InfiniFS: An Efficient Metadata Service for Large-Scale Distributed Filesystems

Wenhao Lv, Youyou Lu, Yiming Zhang, Peile Duan, Jiwu Shu

Presenter: Yufei Wu

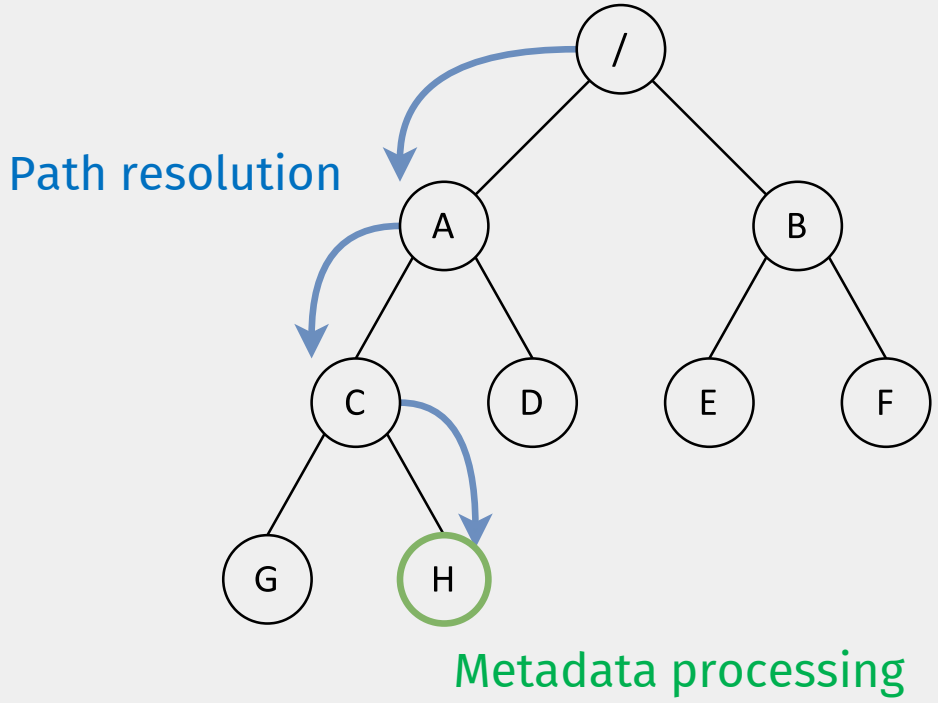
Filesystem Metadata

Filesystem directory tree

- Hierarchical namespace
- Directory and file metadata

Metadata operation

1. Path resolution
2. Metadata processing



Large-scale Filesystem

Single filesystem spans the entire datacenter

- Facebook: billions of files (Tectonic, FAST 21)
- Alibaba Cloud: tens of billions of files (thousands of Pangu)

Bring severe challenges to the metadata service

Outline

Challenges

- Achieve both metadata locality and load balancing
- High latency of path resolution
- High overhead of cache coherence maintenance

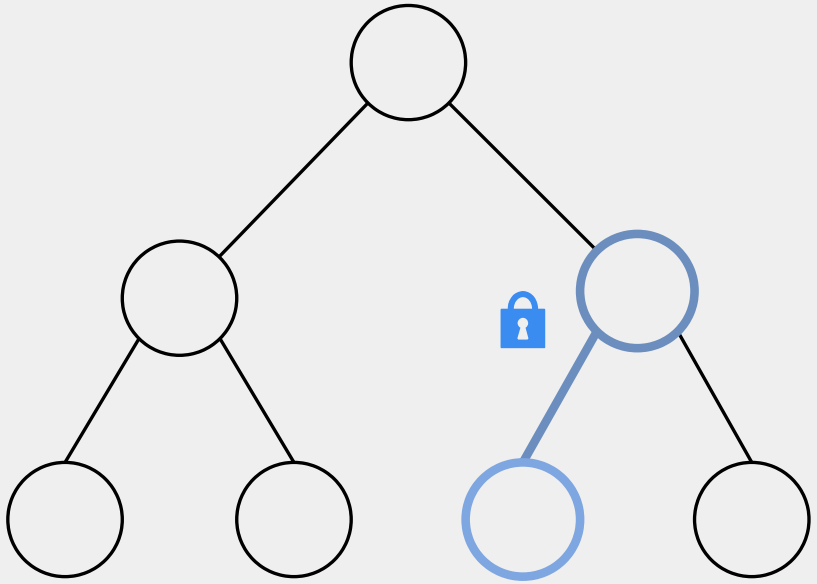
Design

Implementation

Evaluation

Conclusion

Challenge #1

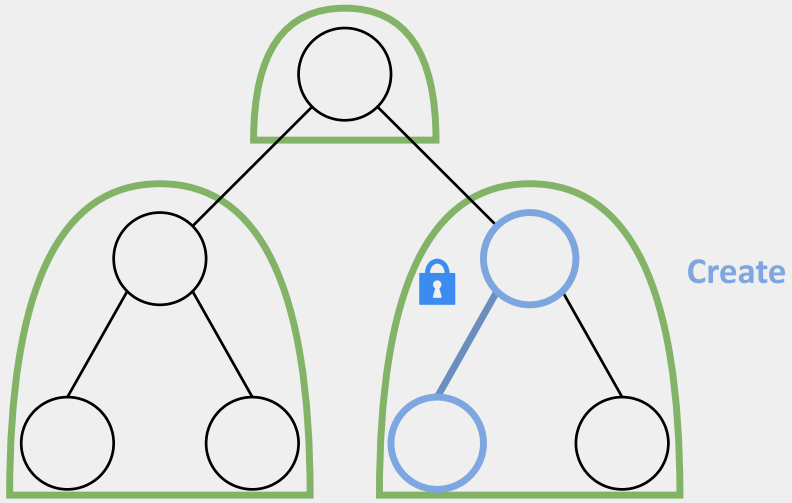


Create

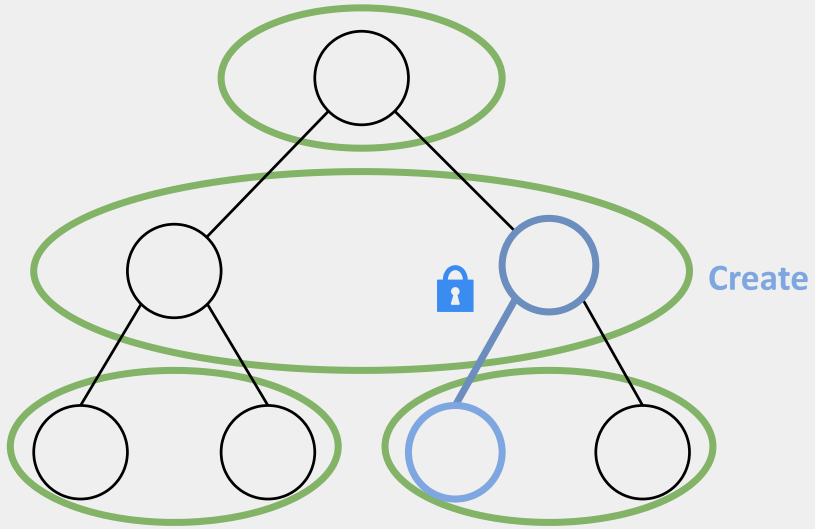
- Lock parent
- Update file metadata
- Update the directory metadata

Challenge #1

Achieve both metadata locality and load balancing



Coarse-grained

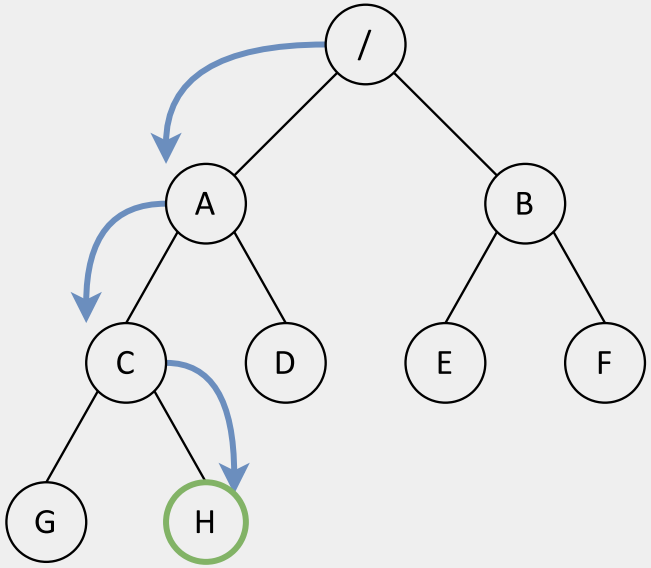


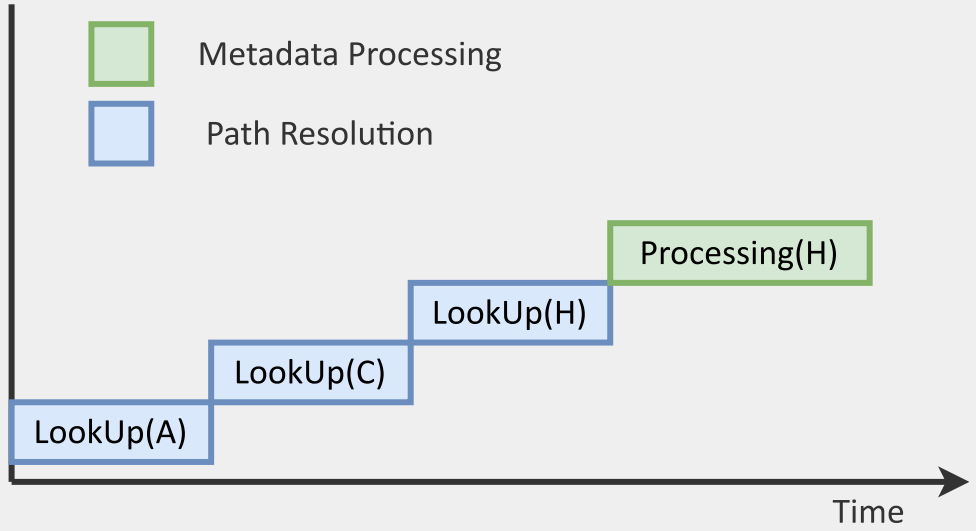
Fine-grained

Metadata locality	✓	✗
Load balance	✗	✓

Challenge #2

High latency of path resolution

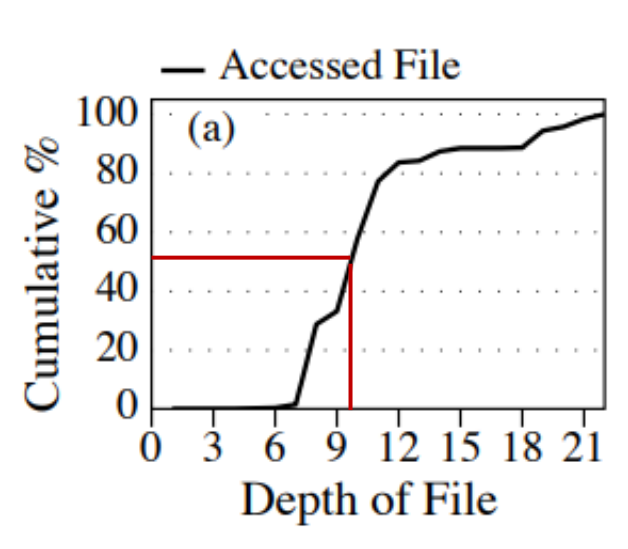


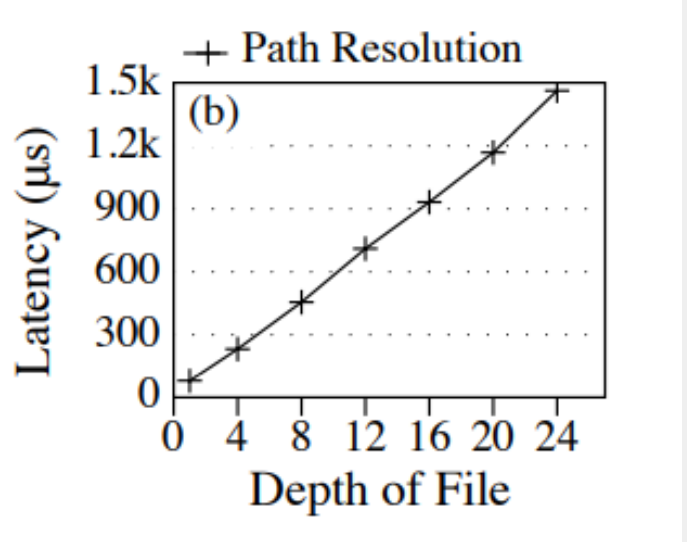


File depth → Latency

Challenge #2

High latency of path resolution

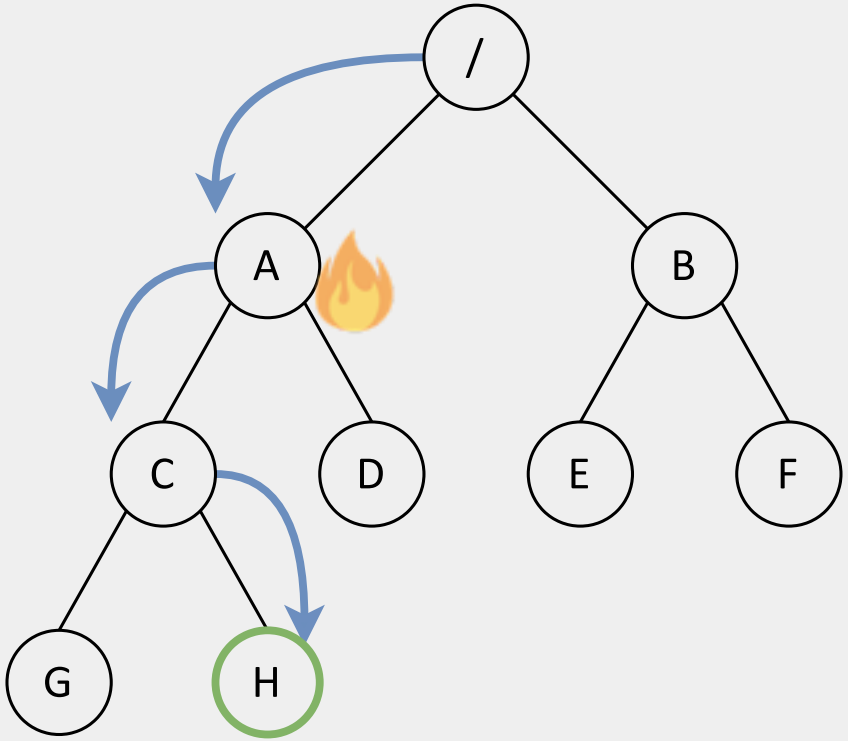




File depth ⟶ Latency

Challenge #3

High overhead of cache coherence maintenance

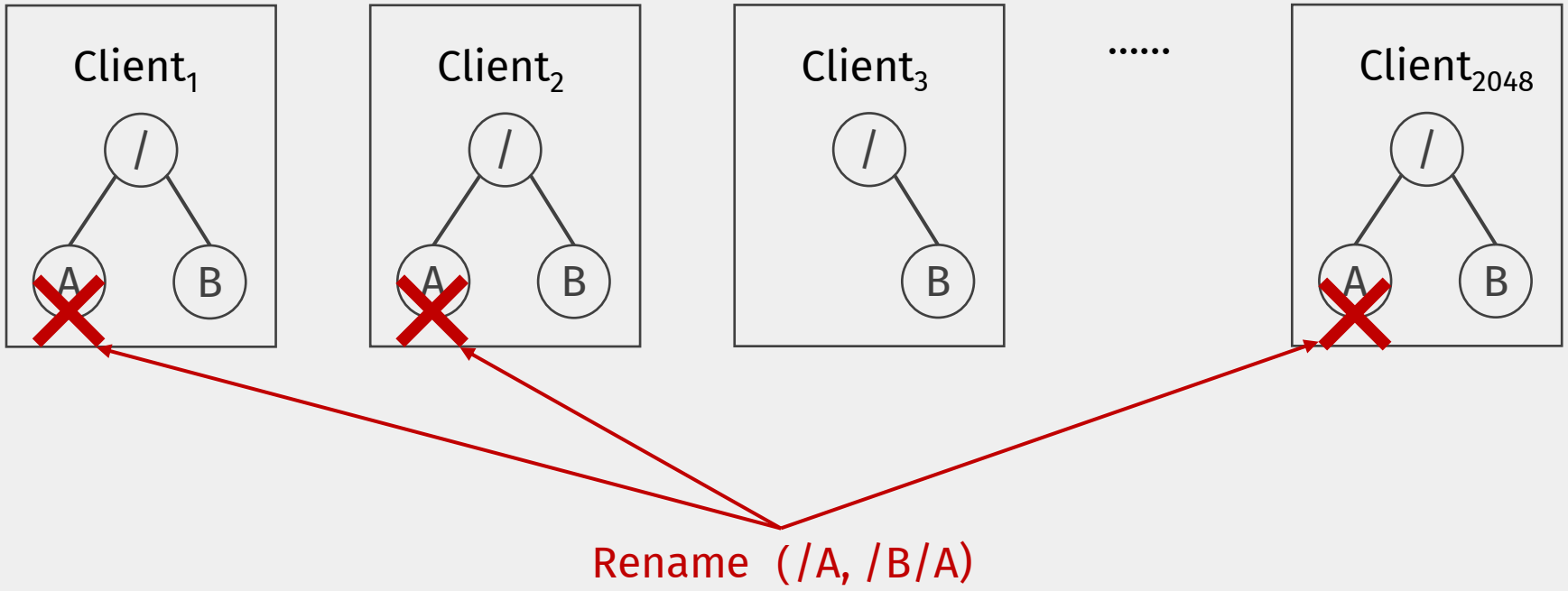


Why cache?

Near-root hotspots

Challenge #3

High overhead of cache coherence maintenance



Number of clients ⟶ Coherence overhead

Workload Characteristics

File Op	95.8%	Directory Op	4.2%	
open/close	54.9%	readdir	93.3%	
stat	12.9%	statdir	6.6%	
create	10.0%	mkdir	0.1%	
delete	12.4%	rmdir	0.1%	
rename	9.7%	rename	0.0%	0.0083%
set_permission	0.1%	set_permission	0.0%	

Three Pangu instance:

- Data analyzing
- Object storage
- Block storage

Outline

Challenges

Design

- Access-content decoupled partitioning
- Speculative path resolution
- Optimistic access metadata cache

Implementation

Evaluation

Conclusion

Overview

Clients

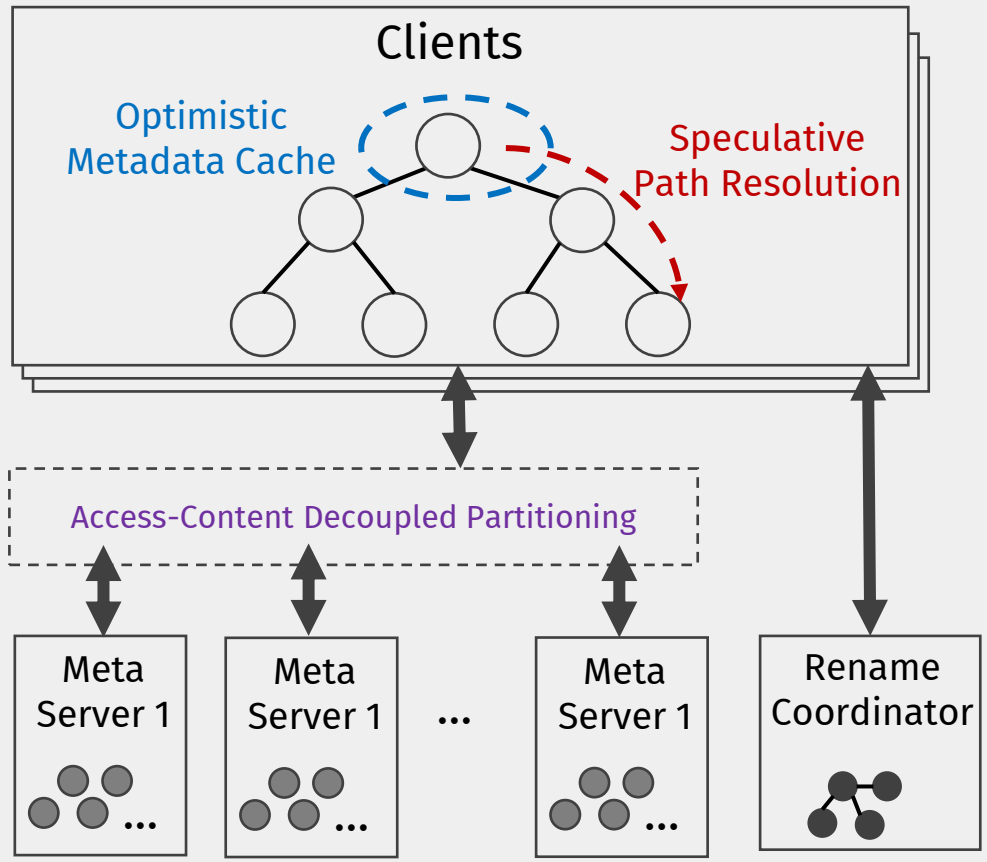
- Speculative path resolution
- Optimistic Metadata cache

Metadata servers

- Access-content decoupled partitioning

Rename coordinator

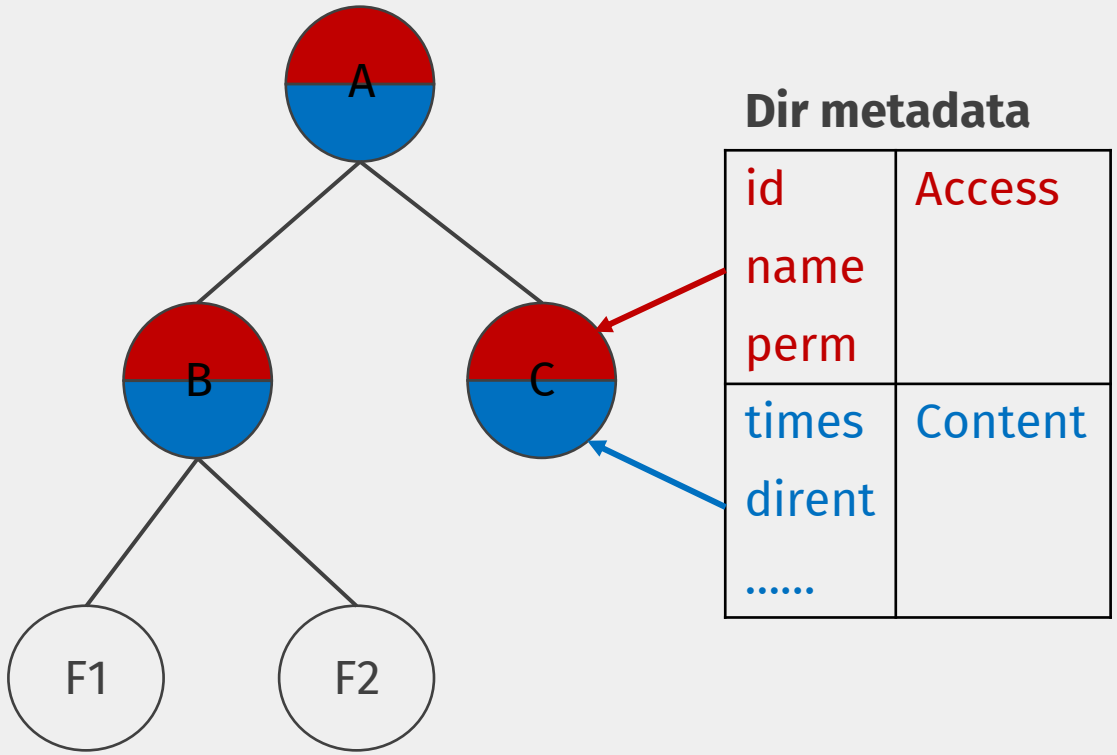
- Check concurrent directory renames



Access-Content Decoupled Partitioning

Decoupling directory metadata

- Access metadata
  Related to directory tree accessing
- Content metadata
  Related to the children

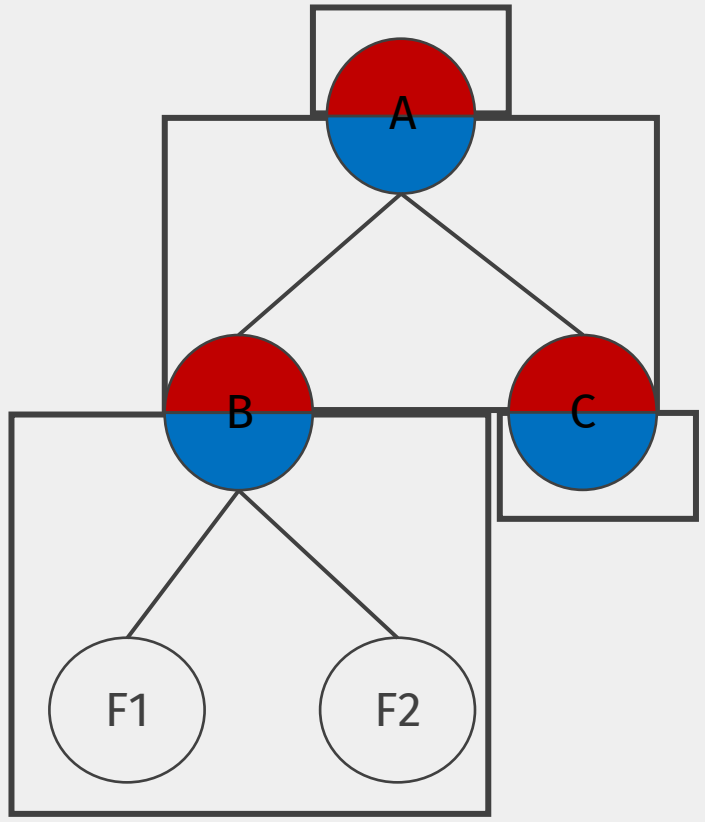


Access-Content Decoupled Partitioning

Decoupling directory metadata

Group related metadata for locality

- Access metadata with the parent
- Content metadata with the children

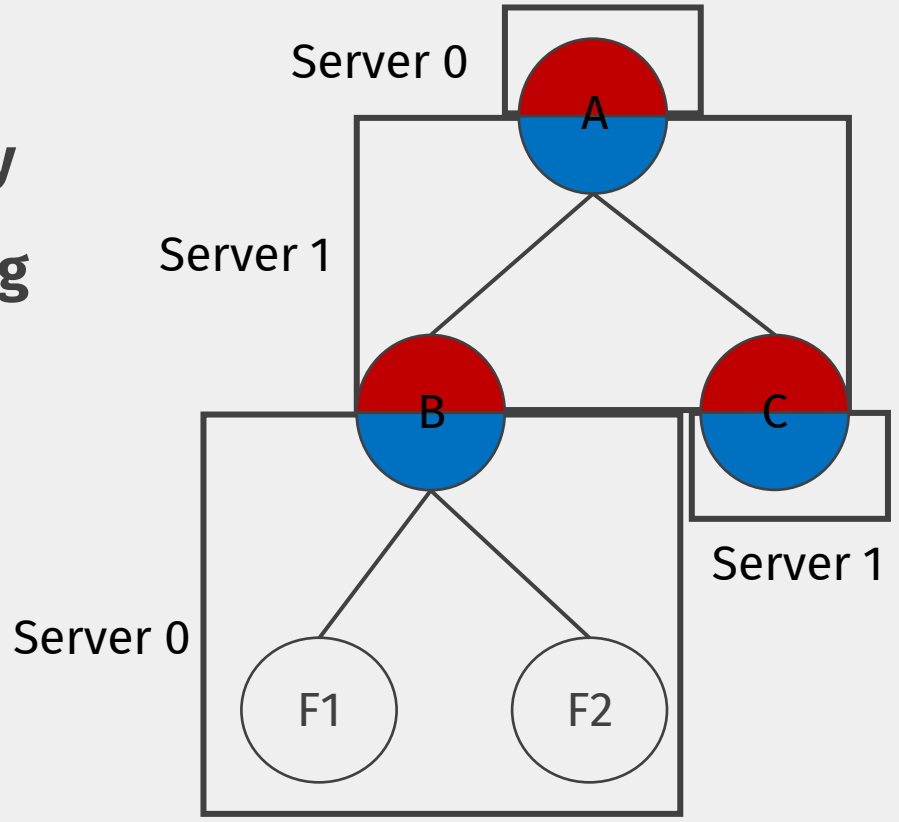


Access-Content Decoupled Partitioning

Decoupling directory metadata

Group related metadata for locality

Hash partitioning for load balancing

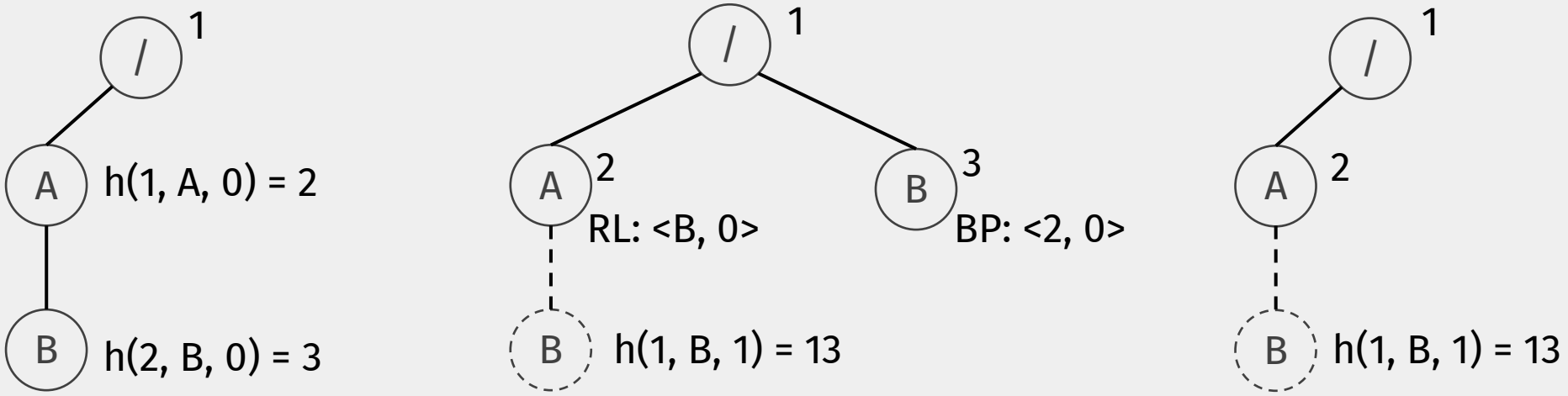


Access-Content Decoupled Partitioning

Metadata Objects	Key	Value	Partitioned by
Dir Access Metadata	`pid, name`	`id, permission`	`pid`
Dir Content Metadata	`id`	`entry list, timestamps, etc.`	`id`
File Metadata	`pid, name`	`file metadata`	`pid`

Speculative Path Resolution

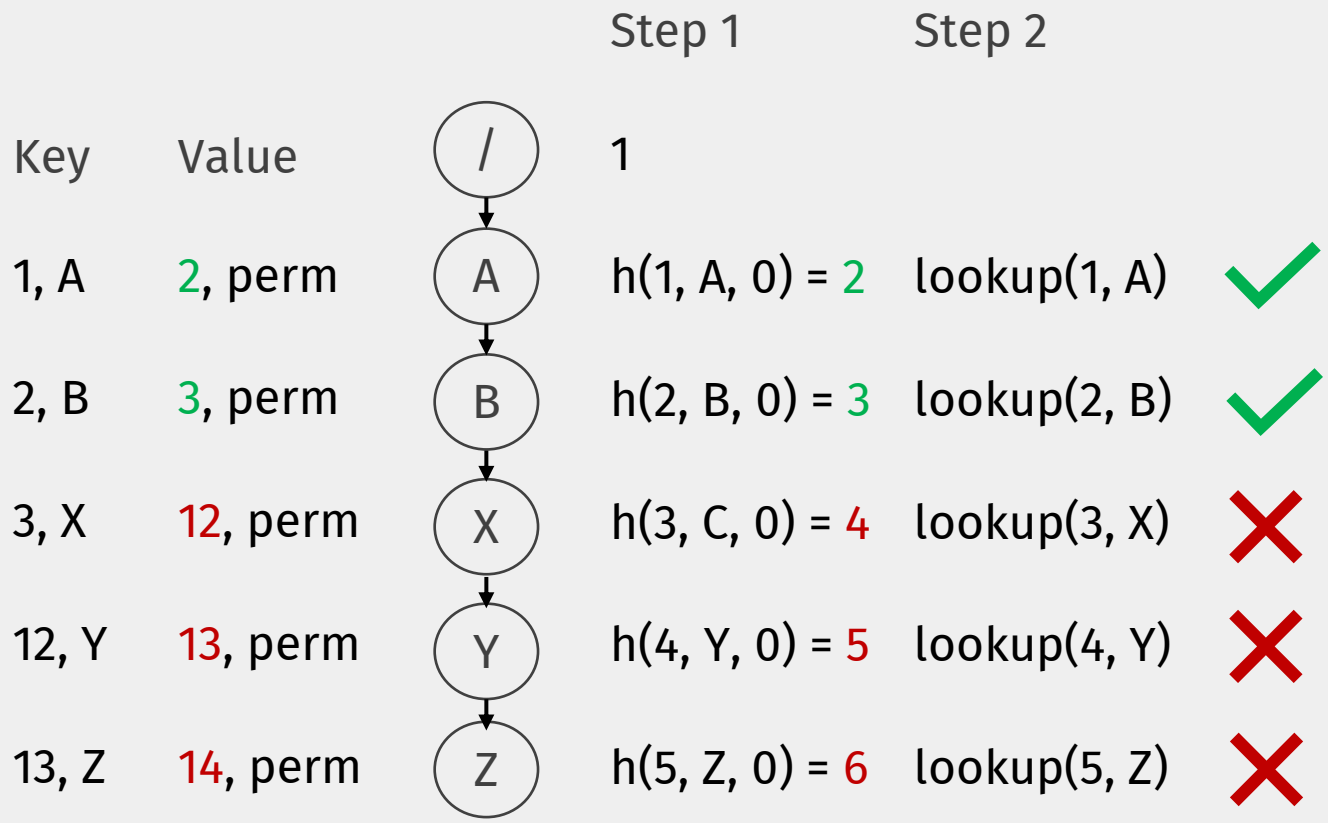
Predictable directory ID: SHA256(parent ID, name, version)



Speculative Path Resolution

Parallel path resolution

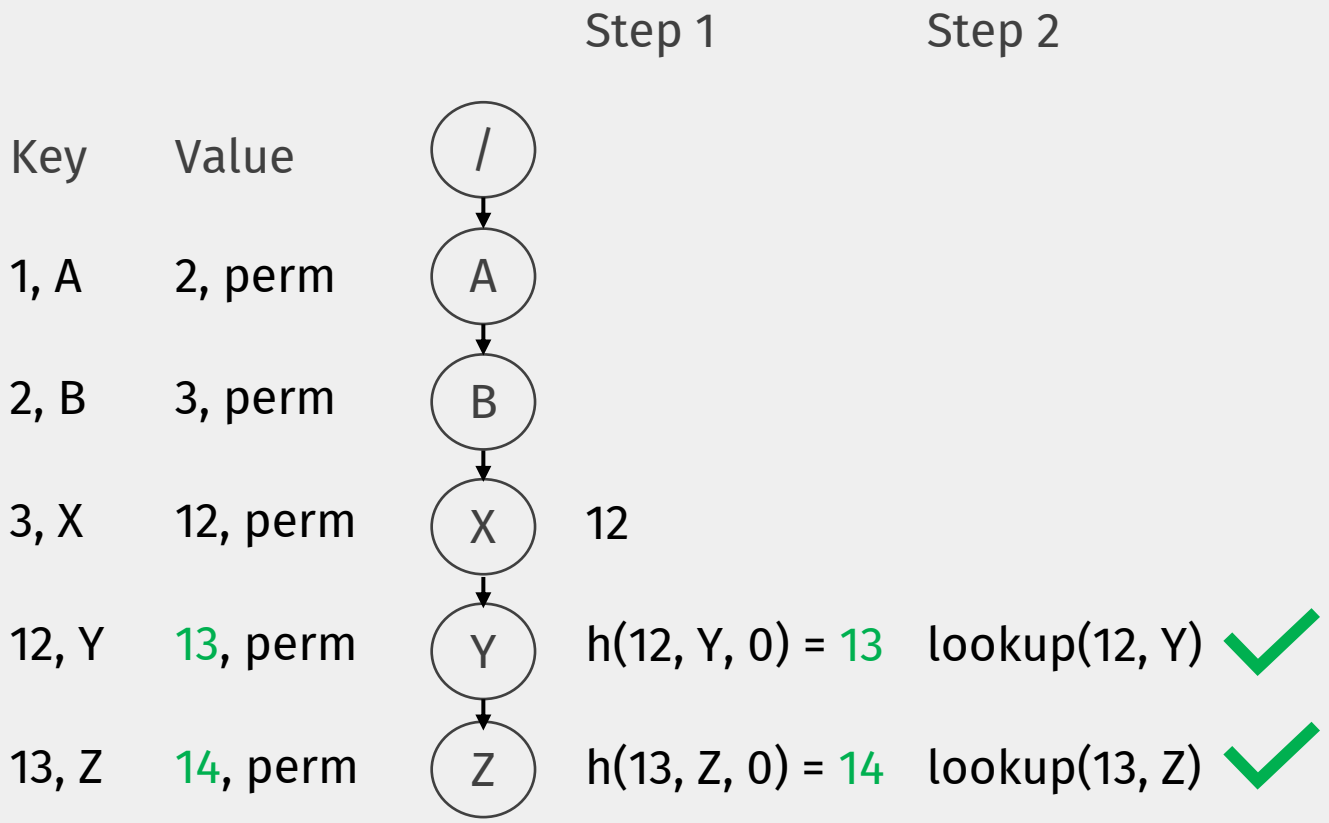
1. Predict directory IDs
2. Send lookups in parallel



Speculative Path Resolution

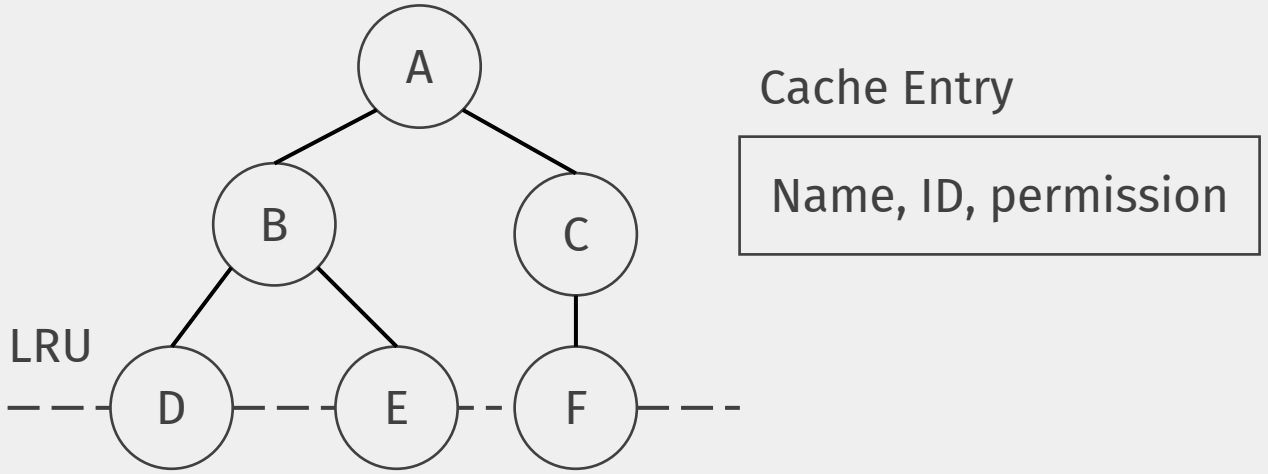
Parallel path resolution

1. Predict directory IDs
2. Send lookups in parallel
3. Repeated until finished

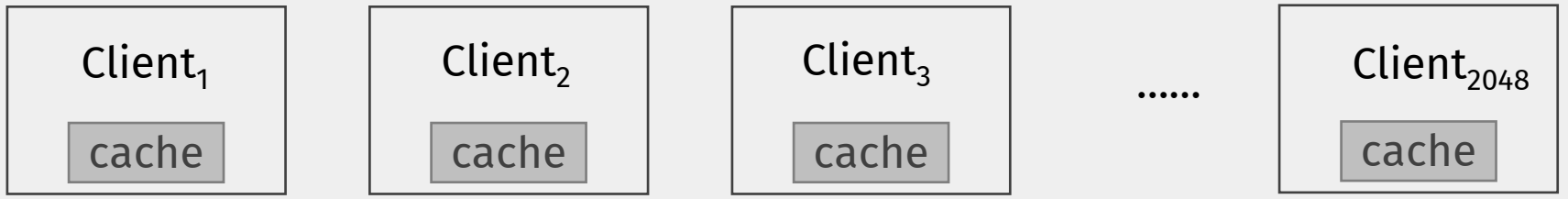


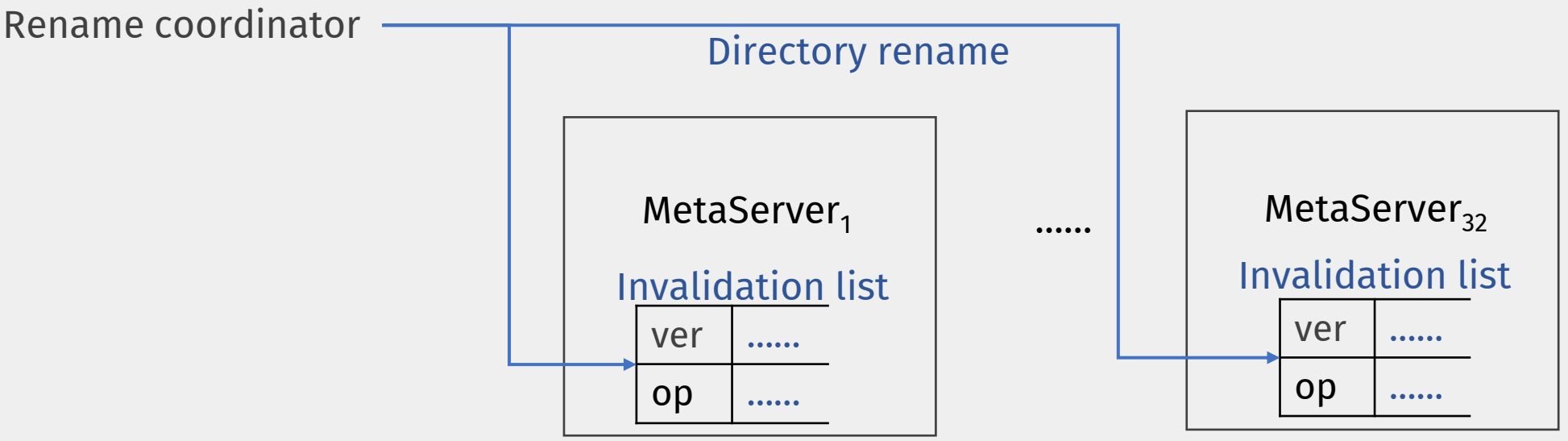
Optimistic Access Metadata Cache

Cache organization

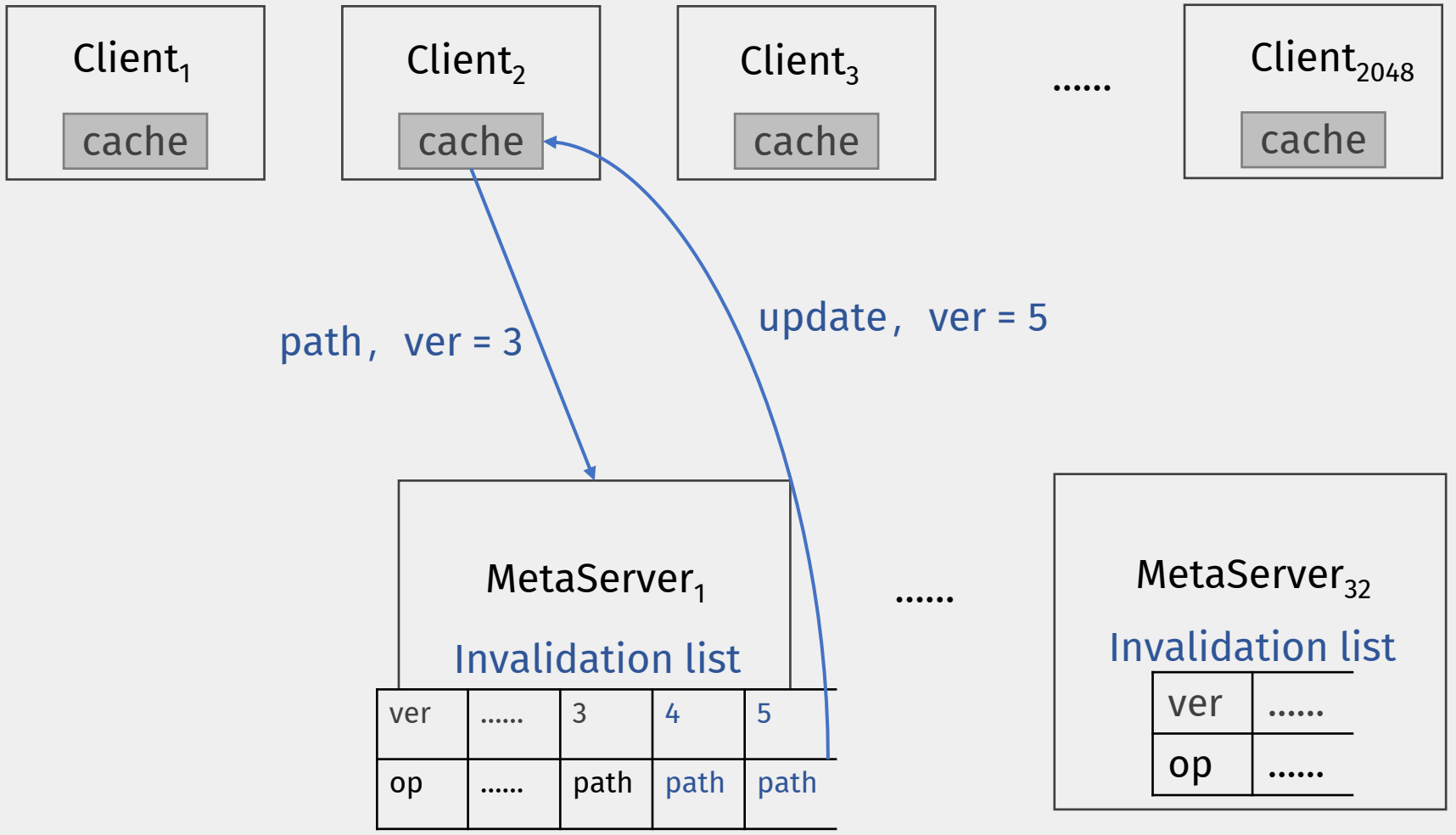


Optimistic Access Metadata Cache





Optimistic Access Metadata Cache



Outline

Challenges

Design

Implementation

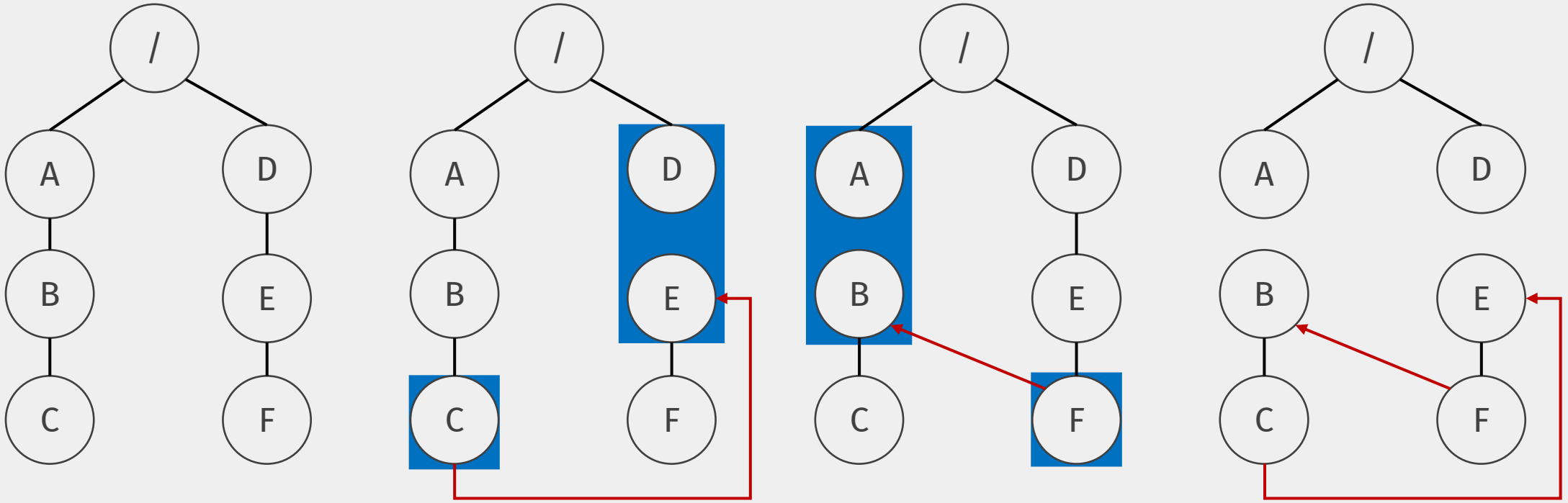
Evaluation

Conclusion

Transactional Metadata Operation

1. Single-server operations
2. Two-server operations
   `mkdir/rmdir/statdir` and file `rename`
   Two-phase commit
3. Rename coordinator
   Directory `rename` and `set_permission`

Orphaned Loop



Rename coordinator:
tracks the source and destination paths of in-flight directory rename

Outline

Challenges

Design

Implementation

Evaluation

Conclusion

Setup

Hardware

CPU	Intel Xeon Platinum 2.50GHz, 96 cores
Memory	Micron DDR4 2666MHz 32GB × 16
Storage	RAMdisk
Network	ConnectX-4 Lx Dual-port 25Gbps

Compared System

- LocoFS (SC 17), IndexFS (SC 14), HopsFS (FAST 17), CephFS

Benchmark

- Mdtest
- All tests create files of zero length

Overall Performance

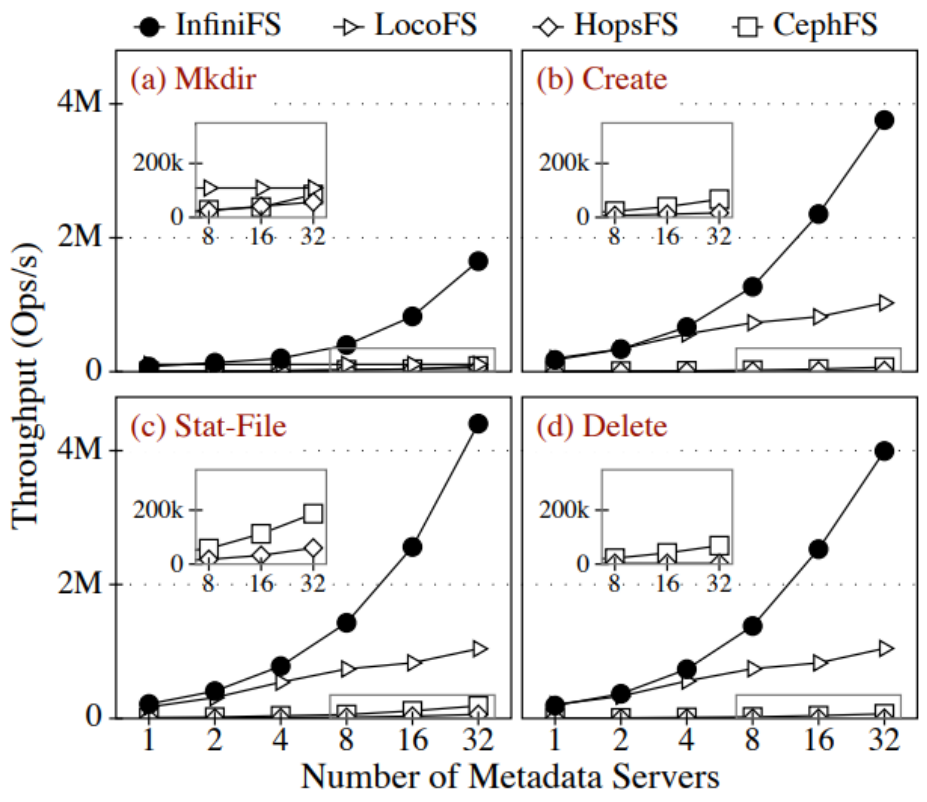
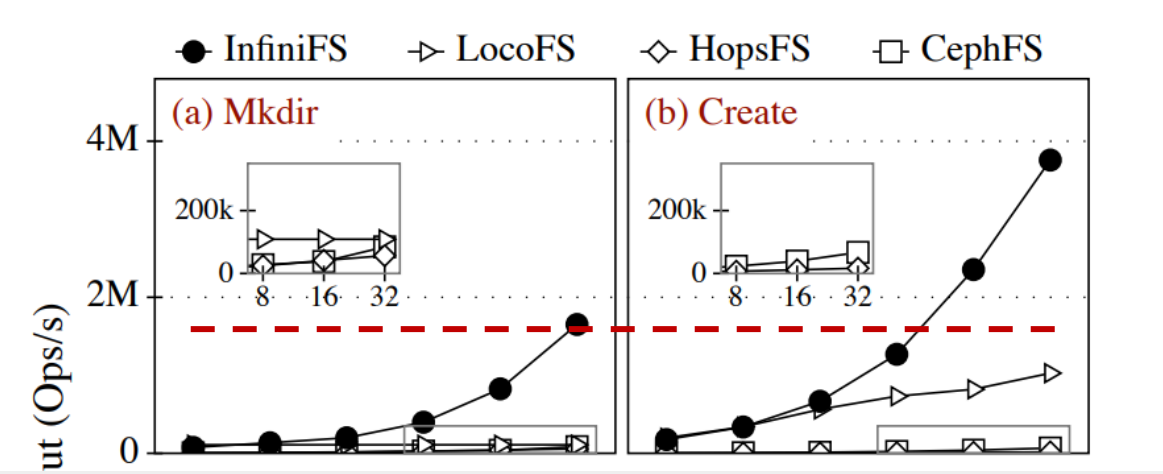


Figure 9: Throughput scalability of metadata operations (`mkdir`, `create`, `stat`, and `delete`). 500 million files.

Near linear scalability

- Path resolution
  - Client-side cache
  - Hash partitioning metadata
- Metadata processing
  - Group for locality

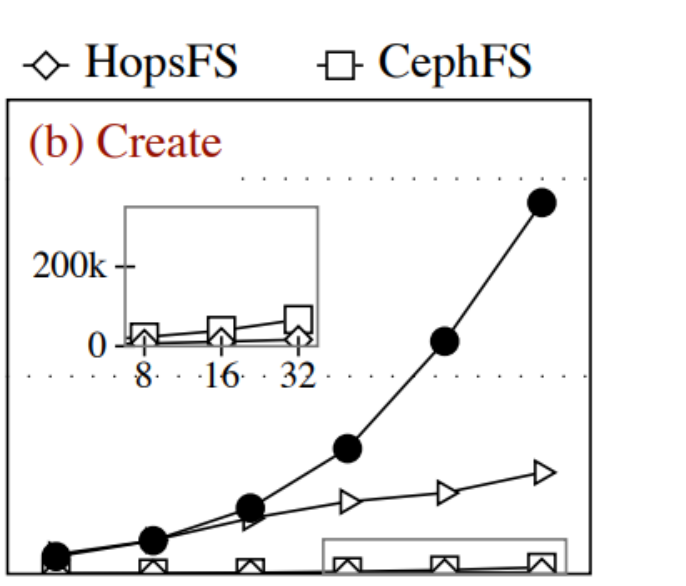
Overall Performance



`Mkdir < create`

Distributed Transaction

Overall Performance



One server, InfiniFS < LocoFS

- Hash based KV store

Overall performance

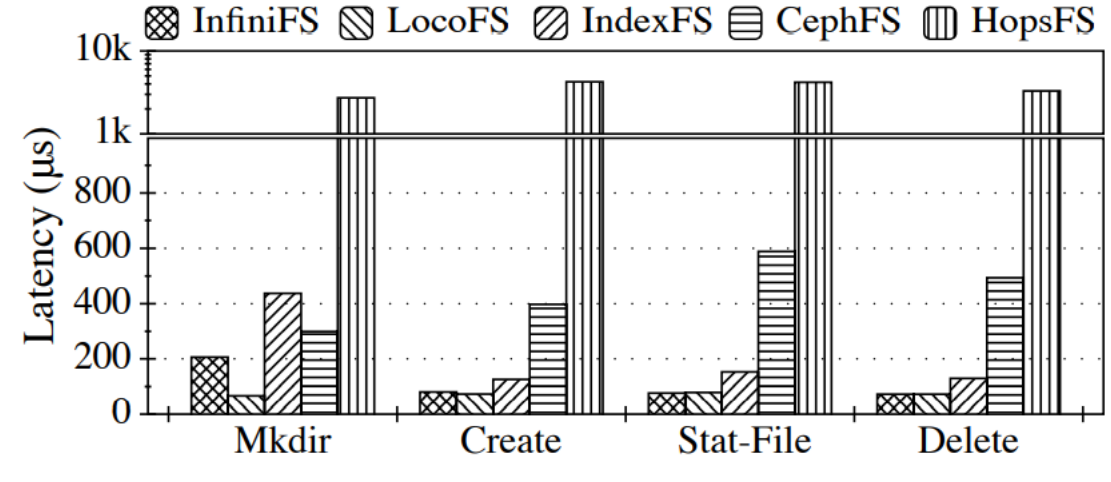


Figure 10: Latency of metadata operations.

Comparable with LocoFS

- Speculative Path Resolution
- Optimistic access cache

`Mkdir` has higher latency

- Distributed Transaction

32 servers

Factor Analysis

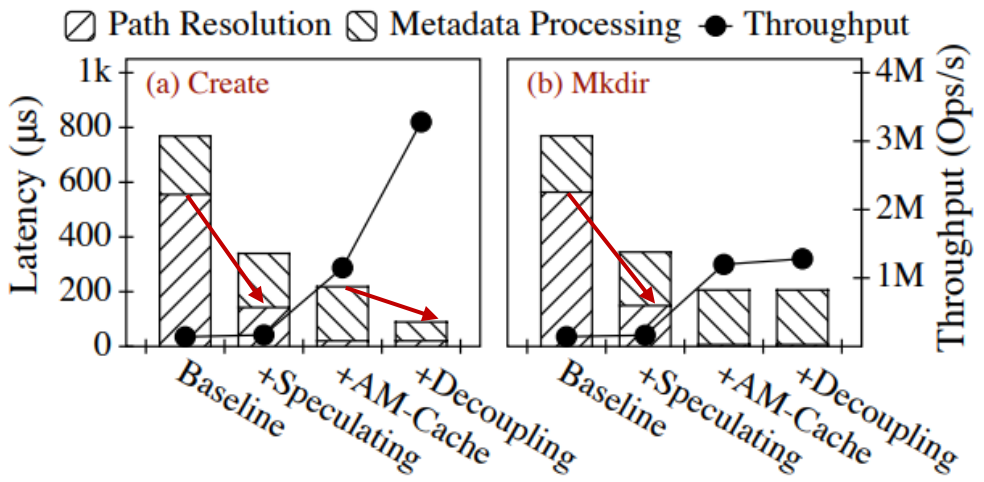


Figure 11: Contributions of design features to the latency (left Y-axis) and throughput (right Y-axis) of INFINIFS. Different segments inside the bar represent the decomposed latency. Design features are accumulated.

32 servers
1024 clients
Depth of 10

Large-Scale Directory Tree

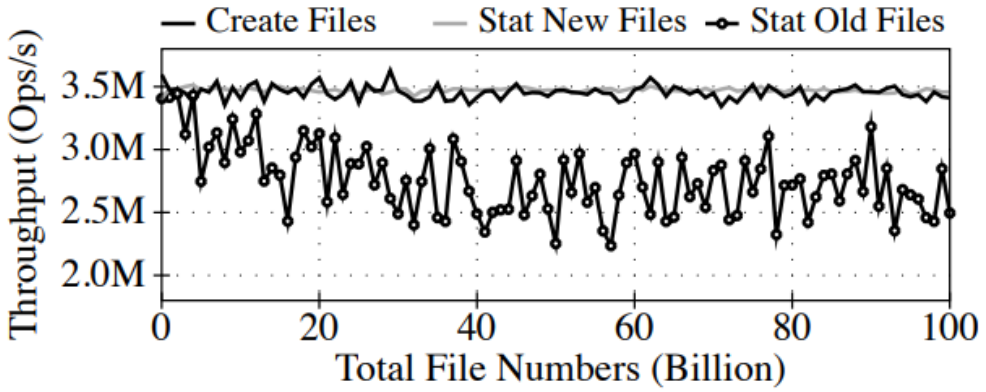


Figure 12: Throughput of INFINIFS with 100 billion files.

Stat Old Files:
RocksDB compaction

32 servers
1024 clients

Overhead of Misprediction

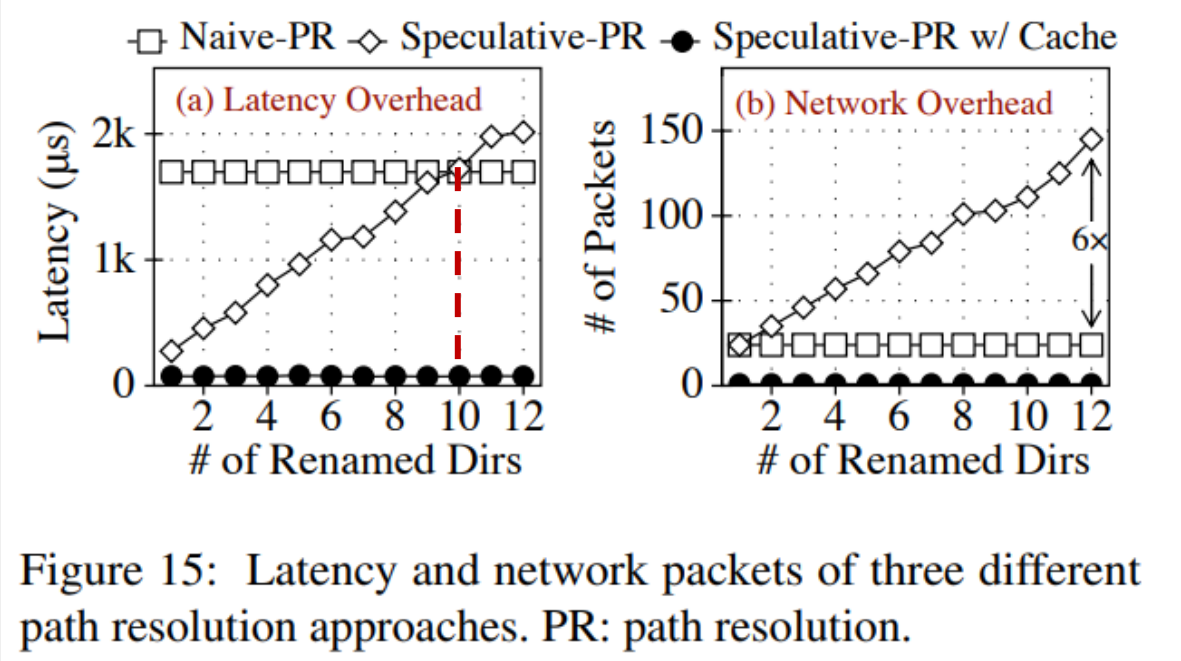


Figure 15: Latency and network packets of three different path resolution approaches. PR: path resolution.

Depth of 24
10K files per client

Cache Efficiency

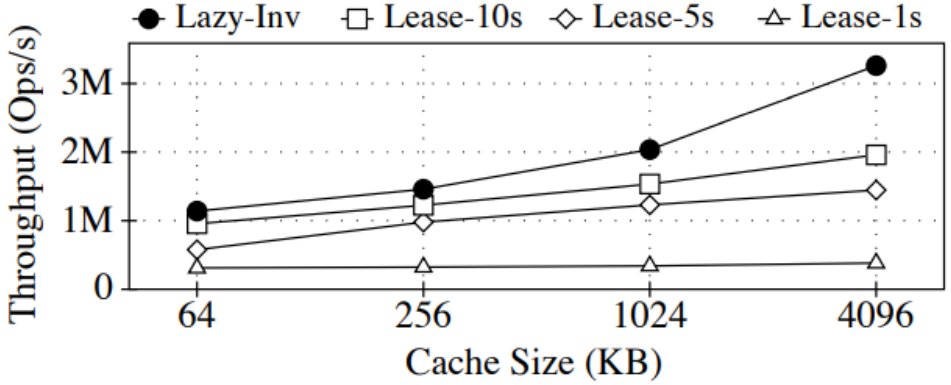


Figure 13: Comparisons of the lazy invalidation with the lease mechanism. The lease expiration time is set to 1s, 5s, and 10s.

Lease expiration time
- Write latency

Cache size
- Near-root hotspot

32 servers
2048 clients
`Stat` files

Outline

Challenges

Design

Implementation

Evaluation

Conclusion

Conclusion

Design

- Access-content decoupled partitioning
- Speculative path resolution
- Optimistic access metadata cache

Q&A

Google Colossus

Big Colossus 的 metadata 存储在 Spanner

这个 Spanner 的 data 存储在 Small Colossus

Small Colossus 的 metadata 较少，单个元数据节点就可以