# BITS Pilani presentation


BITS Pilani
Pilani Campus

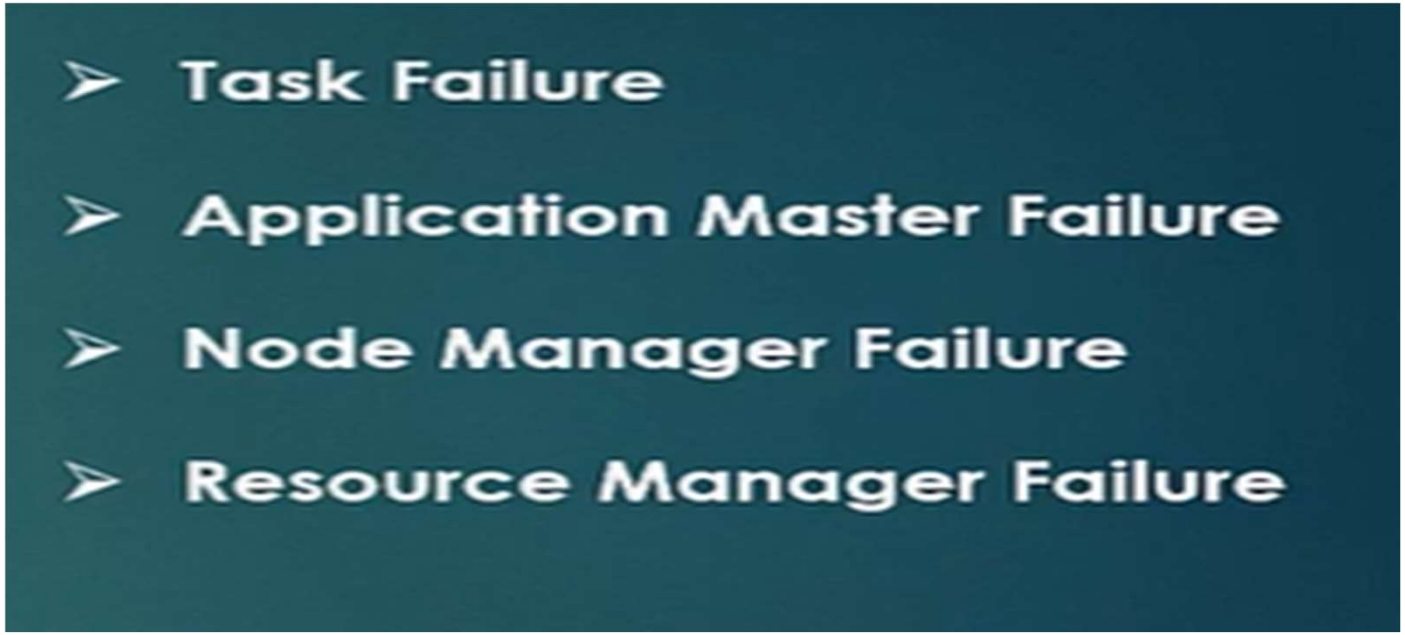
Dr. Vivek V. Jog
Dept. Of Computer Engineering

innovate achieve lead

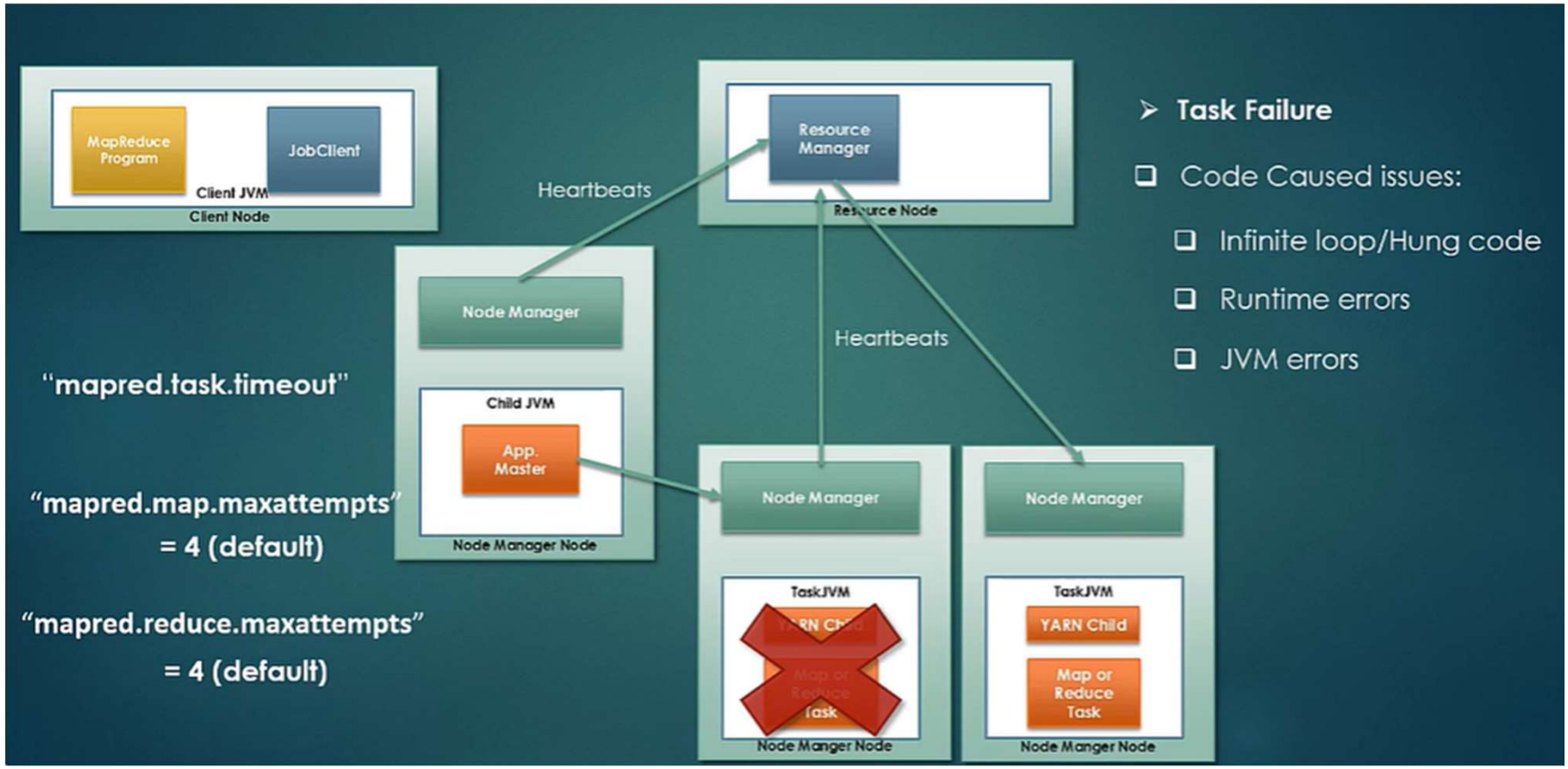
BITS Pilani
Pilani Campus

# Big Data Systems (S1-24_CCZG522)
# **Lecture No.7**

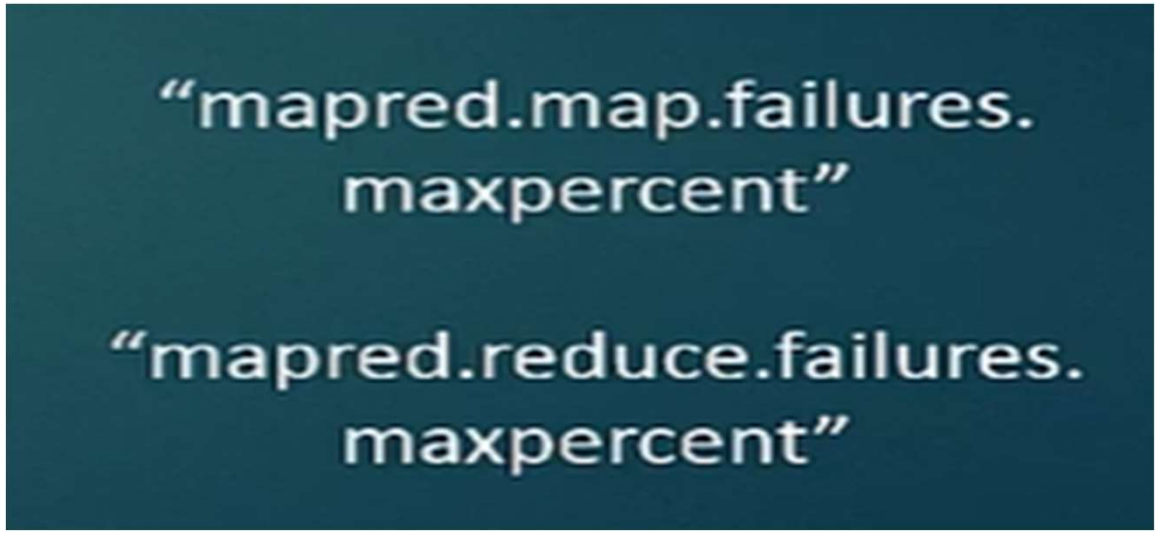
# YARN Failure Cases



# Task Failure



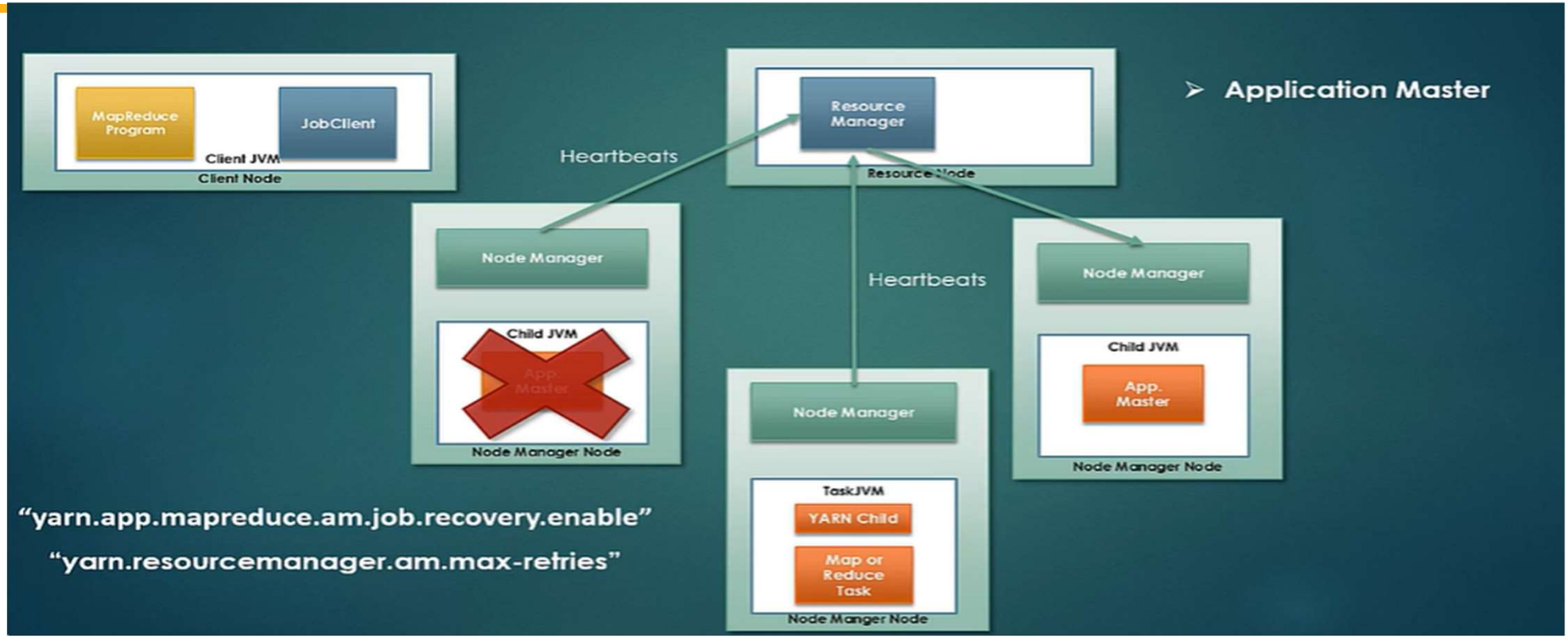

**After Exhausting all attempts …Complete job will be marked as failed**




**Failure of one or two tasks cannot be marked as complete Job Failure**

**Above “mapred.map” and “mapred.reduce” properties are Used to Decide Expectable Percentage of Failure before deciding that Job has Failed**
**In case of task failure Resource Manager will start the Application Manager in a new container**

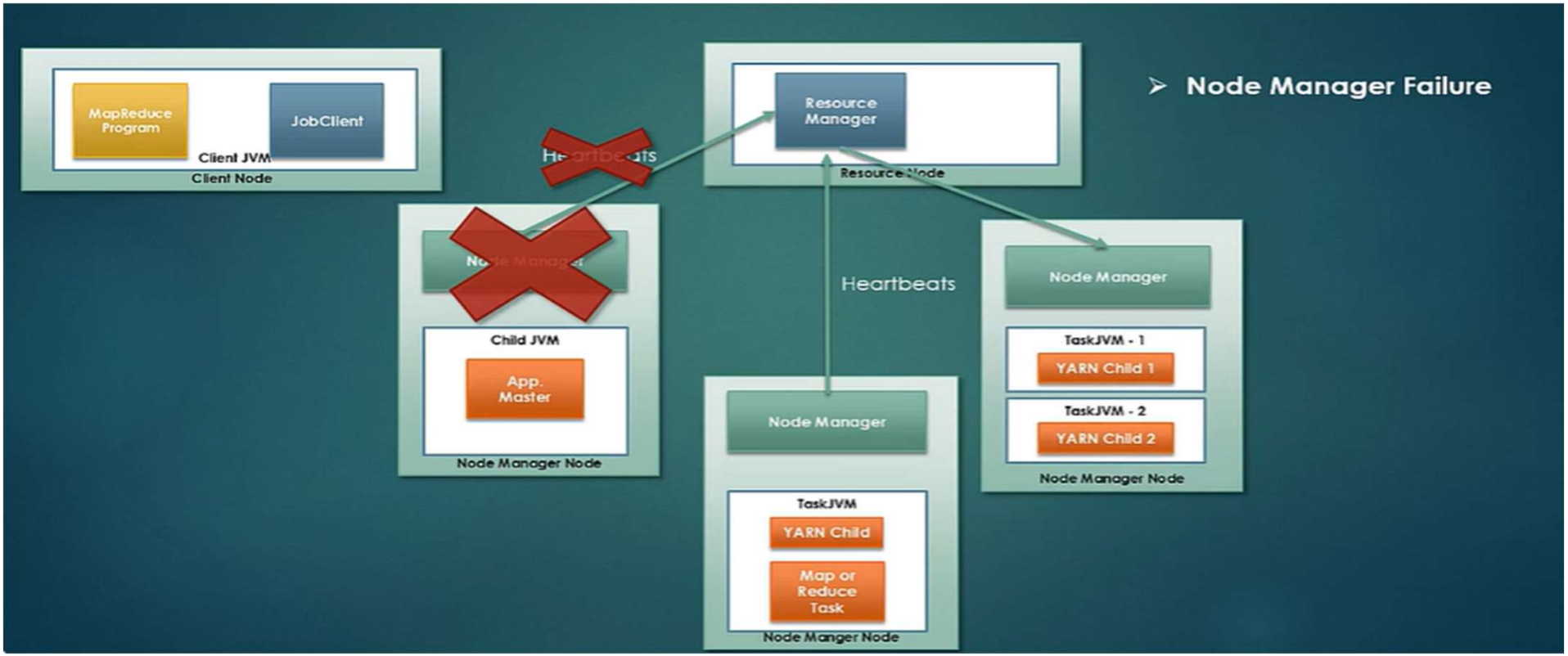
# App Master Failure




Task that has run under Application Master need not be resubmitted , it can be recovered. Provided property ..would be needed to be set to enable.

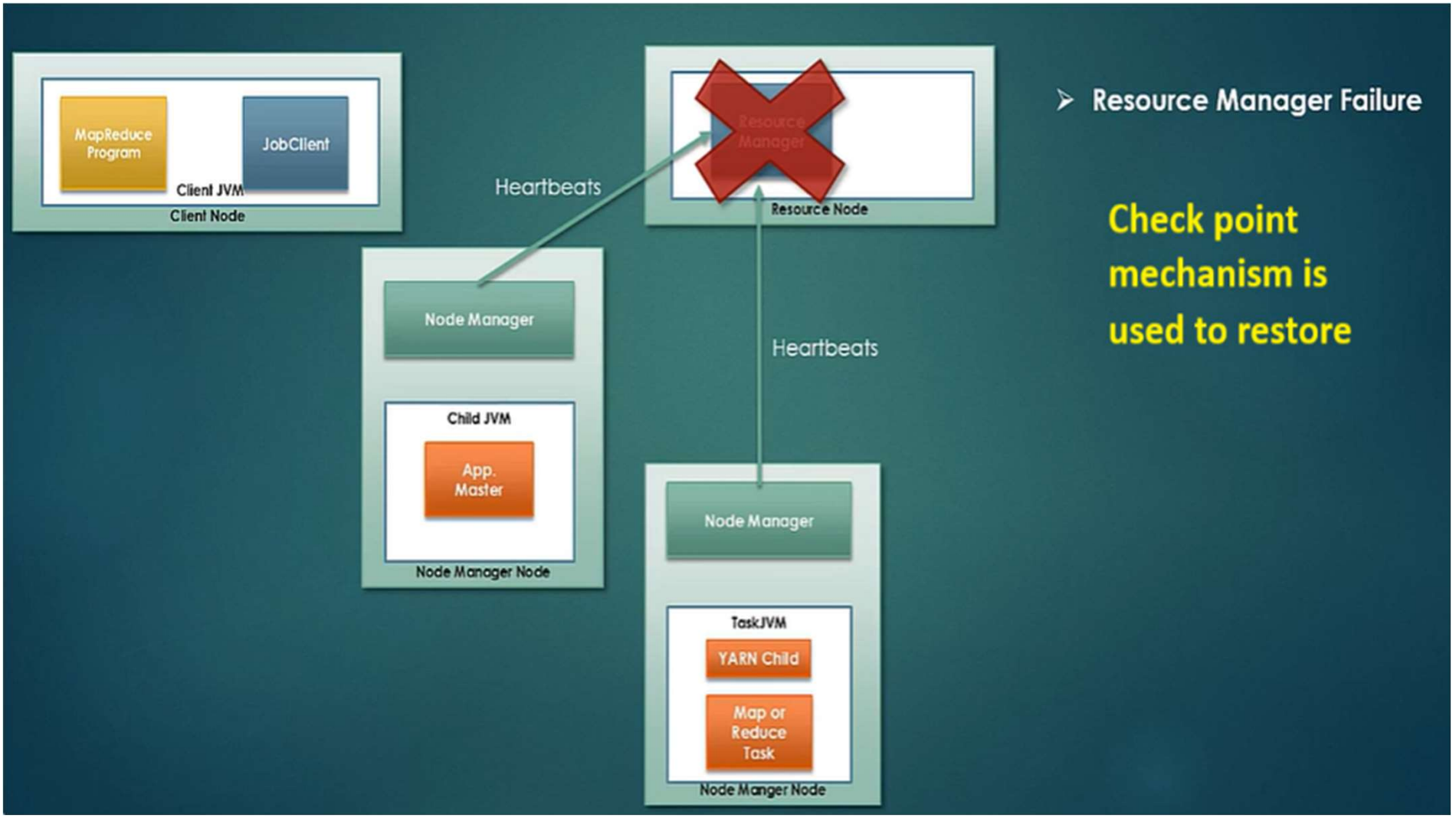
When Application master fail, Resource master stops getting heart beats from Application master. Number of attempts of Application Master is determined by the property "**yarn.resourcemanager.am.max-retries**"

# Node Manager Failure




If Application Master was running under the failed Node Manager than steps described In Application Master are followed. All the remaining tasks are re spawned on new Node Manager. If task under specific Node Manager fail often and crosses the threshold then It is remode from available pool and is **Black Listed**

# Resource Manager Failure



# Hadoop Cluster Capacity Planning



WHAT IS HADOOP CLUSTER?
FACTORS DECIDING HADOOP CLUSTER
HARDWARE REQUIREMENTS
OPERATING SYSTEM REQUIREMENTS

# What is a Hadoop cluster?

# Factors influencing cluster size

### Volume of Data

It is important for a Hadoop Admin to know about the volume of Data he needs to deal with and accordingly plan, organize, and set up the Hadoop Cluster with the appropriate **number of nodes** for an Efficient Data Management

### Data Storage

The obtained data is encrypted and compressed using various **Data Encryption** and **Data Compression** algorithms so that the data security is achieved, and the space consumed to save the data is as minimal as possible.

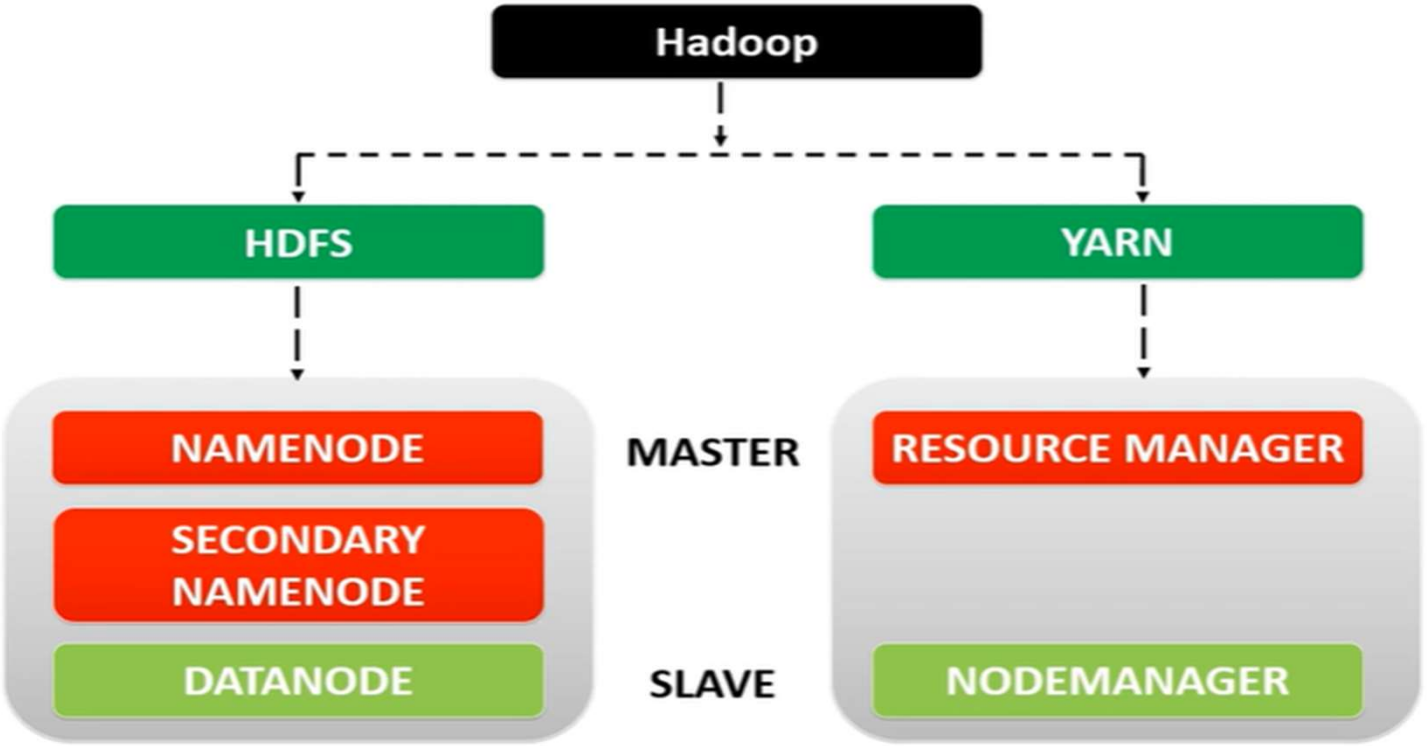
### Data Retention

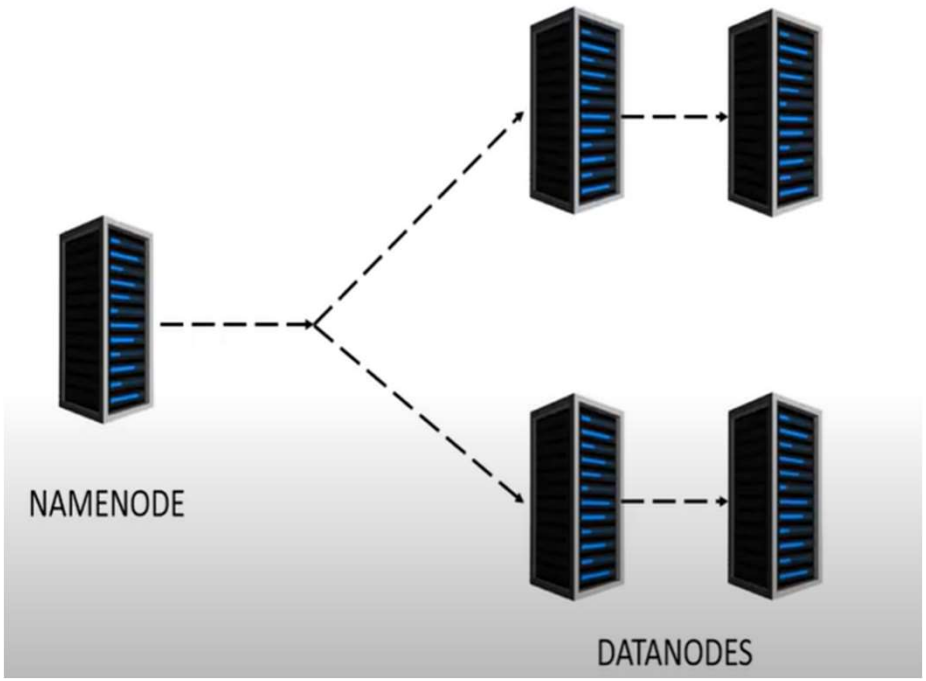
**Data Retention** is a process where the user gets to remove outdated, invalid, and unnecessary data from the Hadoop Storage to save space and improve cluster computation speeds.

### Types of Work-Loads

This factor is purely **performance-oriented**. All this factor deals with is the performance of the cluster. the Workload on the processor can be classified into three types. Intensive, normal, and low.

# H/W Requirements



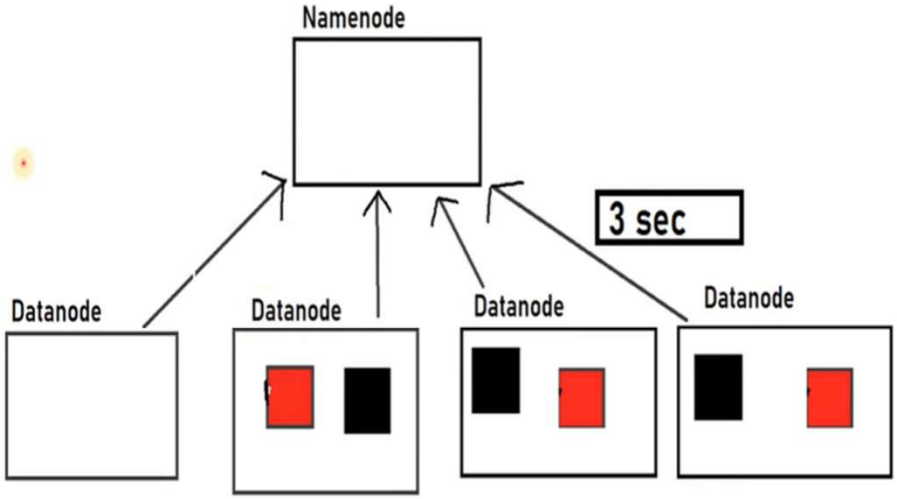



## Name Node

- **Master daemon** manages the Data Nodes.
- Records the metadata of all the files
- Receives Heartbeat and a block report from Data Nodes.

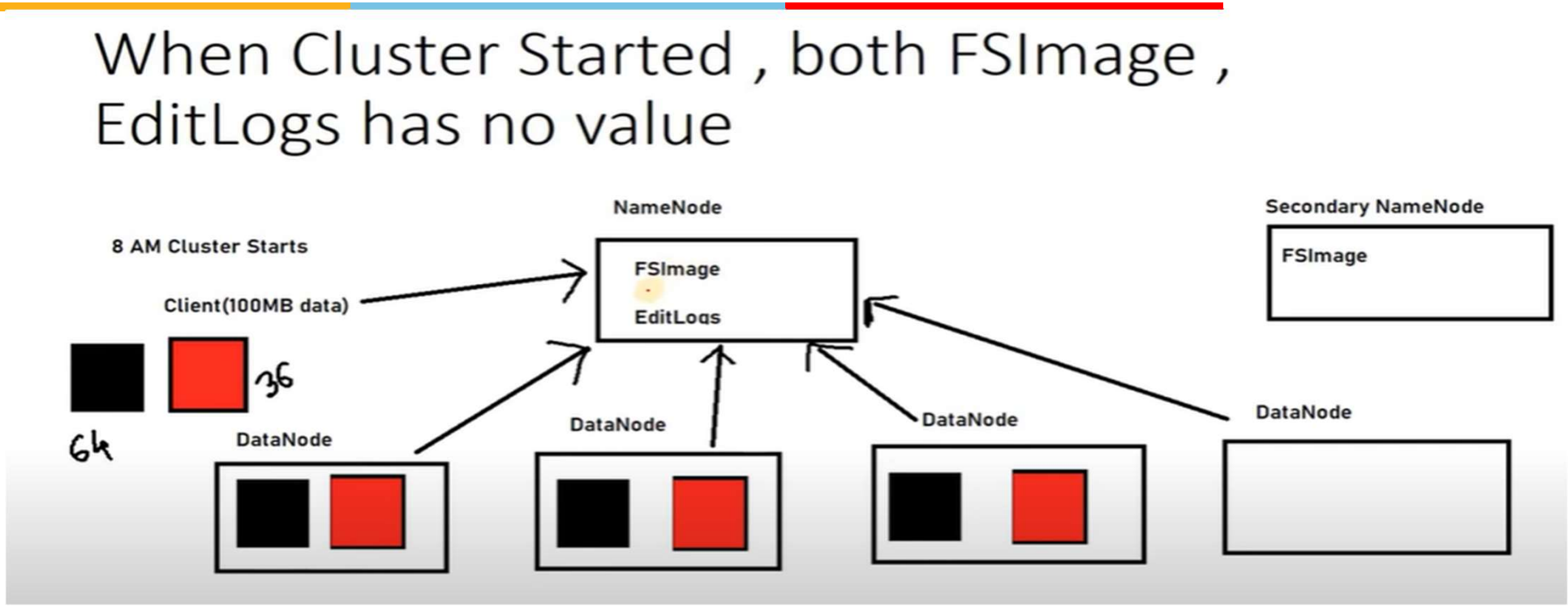
## Data Node

- **Slave daemons** runs on slave machine
- The actual data is stored on Data Nodes
- Responsible for serving read & write requests.




If NameNode fails entire HDFS file system is lost→**NameNode is the single point of failure in HDFS**

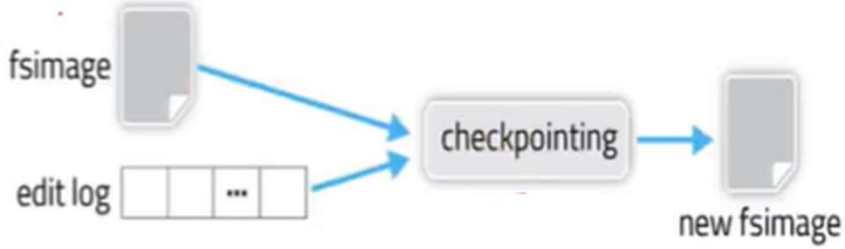
# Secondary NN and Back Up NN

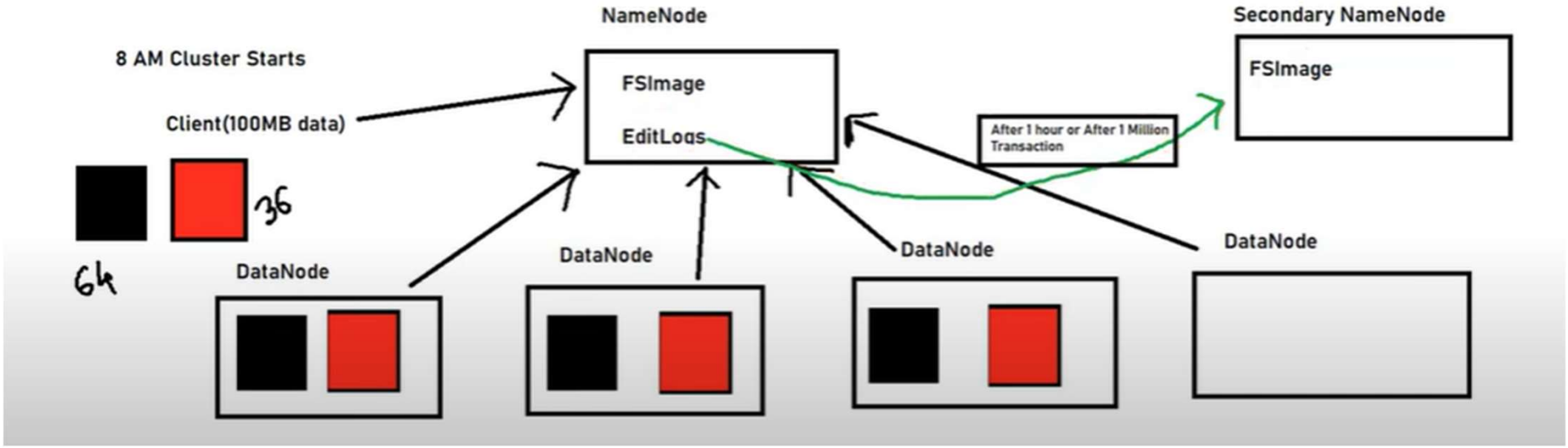



The metadata information related to the client request will be written down in the FSImage & EditLogs of NameNode

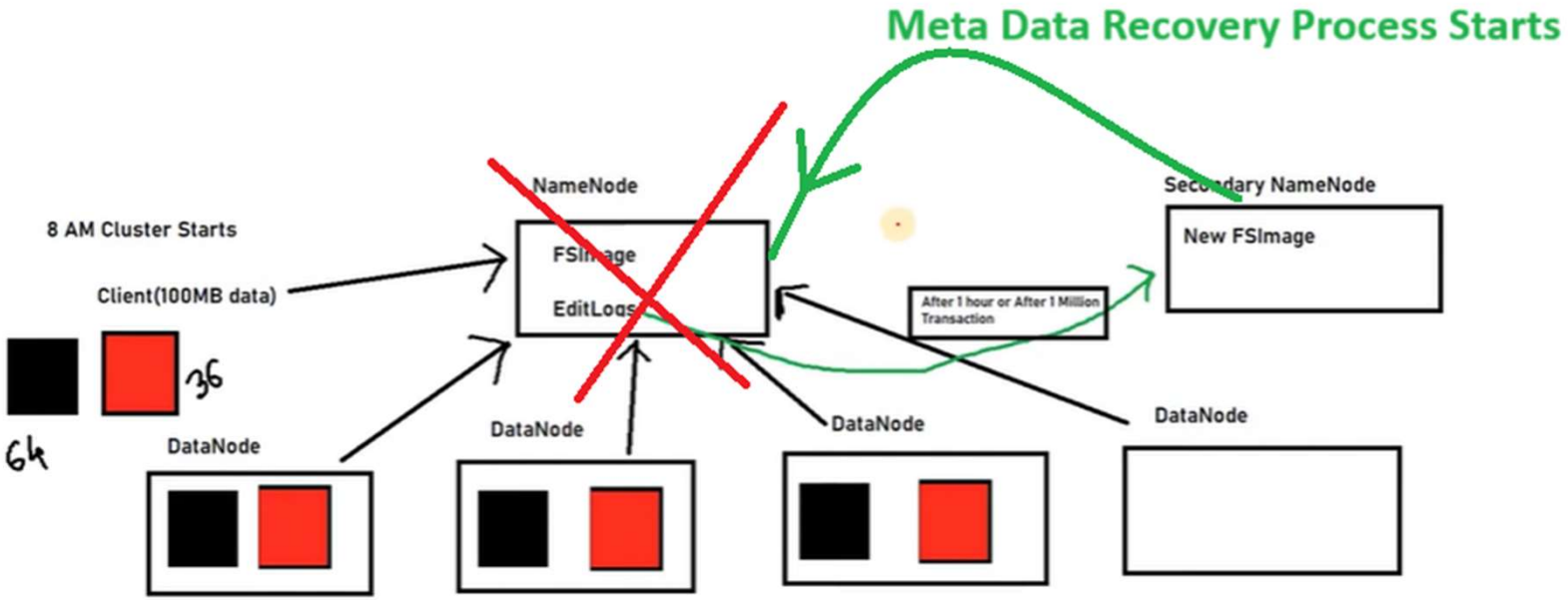
# Checkpointing….Updates

After 1 Hour or after 1 M transaction whatever happens first



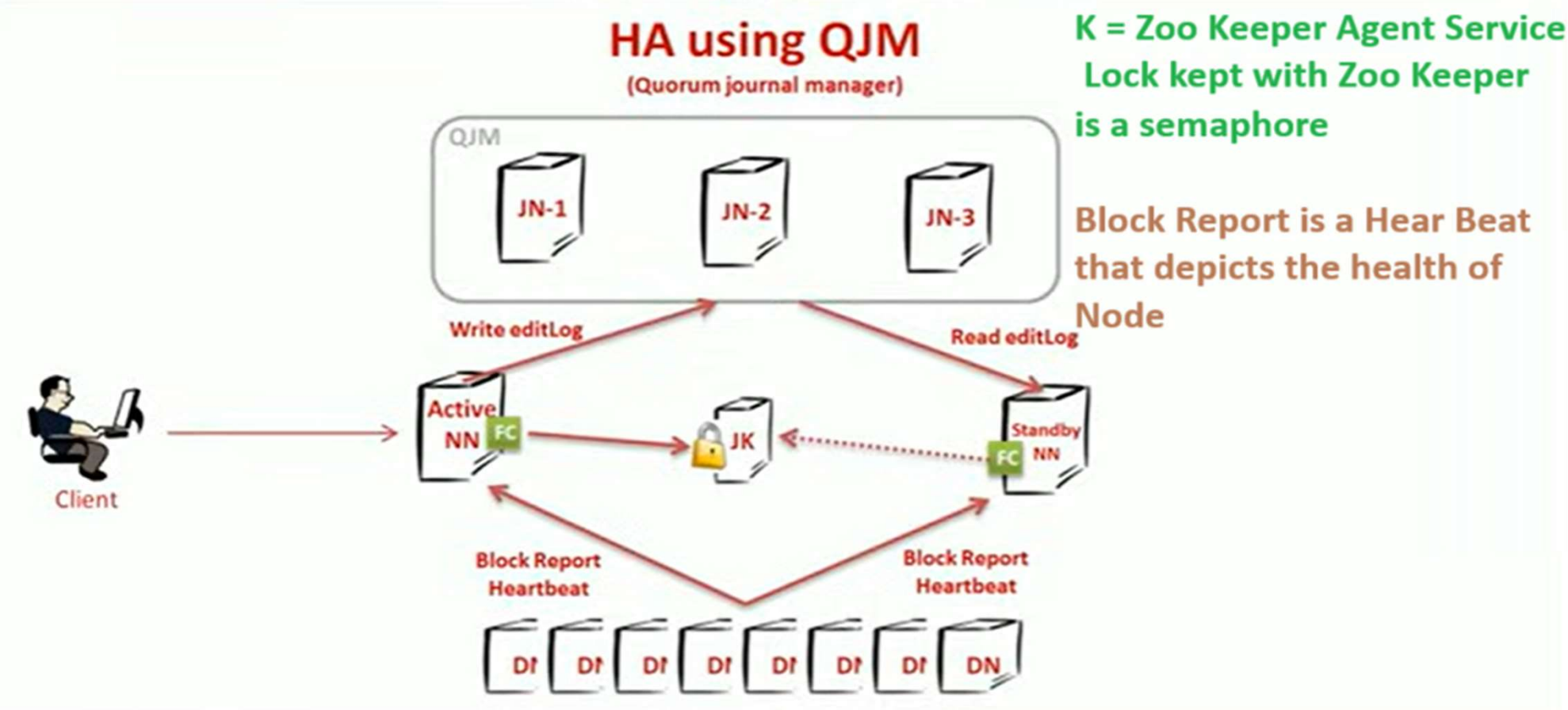
# Safe Mode – Read Only Mode




- Safemode in Apache Hadoop is a maintenance state of NameNode, during which NameNode doesn't allow any modifications to the file system.
- In Safemode, HDFS cluster is in read-only and doesn't replicate or delete Data Blocks.

# Back Up NN better Than Sec NN

# Back Up Node




**Thus …High Availability Hadoop Cluster Does not Need Secondary Name Node**
**Standby node also performs the Checkpointing operation.**

# H/W requirements for Name node and Job Tracker



| Component | Requirement |
|---|---|
| Operating System | 1 Terabyte Harddisk Space |
| FS-Image | 2 Terabyte Harddisk Space |
| Other Softwares(Zookeeper) | 1 Terabyte Harddisk Space |
| Processor | Octa-Core Processor 2.5 GHz |
| RAM | 128 GB |
| Intenet | 10 GBPS |

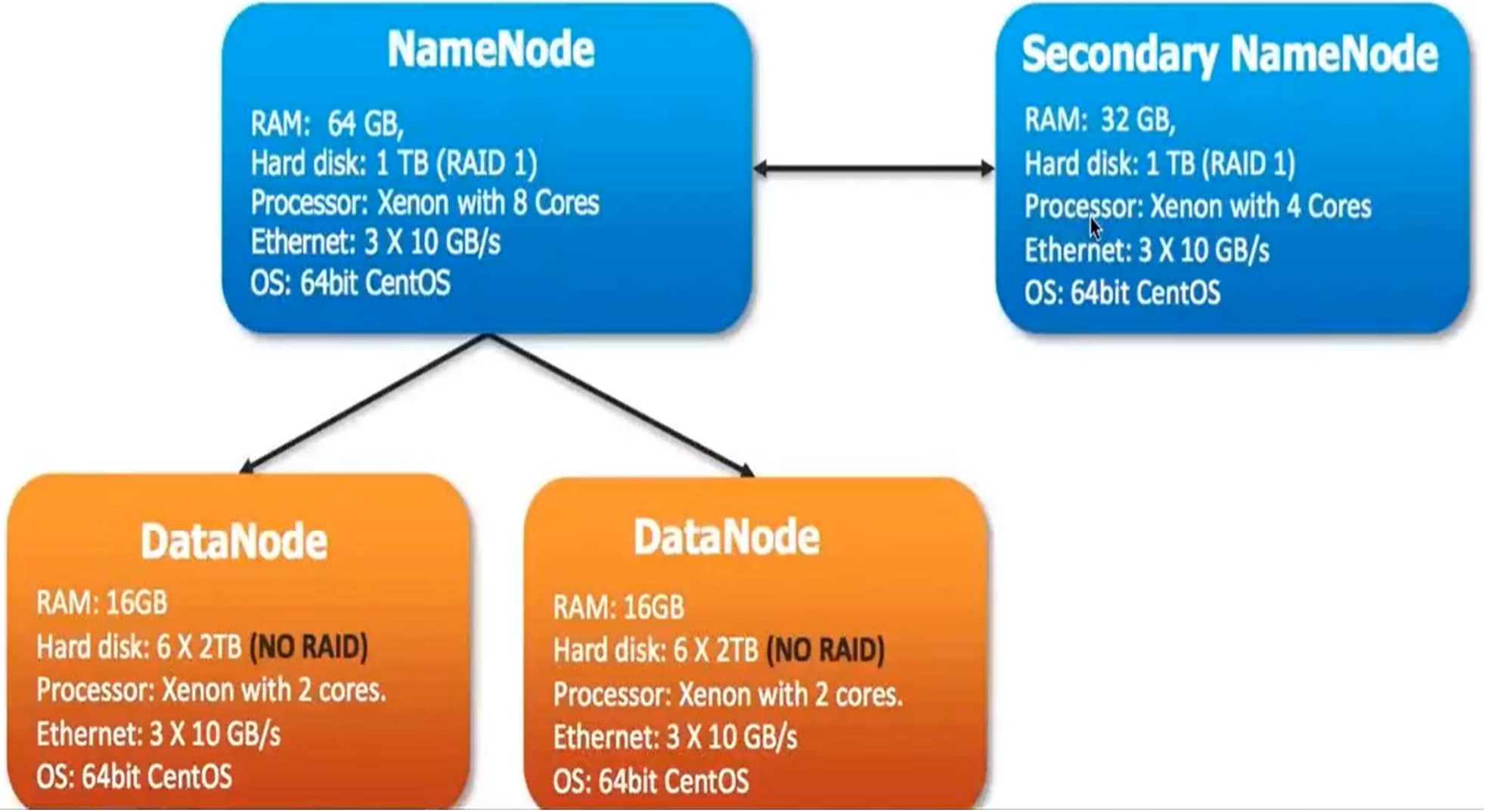
# H/W req. for Data Node and Task trackers



| | |
|---|---|
| Number of Nodes | 24 nodes(4 Terabytes each) |
| Processor | Octa-Core Processor 2.5 GHz |
| RAM | 128 GB |
| Internet | 10 GBPS |

# S/W Req. –Suggested OS

# About Hadoop Cluster – A Typical Cluster Example



# Few Facts about File Blocks



```
Two core components:
        1. Hdfs -       hadoop distributed file system
        2. Mapreduce -  processing


HDFS:
        1. Namenode
        2. Datanode

1 File -> 1 GB

BS => BLOCK SIZE
bs => 256 mb

total no. of blocks = 1 GB / 256 MB => 4 blocks

BS => CONFIGURABLE -> DEFAULT = 64 MB

1 FILE = 1GB
BS = 64 MB
TOTAL NO. OF BLOCKS = 1GB/64MB = 16 blocks

REPLICATION FACTOR => RF        => configurable

1 GB => BS => 64M, RF = 2

total amount of size = 1 gb x 2 = 2 GB / 64 MB = 32 blocks

=============
```

**Considering RF =1**

```
FILE = 400MB
BS = 64 MB
TOTAL NO. OF BLOCKS = 7 blocks
        400 mb / 64 mb = 6.25 blocks
                         6 blocks = 64 mb = 384 mb
                         1 block = 16 mb -> next block
```

```
==========


file = 702 mb
bs = 128 mb
rf = 3

total no. of blocks = ?

                18

                702 / 128 => 5.4
                5 blocks + 1 block


size of each block = ?

                5 blocks = 128 mb
                1 block = 62 mb

RF = 3

                5 x 3   = 15
                1 x 3   = 3
                        ____

                        18 blocks
```

**With Replication Factor = 3**

# Observation



1 GB

**Large Data Set**

BS = 256 MB
RF = 3

TOTAL NO. OF BLOCKS => 12 BLOCKS
1GB / 256 MB = 4 X RF (3) = 12 BLOCKS

1 GB
1024 FILES = >
SIZE OF EACH FILE = 1MB

**Small Data Set**

BS = 256 MB
RF = 3

TOTAL NO. OF BLOCKS => 1024 X 3 = 3072 BLOCKS

**If the Size of the File is less than the Size of the Block**
**Than size of the file becomes the size of the Block -- IMPORTANT**

# How Big Can Be the Size of My Cluster?



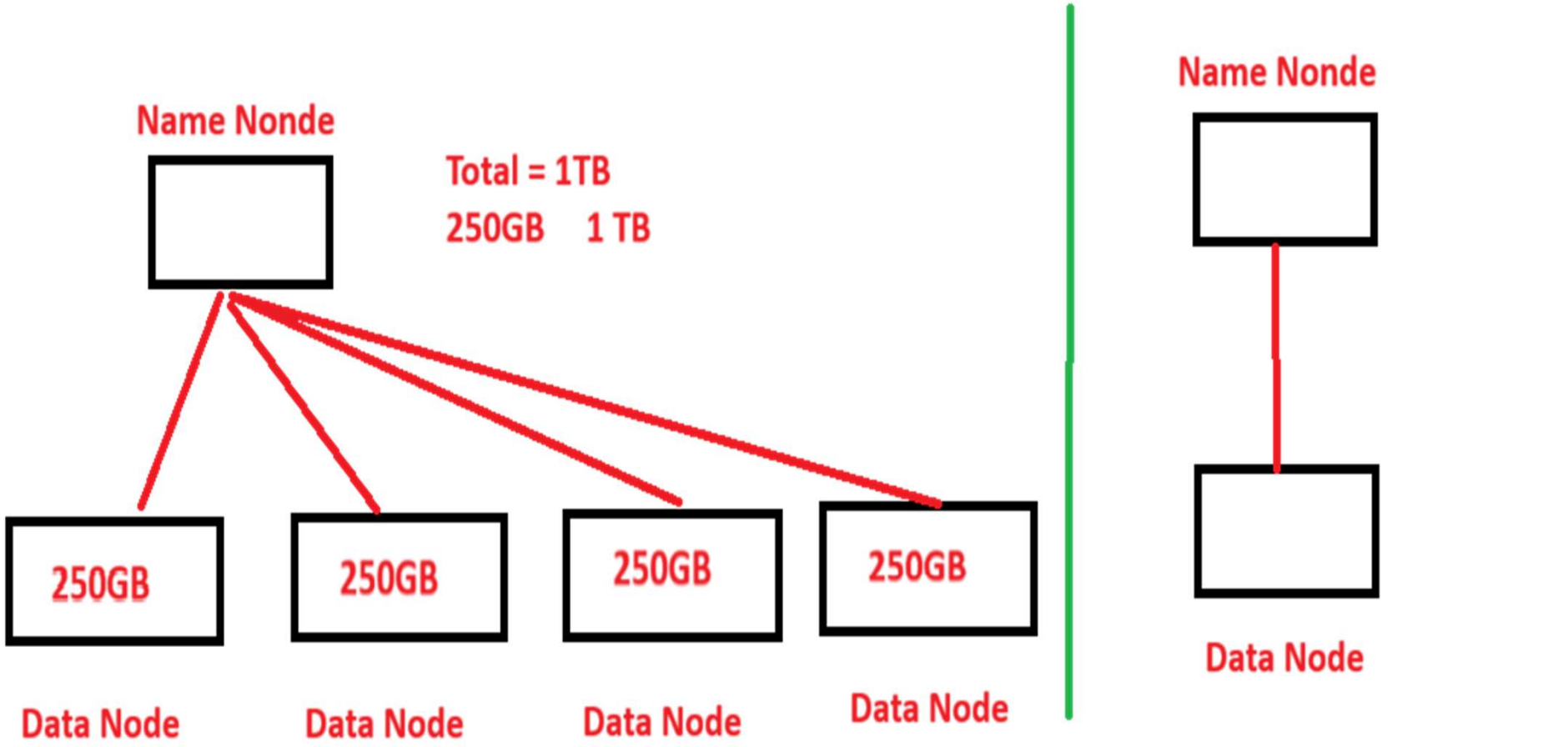
Everything in Hadoop can be viewed as an Object
(Hadoop is written in JAVA)
So Everything . File, Directory as well as Block is an Object
A File of 128 MB having Block Size of 64mb

---

So this file will split in to 2 Block + One More will be for file name. So 2 Objects + 1 Object = **3 Objects**
Every Object occupy 150 Bytes hence totally **150 X 3 = 450** Bytes of memory will be occupies **IN THE NAME NODE**

**If My Block Size is 128mb than total memory requirement Reduces to 2 Blocks ….So it is not advised to have smaller size files**
**So…Total Size of Memory in Name Node divide by File size required to be stored in Name node gives us count of total Number of Objects that Name node can accommodate**

# Name Node Size Matters




Whatever is the total capacity ..Number of Object that Name Node can accommodate is same

# Sample Calculation



If Name Node memory = 64 GB
Lets consider some memory space for NN OS
Lets consider some memory space for NN Daemon

---

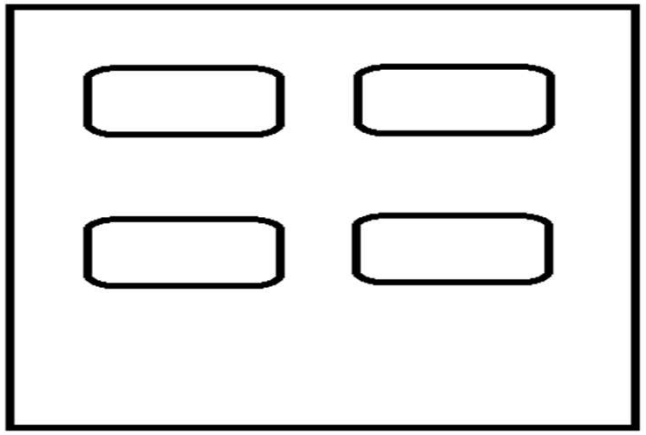
Remaining memory = may be 50 GB
Consider Block size = 64mb
Consider File size = 128mb
So total object size memory required on Name Node
Is 2 Block + 1 Block = 3 Block = 3 X 150 Bytes = 450 Bytes
Now total object that our NN can accommodate
= 50gb/450 bytes
So we can have these many Files in the cluster
If we divide 50gb/150 bytes ..we will get number of Objects

# Name Node - Multithreaded Daemon

So we need to increase the HANDLER COUNT if the cluster is of bigger size

Max allowed = 60

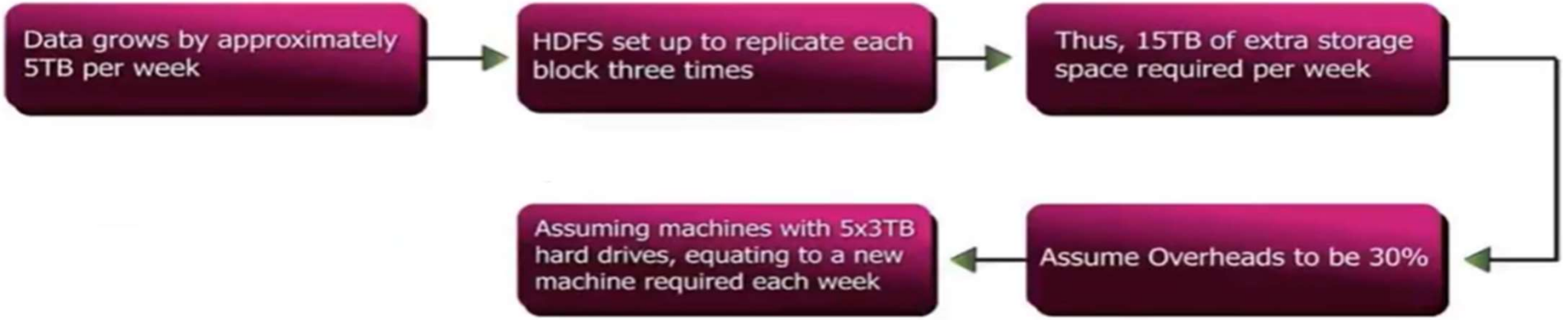
So it is better to have multiple CPU CORE which will run multiple threads and achieve parallelism.



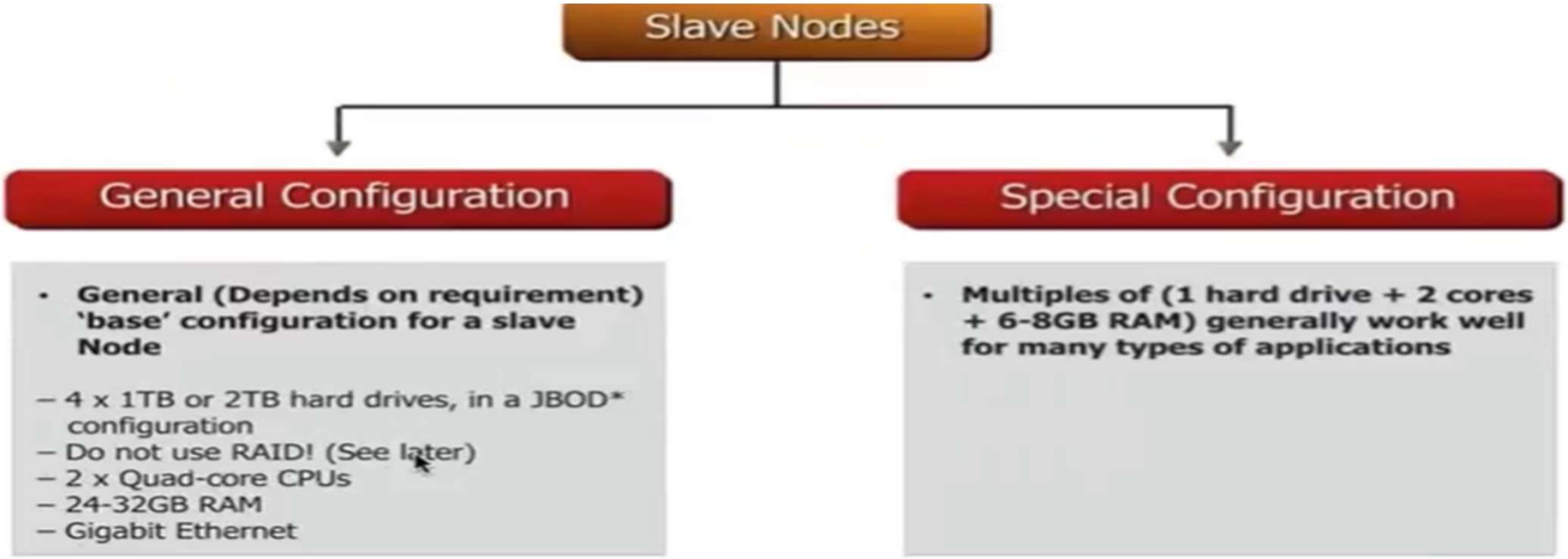
Quad Core CPU Each having Virtualization as well as Hyper Threading Facility

With 2 Threads So 4 Cores means 8 threads

# Data Node Statistics




**We Will Need On e New Machine Every Week**



# Default Mapper & Reducer Size

If Data Node had 4gb of Memory
(considering no memory used for OS & Daemon)

Mapper default size = 1gb
Reducer default size = 1.5gb

Hence we can accommodate 4 mappers
OR
1 mapper & 2 Reducers

**IMPORTANT :** Make sure that there is no swapping
Swapping of Processing (Mapper or Reducer) will reduce the
Performance of the Data node

If we have 4gb of memory and 4 mappers even if Data node is capable
To accommodate the processing if CPU has only one CORE than only one
Processing logic ( One mapper at a time) can be executed.
Hence it is IMPORTANT to have Multi-Core CPU

# How Many Tasks Can I Run on My CPU?

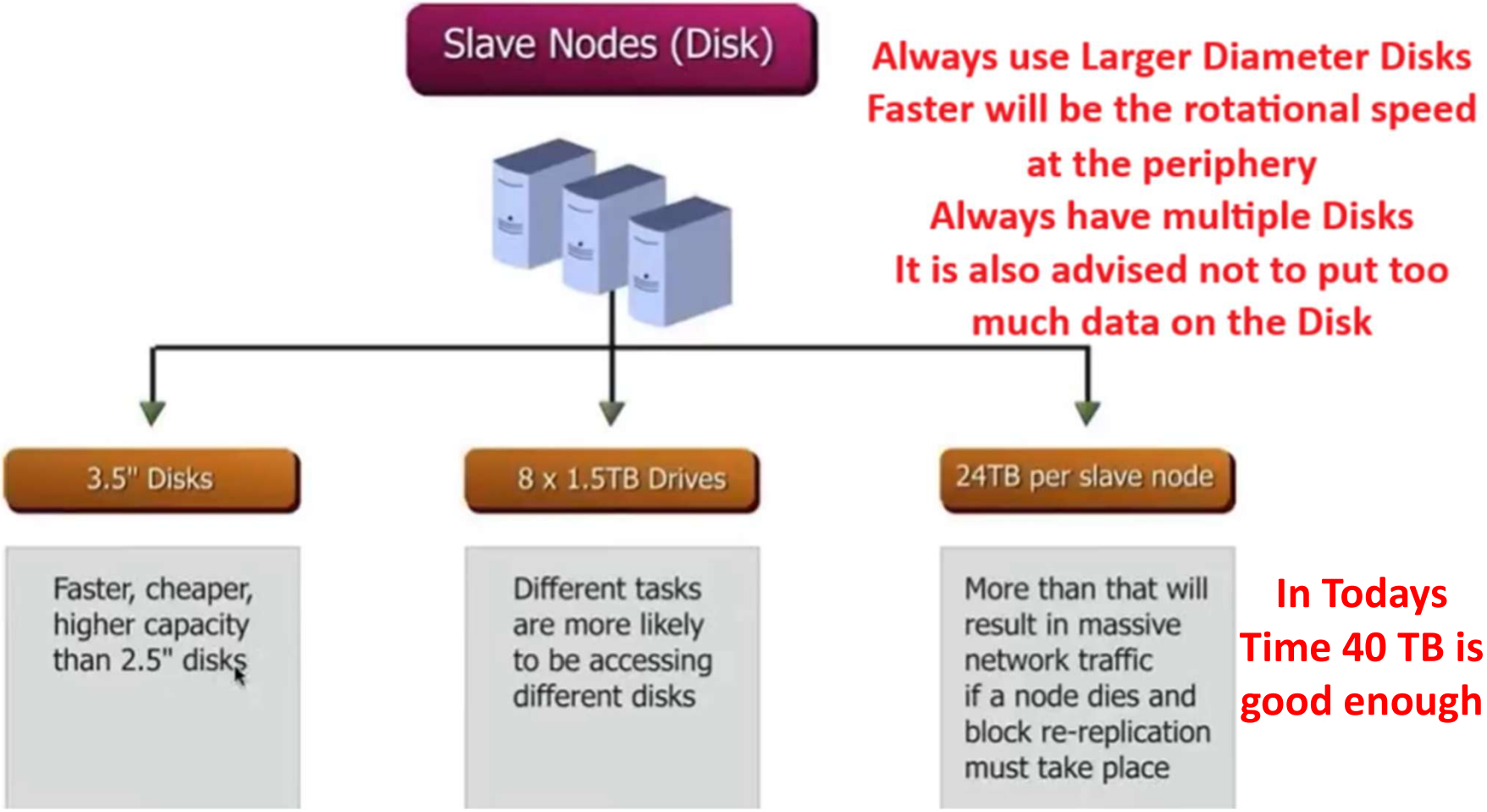


This is give as = 1.5 X CORES

Example = If you have 2 core CPU than we can run 3 tasks
Per CPU since 1.5 X 2 = 3

Here Two tasks can run on 2 CPU and third can run once
Any one of the task from any one of the core completes

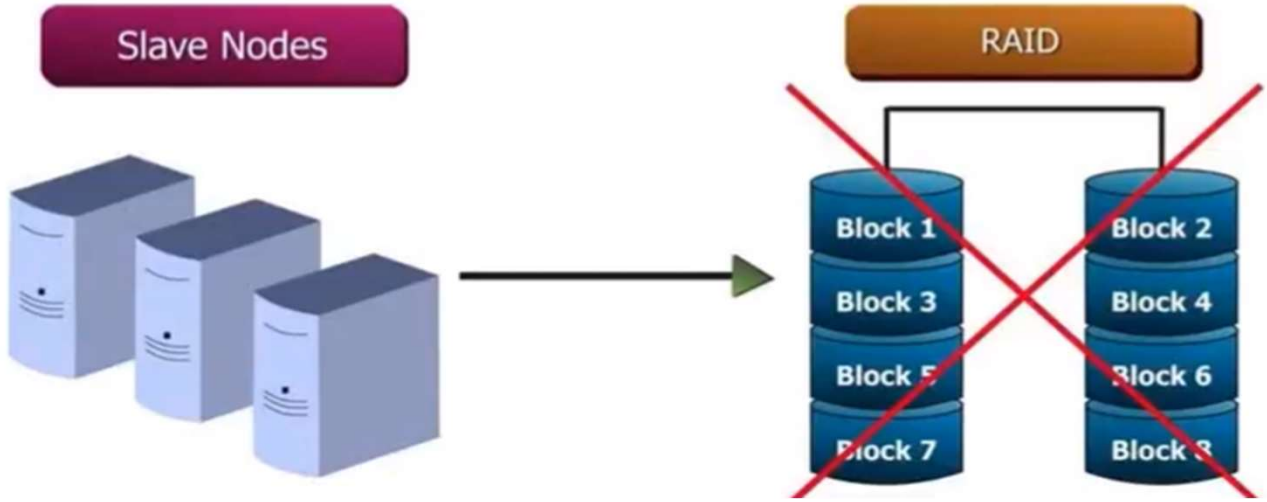
# Too much is too bad



# Say No to RAID on Data Nodes

**Slave Nodes do not benefit from using RAID* storage**

- ✓ HDFS provides redundancy by replicating blocks across multiple nodes
- ✓ RAID striping (RAID 0) is actually slower than the JBOD configuration used by HDFS
- ✓ RAID 0 read and write operations are limited by the speed of the slowest disk in the RAID array
- ✓ Disk operations on JBOD are independent, so the average speed is greater than that of the slowest disk




If we use RAID (RAID1 – MIRRORING) than we will be dividing memory by 6 since
1 x 3 + 1 x 3 = 6  Considering Replication =3
Speed of the RAID is decided by the Slowest Disk used.
Instead JBOD is suggested since with JBOD each Disk is free to run Independently
And so the total speed will be the average

# Just a Buch of Disk - JBOD



If we use RAID (RAID1 – MIRRORING) than we will be dividing memory by 6 since

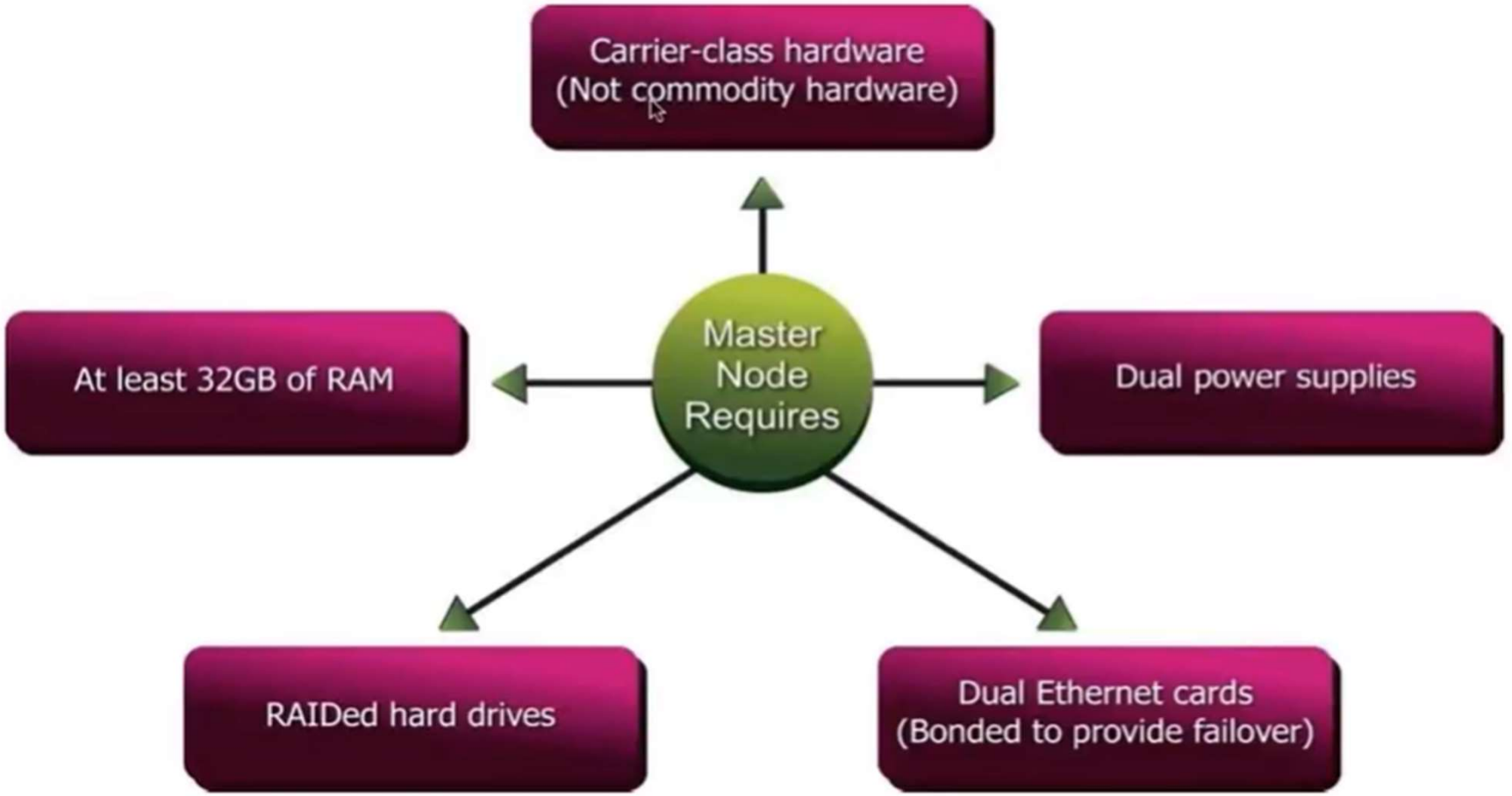
1 x 3 + 1 x 3 = 6  Considering Replication =3

Speed of the RAID is decided by the Slowest Disk used.

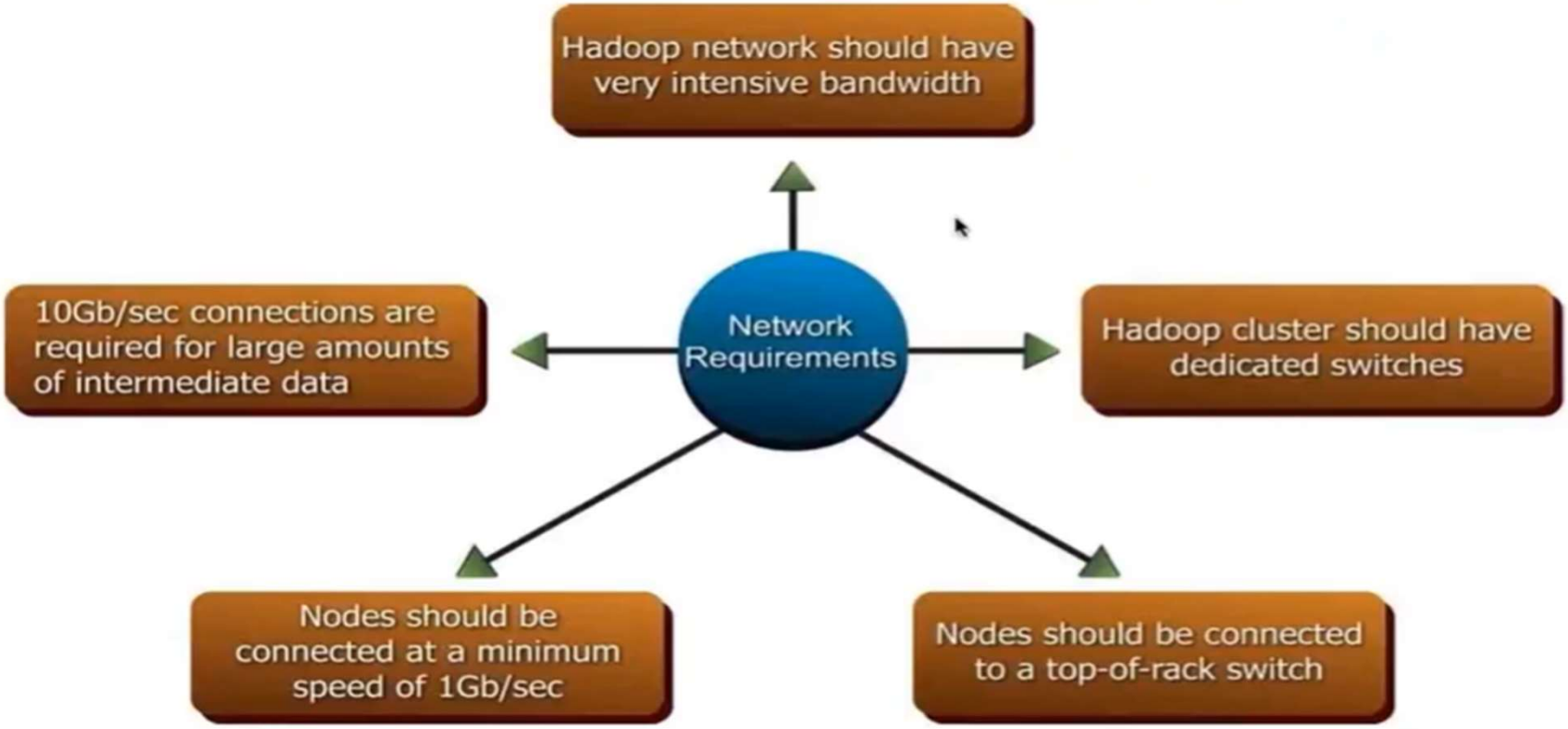
Instead JBOD is suggested since with JBOD each Disk is free to run Independently

And so the total speed will be the average of all

# Master Node H/W Reccomendatios

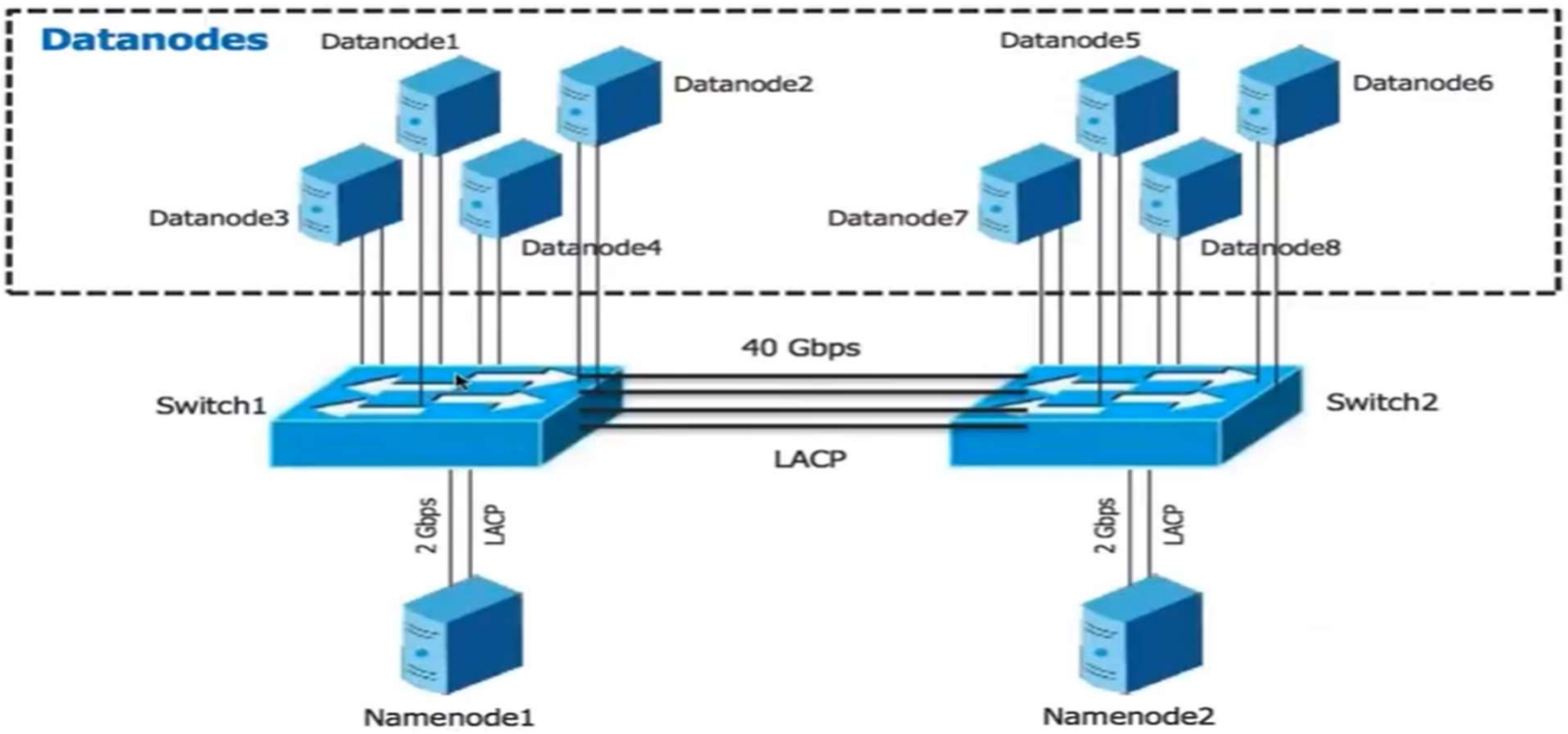


# Network Considerations




**Hadoop Use Jumbo Frames that uses 9000 bytes instead of MTU that support 1500 bytes**

# Advantages of Multiple NIC




Configure multiple NIC to behave as Bonded Interfaces.
Link Aggregation Control Protocol (LACP) gets triggered
If eth0 has 10 mbps and eth1 also has 10 mbps then
It will support Throughput of 20 mbps
If one of the NIC fail yjan system will continue working using remaining NIC

# Planning Hadoop Cluster - Data Size

Hadoop Storage (HS) = CRS / (1-i)

- C= Compression Ratio
- R= Replication Factor
- S= Size of the data to be moved into Hadoop
- i= Intermediate Factor

Calculating Required Number of nodes

- Data Compression, hence, C is 1.
- The standard replication factor for Hadoop is 3.
- The Intermediate factor is 0.25,
- Then the calculation for Hadoop, in this case:

HS = (1*3*S) / (1-(1/4)
HS = 4S

Expected Result is 4 Times the Initial Storage

# Calculating Required Number of nodes



The expected Hadoop Storage instance, in this case, is 4 times the initial storage. The following formula can be used to estimate the number of data nodes.

$$N = HS/D = (CRS/(1-i)) / D$$

$$N = 5000/25 = 200$$

D = Available disk space of Each Node

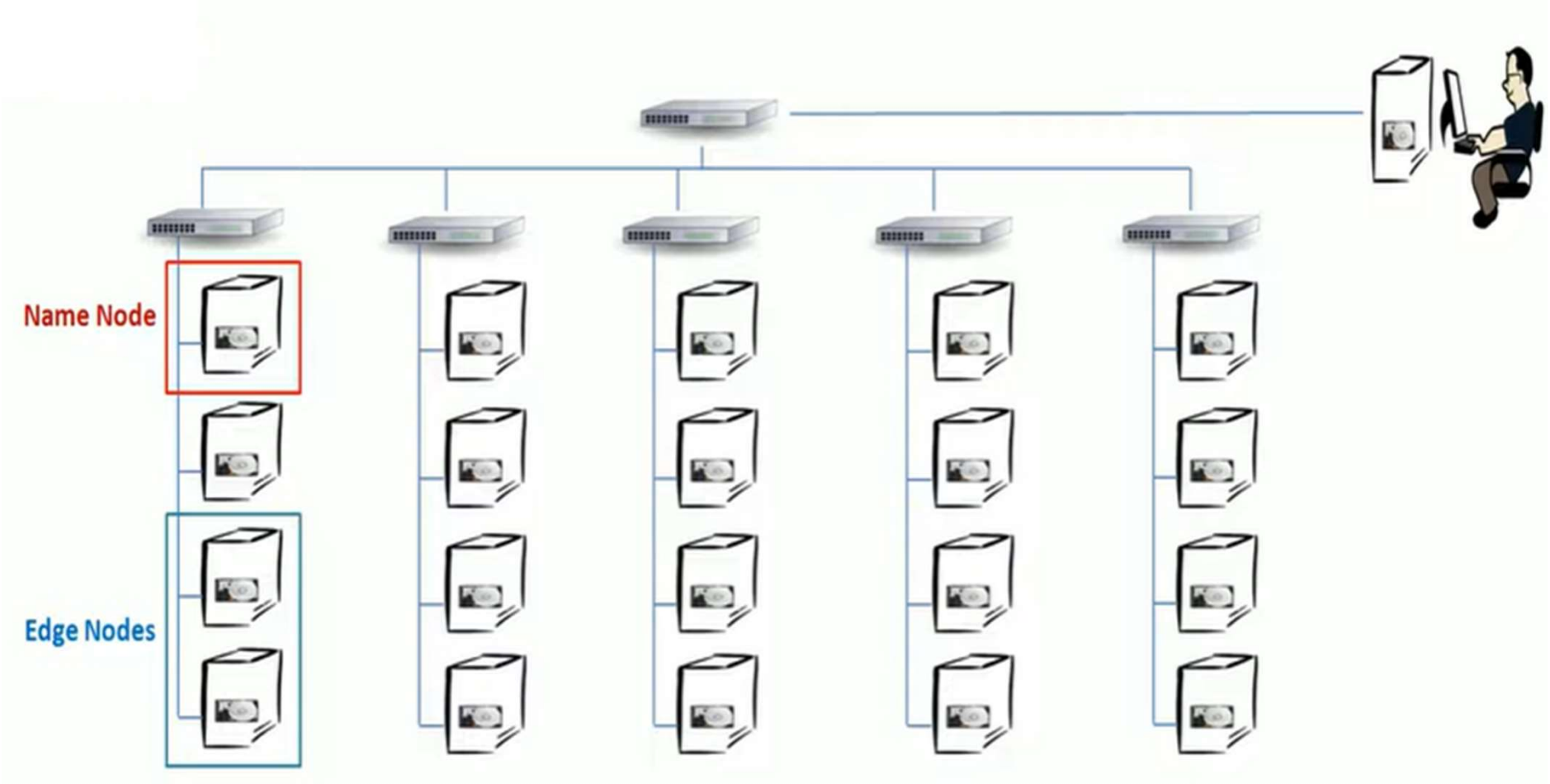
Consider.. Each node to have 27 disks of 1 TB.

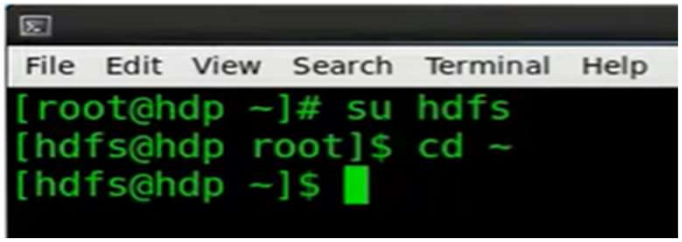
2 TB space goes to OS. Hence …27-2 =25

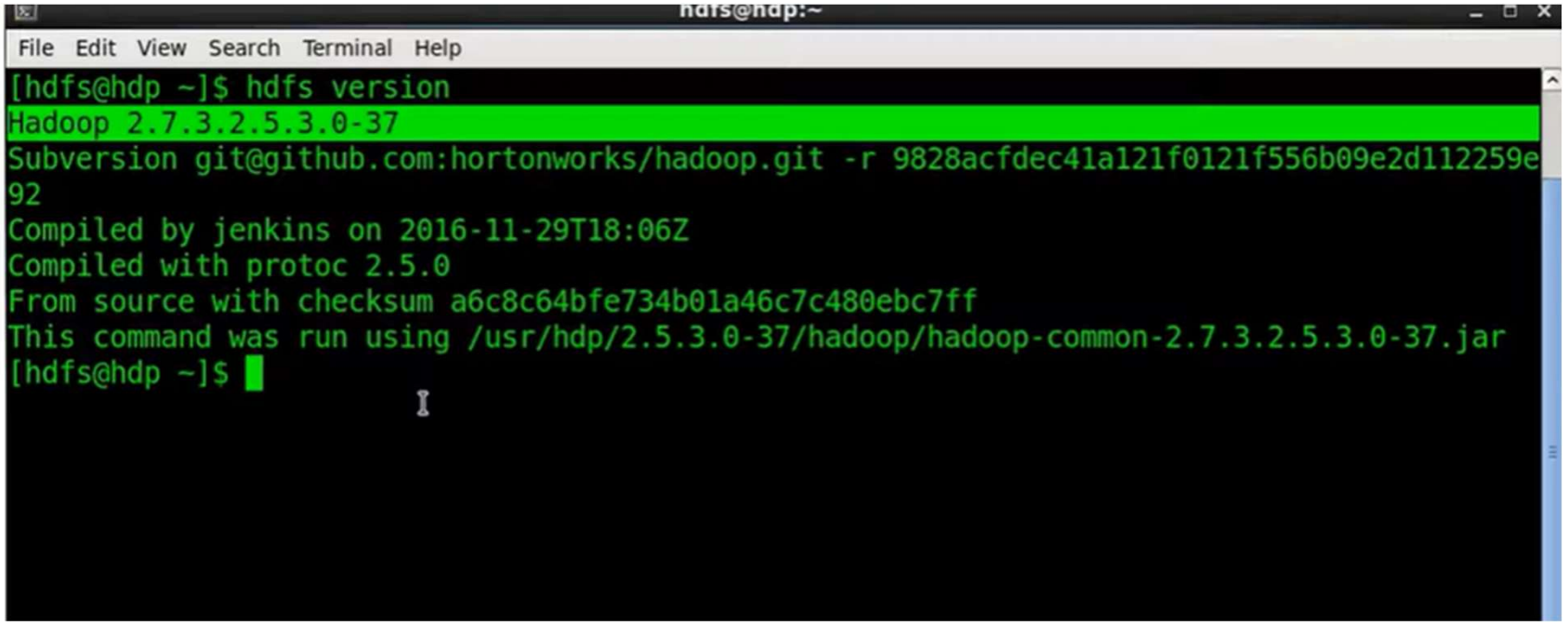
Assuming Initial data to be 5000 TB

**That is how we will need 200 Nodes**

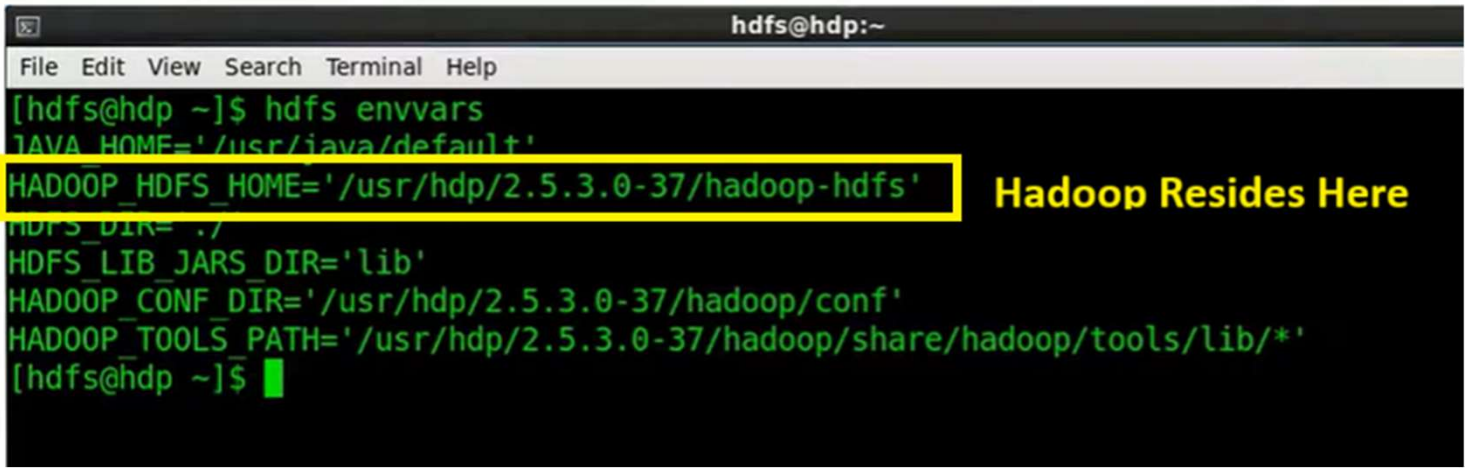
# Edge node – For Day to Day Operations

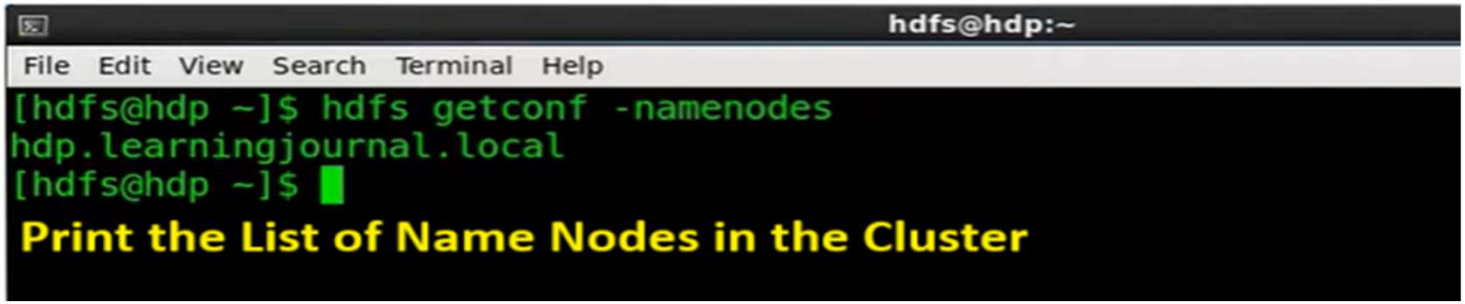


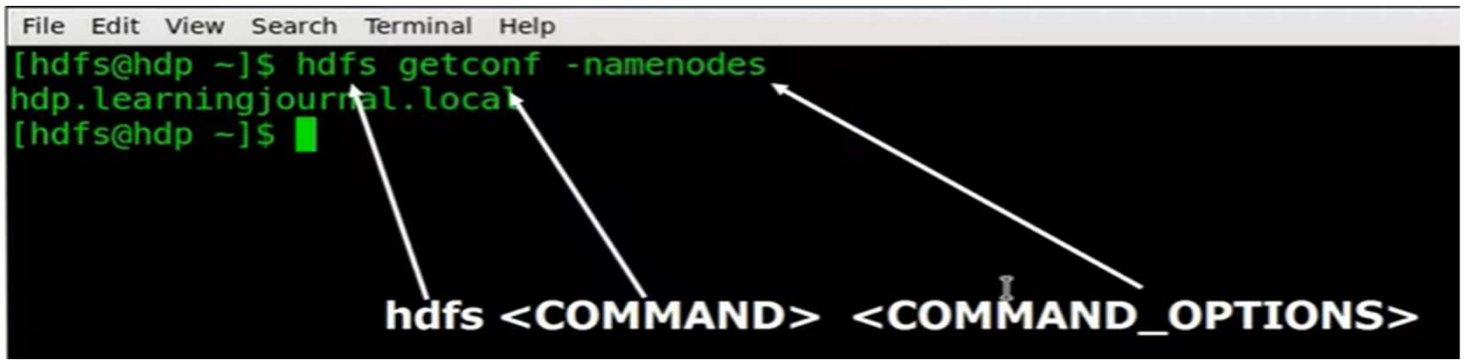

File Edit View Search Terminal Help
[root@hdp ~]# su hdfs
[hdfs@hdp root]$ cd ~
[hdfs@hdp ~]$


hdfs@hdp:~
File Edit View Search Terminal Help
[hdfs@hdp ~]$ hdfs version
Hadoop 2.7.3.2.5.3.0-37
Subversion git@github.com:hortonworks/hadoop.git -r 9828acfdec41a121f0121f556b09e2d112259e
92
Compiled by jenkins on 2016-11-29T18:06Z
Compiled with protoc 2.5.0
From source with checksum a6c8c64bfe734b01a46c7c480ebc7ff
This command was run using /usr/hdp/2.5.3.0-37/hadoop/hadoop-common-2.7.3.2.5.3.0-37.jar
[hdfs@hdp ~]$

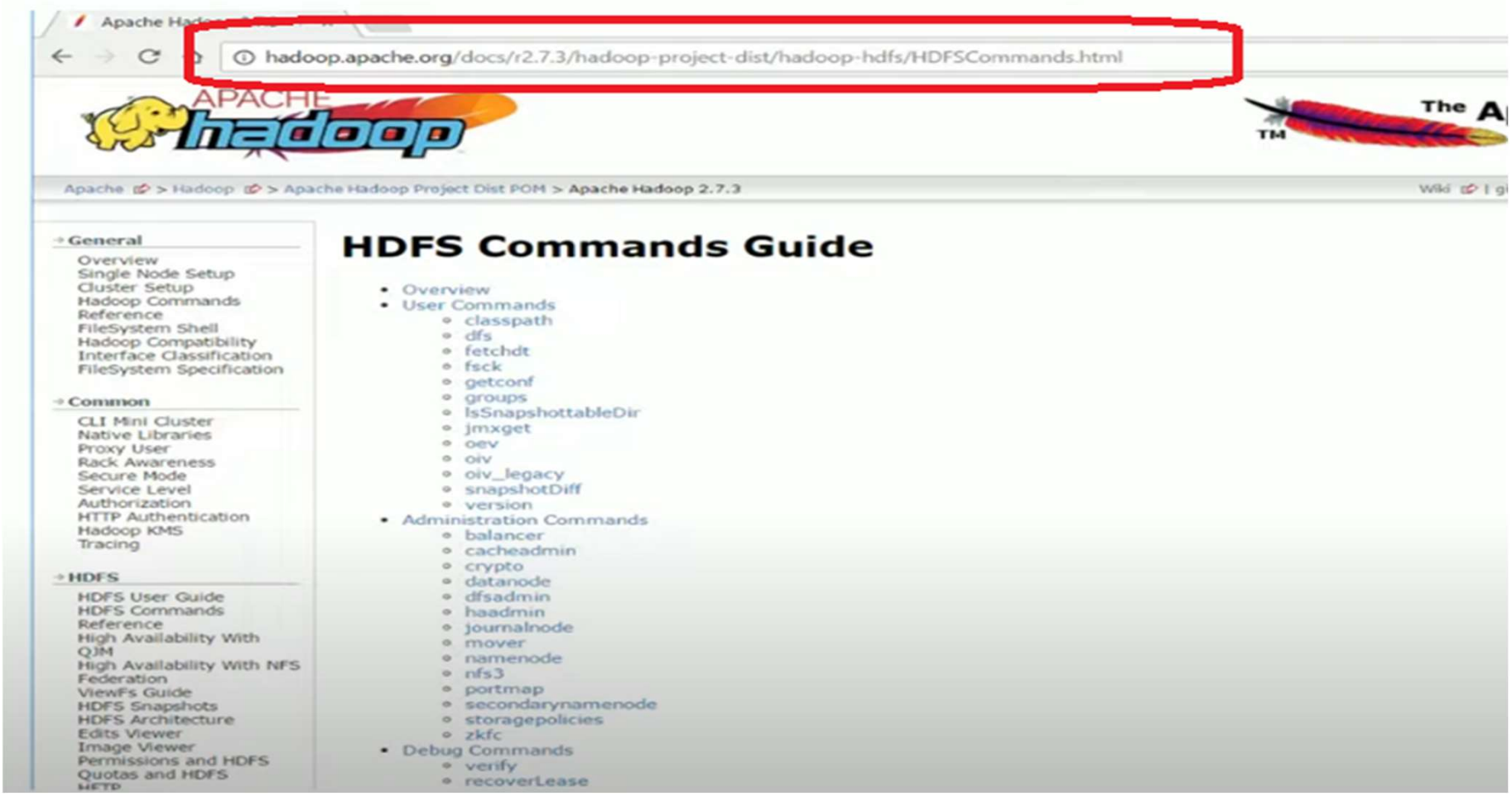
# Hadoop Commands



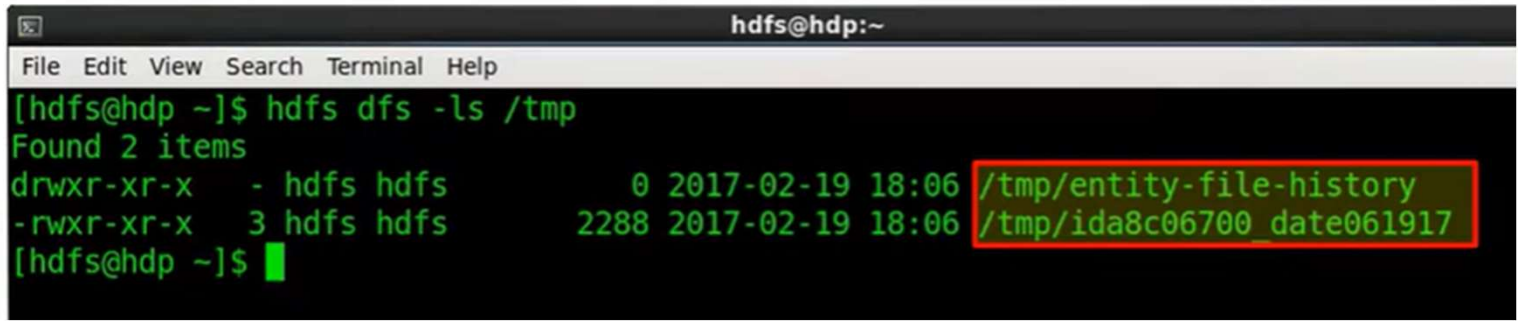



# List of Commands



# Works Just Like LINUX



# Commands to copy data

1. cp
2. copyFromLocal
3. Put
4. appendToFile

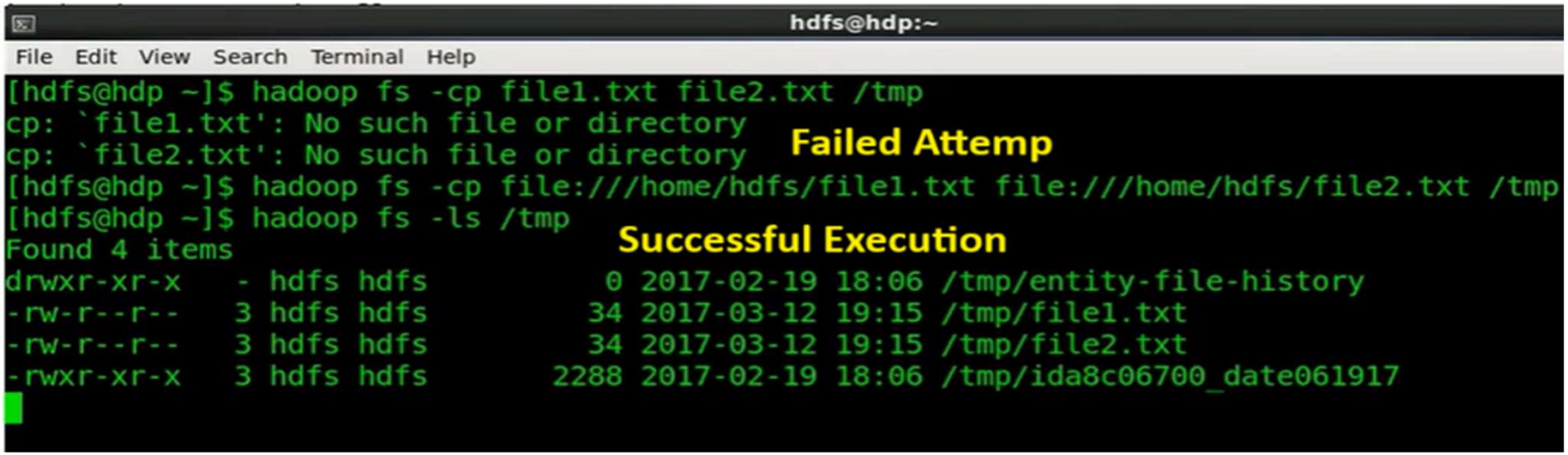
**Snippet from documentation**

**cp**

Usage: `hadoop fs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>`

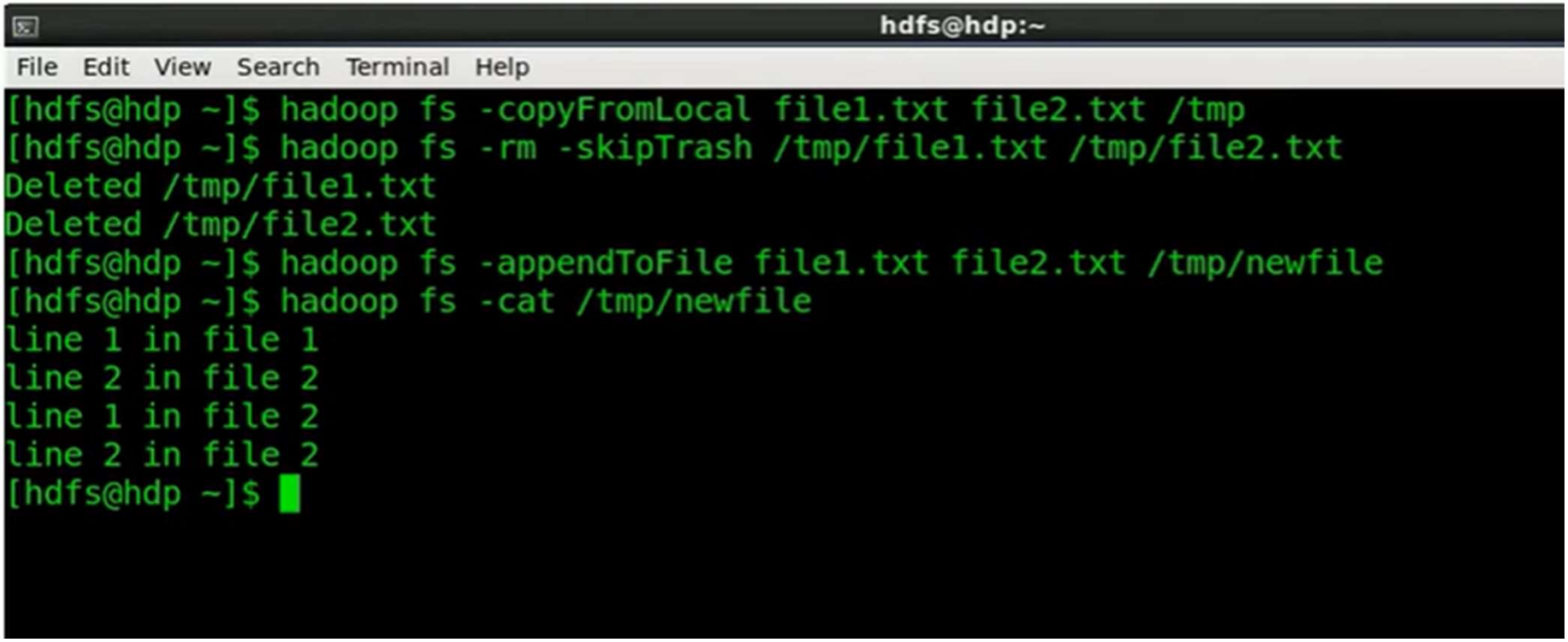
Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

# File Copy from




Copy data from One Directory to another or From Another File system to DHFS

# Copy to Local and Append



That is all for the day