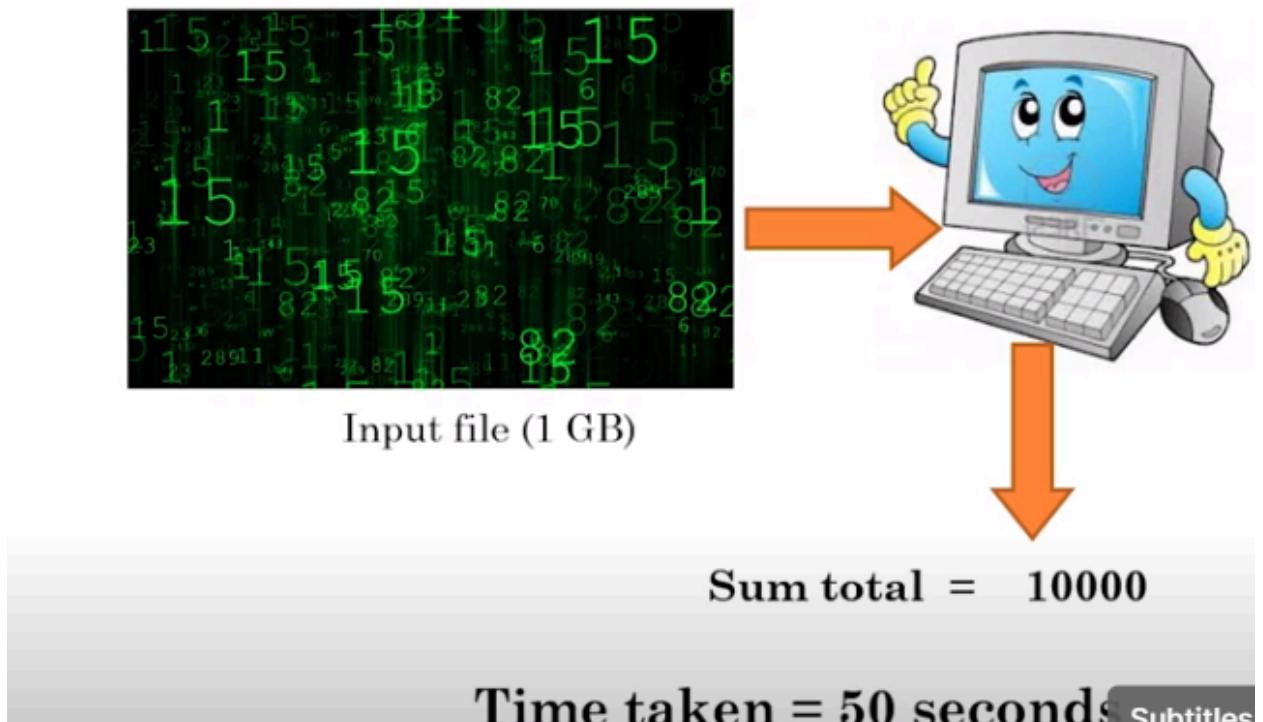


Getting started : Hadoop

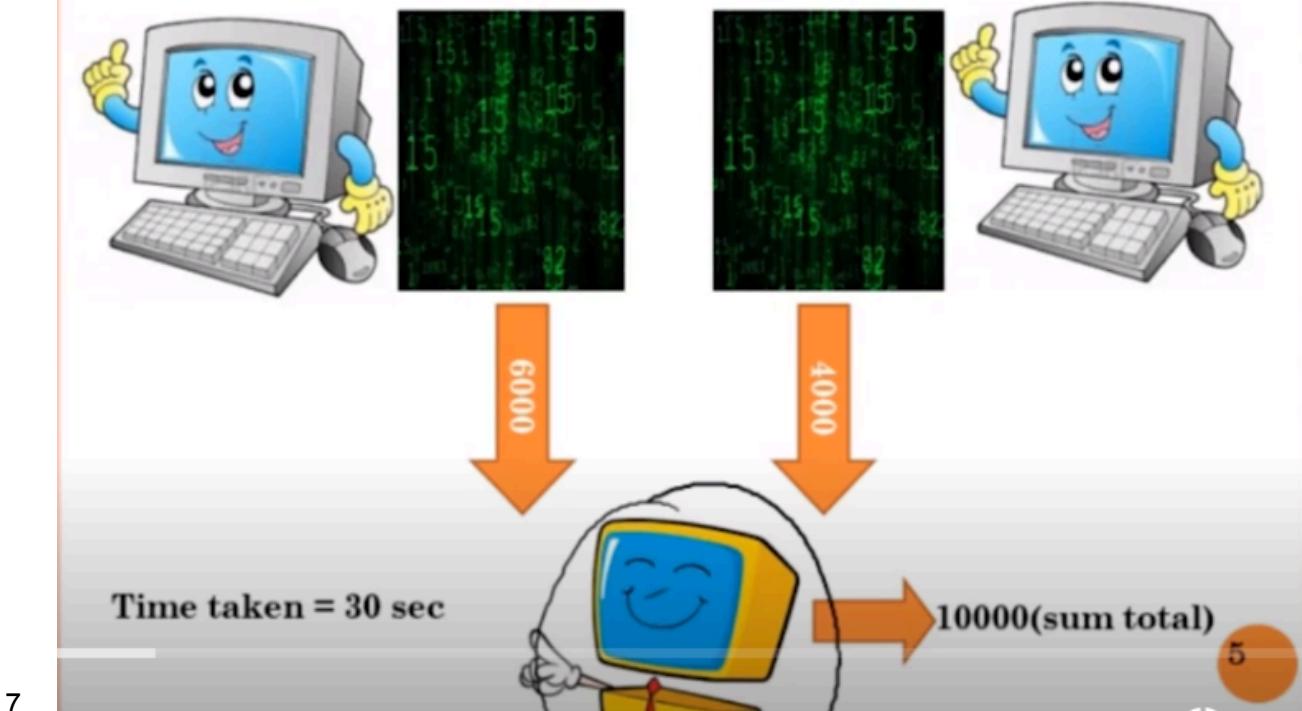
1. Lets take a living example(paint job)
2. Here each wall needs 2 hrs to get painted by a human
3. So what if we need to get all the 5 walls painted in 3hrs? ..we deploy more workers. It is case of problem solving
4. Lets take an another example

Adding the Numbers



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

Parallel Processing – Faster computing !



- 7.
 8. Now what if we divide our file and feed 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

SUPER COMPUTER – cluster of computers



IBM.

FUJITSU

CRAY
THE SUPERCOMPUTER COMPANY



11. 04:45 / 14:03 https://en.wikipedia.org/wiki/History_of_supercomputing

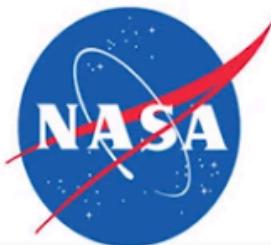
12. What is the need and uses of super computer?

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.



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 by 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 computing 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 commodity hardware
- The software challenges of the organization having to write proprietary softwares is no longer the case.

15. Where was the idea of hadoop generated?

Learning

Late 90's to Early 2000's – dot com Bubble !

- This was the age of the INTERNET BOOM
- The need of the hour was a scalable web search engine
- Major players in this business of internet search engines back then were



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

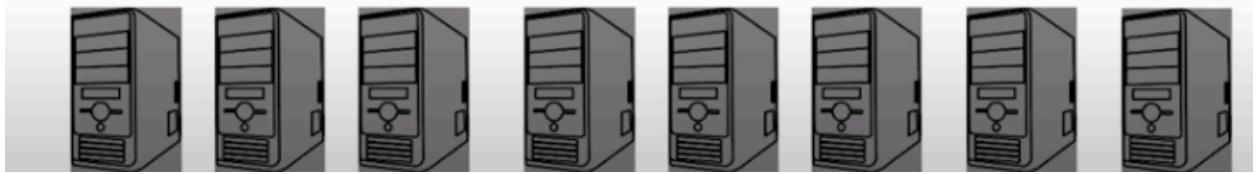
Master-Slave Architecture

Machines in MASTER MODE

Name node Secondary name node JOB_tracker



Data nodes in slave mode



- 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.

10.

5

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 hadoop 2.0 we have a hot standby in the event of failures
15. And the datanodes will be powerful than namenodes

What is a job in HADOOP eco system?

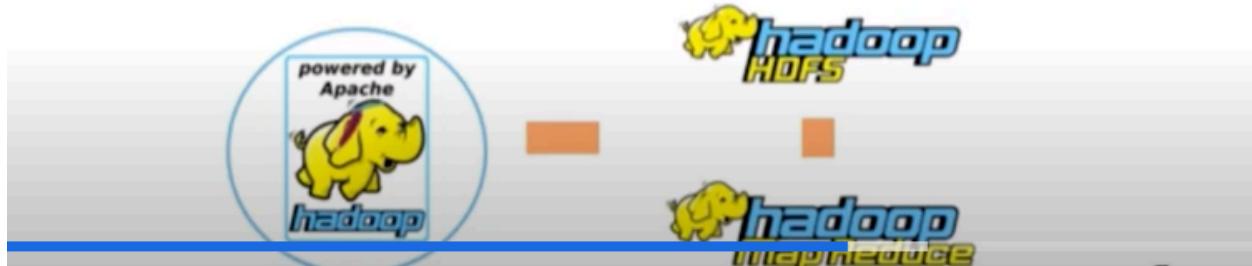
- A job usually is some **task** submitted by the user to the Hadoop cluster.
- The job is in the form of a **program or collection of programs** (a JAR file) which needs to be executed.
- A job would have the following attributes to it
 - 1)The actual program/s
 - 2)Input data to the program (a file or a collection of files in a directory)
 - 3)The output directory where the results
of execution is collected in a file/s

17.

Apache HADOOP Core Features

Core features of Apache Hadoop are:

- .HDFS (Hadoop Distributed File System) – data storage
- .MapReduce Framework – compute in distributed environment
 - A Java framework responsible for processing jobs in distributed mode
 - User-defined map phase, which is a parallel, share-nothing processing of input
 - User-defined reduce phase aggregates of the output of the map phase

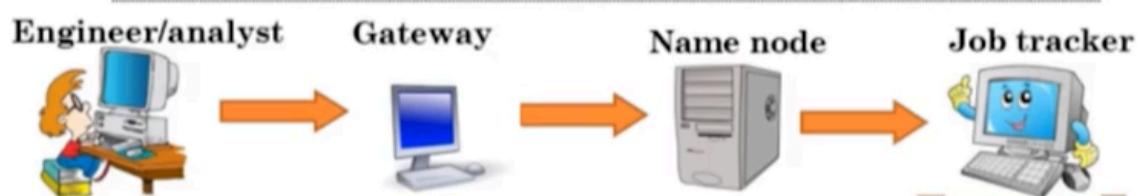


18. Hadoop comes with HDFS ..hadoop used HDFS for storing and retrieving the data

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

Submitting and executing a job in a hadoop cluster



The engineer/analyst's machine is not a part of Hadoop cluster. Usually Hadoop would be installed in pseudo-distributed mode on his/her machine. The job (program/s) would be submitted to the gateway machine

The gateway machine would have the necessary configuration to communicate to the name node and job tracker.

Job gets submitted to the name node and eventually job tracker is responsible for scheduling the execution of the job on the data nodes in the cluster.



HDFS: What and Why?

Agenda

Architecture of HDFS

How does HDFS store the file internally

Failure handling and recovery mechanism

Rack awareness

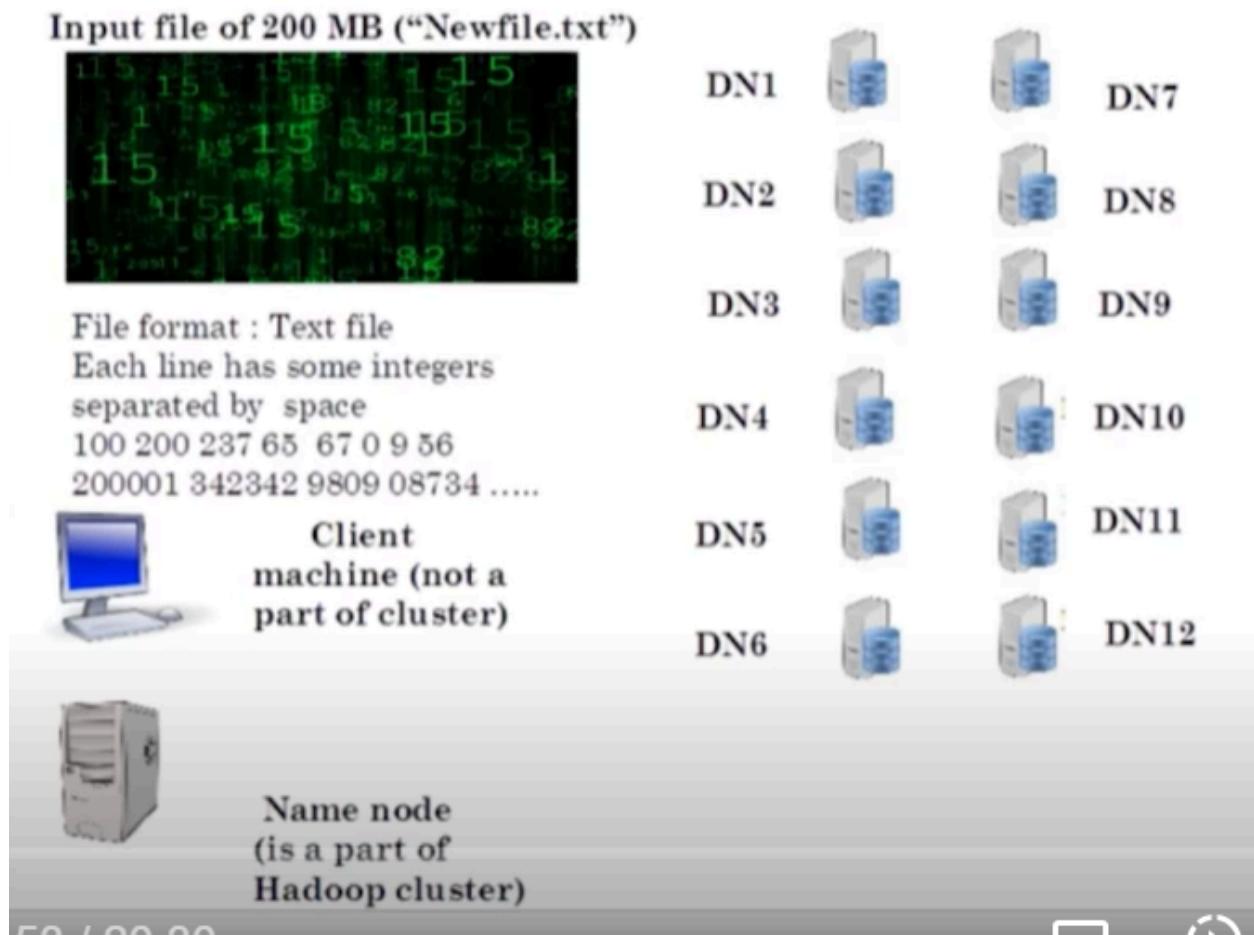
Role of name node and secondary name node

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. However 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

HDFS – The Storage Layer in Hadoop



Splitting of file into blocks in HDFS

Default split size is 64MB(It can be changed)

Original File size is 200MB. 200Mb file is

Split into 4 blocks of N1, N2, N3 and N4

The block N4 is just 8 MB ($200 - 64 \times 3$)

Each block is now a separate FILE & DN

N1, N2, N3 and N4 are file names.



- 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

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 !

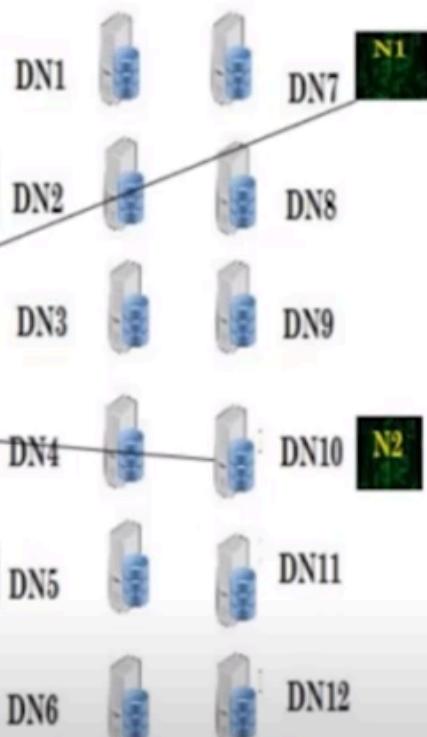


Client machine



Name node

Client machine directly writes the files to the data nodes once the name node provides the details about data nodes.



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:

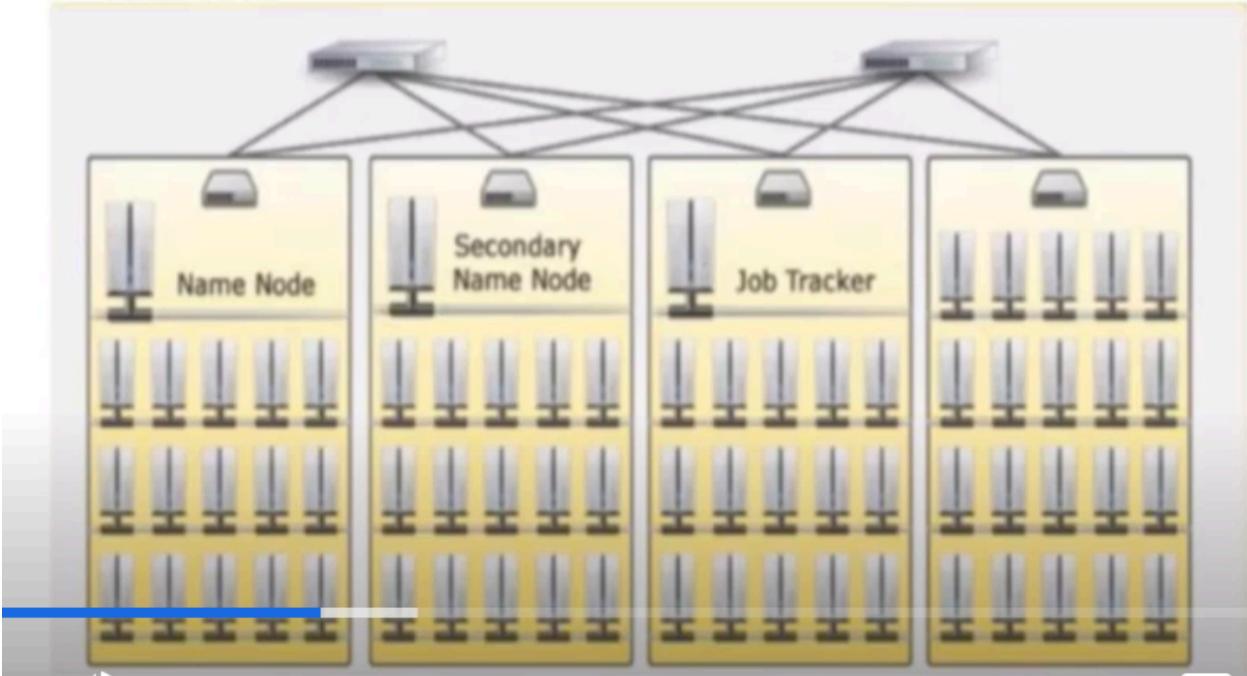
- HDFS replicates blocks across multiple DataNodes for data protection and availability.
- The NameNode strategically places replicas across different racks in a data center to prevent data loss if a rack fails.
- The default replication factor is 3, meaning each block is stored on 3 different DataNodes.

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

Hadoop cluster deployed in a production environment into multiple racks

Multilayer switches/routers interconnect the switches on each rack



- 11.
12. Failure of a datanode

13. Here lets assume datanode 10 has failed

Failure of a data node

Q)What happens in the event of a data node

Failure ? (eg : DN 10 fails)

A)Data saved on that node will be lost

To avoid loss of data, copies of the Data blocks on data node is stored on multiple data nodes. This is called data replication.



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)

Replication of data blocks

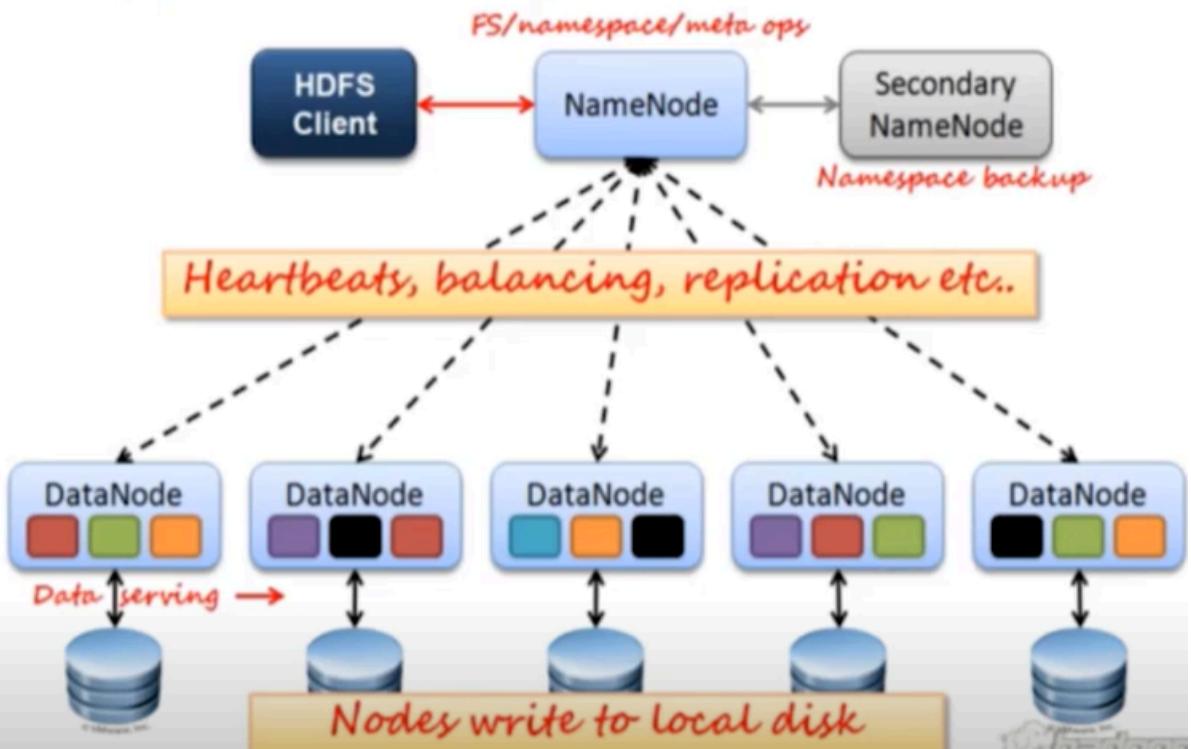
- **How many copies of each block to save?**

Its decided by REPLICATION FACTOR (by default its **3**, i.e. every block of data on each data node is saved on 2 more machines so that there is total 3 copies of the same data block on different machines)

This replication factor can be set on per file basis while the file is being written to HDFS for the first time.

- 15.
16. We can also configure our replication factor

Design and Architecture Overview



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

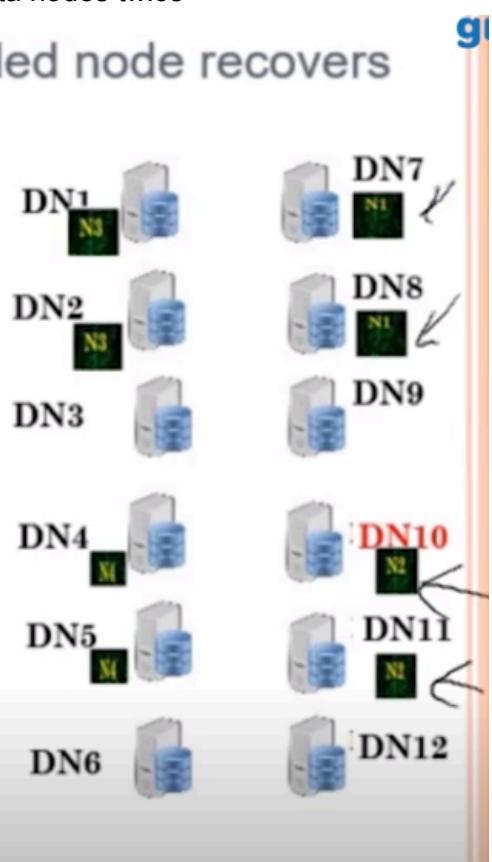
22. Lets assume our RF = 2 and our block has replicated in data nodes twice

Data replication on failure and failed node recovers

Replication factor =2

DN10 fails

Replication factor for block N2 is now 1 !



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

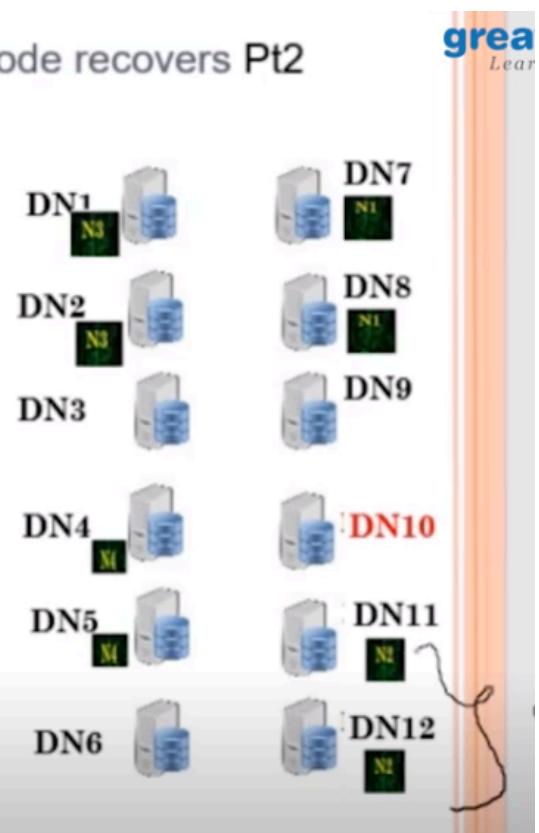
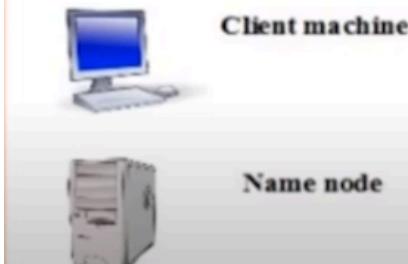
which is close to DN11

Data replication on failure and failed node recovers Pt2

Replication factor =2

DN10 fails

The data on the failed node would be copied to some other node in the cluster automatically. In this case its copied to DN12 from its nearest neighbour DN11



24. But what is DN10 come back from failure

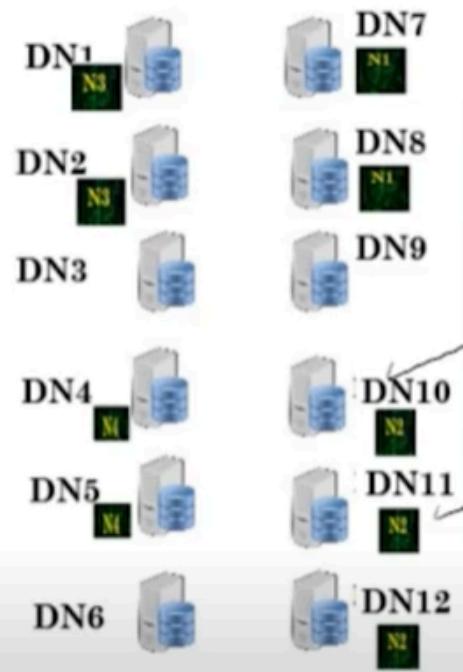
25. Now the replication factor becomes 3

DN10 is up again after some time ...

Replication factor = 3 for block N2

HDFS will delete one extra copy of N2 from any of the 3 nodes (DN10, DN11 or DN12)

This ensures that the replication count is maintained all the time



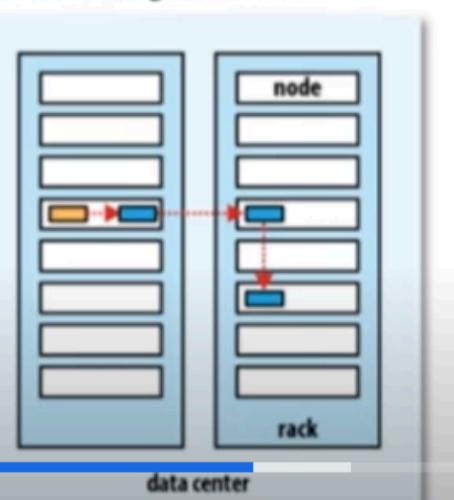
26. This will replication will be done internally..no need for us to intervene

27. Replica Placement Strategy

Replica Placement Strategy

Q. How does namenode choose which datanodes to store replicas on?

- Replica Placements are rack aware. Namenode uses the network location when determining where to place block replicas.
- Tradeoff: Reliability v/s read/write bandwidth e.g.
 - If all replica is on single node - lowest write bandwidth but no redundancy if nodes fails
 - If replica is off-rack - real redundancy but high read bandwidth (more time)
 - If replica is off datacenter – best redundancy at the cost of huge bandwidth
- Hadoop's default strategy:
 - 1st replica on same node as client
 - 2nd replica on off rack any random node
 - 3rd replica is same rack as 2nd but other node
- Clients always read from the nearest node
- Once the replica locations is chosen a pipeline is built taking network topology into account



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

Name node & secondary name node in Hadoop 1.0



I know where
the file blocks are..

Secondary name node



I shall back up the data of
name node

BEWARE !

I do not work in HOT STANDBY
mode in the event of name node
failure.....

In Hadoop 1.0, there is no active standby secondary name node.
(HA : Highly available is another term used for HOT/ACTIVE STANDBY)

If the name node fails, the entire cluster goes down ! We need to manually
restart

The name node and the contents of the secondary name node has to be copied
to it.

- 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

When to/not to use HDFS?

- HDFS is Good for...
- **Storing large files**
 - Terabytes, Petabytes, etc.
 - millions rather billions of files (less number of large files)
 - Each file typically 100MB or more
- **Streaming data**
 - WORM - write once read many times patterns
 - Optimized for batch/streaming reads rather than random reads
 - Append operation added to Hadoop 0.21
 - Cheap commodity hardware
- **HDFS is not so Good for...**
 - Large amount of small files
 - Better for less no of large files instead of more small files
 - Low latency reads
 - Many writes: write once, no random writes, append mode write at end of file

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

Introduction to MAP-REDUCE

Agenda

- What is MapReduce
- Use cases of MapReduce
- MapReduce Logical
dataflow - Input split, record reader,
Mapper, Sort, Shuffle, Reducer, Output
- MapReduce Design and Execution

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

Problem statement: To count the frequency of words in a file

Input file name : **secret.txt** and has just 2 lines of data. Contents of the file:
this is not a secret if you read it

it is a secret if you do not read it

THE EXPECTED OUTPUT AS BELOW

This	1
is	2
not	2
a	2
secret	2
if	2
you	2
do	1
read	2
it	2

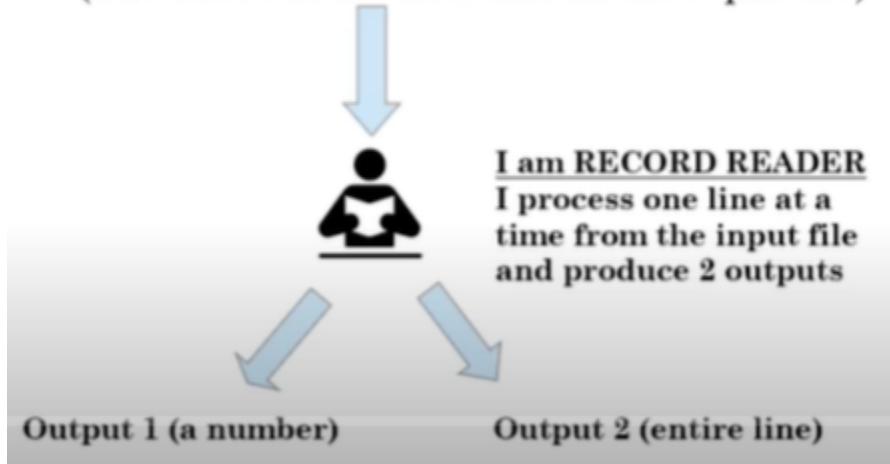
4. We can easily solve this problem using c,c++ and py
5. But what if we have billions of lines?
6. To solve this we need to approach via parallel processing..or we can also use map reduce approach
7. Map & Reduce

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

THE MAP STAGE

this is not a secret if you read it

(The above is the first line in the input file)



9. Here we consider

Output of the record reader

output 1 (A number) output 2 (The entire line)

Output 1 is always called as KEY

Output 2 is always called as VALUE

(The same naming conventions would be used throughout the discussion hereafter)

KEY : 0 (file offset)

VALUE : this is not a secret if you read it (first line)

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

11. What is file offset?

WHAT IS FILE OFFSET ?

Consider this file having 2 lines It's a new file
Which is almost used for nothing !

Each character in the file occupies one byte of data

First character of line one starts at location 0

Number of characters in the first line

It's : 4 space : 1 (total 5)

a : 1 space : 1 (total 2)

new : 3 space : 1 (total 4)

file : 4 new line : 1 (total 5)

Total of 16 characters or 16 bytes. The next line would begin at location 17. File offset for next line is 17

7

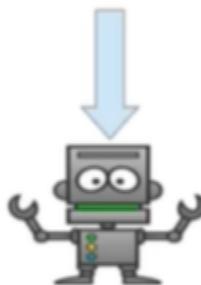
12. From the above image we can say that line 2(starts with which) has file offset of 17

13. File offset is obtained by adding up characters of previous lines including space

Record reader's o/p to mapper

Output of the record reader is fed to the MAPPER

0 (KEY) this is not a secret if you read it (VALUE)



I'm the MAPPER
I can be PROGRAMMED !
I accept only one key
value pairs as input and
produce key-value-pairs
as output

Mapper can process only one key & value at a time

Produce output in key, value pairs based on what its programmed to perform

14.

15. Record reader will produce 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

Programming the mapper

- o Mapper can be programmed based on the problem statement
- o The input is a key value pair (file offset, one line from file)

0, this is not a secret if you read it

- o In the word count problem we shall program the mapper to do the following

Step 1 : Ignore the key (file offset)

Step 2: Extract each word from the line

Step 3: Produce the output in key value pairs where key is each word of the line and value as 1 (integer/a number)

18.

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

Output of the mapper

↗ 0, this is not a secret if you read it



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



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

Input to the sort phase

this 1

is 1

(keys not sorted)
(partial output of the
mapper is shown here)

not 1

a 1

secret 1

if 1 1

you 1

read 1

it 1

it 1

is 1

12:06 / 20:58

a 1



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

Output of the sort phase

```
a 1  
a 1  
do 1  
if 1  
if 1  
is 1  
it 1    ↴  
it 1  
it 1
```

```
not 1
```

```
not 1
```

```
read 1
```

```
12:19 / 20:58
```



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

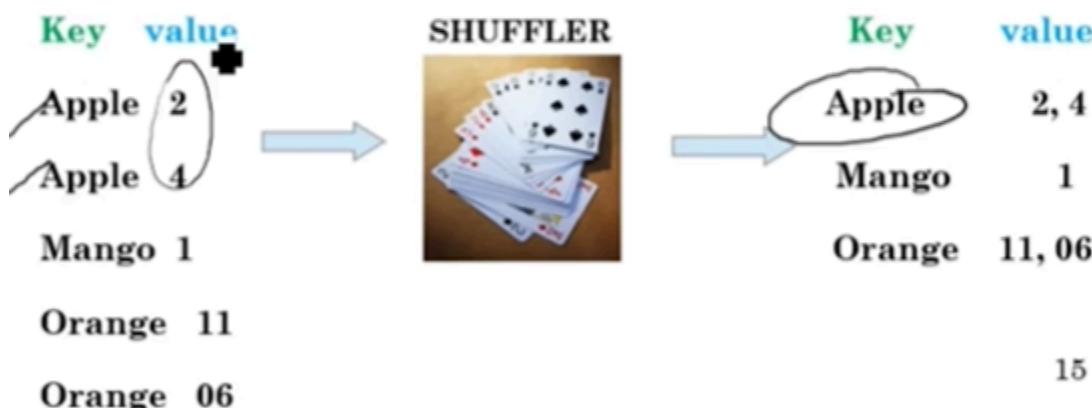
Shuffle/aggregate phase in REDUCE stage

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

Consider the simple example

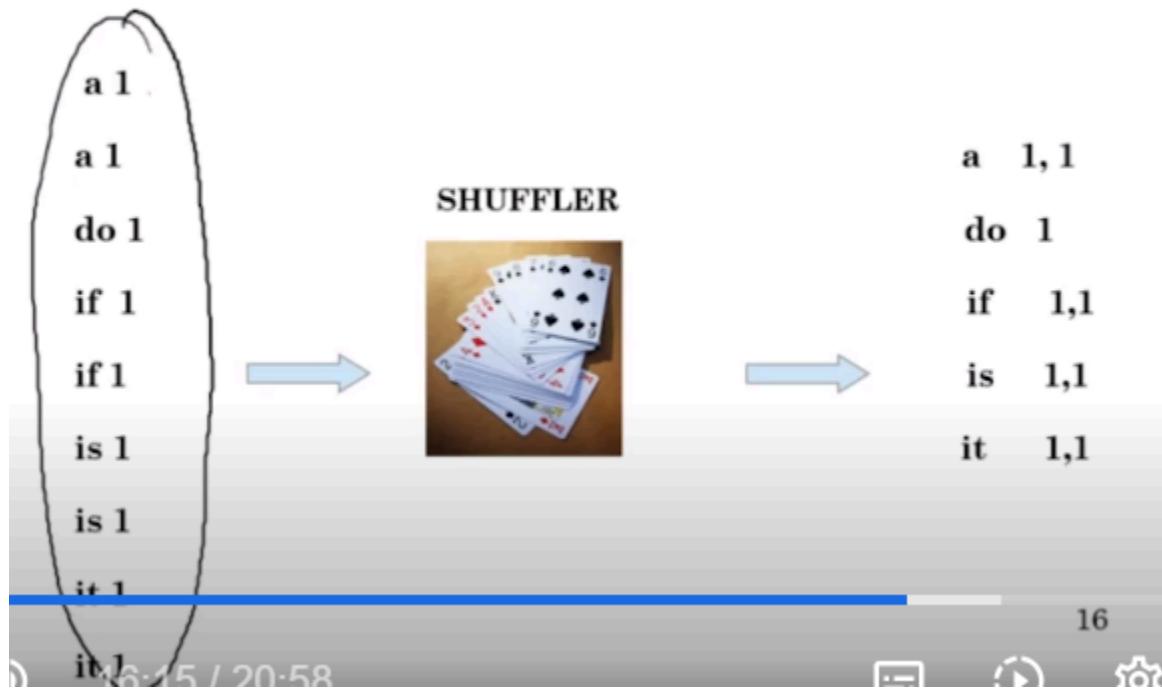
- Input is key value pairs (**contains duplicate keys**)

- Output is a set of key value pairs **without duplicates**



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

Shuffle operation at the reduce stage

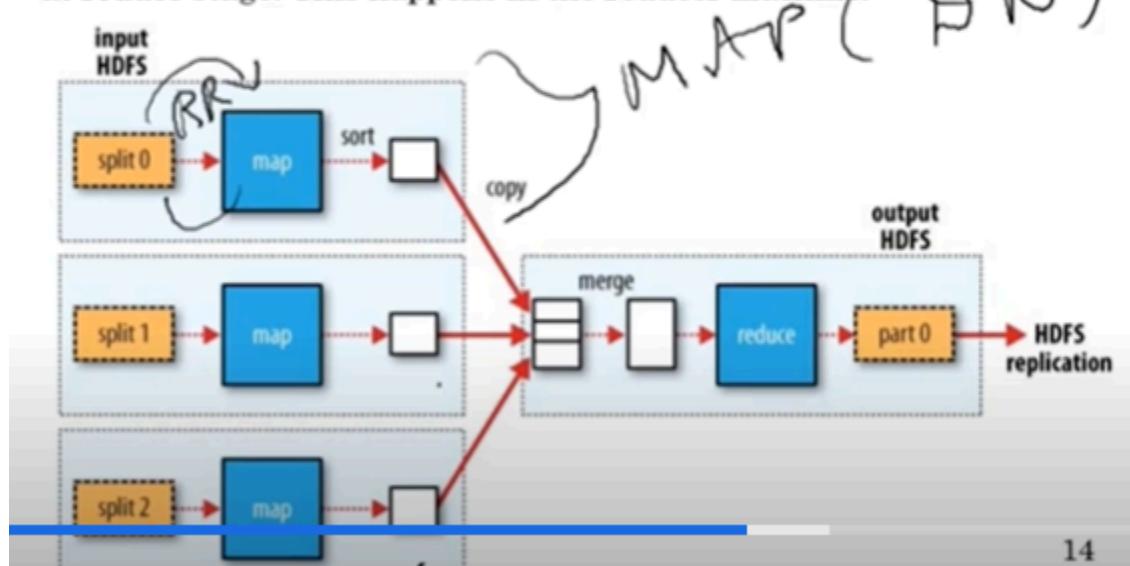


28. Reduce stage

REDUCE stage

It has 3 sub-stages - merge, shuffle and reducer operation

The output of the several mapper's will be merged into a single file at reduce stage. This happens in the reducer machine.



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

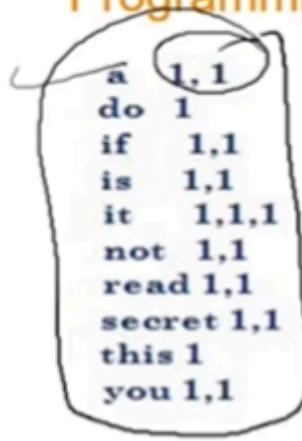
The REDUCER operation in REDUCE stage

Reducer accepts the input from the shuffle stage

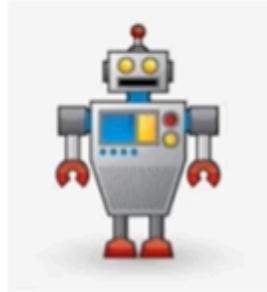


Reducer produces output in key value pairs based on what it is programmed to do as per the problem statement

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

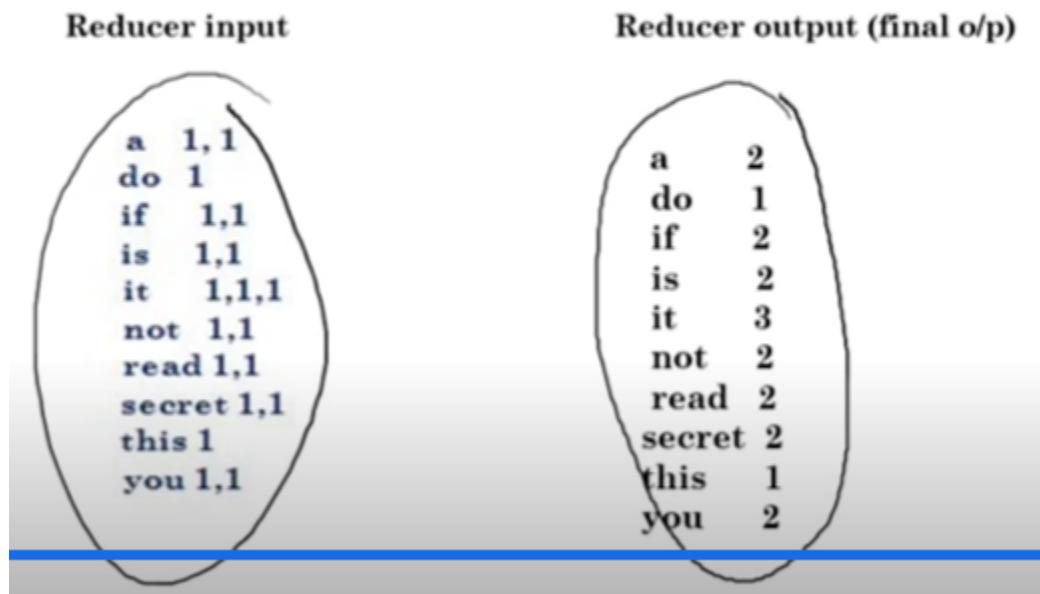
Step3: Output the key and sum as output key, value pairs to an
output file. The o/ would look like (a) 2

18

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

33. What reducer does is

Final output of the reducer

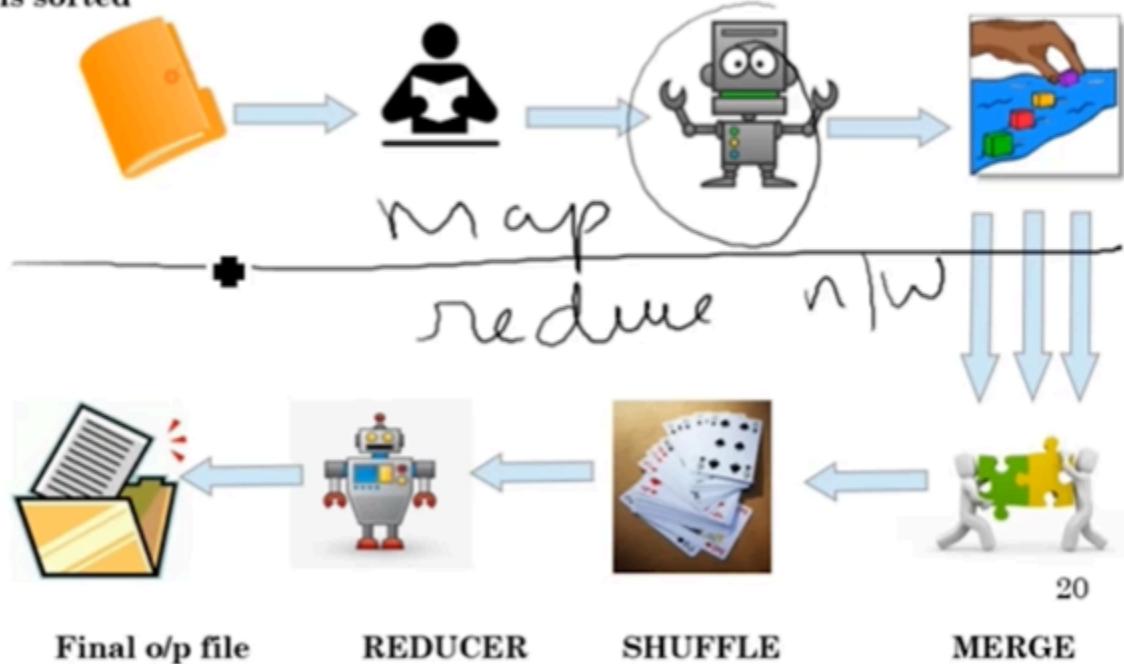


34. Follow ppt for correct order of slides

35. In a nutshell

Everything in a nutshell

Input file processed by RECORD READER o/p goes to MAPPER & its o/p is sorted



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>