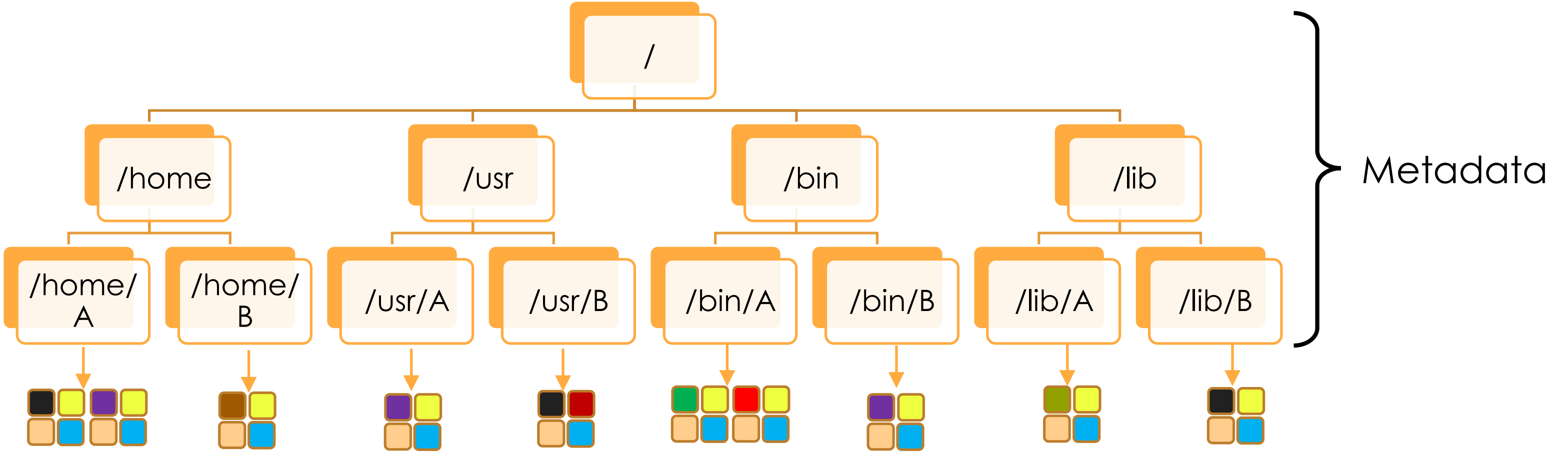
HopsFS
Scaling Distributed Hierarchical File Systems Using NewSQL Databases

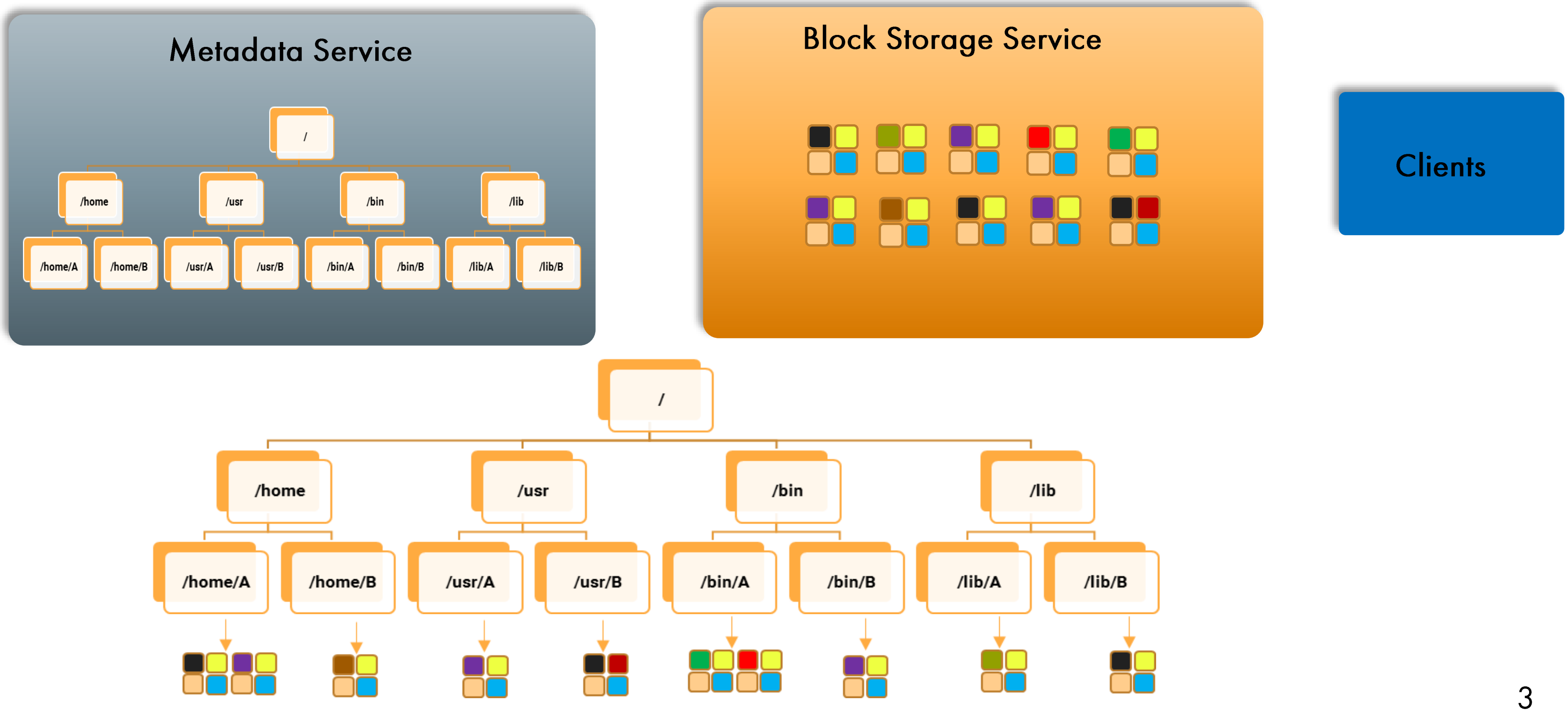
Salman Niazi

Hierarchical File System



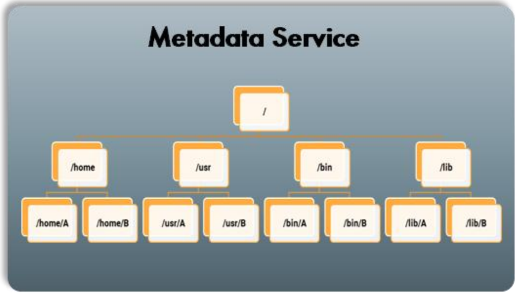
- Strongly consistent metadata.
  - Atomic file system operations, such as, move and create

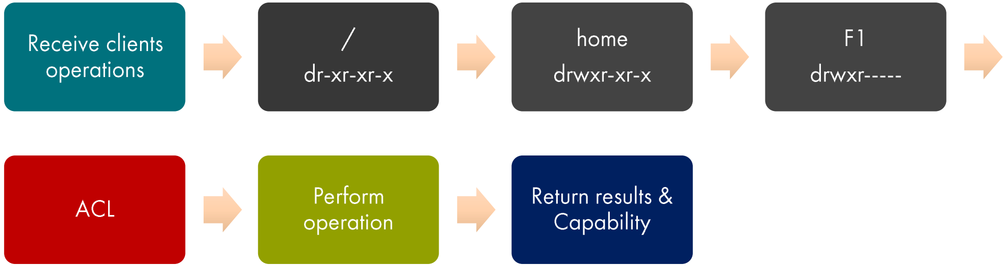
Distributed Hierarchical File System

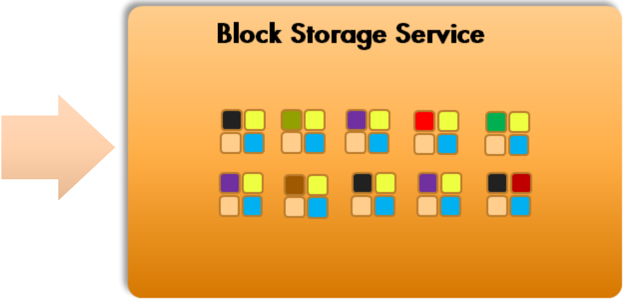


Typical Hierarchical File System Operation

- `{operation} [flags] {path(s)}`
    - cat /home/F1

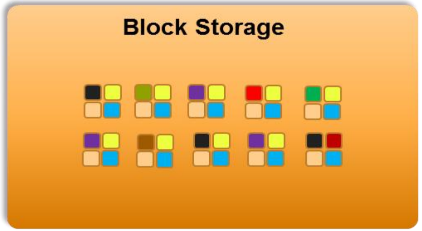




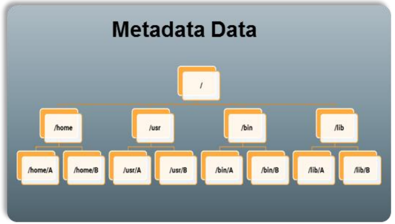


Data Blocks Storage Layer

- Thousands of servers
- Uses data replication and erasure coding for high availability



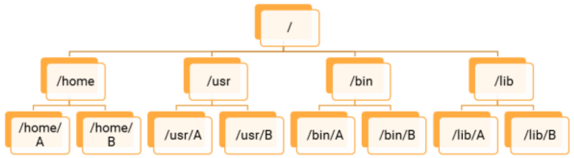
Metadata Service Layer



- Atomic File System Operations
- Due to complexity of metadata service monolithic architecture is the most popular solution
  - HDFS, GFS, AFS

Why not use Databases?

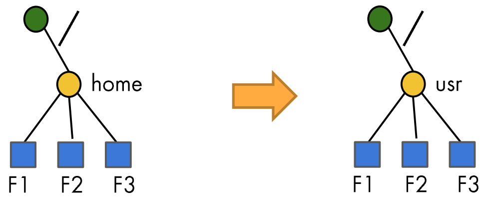
- Metadata consists of lots of very small data
- Databases specialize is storing and manipulating large amounts of small data



- Traditional databases do not provide high throughput required by distributed file systems
- High operational latencies for resolving file paths

Using Databases

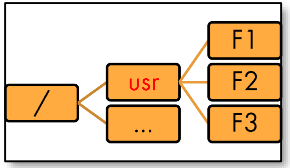
- WinFS by Microsoft
- GiraffaFS, CassandaraFS, CalvinFS



inode	...
/	
/usr	
/usr/F1	
/usr/F2	
/usr/F3	

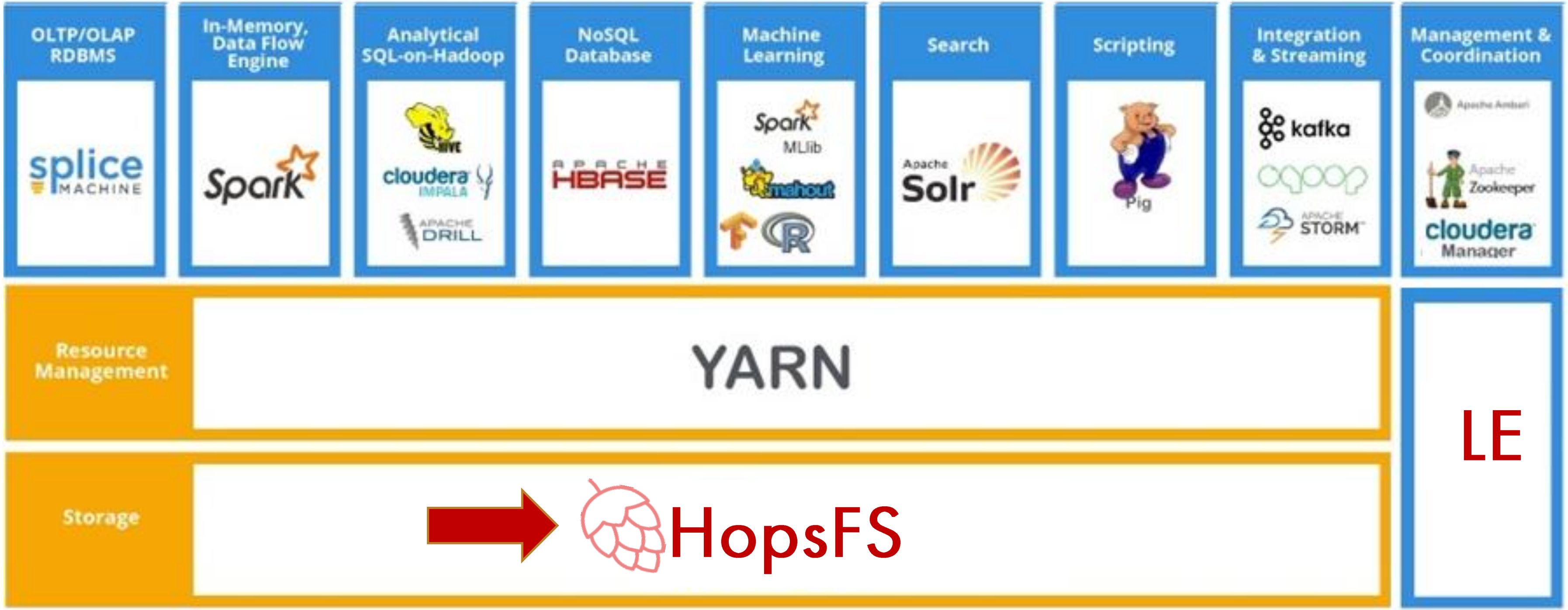
Denormalized

ID	name	PID	
0	/	-1	
1	usr	0	
2	F1	1	
3	F2	1	
4	F3	1	



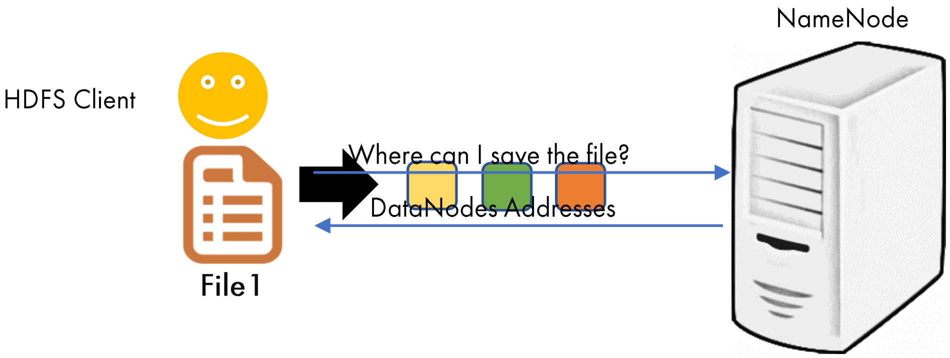
Normalized

HopsFS



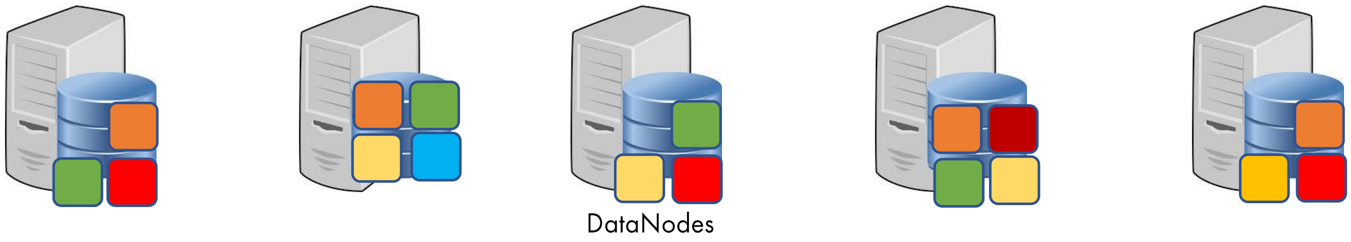
Hadoop Software Stack

HDFS Architecture

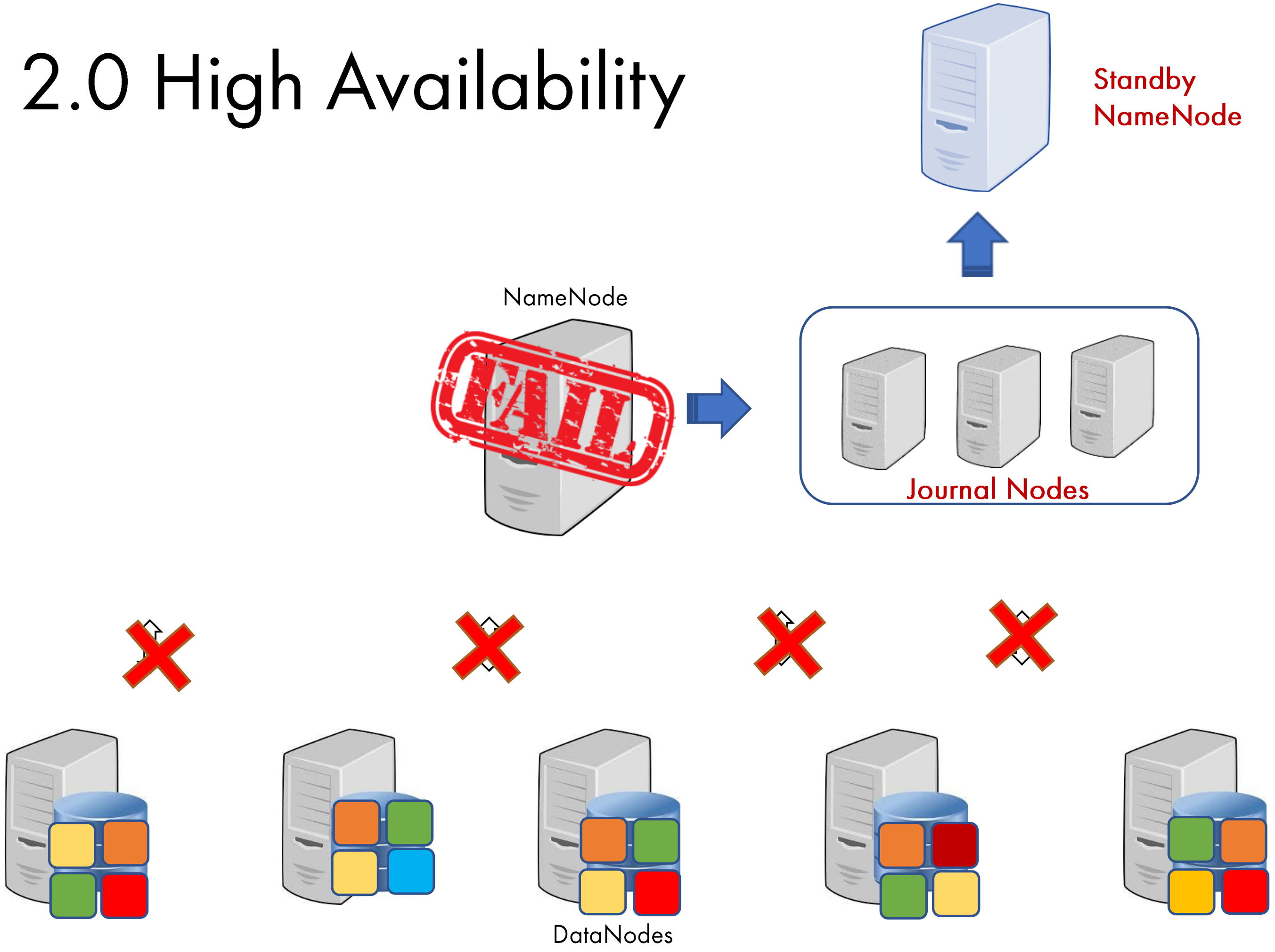


File System Metadata

File	Blocks Mappings
File1	Blk1 → DN1, Blk2 → DN5, Blk3 → DN3
File2	Blk1 → DN1, Blk2 → DN4
File3	Blk1 → DN1, Blk2 → DN2, Blk3 → DN3
File4	Blk1 → DN100
File5	Blk1 → DN4, Blk2 → DN2, Blk3 → DN9
	… … … …
FileN	Blk1 → DN2, Blk2 → DN8



HDFS 2.0 High Availability

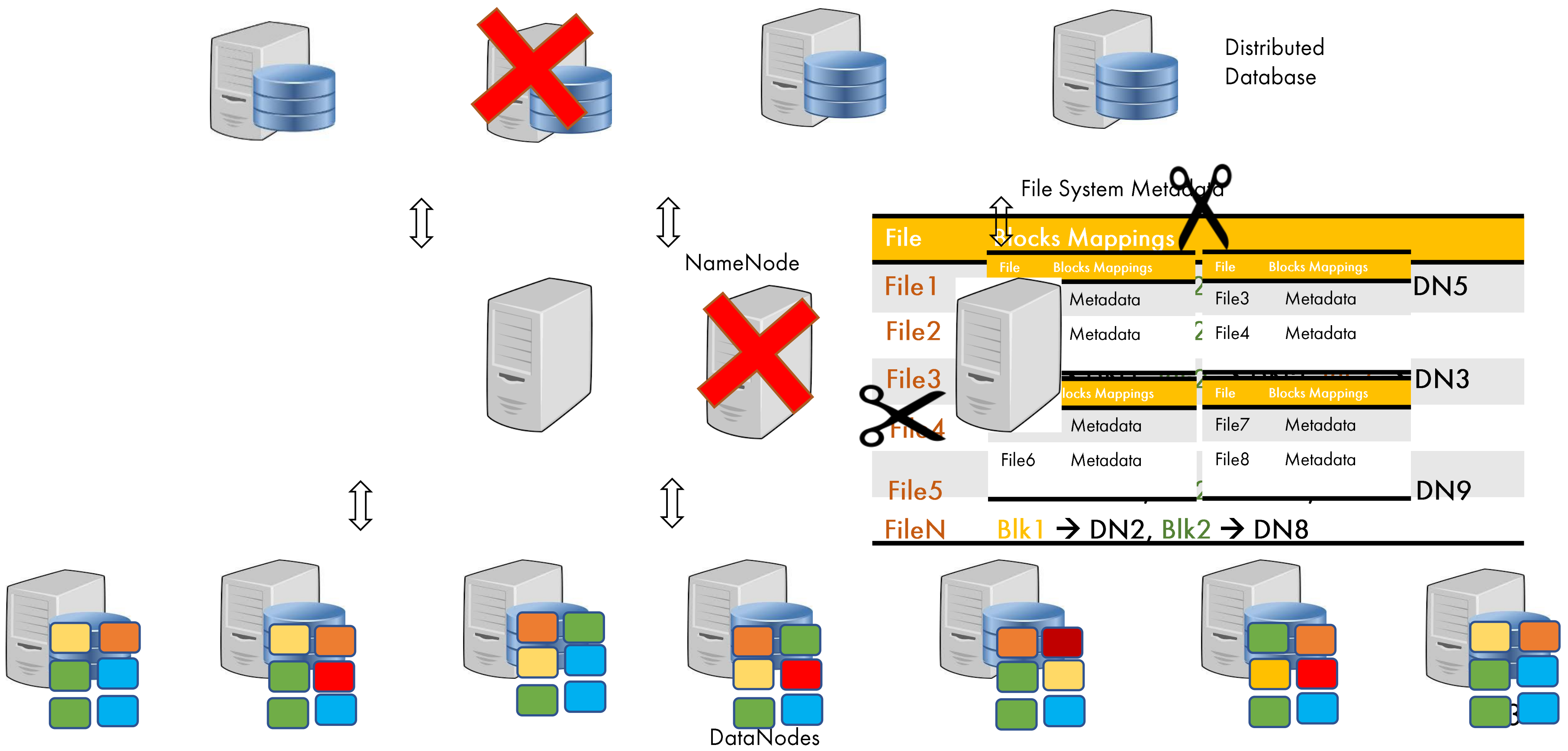


HDFS Limitations

- HDFS has been scaled to store **100 PB – 200 PB** on **4000 – 5000 datanodes**
- Namespace size upper bound: **~ 500 million files**

- At most **70 – 80 thousand** file system operations / sec

HopsFS Architecture



NewSQL DB

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

- **Open Source**
- **Commodity Hardware**
  - Scales to 48 database nodes
    - 200 Million Read Ops/Sec* using NDB native API
  - Read Committed Transaction Isolation
    - Row-level Locking
  - User-defined partitioning

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

Transaction Isolation

Transaction Isolation Levels

MySQL Cluster Network Database Engine only supports Read-Committed Transaction Isolation Level

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	may occur	may occur	may occur
Read Committed	-	**may occur**	**may occur**
Repeatable Read	-	-	may occur
Serializable	-	-	-

Read Committed Transaction Isolation Level

Transaction 1

Start Tx
Read x (x = 1)

Read x (x = 1)

Read x (x = 10)

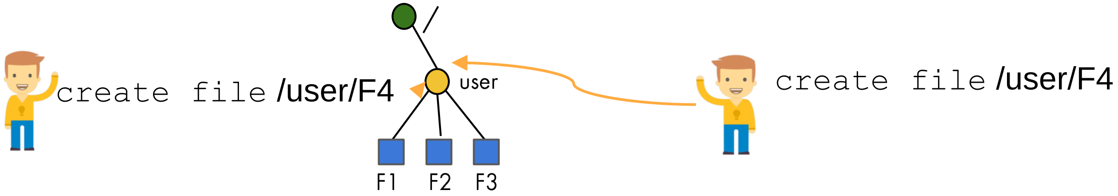
The value of X has changed before the transaction is committed

Transaction 2

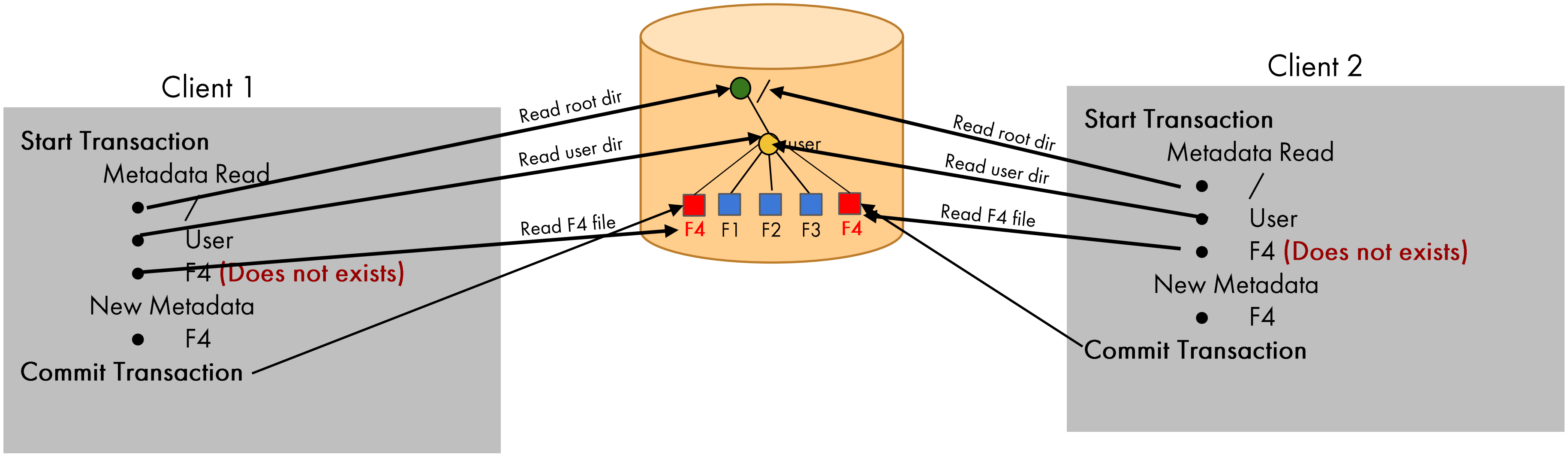
Start Tx
Read x (x = 1)
Update x = 10

Commit

Read Committed Transaction Isolation



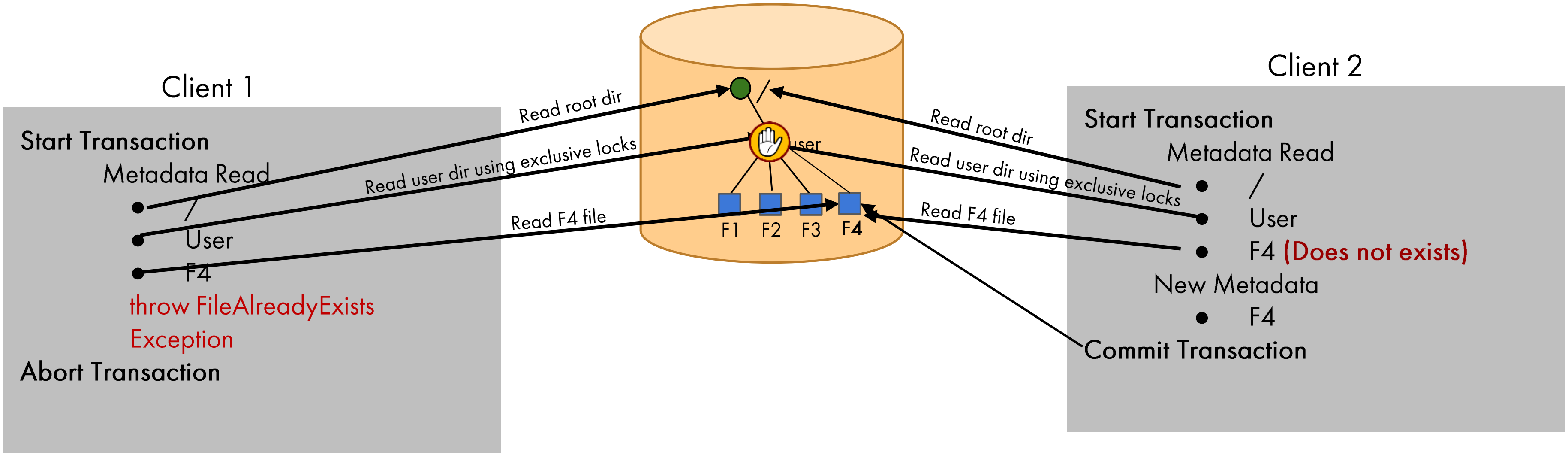
Read Committed Transaction Isolation



Read Committed Transaction Isolation

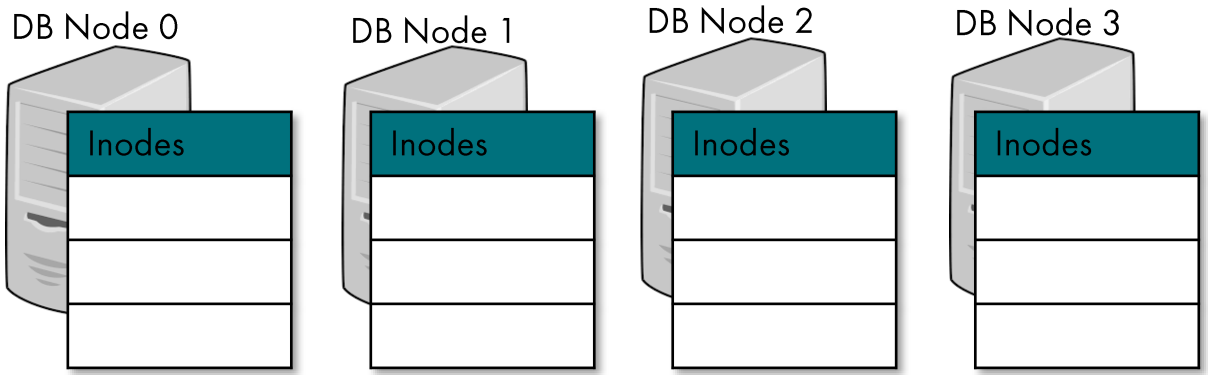
Use row level locking to serialize conflicting file operation

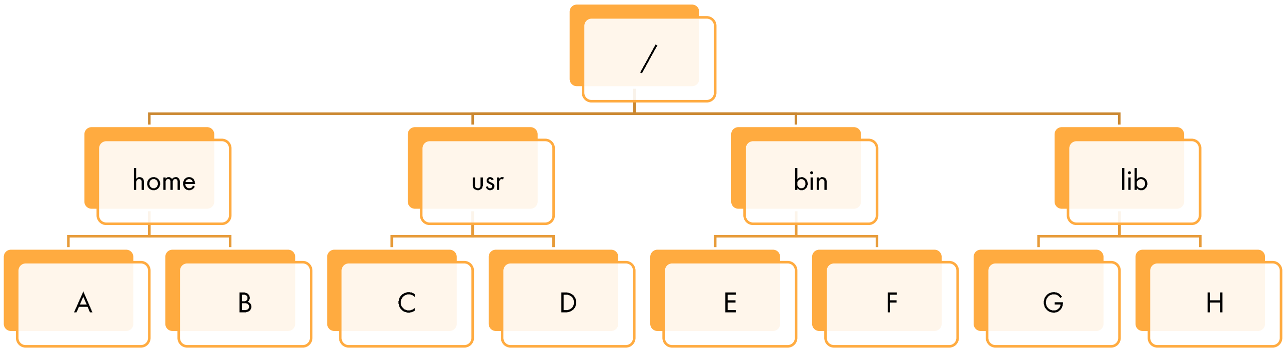
Read Committed Transaction Isolation With Locking



Database Operations
&
Data Partitioning

Distributed Metadata Design





Distributed Metadata Design

TC

ID	name	PID
1	home	0
7	C	2
10	F	3
	...	

DB Node 0

TC

ID	name	PID
5	A	1
8	D	2
4	lib	0
	...	

DB Node 1

TC

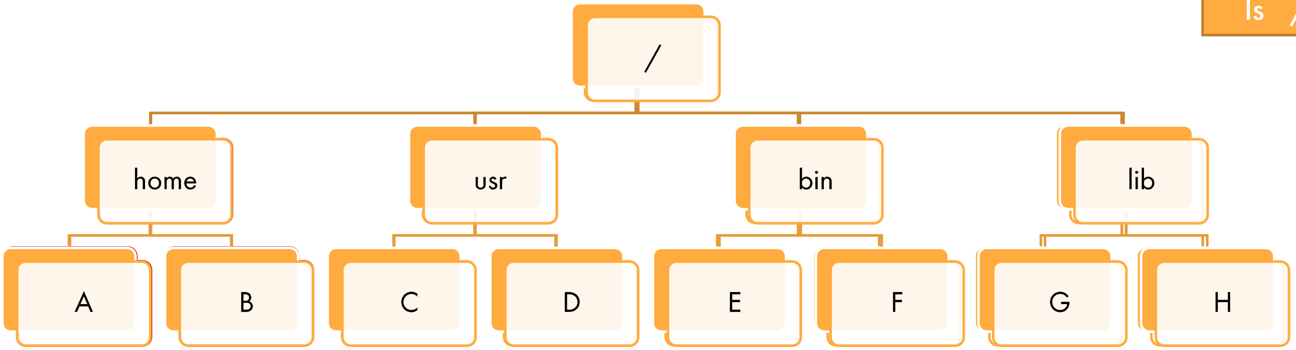
ID	name	PID
6	B	1
3	bin	0
11	G	4
	...	

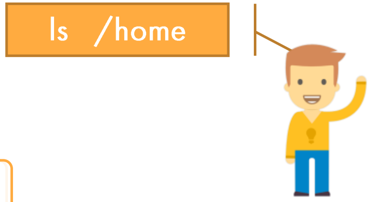
DB Node 2

TC

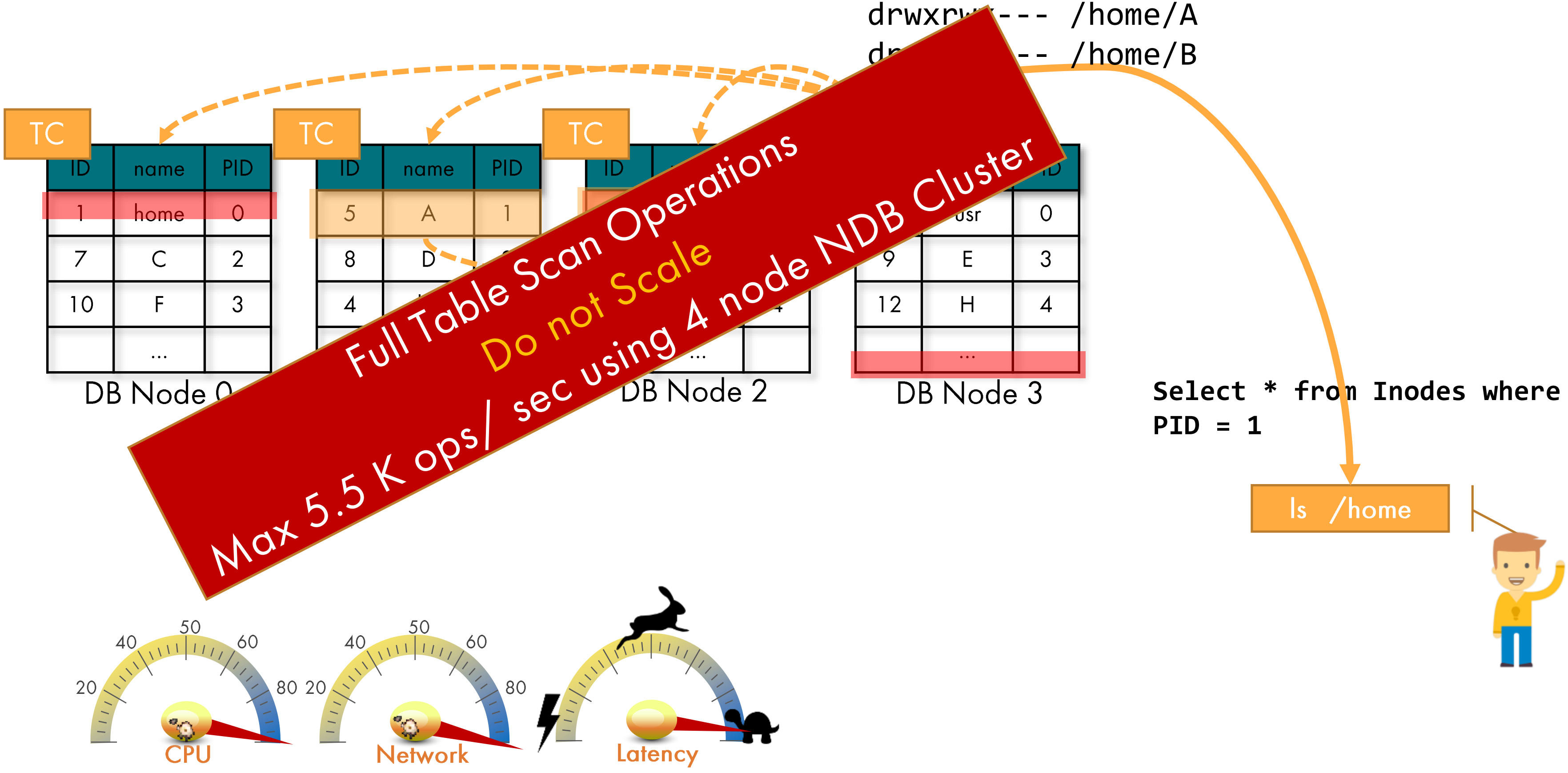
ID	name	PID
2	usr	0
9	E	3
12	H	4
	...	

DB Node 3





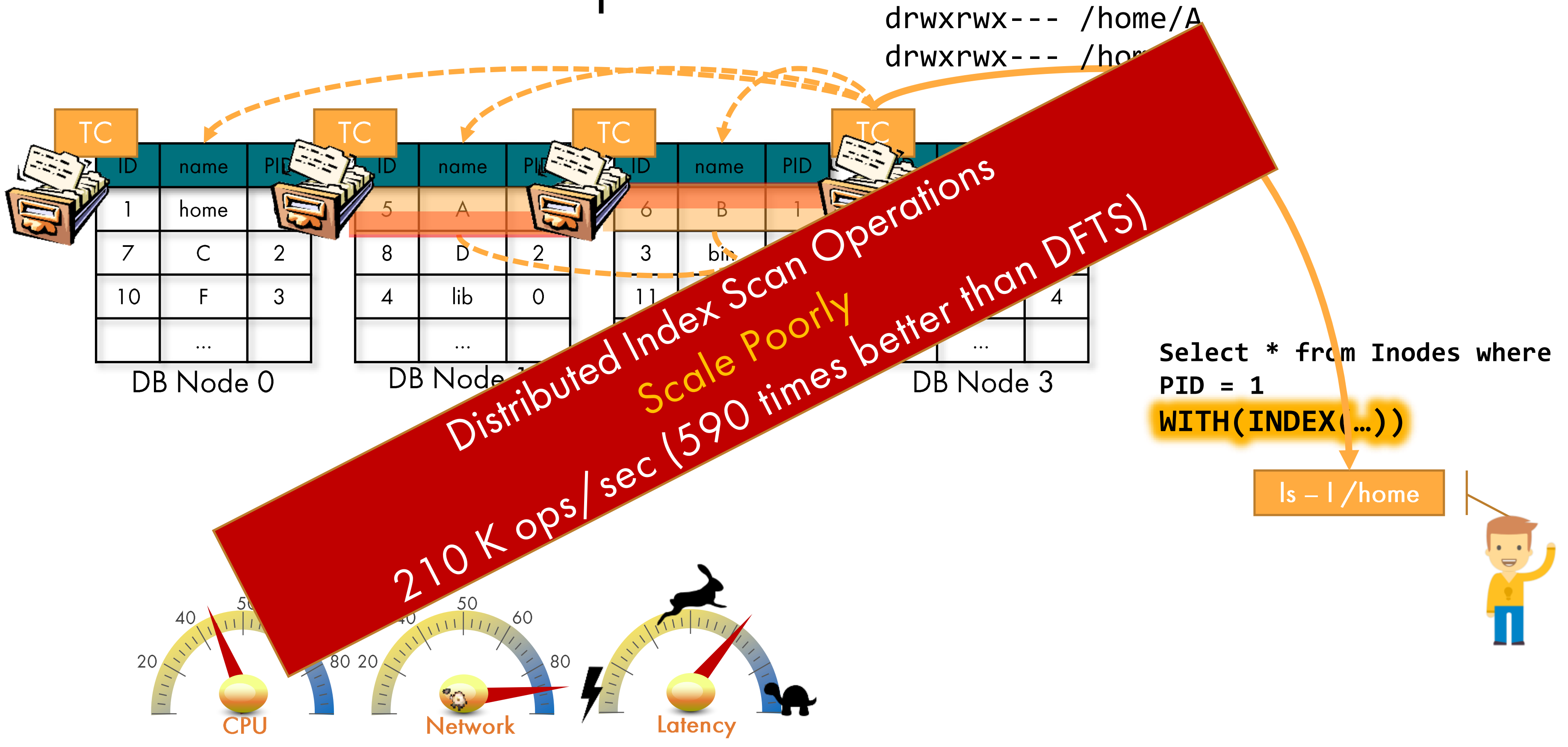
Distributed Full Table Scans



drwxrw---- /home/A
dr -- /home/B

Select * from Inodes where PID = 1

Distributed Index Scan Operations



HopsFS

- Uses NewSQL Relational Database that allows

1. User Defined Partitioning (UDP): Take control of how the data is distributed across difference database nodes
2. Distribution Aware Transactions (DAT): Take control over which Transaction Coordinator handles which file system operations.

Solution (User Defined Partitioning)

DB Node 0

ID	Name	PID
1	/	0
9	E	4
10	F	4
15	K	8
15	L	8

DB Node 1

ID	Name	PID
2	home	1
3	usr	1
4	bin	1
11	G	5
12	H	5

DB Node 2

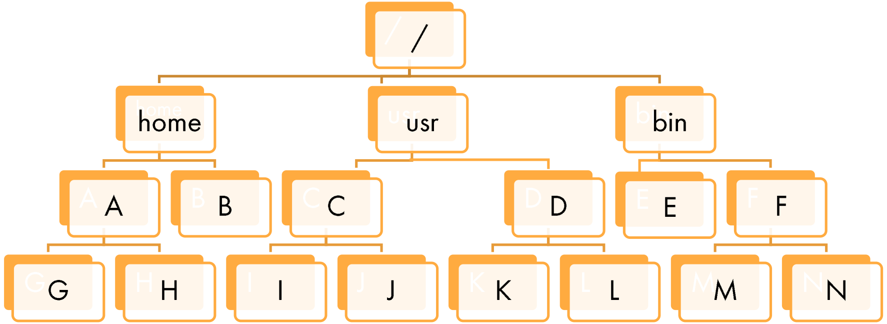
ID	Name	PID
5	A	2
6	B	2
16	M	10
17	N	10

DB Node 3

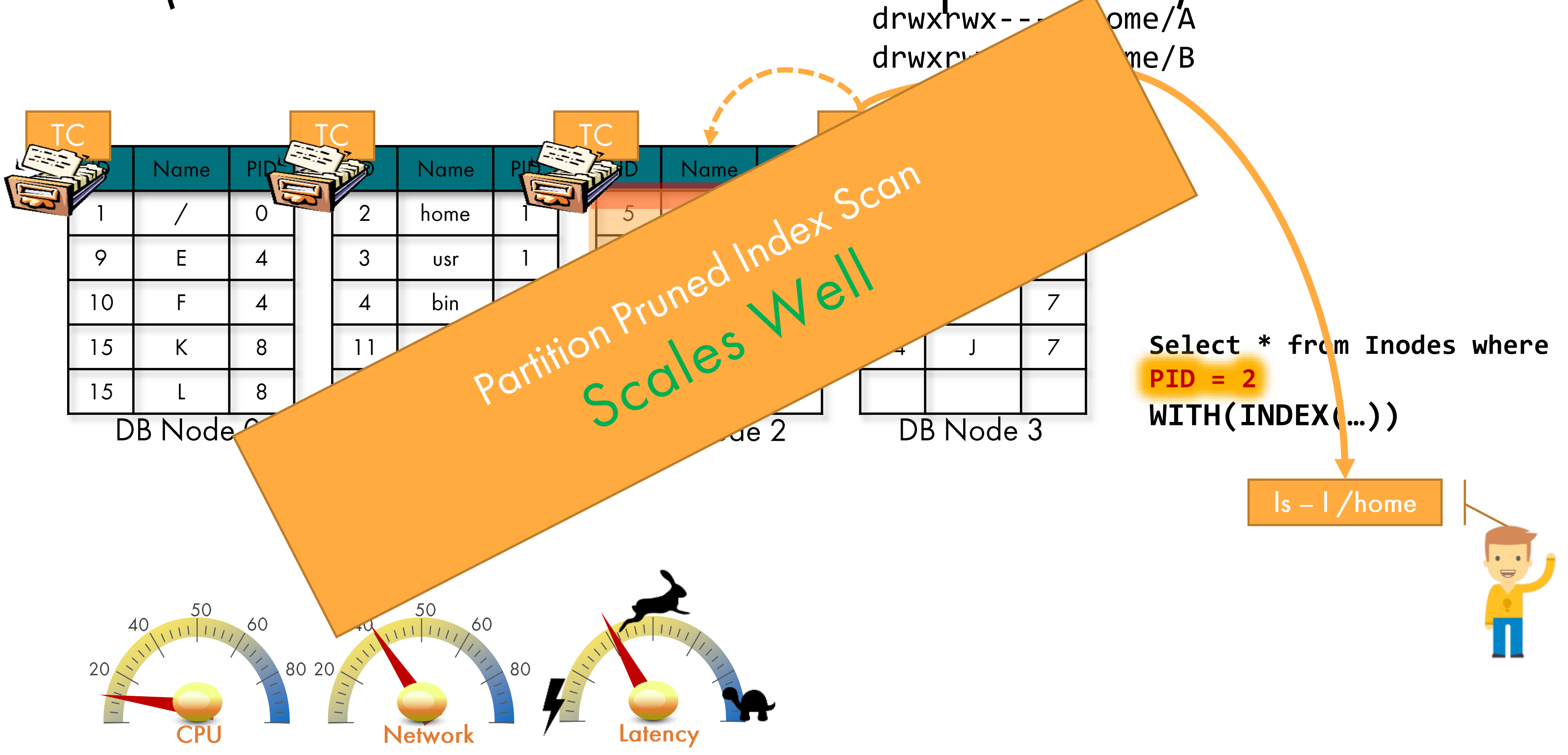
ID	Name	PID
7	C	3
8	D	3
13	I	7
14	J	7

Hash Fn

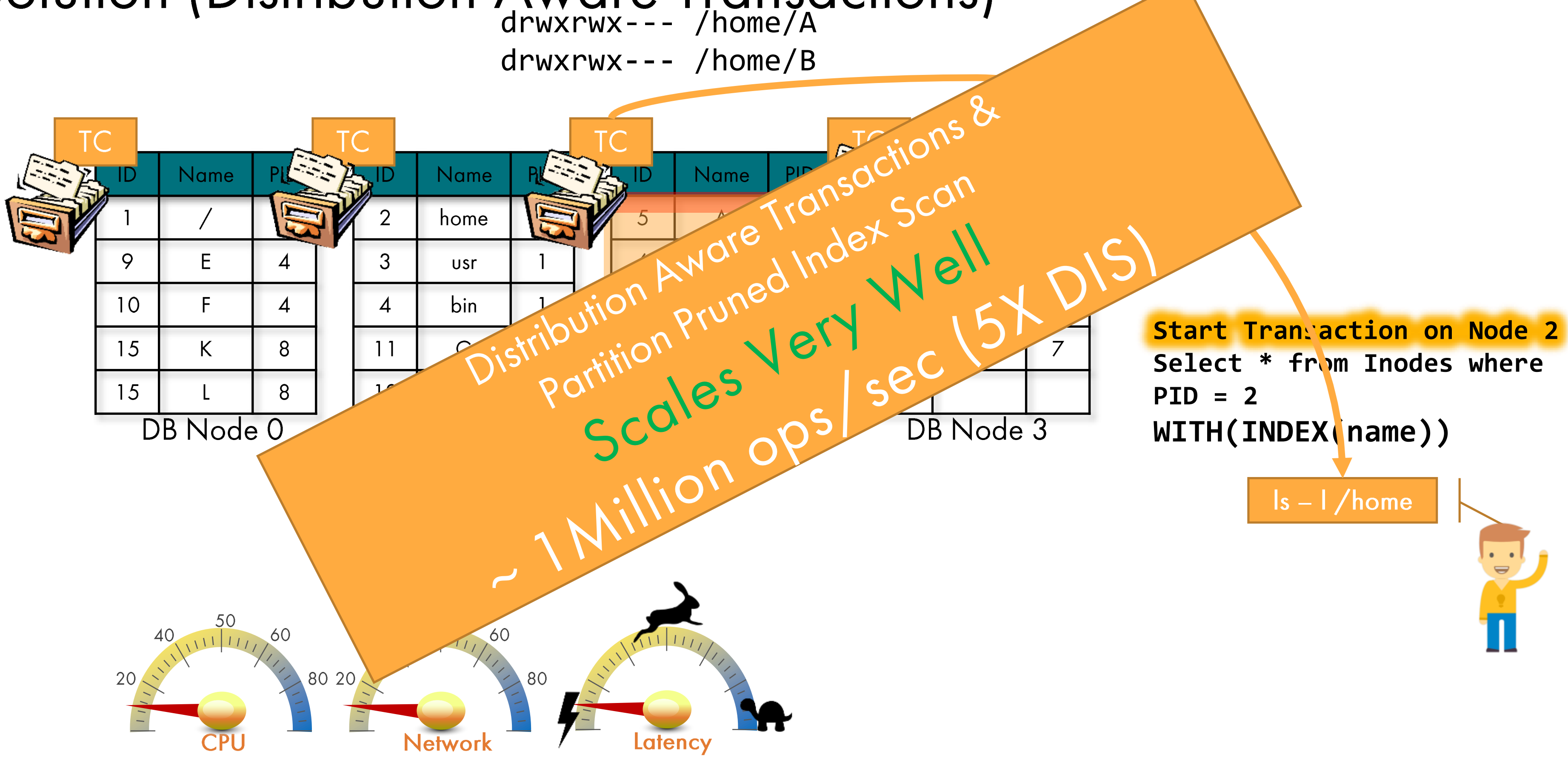
PID % 4 = Partition No



Solution (Partition Pruned Index Scan Operations)



Solution (Distribution Aware Transactions)



Transactional FS Operations

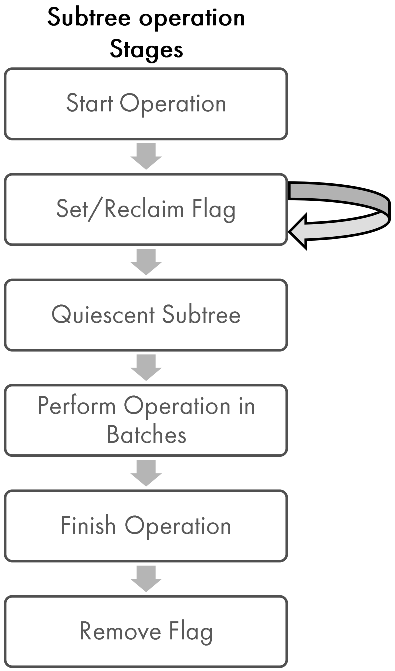
- File System Operation
  - Distributed Transaction
    - START DISTRIBUTION AWARE TRANSACTION
      - Primary Key Ops
      - Partition Pruned Index Scan Ops
      - Batching and Caching
    - COMMIT TRANSACTION

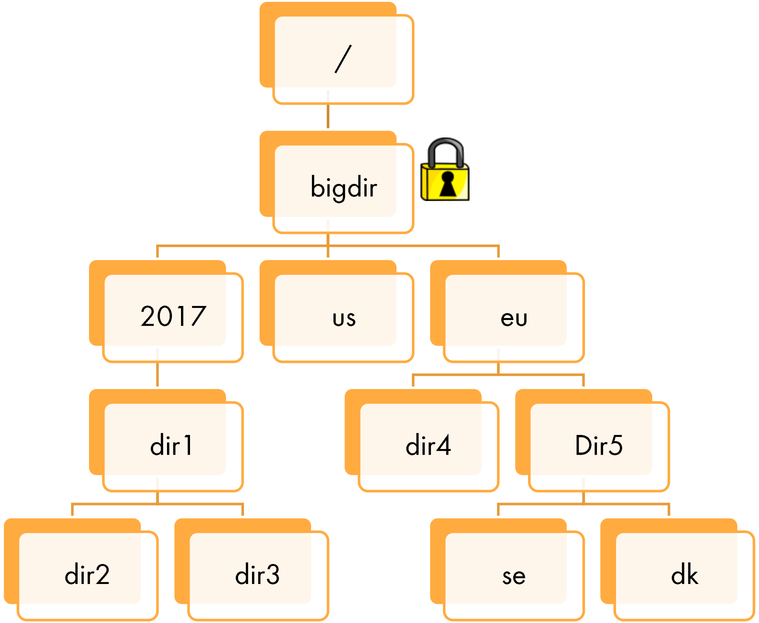
Solution (Contd.)

- ~40 Tables & hundreds of FS operations.
- Most NewSQL databases do not provide serializable transaction isolation level
  - HopsFS uses **read committed** transaction isolation level and **row level locks**
- Avoiding deadlocks in file system operation.
  - Total Order Locking
- Implementing large file system operations that do not fit in a transaction

Large File System Operations

Subtree Operations





Failures during Subtree Operations

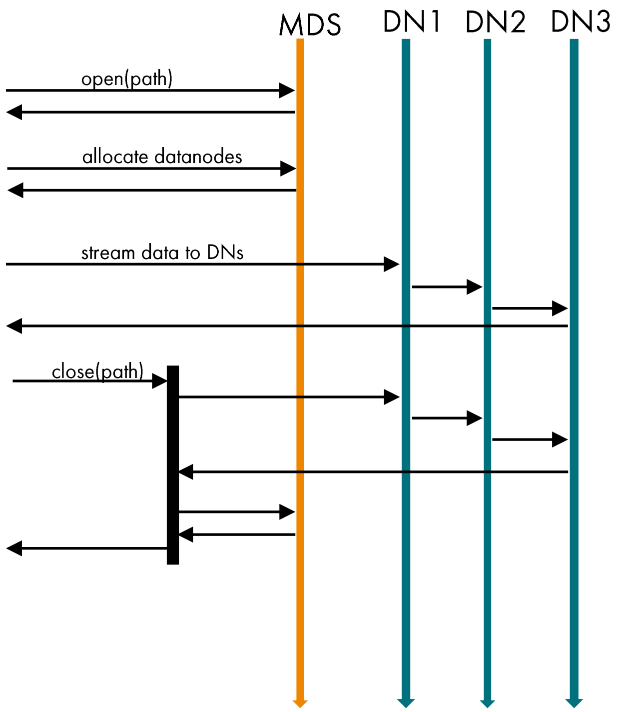
All subtree operations are implemented in such a way that if the operations fail halfway, then the namespace is not left in an inconsistent state.

HopsFS Performance

- Throughput
  - 16X the throughput of HDFS (Spotify Workload).
  - 38X the throughput of HDFS for 20% write intensive workload
- Low Latency
  - Identical avg op latency (~3ms) for small number (50) of clients
  - **10X** lower latency for large number (6500) of clients
- Metadata Scaling
  - **37X** more metadata.

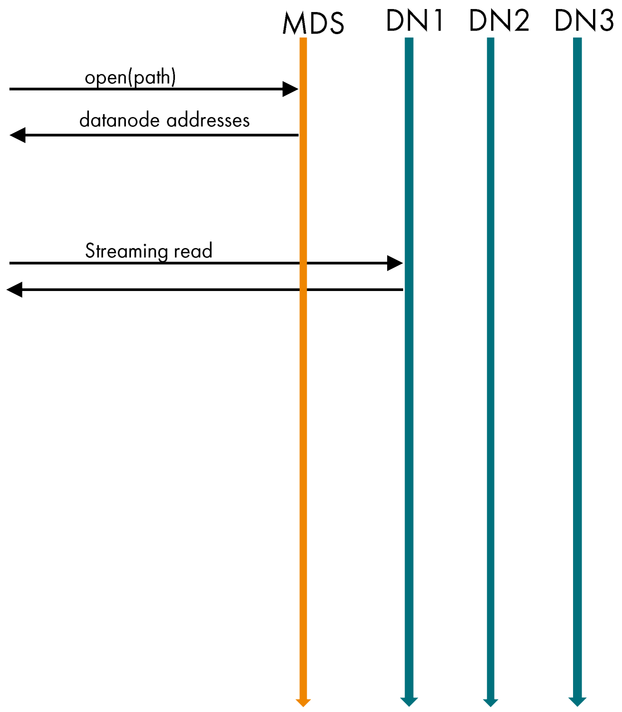
Support for Small Files

HDFS/HopsFS Write Operation



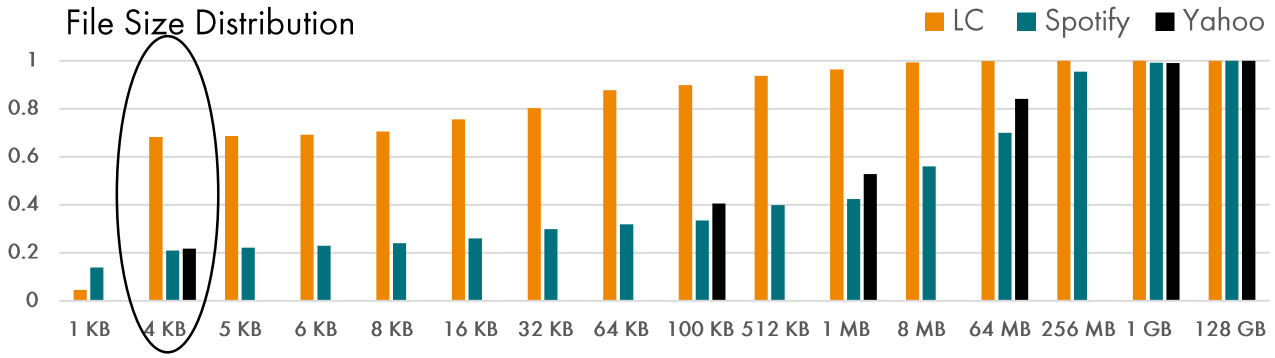
HDFS / Vanilla HopsFS

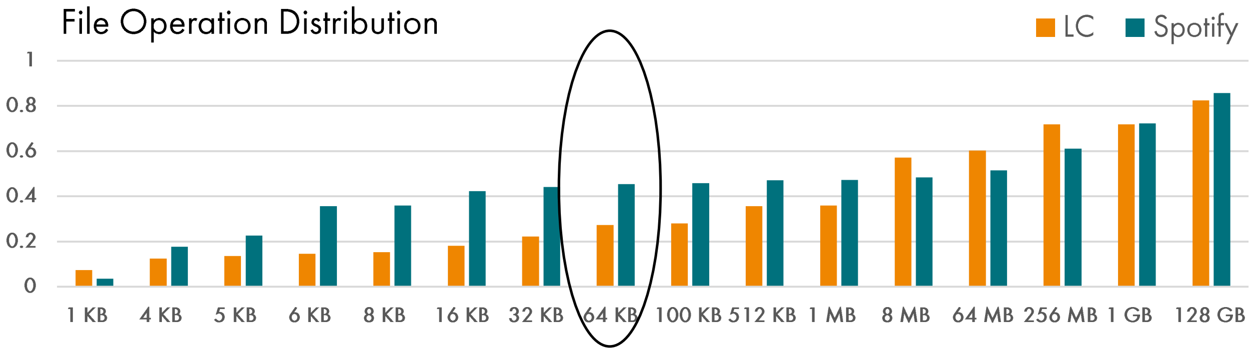
HDFS/HopsFS Read Operation



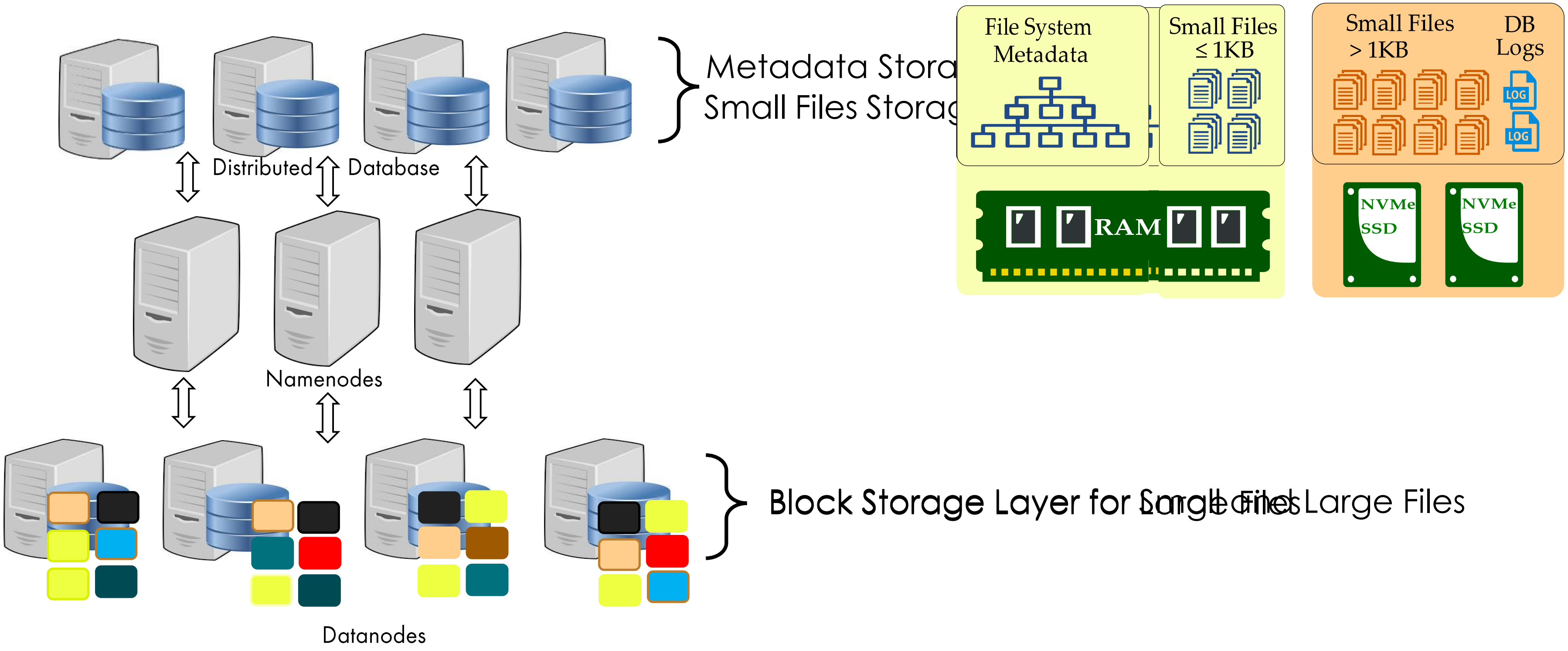
HDFS / Vanilla HopsFS

Prevalence of Small Files In Hadoop

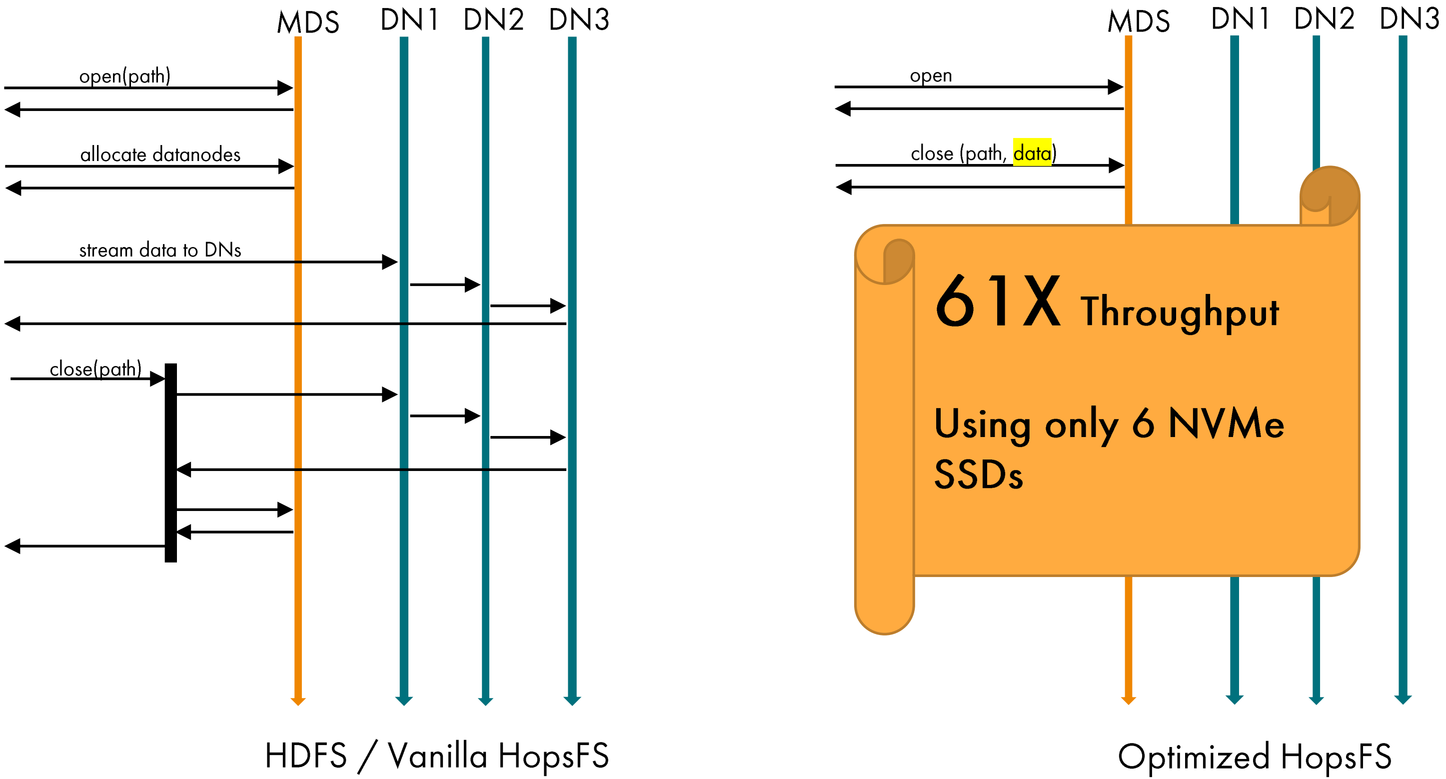




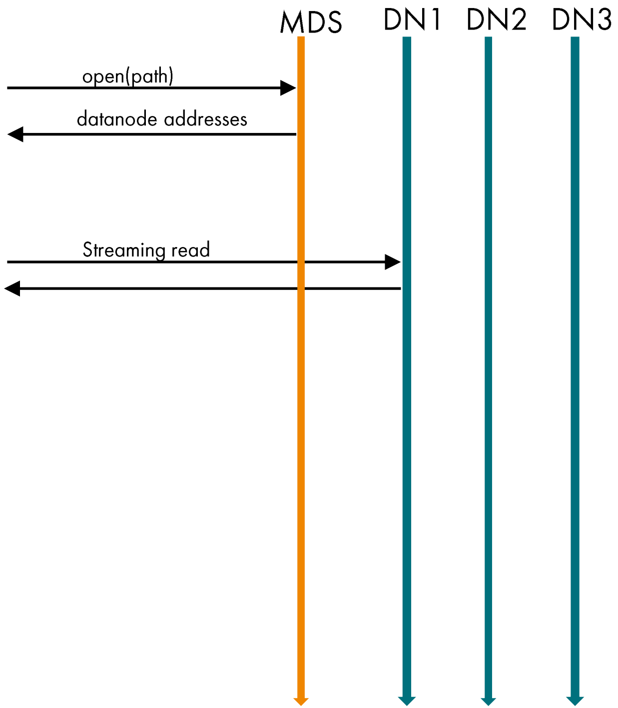
Small Files' Support in HopsFS



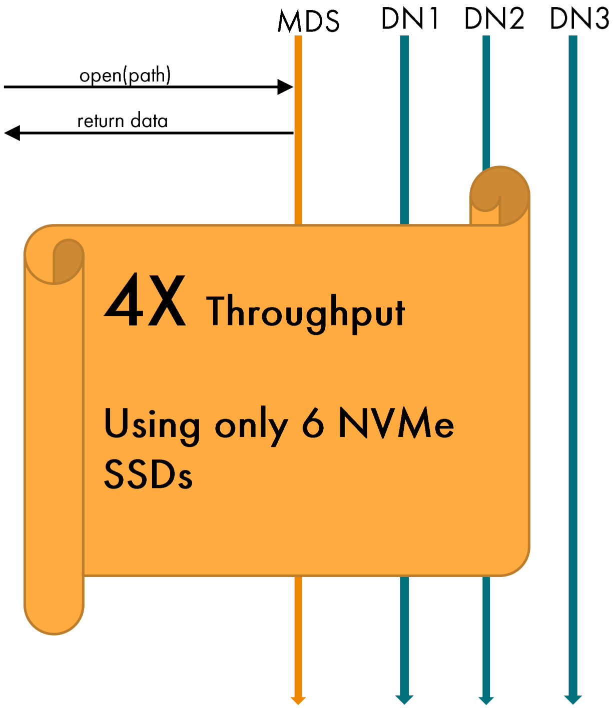
Optimizing Write Operation For Small Files



Optimizing Read Operations For Small Files



HDFS / Vanilla HopsFS



Optimized HopsFS

Questions

http://www.hops.io
http://github.com/hopshadoop
@hopshadoop

Read More

Scaling Distributed Hierarchical File Systems Using NewSQL Databases
Salman Niazi. Ph.D. Thesis. KTH Royal Institute of Technology

Scaling hierarchical file system metadata using newsql databases
S Niazi, M Ismail, S Haridi, J Dowling, S Grohsschmiedt, M Ronström
15th USENIX Conference on File and Storage Technologies (FAST 17), 89-104

Scaling HDFS to more than 1 million operations per second with HopsFS
M Ismail, S Niazi, M Ronström, S Haridi, J Dowling
2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid …

Size Matters: Improving the Performance of Small Files in Hadoop
S Niazi, M Ronström, S Haridi, J Dowling
Proceedings of the 19th International Middleware Conference, 26-39