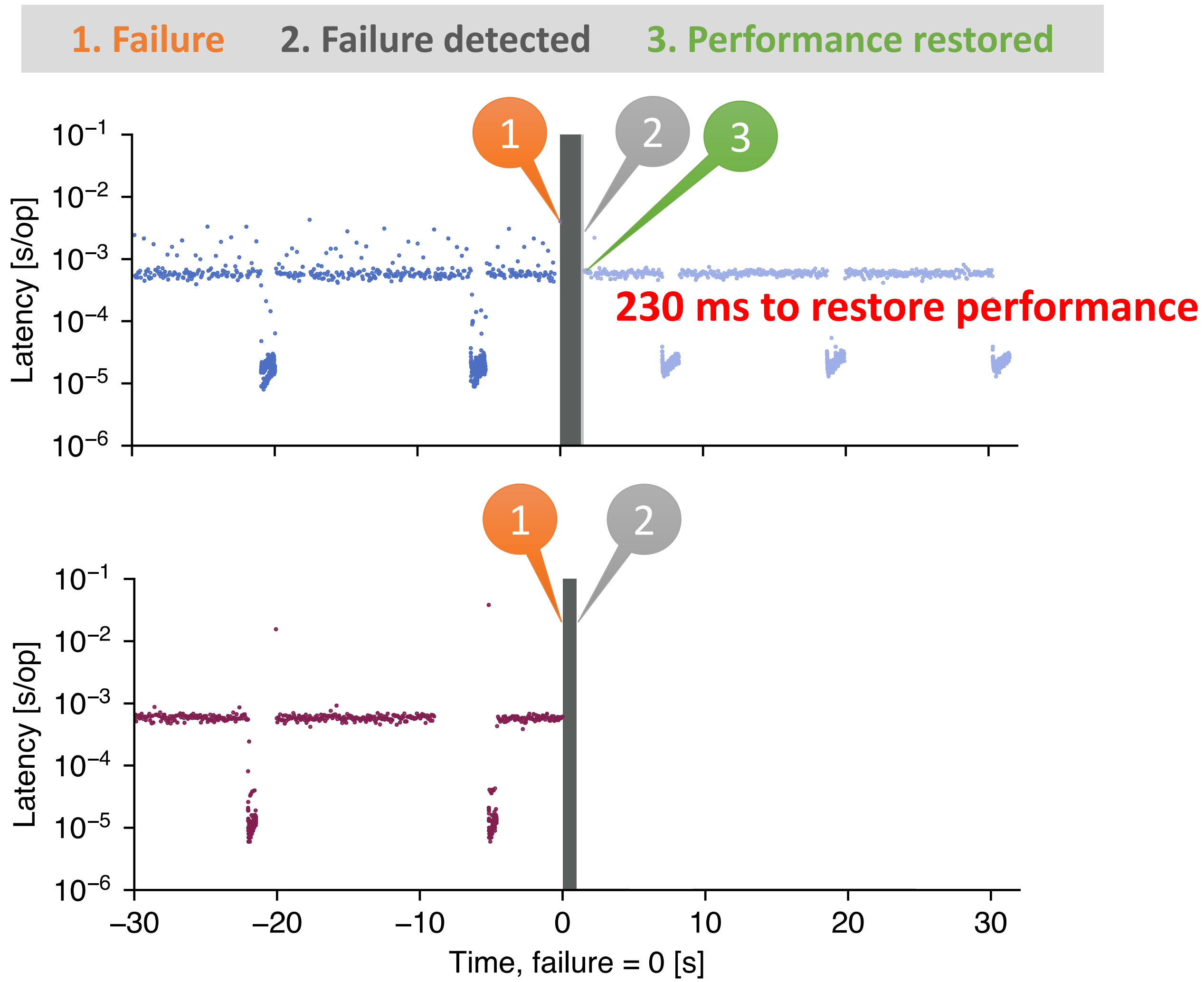
LevelDB: Fail-over to backup



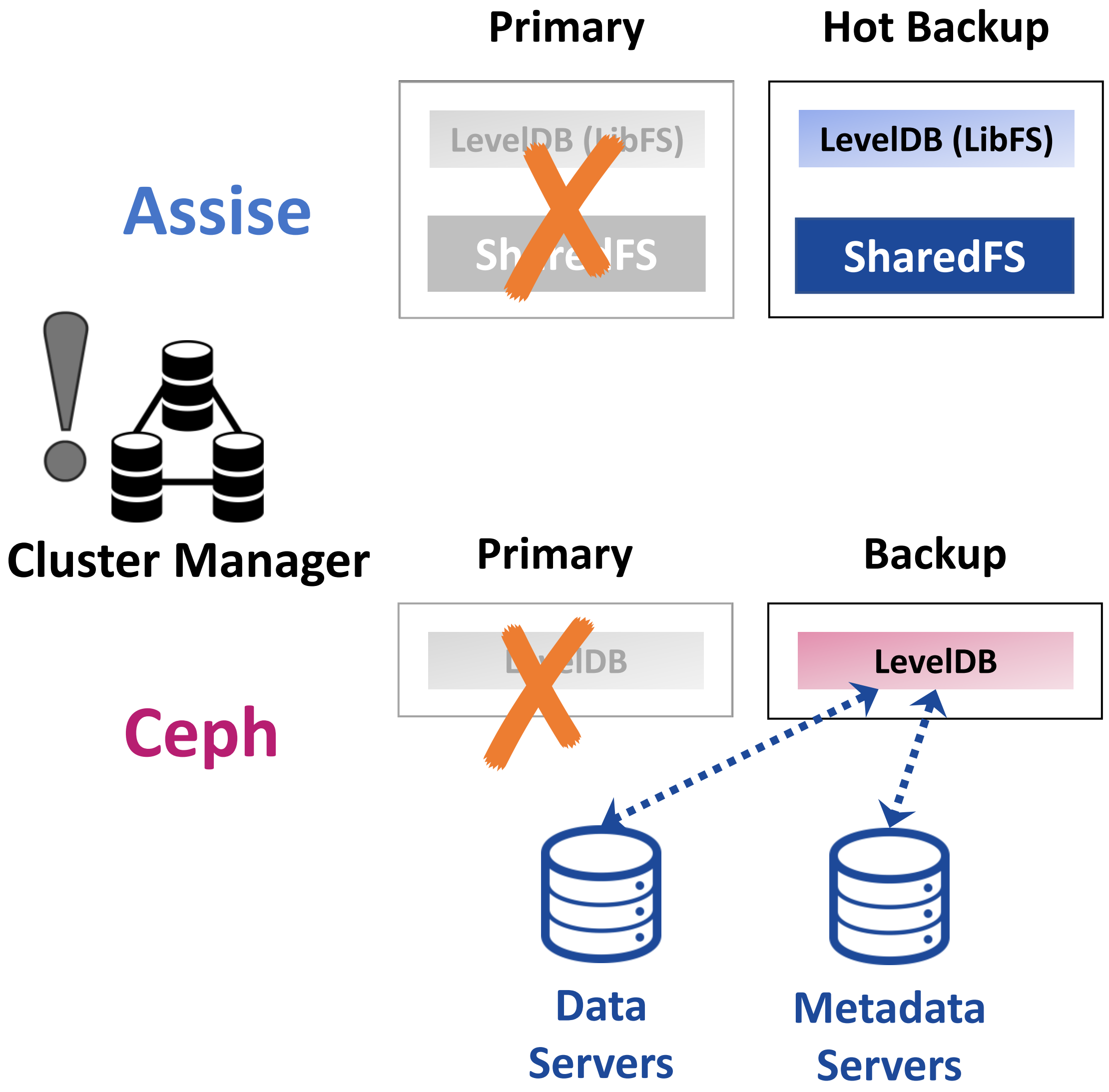


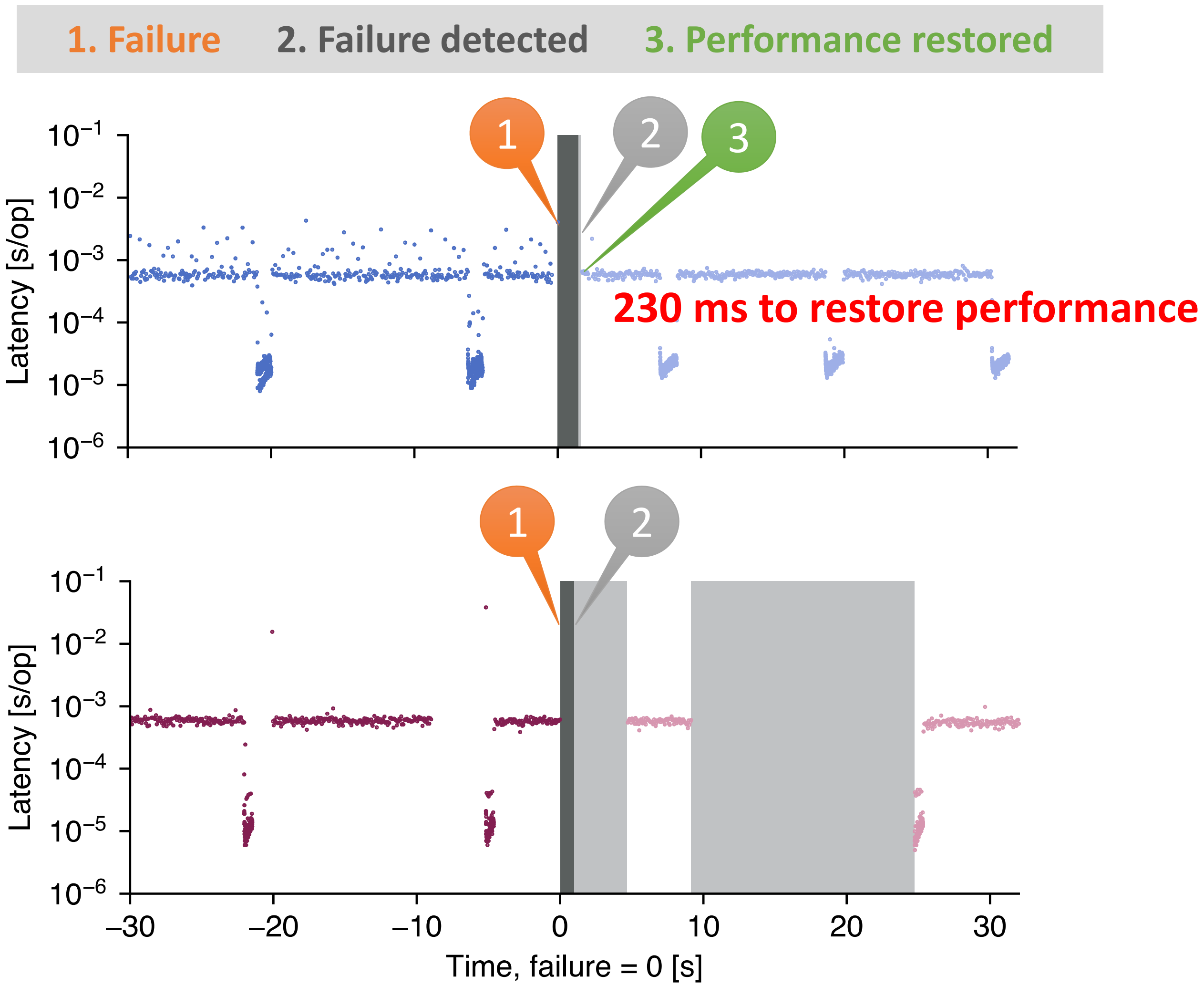
LevelDB: Fail-over to backup



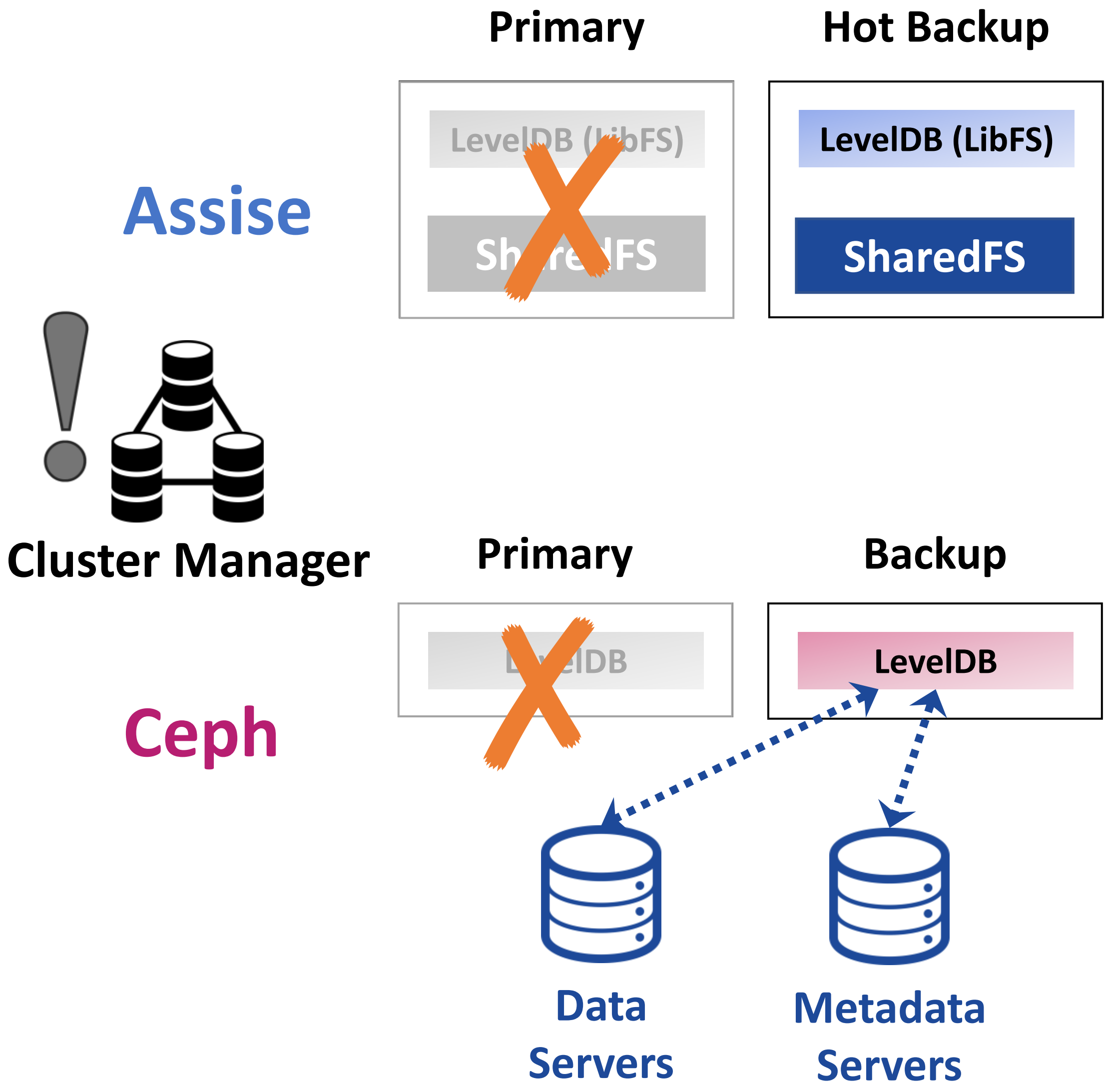


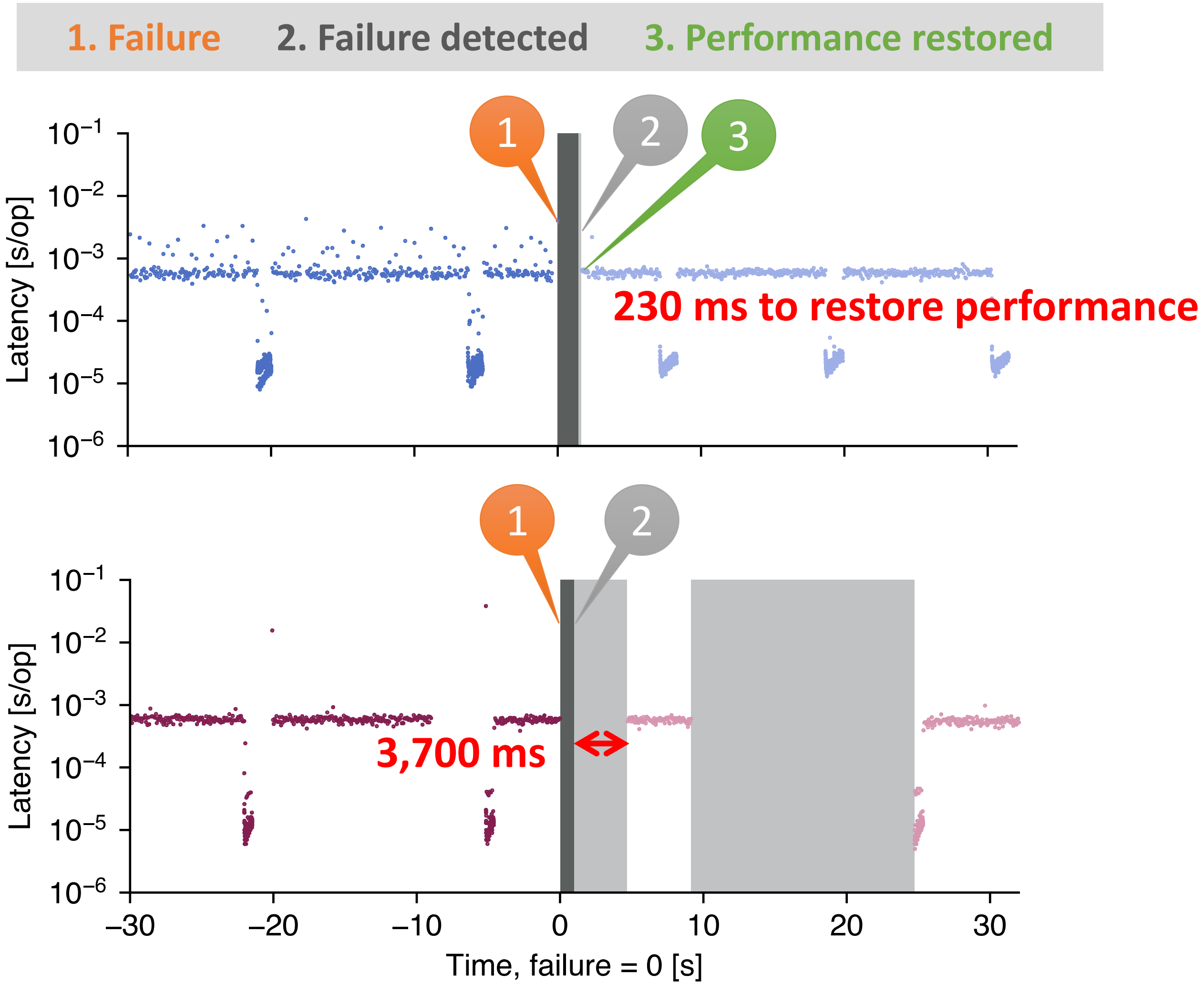
LevelDB: Fail-over to backup



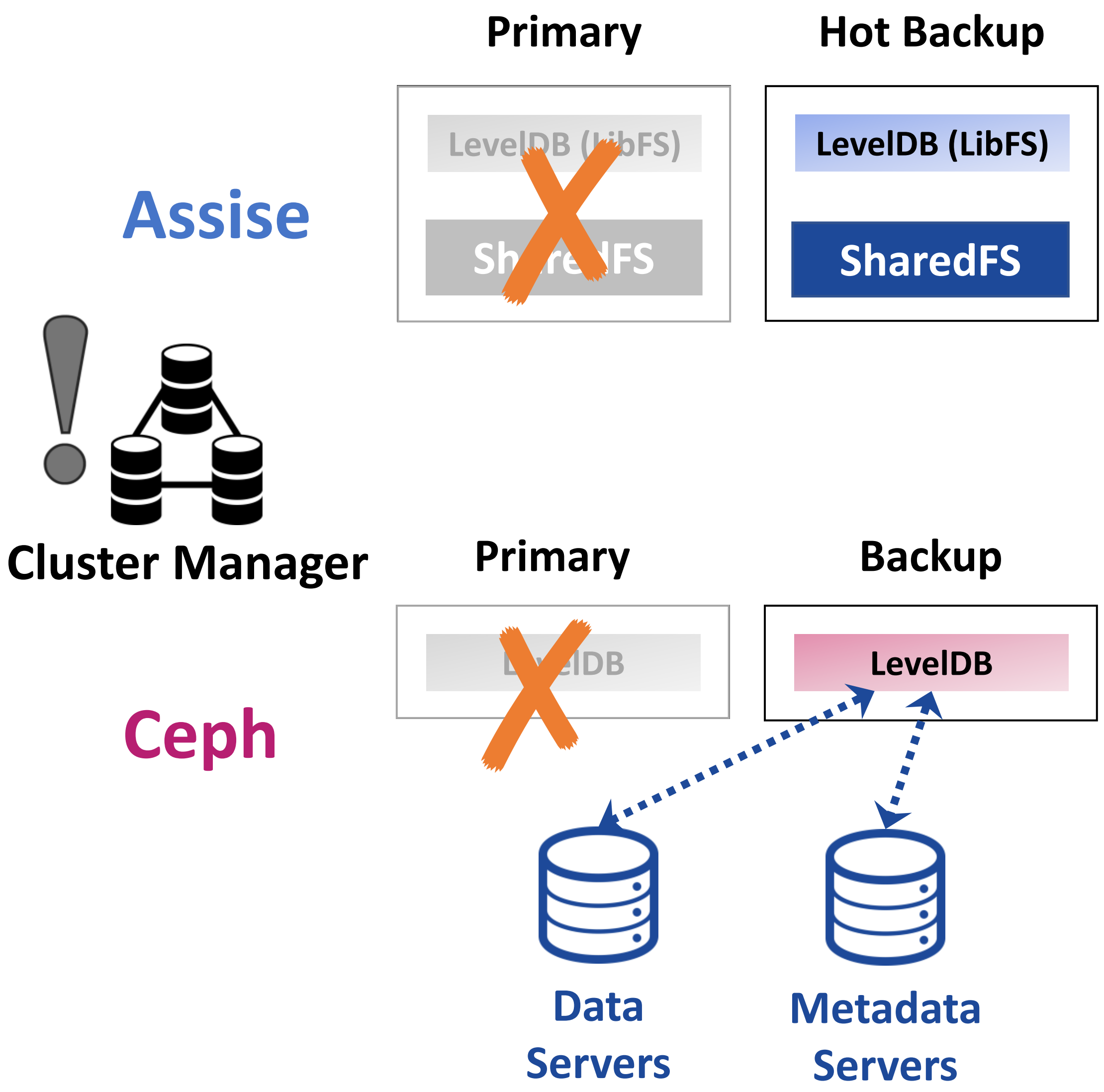


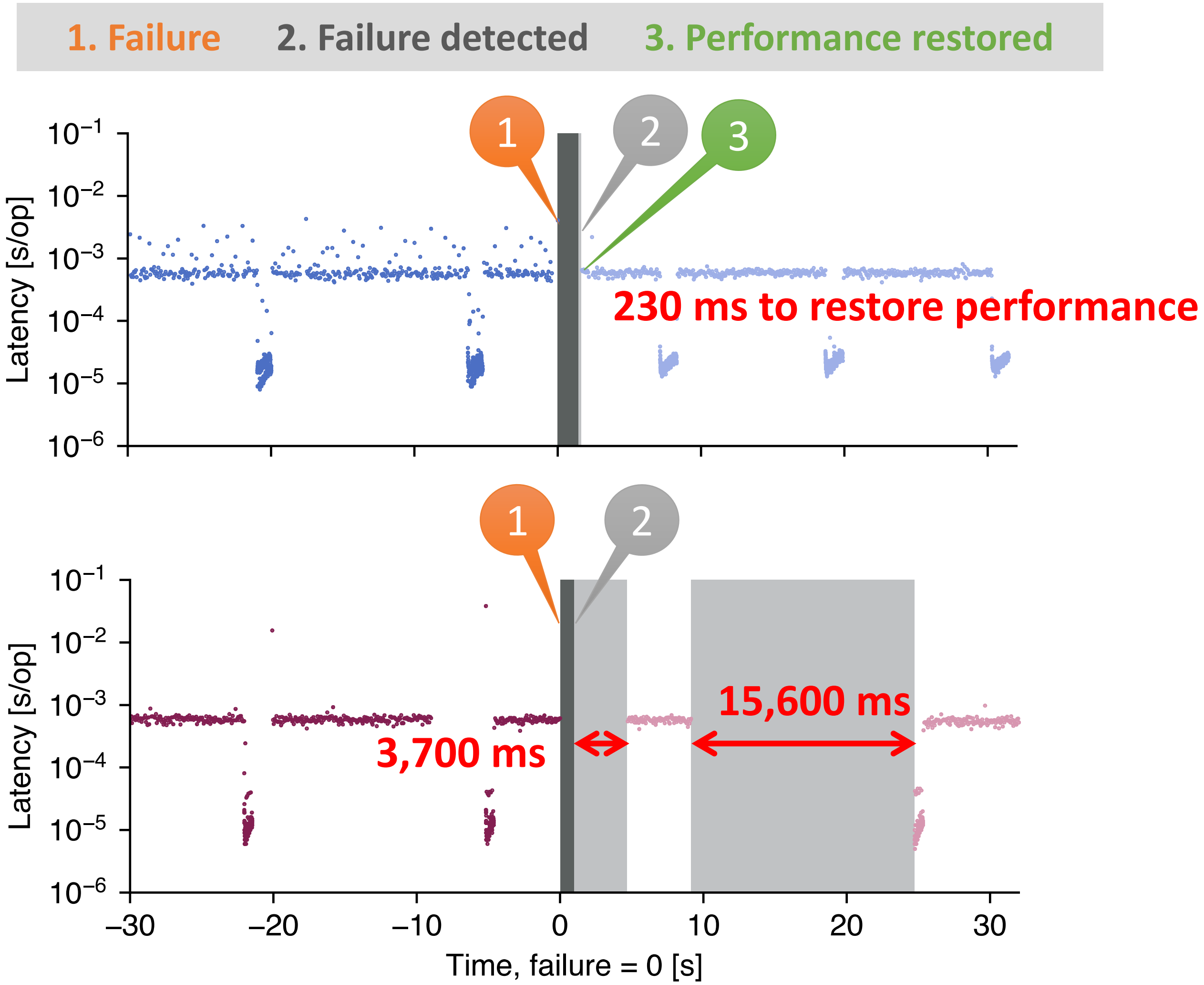
LevelDB: Fail-over to backup



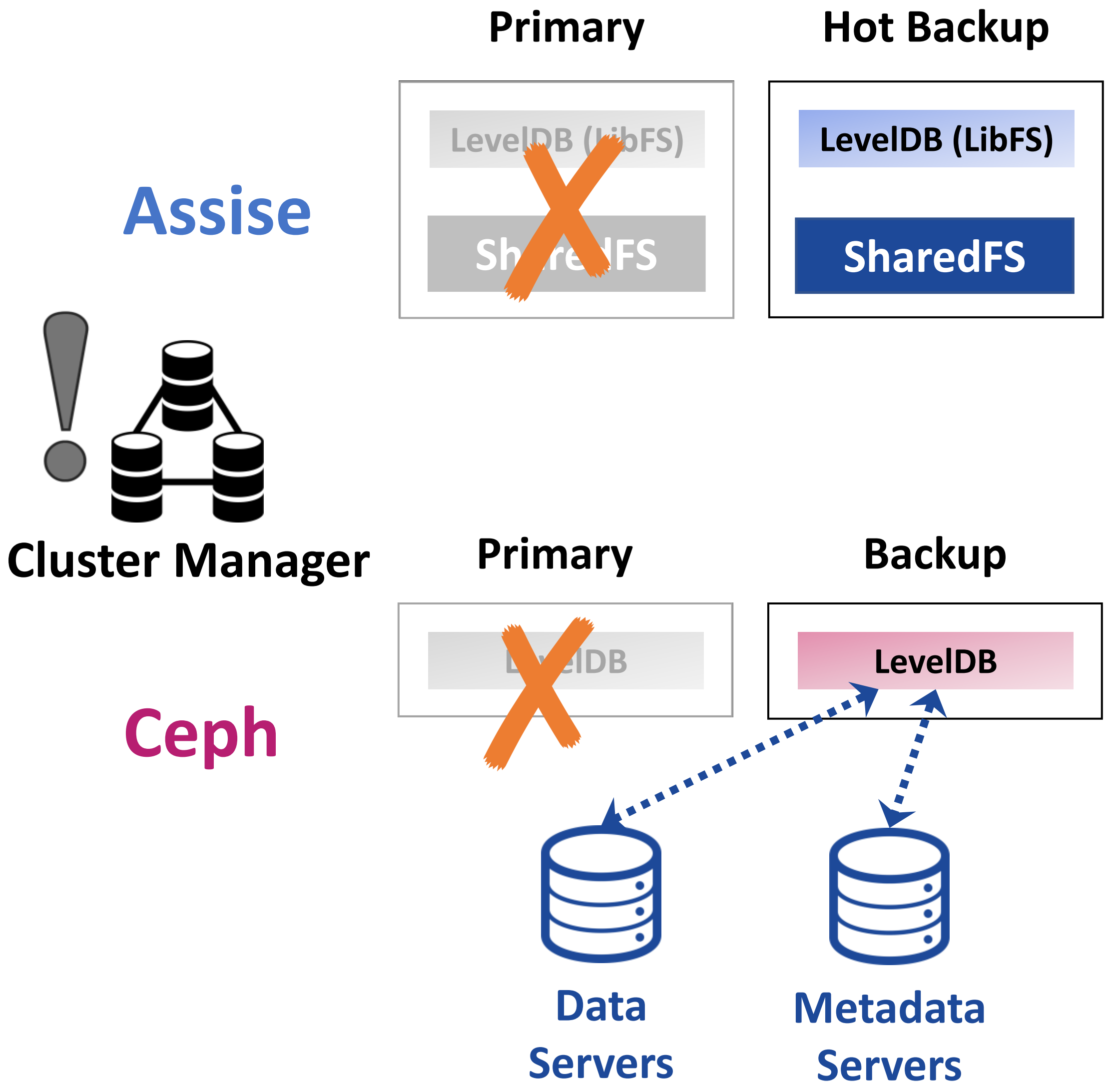


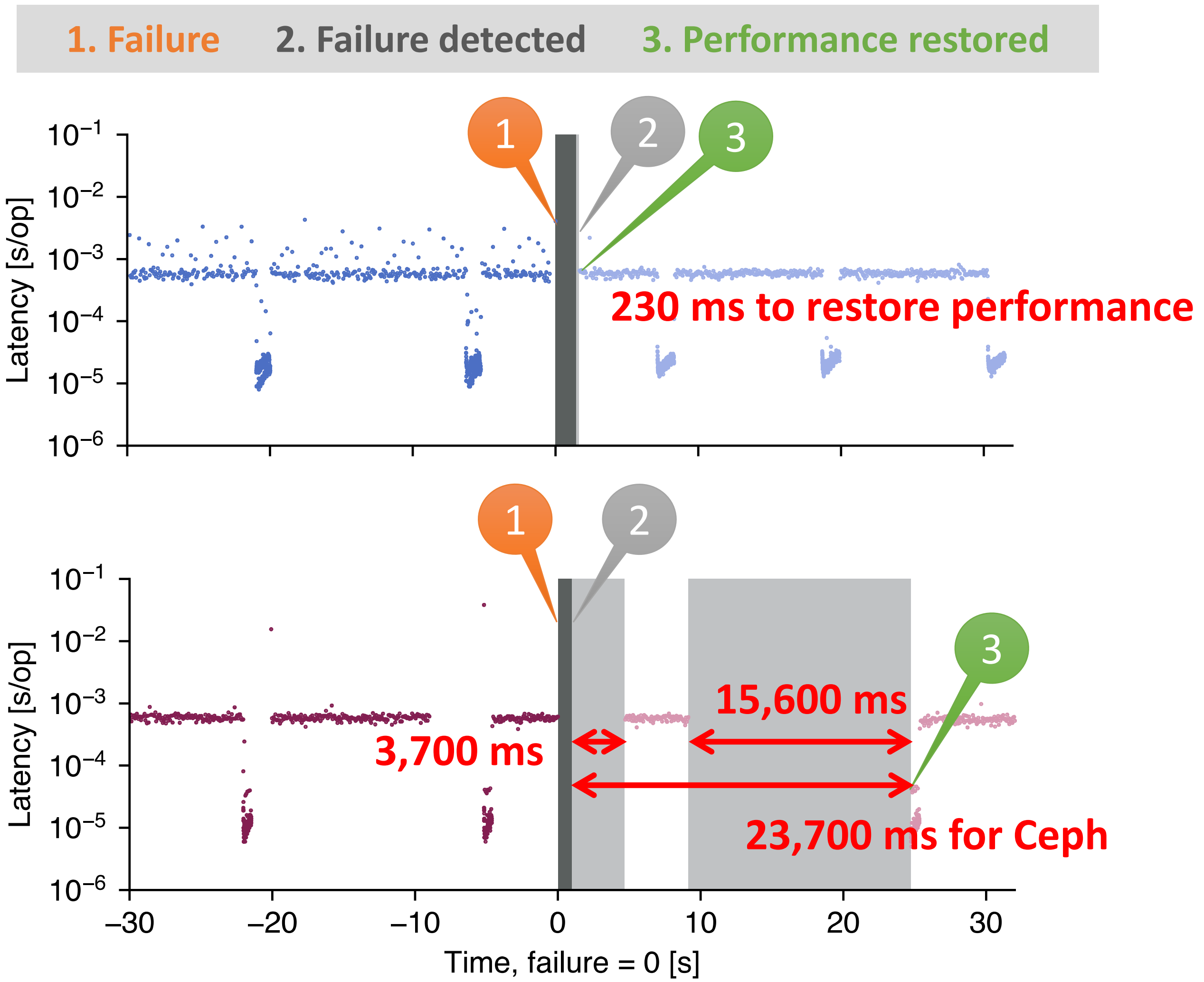
LevelDB: Fail-over to backup



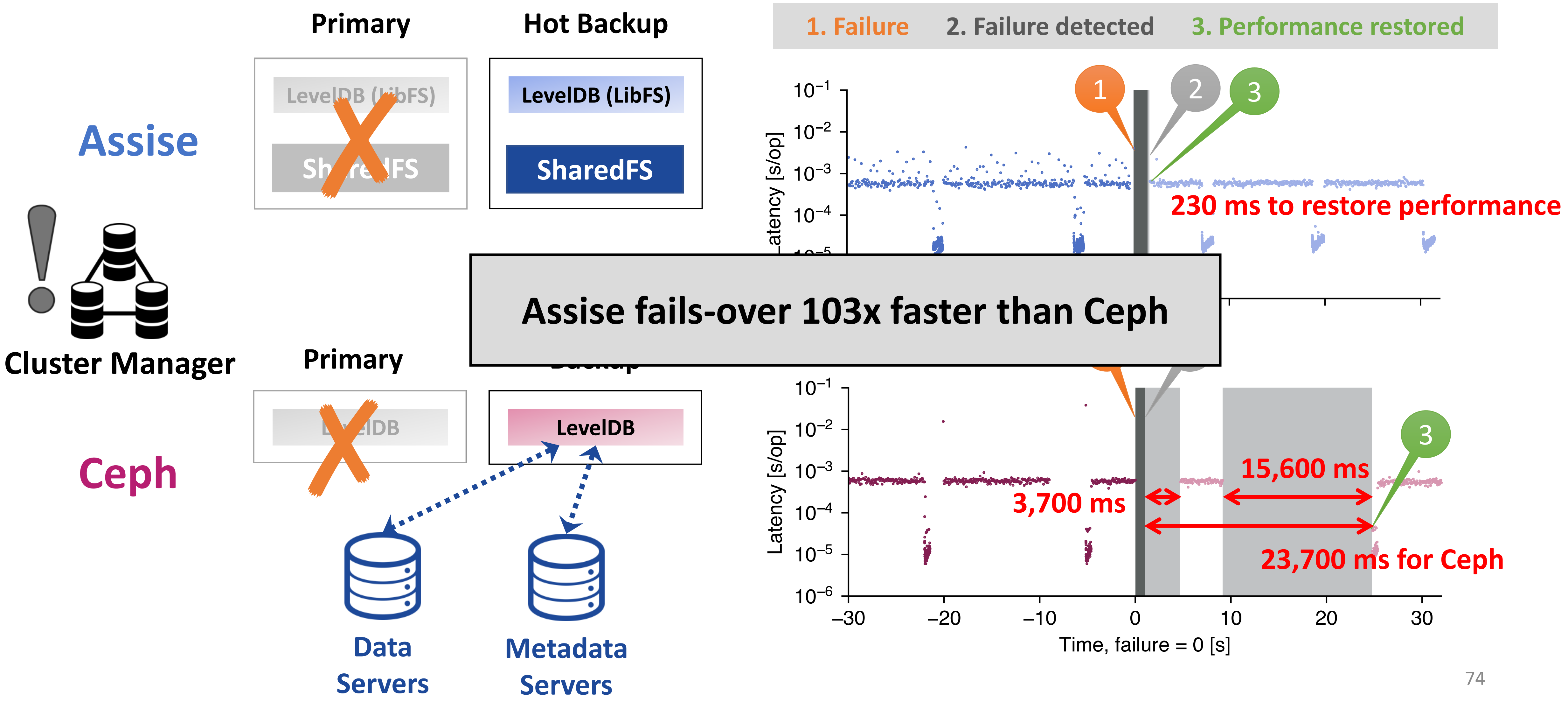


LevelDB: Fail-over to backup

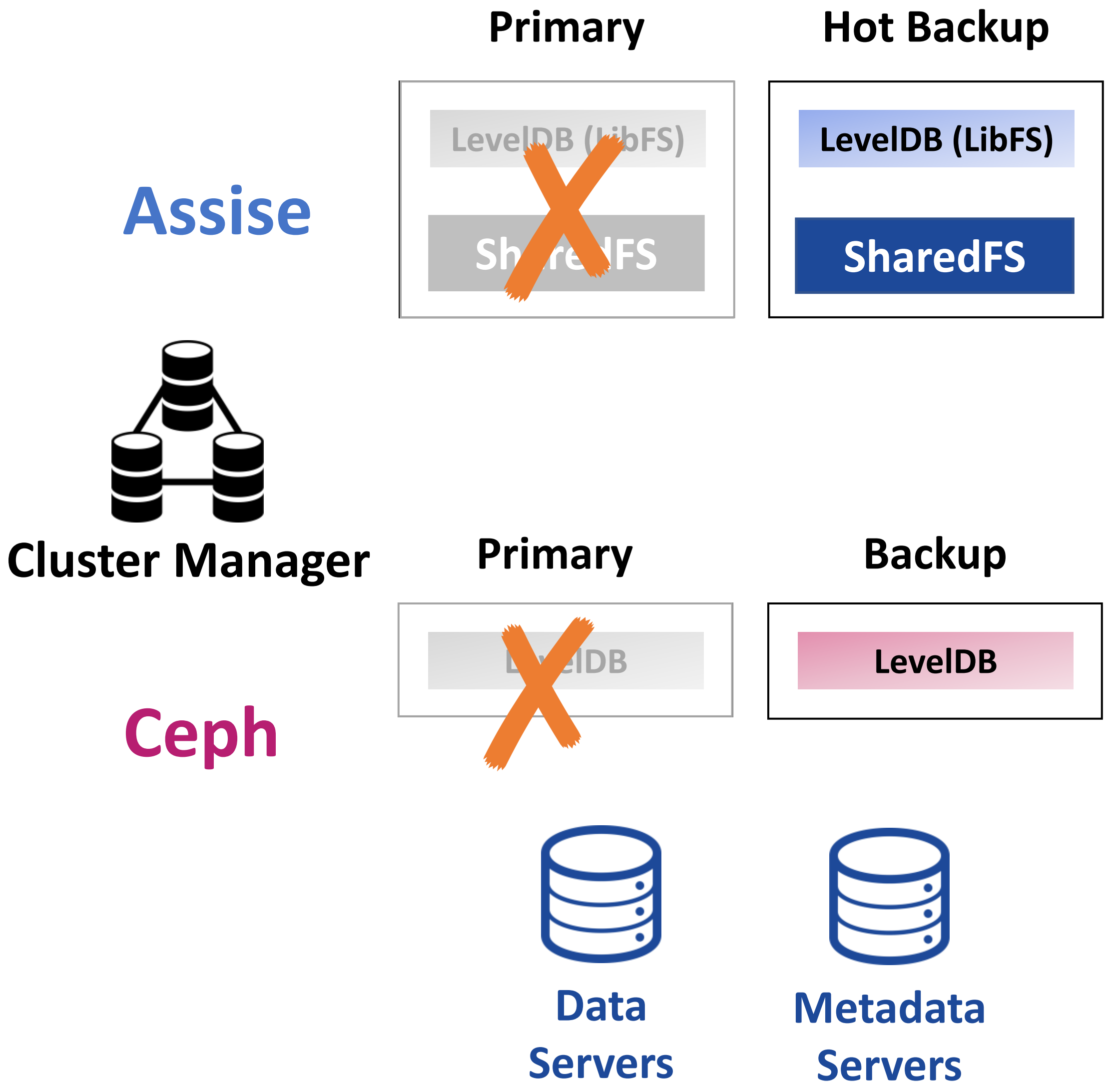




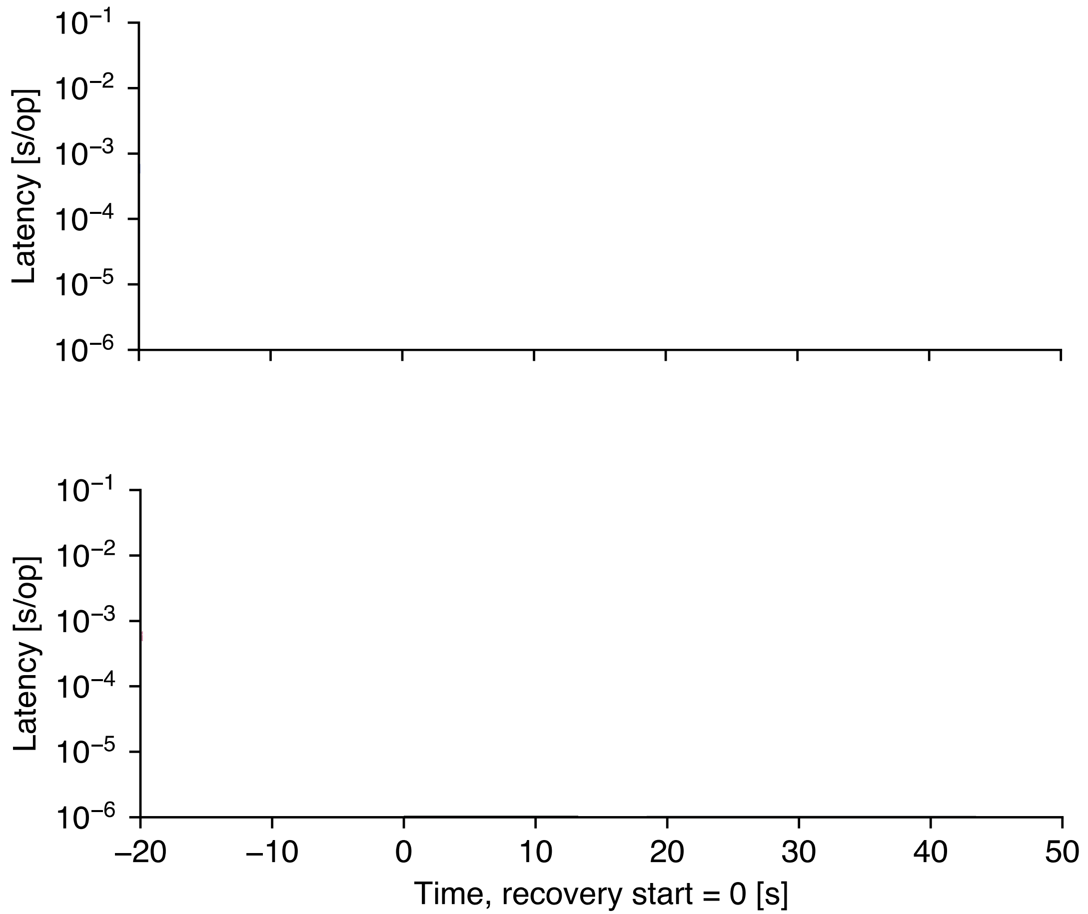
LevelDB: Fail-over to backup



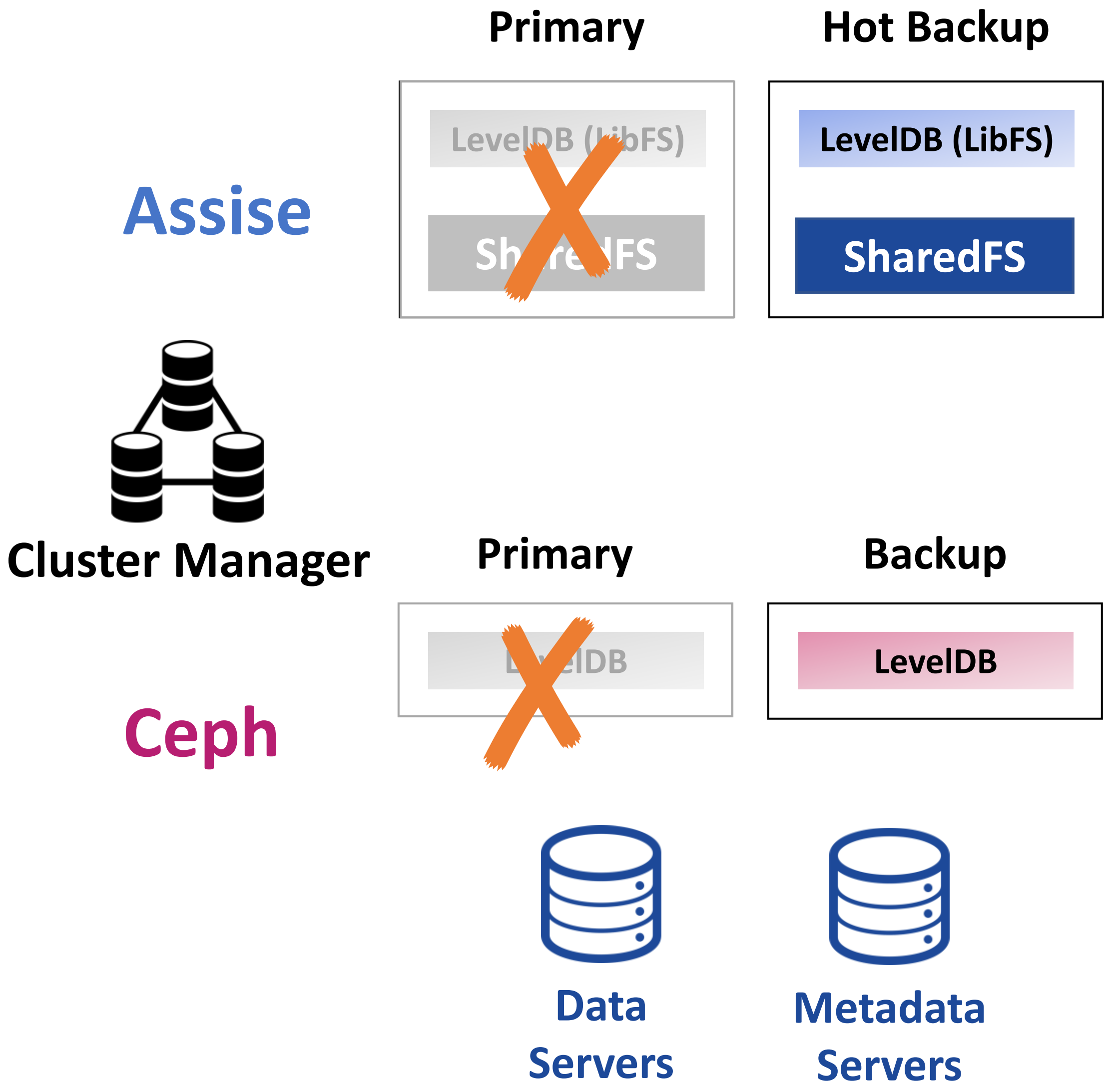
LevelDB: Primary recovery

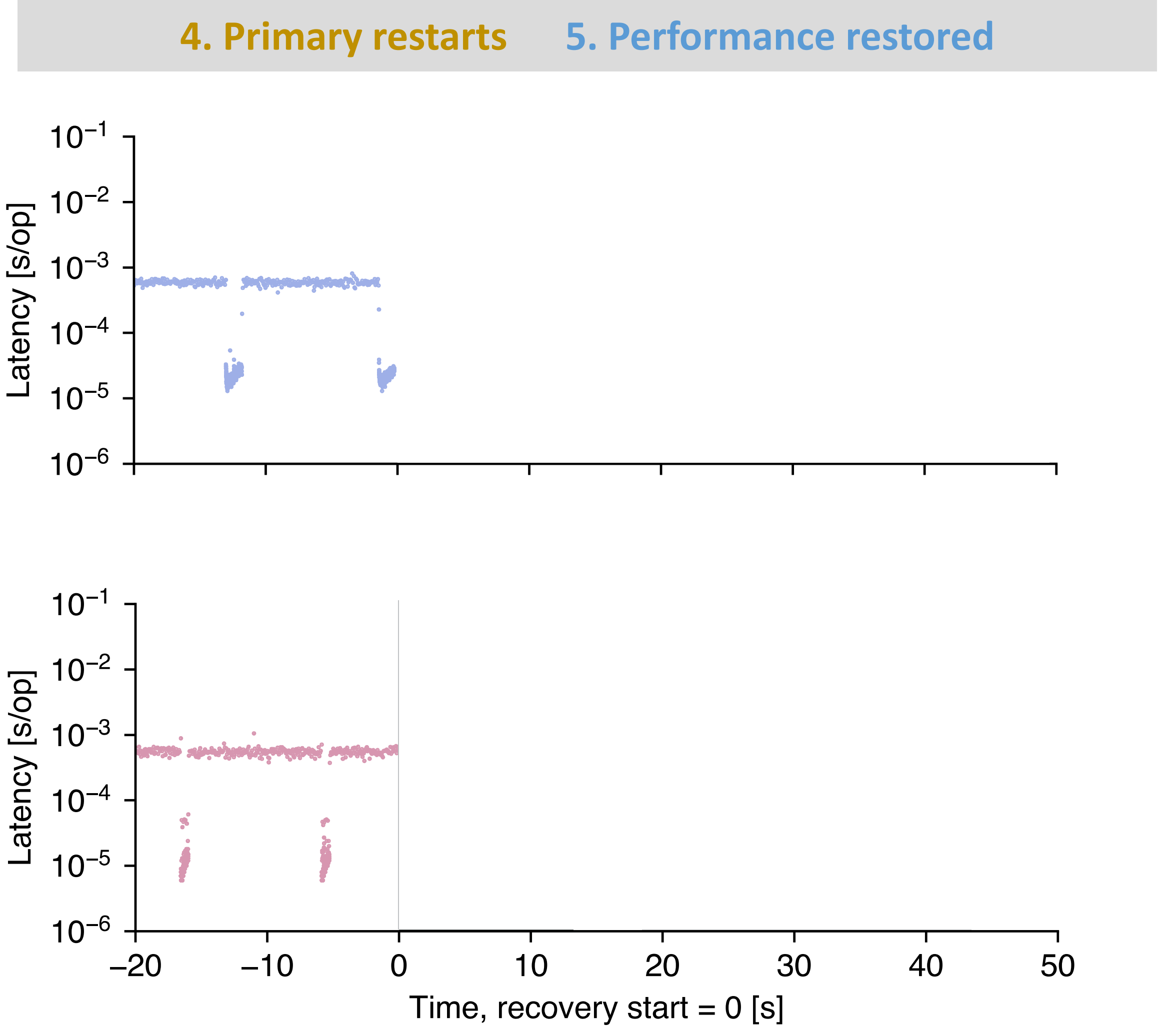


4. Primary restarts 5. Performance restored

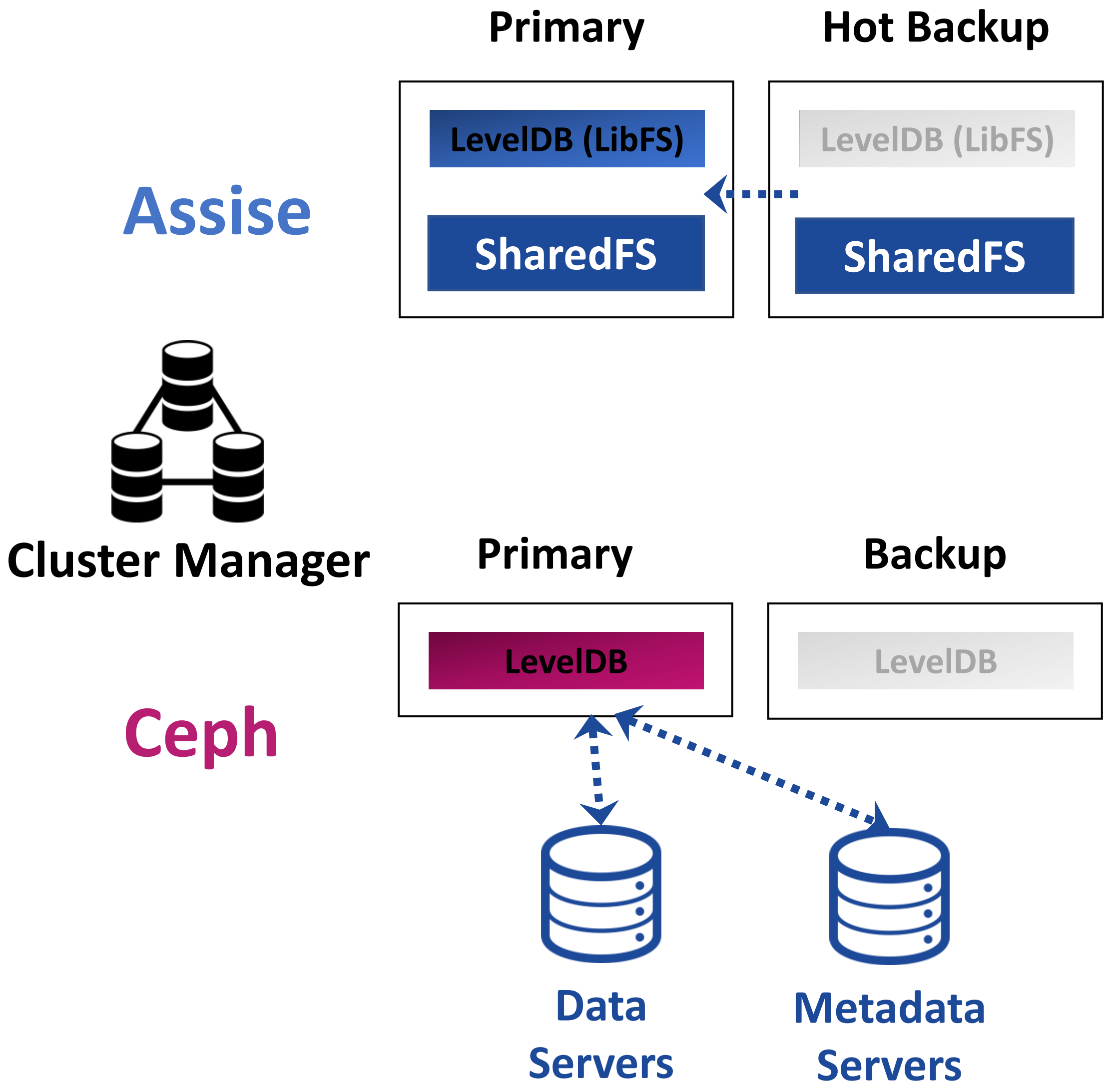


LevelDB: Primary recovery

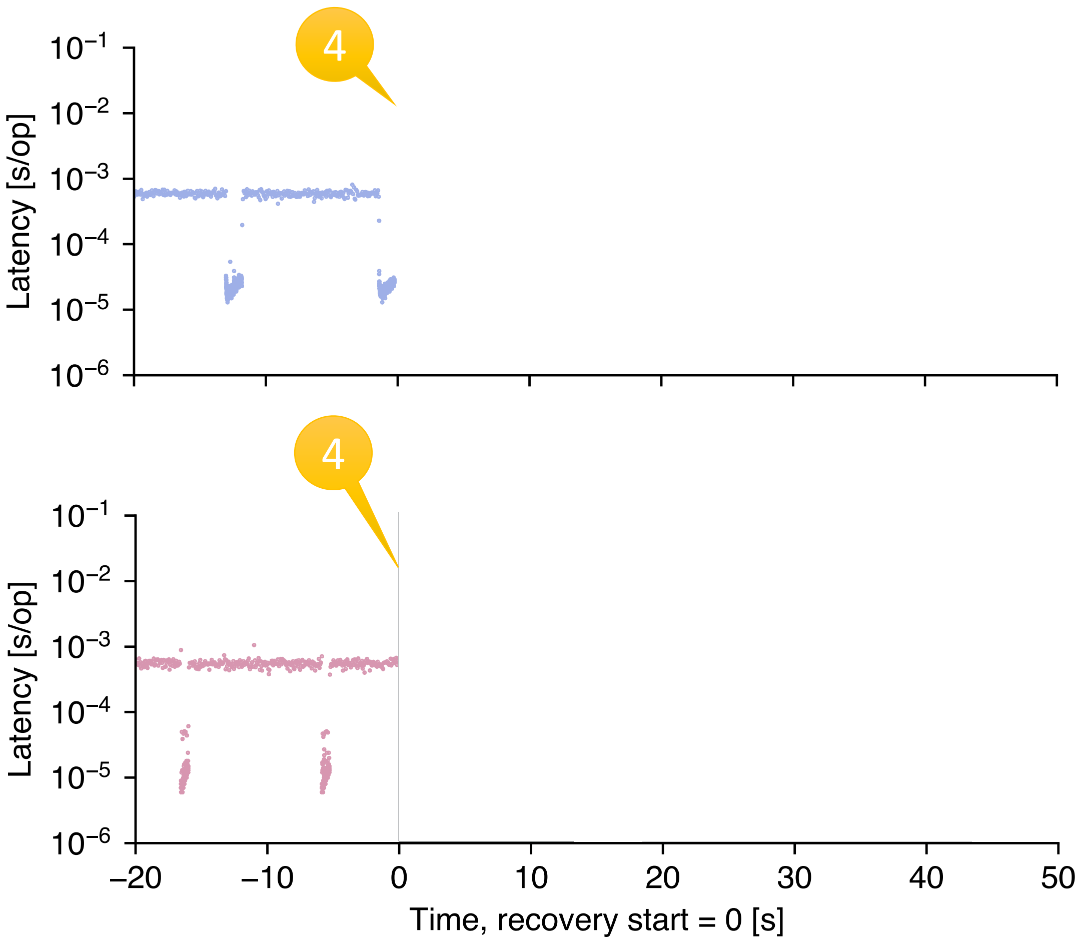




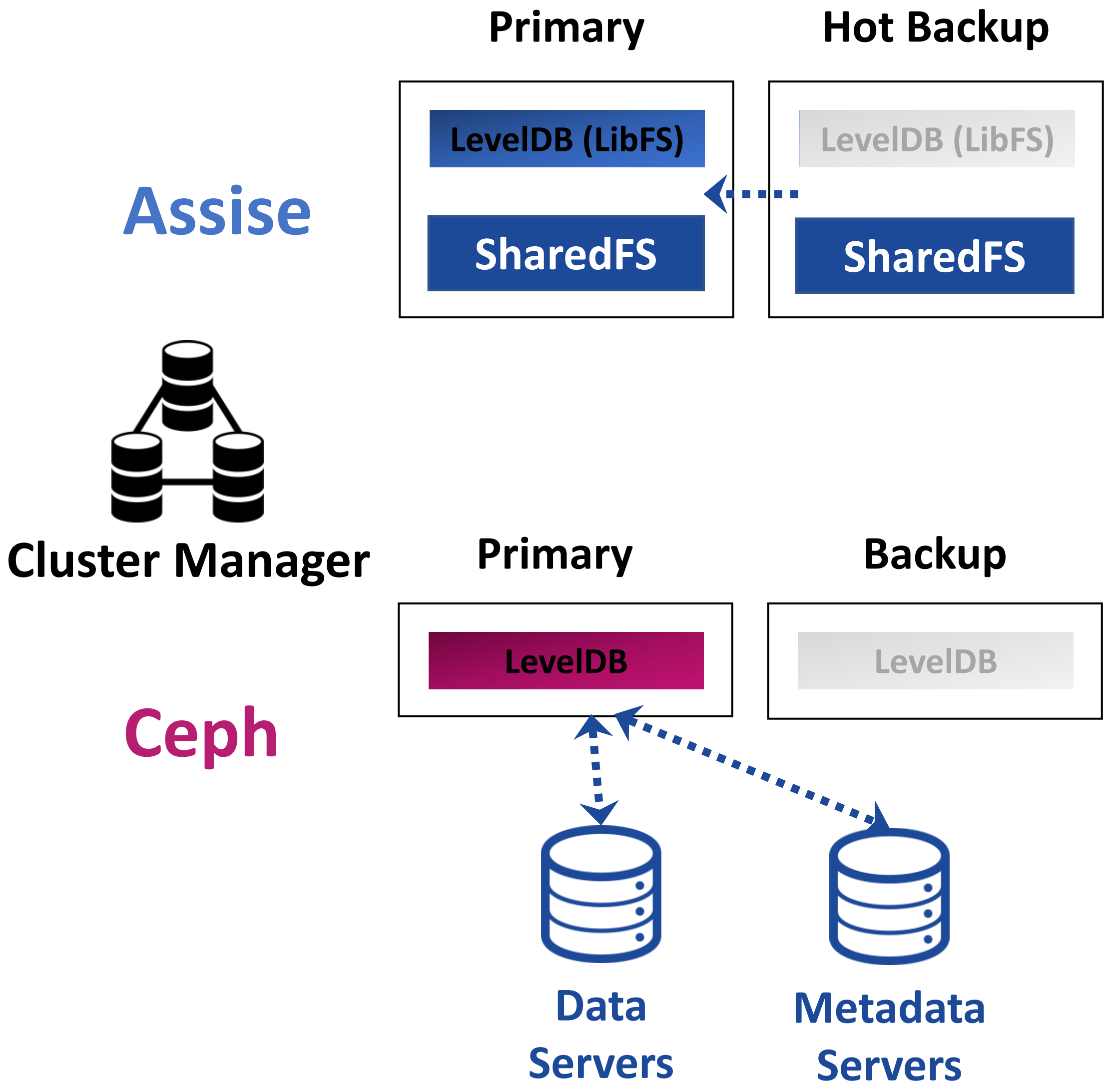
LevelDB: Primary recovery

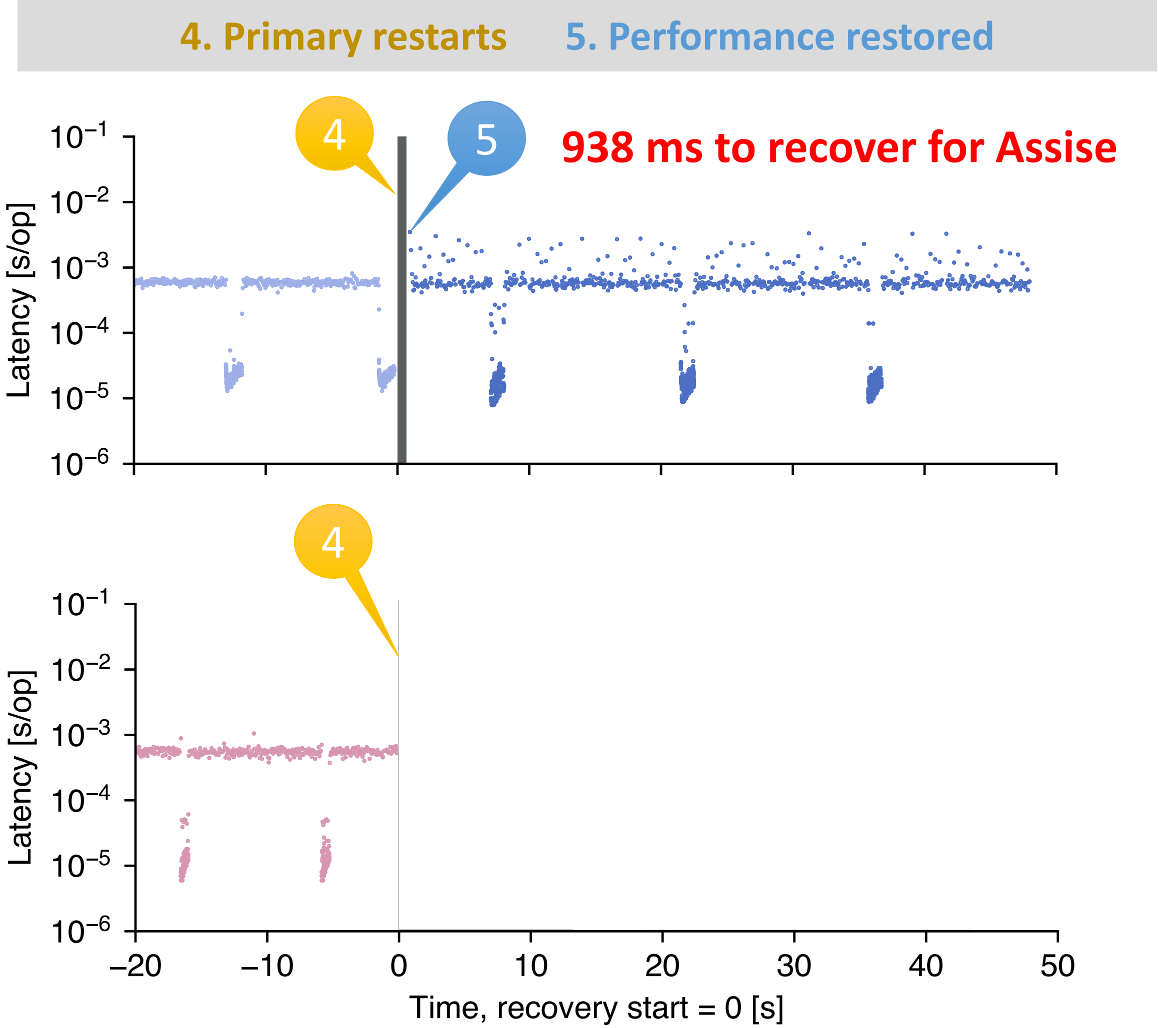


4. Primary restarts 5. Performance restored

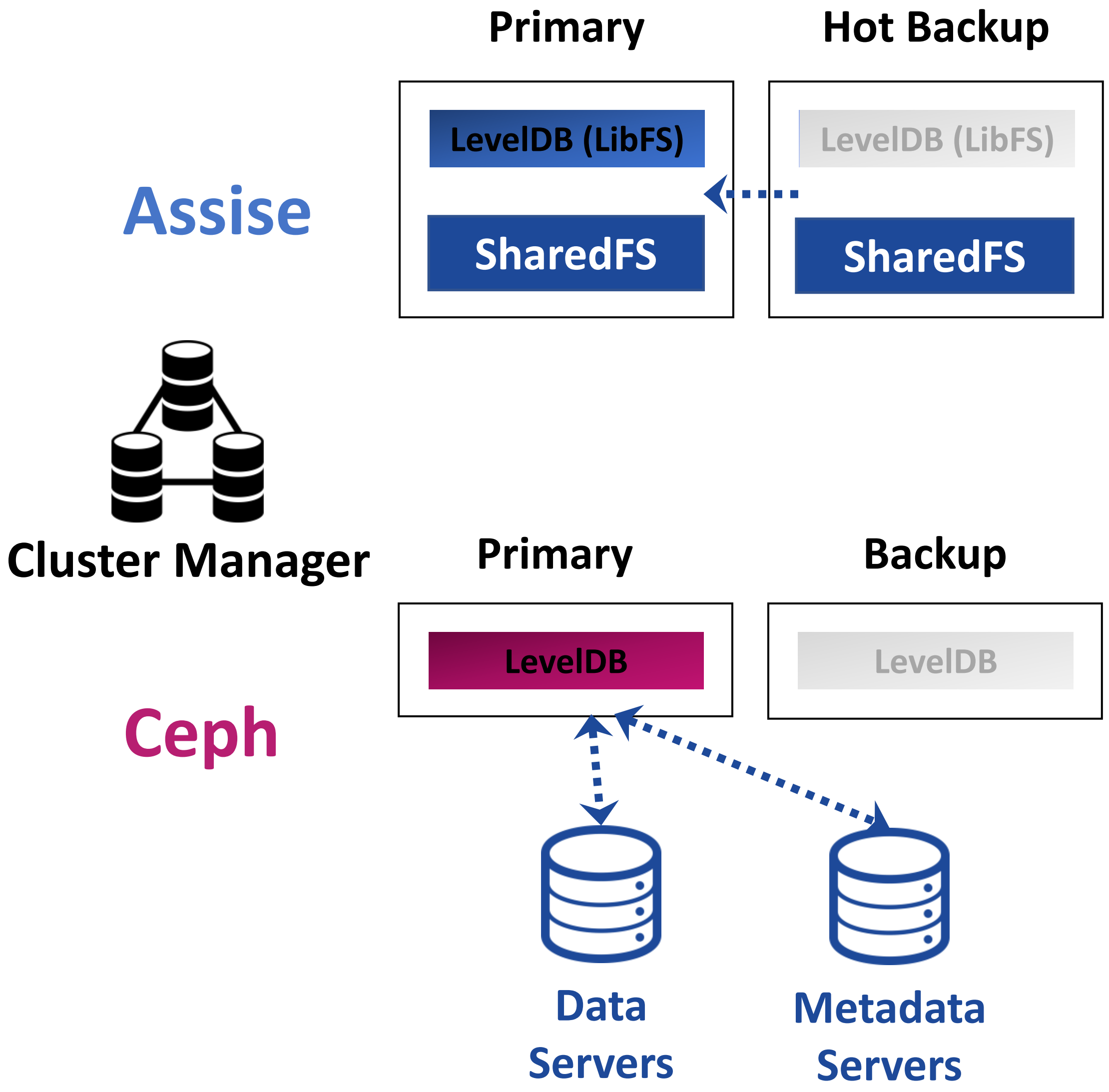


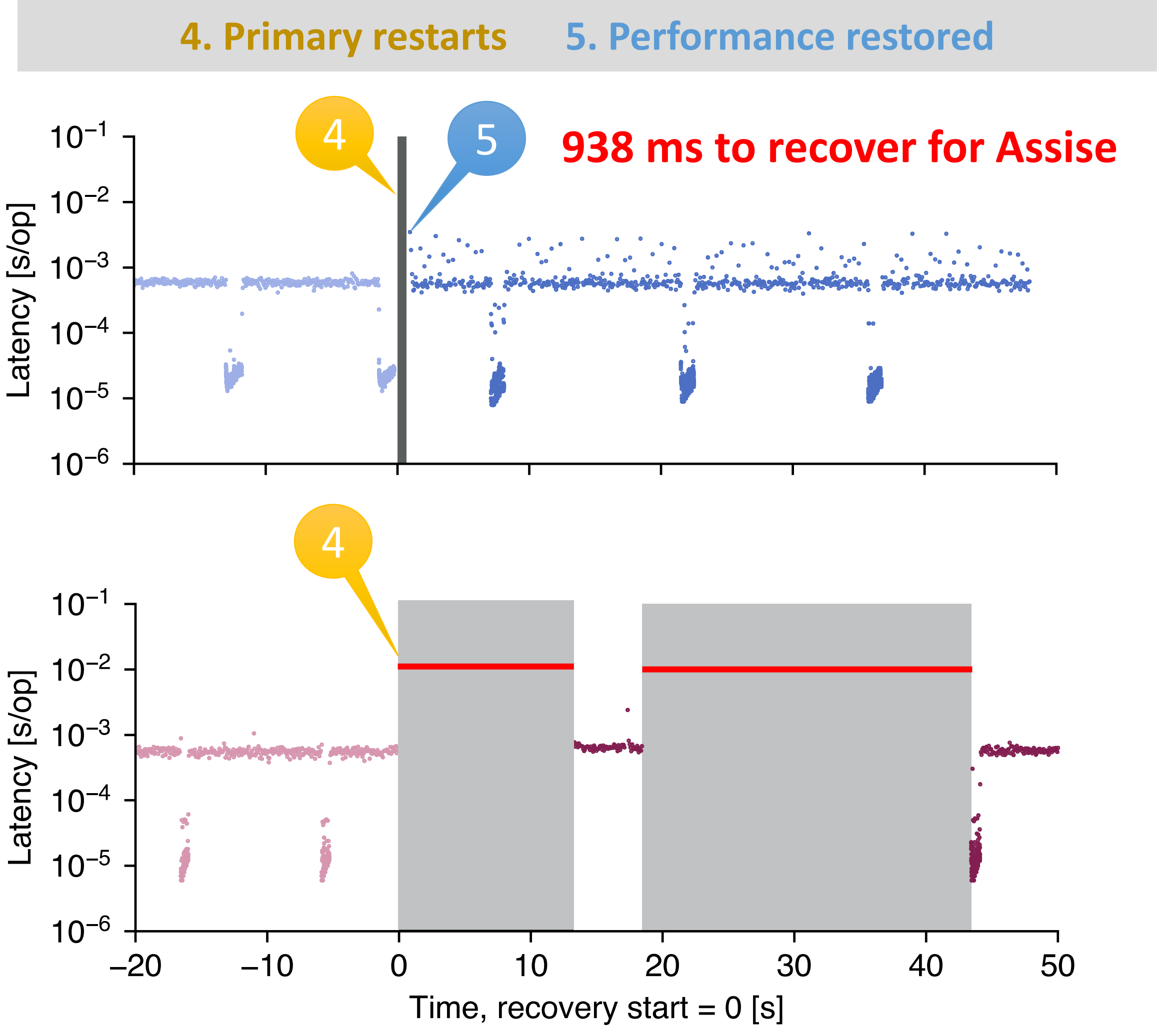
LevelDB: Primary recovery



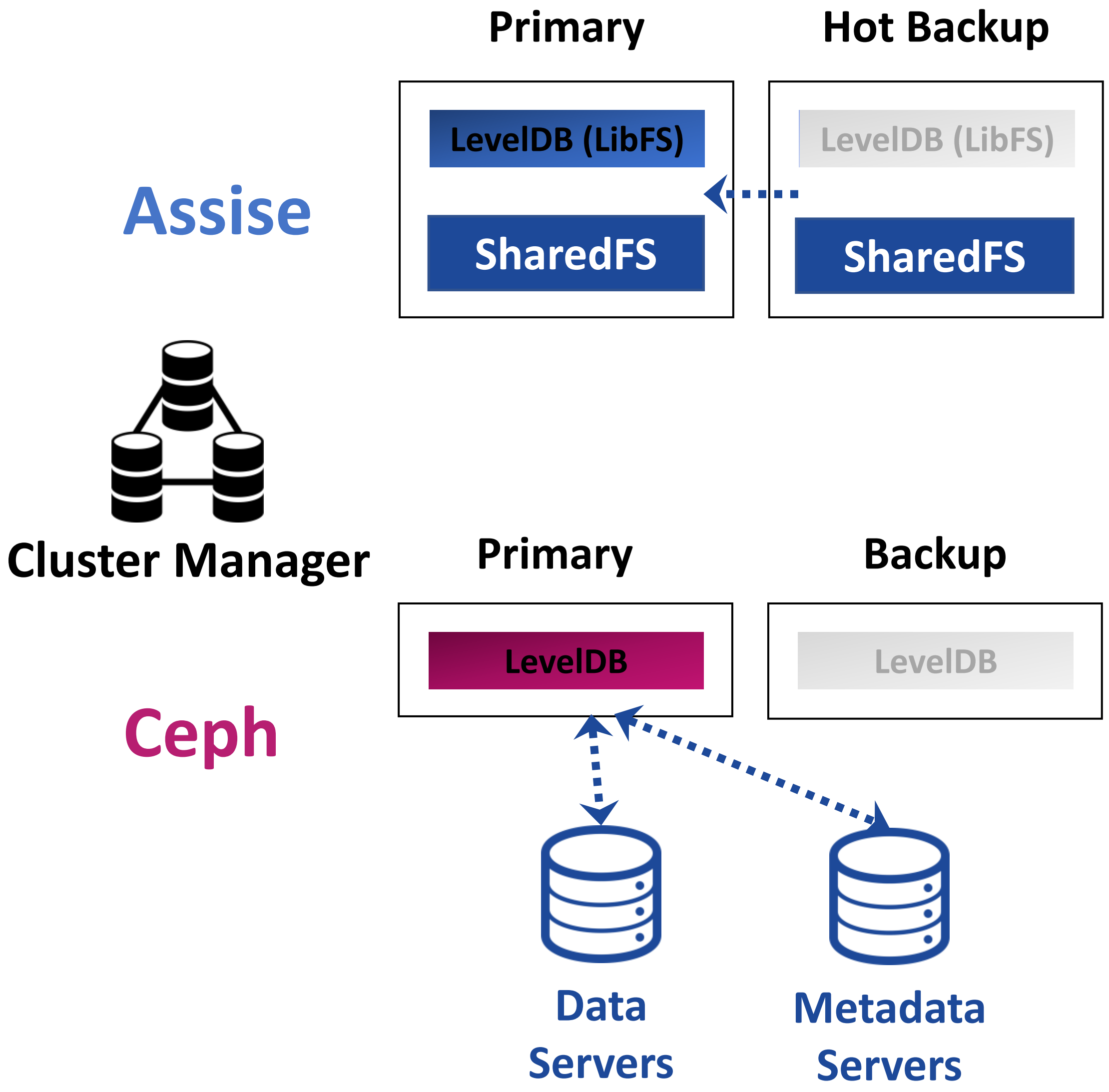


LevelDB: Primary recovery

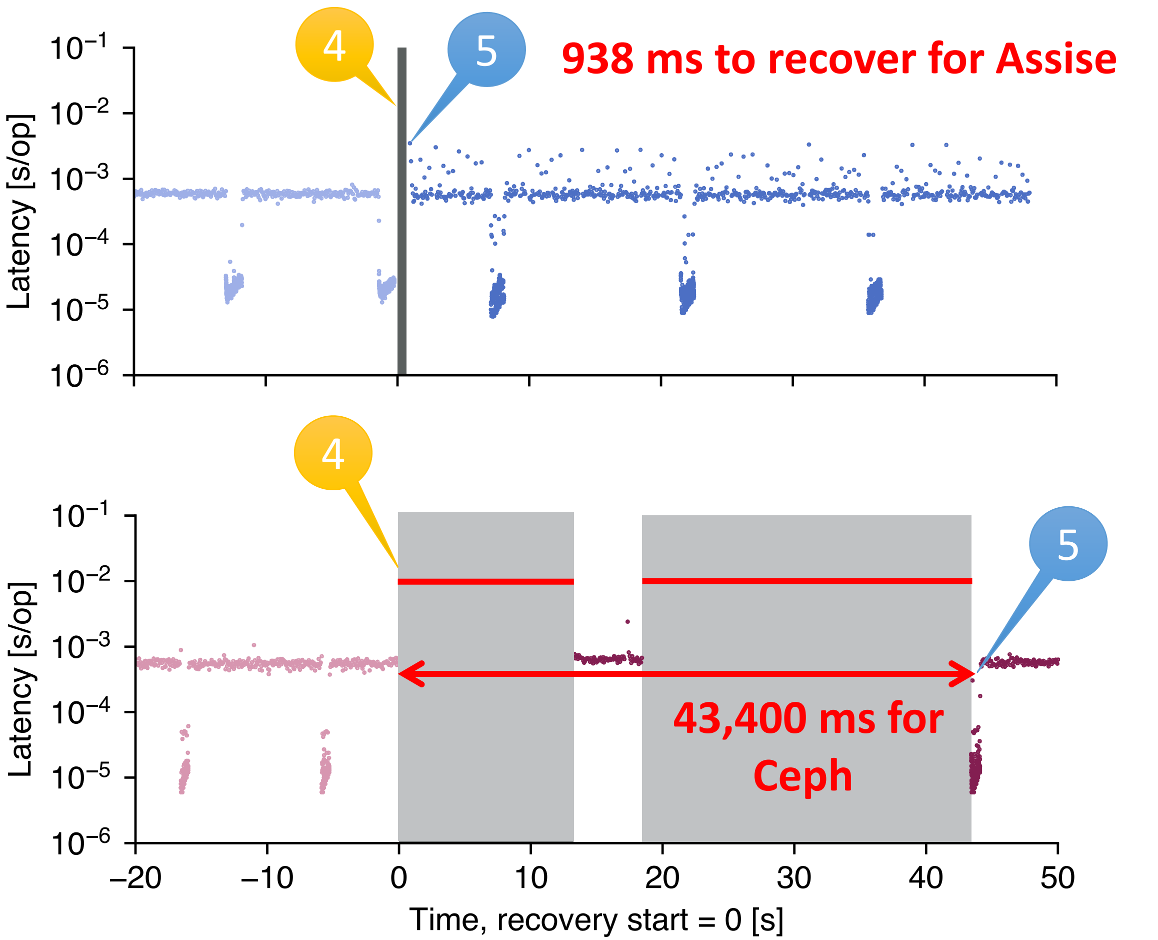




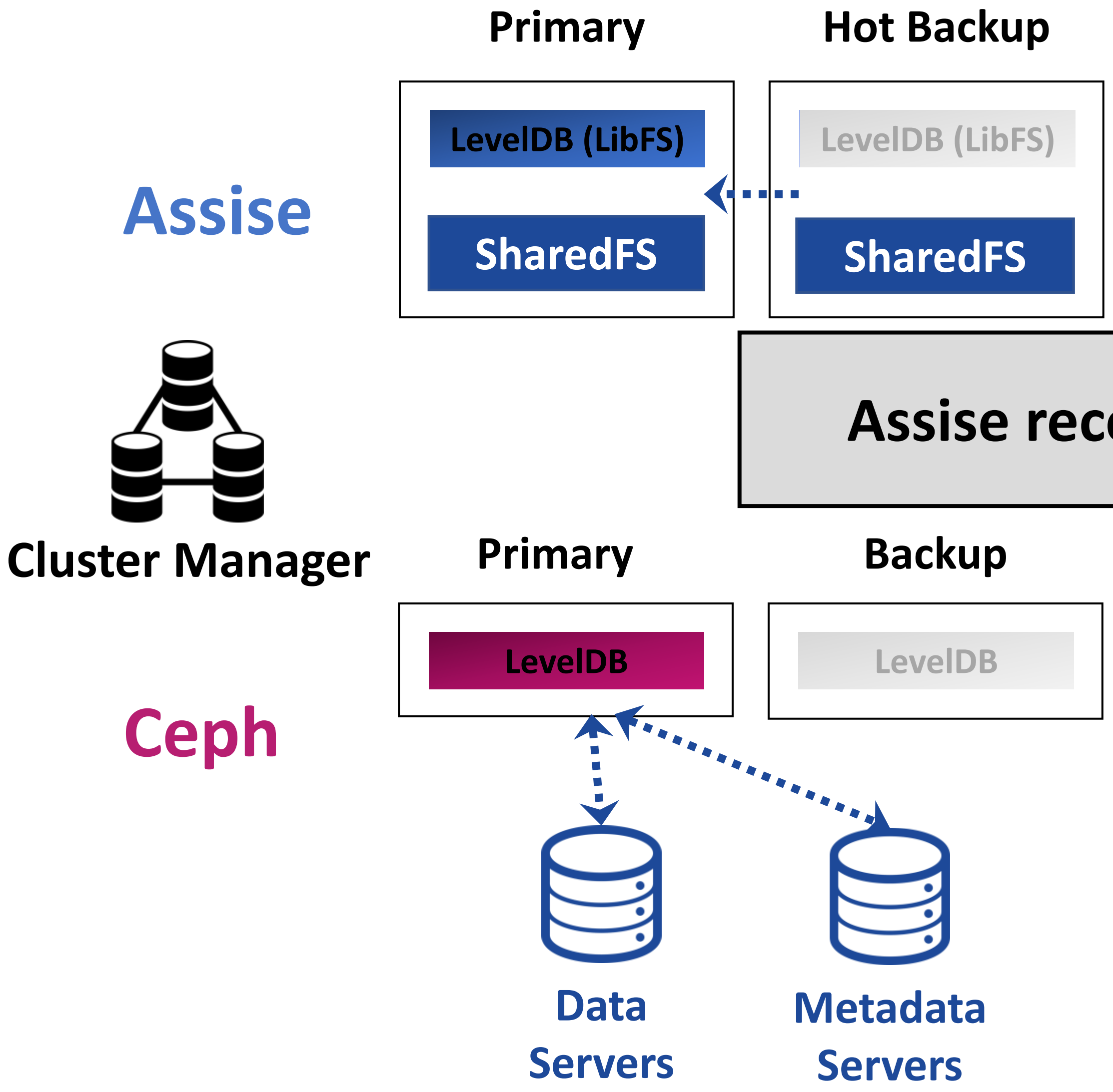
LevelDB: Primary recovery

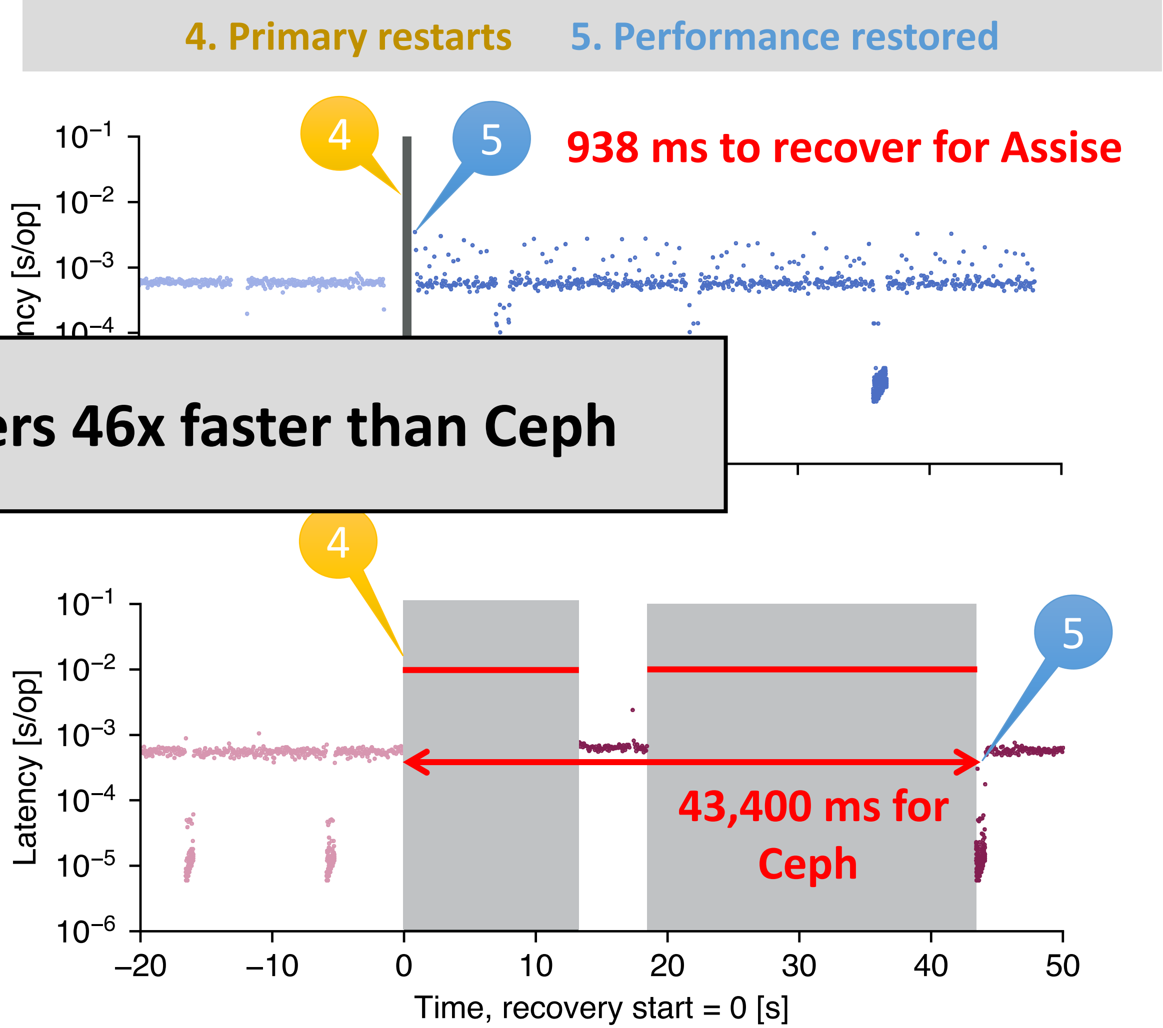


4. Primary restarts 5. Performance restored



LevelDB: Primary recovery





Assise recovers 46x faster than Ceph

Summary

- Assise: a distributed file system designed for client-local NVM
  - Low IO latency
  - Scalability
  - High availability

Userspace local/remote IO, crash-consistent
CC-NVM: scalable, linearizable, crash consistent
Fail-over to cache-hot client replicas

- 22x lower write latency , 103x faster fail-over, 6x better scalability
  - Comparisons against Ceph using real applications (Postfix, LevelDB, ...)

Source code available at: https://github.com/ut-osa/assise

Questions? Email assise@cs.utexas.edu