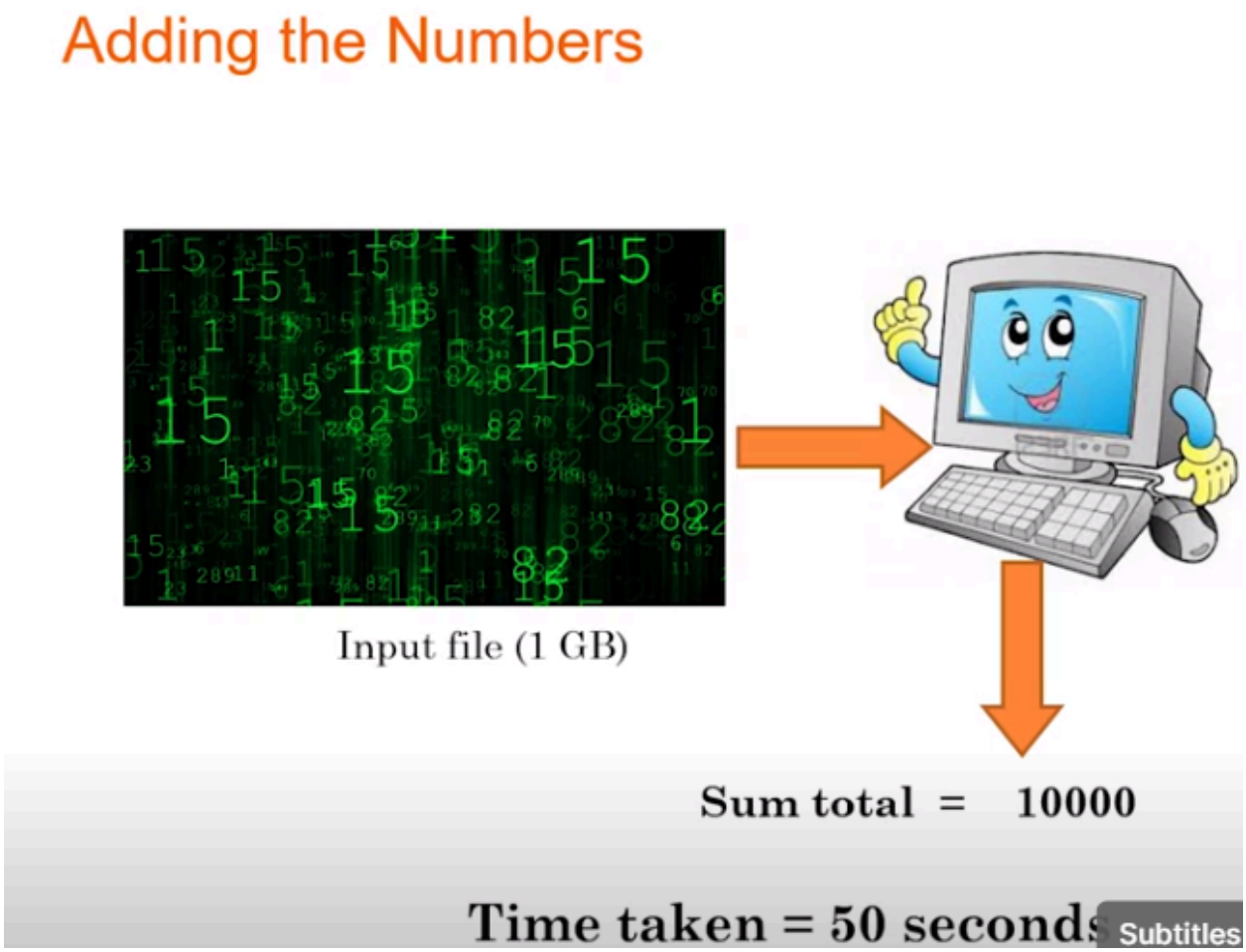
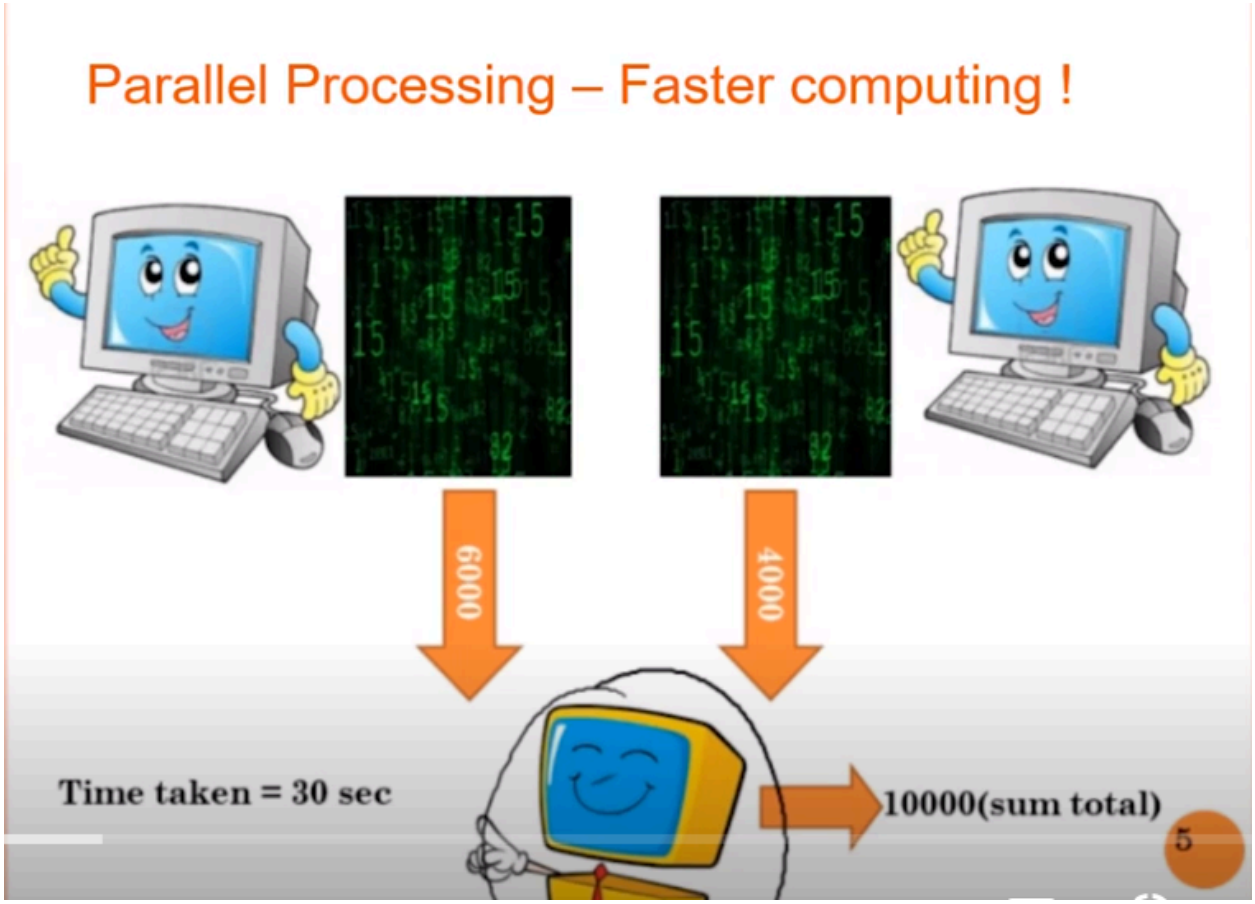
Getting started : Hadoop

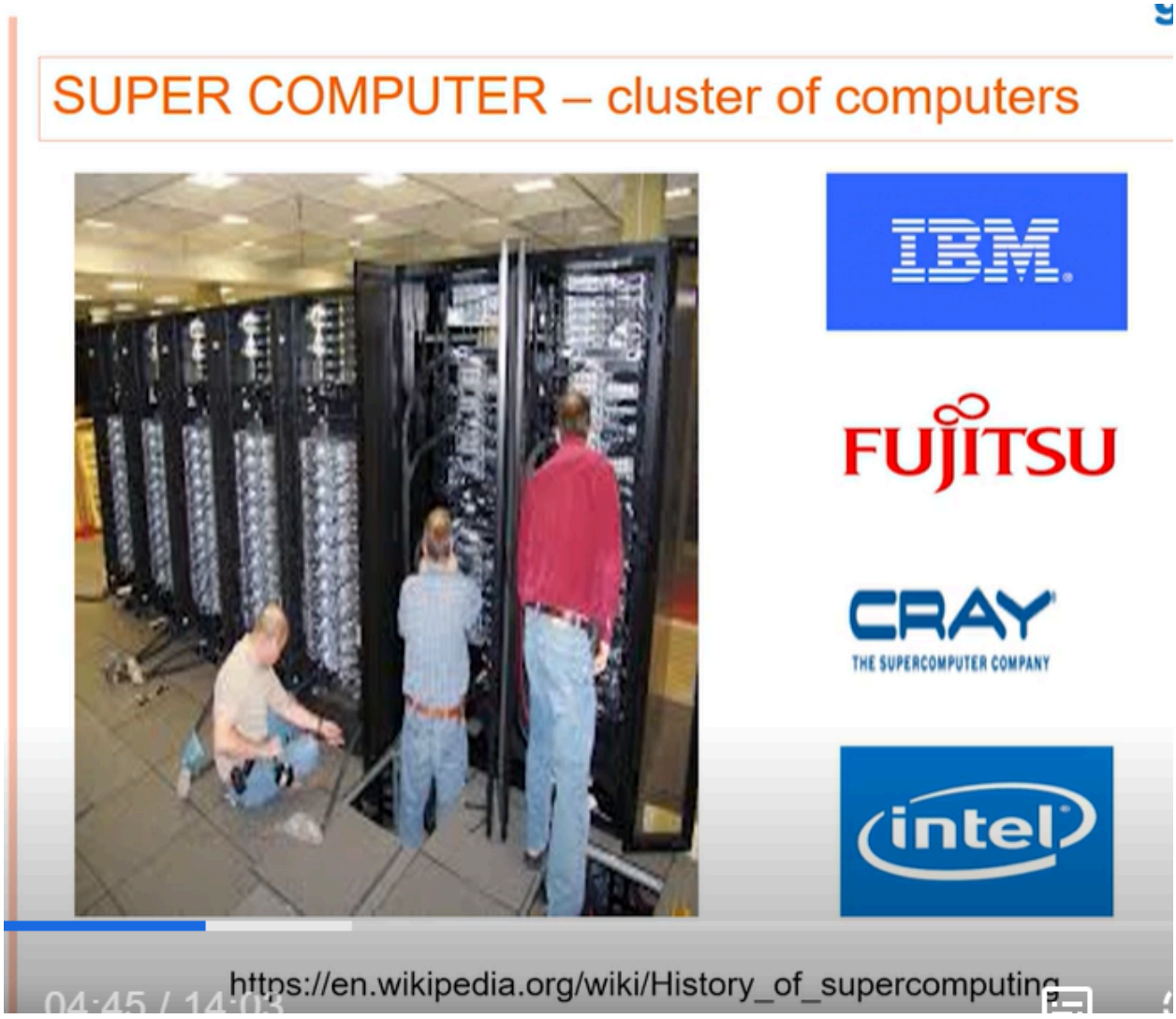
1. Lets take a living example(paint job)
2. Here each wall needs 2 hrs to get painted by a human
3. So what if we need to get all the 5 walls painted in 3hrs? ..we deploy more workers. It is case of problem solving
4. Lets take an another example



5. Here the input file is 1GB which has numerical values…our goal is to find the total sum of these value…
6. Now using a one system it takes 50 seconds to compute

7. 

8. Now what if we divide our file and fed into two diff systems and there will be 3rd system which combines the result of these two systems…then our total time taken for computation will be cut down.
9. Actual it has to be completed in 25 sec as we have 2 system working…but the extra 5 sec is taken by the 3rd system to coordinate the results.
10. This comes under divide and conquer strategy..and all those which comes under divide and conquer can be parallelised

11. 
SUPER COMPUTER – cluster of computers
IBM
FUJITSU
CRAY
THE SUPERCOMPUTER COMPANY
intel
https://en.wikipedia.org/wiki/History_of_supercomputing
04:45 / 14:03

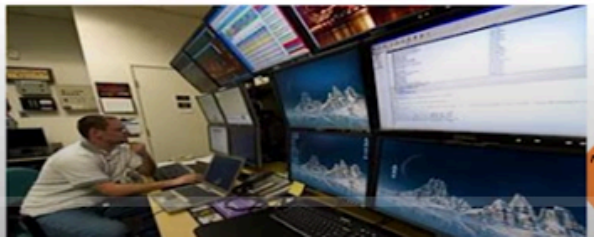
12. What is the need and uses of super computer?

Uses of Super Computers

Used in weather forecasting, seismic data analysis etc.

University research labs for studies related to computational fluid dynamics, bioinformatics etc.

Department of defense for nuclear research.



7

13. Challenges

Challenges With Super Computing

- A general purpose operating system like framework for parallel computing needs did not exist
- Companies procuring super computers were locked to specific vendors for hardware support
- High initial cost of the hardware
- Develop custom software for individual use cases
- High cost of software maintenance and upgrades which had to be taken care in house the organizations using a super computer.
- Not simple to scale horizontally

14. Now all these challenges are solved by hadoop

HADOOP Comes to the Rescue

- A general purpose operating system like framework for parallel comuting needs
- Its free software (open source) with free upgrades
- Has options for upgrading the software and its free !
- Opens up the power of distributed computing to a wider set of audience.
- Mid sized organizations need not be locked to specific vendors for hardware support – Hadoop works on commoditity hardware
- The software challenges of the orgnazation having to write proprietary softwares is no longer the case.

15. Where was the idea of hadoop generated?

Evolution of Hadoop From Internet Search Engines

- The need of the hour was scalable search engine for the growing internet
- Internet Archive search director Doug Cutting and University of Washington graduate student Mike Cafarella set out to build a search engine and the project named NUTCH in the year 2001-2002
- Google's distributed file system paper came out in 2003 & first file map-reduce paper came out in 2004
- In 2006 Dough Cutting joined YAHOO and created an open source framework called HADOOP (name of his son's toy elephant) HADOOP traces back its root to NUTCH, Google's distributed file system and map-reduce processing engine.
- It went to become a full fledged Apache project and a stable version of Hadoop was used in Yahoo in the year 2008

16.
17. From 2006 to now there are 3 versions of hadoop..and the most stable version is hadoop 2
18. PPT's : https://olympus.mygreatlearning.com/courses/10977/files/745868?module_item_id=449009

Hadoop Framework : stepping into hadoop

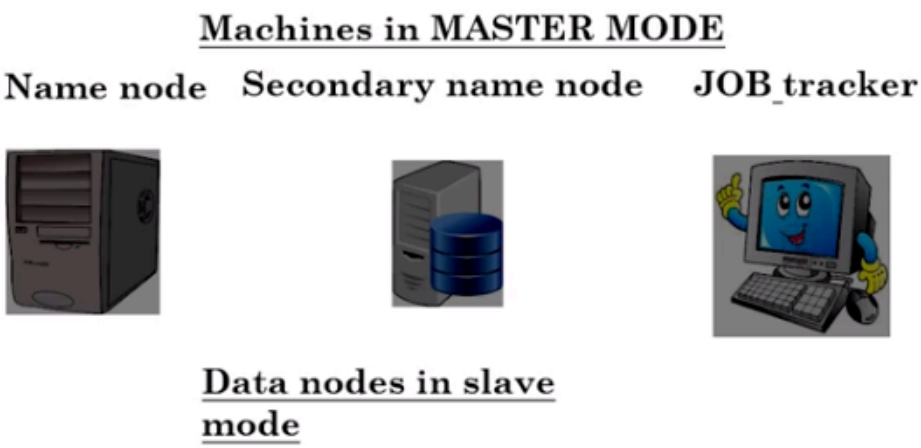
Key Terms

Some key terms used while discussing Hadoop

- Commodity hardware: PCs which can be used to make a cluster
- Cluster/grid: Interconnection of systems in a network
- Node: A single instance of a computer
- Distributed System: A system composed of multiple autonomous computers that communicate through a computer network
- ASF: Apache Software Foundation
- HA: High Availability
- Hot standby : Uninterrupted failover

1.
2. Architectural components of hadoop 1.0

Master-Slave Architecture



3.
4. There will always be 3 masters nodes in the cluster along with the data nodes and they are going to be physically connected by ethernet cable etc

HADOOP Deployment Modes

HADOOP supports 3 configuration modes when its is implemented on commodity hardware:

Standalone mode : All services run locally on single machine on a single JVM (seldom used)

Pseudo distributed mode : All services run on the same machine but on a different JVM (development and testing purpose)

Fully distributed mode: Each service runs on a seperate hardware (a dedicated server). Used in production setup.

Note: Service here reffers to namenode, secondary name node, job tracker and data node.

5.
6. If we install hadoop on each data node individually then it is called as fully distributed mode …which is used by current market
7. Pseudo distributed mode ..is used for educational purpose to learn and practise hadoop and we cannot work with large datasets here

Imagine you want to test your Hadoop skills without needing multiple computers. That's where pseudo-distributed mode comes in. It allows you to run a **miniature Hadoop cluster** on just **one machine**.

Think of it like a movie set. On a movie set, you have actors and props arranged to look like a real city, even though it's all on one stage. Similarly, in pseudo-distributed mode, you have **software processes mimicking different Hadoop components** (like NameNode, DataNode, and JobTracker) running on your single machine, pretending to be a full-fledged cluster.

8. Functionality of each component in hadoop eco system
9. Hadoop has HDFS in it..which saves files on multiple nodes.

Functionality of Each Component of HADOOP

Master nodes

Name node : Central file system manager

Secondary name node : Data backup of name node (not hot standby)

Job tracker : Centralized job scheduler

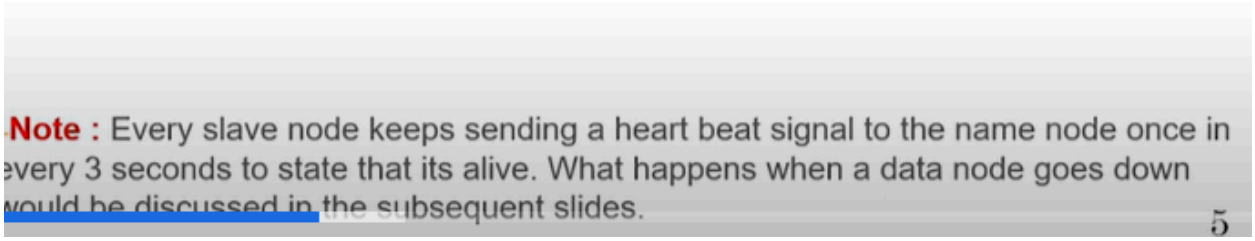
Slave nodes and deamons/software services

Data node : Machine where files gets stored and processed

Task tracker : A software service which monitors the state of job tracker

Note : Every slave node keeps sending a heart beat signal to the name node once in every 3 seconds to state that its alive. What happens when a data node goes down would be discussed in the subsequent slides.

5

10. 

11. In older versions of Hadoop (prior to Hadoop 2.0), the **JobTracker** played a crucial role in managing and orchestrating MapReduce jobs within the cluster. While it has been superseded by YARN (Yet Another Resource Negotiator) in newer versions, understanding its function can still be valuable for historical context and comprehension of distributed processing concepts.
12. The name node has all the metadata of data nodes..and acts as a manager for data present in the datanodes
13. And secondary name node will maintain the backup of namenode..in the event to namenode failure..we will copy the data from secondary node to name node in order to work normally again..here secondary NN is not a hot standby
14. But in hadopp 2.0 we have a hot standby in the event of failures
15. And the datanodes will be powerful than namenodes

16.

What is a job in HADOOP eco system?

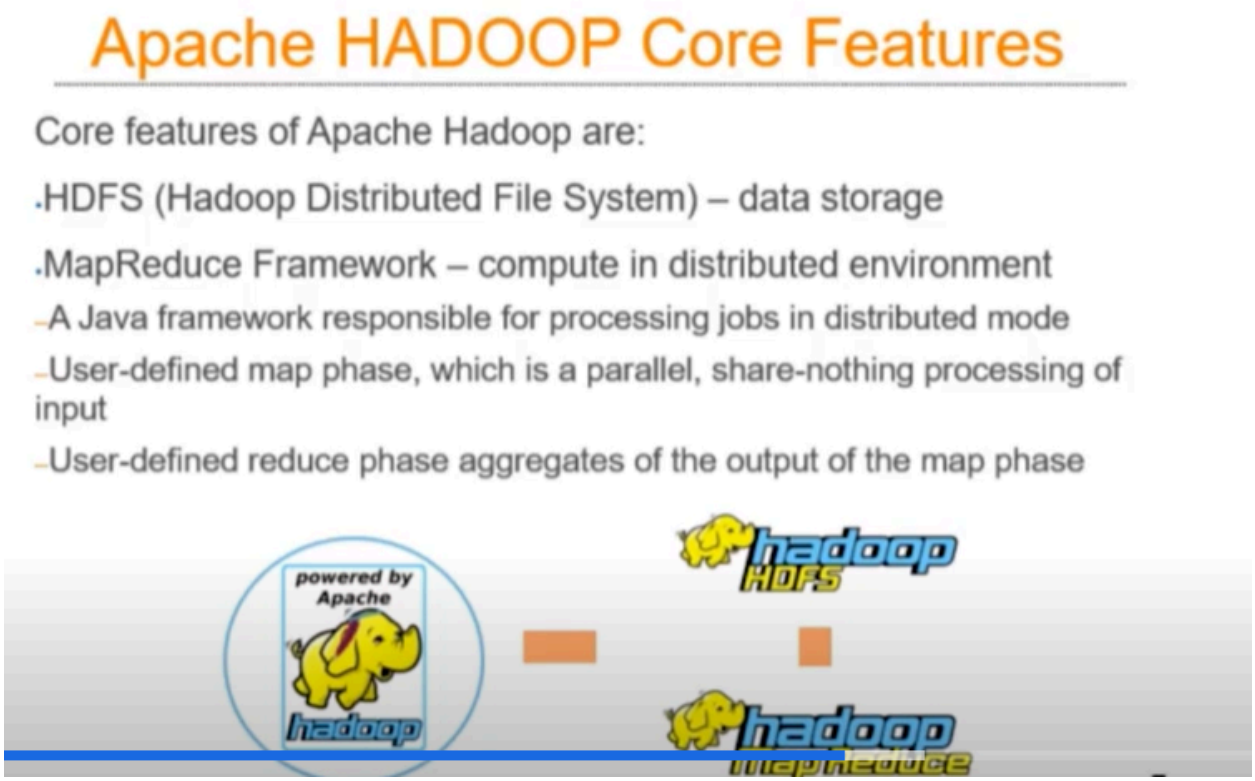
- A job usually is some **task** submitted by the user to the Hadoop cluster.
- The job is in the form of a **program or collection of programs** (a JAR file) which needs to be executed.
- A job would have the following attributes to it

  1)The actual program/s

  2)Input data to the program (a file or a collection of files in a directory)

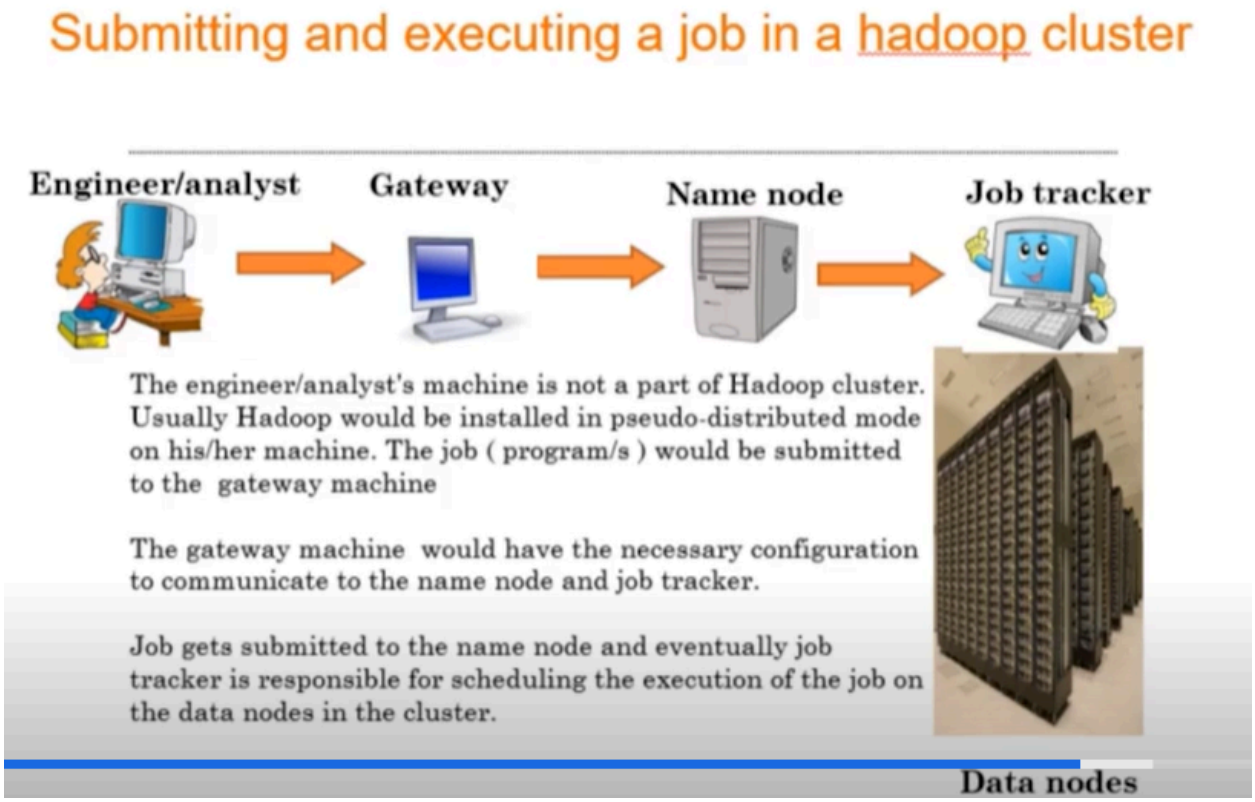
  3)The output directory where the results of execution is collected in a file/s

17.



18. Hadoop comes with HDFS ..hadoop used HDFS for storing and retrieving the data
19. To process the data which is in HDFS ..we use map reduce which is a parallel processing framework

20. Now lets see how a job is executed in hadoop



HDFS: What and Why?

Agenda

Architecture of HDFS

How does HDFS store the file internally

Failure handling and recovery mechanism

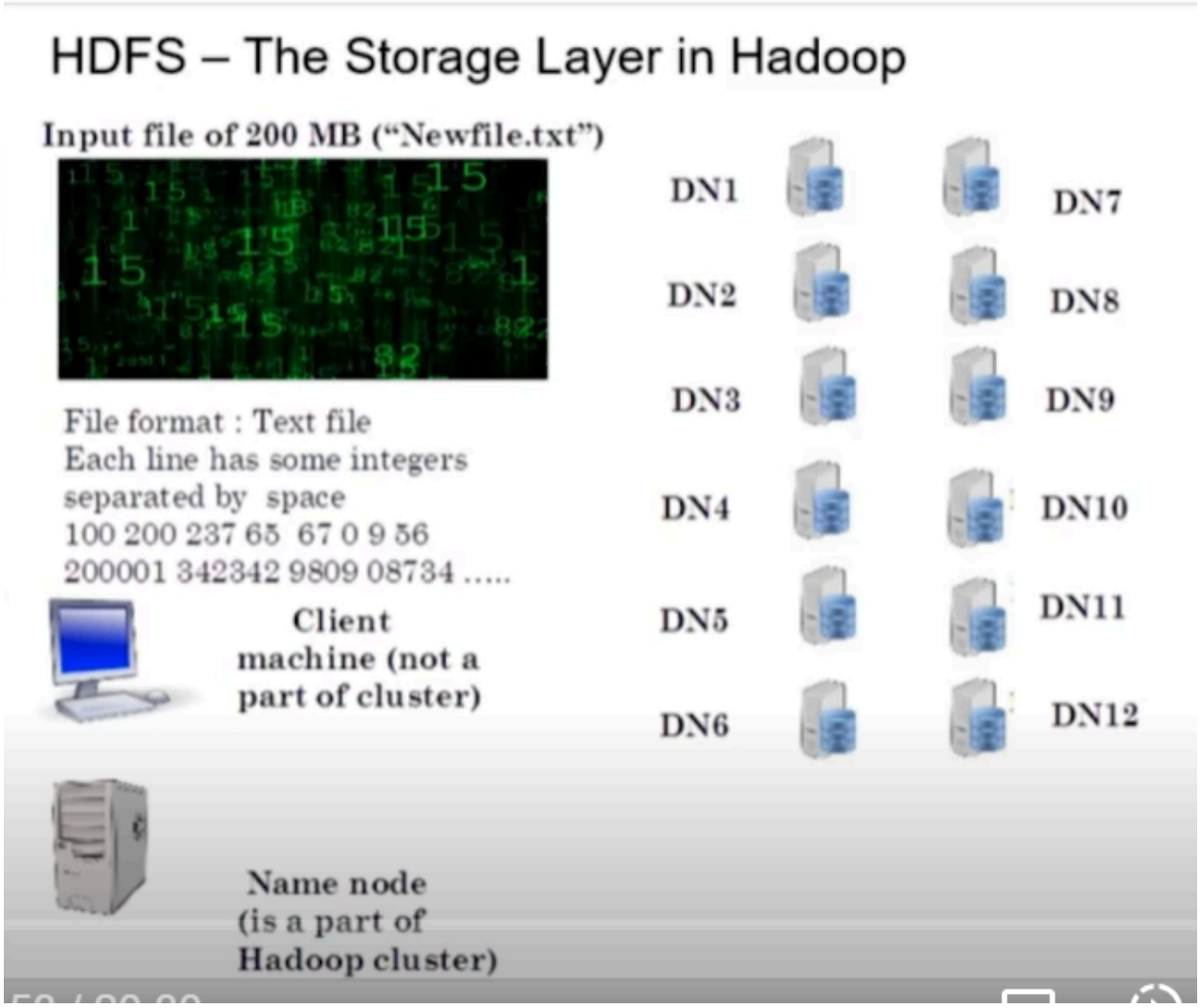
Rack awareness

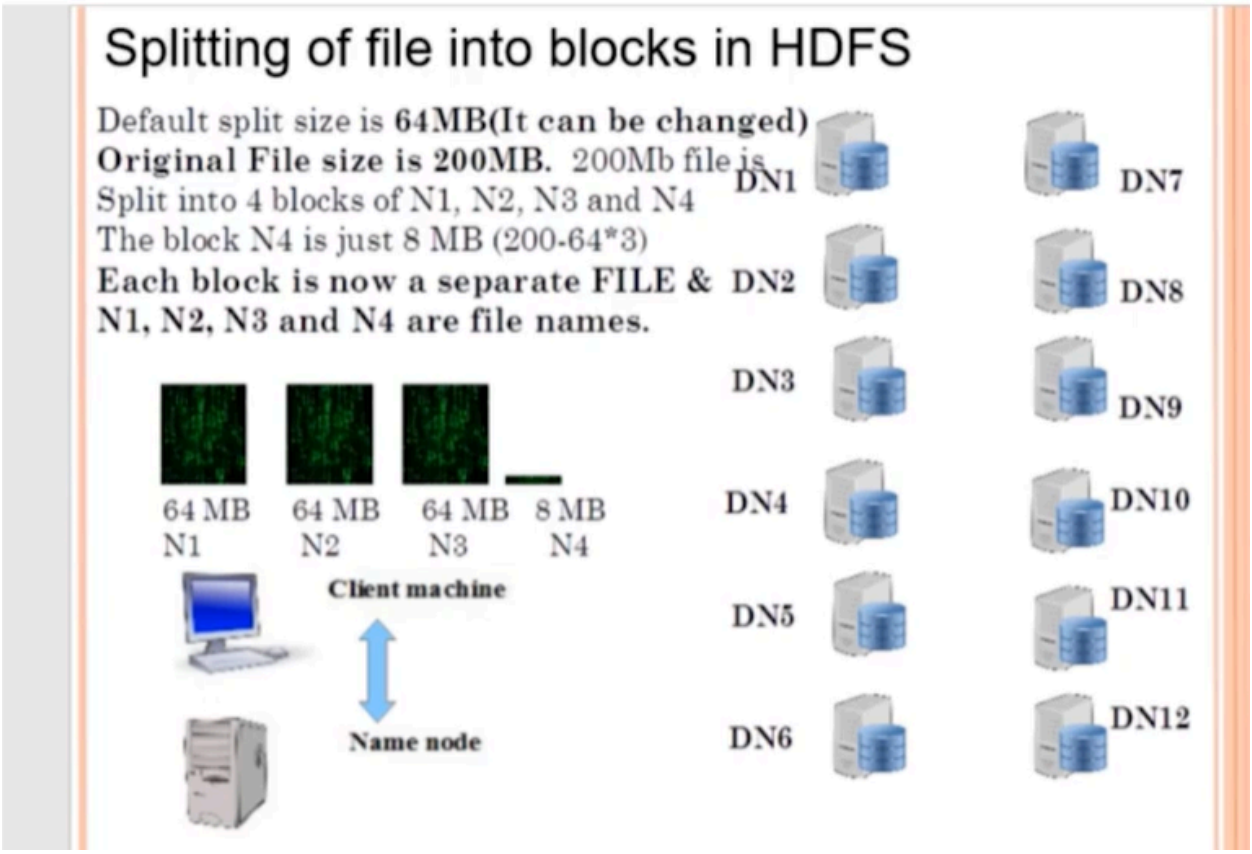
Role of name node and secondary name node

When to use HDFS and when not to use it

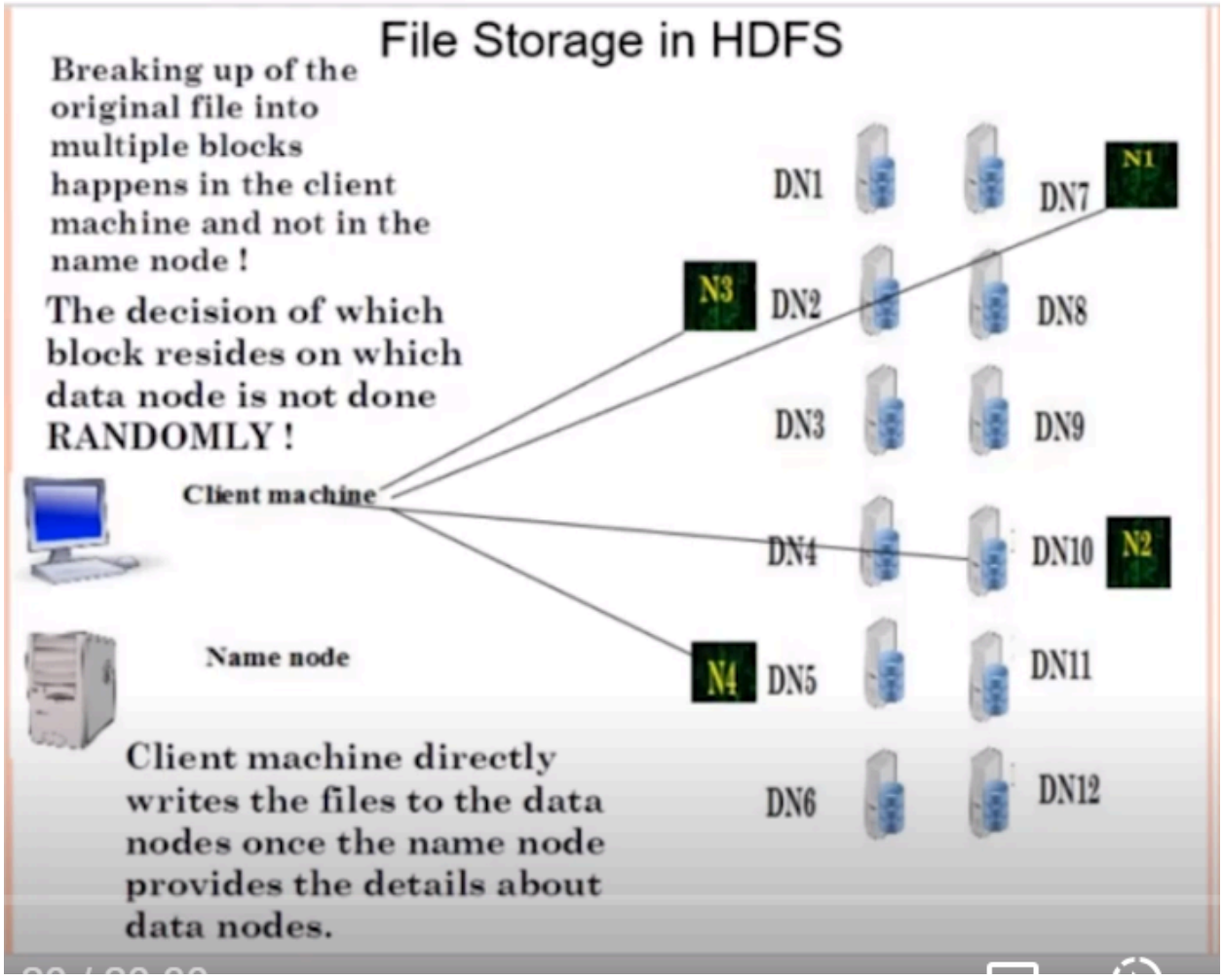
1.
2. Lets imagine we have file of 1GB..and assume we have downloaded it from internet ..now this file will be stored in single location(your drive)

3. How ever in hadoop it is diff…assume we have a file of 200MB ..and now we decided to store this file in hadoop by contacting name node



4. 

5. As we install hadoop by default we get split size = 64MB ..but we can configure to only multiple of 64mb (like 128MB, 256MB,192MB etc)
6. Here the name node will split the files into N1,N2,N3,N4 as see in pic(point 4) and here client machine is us.
7. And we have 12 datanodes and each datanode has their own hard drive to store the data

8. 
File Storage in HDFS
Breaking up of the original file into multiple blocks happens in the client machine and not in the name node !
The decision of which block resides on which data node is not done RANDOMLY !
Client machine
Name node
Client machine directly writes the files to the data nodes once the name node provides the details about data nodes.
DN1
DN2
DN3
DN4
DN5
DN6
DN7
DN8
DN9
DN10
DN11
DN12
N1
N2
N3
N4

9. Now first client machine will request

1. Client Initiates File Write:

- A client requests to write a file to HDFS.
- The client first contacts the NameNode, which acts as the master server managing the file system namespace and metadata.

2. NameNode Allocates Blocks:

- The NameNode doesn't store actual file data; it manages where blocks are located.
- It determines the number of blocks needed based on the file size and the configured block size (typically 128 MB).
- It assigns unique block IDs to each block.

3. Client Writes to DataNodes:

- The NameNode informs the client which DataNodes to write the blocks to.
- The client directly writes each block to the specified DataNodes.
- DataNodes store blocks on their local disks and periodically report their status (heartbeats) to the NameNode.

4. NameNode Updates Metadata:

- The NameNode receives block reports from DataNodes, confirming successful writes.
- It updates its metadata to reflect the block locations and replication status.
- It maintains a mapping of which blocks belong to which files and where those blocks are stored on DataNodes.

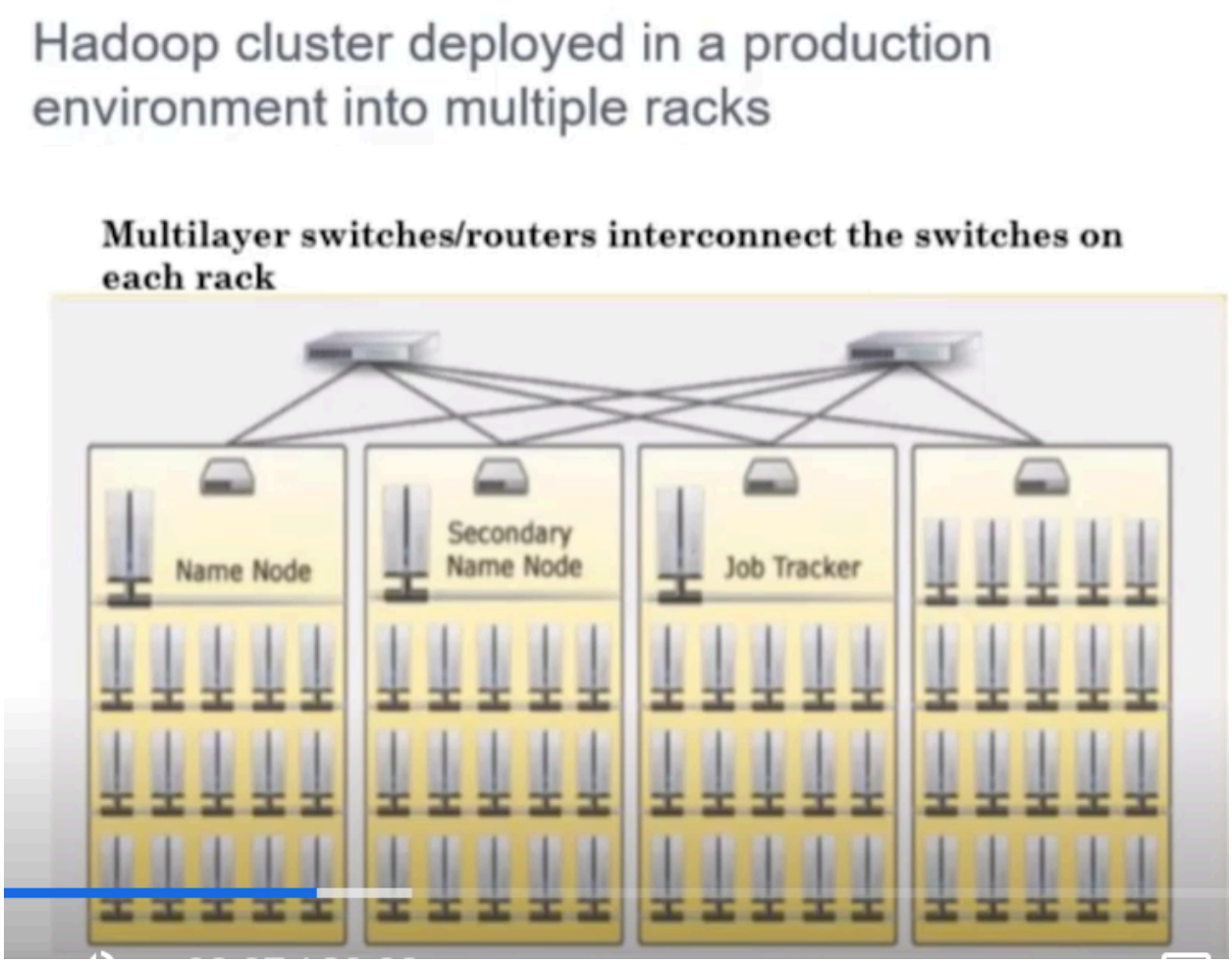
5. Replication for Fault Tolerance:

- HDFS replicates blocks across multiple DataNodes for data protection and availability.
- The NameNode strategically places replicas across different racks in a data center to prevent data loss if a rack fails.
- The default replication factor is 3, meaning each block is stored on 3 different DataNodes.

Example:

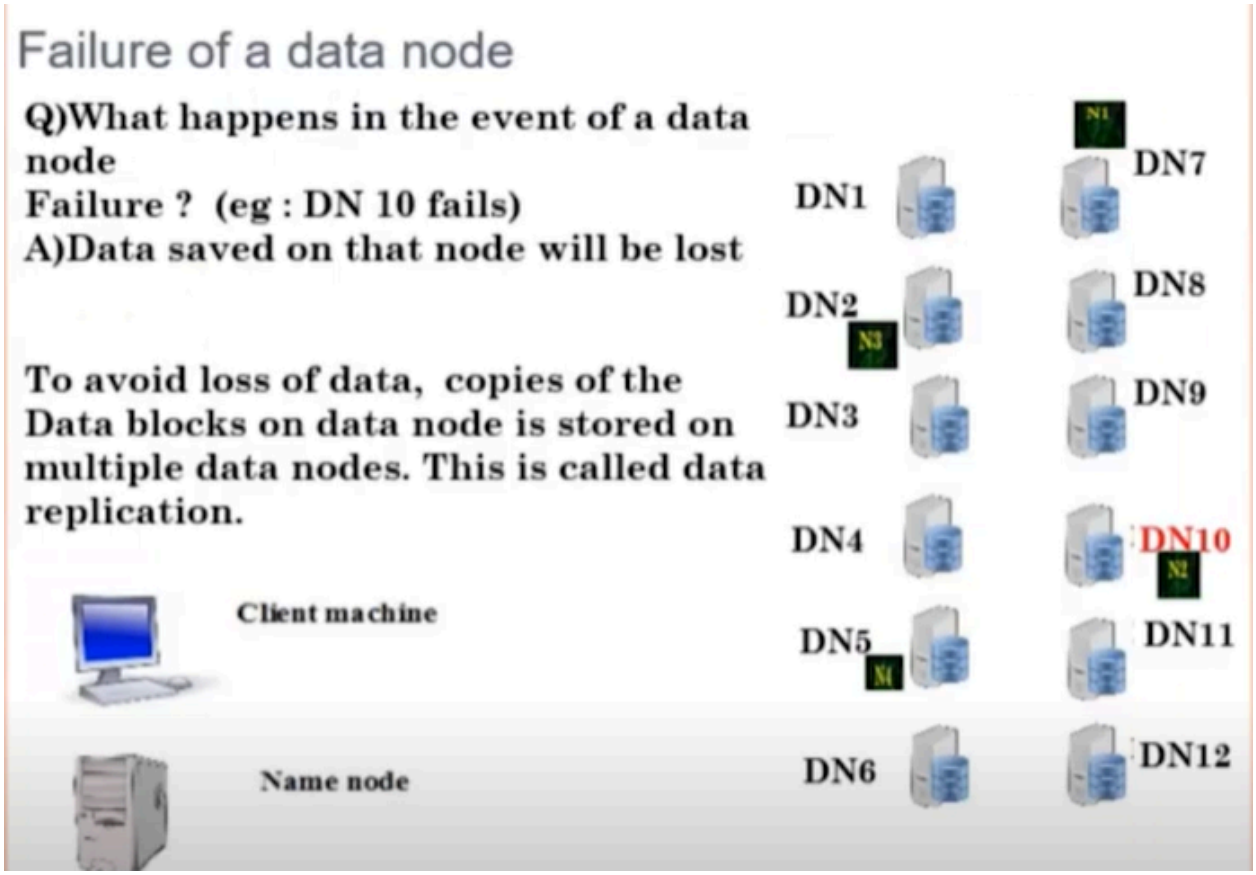
- Client wants to write a 500 MB file to HDFS.
- NameNode allocates 4 blocks (500 MB / 128 MB block size = 3.9 blocks, rounded up).
- NameNode assigns block IDs (e.g., block1, block2, block3, block4).
- NameNode directs the client to write block1 to DataNode1, block2 to DataNode2, and so on.
- Client writes blocks to specified DataNodes.
- DataNodes report successful writes to NameNode.
- NameNode updates metadata with block locations (e.g., file1 consists of block1 on DataNode1, block2 on DataNode2, etc.).
- NameNode ensures replication of blocks on other DataNodes for fault tolerance.

10. After datanodes reports to the namenode..namenode will contain all the metadata which is called as FSImage

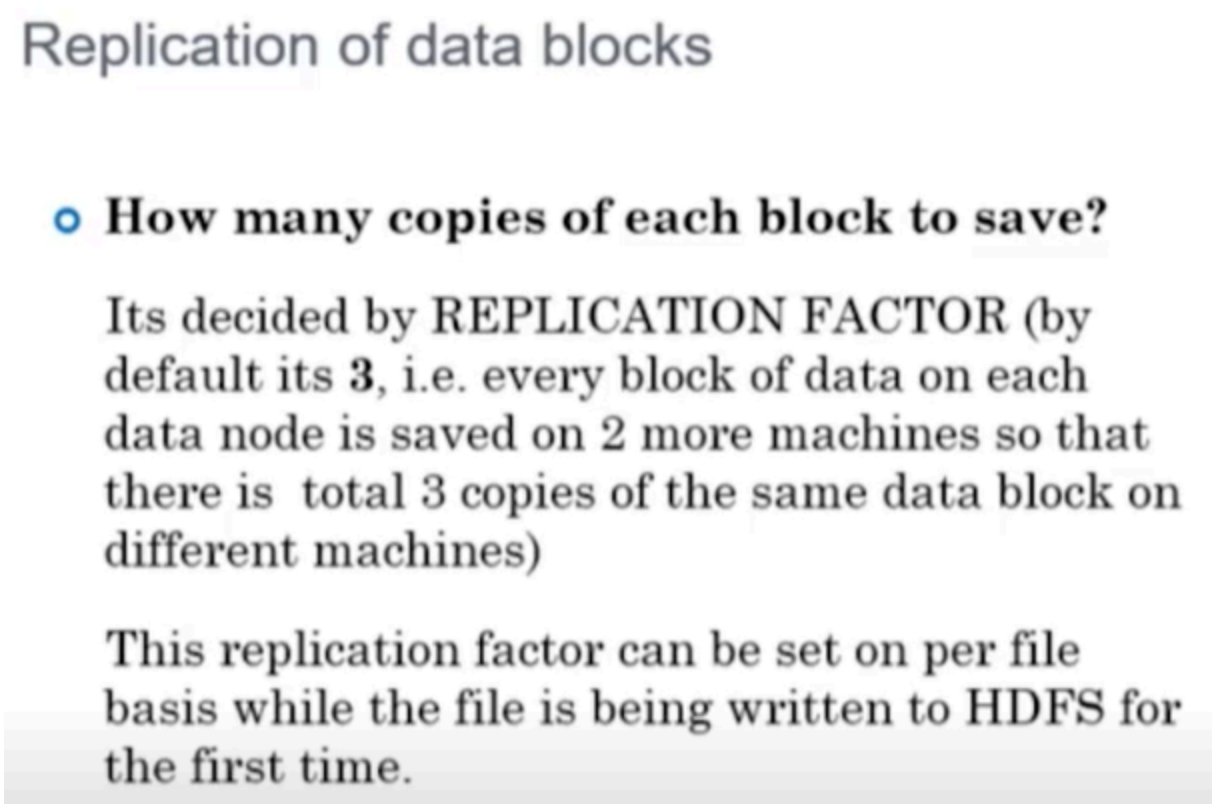
11. 

12. Failure of a datanode

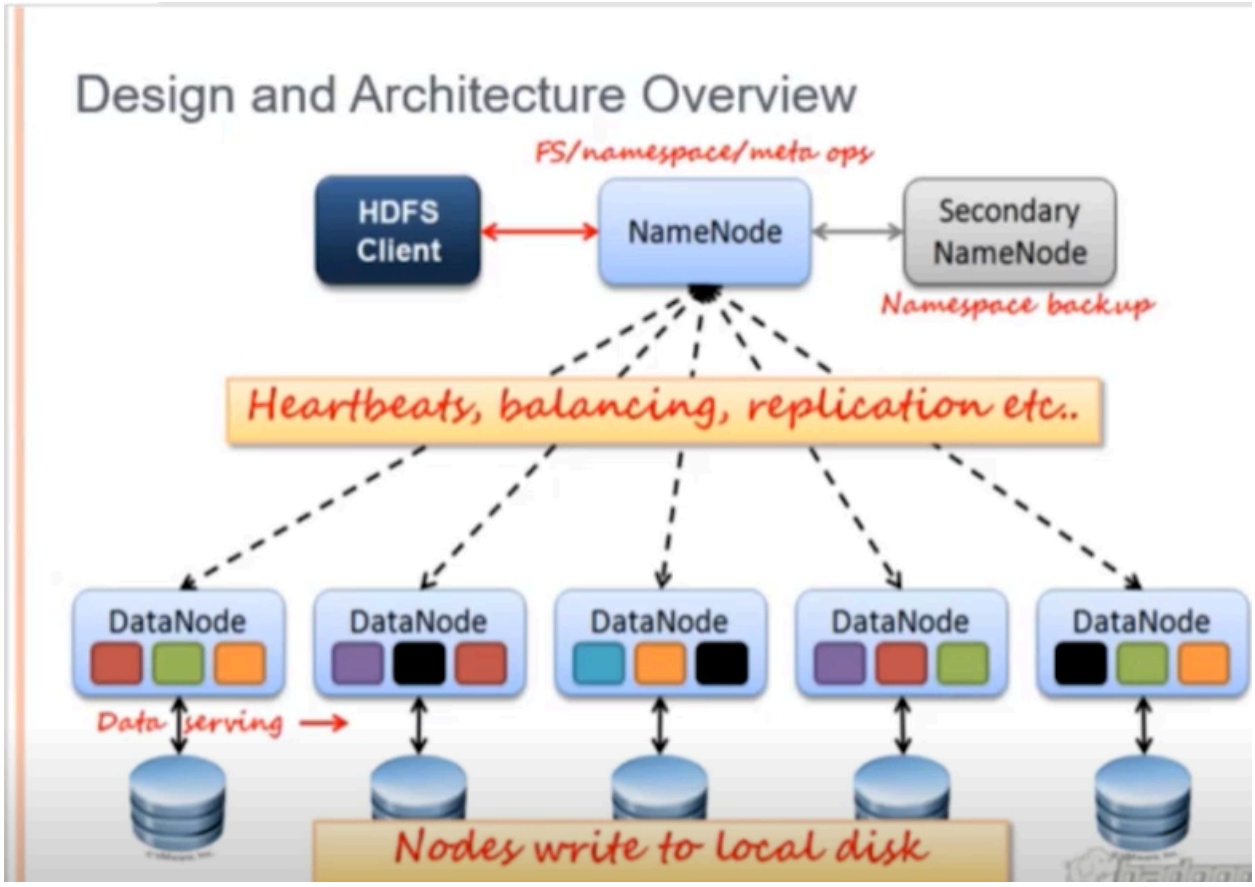
13. Here lets assume datanode 10 has failed



14. So suppose if we push a file of 200MB into hadoop..then overall it is going to occupy 600MB space on hadoop clusters(as the data is replicated 3 times)

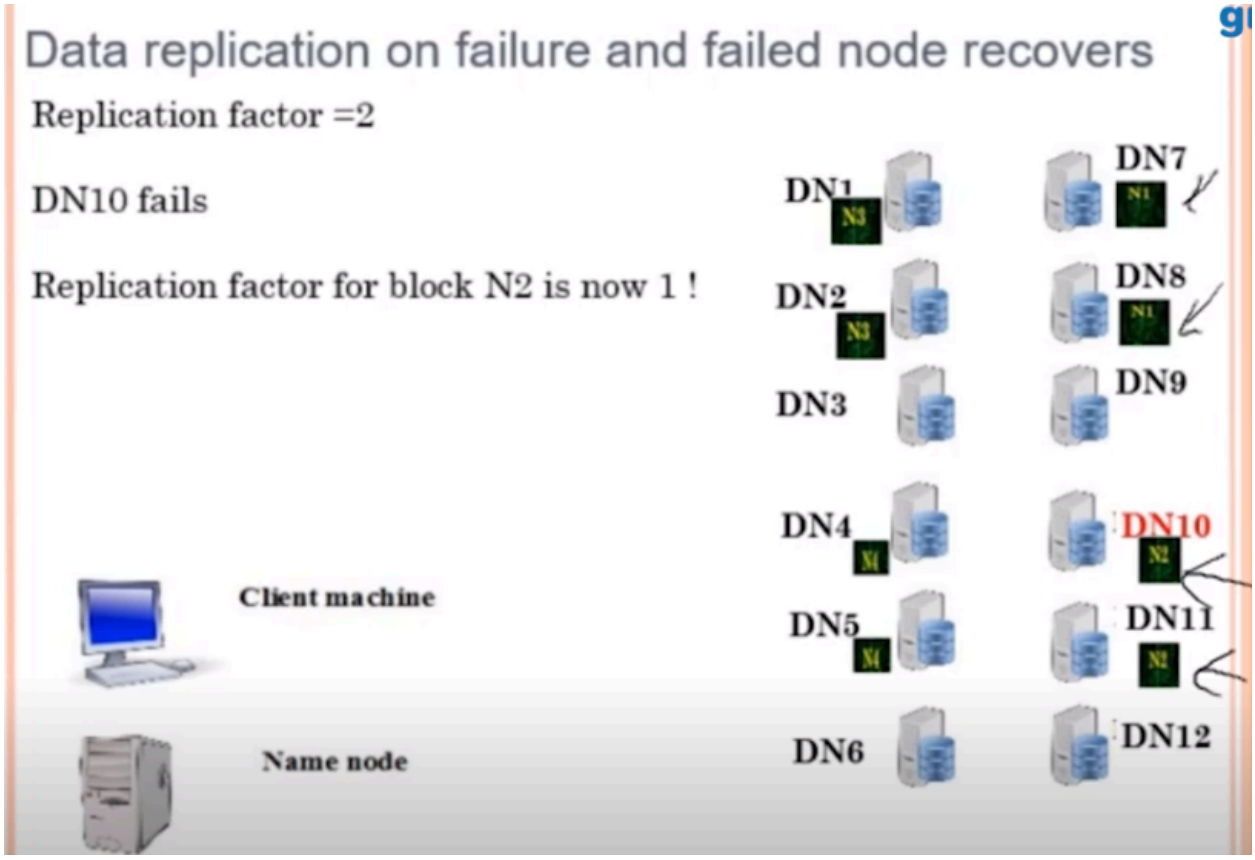
15. 

16. We can also configure our replication factor

17. 

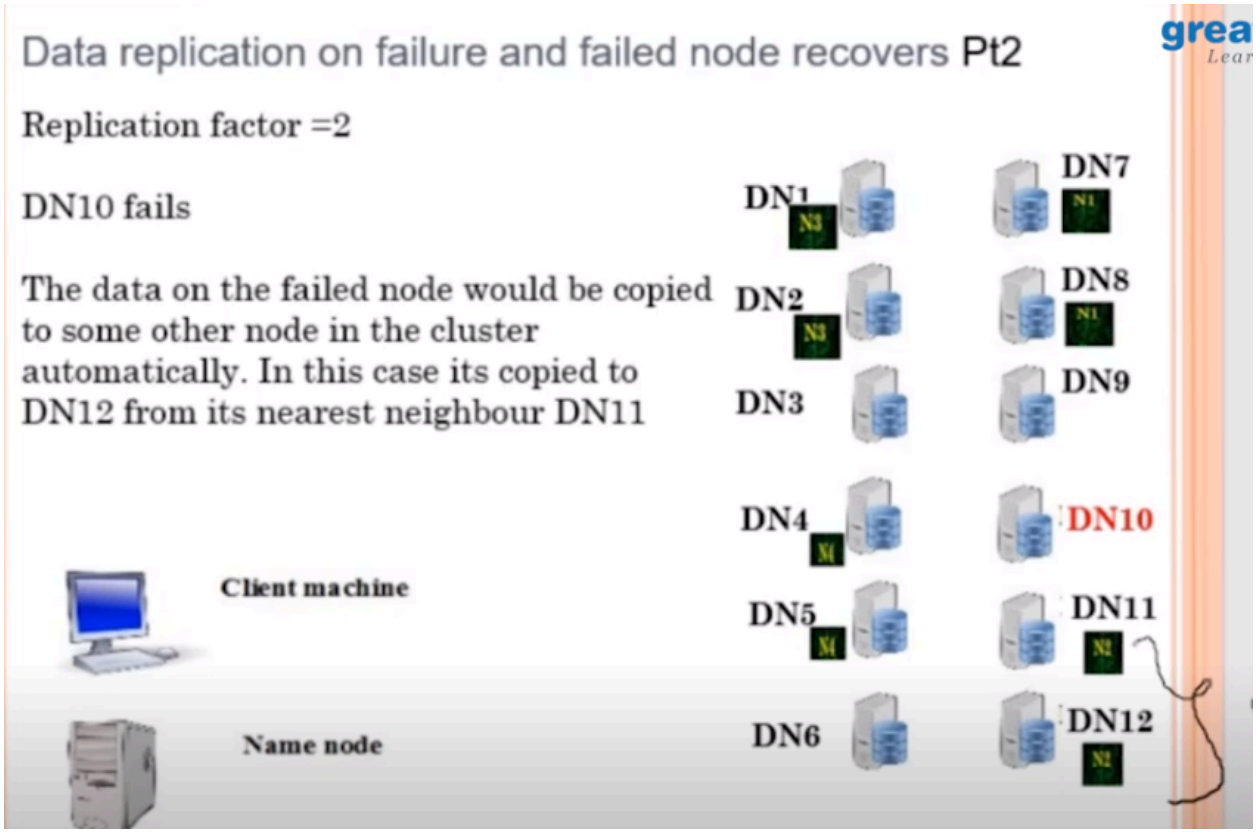
18. Here we can see every block of data(colord) in data node are replicated 3 times in various data nodes
19. Every datanode will send a heartbeat signal to namenode for every 3sec..if a datanode do not send signal..then namenode assumes it as failed
20. Now as soon as one data node fails ..the replication of the blocks which are in the failed DN will be 2…now the name node will send a signal to DN's to replicate these blocks of data and maintain the replication factor
21. Example of failover

22. Lets assume our RF = 2 and our block has replicated in data nodes twice



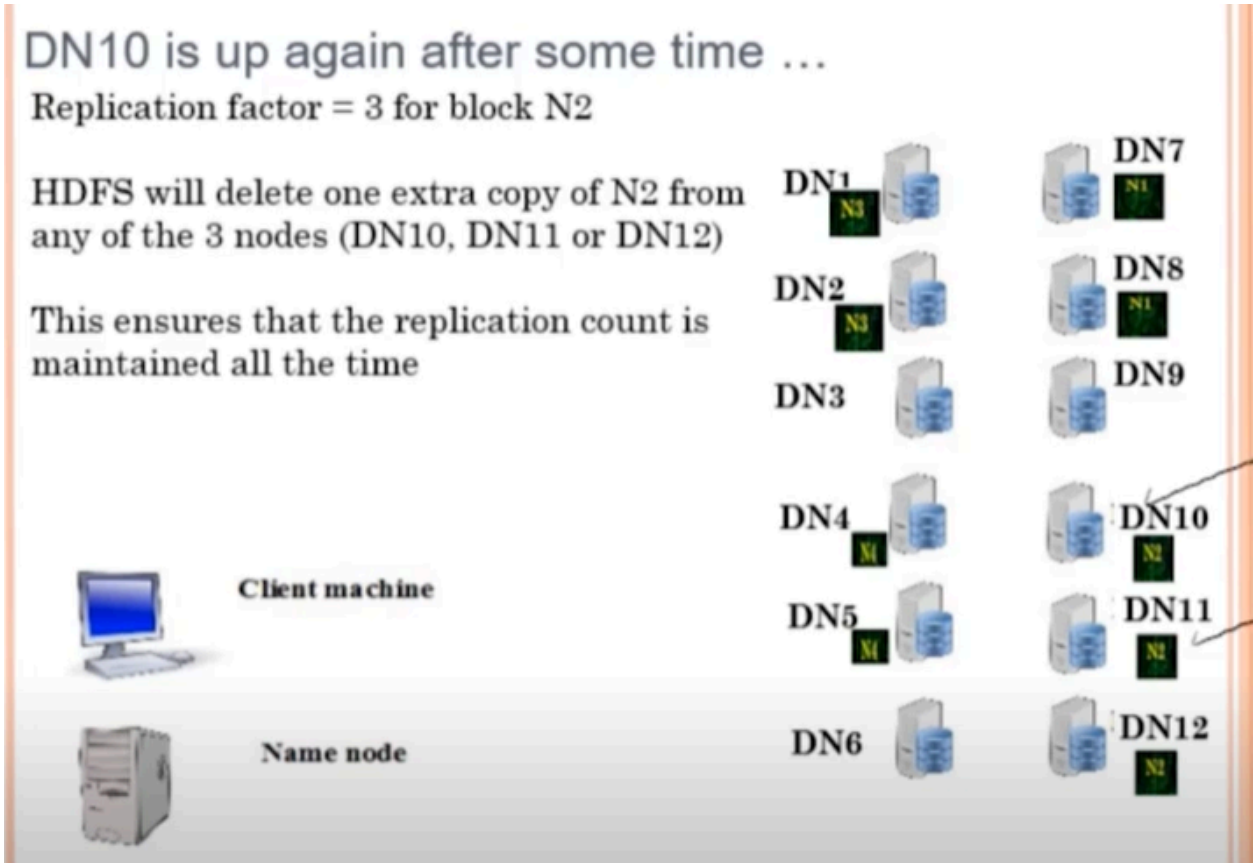
23. Now if the DN10 fails..it stops sending heartbeat signal to namenode..then namenode will ask the datanode(DN11) which has same data of DN10 to replicate it to another DN

which is close to DN11



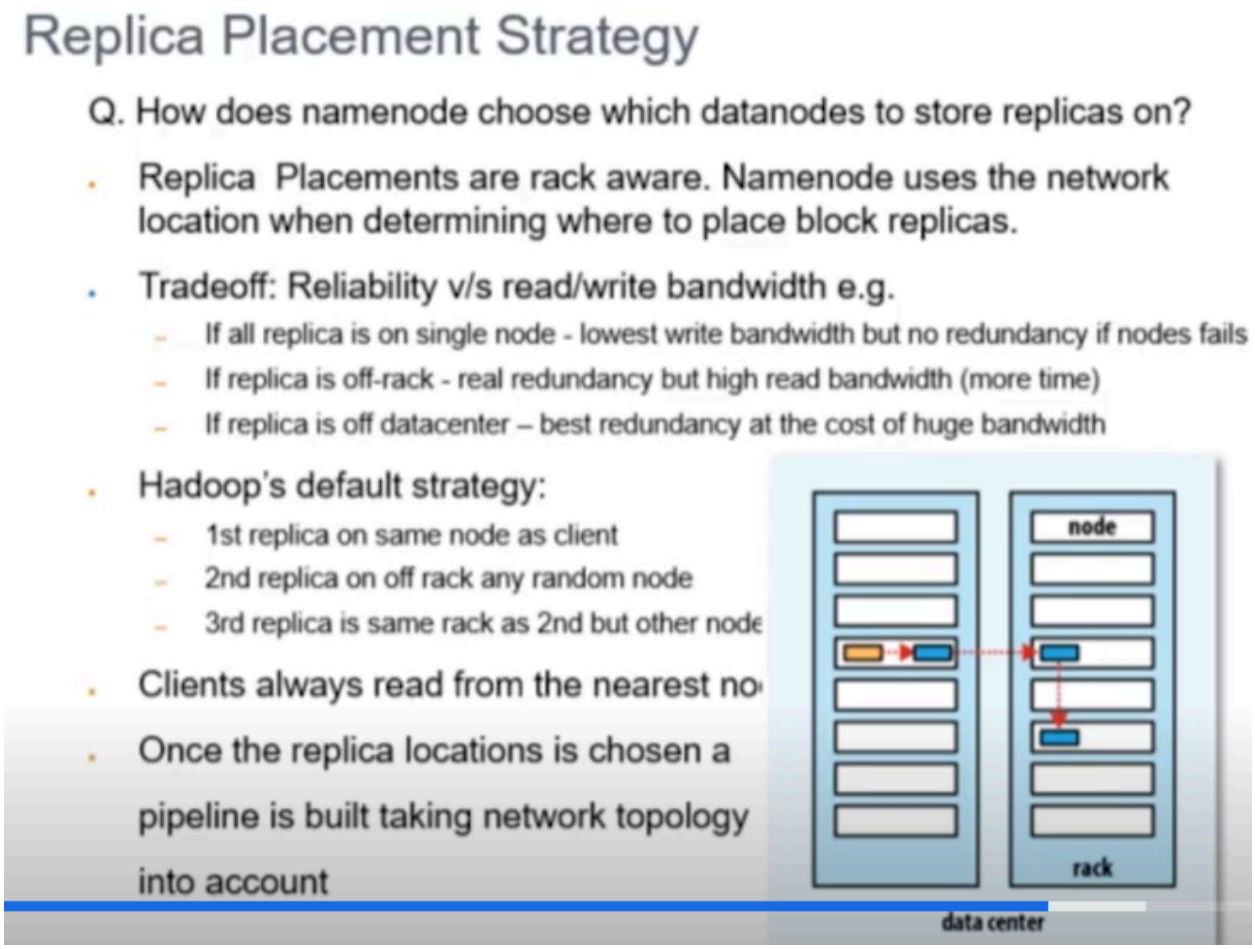
24. But what is DN10 come back from failure

25. Now the replication factor becomes 3

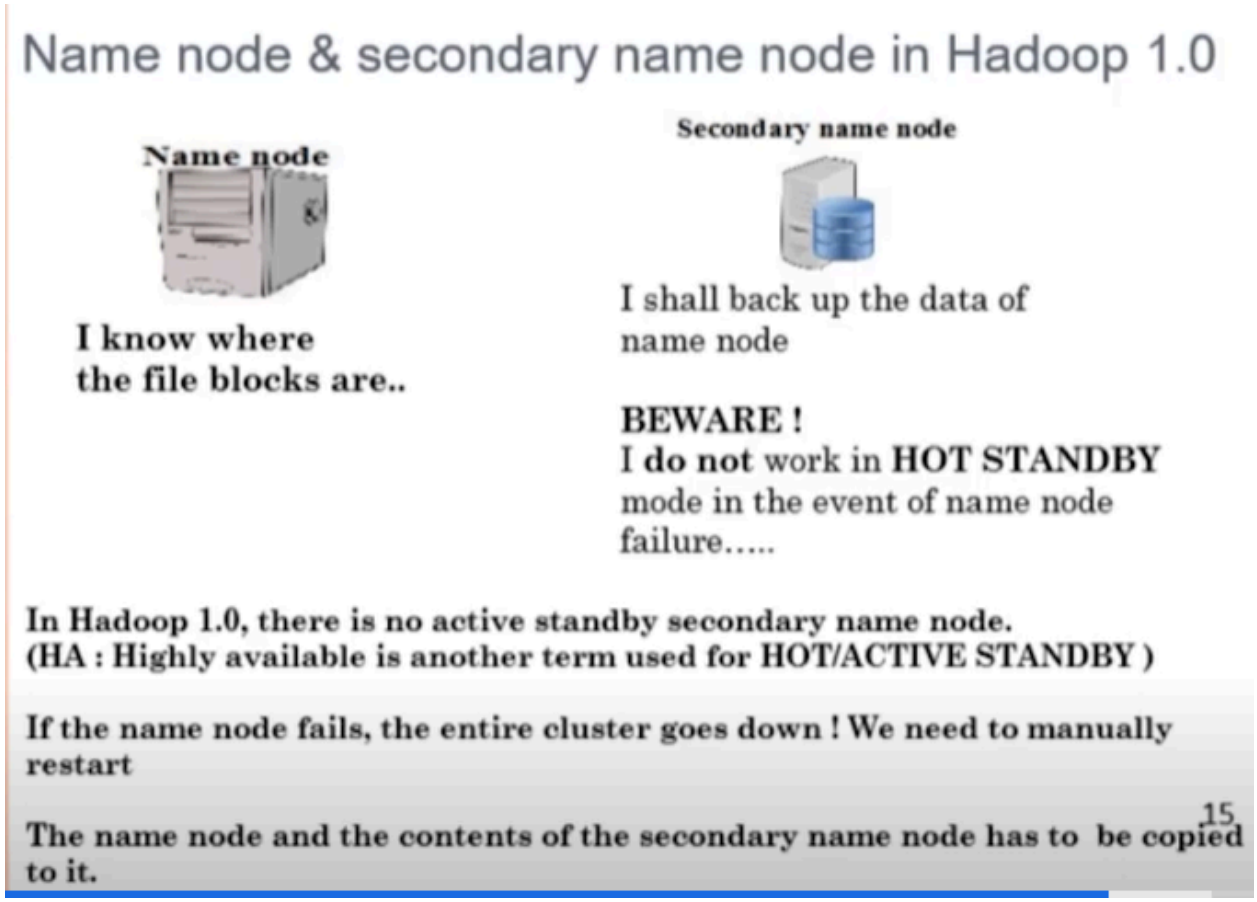


26. This will replication will be done internally..no need for us to intervene

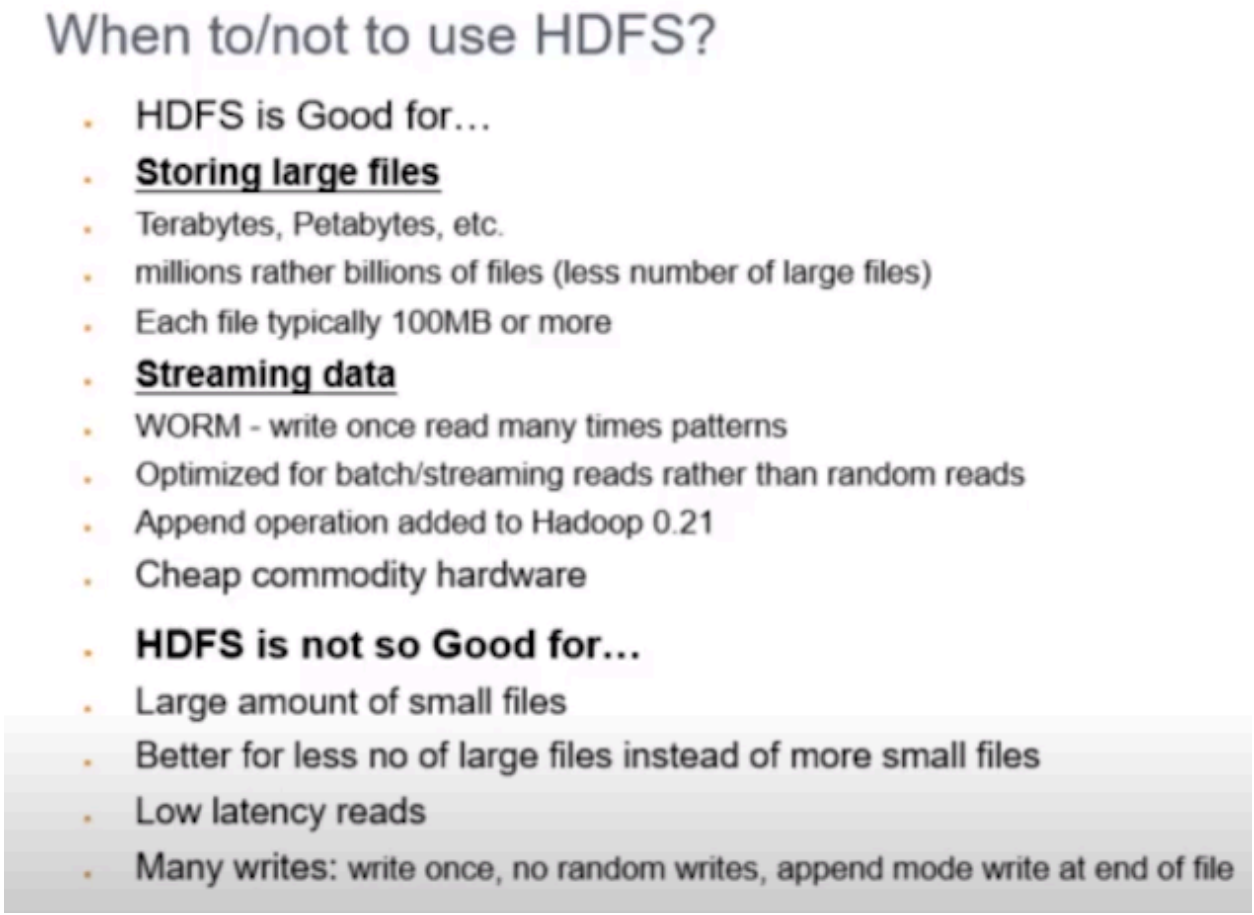
27. Replica Placement Strategy



28. Yellow is our original data and blue are copies of original data
29. Now here we have 2 racks and data replications always happen on adjacent racks. So if data node at one rack fail..it can easily get backs up in short time
30. Each rack has switch networks ..which racks use for communication
31. Name node and secondary name node in hadoop 1.0

32. 

33. Now if the namenode fails..the data in datanodes will not get deleted…but we don't know the location of our data blocks.
34. We use secondary name nodes in case of namenode failures
35. In the event of namenode failure…the admin will copy the back up of FSImage from secondary NN and will restore the namenode
36. This problem is fixed in Hadoop 2.0 where there will be hotstandby incase of NN failure

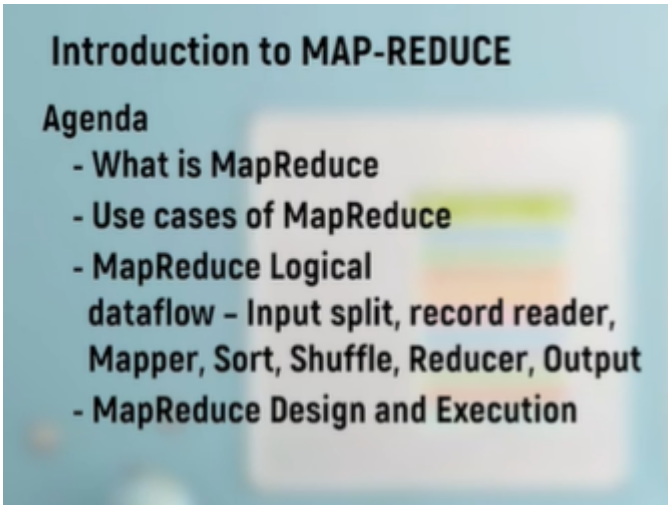
37. 

38. HDFS PPT : https://olympus.mygreatlearning.com/courses/10977/modules/items/449012

Working on HDFS

1. Just follow video(it shows VM ware installation and HDFS usage on linux)

MapReduce: A Programming paradigm



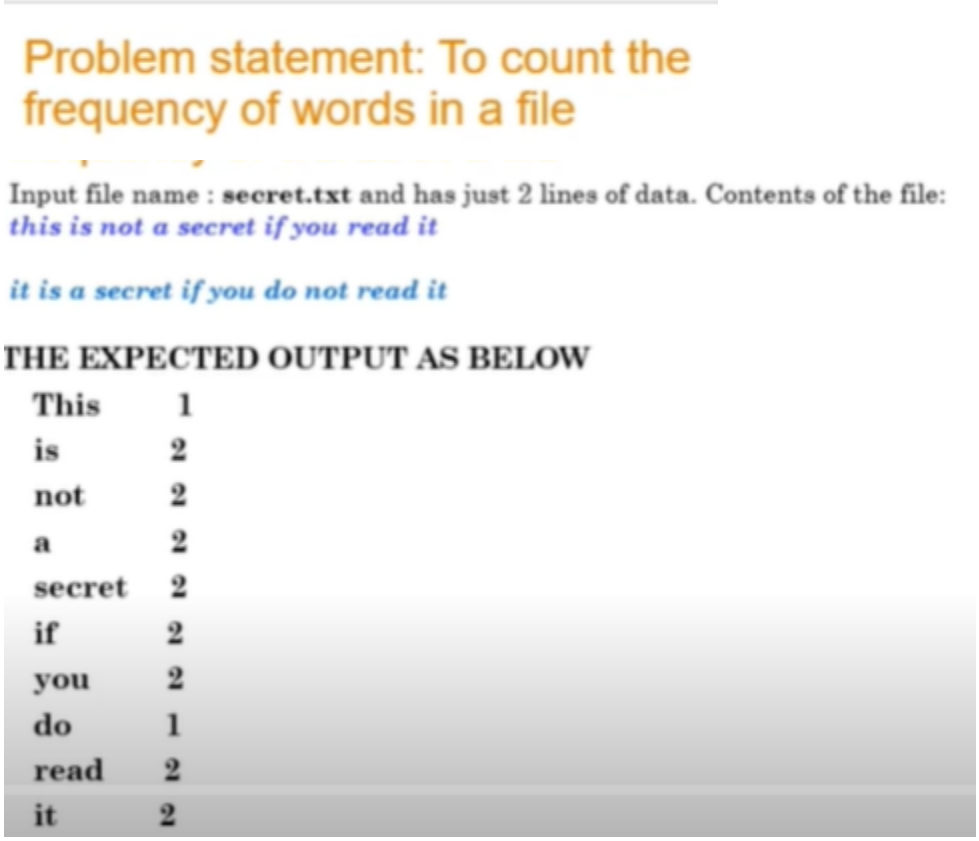
1. We'll learn this topics in this video
2. Apache has 2 important components in its ecosystem

HDFS (storage) - stores data in chunks by splitting it into 64MB each block, is from the "infrastructural" point of view

MapReduce (processing) - is from the "Programming" aspect

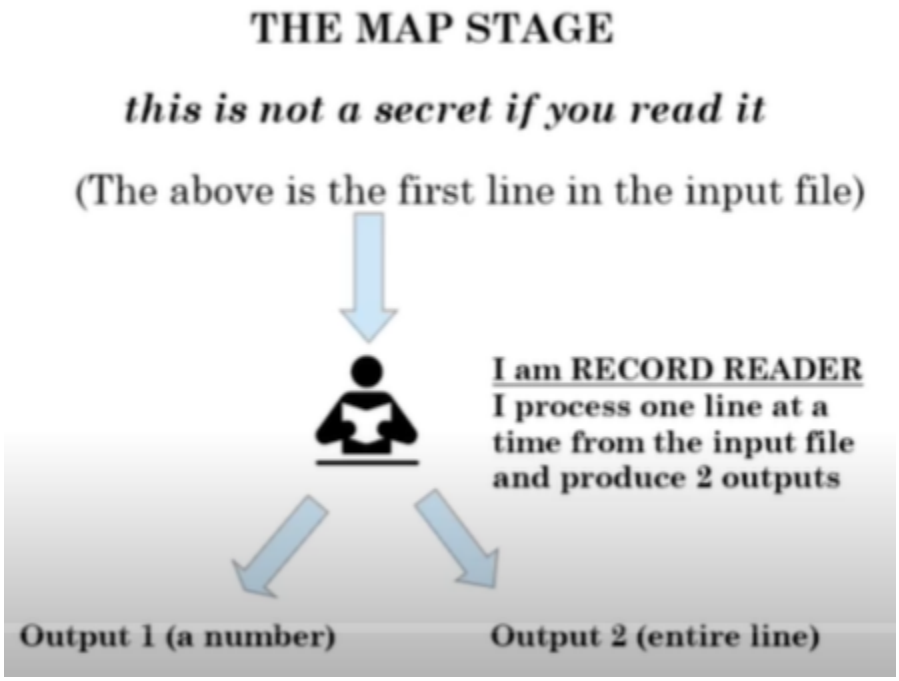
- A Java framework for processing parallelizable problems across huge datasets, using commodity hardware, in a distributed environment
- Google has used it to process its "big-data" sets (~ 20,000 PB/day)
- Can be implemented in many languages: Java, C++, Ruby, Python etc.

3. Lets consider this problem statement

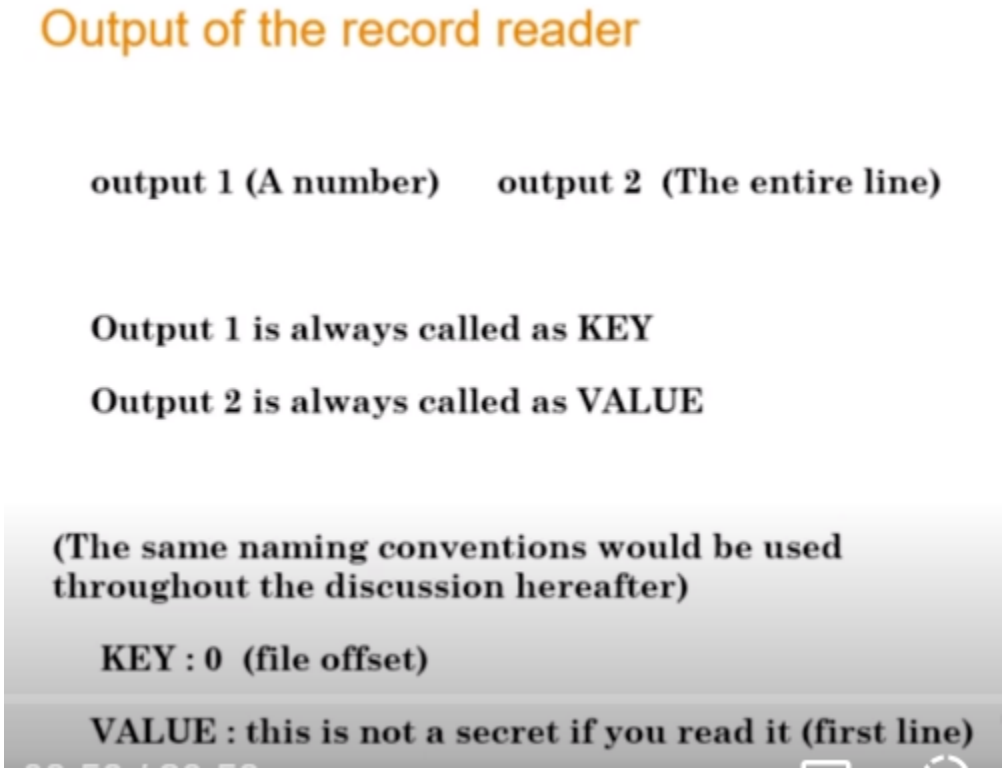


4. We cna easily solve this problem using c,c++ and py
5. But what if we have billions of lines?
6. To solve this we need to approach via parallel processing..or we can also use map reduce approach
7. Map & Reduce

8. Here the first stage is map stage(Record Reader)

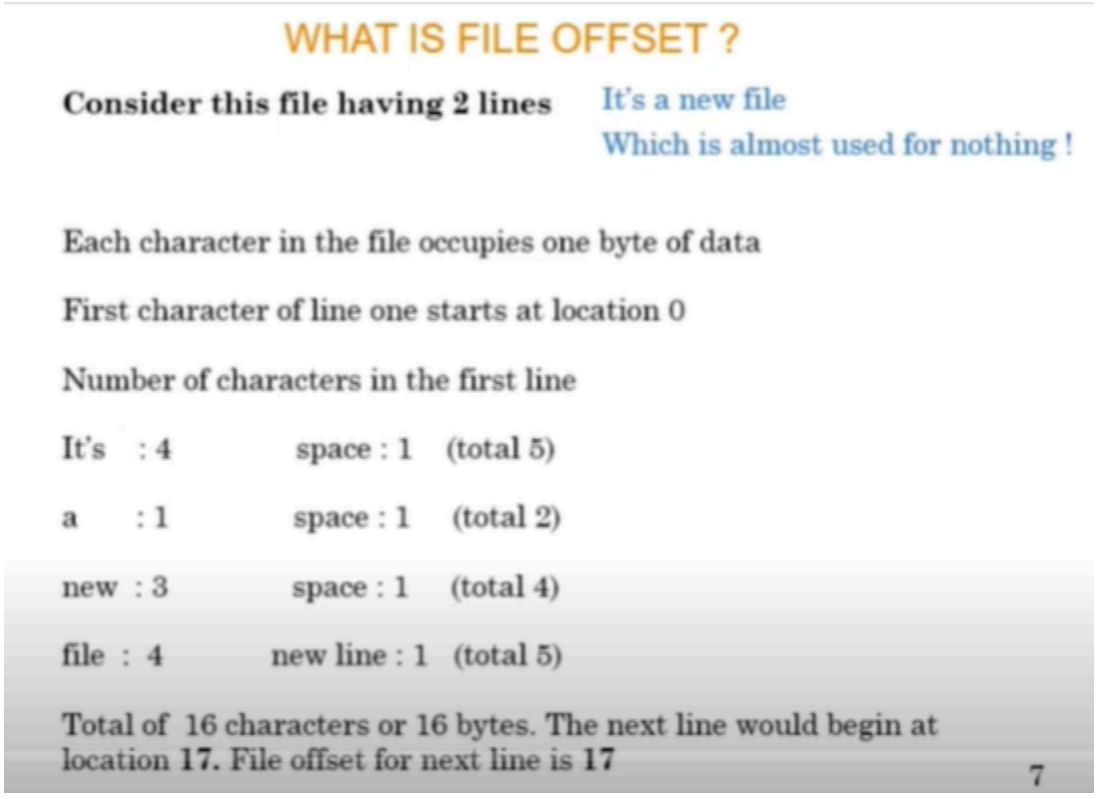


9. Here we consider

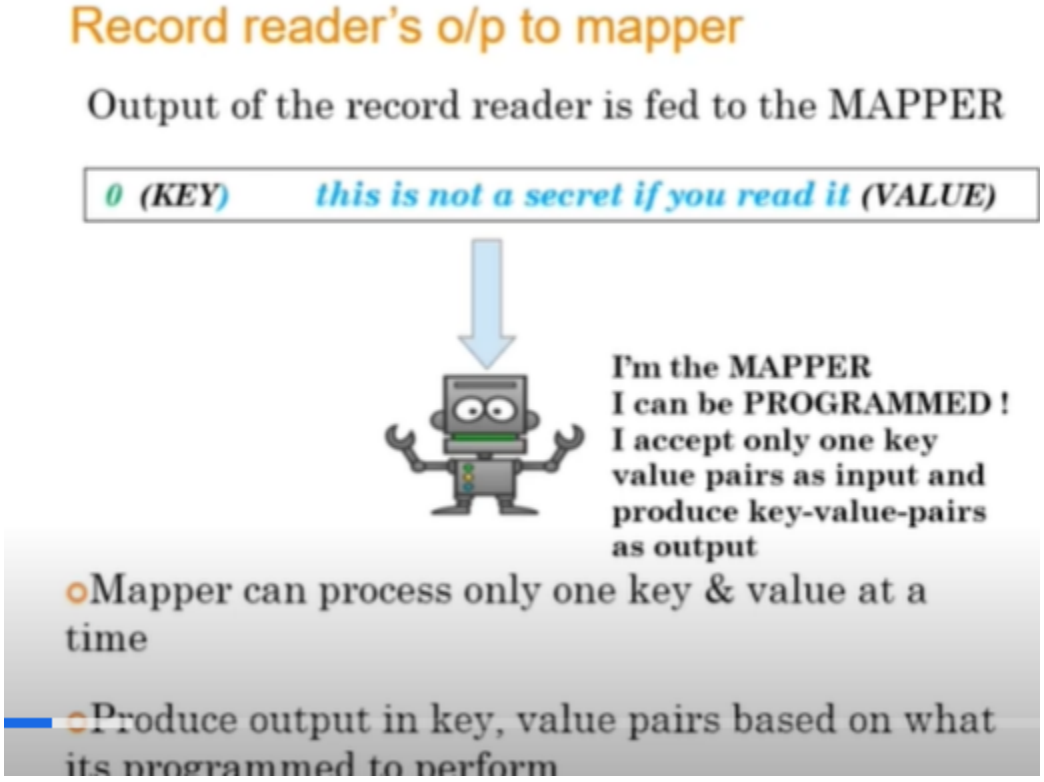


10. Here key : 0 represents the starting location of first line in our file

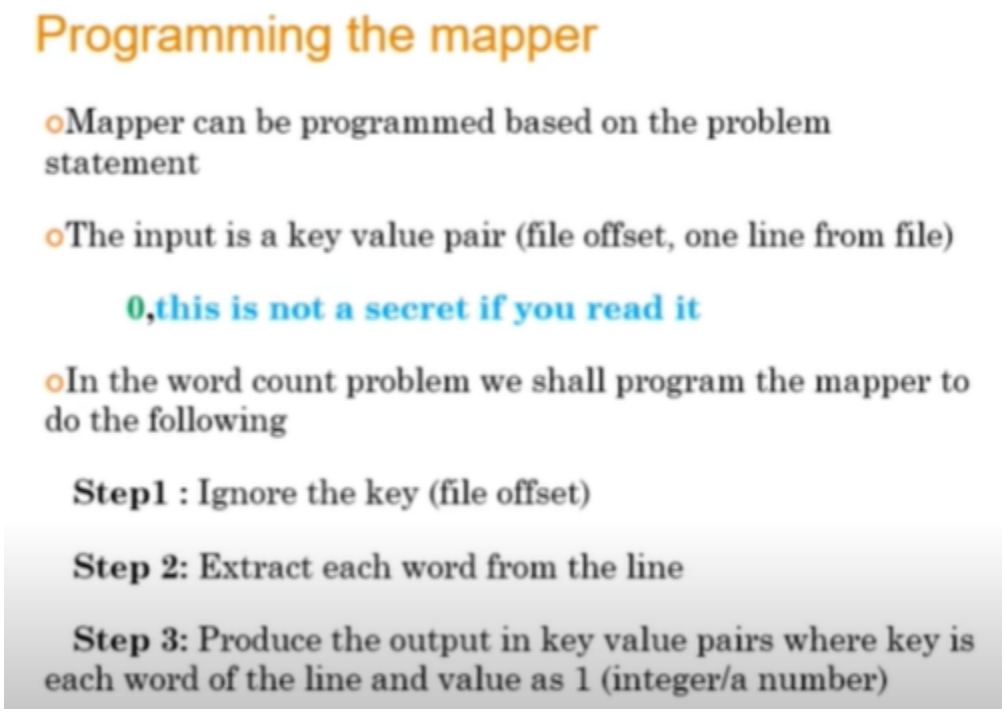
11. What is file offset?



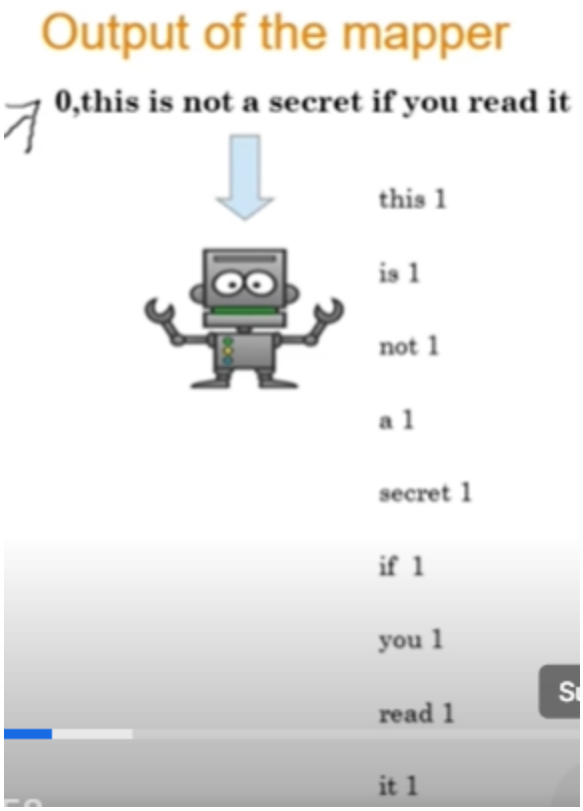
12. From the above image we can say that line 2(starts with which) has file offset of 17
13. File offset is obtained by adding up characters of previous lines including space



14.

15. Record reader will produces 2 outputs ..one is key and the other one is value
16. Here key is our file offset and value is our line
17. We have mapper which takes one key-value pair as input and produces key-value-pairs
18. 

19. Here we designed mapper to produce word_count by tokenizing it

20. 

21. The output of the mapper will look like this

22. Next the output of mapper will undergo sort operation

The sort operation

(This happens in the memory of the mapper machine)

Output of the mapper is fed into the **sorter** which sorts the mapper output in ascending order of the KEYS ! (lexicographic ordering or dictionary ordering since the keys are of string type)

Consider the simple example

1

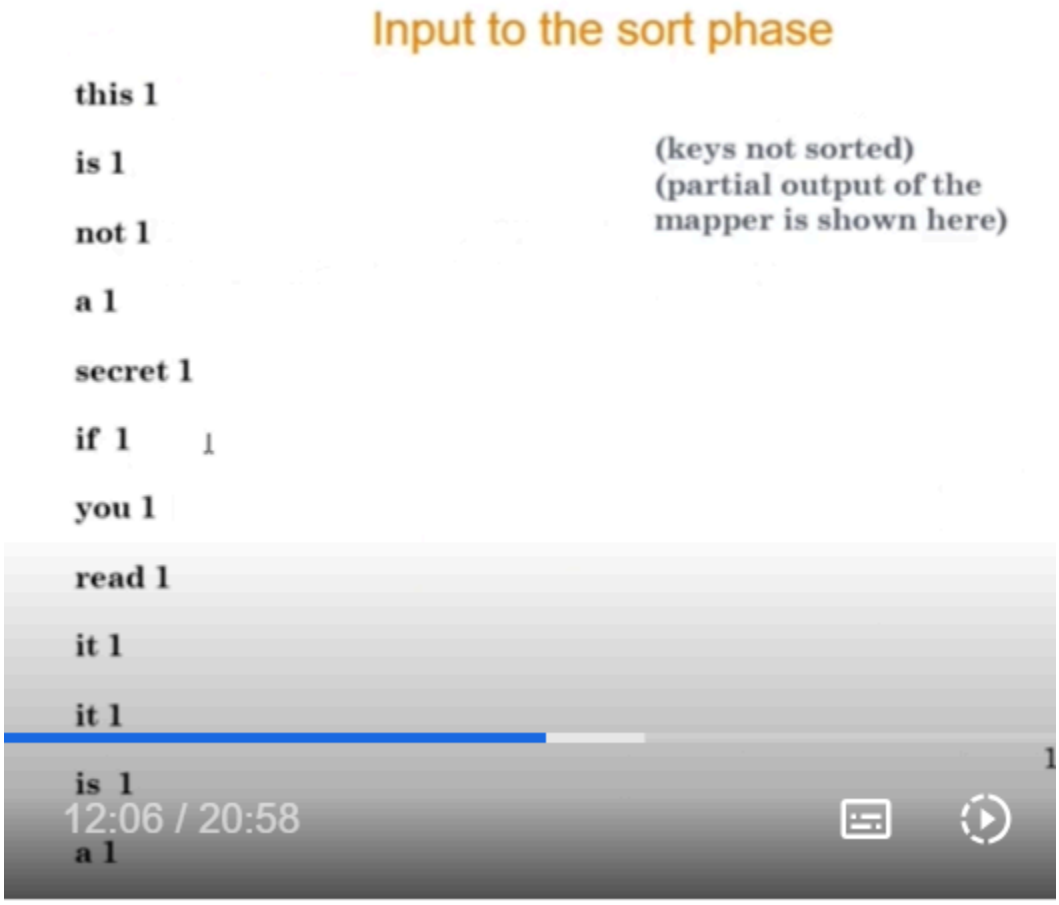
SORTER

Note : Sorter can be reprogrammed (overridden) to sort based on values if required. Its called the sort comparator.

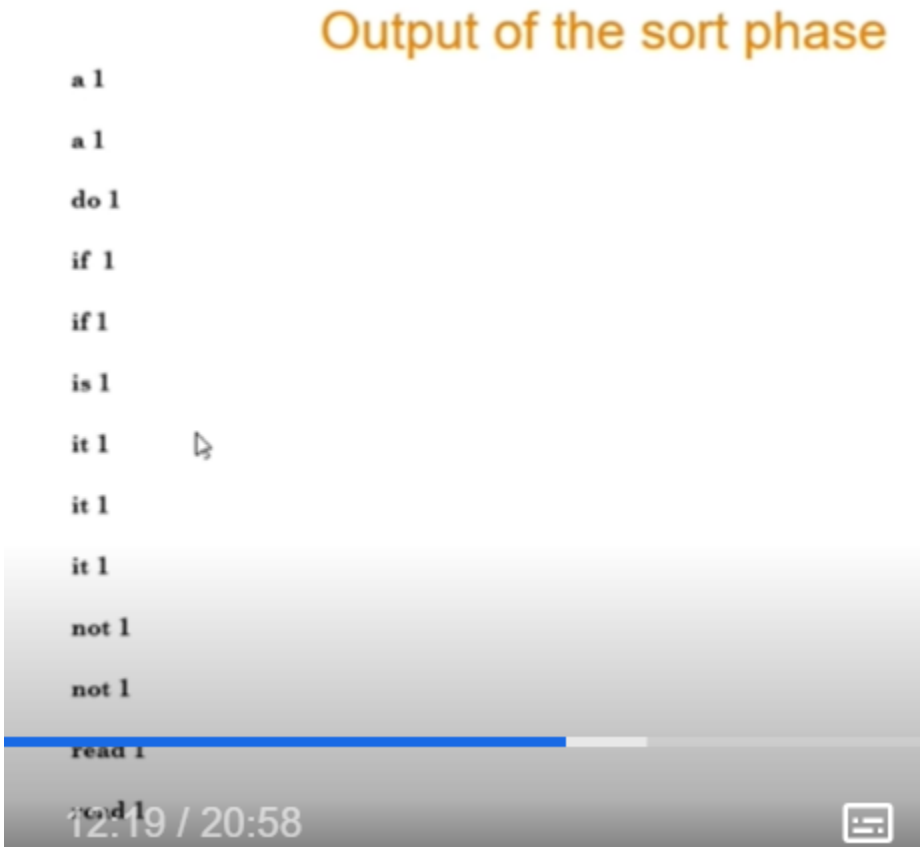
11

23. Here the input to the sort phase is

24. This is the output of our mappers of first two lines



25. The output of the sort phase is(sorts based on keys)



26. After the merge operation..we have shuffle operation

Shuffle/aggregate phase in REDUCE stage

- Shuffling is a phase where duplicate keys from the input are aggregated.

Consider the simple example

- Input is key value pairs (**contains duplicate keys**)
- Output is a set of key value pairs **without duplicates**

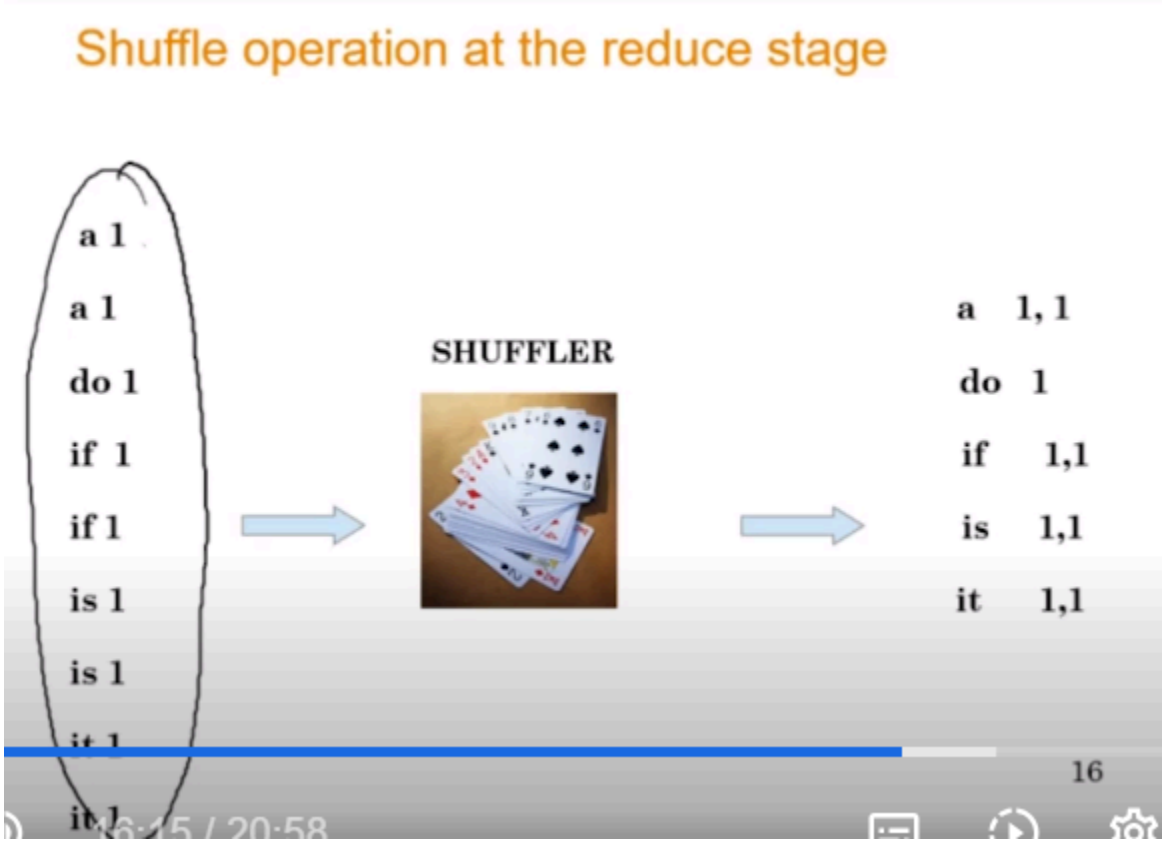
Key	value
Apple	2
Apple	4
Mango	1
Orange	11
Orange	06

SHUFFLER

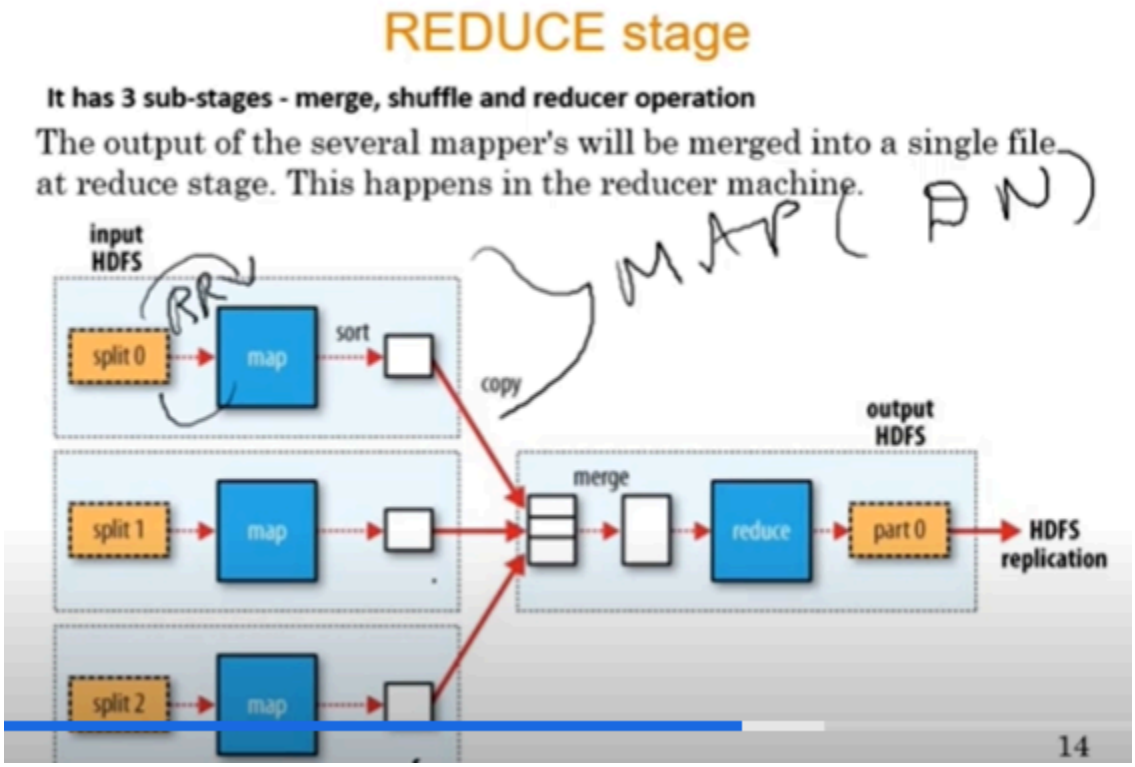
Key	value
Apple	2, 4
Mango	1
Orange	11, 06

15

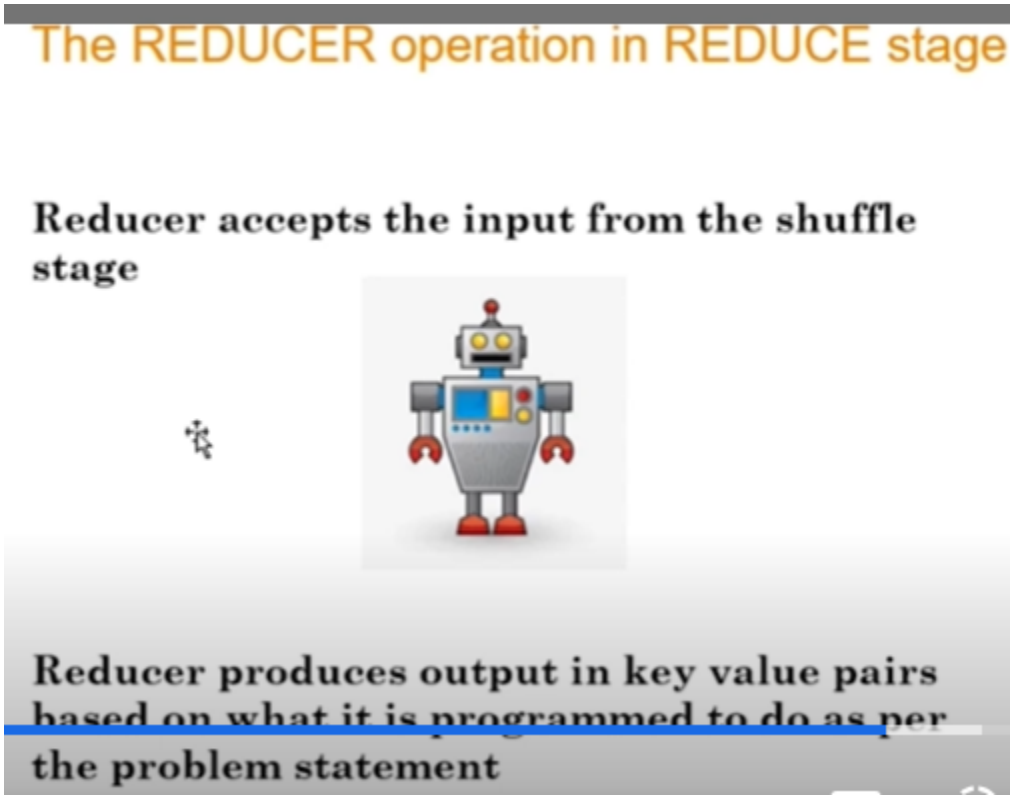
27. IF we have this data then the output of shuffle will look like this



28. Reduce stage



29. Here there are 3 DN processing the mapper and sort functions and they will produce a result
30. We have another DN which has a reducer installed. Now the output from the DN's will be pushed to Reducer node..where they will merge first
31. The output of shuffle will fed to Reducer



Programming the reducer

Output of the shuffle is the input to the REDUCER

Reducer can handle only one key value pair at a time

Step1: Input to the reducer is **a 1, 1**

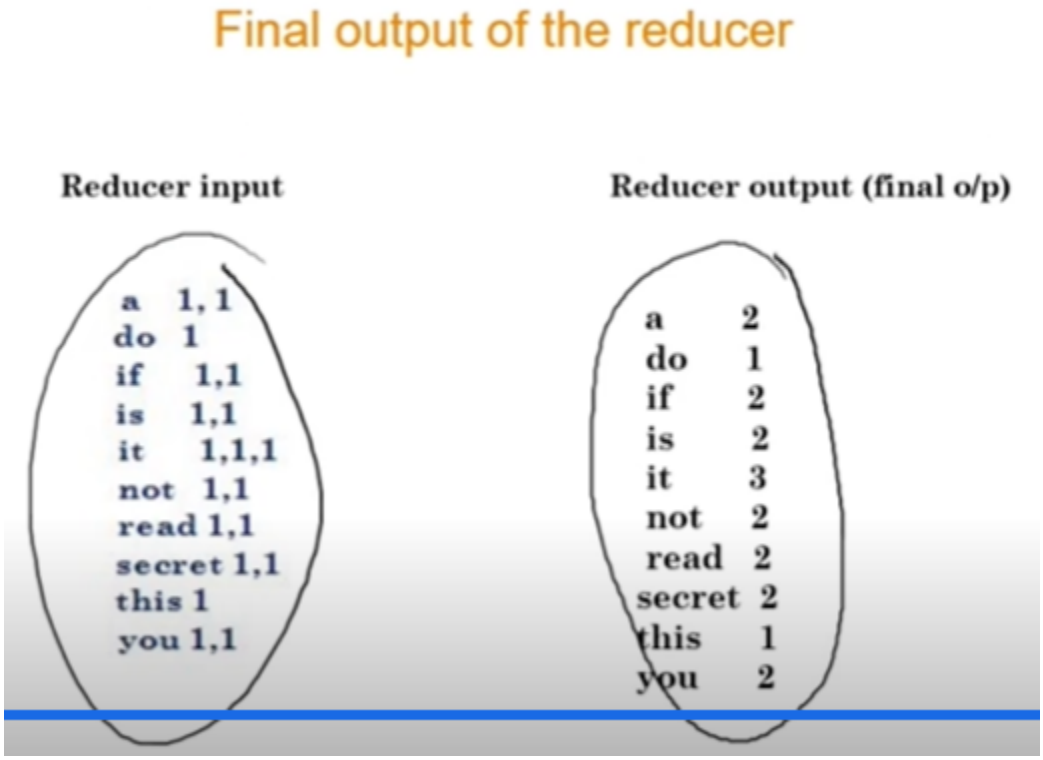
Step2: The reducer must add the list of values from the input i.e sum=1+1 = 2

Step3: Output the key and sum as output key, value pairs to an output file. The o/ would look like **a 2**

Step 4: Repeat the above operations (1,2&3) for entire input

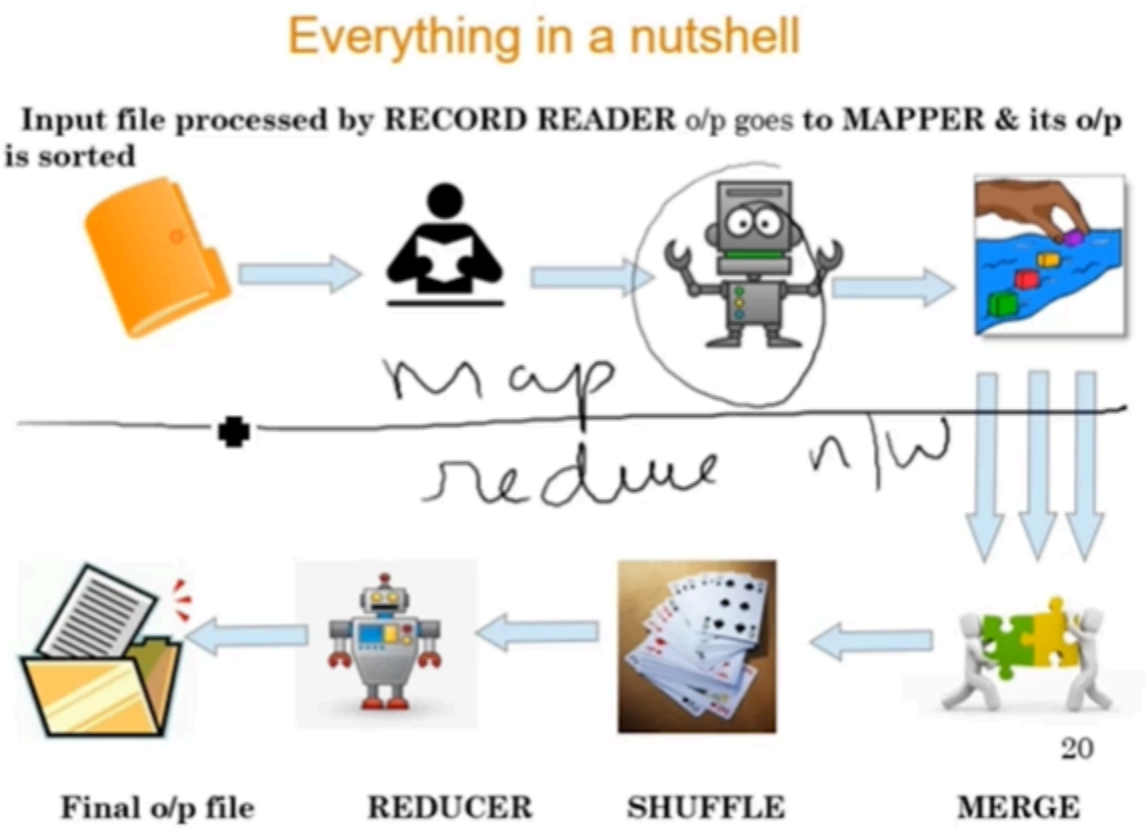
32.

33. What reducer does is



34. Follow ppt for correct order of slides

35. In a nutshell



36. Here the input to map reduce comes from the hdfs and output write to hdfs as well
37. Ppt: https://olympus.mygreatlearning.com/courses/10977/modules/items/449017