HopsFS: Scaling Hierarchical File System Metadata Using NewSQL Databases

Salman Niazi[1], Mahmoud Ismail[1],
Seif Haridi[1], Jim Dowling[1], Steffen Grohsschmiedt[2], Mikael Ronström[3]
1. KTH - Royal Institute of Technology, **2.** Spotify AB, **3.** Oracle

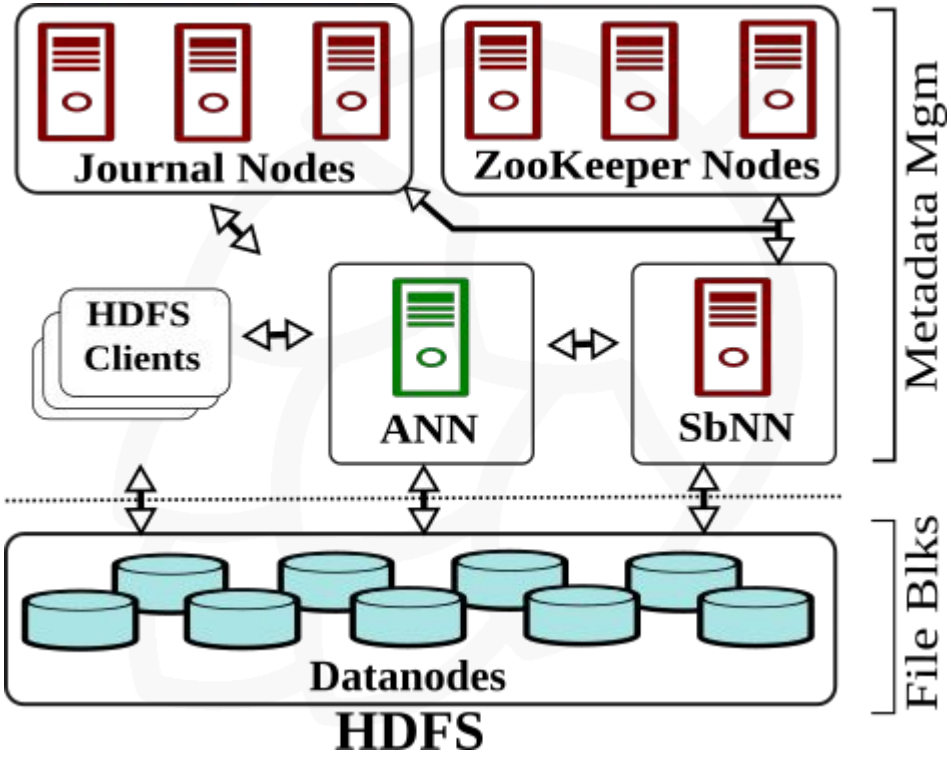
HopsFS

>1 million ops/sec

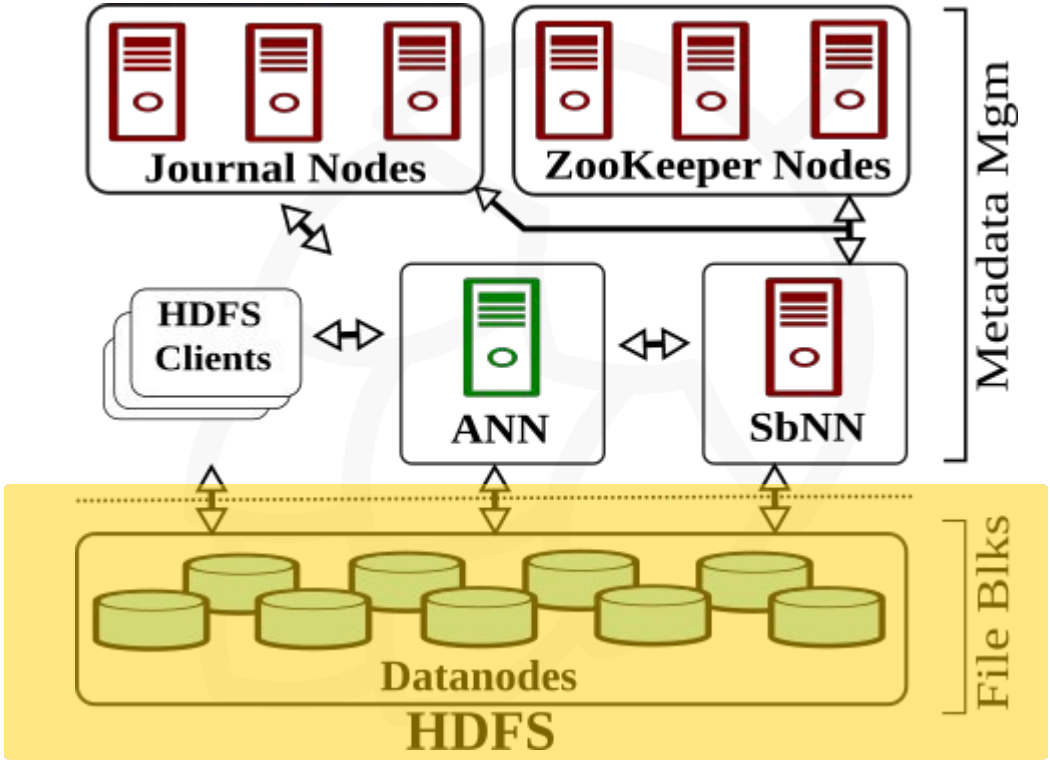
Hadoop compatible distributed hierarchical file system that stores billions of files.

Motivation

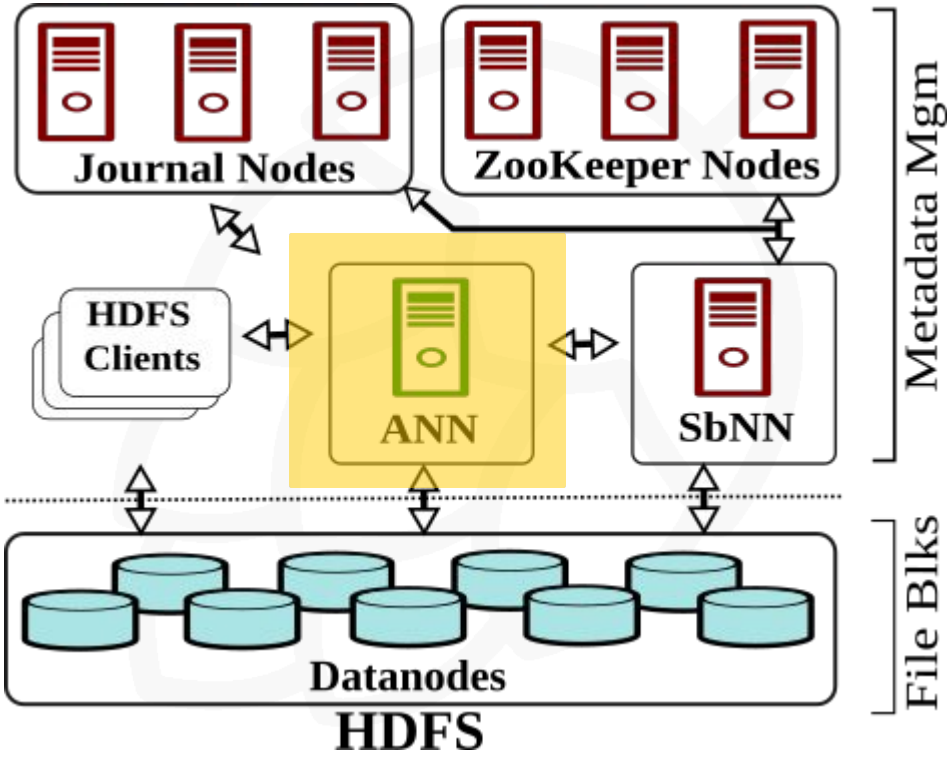
Hadoop Distributed File System



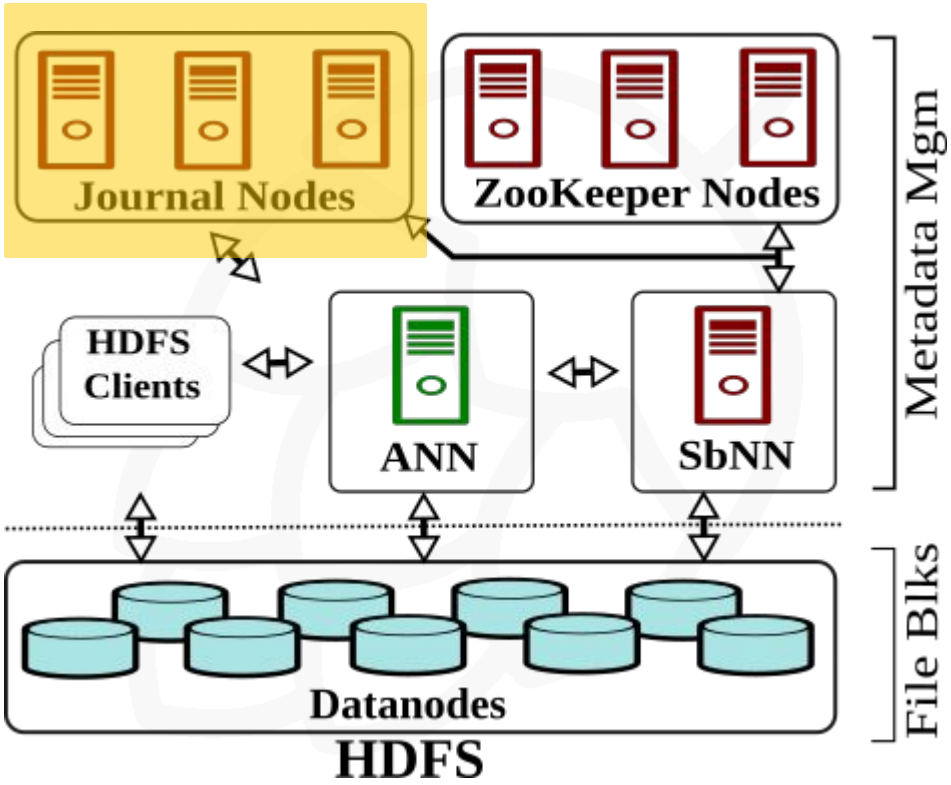
Hadoop Distributed File System



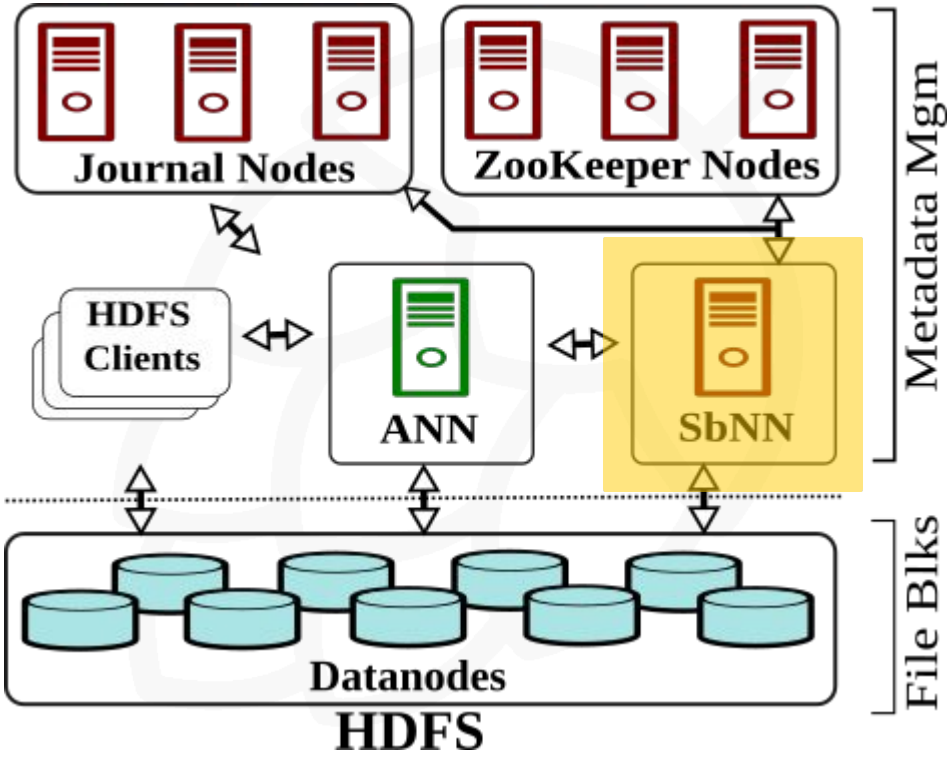
Hadoop Distributed File System



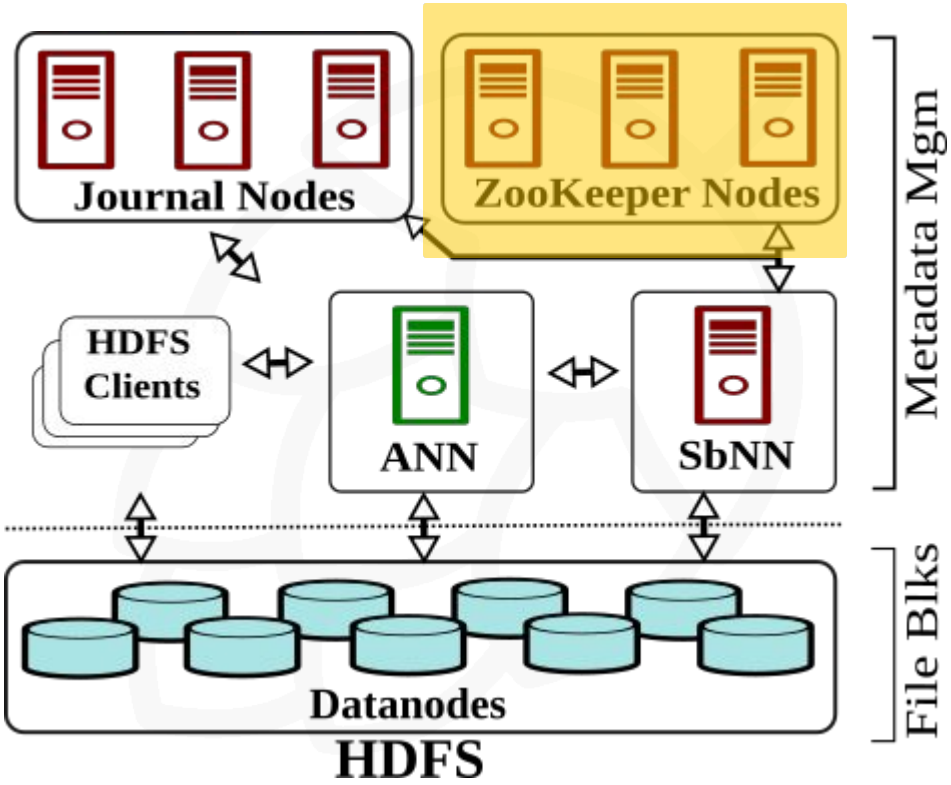
Hadoop Distributed File System



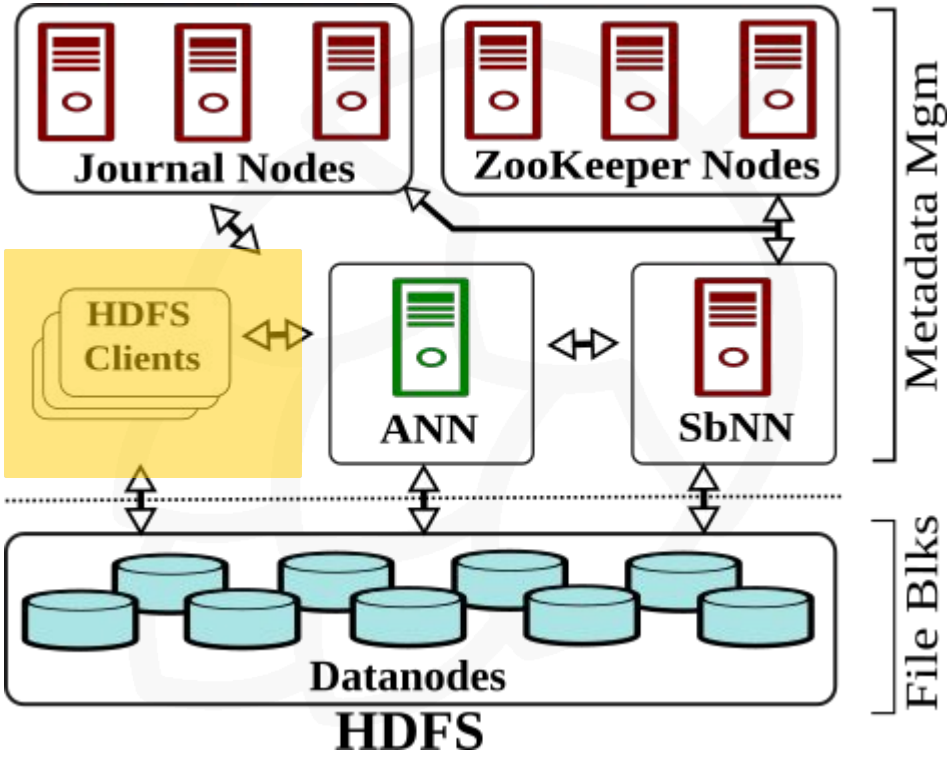
Hadoop Distributed File System



Hadoop Distributed File System



Hadoop Distributed File System



HDFS’ Namenode Limitations

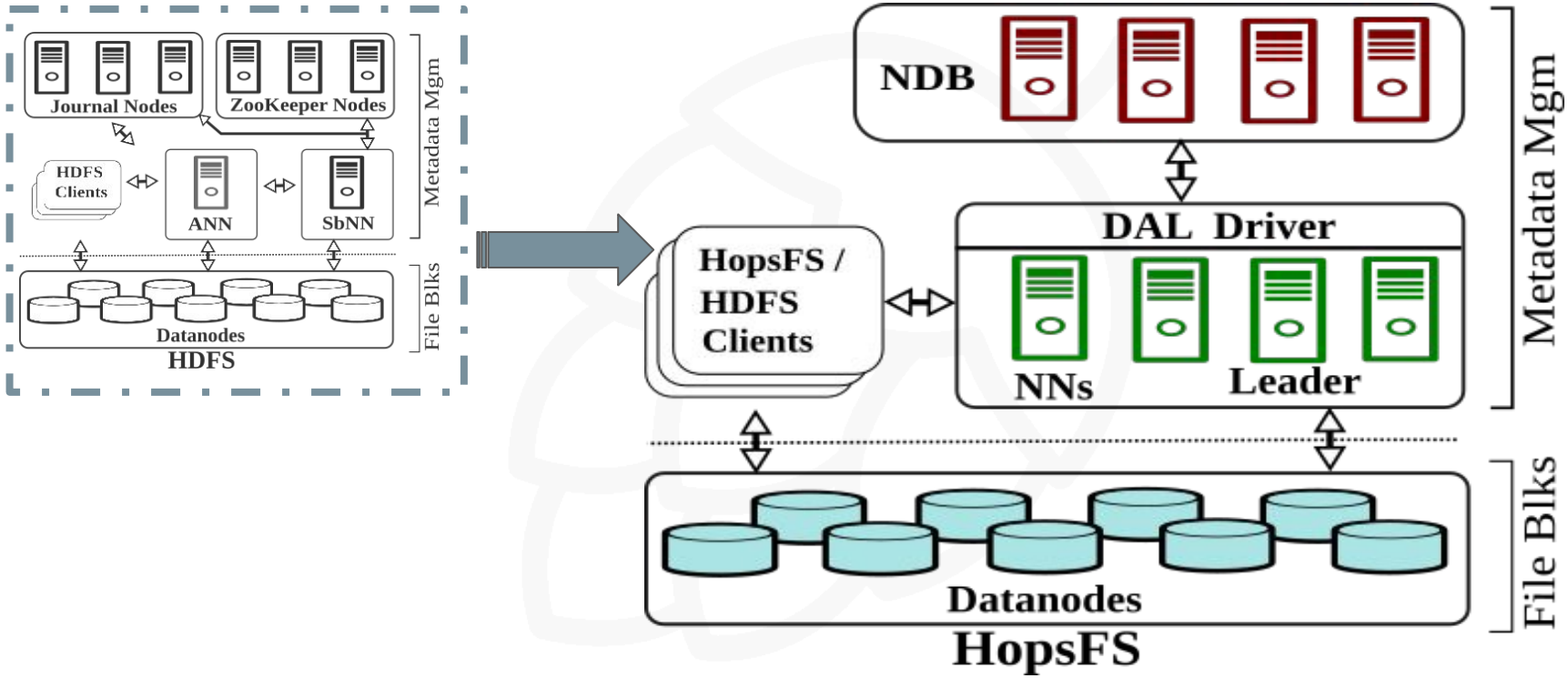
- Limited namespace
  - Namenode stores metadata on a JVM Heap
- Limited concurrency
  - Strong consistency for metadata operations are guaranteed using a single global lock (single-writer, multiple readers)

Solution

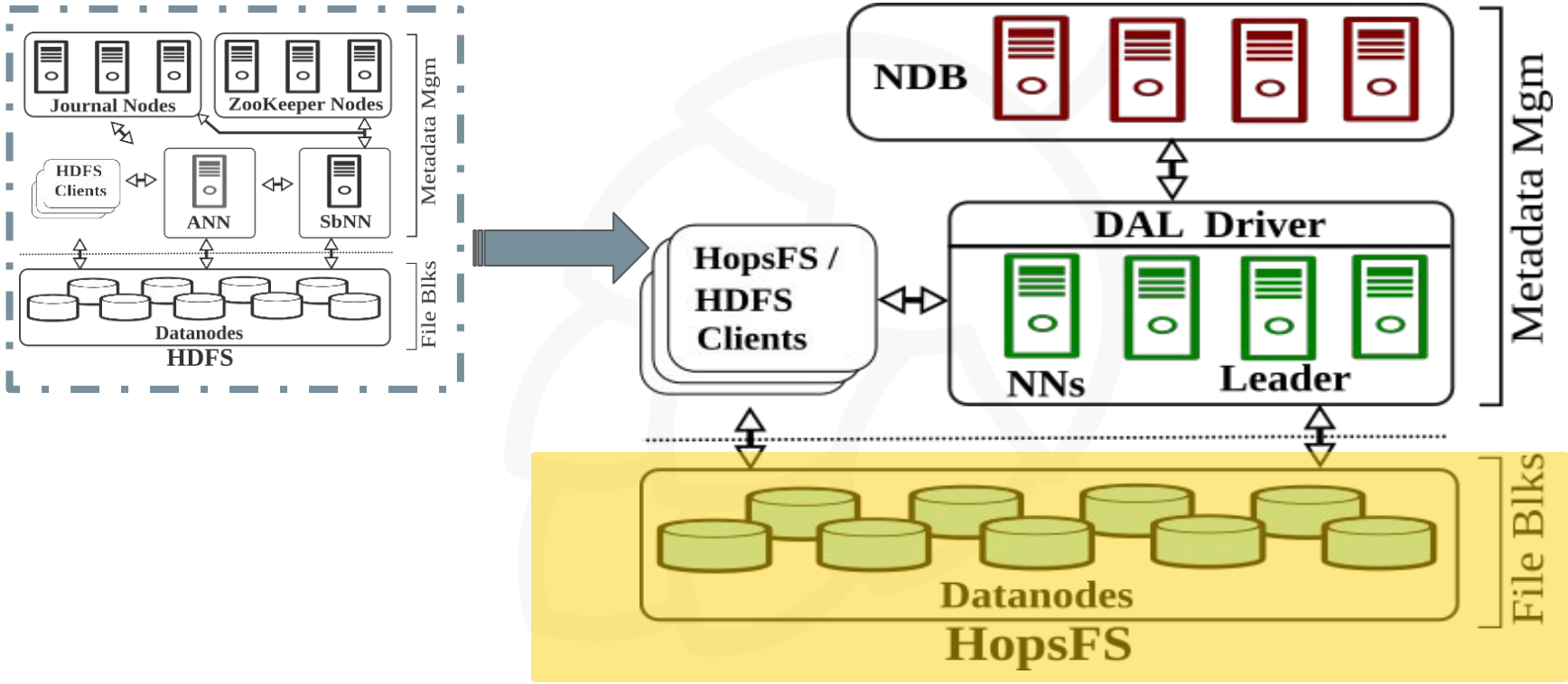
Solution

- Increased size of the namespace
  - Move the Namenode metadata off the JVM Heap and store the metadata in an in-memory distributed database
  - Multiple stateless Namenodes access and update the metadata in parallel
- Improved concurrency model and locking mechanisms to support multiple concurrent read and write operations (multiple-writers, multiple-readers)

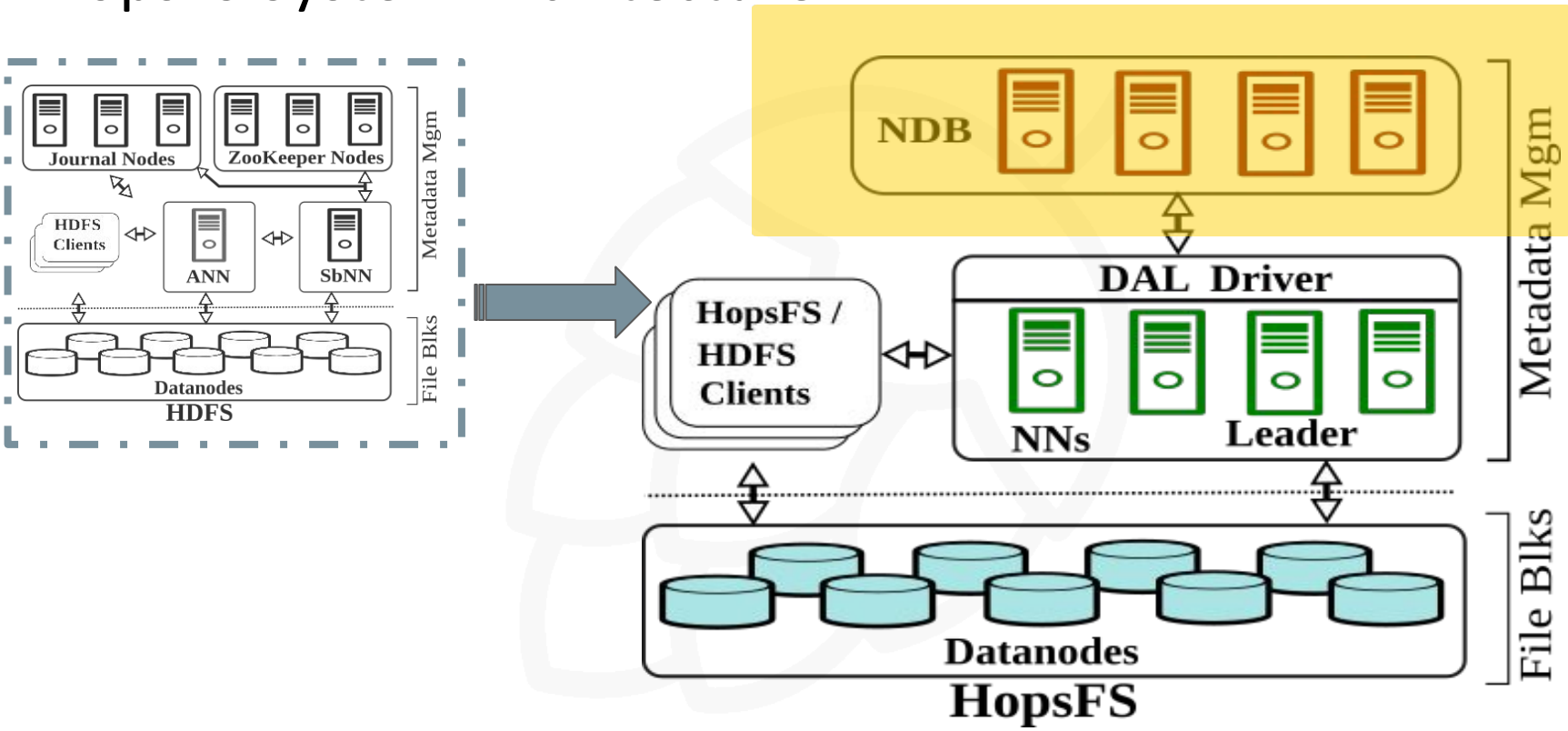
HopsFS System Architecture



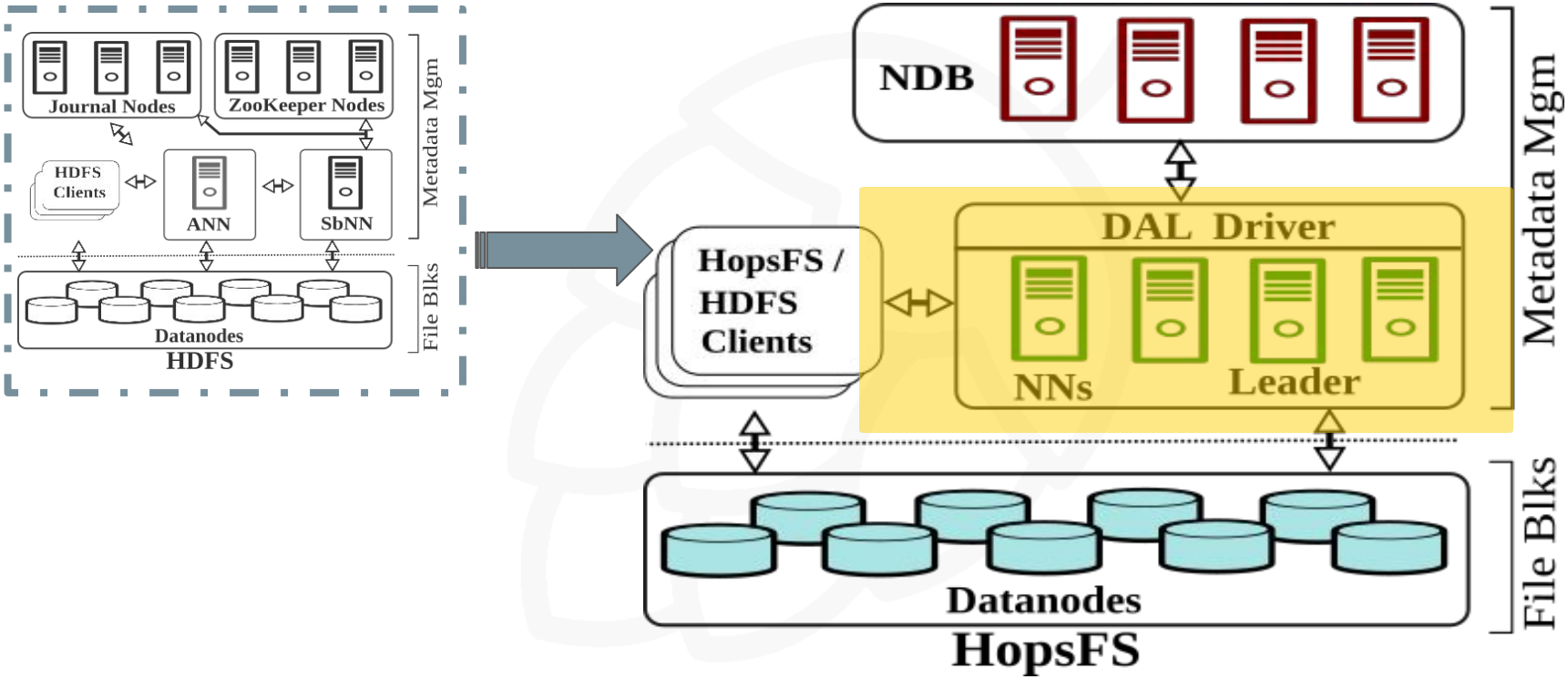
HopsFS System Architecture



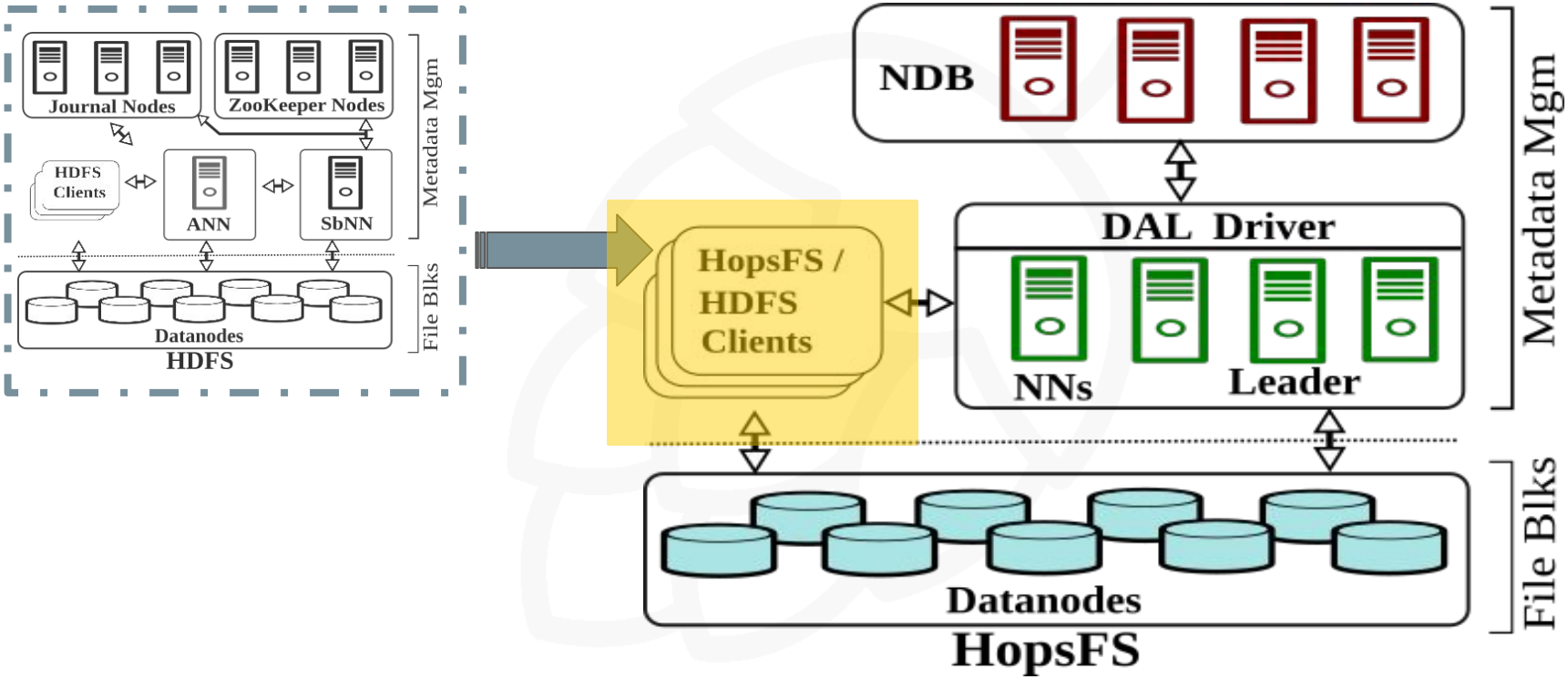
HopsFS System Architecture



HopsFS System Architecture



HopsFS System Architecture



NewSQL DB

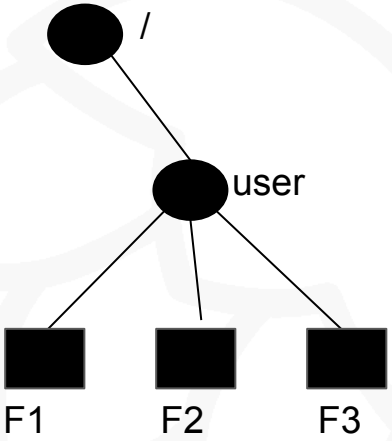
MySQL Cluster: **Network Database Engine (NDB)**

- **Commodity Hardware**
  - Scales to 48 database nodes
    - 200 Million NoSQL Read Ops/Sec*
- **NewSQL (Relational) DB**
  - Read Committed Transaction Isolation
  - Row-level Locking
  - User-defined partitioning

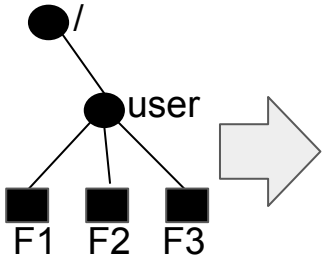
*https://www.mysql.com/why-mysql/benchmarks/mysql-cluster/

HopsFS Metadata
&
Metadata Partitioning

HopsFS Metadata and Metadata Partitioning



HopsFS Metadata & Metadata Partitioning (contd.)

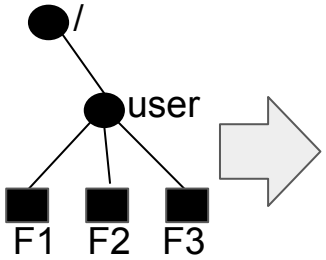


INode Table

Inode_ID	Name	Parent_ID	...

- Inode ID
- Parent INode ID
- Name
- Size
- Access Attributes
- ...

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

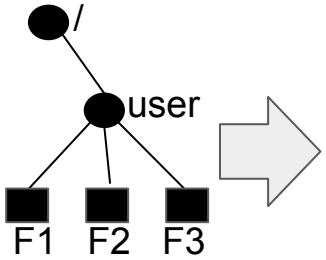
Inode_ID	Name	Parent_ID	...

Block Table

Block_ID	Inode_ID	...

- File INode to Blocks Mapping
- Block Size
- ...

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...

Block Table

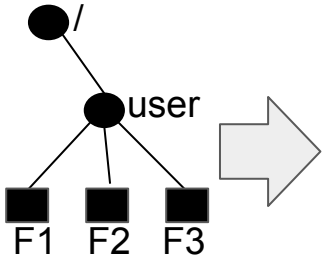
Block_ID	Inode_ID	...

Replica Table

Inode_ID	Block_ID	DataNode_ID	...

- Location of blocks on Datanodes
- ...

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...

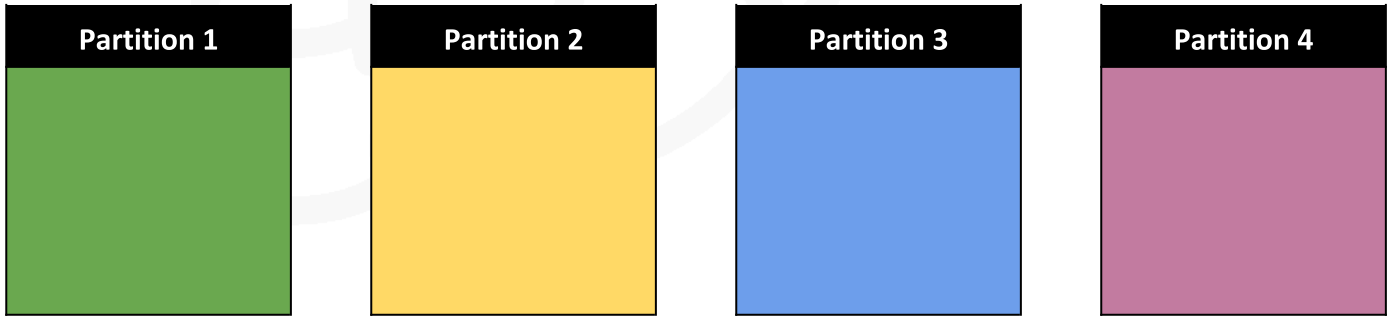
Block Table

Block_ID	Inode_ID	...

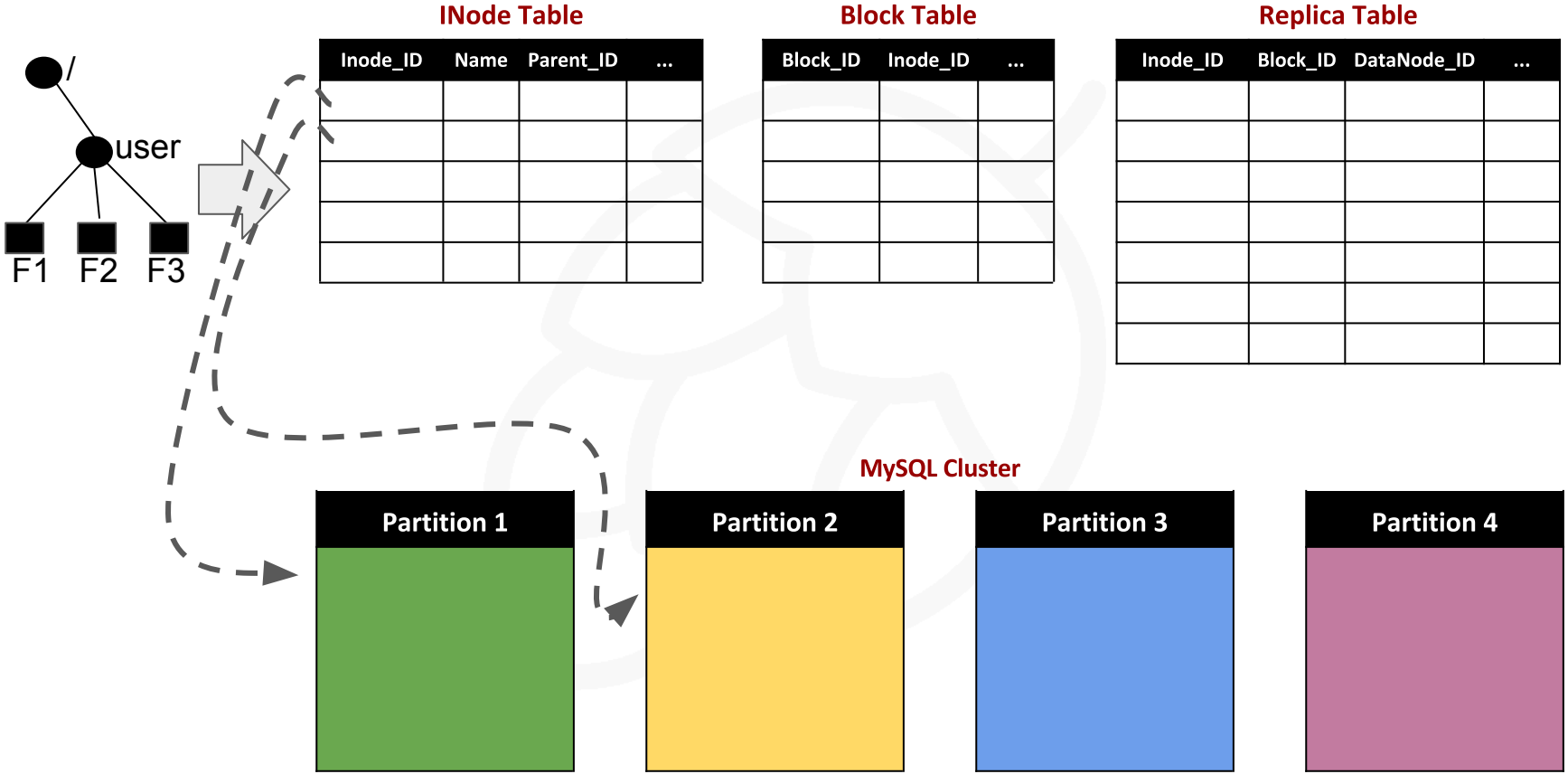
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

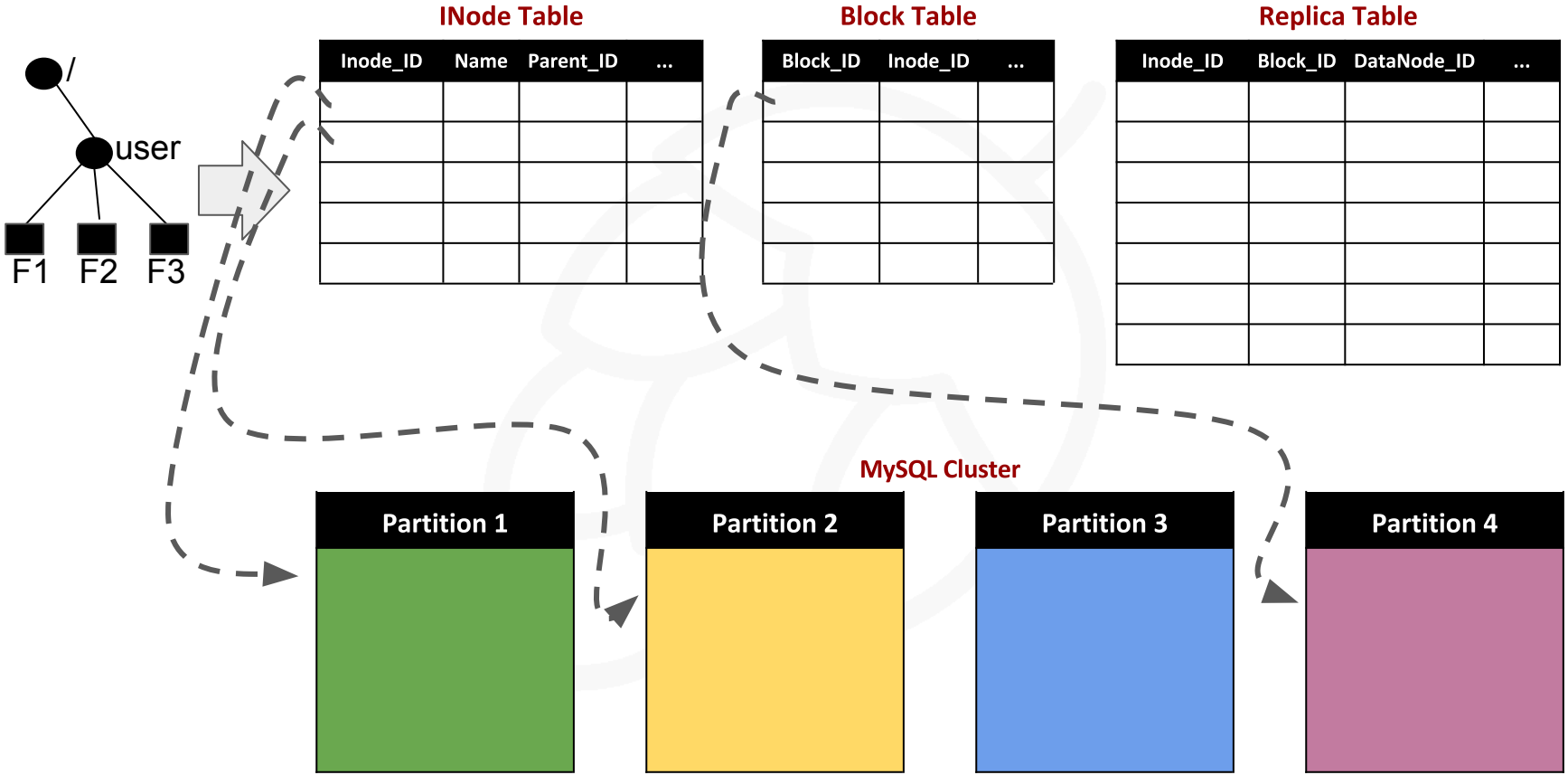
MySQL Cluster



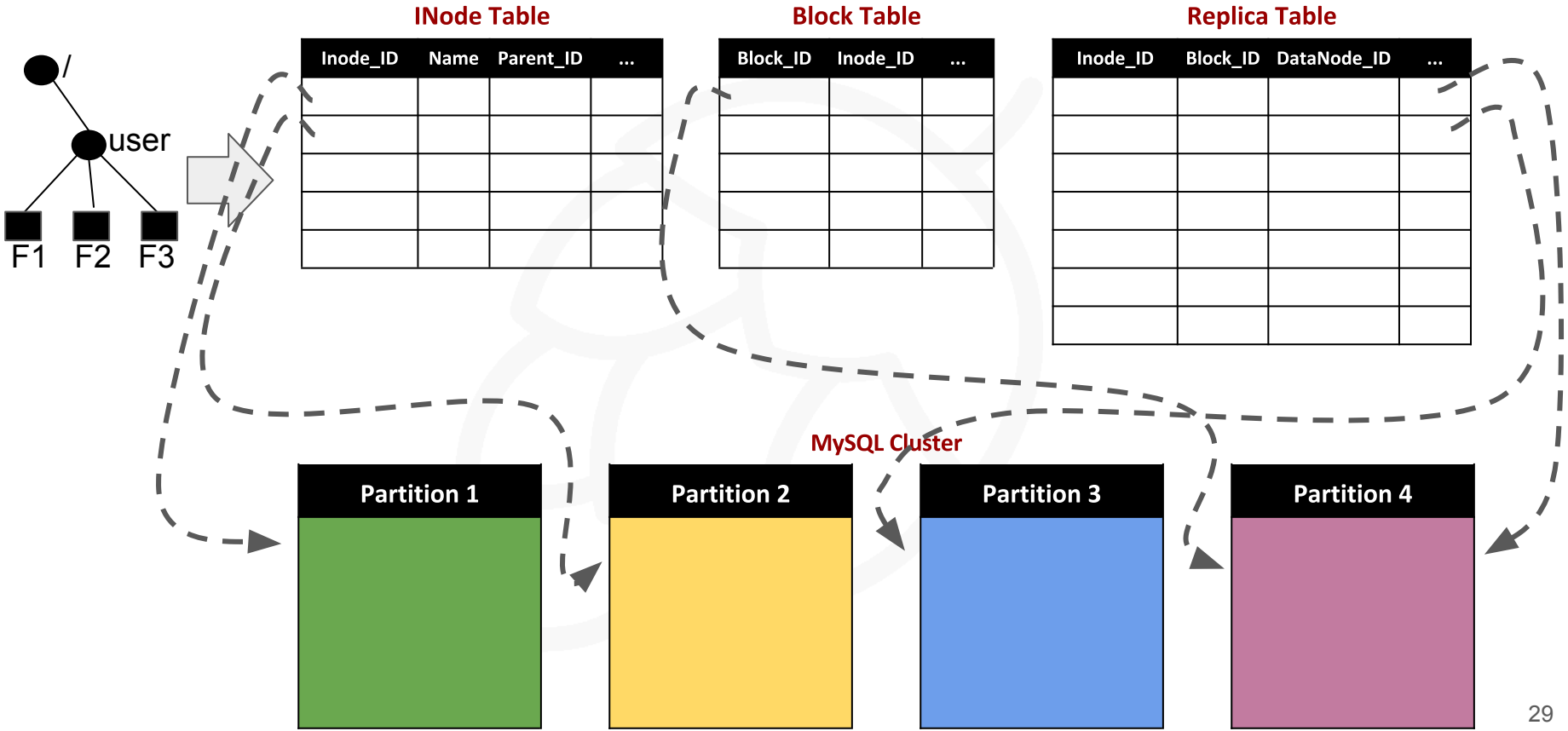
HopsFS Metadata & Metadata Partitioning (contd.)



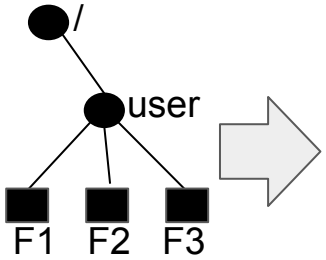
HopsFS Metadata & Metadata Partitioning (contd.)



HopsFS Metadata & Metadata Partitioning (contd.)



HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...

Block Table

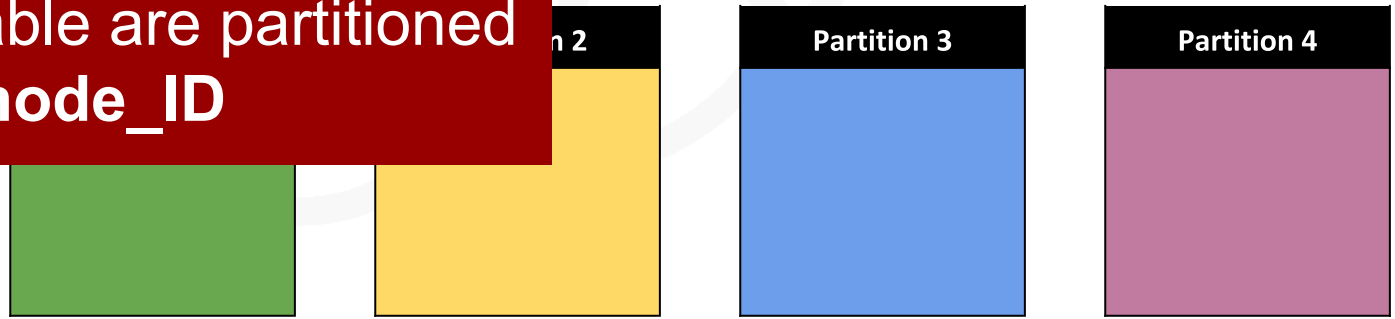
Block_ID	Inode_ID	...

Replica Table

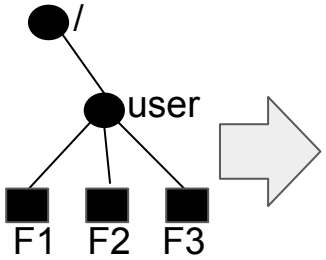
Inode_ID	Block_ID	DataNode_ID	...

All the tables except the INode table are partitioned by the **Inode_ID**

MySQL Cluster



HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...

Block Table

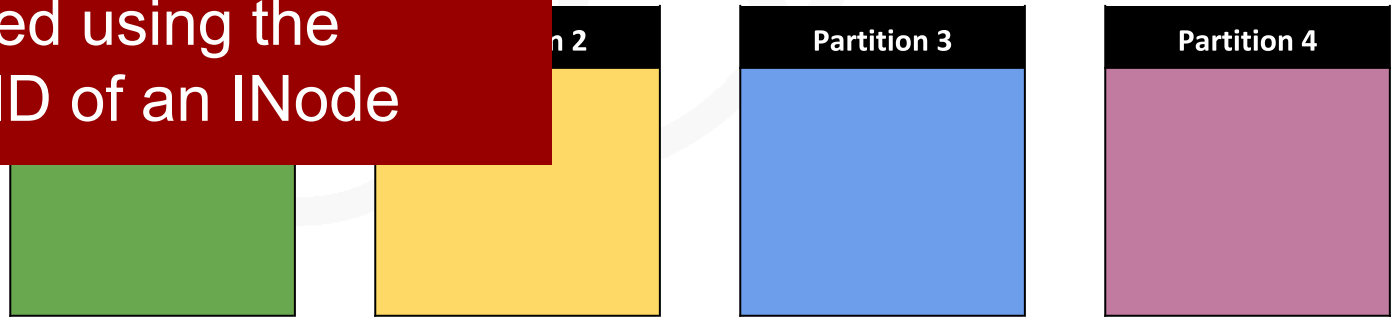
Block_ID	Inode_ID	...

Replica Table

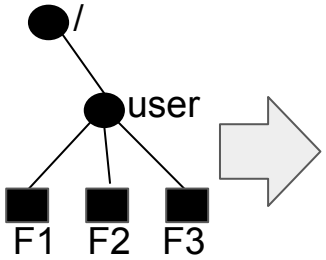
Inode_ID	Block_ID	DataNode_ID	...

The INode table is partitioned using the Parent_ID of an INode

MySQL Cluster



HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...

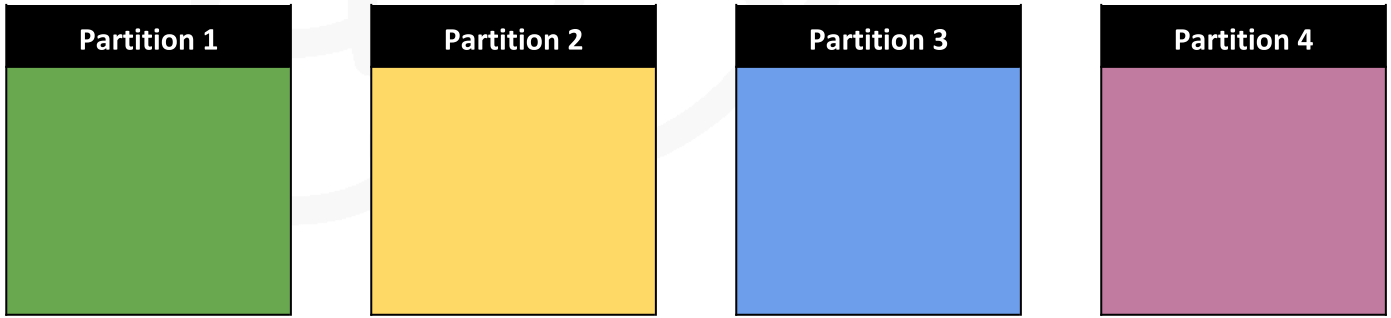
Block Table

Block_ID	Inode_ID	...

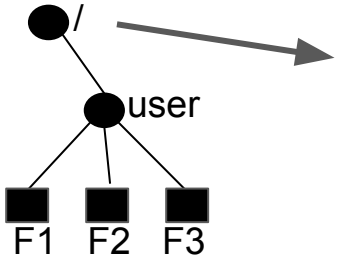
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster



HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	

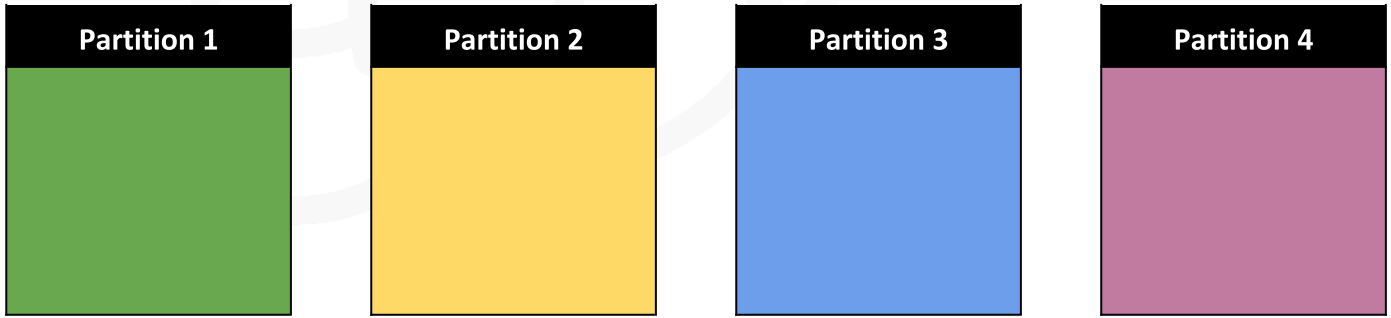
Block Table

Block_ID	Inode_ID	...

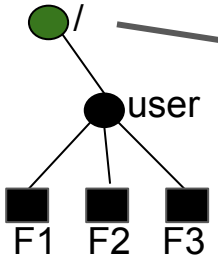
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster



HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	

Block Table

Block_ID	Inode_ID	...

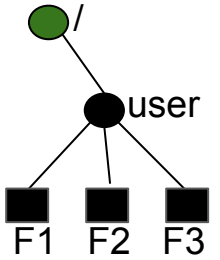
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/			

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	

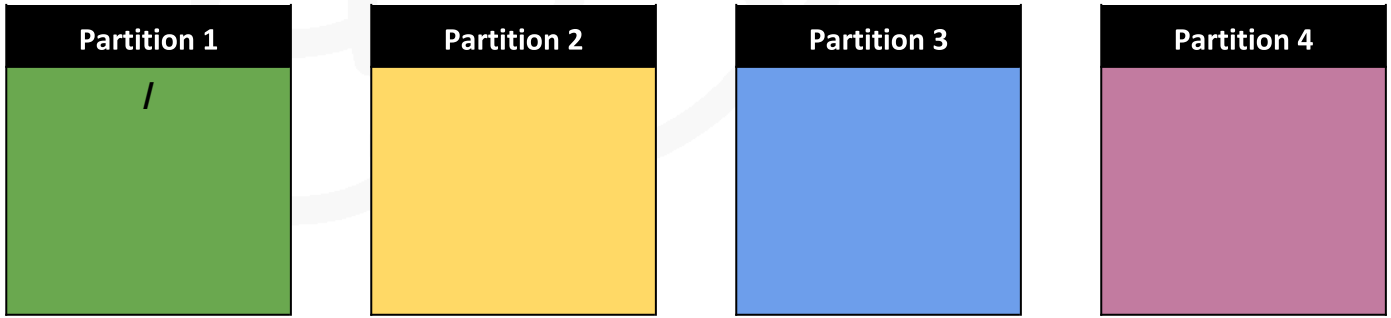
Block Table

Block_ID	Inode_ID	...

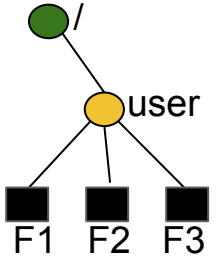
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster



HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	

Block Table

Block_ID	Inode_ID	...

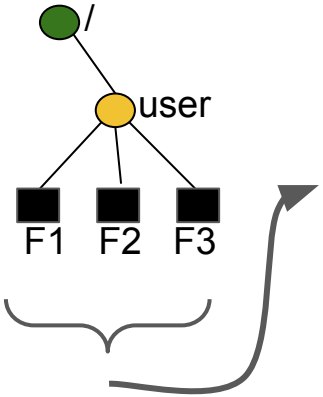
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user		

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Block_ID	Inode_ID	...

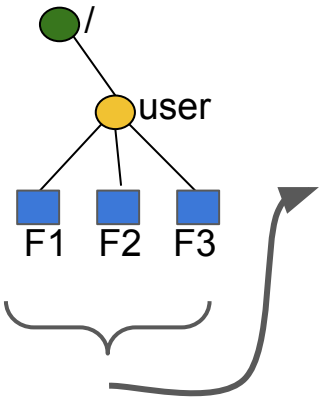
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user		

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Block_ID	Inode_ID	...

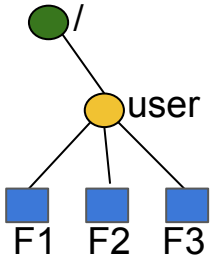
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Inode_ID	Block_ID	...
3	1	
3	2	
3	3	

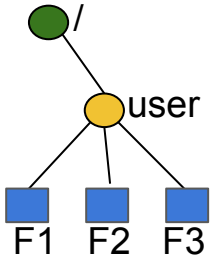
Replica Table

Inode_ID	Block_ID	DataNode_ID	...

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Inode_ID	Block_ID	...
3	1	
3	2	
3	3	

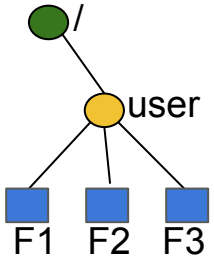
Replica Table

Inode_ID	Block_ID	DataNode_ID	...
3	1	1	
3	1	2	
3	1	3	
3	2	4	
3	2	5	
3	...	...	

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Inode_ID	Block_ID	...
3	1	
3	2	
3	3	

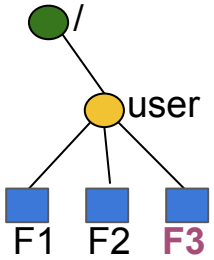
Replica Table

Inode_ID	Block_ID	DataNode_ID	...
3	1	1	
3	1	2	
3	1	3	
3	2	4	
3	2	5	
3	...	...	

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Inode_ID	Block_ID	...
3	1	
3	2	
3	3	

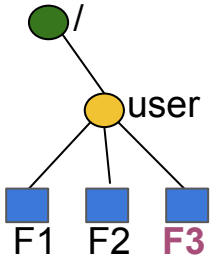
Replica Table

Inode_ID	Block_ID	DataNode_ID	...
3	1	1	
3	1	2	
3	1	3	
3	2	4	
3	2	5	
3	...	...	

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	[{3,1},{3,2},{3,3}],[{3,1,1},{3,1,2},{3,1,3},{3,2,4}…{3,3,9}]

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Inode_ID	Block_ID	...
3	1	
3	2	
3	3	

Replica Table

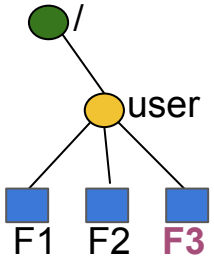
Inode_ID	Block_ID	DataNode_ID	...
3	1	1	
3	1	2	
3	1	3	
3	2	4	
3	2	5	
3	...	...	

```
$> ls  /user/*
```

MySQL Cluster

Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	[{3,1},{3,2},{3,3}],[{3,1,1},{3,1,2},{3,1,3},{3,2,4}…{3,3,9}]

HopsFS Metadata & Metadata Partitioning (contd.)



INode Table

Inode_ID	Name	Parent_ID	...
1	/	0	
2	user	1	
3	F1	2	
4	F2	2	
5	F3	2	

Block Table

Inode_ID	Block_ID	...
3	1	
3	2	
3	3	

Replica Table

Inode_ID	Block_ID	DataNode_ID	...
3	1	1	
3	1	2	
3	1	3	
3	2	4	
3	2	5	
3	...	...	

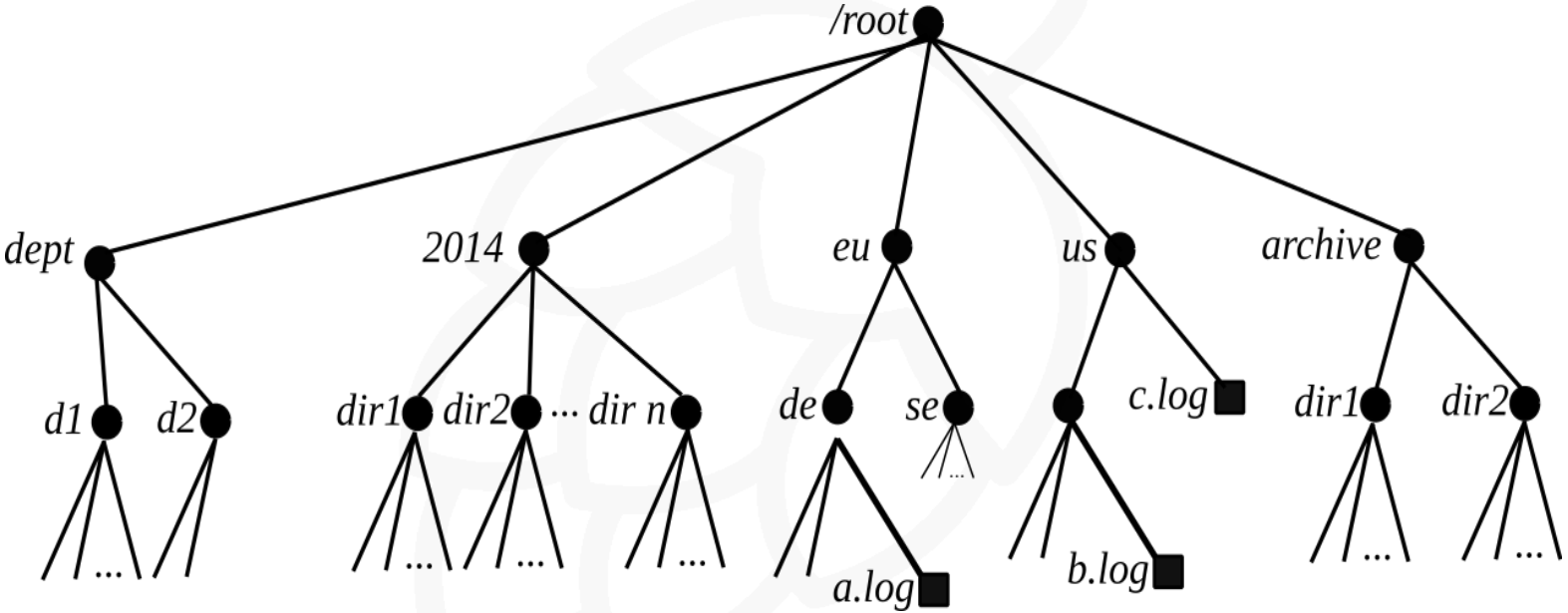
```
$> cat  /user/F1
```

MySQL Cluster

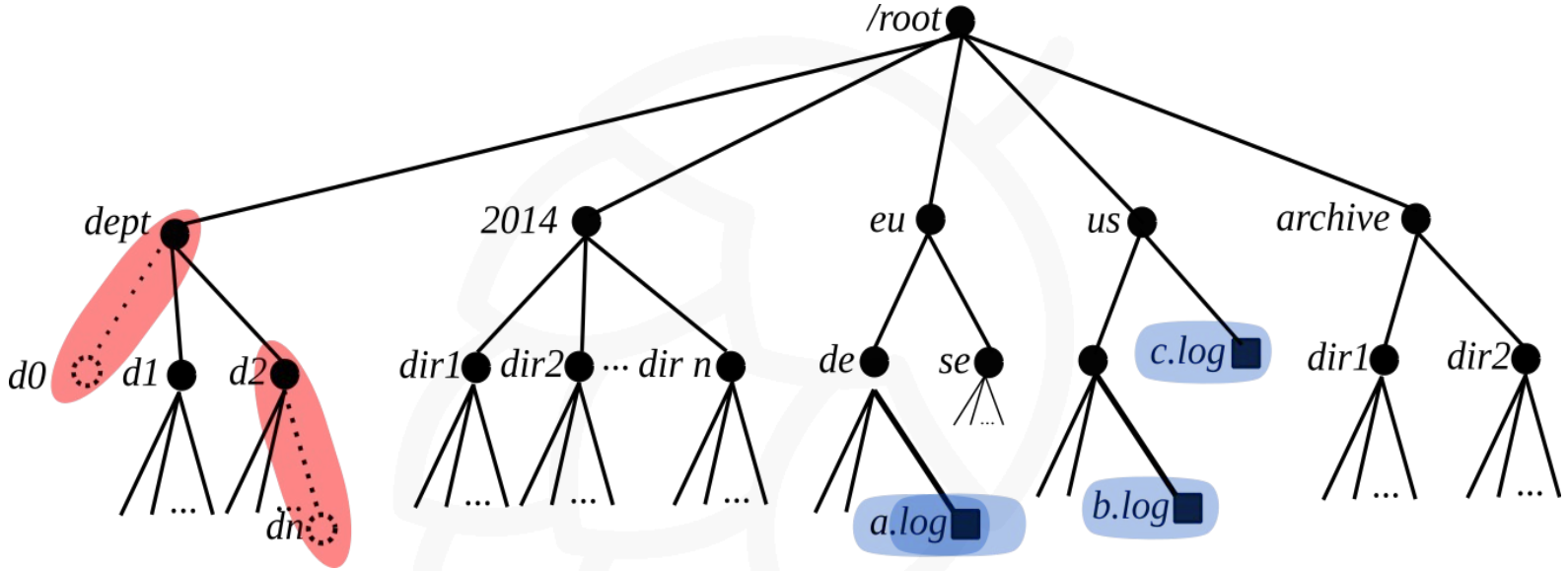
Partition 1	Partition 2	Partition 3	Partition 4
/	user	F1 F2 F3	[{3,1},{3,2},{3,3}],[{3,1,1},{3,1,2},{3,1,3},{3,2,4}…{3,3,9}]

Fine Grain Locking & Transactional Operations

Metadata Locking

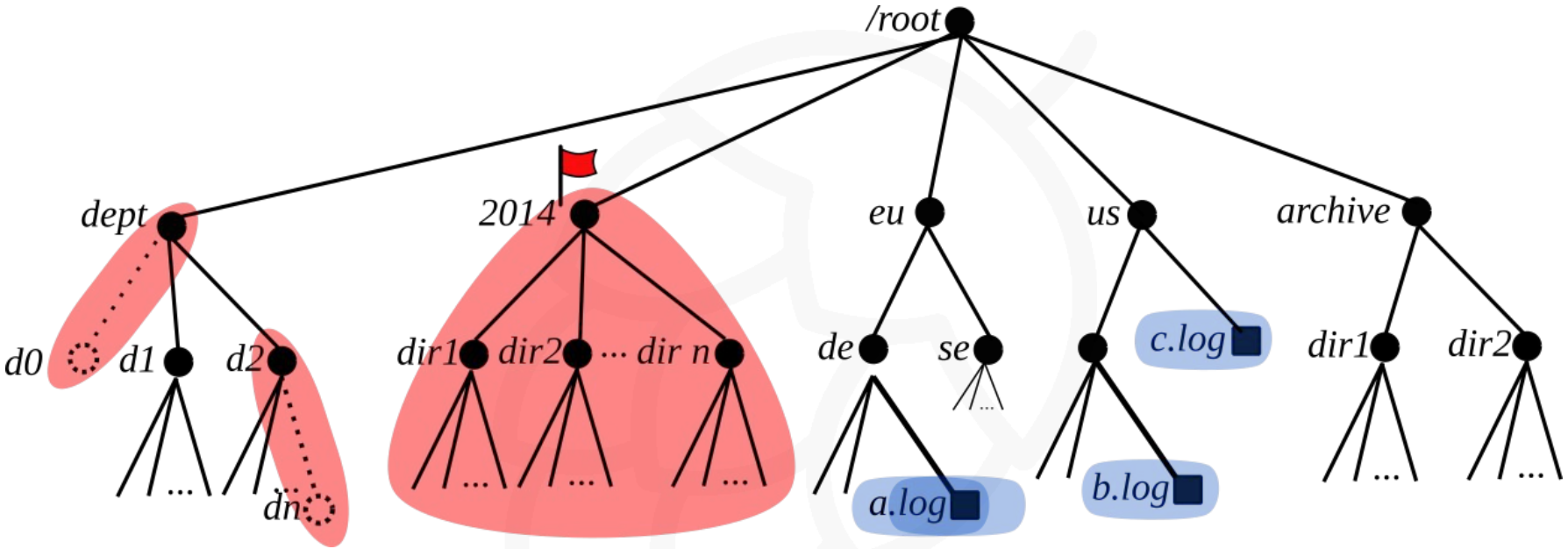


Metadata Locking (contd.)



- **Exclusive Lock**
- **Shared Lock**

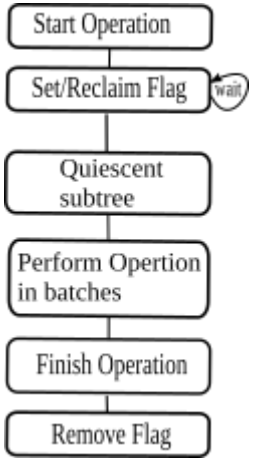
Metadata Locking (contd.)

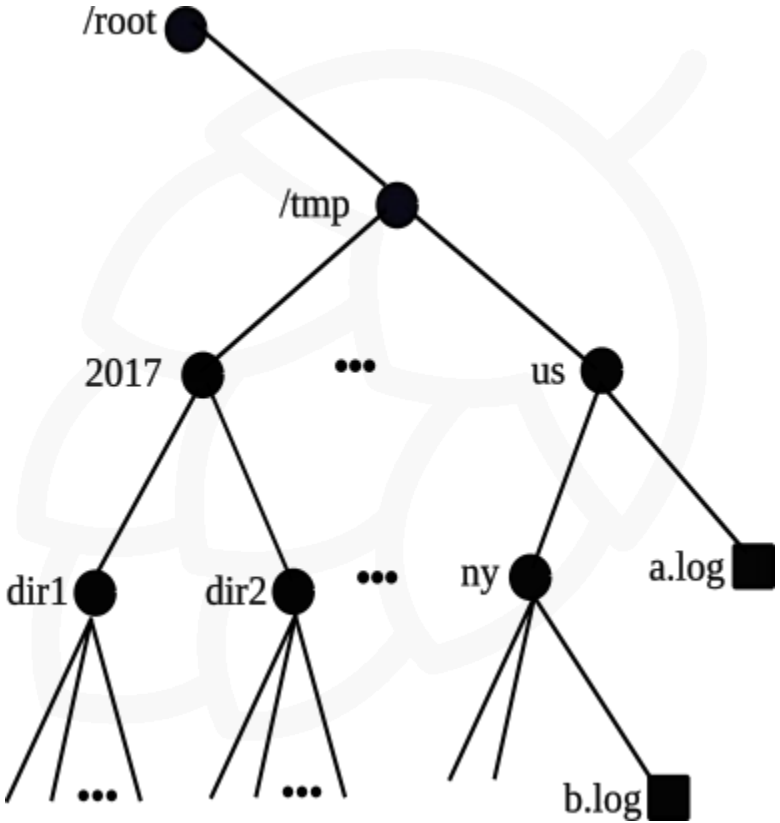


- **Exclusive Lock**
- **Shared Lock**
- **Subtree Lock**

Subtree Locking

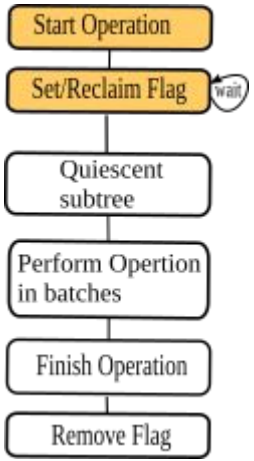
Subtree operation Stages

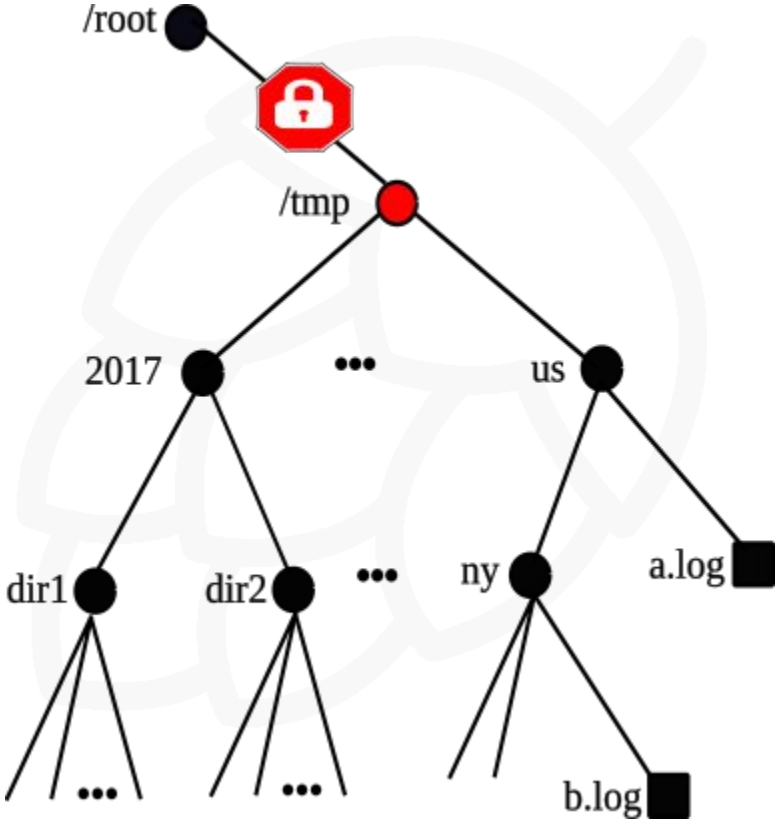




Subtree Locking (contd.)

Subtree operation Stages

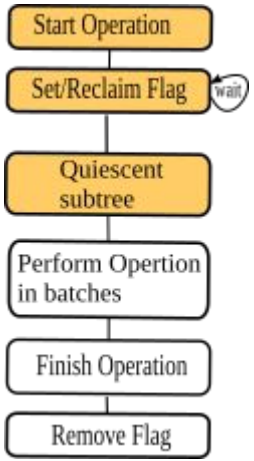


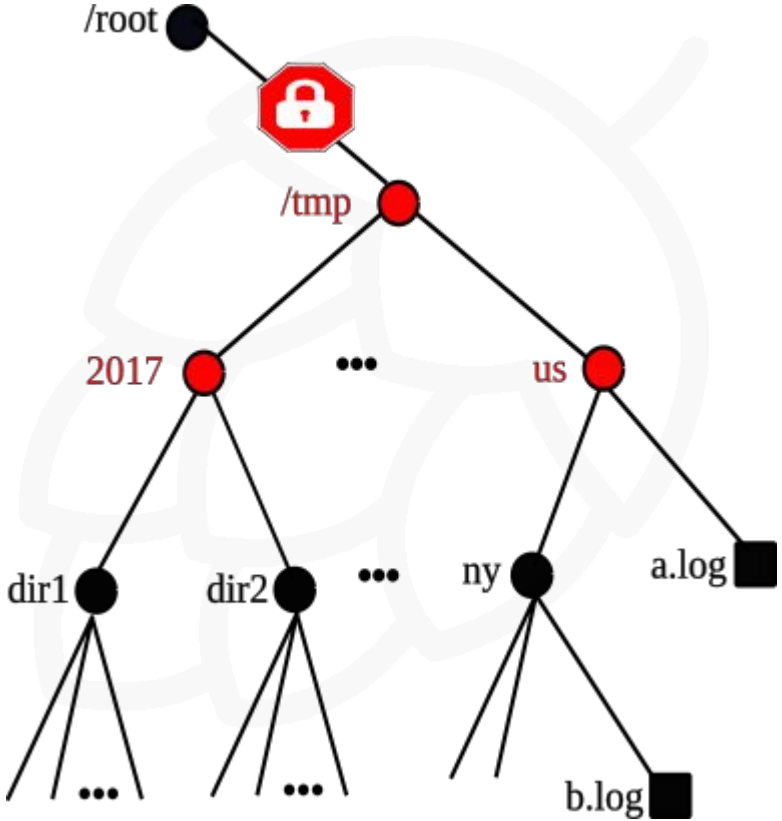


Subtree Locking Progresses Downwards

Subtree Locking (contd.)

Subtree operation Stages

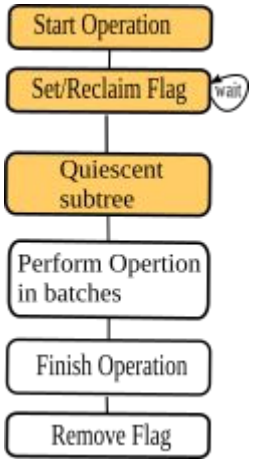


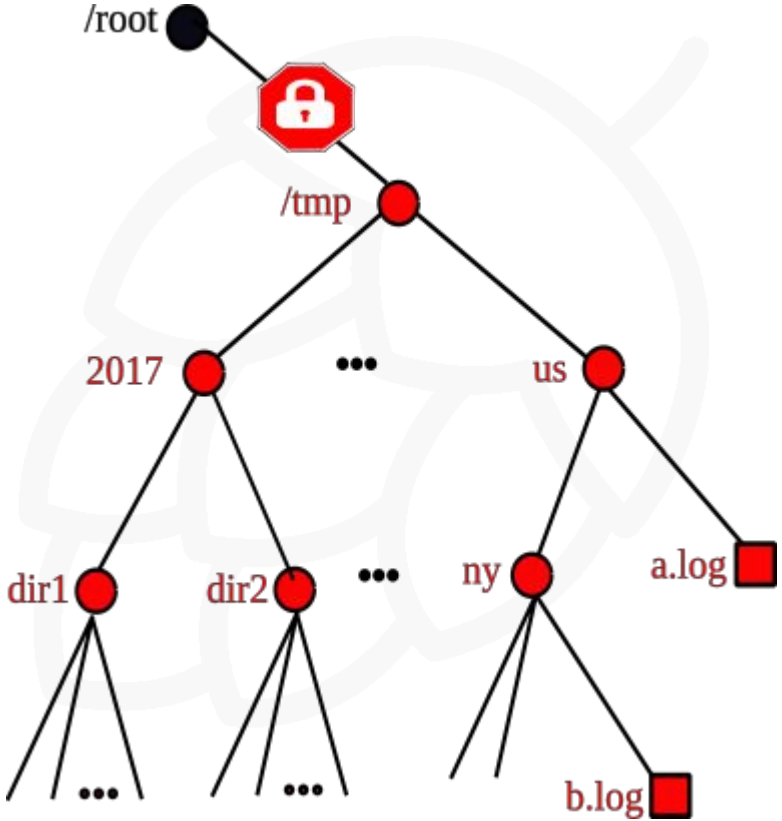


Subtree Locking Progresses Downwards

Subtree Locking (contd.)

Subtree operation Stages

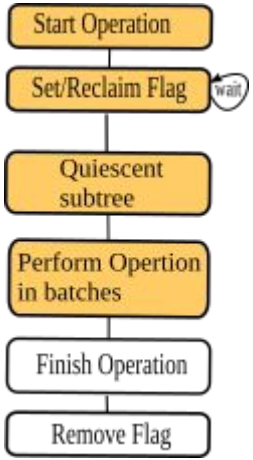


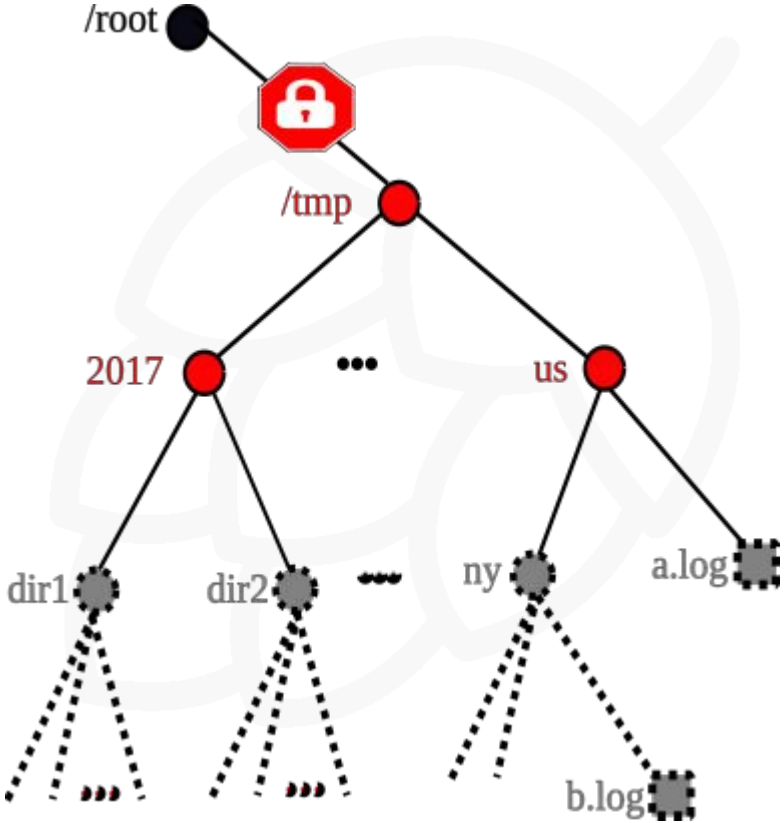


Subtree Locking Progresses Downwards

Subtree Locking (contd.)

Subtree operation Stages

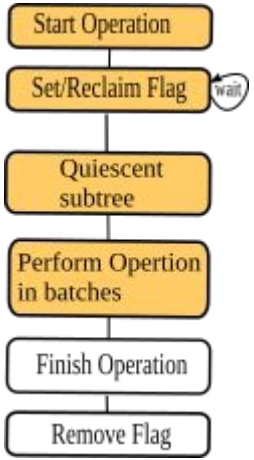


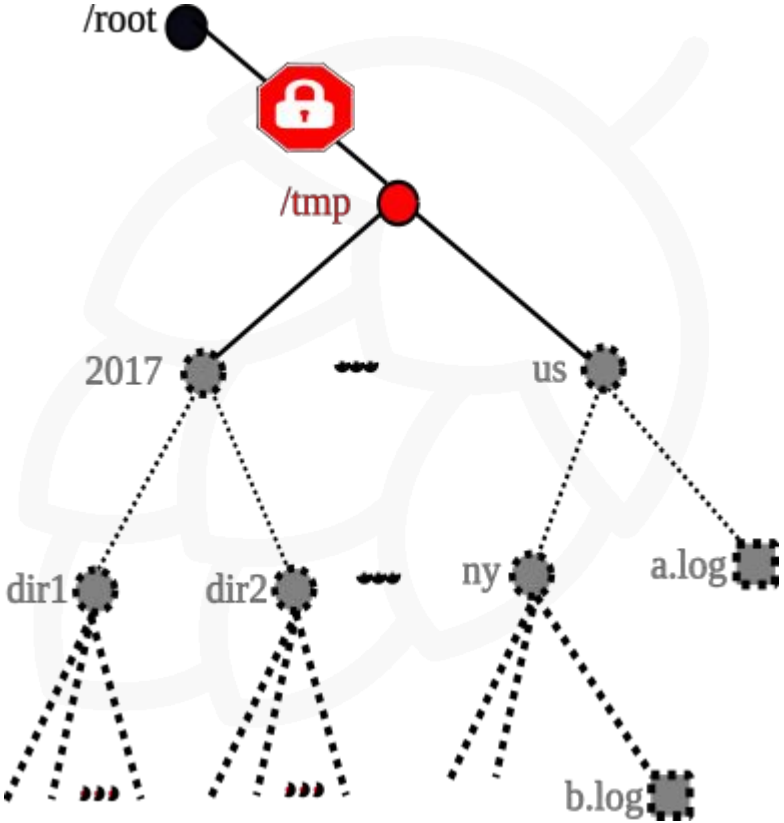


Delete Progresses Upwards

Subtree Locking (contd.)

Subtree operation Stages

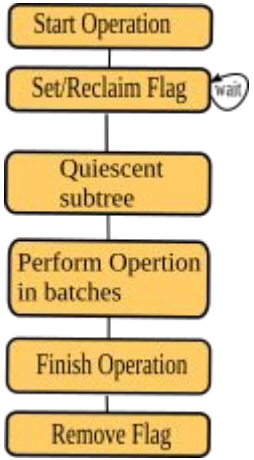


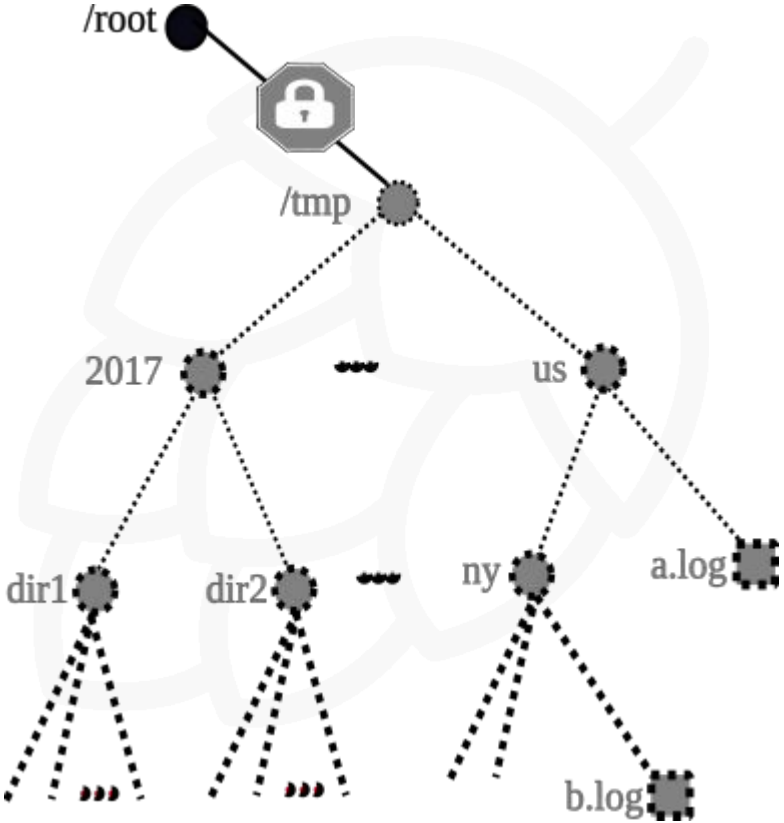


Delete Progresses Upwards

Subtree Locking (contd.)

Subtree operation Stages





Delete Progresses Upwards

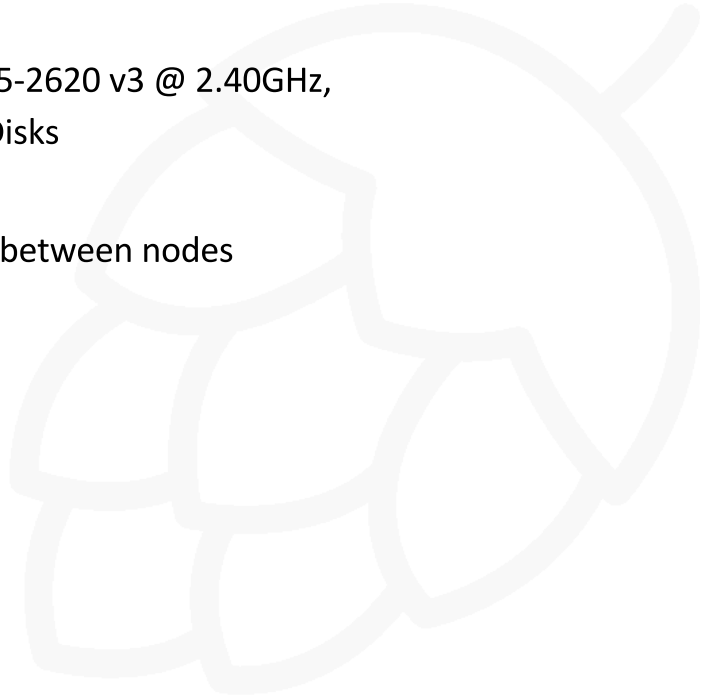
Failures during Subtree Operations

- All subtree operations are implemented in such a way that if the operations fail halfway, then the namespace is not left in an inconsistent state.
- If the NameNode executing the operation failed, the next NameNode to access the root of the subtree will reclaim the subtree lock (as the old NameNode will be marked as dead by the group membership service)

Evaluation

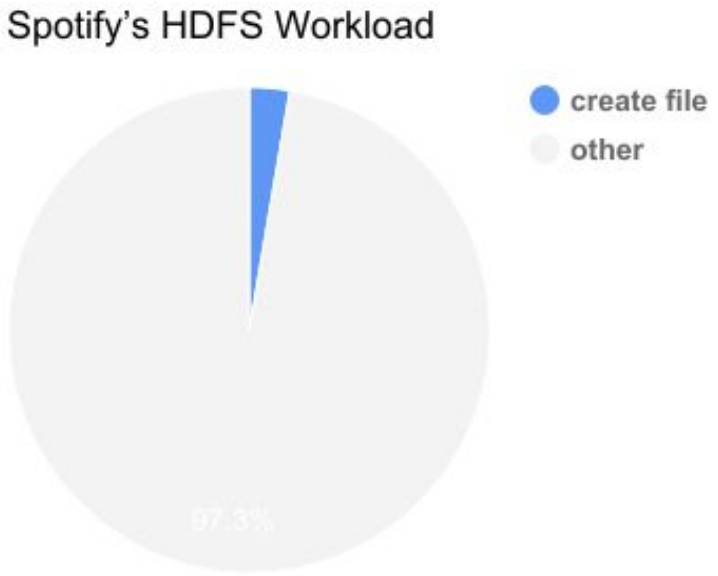
Evaluation

- On Premise
  - Dual Intel® Xeon® E5-2620 v3 @ 2.40GHz,
  - 256 GB RAM, 4 TB Disks
- 10 GbE
  - 0.1 ms ping latency between nodes



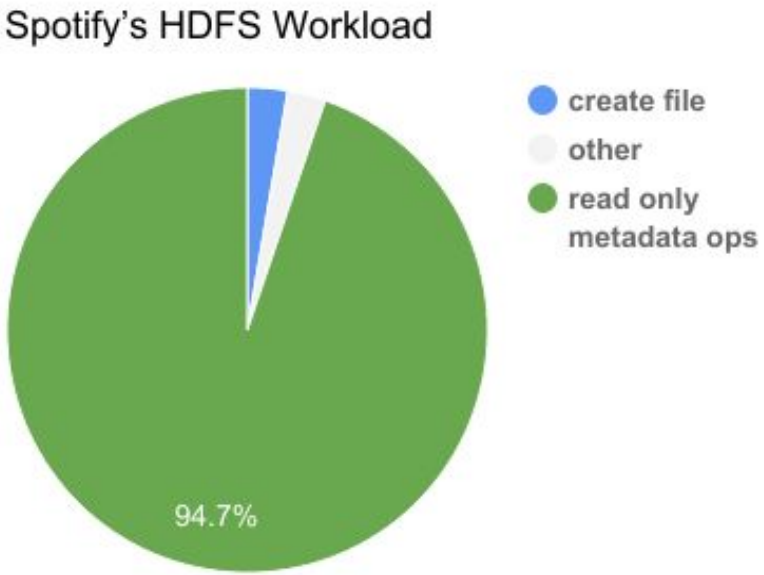
Evaluation

- On Premise
  - Dual Intel® Xeon® E5-2620 v3 @2.40GHz
  - 256 GB RAM, 4 TB Disks
- 10 GbE
  - 0.1 ms ping latency between nodes



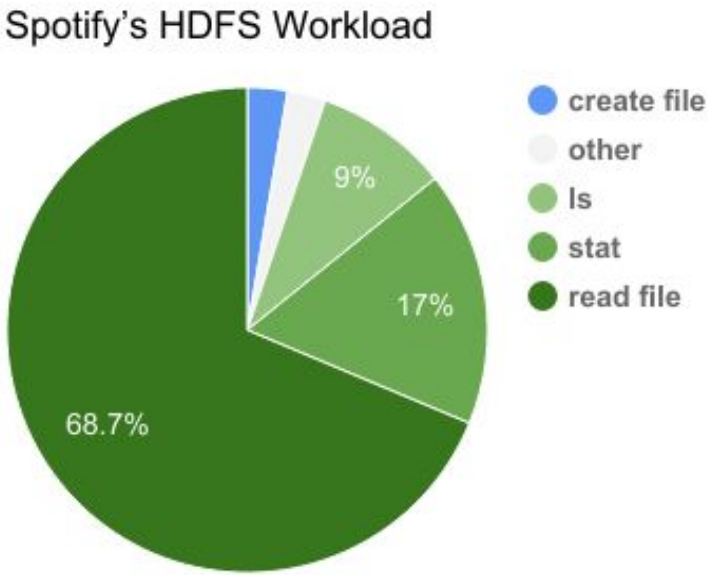
Evaluation

- On Premise
  - Dual Intel® Xeon® E5-2620 v3 @2.40GHz
  - 256 GB RAM, 4 TB Disks
- 10 GbE
  - 0.1 ms ping latency between nodes

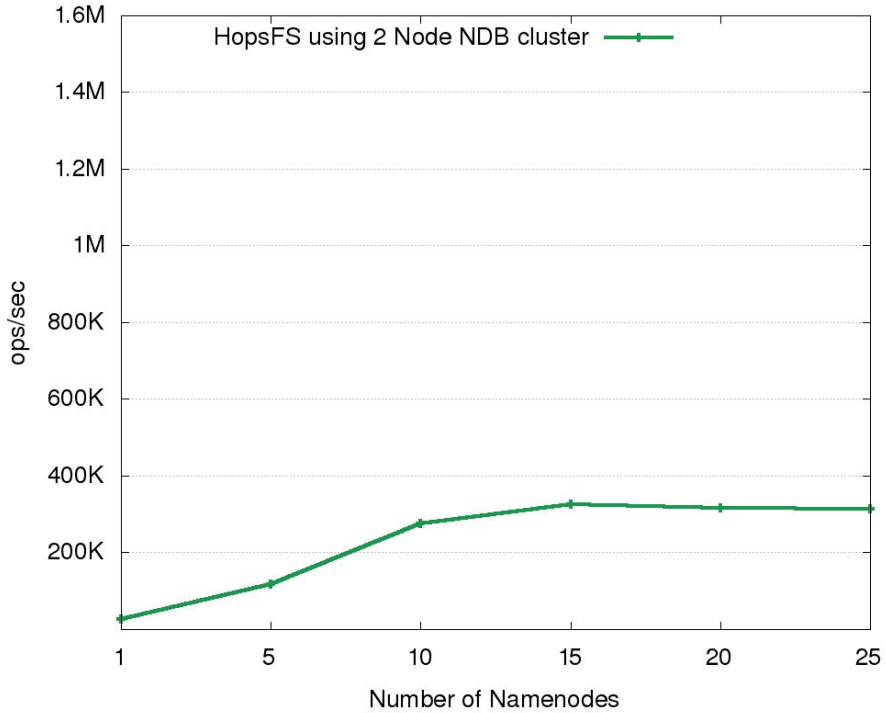


Evaluation

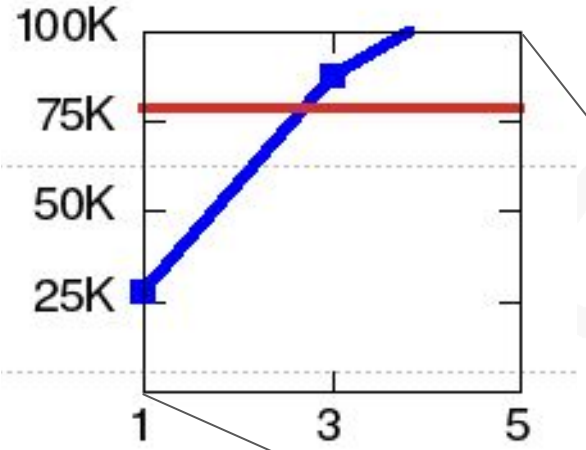
- On Premise
  - Dual Intel® Xeon® E5-2620 v3 @2.40GHz
  - 256 GB RAM, 4 TB Disks
- 10 GbE
  - 0.1 ms ping latency between nodes

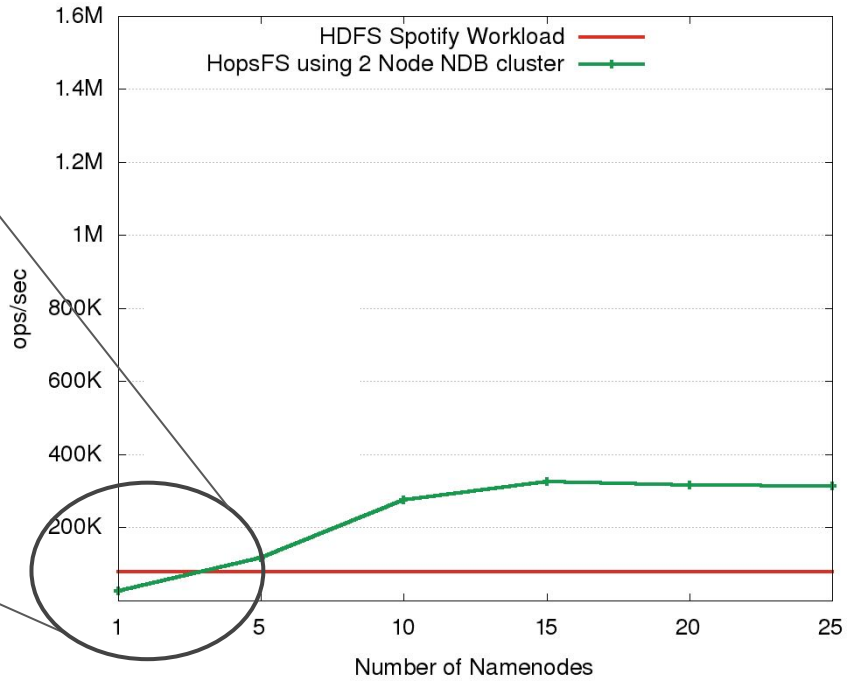


Evaluation: Industrial Workload



Evaluation: Industrial Workload

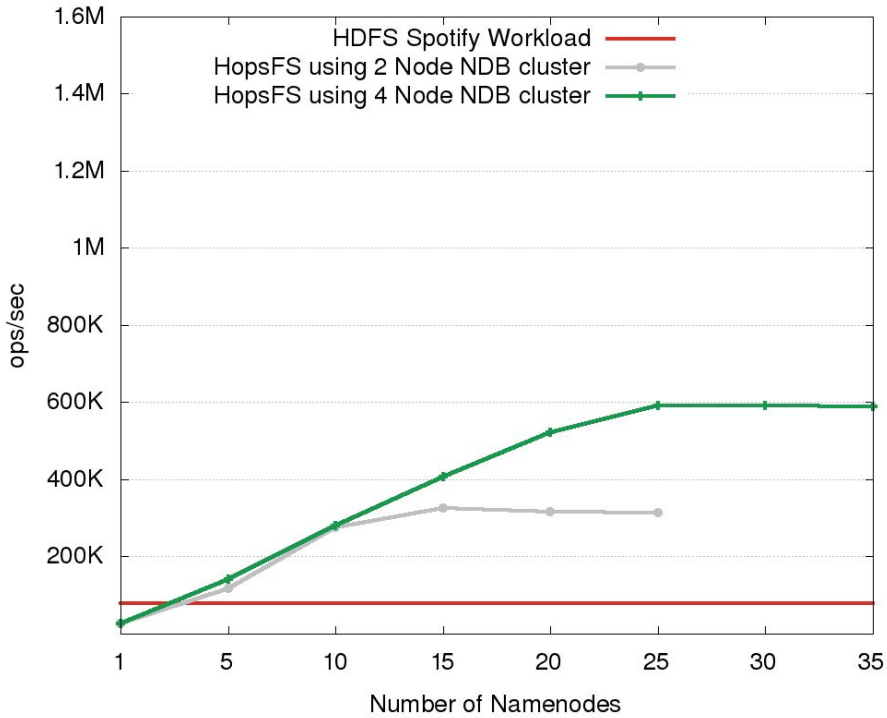




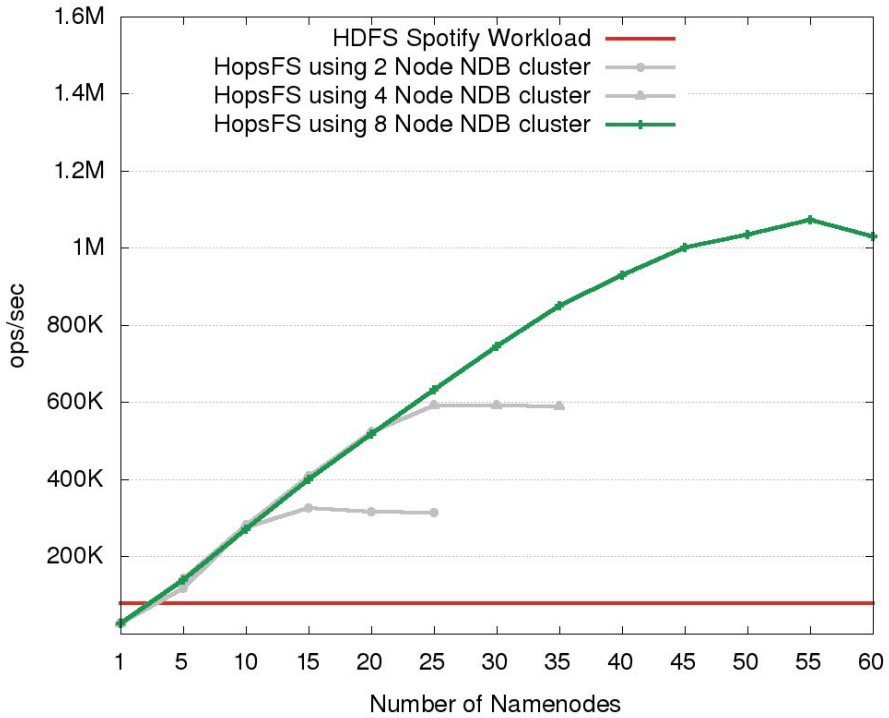
HopsFS outperforms with equivalent hardware: HA-HDFS with **Five** Servers

- 1 Active Namenode
- 1 Standby NameNode
- 3 Servers
  - Journal Nodes
  - ZooKeeper Nodes

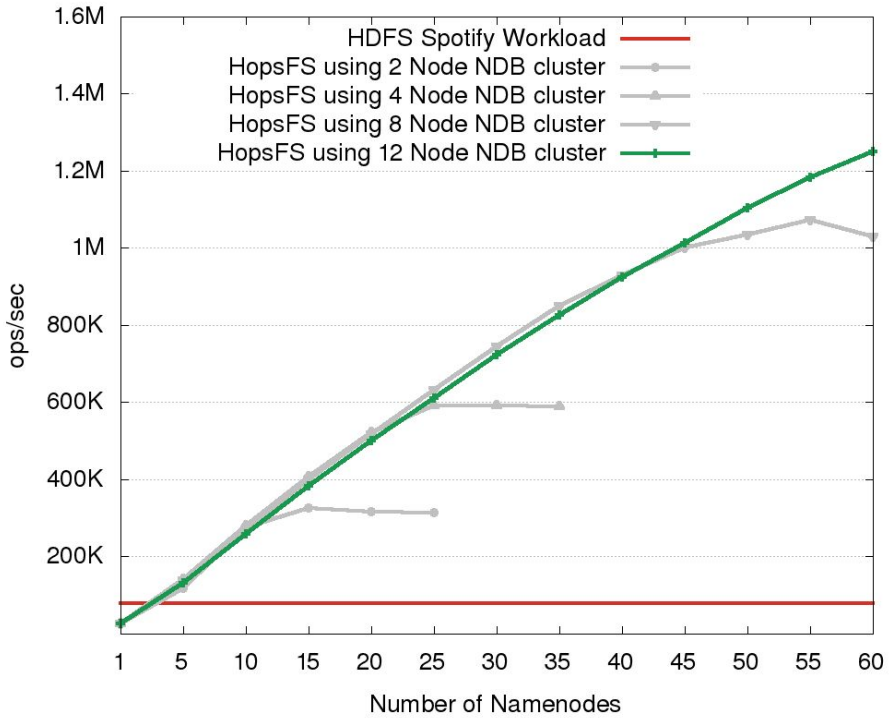
Evaluation: Industrial Workload (contd.)



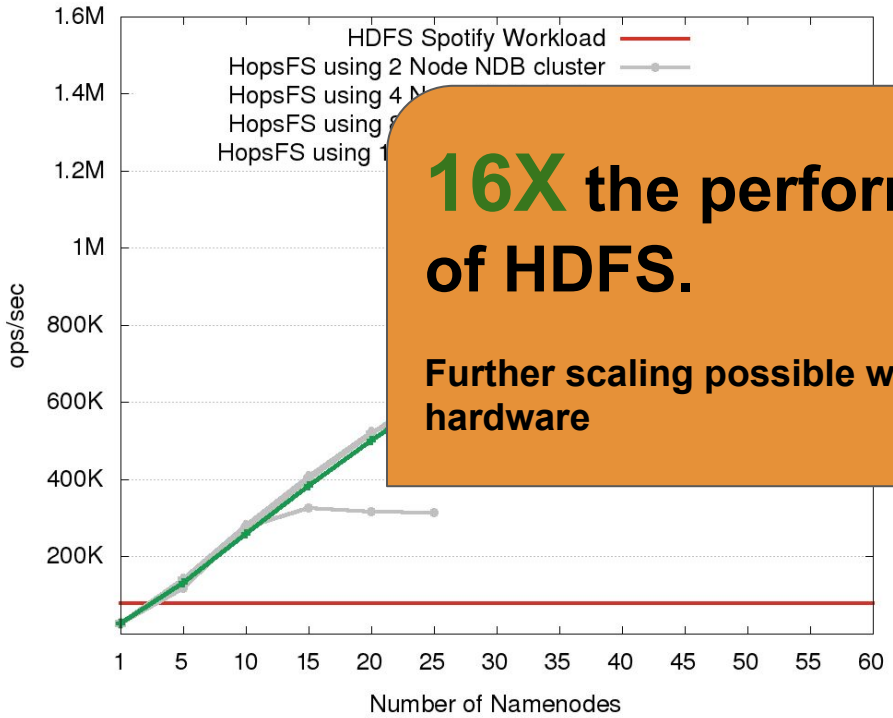
Evaluation: Industrial Workload (contd.)



Evaluation: Industrial Workload (contd.)



Evaluation: Industrial Workload (contd.)



16X the performance of HDFS.

Further scaling possible with more hardware

Write Intensive workloads

Workloads	HopsFS ops/sec	HDFS ops/sec	Scaling Factor
Synthetic Workload (5.0% File Writes)	**1.19 M**	**53.6 K**	**22**
Synthetic Workload (10% File Writes)	1.04 M	35.2 K	30
Synthetic Workload (20% File Writes)	0.748 M	19.9 K	37

Scalability of HopsFS and HDFS for write intensive workloads

Write Intensive workloads

Workloads	HopsFS ops/sec	HDFS ops/sec	Scaling Factor
Synthetic Workload (5.0% File Writes)	1.19 M	53.6 K	22
Synthetic Workload (10% File Writes)	1.04 M	35.2 K	30
Synthetic Workload (**20%** File Writes)	0.748 M	19.9 K	**37**

Scalability of HopsFS and HDFS for write intensive workloads

Metadata Scalability

HDFS Metadata **200 GB** → Max **500 million** files/directories

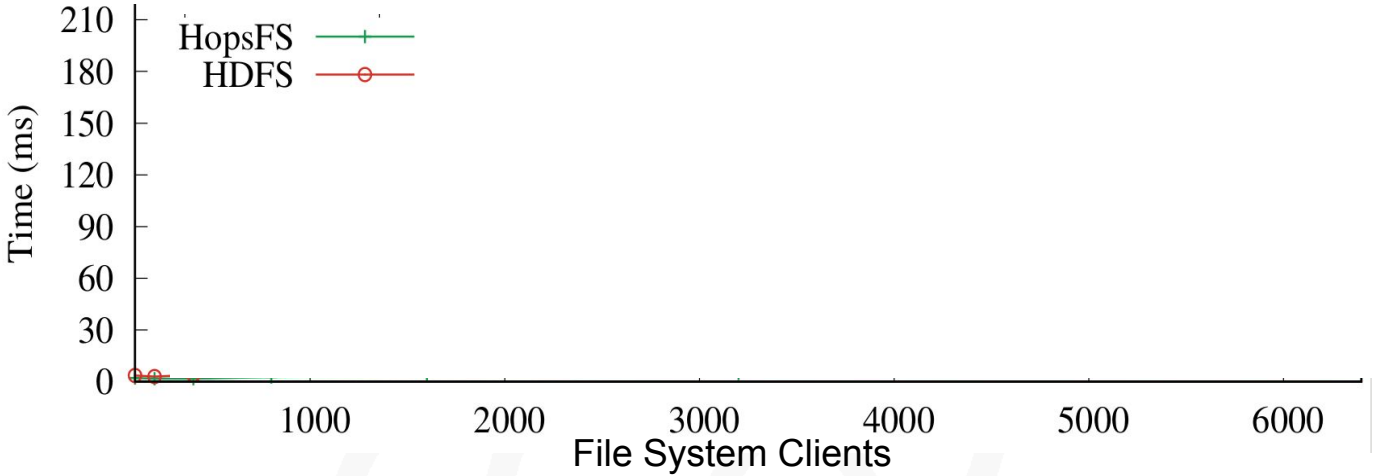
Metadata Scalability

HDFS Metadata **200 GB** → Max **500 million** files/directories

HopsFS Metadata **24 TB** → **Max 17 Billion** files/directories

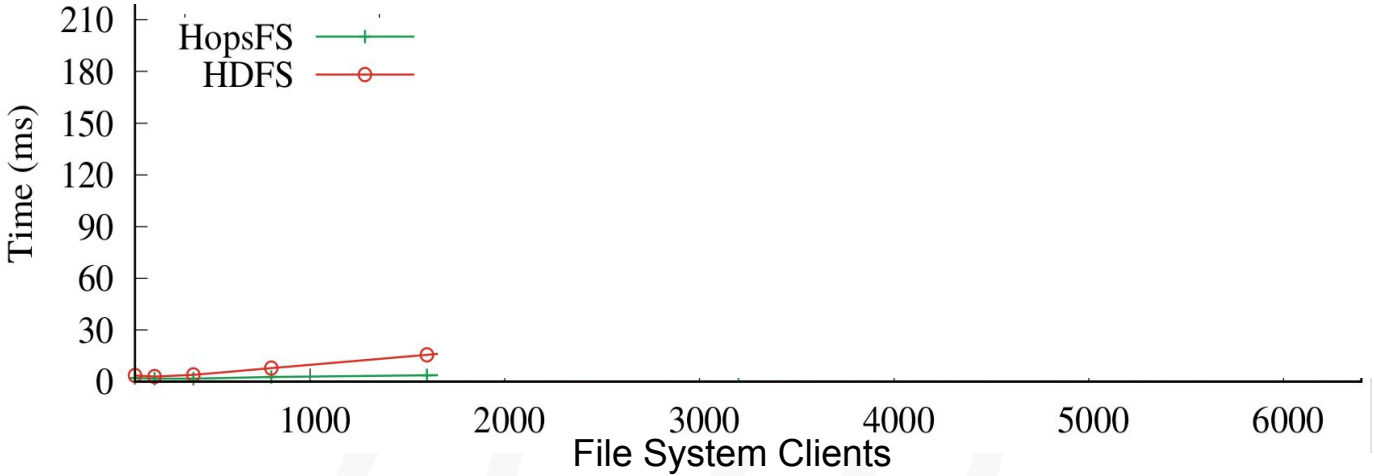
➢ **37 times** more files than HDFS

Operational Latency



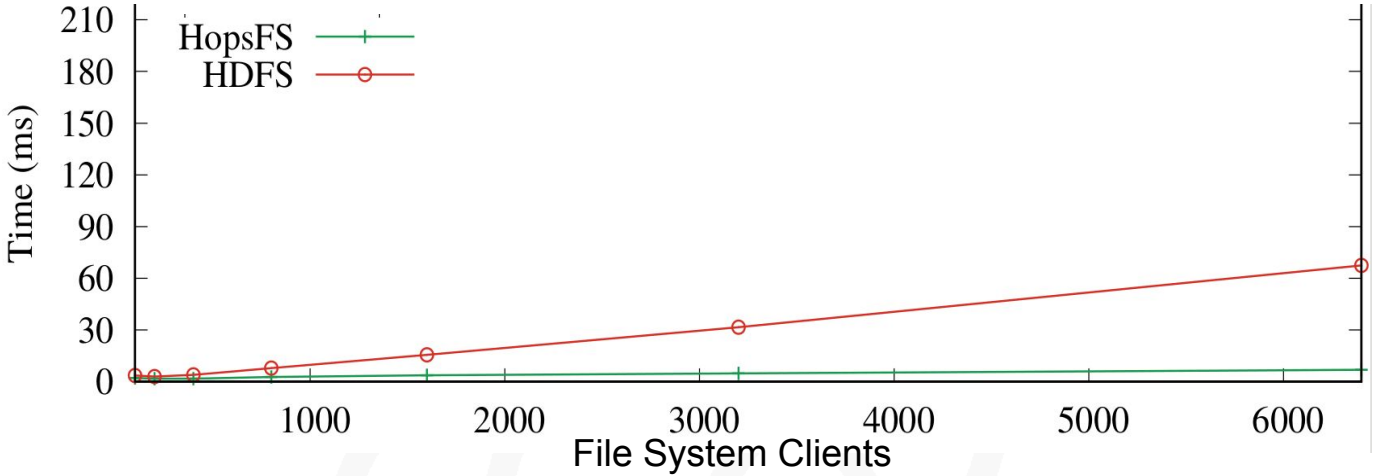
No of Clients	HopsFS Latency	HDFS Latency
50	3.0	3.1

Operational Latency



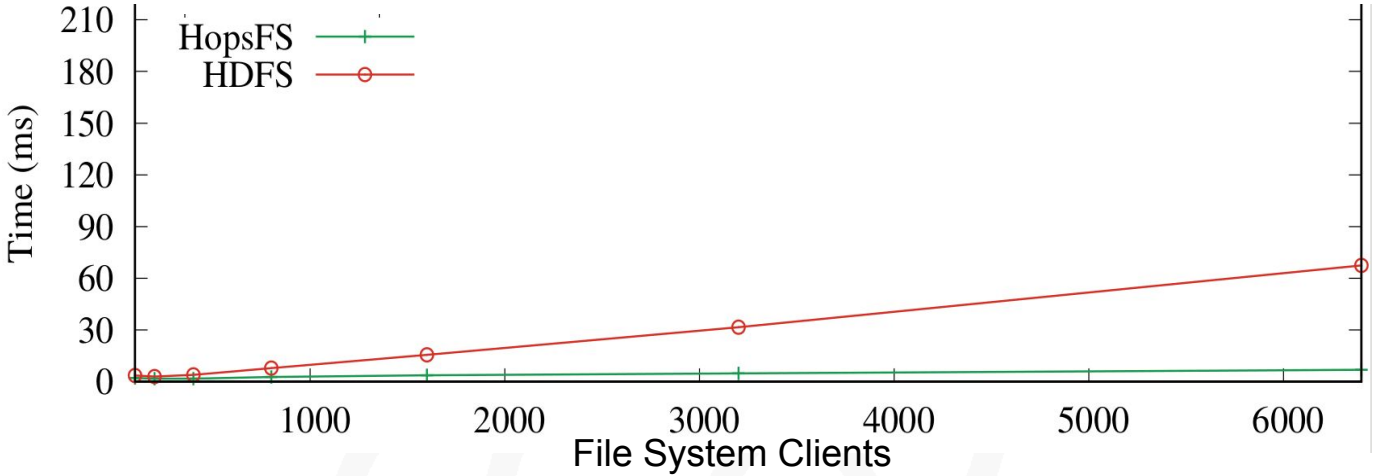
No of Clients	HopsFS Latency	HDFS Latency
50	3.0	3.1
1500	**3.7**	**15.5**

Operational Latency



No of Clients	HopsFS Latency	HDFS Latency
50	3.0	3.1
1500	3.7	15.5
6500	**6.8**	**67.4**

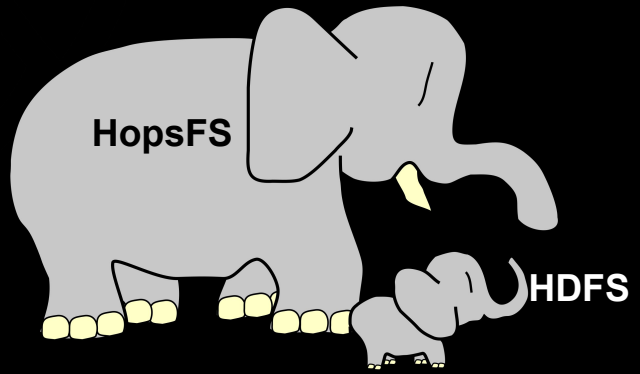
Operational Latency



For 6500 clients HopsFS has **10 times** lower latency than HDFS

Conclusion

- Production-quality distributed hierarchical file system with transactional distributed metadata
- HopsFS has 16X-37X the throughput of HDFS, with even higher throughput possible with more hardware



Questions


http://www.hops.io


http://github.com/hopshadoop


@hopshadoop