ShardFS

Replication v.s. Caching Strategies

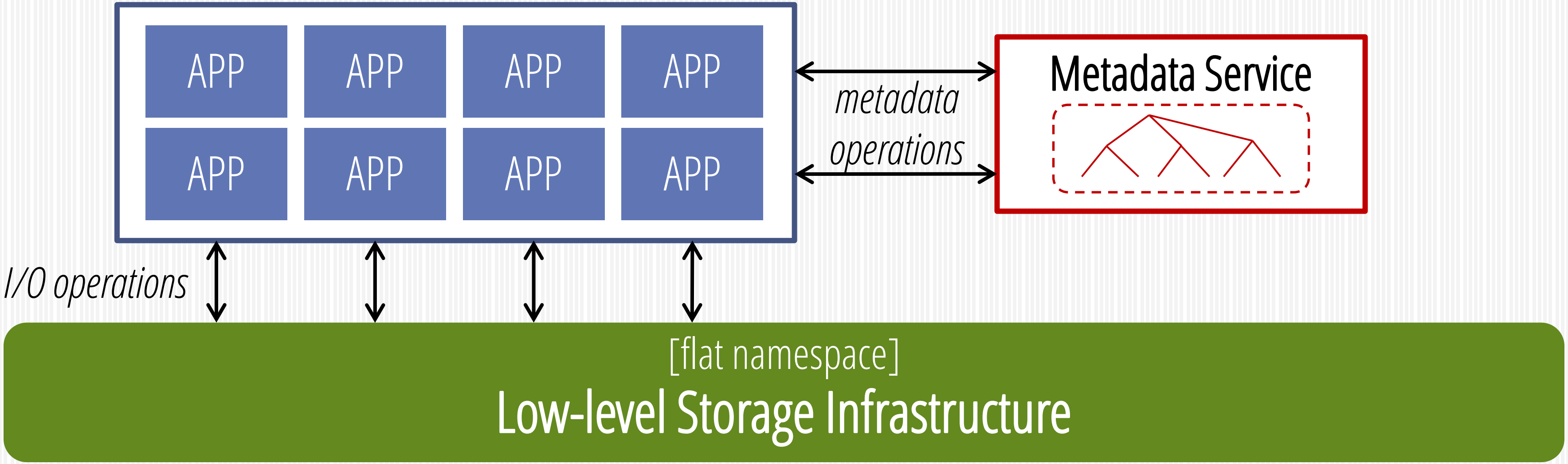
for Distributed Metadata Management in Large-scale Data Centers

IndexFS

Qing Zheng

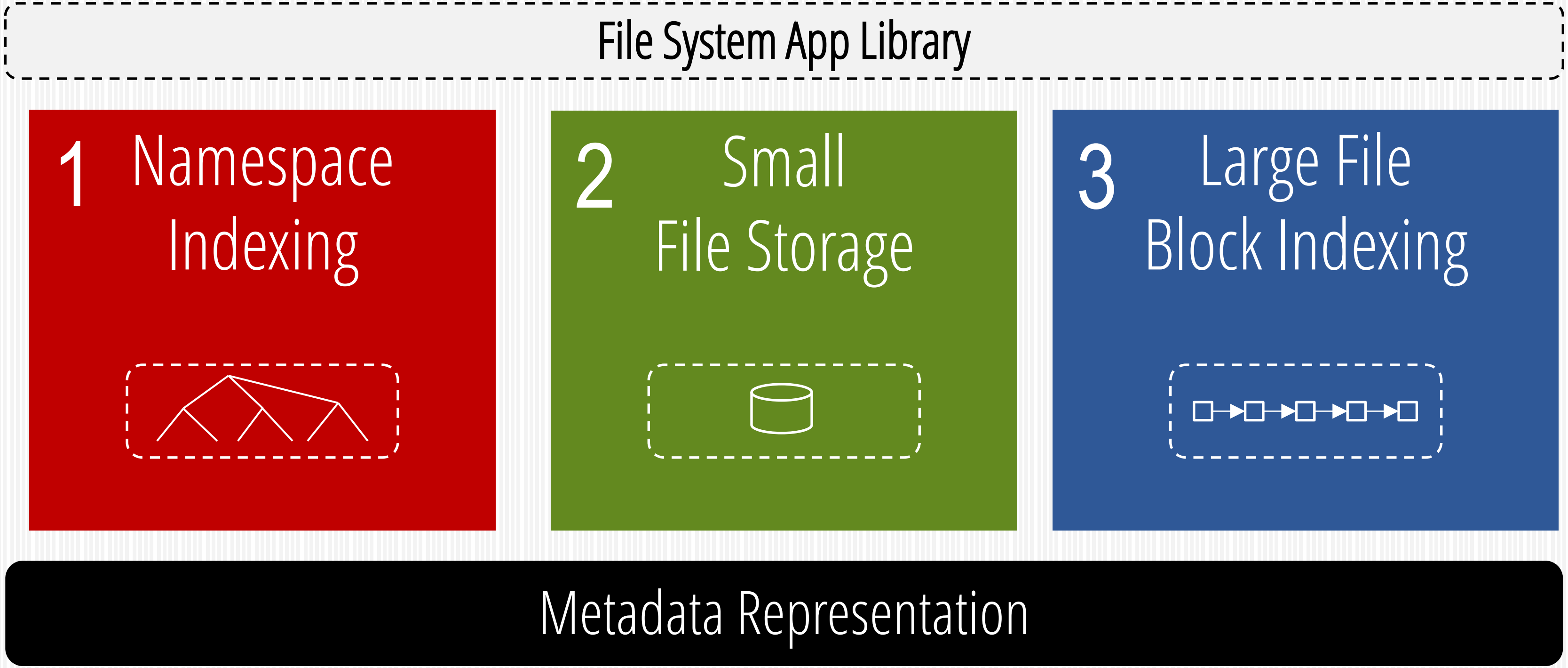
Lin Xiao, Kai Ren, Garth Gibson

File System Architecture



Parallel data path with decoupled metadata path

Metadata = 1 + 2 + 3



File System App Library

1 Namespace Indexing

2 Small File Storage

3 Large File Block Indexing

Metadata Representation

Decoupled != Scalable

Single metadata server

HDFS, Lustre 1.x

Statically partitioned metadata servers

PVFS, Federated HDFS, NFS v4.1

Many existing metadata service don’t scale

Our Goal

is To Have Really

Scalable

Metadata

Our Goal is To Have Really Scalable Metadata

Outline

1. Pathname lookup

   important limitation on scalability

2. Client-side caching

   represented by IndexFS

3. Replicated state

   represented by ShardFS

4. Experimental results

Path Resolution

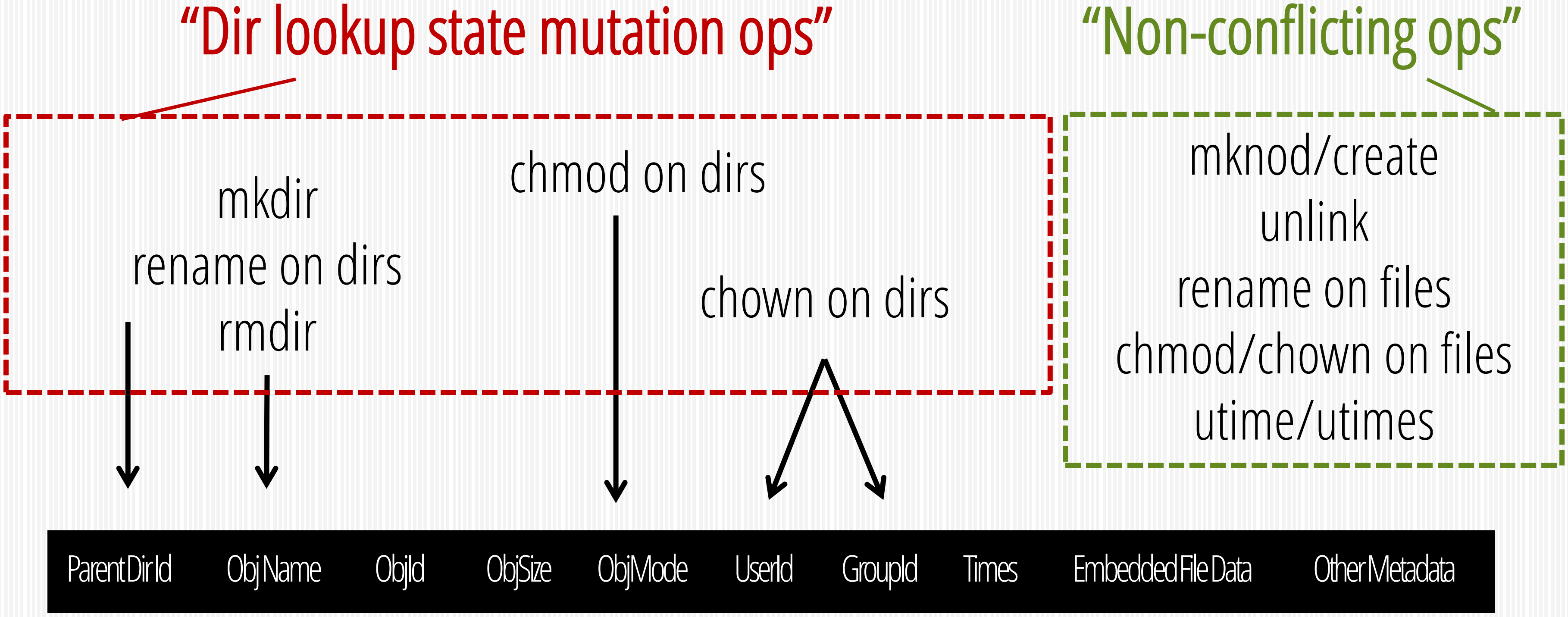
Hierarchical permission checking

In order to resolve /a/b/c/..., need to test /a, a/b, b/c, ...

1) *Permissions to lookup names under an intermediate directory*
2) *The existence of the name*
3) *The name represents a directory*

A set of recursive tests starting from the root

2 Categories of Ops



"Dir lookup state mutation ops"

mkdir
rename on dirs
rmdir

chmod on dirs

chown on dirs

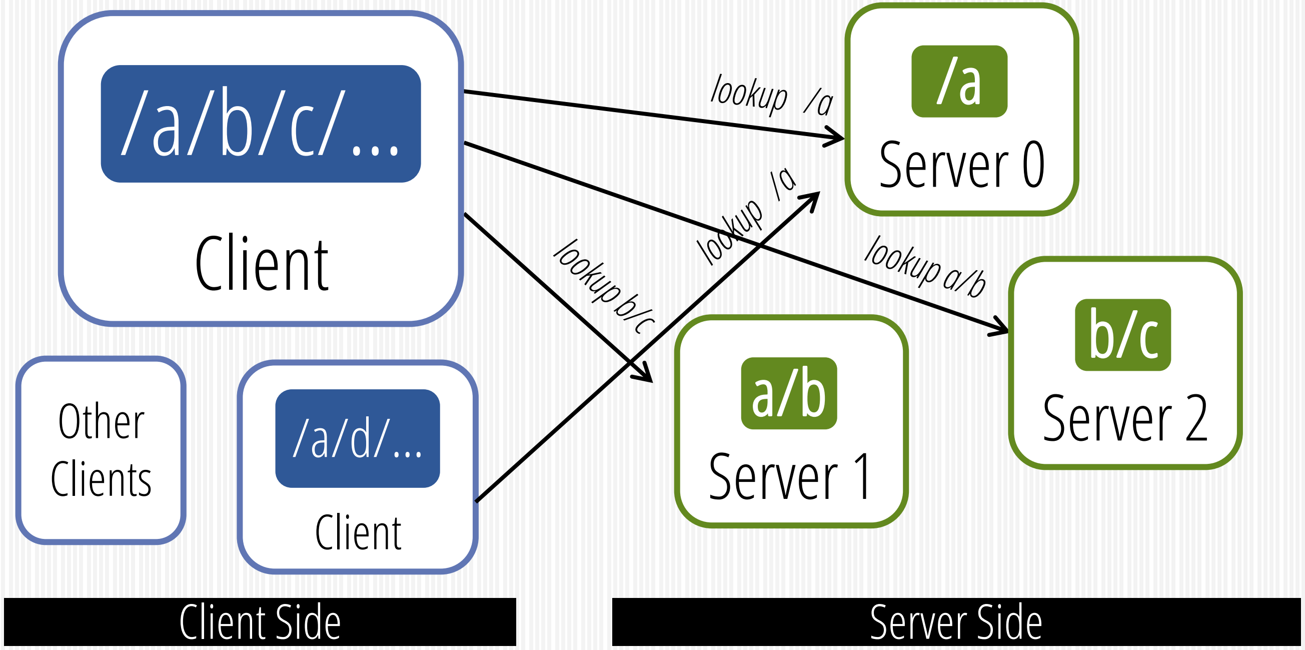
"Non-conflicting ops"

mknod/create
unlink
rename on files
chmod/chown on files
utime/utimes

ParentDirId	ObjName	ObjId	ObjSize	ObjMode	UserId	GroupId	Times	Embedded File Data	Other Metadata

Naive Implementation

1 lookup RPC to server for each intermediate dir



BOTTLENECKS

1. Repeated RPCs
2. Hot spot servers holding names at the top of the tree

Outline

1. Pathname lookup

   important limitation on scalability

2. Client-side caching

   represented by IndexFS

3. Replicated state

   represented by ShardFS

4. Experimental results

Design Choice #1

Lease and cache dir lookup states at clients

Block mutation ops until all leases have expired

Cache-Entry	Expiration-time

client-side cache table

Cache-Id	Max-expiration-time

server-side cache table

Fewer repeated RPCs & simple server states

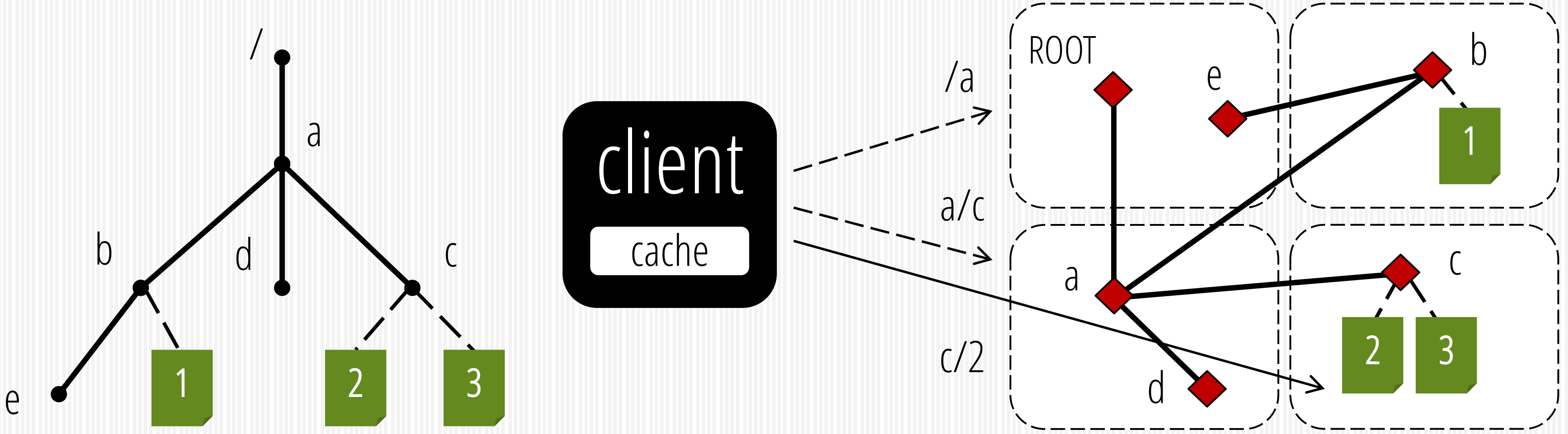
Outline

1. Pathname lookup
   important limitation on scalability
2. Client-side caching
   represented by IndexFS
3. Replicated state
   represented by ShardFS
4. Experimental results

IndexFS Design

Distributes namespace on a per-dir partition basic

Path resolution conducted by clients with an consistent lease-based lookup cache

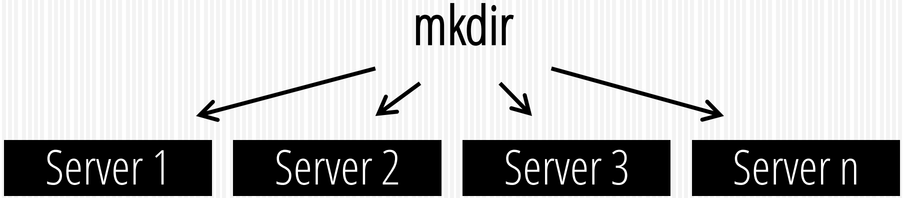


Outline

1. Pathname lookup
   important limitation on scalability
2. Client-side caching
   represented by IndexFS
3. Replicated state
   represented by ShardFS
4. Experimental results

Design Choice #2

Replicates dir lookup states to all servers & broadcasts mutation ops to all servers



mkdir

Server 1 Server 2 Server 3 Server n

Principally a better decision if #client >> #server

Outline

1. Pathname lookup

   important limitation on scalability

2. Client-side caching

   represented by IndexFS

3. Replicated state

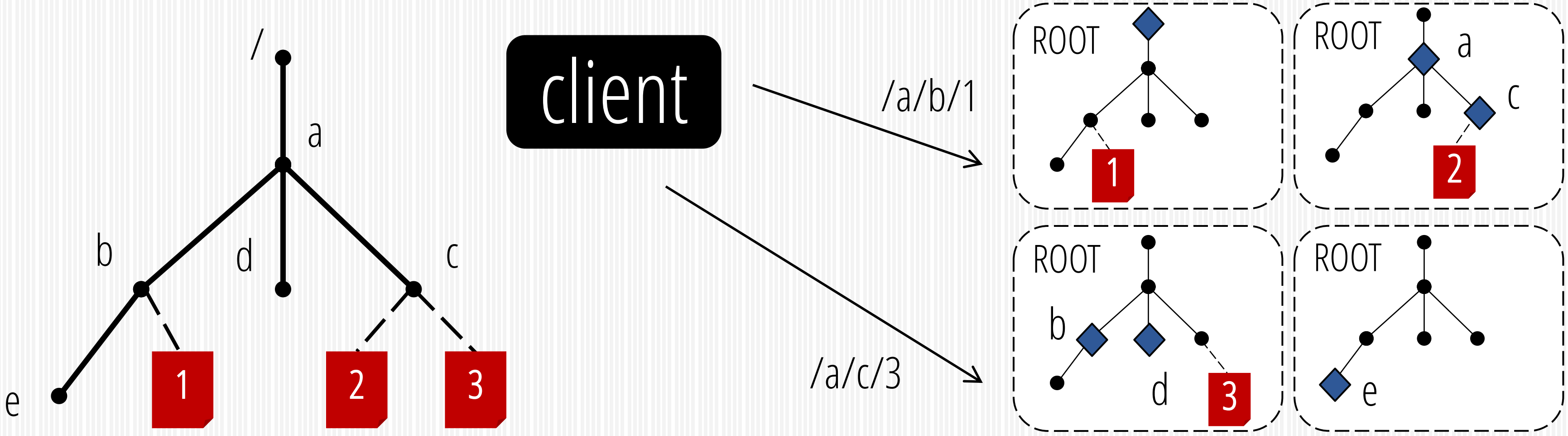
   represented by ShardFS

4. Experimental results

ShardFS Design

Distributes namespace on a per-file basic (sharding)

All metadata servers can accept new files and perform path resolution



RPC Amplification

** dir lookup state mutation op*		IndexFS	ShardFS
path resolution		0 ~ #path_depth	0
mknod	#path_lookups +	1	1
unlink/getattr		1	1
mkdir*		1 + 1	#metadata_servers
rmdir*/readdir		1 + #partitions	#metadata_servers
chmod/chown on file		1	1
chmod*/chown* on dir		1	#metadata_server
utime on file/dir		1	1

Micro Benchmark

YCSB++ [SoCC11]

Balanced tree

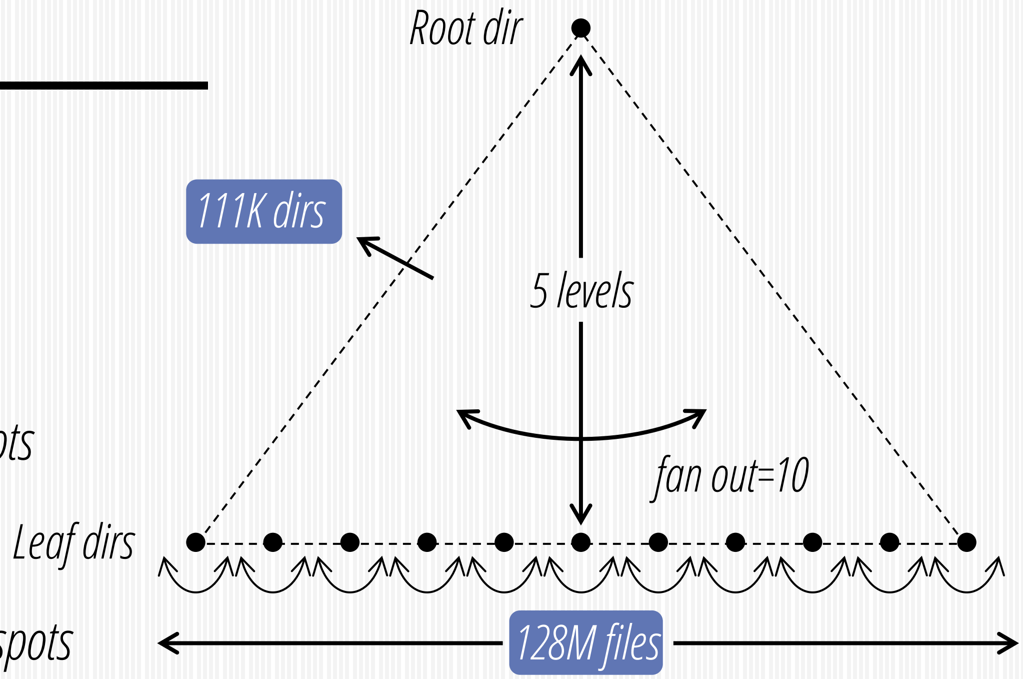
1280 files per leaf dir

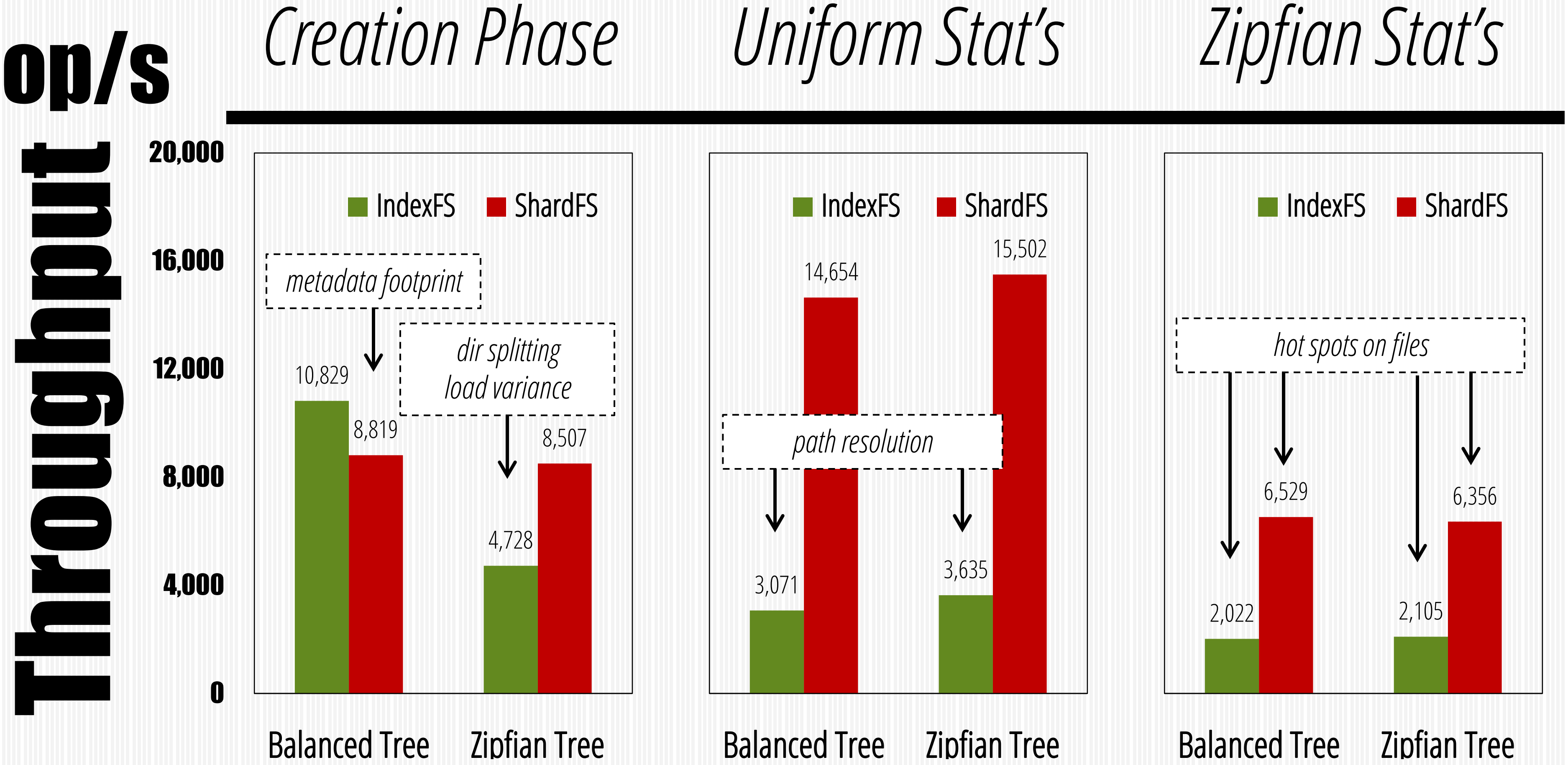
Zipfian tree

files distributed unevenly, creating static hot spots

Uniform/zipfian stat's

random getattrs on files, creating dynamic hot spots





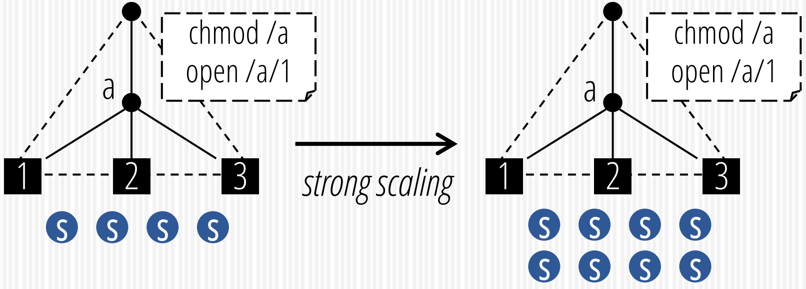
Throughput
op/s
Creation Phase
Uniform Stat's
Zipfian Stat's
20,000
16,000
12,000
8,000
4,000
0
🟩 IndexFS
🟥 ShardFS
metadata footprint
dir splitting
load variance
10,829
8,819
4,728
8,507
Balanced Tree
Zipfian Tree
🟩 IndexFS
🟥 ShardFS
14,654
15,502
path resolution
3,071
3,635
Balanced Tree
Zipfian Tree
🟩 IndexFS
🟥 ShardFS
hot spots on files
6,529
6,356
2,022
2,105
Balanced Tree
Zipfian Tree

Macro Benchmark

YCSB++ [SoCC11]

Strong scaling

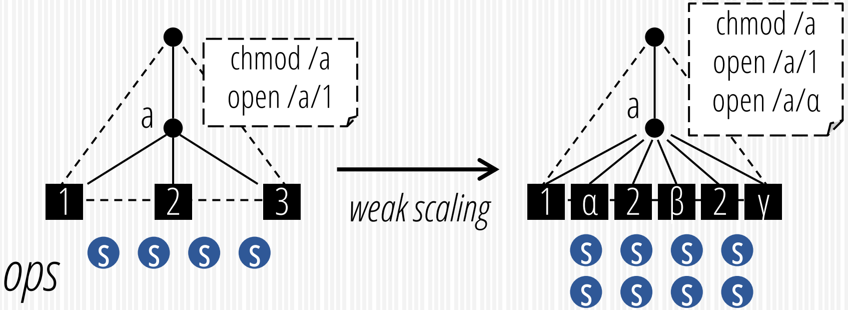
fixed namespace and fixed fs ops



Weak scaling

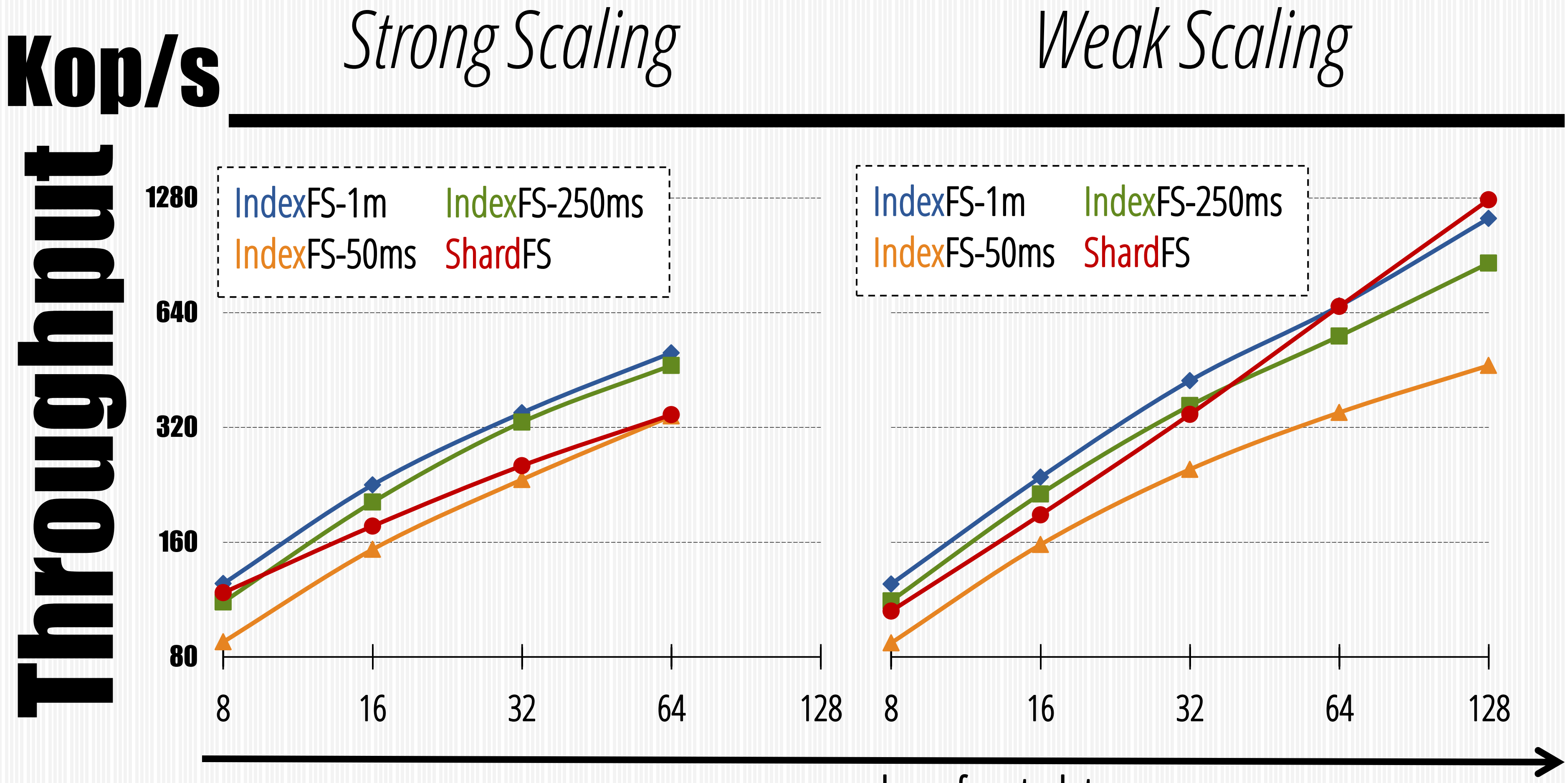
fixed dir structure and fixed dir lookup state mutation ops

with scaling number of files and scaling number of other fs ops



LinkedIn trace replay

1-day HDFS trace with 1.9M dirs & 11.4M files



Kop/s
Throughput
Strong Scaling
Weak Scaling
IndexFS-1m
IndexFS-250ms
IndexFS-50ms
ShardFS
IndexFS-1m
IndexFS-250ms
IndexFS-50ms
ShardFS
1280
640
320
160
80
8
16
32
64
128
8
16
32
64
128
number of metadata servers

References

IndexFS: Scaling File System Metadata Performance with Stateless Caching and Bulk Insertion. Kai Ren, Qing Zheng, Swapnil Patil and Garth Gibson. SC 2014

TableFS: Enhancing metadata efficiency in local file systems. Kai Ren and Garth Gibson. USENIX ATC 2013

Scale and Concurrency in GIGA+: File System Directories with Millions of Files. Swapnil Patil and Garth Gibson. FAST 2009

YCSB++: Benchmarking and Performance Debugging Advanced Features in Scalable Table Stores. Swapnil Patil, Milo Polte, Kai Ren, Wittawat Tantisiriroj, Lin Xiao, Julio Lopez, Garth Gibson, Adam Fuchs, Billie Rinaldi. SoCC 2011

BACKUP SLIDES

Target Namespace

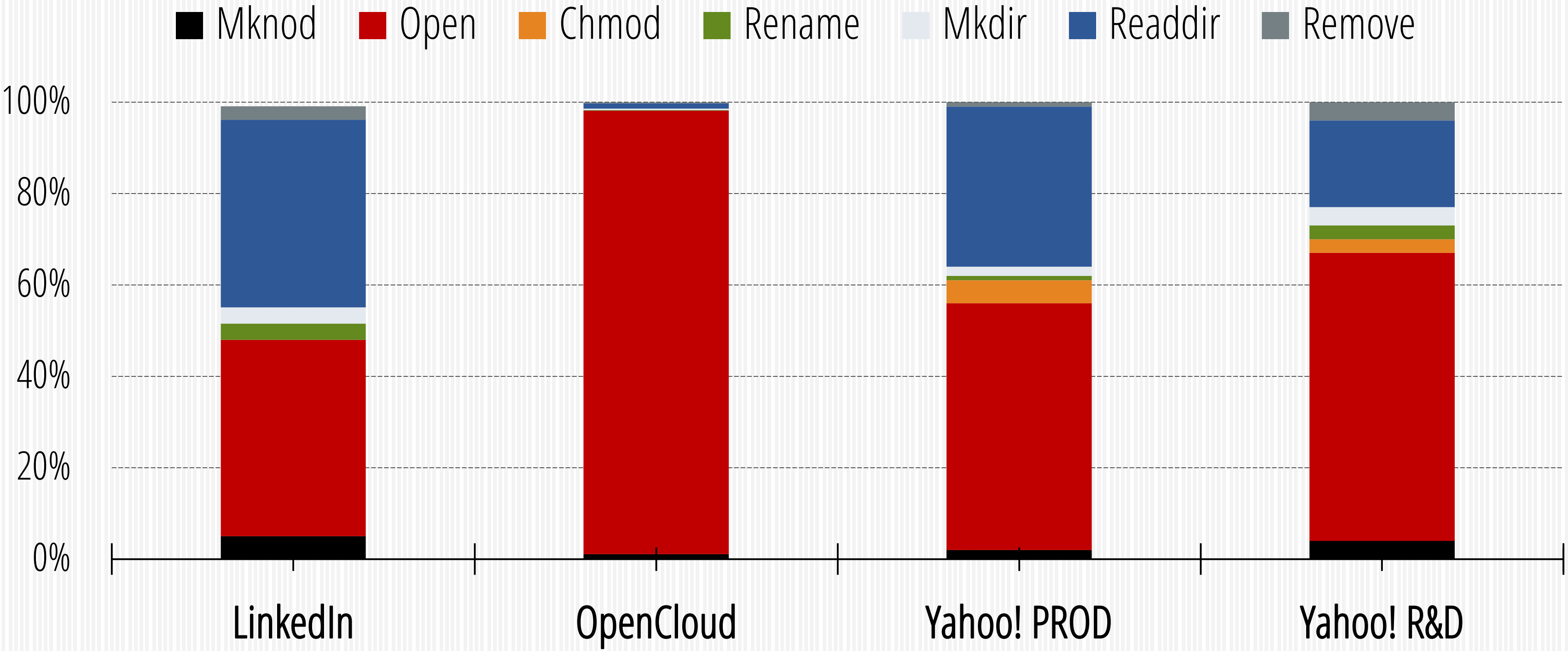
90% dirs are small (less than 128 entries)

Large dirs are really huge

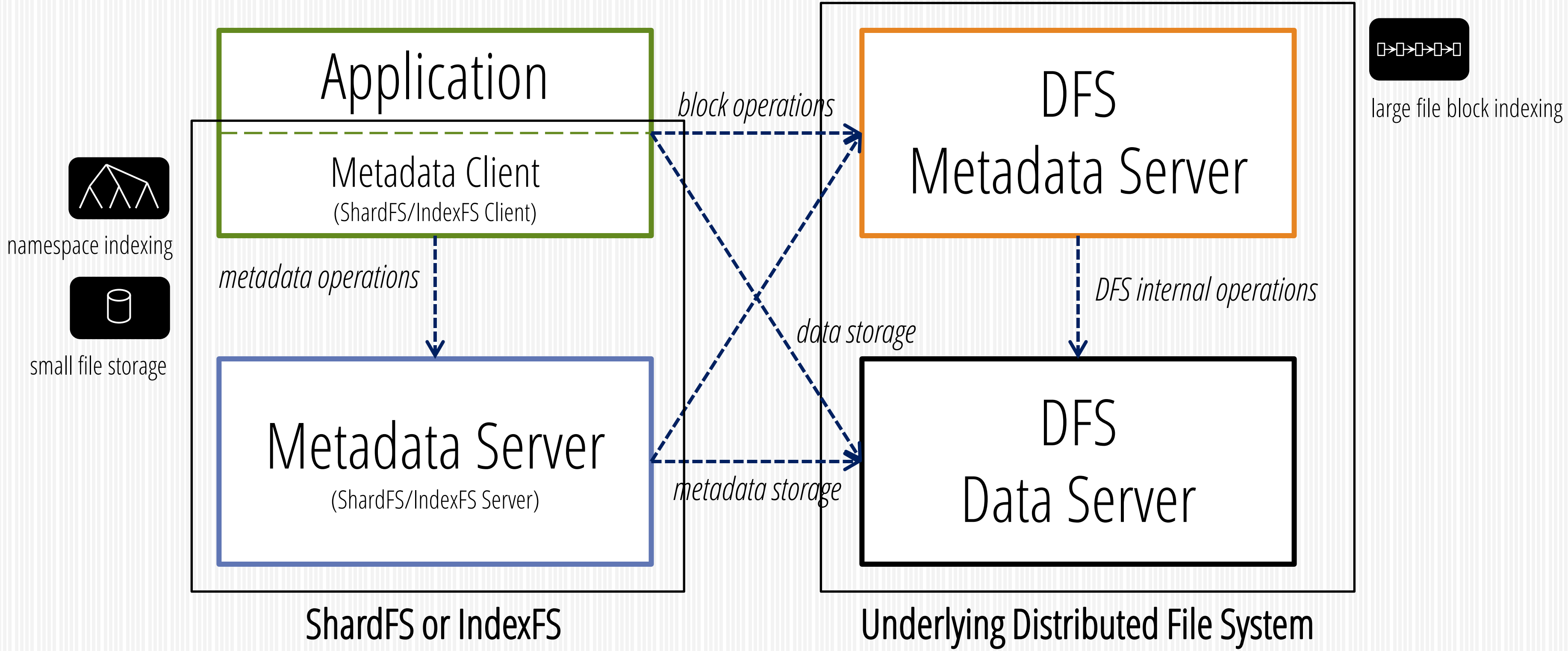
90% dirs are of depth 16 or more

Median file size smaller then 64KB for many fs'es

Distribution of FS Ops



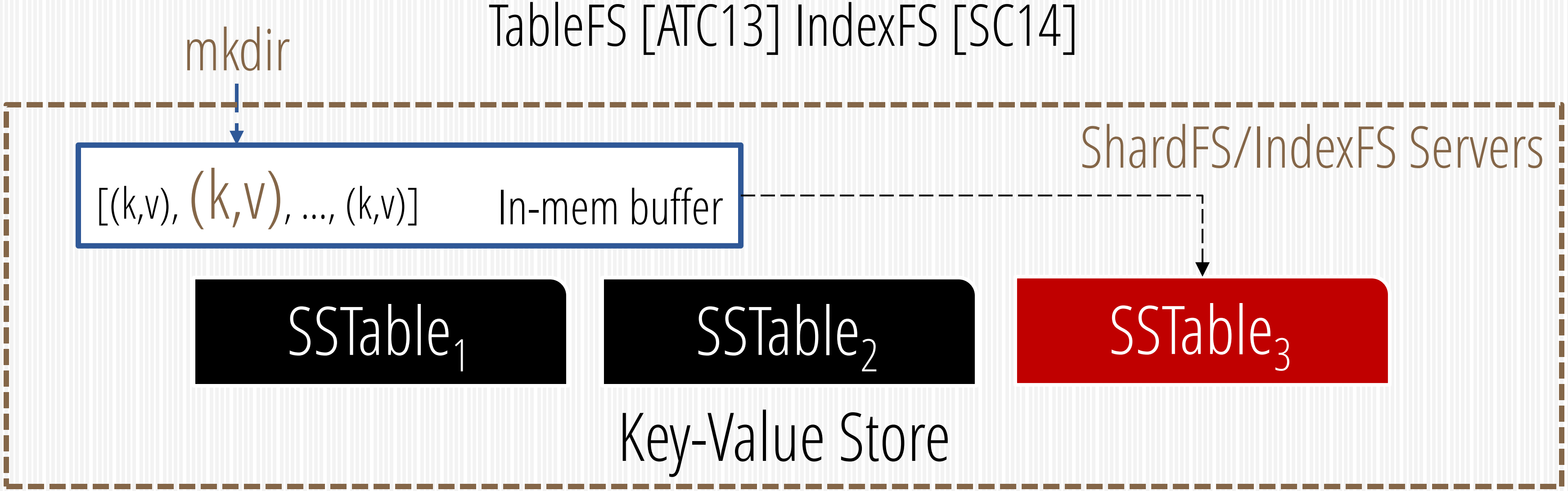
ShardFS/IndexFS Overview



Metadata Representation

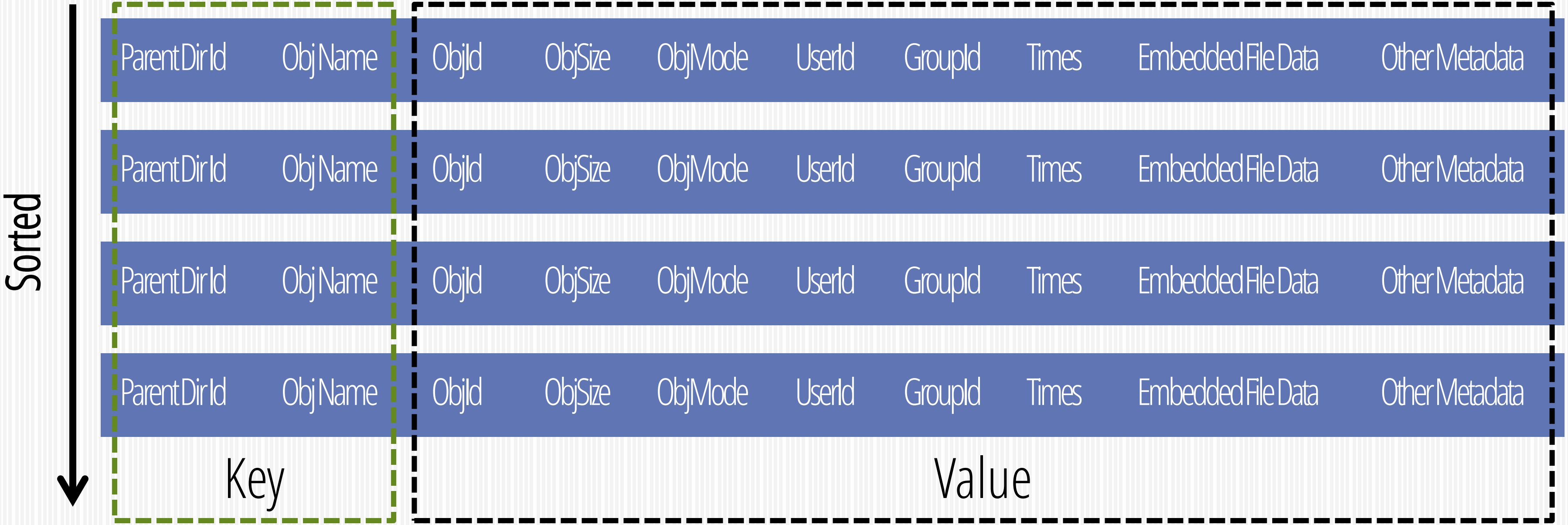
Log-structured and indexed data structure

TableFS [ATC13] IndexFS [SC14]



Namespace Metadata

= *dir index + object attributes + file data for small files*



ParentDirId	ObjName	ObjId	ObjSize	ObjMode	UserId	GroupId	Times	Embedded File Data	Other Metadata
ParentDirId	ObjName	ObjId	ObjSize	ObjMode	UserId	GroupId	Times	Embedded File Data	Other Metadata
ParentDirId	ObjName	ObjId	ObjSize	ObjMode	UserId	GroupId	Times	Embedded File Data	Other Metadata
ParentDirId	ObjName	ObjId	ObjSize	ObjMode	UserId	GroupId	Times	Embedded File Data	Other Metadata