# HDFS Architecture

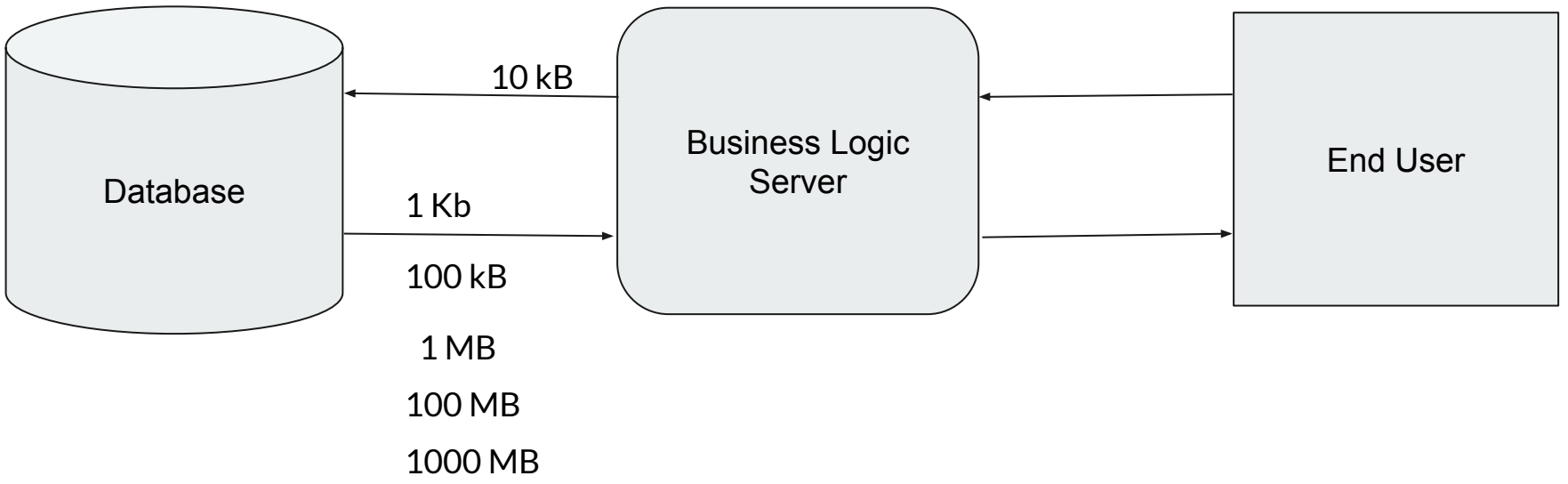
- Abhay Dandekar

# HDFS Architecture - Agenda

1. Key design assumptions and goals
2. Node types
3. FileSystem namespace
4. Federated HDFS
5. Scaling and Rebalancing
6. Data Replication
7. Rack-awareness
8. Node failure management
9. HDFS High Availability

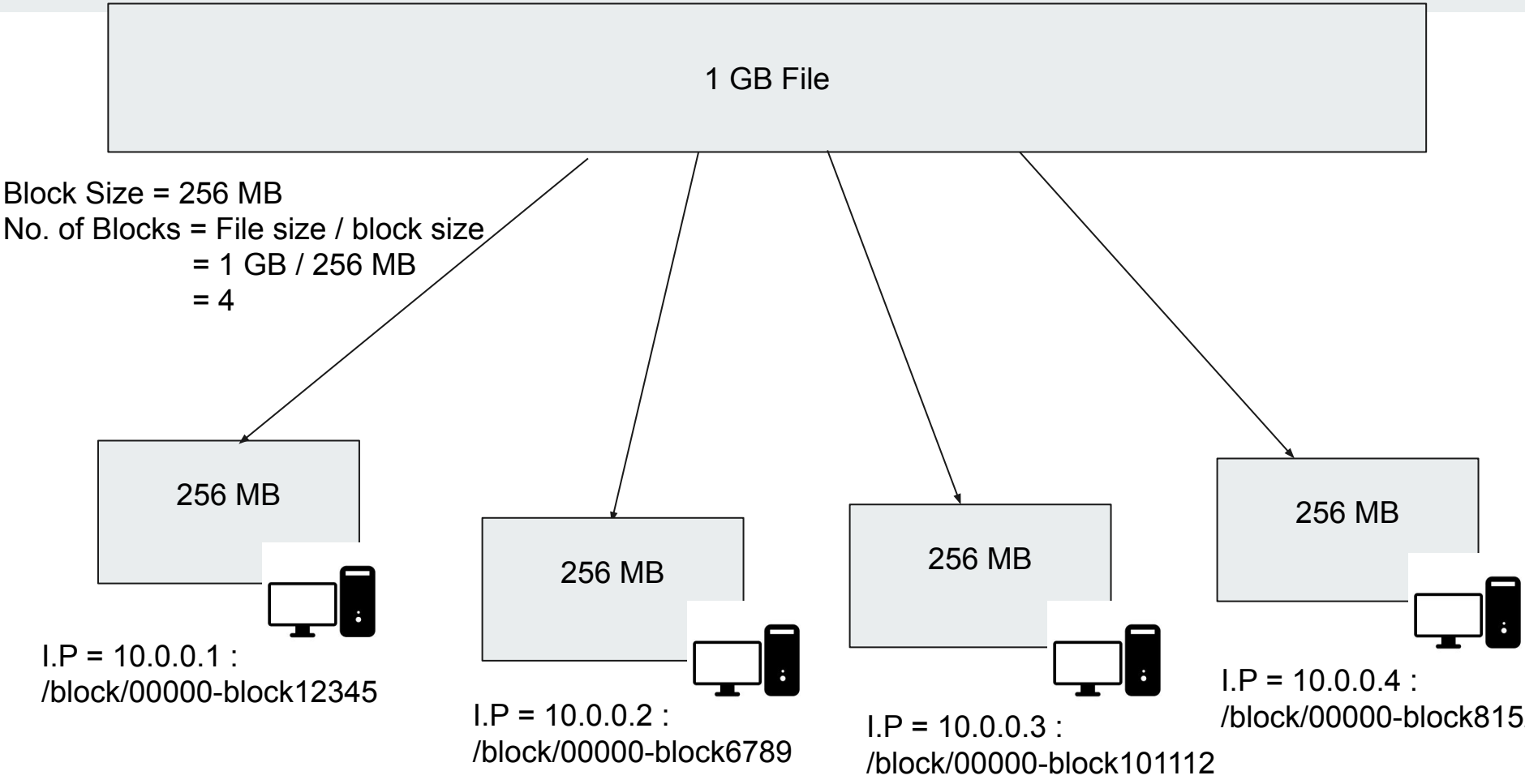
# Design Assumptions and Goals

1. Hardware Failure
2. Streaming data access
3. Large Data Sets
4. Simple Coherency Model
5. Data Locality
6. Portability across Heterogenous Hardware and Software

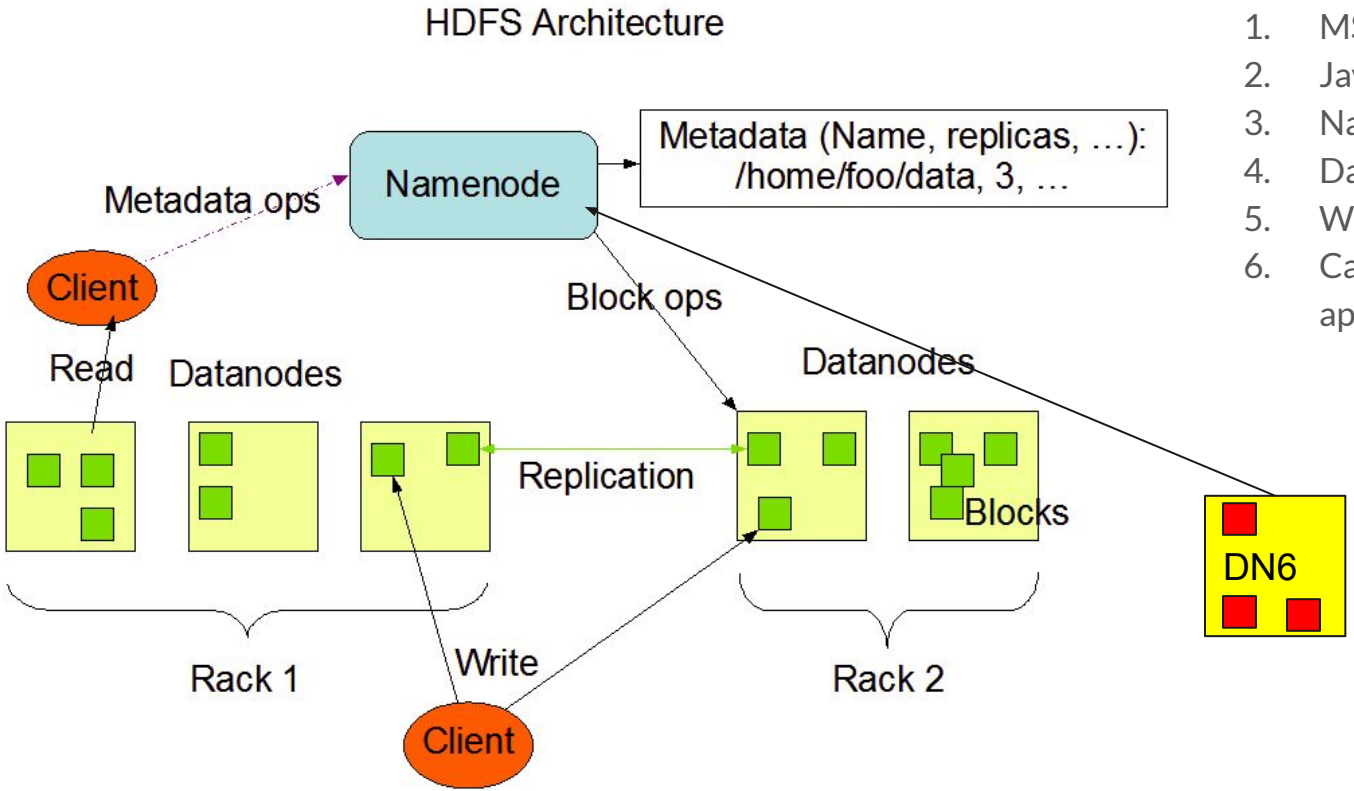

Database
10 kB
1 Kb
100 kB
1 MB
100 MB
1000 MB
Business Logic Server
End User

# Node Types

1. NameNode - Manages the namespace for HDFS
2. SecondaryNamenode - Acts as a backup for Active Namenode
3. JournalNode - Helps with the migration of edit logs
4. DataNode - Holds data blocks


1 GB File
Block Size = 256 MB
No. of Blocks = File size / block size
= 1 GB / 256 MB
= 4
256 MB
I.P = 10.0.0.1 :
/block/00000-block12345
256 MB
I.P = 10.0.0.2 :
/block/00000-block6789
256 MB
I.P = 10.0.0.3 :
/block/00000-block101112
256 MB
I.P = 10.0.0.4 :
/block/00000-block815

# What is HDFS ? - HDFS Architecture




1. MS architecture
2. Java language
3. Namenode
4. Datanode
5. WORM model
6. Cannot edit files but can append

# FileSystem Namespace

1. Traditional hierarchical file system
2. Directories allowed
3. Files allowed
4. Hard links not supported
5. Soft links not supported

# Hard link and soft link

HL -> /home/abhay/file1

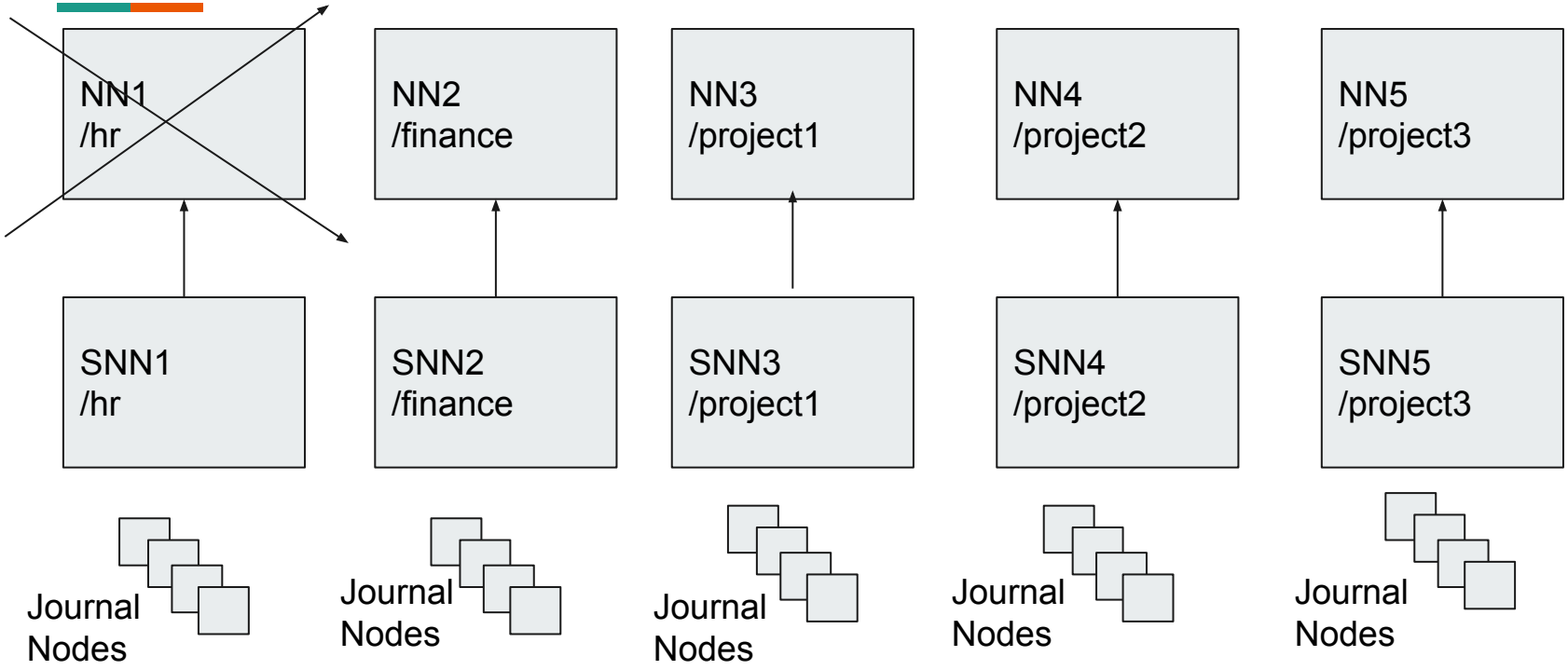
SL -> /home/abhay/file2

# Break

# Federated HDFS

1. HDFS problem with RAM size?
2. What federated HDFS provides to the cluster?
3. Benefits
    a. Namespace Scalability
    b. Performance
    c. Isolation
4. Huge space can be allocated to new data

# Federated Machines structure



# Federated HDFS (Maths)

/home/foo/data, 5

1024 bytes + 1024 bytes + 1024 bytes + 1024 bytes + 1024 bytes = 5kb

Total RAM (namenode1) = 200kb [ /input ]

Total RAM (namenode2) = 100kb [ /user ]

Total size for each file ( of 4 blocks ) = 256MB ( DATA) + Metadata = 5kb * 4 blocks = 20kb

Total files(of one block each) that can be stored = 100 kb / 5 kb = 20 files + 20 files = 40 files [ On two namenodes having 100kB of allocated memory each ]

# Scaling and ReBalancing

1. Adding a node in Datanode role via ambari
2. Or starting the command **$ hadoop-daemon.sh datanode start** with proper configurations in hdfs-site.xml

1. For rebalancing, we can execute the hdfs rebalancer : $ hdfs balancer
2. For balancing, we can set the balancing bandwidth to make it a bit faster
    a. Hdfs dfsadmin -setBalancerBandwidth <bandwidth_in_bytes_per_second>

# Data Replication

1. How is data replicated?
2. Anomalies in data-replication.
3. $ hdfs fsck /
    a. Over-replicated blocks - Blocks greater than replication setting
    b. Under-replicated blocks - Blocks lesser than replication setting
    c. Misreplicated blocks - Blocks with replication satisfied but not on correct nodes
    d. Corrupt blocks - Block that have gone corrupt
    e. Missing replicas - Blocks with no replicas anywhere in the cluster

10.0.1.1 - 255

10.0.2.1 - 255

10.0.3.1 - 255

10.0.4.1 - 255

10.0.5.1 - 255

# Rack-Awareness

1. What do you mean by rack-aware?
2. What makes Hadoop components become rack-aware?

# Node failure management

1. Remove a node
2. Add a node
3. Rebalance HDFS (Share the command)

# HDFS High Availability

1. HA can be achieved in two ways
    a. Have a quorum journal manager (more than 3 journal nodes)
    b. Have a conventional shared storage (NFS shared drive)
2. Two separate machines configured as namenodes
3. One is active at a time, other is in stand-by.
4. Whenever, one fails, the other machine takes over.
5. To test, one can kill the active using kill -9 command
6. Failover should happen

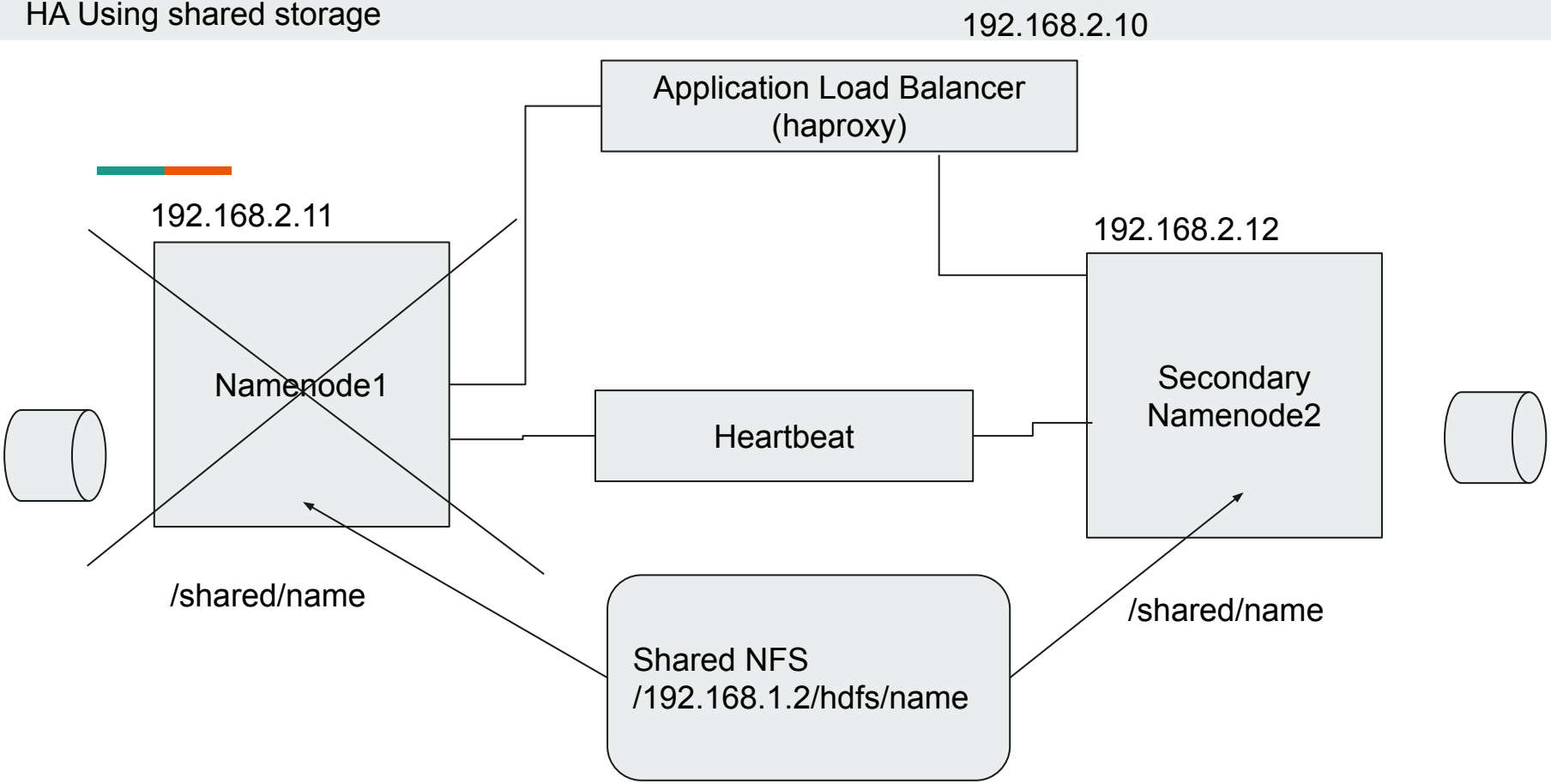
# HDFS File checks

1. $ hdfs fsck /
2. Types of block replication status
    a. Over replicated blocks
    b. Under replicated blocks
    c. Misreplicated blocks
    d. Corrupt blocks
    e. Missing replicas
3. DFS blockscanner
    a. http://<YOUR_DATA_NODE>:9864/blockScannerReport

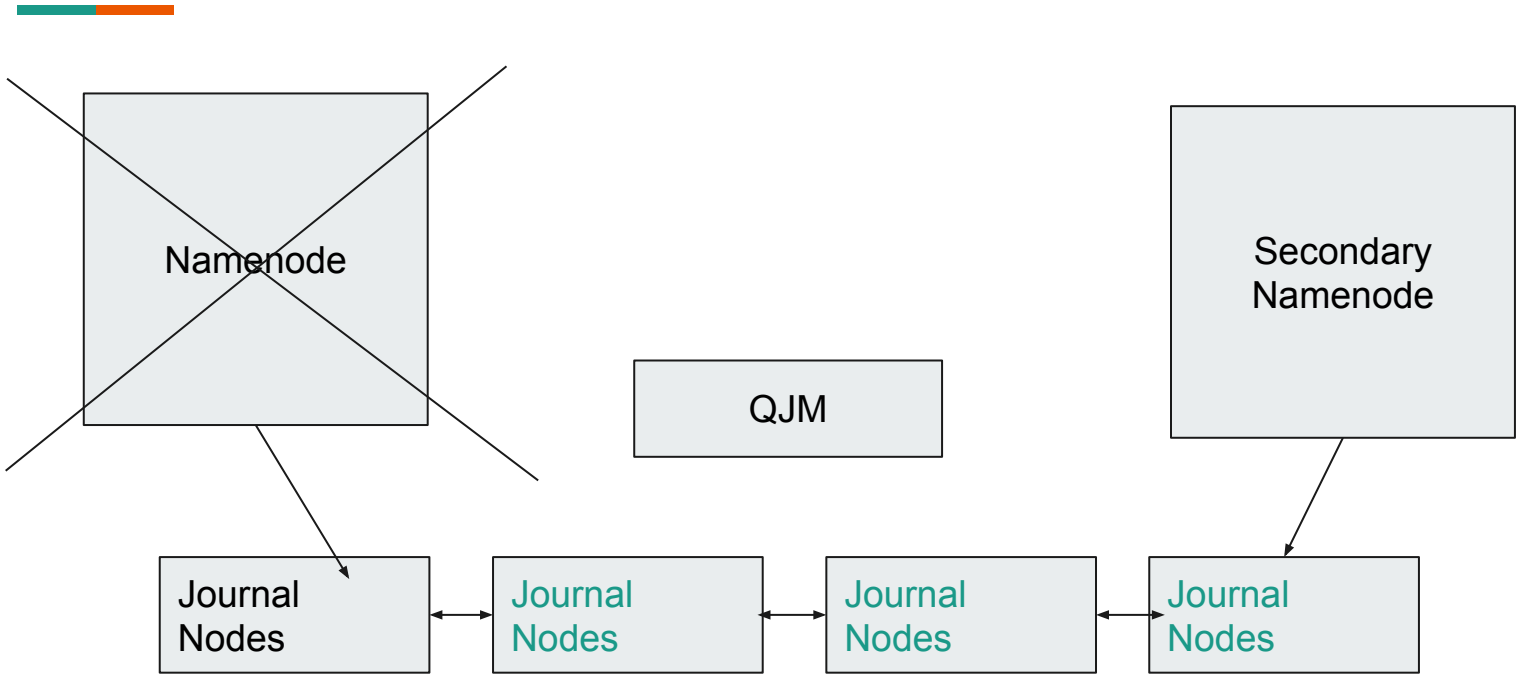
# Thank you

See you in Lab session :)

# HA Using shared storage



# HA Using QJM


Namenode
QJM
Secondary
Namenode
Journal
Nodes
Journal
Nodes
Journal
Nodes
Journal
Nodes