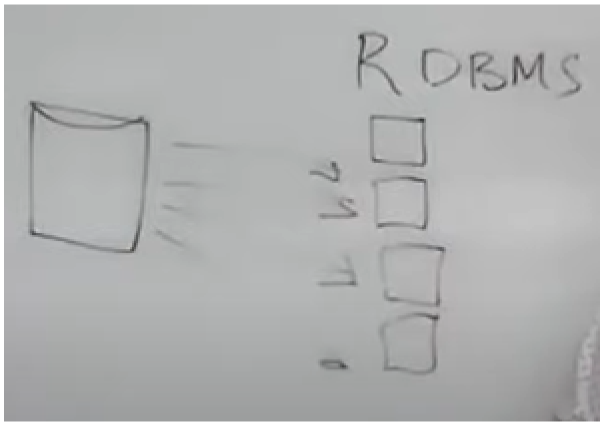
Intro to Big Data

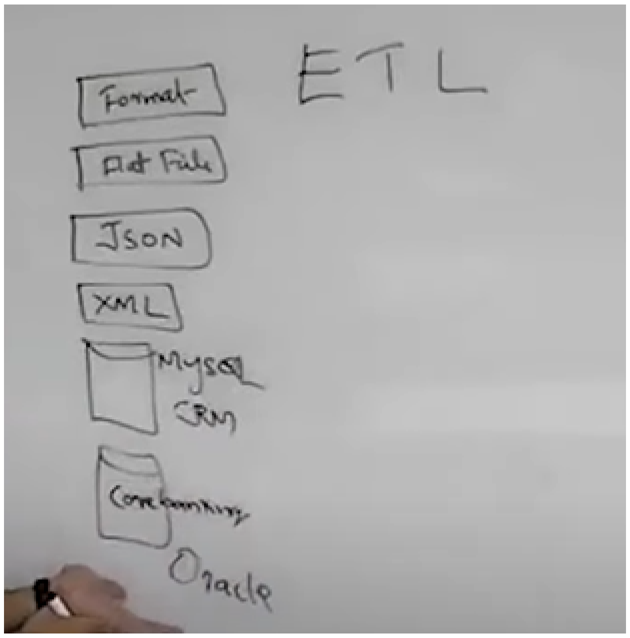
1. Initially in 2005 people use to store data in RDBMS(it has a row-column table format and we insert data)..
2. Fast forward in 2011 ICICI bank wants to migrate to big data they were facing lot of problems bcuz of RDBMS
3. One of the problem with RDBMS is ..If we have less data..then there will be no issue..but as the data gets increasing every day there will be a issue
4. IN oracle database will be separate and processing will be seperate


so we can think like we can add many storage systems,..but at some point what if it reaches to tera bytes of data?..then our RDBMS require more time and more processing to retrieve the data
5. Size of the data increases..RDBMS cannot process it
6. Just imagine how flipkart retrieve data ..it has 10 million products and each product has 5 images…So to retrieve data fastly major companies uses NOSQL
7. Amazon uses dynamoDB…the data is stored in key-value pair…so if a user click one product(key) it gives images(values)
8. Big data : Huge amount of data which we cannot process using traditional methods

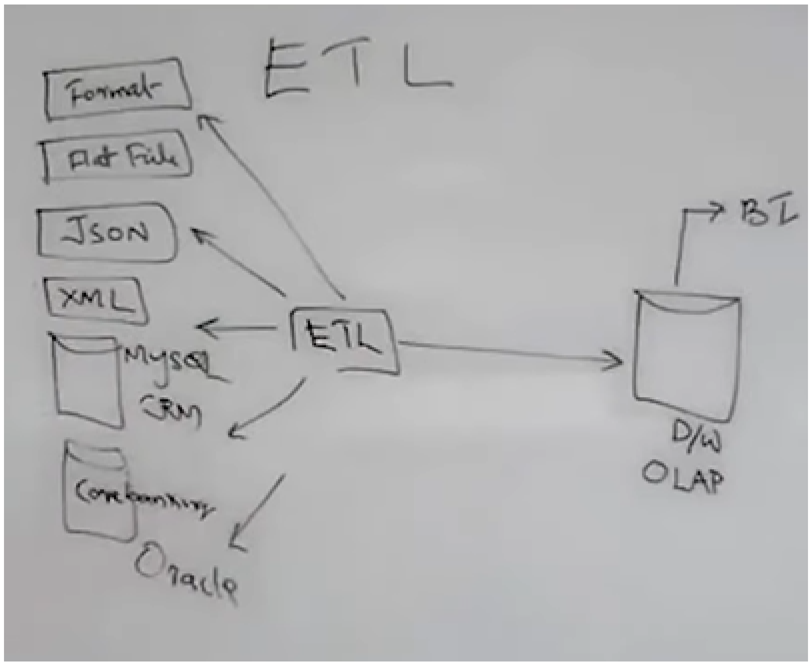
ETL

1. Imagine ICICI bank example

2. ICICI stores core banking data on oracle, and CRM(customer)data on MYSQL, It gets JSON data from social media and customer care logs in text file



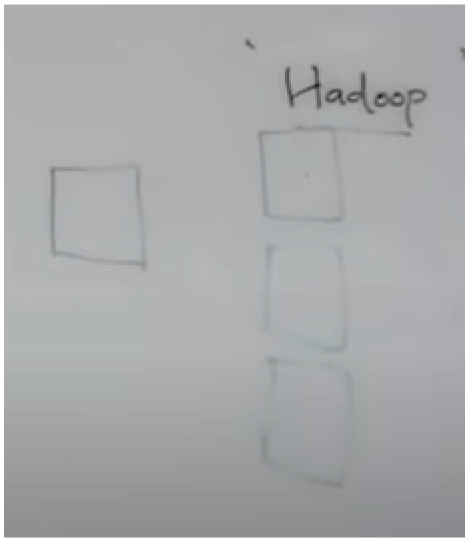
3. Now with the help of ETL tool we consume data from different sources and dump it in data warehouse
4. While consuming we can also transform the data…(BI - business intelligence)

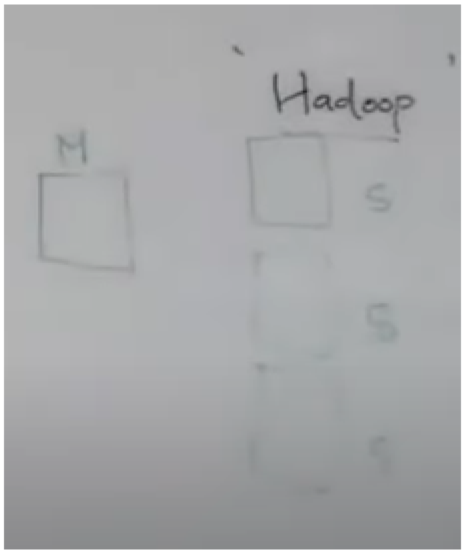


Hadoop Architecture

1. MapReduce is default programming language on top of hadoop..so if anyone want to analyze the data on top of hadoop..they use mapreduce

Distributed computing

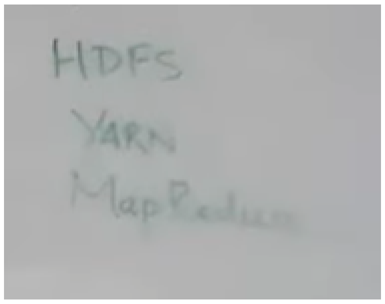
1. Distributed computing was previously used..but in hadoop rather than one machine we can 4(supposed) machines 

2. Now we can install cloudera hadoop in this machines…while installing it asks us "what is the master node and what is the slave node"? 

here M is the master and remaining three are slaves
3. Now what hadoop can do is take the storage space of all slaves nodes and combine it as one
4. So if each slaves have 2 tb capacity…then we have total 6TB capacity
5. So what if our 6tb get full? Then we can add many slaves nodes on the go(allows resizing without downtime)

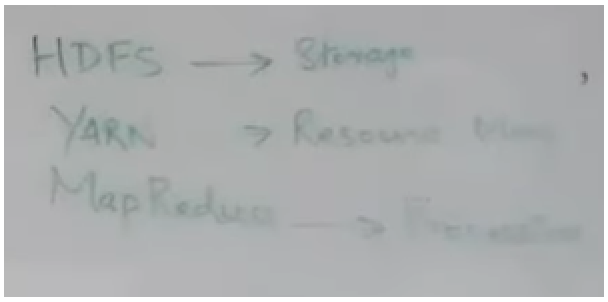
6. But we cannot remove the nodes on the go while the node is still running. If the node is not running we can remove it
7. Raghu used to work at a company ..they used to have 50 nodes..each node has capacity of 256GB RAM and 100TB data storage
8. Now this 50 nodes are called as hadoop cluster and these nodes are called as commodity hardware(cheap assembled servers)
9. We will learn later how hadoop handles failure
10. We can even build hadoop cluster through our desktops

Hadoop Architecture

1. There are these things in hadoop 

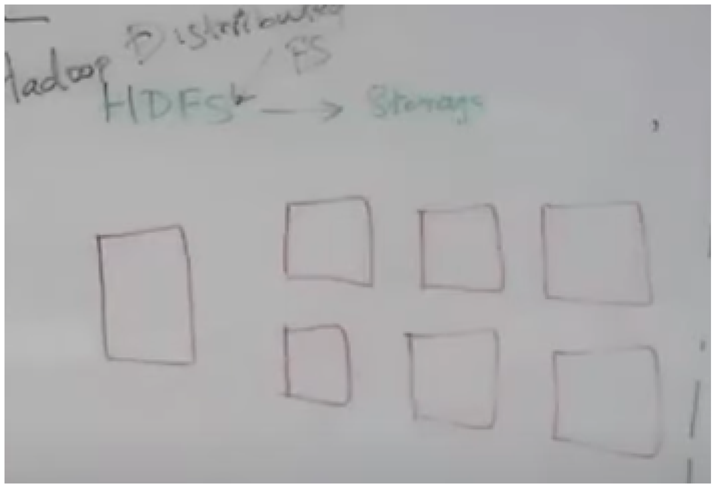
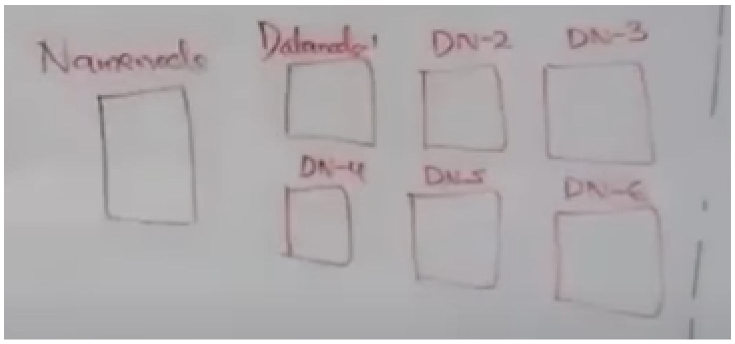
HDFS
YARN
MapReduce

2. When we install hadoop these 3 components will come by default
3. And in those components HDFS-handles storage, YARN-handles resource management, MapReduce- Handles processing



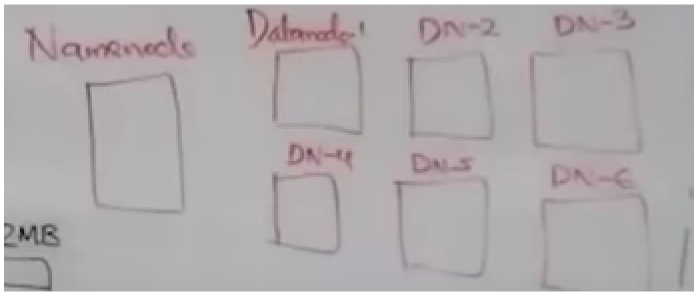
HDFS → Storage
YARN → Resource Mang
MapReduce → Processing

4. HDFS - Hadoop distributed file system

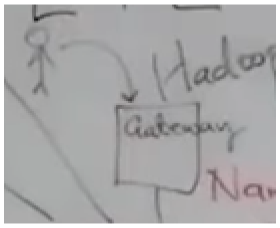
5. Lets suppose we have a situation wer we installed hadoop cluster  Here we have one master node and 6 slave nodes
6. In master machine we have a processor called name node…slave will run a processor called data node 
7. Now lets say you want to store a file in hadoop cluster(hadoop doesnt care about the file formar)
8. We have to make sense of the data while processing(in map reduce)
9. Important-If we store any file in hadoop..then we cannot modify it(only store it and delete it)
10. Why we cant do modifications becuz..hadoop is built for big data(TB,PB) and accessing a file in it is practically not advised and top of that they are commodity hardware which are very slow.
11. Appending is possible but picking up a file and modifying it is not possible.
12. If hadoop was an RDBMS then we can do modification..but hadoop is not a RDBMS

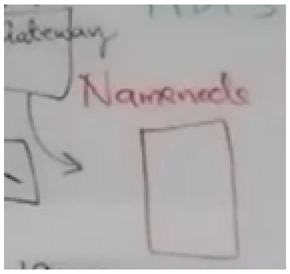
HDFS File Storage

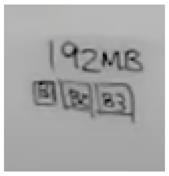
1. Lets say we have a file size of 192MB and we want to store it in Hadoop

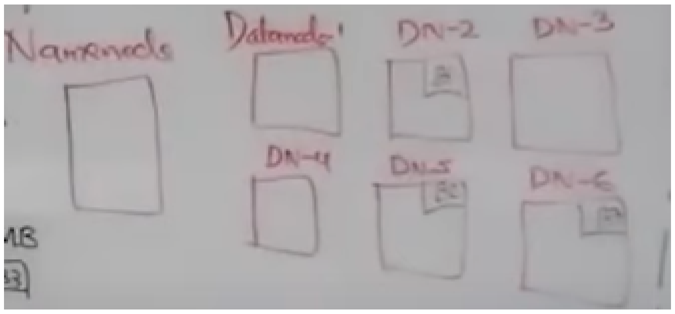


2. In the hadoop there will be a gateway to access the master node..so a user has credentials of the gateway and now this gateway give us the access to hadoop



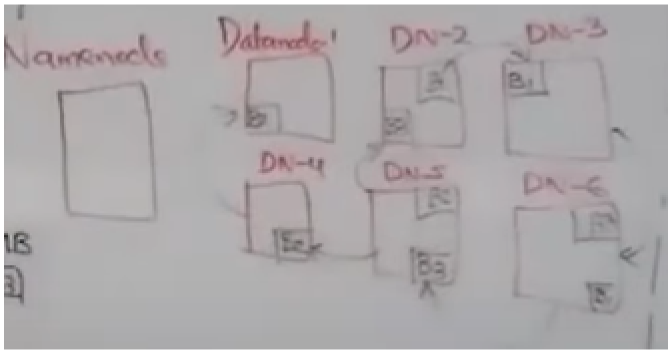


3. Now the master node will tells us the block size(These block size can be configured while installing the hadoop) Now lets assume our block size is 64MB
4. Block size tells us maximum size of data we can store So now our 192MB will split into 3 parts
5. Now namenode informs the gateway that I can only store upto 64MB in each block and tells to inform the client
6. Now the client will divide the 192MB data into blocks of 64MB 

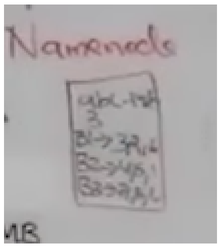
..now we have 3 blocks of data B1,B2,B3
7. Now we go to namenode(master) along with this blocks and now namenode will store the data in different datanodes 

8. So here we are dividing and distribute the data

9. By default our data in hadoop will be replicated 3 times



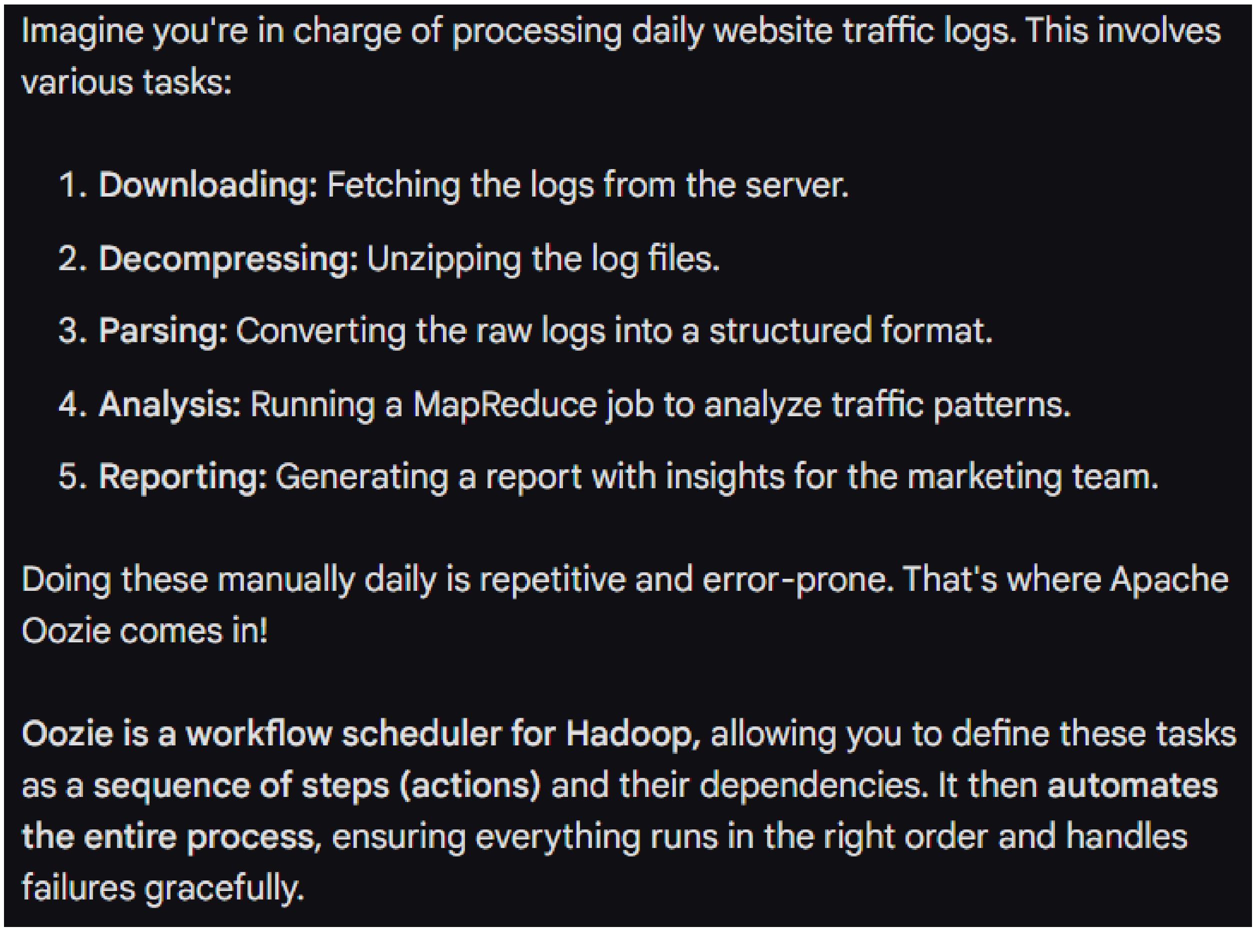
10. Now even if our datanode 1 crashes ..we'll be having the same data in another



11. The namenode has all the metadata of replication
12. Now if the namenode crashes then we cannot access the hadoop ..
13. And even if our namenode crashes we'll be having a standby namenode it also has the metadata of slavenodes

Intro to Oozie

1. We have a tool called apache Oozie which come along with hadoop if we install

2. 

   Imagine you're in charge of processing daily website traffic logs. This involves various tasks:

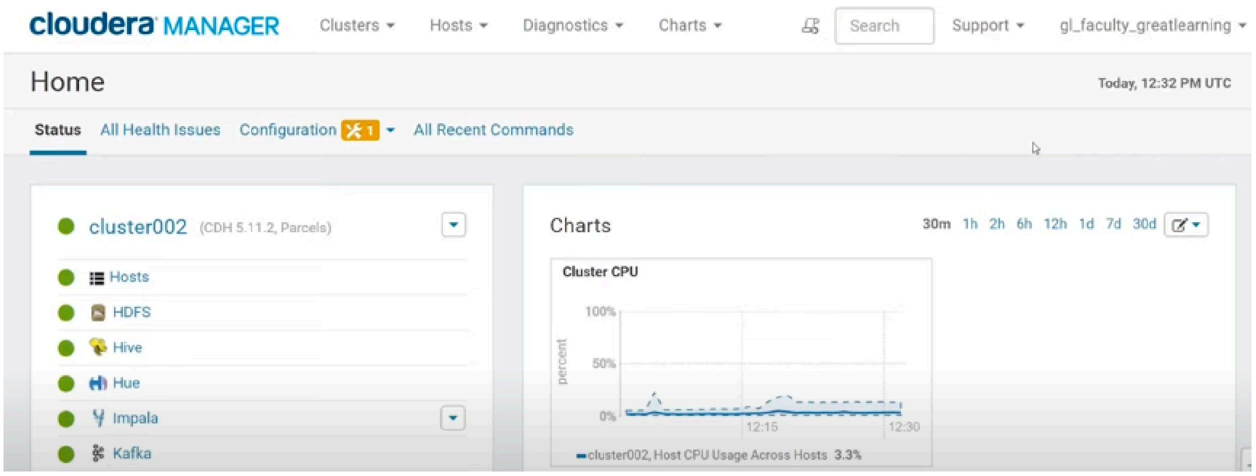
   1. **Downloading:** Fetching the logs from the server.
   2. **Decompressing:** Unzipping the log files.
   3. **Parsing:** Converting the raw logs into a structured format.
   4. **Analysis:** Running a MapReduce job to analyze traffic patterns.
   5. **Reporting:** Generating a report with insights for the marketing team.

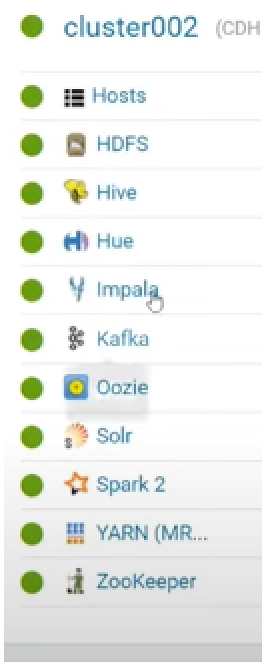
   Doing these manually daily is repetitive and error-prone. That's where Apache Oozie comes in!

   Oozie is a workflow scheduler for Hadoop, allowing you to define these tasks as a **sequence of steps (actions)** and their dependencies. It then **automates the entire process**, ensuring everything runs in the right order and handles failures gracefully.
3. Oozie communicates with master node and schedule the workflow
4. We also have a queue
5. Queue is used to allocate resources ..(for example data engineer need more resources then data analyst) like that
6. Imagine we have 10nodes with 100GB(storage) and 40 processors…now we cant allow single person to use all the processors..so to handle resource allocations we use OOzie
7. So we can write policies with Oozie, and that will be FIFO based(write policies first take resources)

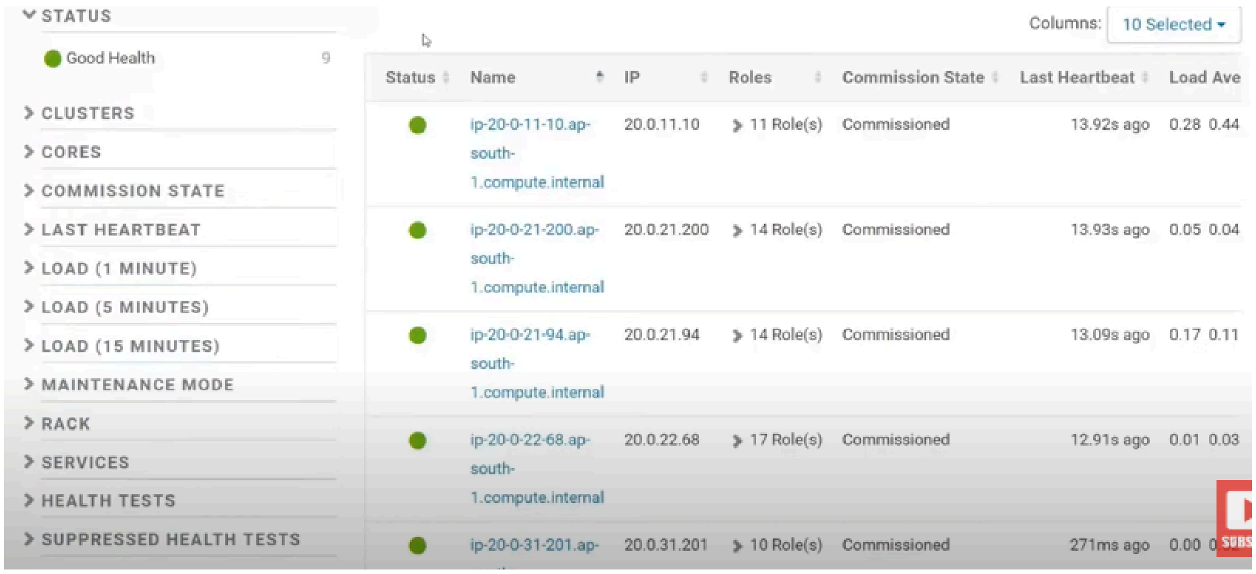
Hadoop Clusters

1. Lets see hadoop cluster in Cloudera



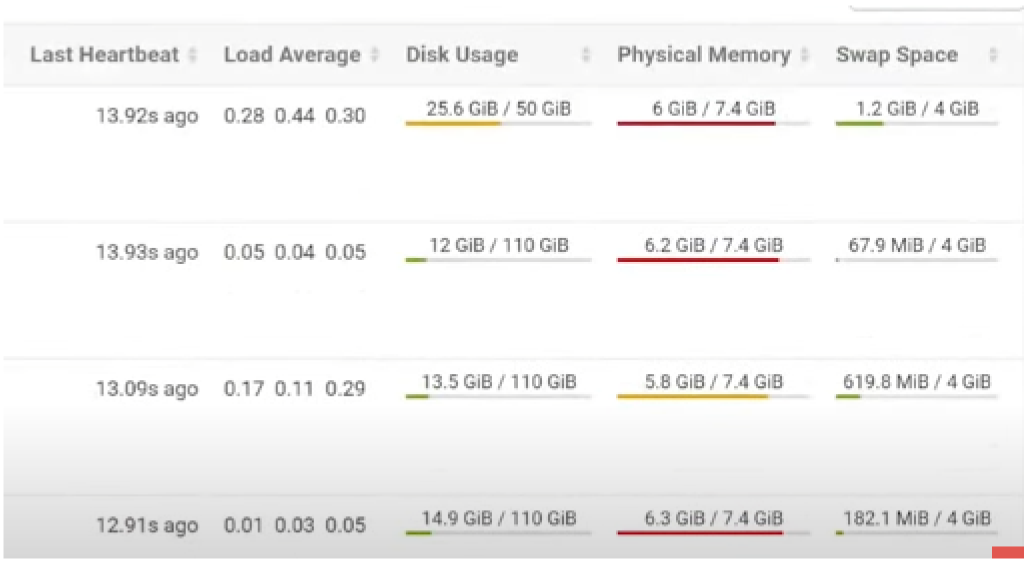


if we go to the hosts menu..we can see our machines(nodes) in the cluster



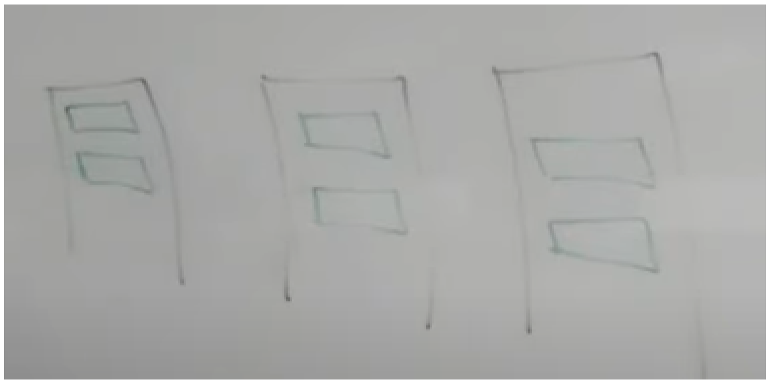
Status	Name	IP	Roles	Commission State	Last Heartbeat	Load Ave
●	ip-20-0-11-10.ap-south-1.compute.internal	20.0.11.10	11 Role(s)	Commissioned	13.92s ago	0.28 0.44
●	ip-20-0-21-200.ap-south-1.compute.internal	20.0.21.200	14 Role(s)	Commissioned	13.93s ago	0.05 0.04
●	ip-20-0-21-94.ap-south-1.compute.internal	20.0.21.94	14 Role(s)	Commissioned	13.09s ago	0.17 0.11
●	ip-20-0-22-68.ap-south-1.compute.internal	20.0.22.68	17 Role(s)	Commissioned	12.91s ago	0.01 0.03
●	ip-20-0-31-201.ap-	20.0.31.201	10 Role(s)	Commissioned	271ms ago	0.00 0

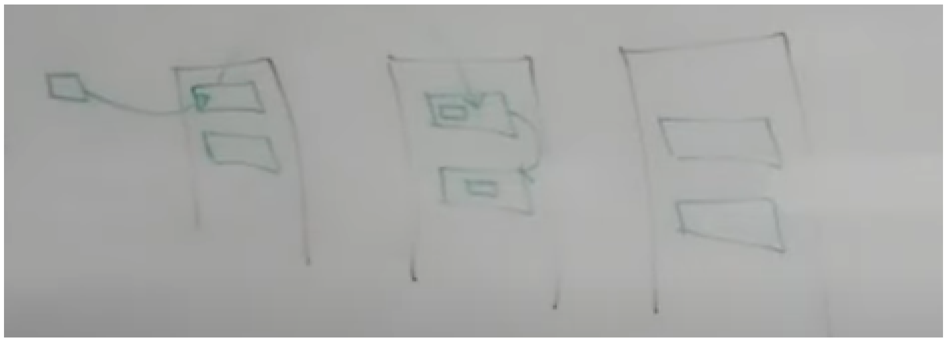
2. We can also see resources allocated for our machines

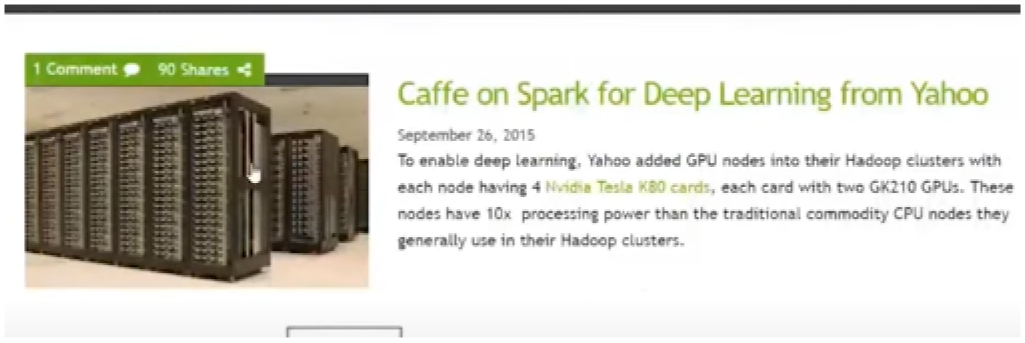


Last Heartbeat	Load Average	Disk Usage	Physical Memory	Swap Space
13.92s ago	0.28 0.44 0.30	25.6 GiB / 50 GiB	6 GiB / 7.4 GiB	1.2 GiB / 4 GiB
13.93s ago	0.05 0.04 0.05	12 GiB / 110 GiB	6.2 GiB / 7.4 GiB	67.9 MiB / 4 GiB
13.09s ago	0.17 0.11 0.29	13.5 GiB / 110 GiB	5.8 GiB / 7.4 GiB	619.8 MiB / 4 GiB
12.91s ago	0.01 0.03 0.05	14.9 GiB / 110 GiB	6.3 GiB / 7.4 GiB	182.1 MiB / 4 GiB

3. Lets suppose we are having 3 racks and each rack has 2 data nodes..

Now we have 6 datanodes in 3 racks

4. Now if we enable a feature called rack awareness then hadoop will replicate our data in different racks 
5. Now the biggest hadoop cluster setup is with yahoo(42000 machines)



Caffe on Spark for Deep Learning from Yahoo

September 26, 2015

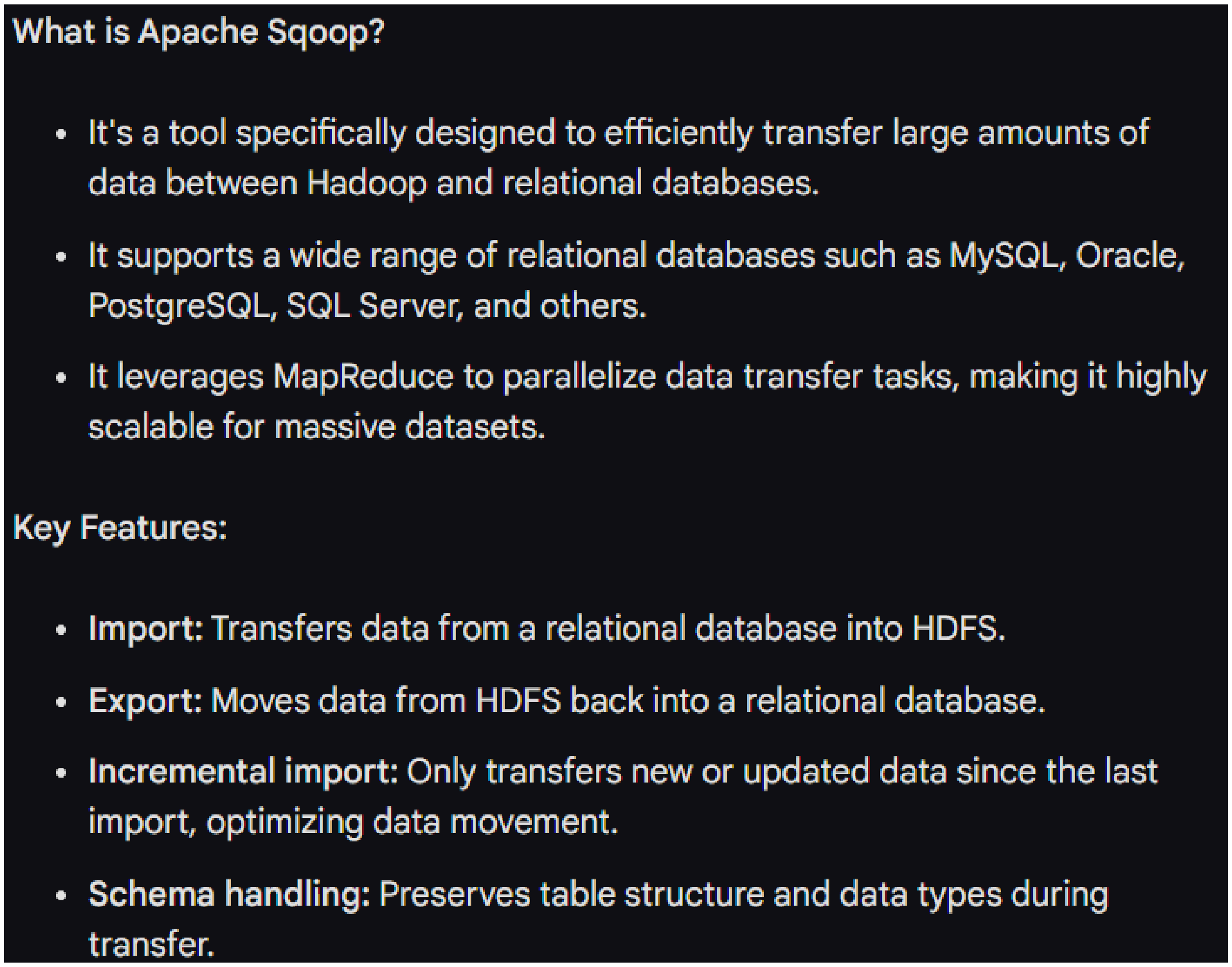
To enable deep learning, Yahoo added GPU nodes into their Hadoop clusters with each node having 4 Nvidia Tesla K80 cards, each card with two GK210 GPUs. These nodes have 10x processing power than the traditional commodity CPU nodes they generally use in their Hadoop clusters.

6. While raghu was working for GE..they used to collect flight data ..like pressure, velocity etc…which was used to analyze the turbulence patterns etc
7. The amount of data generated is very high..and they dont analyze every data…They used to have an algorithm which only takes subset of data..and the rest of data they used to delete or dump
8. In hadoop we also have file format like avro, parquet and we can also have compression techniques..
9. Avro is a serialization format and parquet and all will compress the data.
10. The drawback of compression is ..we need many resources for decompressing the data

Hadoop Ecosystem

1. Lets take ICICI usecase
2. Traditionally we used to have ETL Data warehouse
3. But here we do ELT
4. Now how do we have extract the data and load into hadoop?

5. Here to extract the relational data we have a tool called scoop



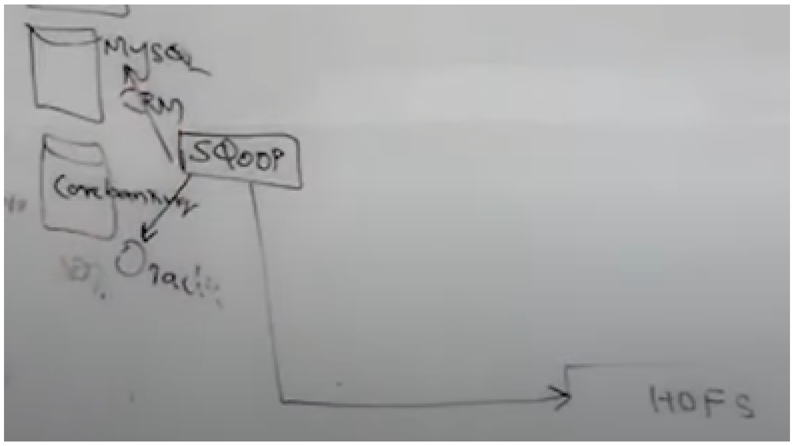
What is Apache Sqoop?

- It's a tool specifically designed to efficiently transfer large amounts of data between Hadoop and relational databases.
- It supports a wide range of relational databases such as MySQL, Oracle, PostgreSQL, SQL Server, and others.
- It leverages MapReduce to parallelize data transfer tasks, making it highly scalable for massive datasets.

Key Features:

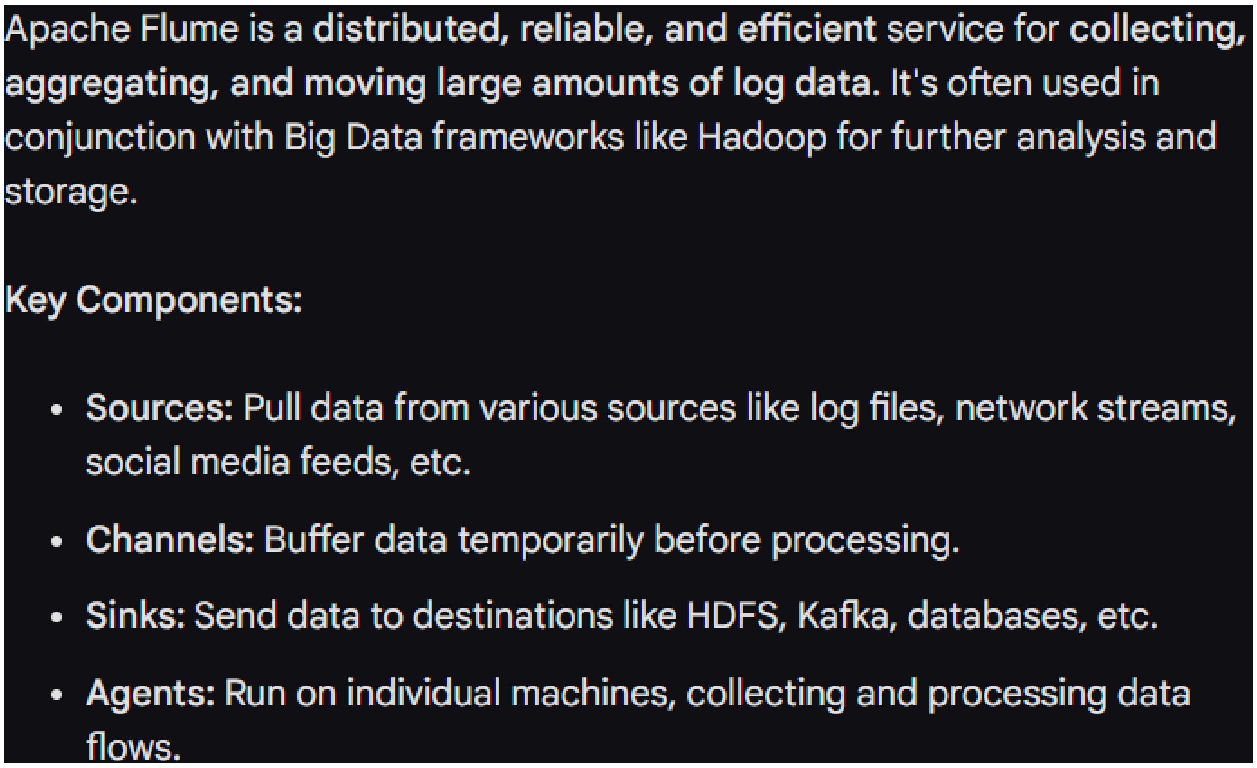
- **Import:** Transfers data from a relational database into HDFS.
- **Export:** Moves data from HDFS back into a relational database.
- **Incremental import:** Only transfers new or updated data since the last import, optimizing data movement.
- **Schema handling:** Preserves table structure and data types during transfer.

6. For extracting relational data to hadoop we use Sqoop
7. While installing cloudera's hadoop we will also get sqoop ..it is part of hadoop ecosystem



it only takes data where we can run sql
8. The problem while working with icici bank..was they were not used to have access to core banking data

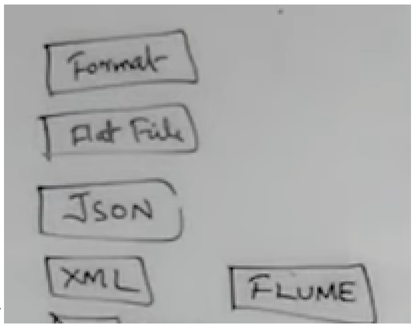
9. So icici used to have a CDC data and a replica of some data which the icici hadoop engineers was allowed to use
10. So to extract data from non flat files we use a tool called FLUME



Apache Flume is a **distributed, reliable, and efficient** service for **collecting, aggregating, and moving large amounts of log data.** It's often used in conjunction with Big Data frameworks like Hadoop for further analysis and storage.

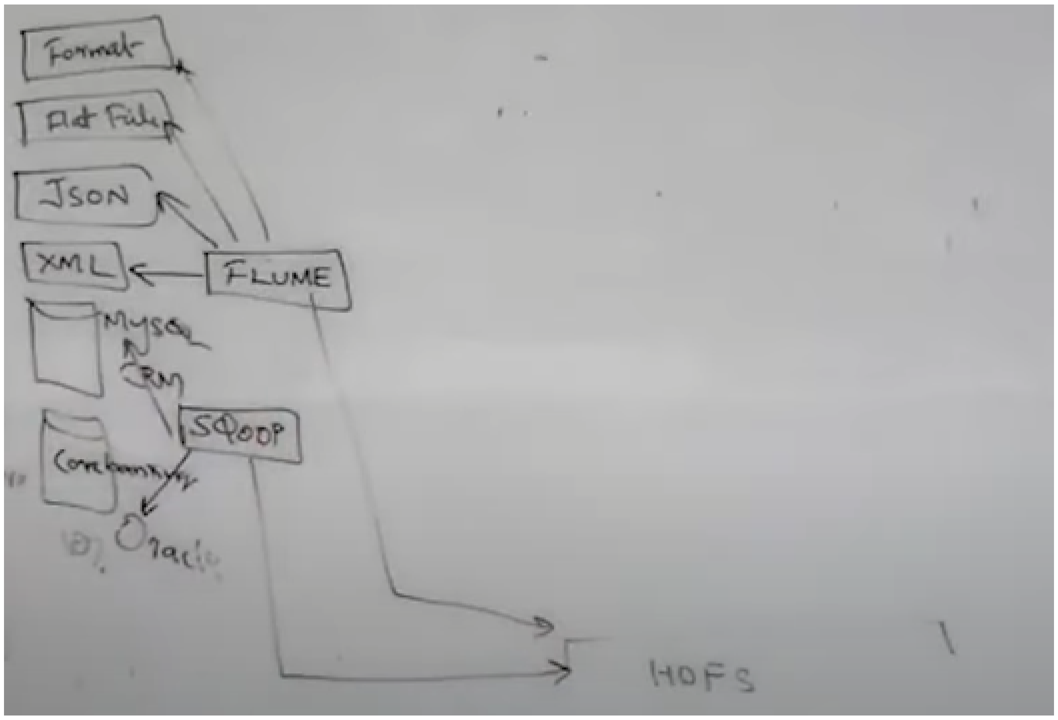
Key Components:

- **Sources:** Pull data from various sources like log files, network streams, social media feeds, etc.
- **Channels:** Buffer data temporarily before processing.
- **Sinks:** Send data to destinations like HDFS, Kafka, databases, etc.
- **Agents:** Run on individual machines, collecting and processing data flows.

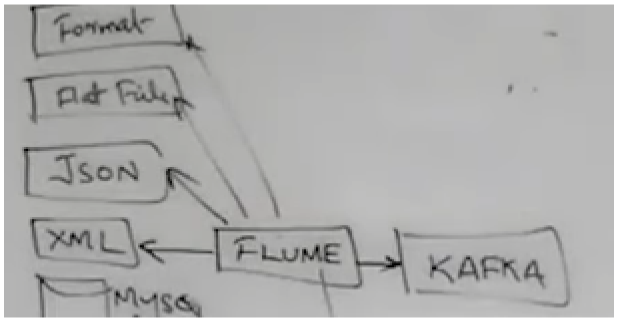
11. FLUME is a point to point delivery tool..the primary objective is to get the data from source system and deliver to destination and it is mainly used to get unstructured data
12. For example in a folder we are getting all the XML,JSON etc..then we can create a flume tool ..which reads the new data from the folder 

13. Flume can do 2 things

14. It can just read the data from the files and just sent to HDFS

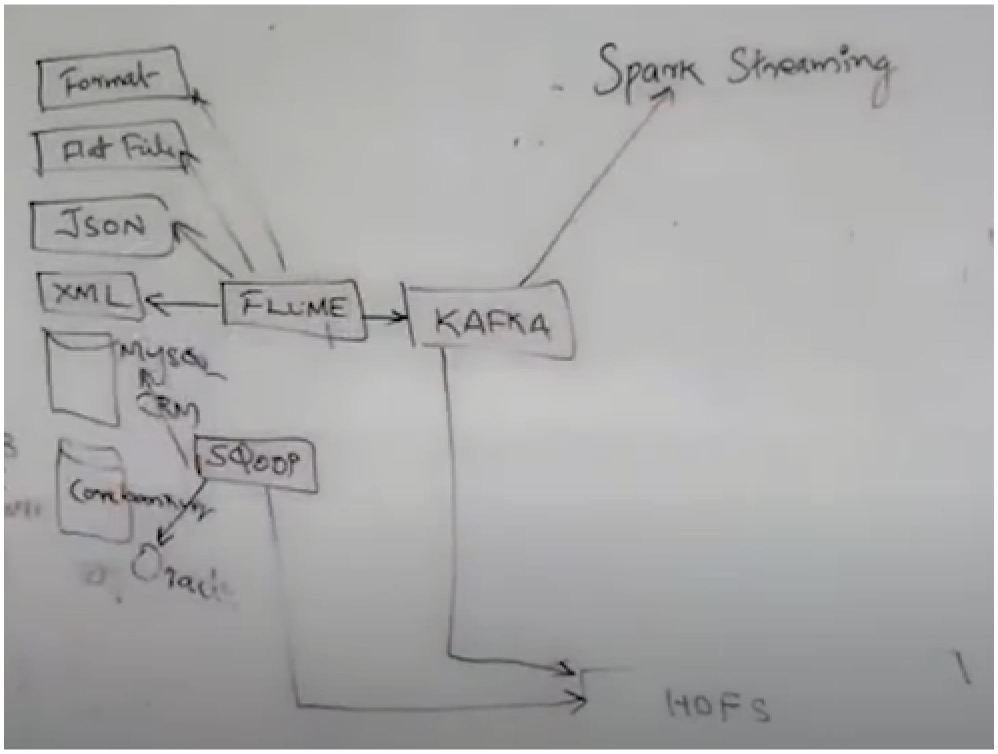


15. But ICICI bank wants to analyze the data and identify the potential customers…now kafka comes into the picture
16. What kafka does is..it runs it own cluster(20-30machines)..it can get data from flume

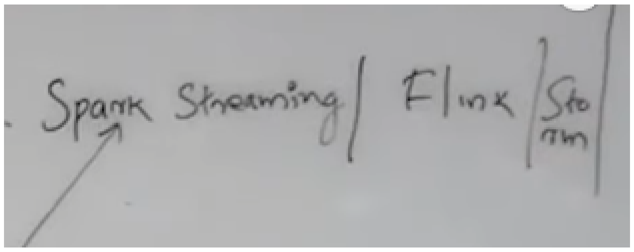


17. By default it stores data till 7 days..and anybody can go to kafka and get the data
18. Flume is a point to point data transfer..wer as in kafka..we can analyse the data..
19. Now we first get the data from FLUME - Kafka(stores data for 7 days) -then we can just push the data to Hdfs
20. Now ICICI bank wants to analyze the data…

21. After getting the data to kafka..how can we analyze the data? By using spark streaming

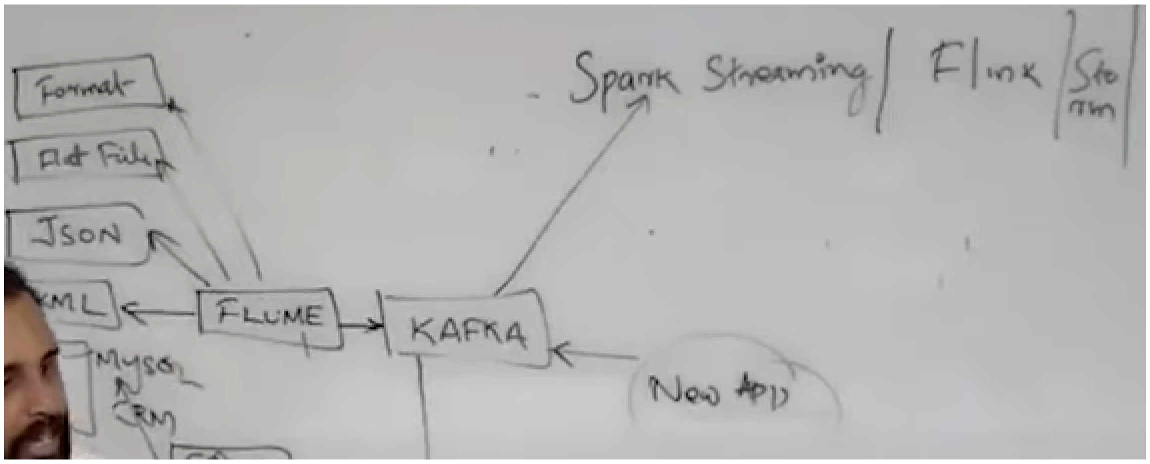


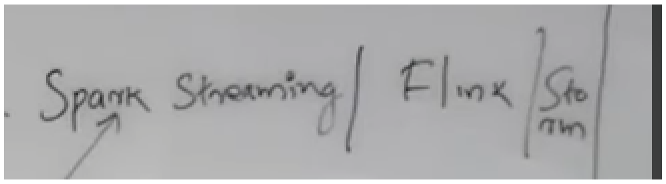
22. Here flume will get the data to kafka first..and kafka sends one copy of data to HDFS and other copy to spark streaming for performing analysis
23. Spark can do real time processing..the moment we gave data to spark..it starts processing(like sending sms etc)
24. Apart from spark we have Flink/Storm etc which are used for real time processing



25. The architecture in ICICI bank was this..like it gets data from FLUME ..then It directs to KAFKA. Now kafka sends one copy to HDFS and other one to Spark streaming
26. In kafka we can configure things..like we need only 40GB of data and store it for like 10days etc

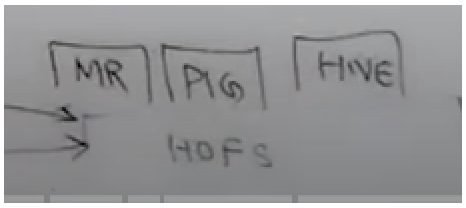
27. Flume can only send data to one entity. So here we directed it to kafka and from kafka ..we can have multiple copies



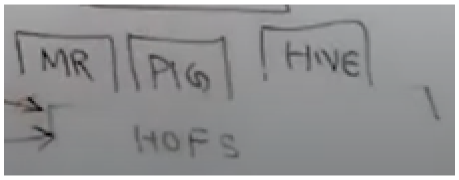
28. Now lets have real life example
29. If i swipe my credit card..how banks will send us sms and otp to confirm whether its really us?
30. Major banks store all the real time data coming from credit card transactions etc and it pushes to FLUME and from FLUME to KAFKA and from KAFKA to Spark streaming for real time analysis
31. Now while analyzing the data..if the card makes fraud transaction ..then the next time while withdrawing banks will sends us the msg to verify ourselves…so this is how real time works
32. Or if a person is withdrawing beyond his limits then..banks will send us real time sms to verify the customer
33. 

34. Here spark streaming will receive a bunch of data for 2 seconds and then will trigger the alert
35. But storm can analyze even single credit card transaction too
36. Also spark streaming to work in real time it also has ML algorithms which runs on previous dump of data to identify fraud transactions
37. Now all this architecture to work ..we need heavy resources to perform this tasks
38. MapReduce
39. Now if our data is in HDFS and we want to do batch processing..the default framework is mapreduce

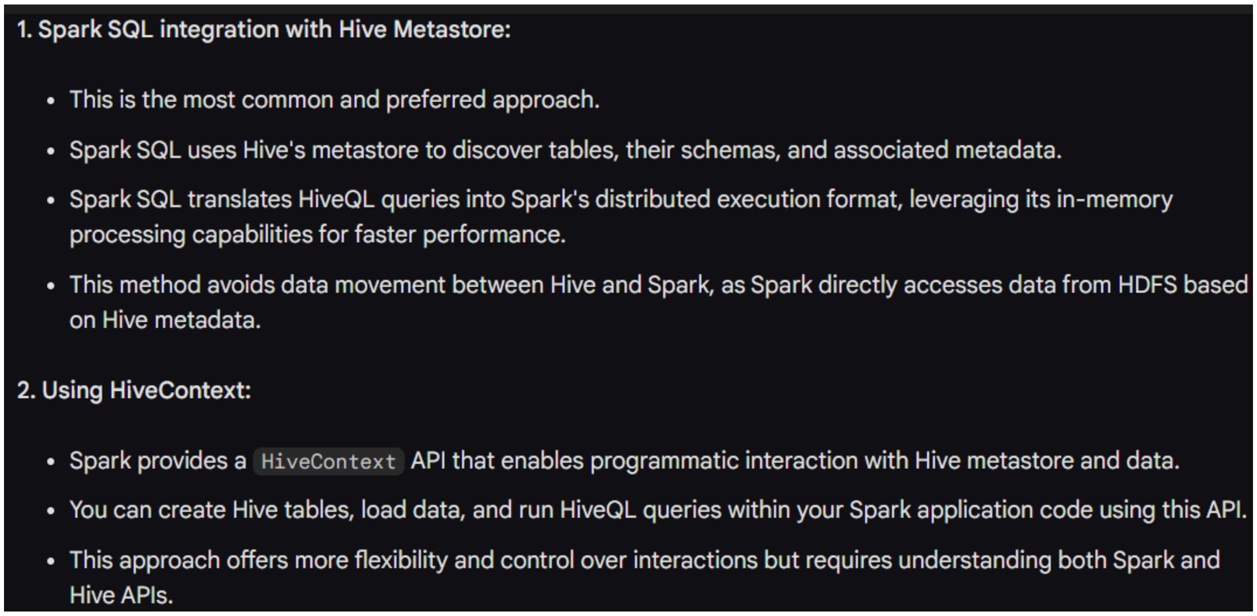
40. We also have pig which is a scripting tool not very popular now..because of hadoop



41. Next we have HIVE
42. Basically what HIVE allows us to do is write SQL queries on top of hadoop
43. When we write a HIVE-sql query and hit run ..it will convert our query into map reduce query
44. So instead of writing java code on map-reduce..we write HIVE sql query
45. Once upon a time at flipkart HIVE took around 12 hrs to complete the run
46. These are all batch processing systems



47. And on top of that we have spark(it also contains spark streaming) which is also a batch processing system but it is very fast.
48. The only diff bw mapreduce and spark is that.. spark is very fast and it is considered as near real time
49. In Spark it also contains spark sql which can communicate with HIVE
50. How spark communicates with HIVE



1. Spark SQL integration with Hive Metastore:

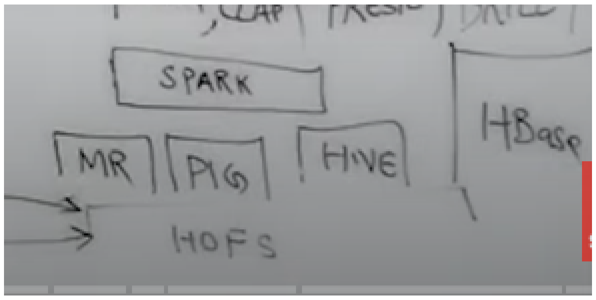
- This is the most common and preferred approach.
- Spark SQL uses Hive's metastore to discover tables, their schemas, and associated metadata.
- Spark SQL translates HiveQL queries into Spark's distributed execution format, leveraging its in-memory processing capabilities for faster performance.
- This method avoids data movement between Hive and Spark, as Spark directly accesses data from HDFS based on Hive metadata.

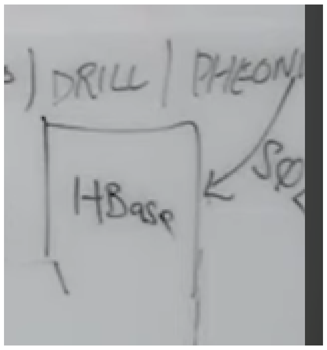
2. Using HiveContext:

- Spark provides a `HiveContext` API that enables programmatic interaction with Hive metastore and data.
- You can create Hive tables, load data, and run HiveQL queries within your Spark application code using this API.
- This approach offers more flexibility and control over interactions but requires understanding both Spark and Hive APIs.

51. Now suppose if you need only one row to process (like for example retrieve * where name = “kaushik”)
52. Now for that y shud i load the full data..but hadoop says load everything and process
53. This is wer PRESTO, IMPALA, HAWQ, LLAP comes into picture which are MPP engines

54. Now what these MPP does is, It remembers the metadata of HDFS and it knows the location. Now to runs small queries like finding name etc..we can fire IMPALA/MPP engines to process this queries
55. It will connect to HDFS and pick up the data as it has metadata.
56. Then we ask why dont we use IMPALA instead of HIVE as it gives result fast…IMPALA is not reliable “if we are running an IMPALA query and if any of the cluster’s machine crashes then we dont get any query results”...but in HIVE even if 10 machines crashes it gives result by taking more time. MapReduce and spark have very less chance of failures.
57. Hbase is the Database of hadoop which is NoSQL DB

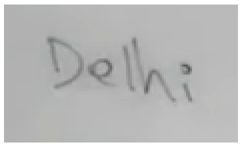


58. Hbase is a column oriented DBMS that runs on top of the HDFS, a main component of Apache hadoop
59. Hbase is an API..if we install hbase on hadoop ..it can do random reads/writes as it is a Non relational DB of hadoop
60. But drawback is Hbase has its own language and phoenix which is a MPP engine can write SLQ queries on Hbase 

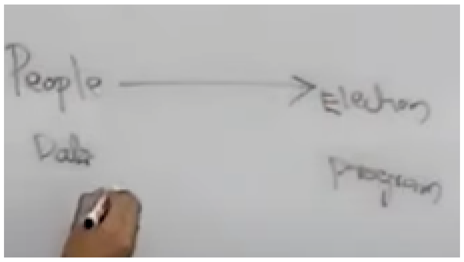
61. We just write SQL queries on phoenix ..it turns to HBASE and gives us the result.
62. This is a complete hadoop ecosystem

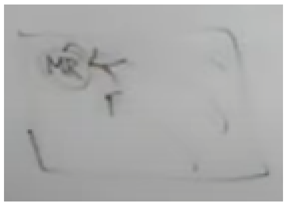
Intro to MapReduce

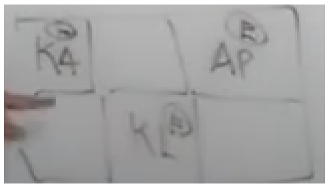
1. Basic idea of MapReduce is divide and conquer
2. Mapreduce was there since 1980’s…even MongoDB,Splunk uses MR
3. Let's take scenario of elections

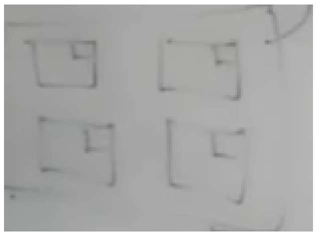
4. 

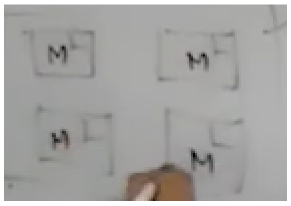
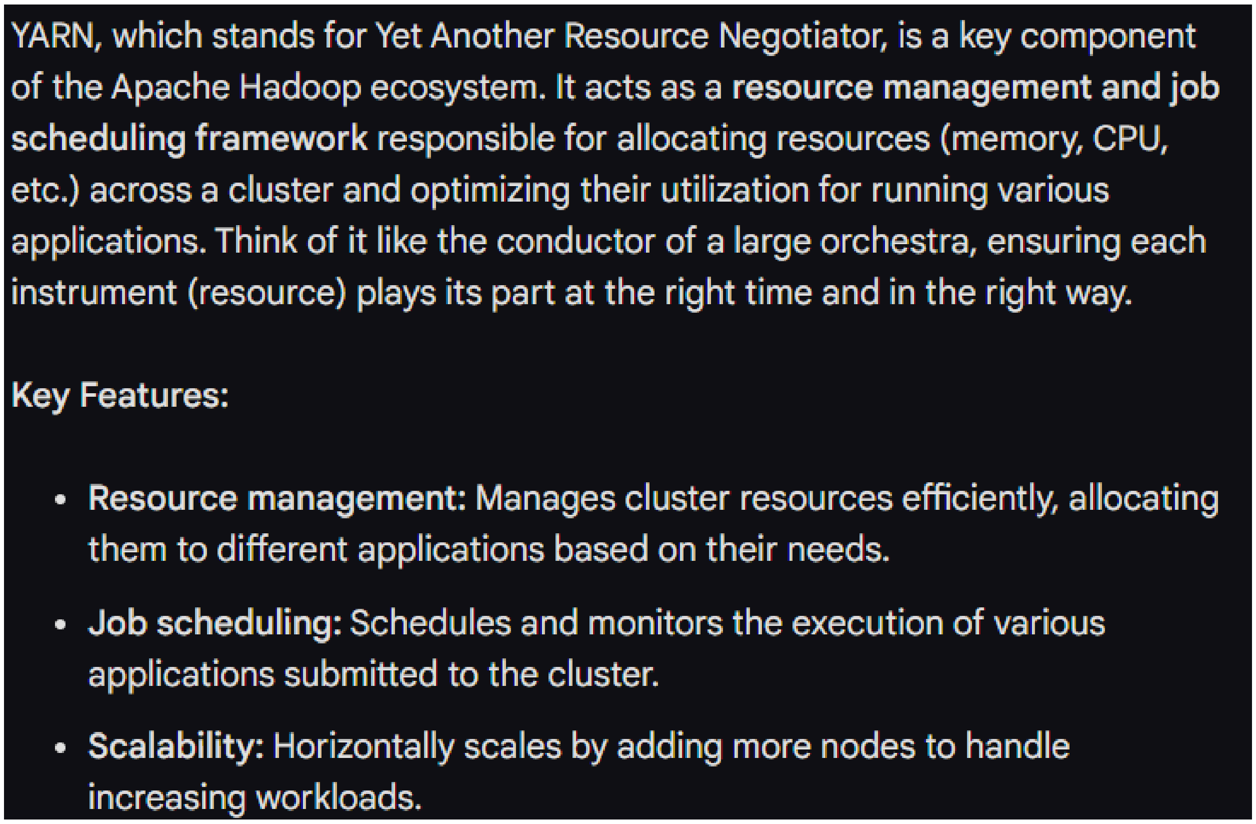
assume that elections happens in only one place in INDIA and people shud travel to delhi to vote
5. As the biggest problem is people must travel long way to delhi and vote.
6. Now the advantage of this is ..we can complete elections in one attempt.
7. This is would be possible if india was a small country of 10k population.
8. Now imagine this election as a program and people coming to vote as data



9. If india was small country then this election would be successful..similarly the data will be sent to one program and the program would be successful
10. But the main problem is the population of india is very high..and so is the data
11. Now lets assume we have a cluster with slave nodes and a MR.
12. If we send data from all nodes to MR then it will be a complete mess..as the data must travel across the blocks and direct to MR. 

13. So here we do the reverse..rather than data goes to program…here the program(MR) goes to the data
14. To say in election scenario… 

election commission will go to every state ..so people can vote easily
15. Now each state has their own election result..and later the results will be consolidated and gives the final result
16. Now lets consider a hadoop cluster with some nodes  and we have a data inside node
17. Now if we want to analyze the data..we will write a map reduce program..

18. In mapreduce we have 2 programs..one is mapper and 2nd is reducer ..
19. Mapper is the phase 1 of elections
20. Mapper will divide and replicate in the data nodes  now mapper runs in the data node paralley..and at last each node has its own set of result
21. Now assume we have a reducer on one data node..and it collects all the results of mapper
22. Here as we have only few data nodes ..we only took one reducer
23. We can manage how many reducers we want based on number of nodes
24. Now all these data nodes with mappers need resources to compute the results and these resources will be allotted by YARN.
25. 

YARN, which stands for Yet Another Resource Negotiator, is a key component of the Apache Hadoop ecosystem. It acts as a **resource management and job scheduling framework** responsible for allocating resources (memory, CPU, etc.) across a cluster and optimizing their utilization for running various applications. Think of it like the conductor of a large orchestra, ensuring each instrument (resource) plays its part at the right time and in the right way.

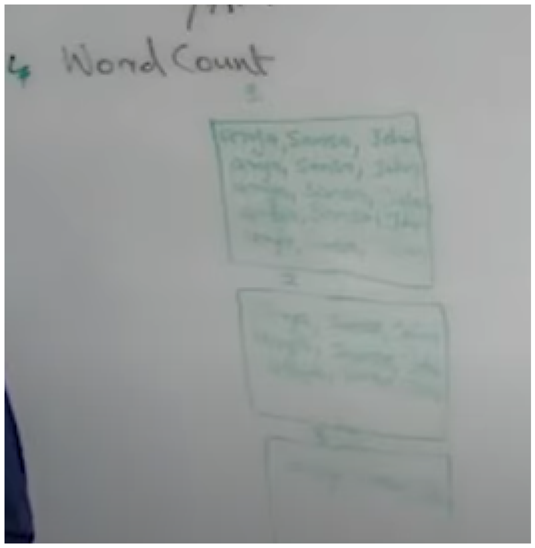
Key Features:

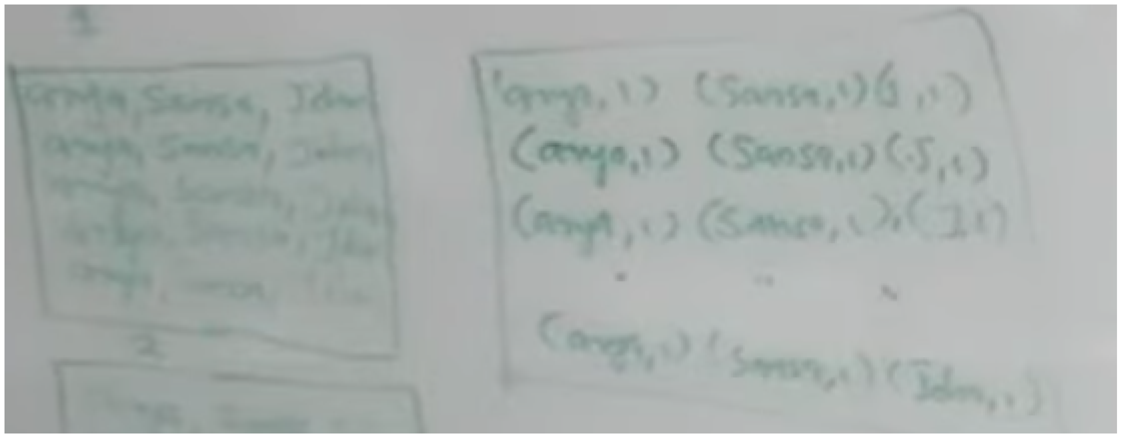
- **Resource management:** Manages cluster resources efficiently, allocating them to different applications based on their needs.
- **Job scheduling:** Schedules and monitors the execution of various applications submitted to the cluster.
- **Scalability:** Horizontally scales by adding more nodes to handle increasing workloads.

26. MR is required when we have TB's of data across all the data node

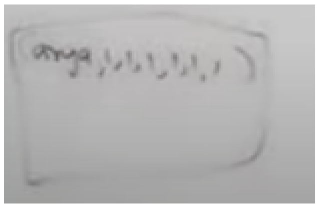
Understanding MR with an example

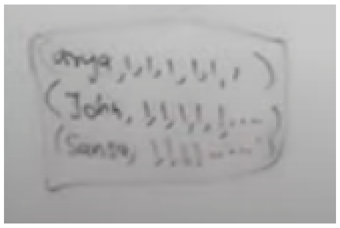
1. We will use word count example to understand MR..
2. If anyone wants to learn MR ..the first program they write is word count.

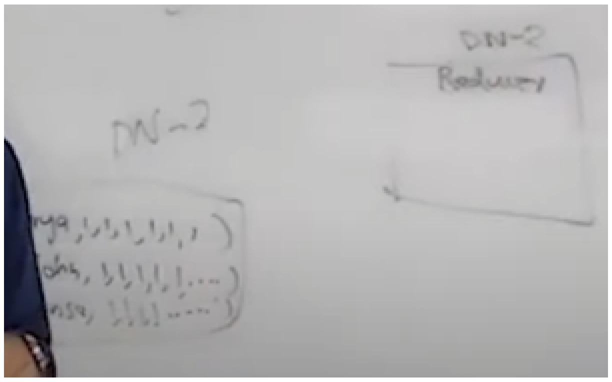
3. Lets assume we have a textfile which spreads across 3 data nodes 

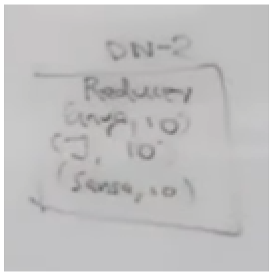
Here in each text file we have names…first node has 5 lines, 2nd DN has 3 lines, 3rd DN has 2 lines
4. By seeing the text we can say there are 10 Aryans,10s sansa, 10 idlies..we can jst see and count the words
5. The output of MR will be in key value pair
6. Now each data node has mapper function..and what mapper function does is it uses string tokenizer and count the each word
7. Here for 1st line in first node the output will be (aryan,1),(sansa, 1),(Idli,1) similar output will be for all the remaining lines 

it takes every word as key and adds 1 as value..which will be in key value pair
8. This is also fault tolerant..If one data node crashes…will be either waiting for it to come back or else we’ll get the result from replicated node
9. After the output of mappers function is ready..then the program automatically runs shuffle and sort

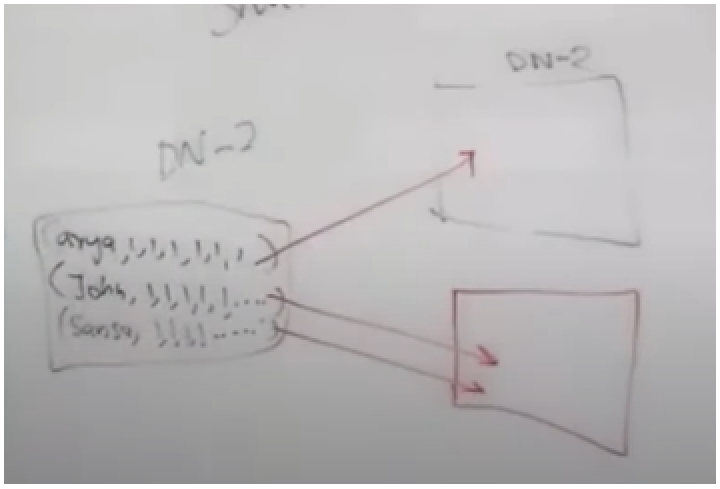
10. What shuffle and sort does is.. for example it gathers the each word occurrence and append its count 

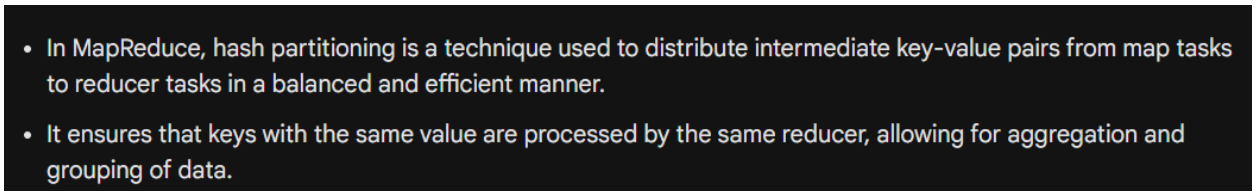
..it brings all the value of key together


11. Now the reducer will kick in..lets suppose reducer is in datanode 2


12. In mapper there will be a reducer program which takes keys from shuffle & sort and sums up its value.. The output will look like 

13. So if our logic defines in mapper is wrong ..then shuffle & sort gets wrong and reducer get wrong as well
14. There can be multiple nodes doing shuffle & sort …if the data is very large
15. Now what if we have 2 reducer's? How the data will go to multiple reducer? ..here MR will use hash partitioning to equally distribute the value of shuffle's to reducer

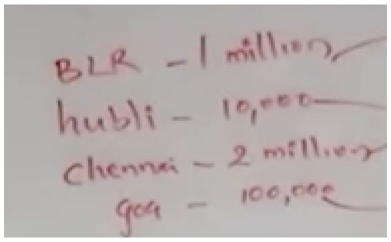


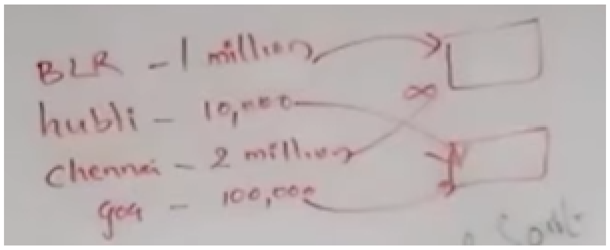


- In MapReduce, hash partitioning is a technique used to distribute intermediate key-value pairs from map tasks to reducer tasks in a balanced and efficient manner.
- It ensures that keys with the same value are processed by the same reducer, allowing for aggregation and grouping of data.

16. Or we can also write our own partition algo based on number of reducer's

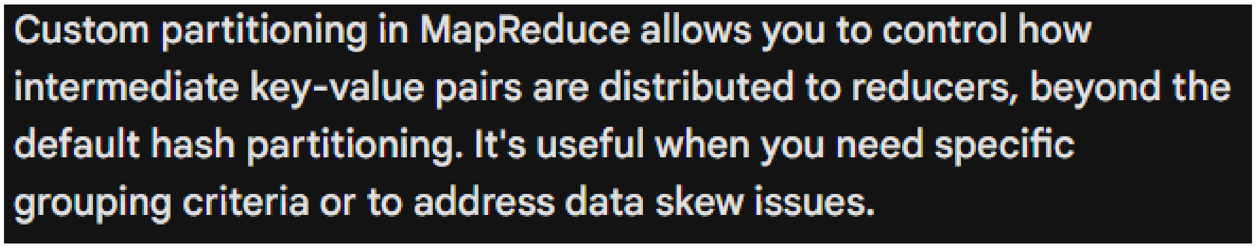
MapReduce practical example

1. Lets come back to ICICI bank example
2. Assume ICICI has 4 branches and transactions as follows 

3. Here branch is the key and each transaction amount will be the value for the key.
4. So here for blr has 1 million key-value pair..similarly for other branches
5. We can see there's skewness in the data
6. Imagine if we have only 2 reducer's ..it automatically evenly distributes like this


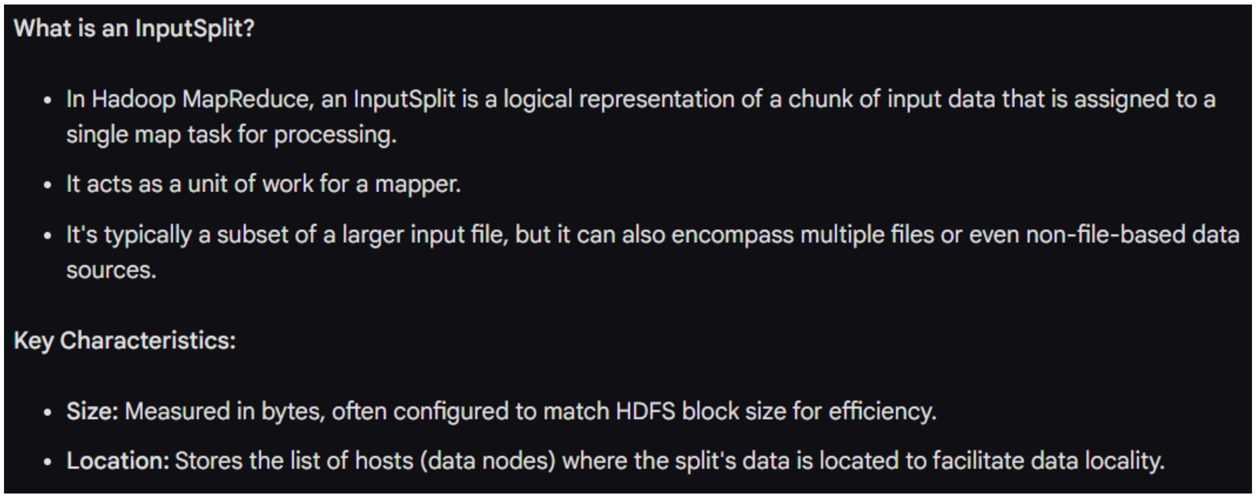
1 reducer will have process around 3 Million value and 2nd reducer process 110000

7. This is where custom partitioning comes into scenario



Custom partitioning in MapReduce allows you to control how intermediate key-value pairs are distributed to reducers, beyond the default hash partitioning. It's useful when you need specific grouping criteria or to address data skew issues.

8. To evenly split the data in the mapper's hadoop uses input split

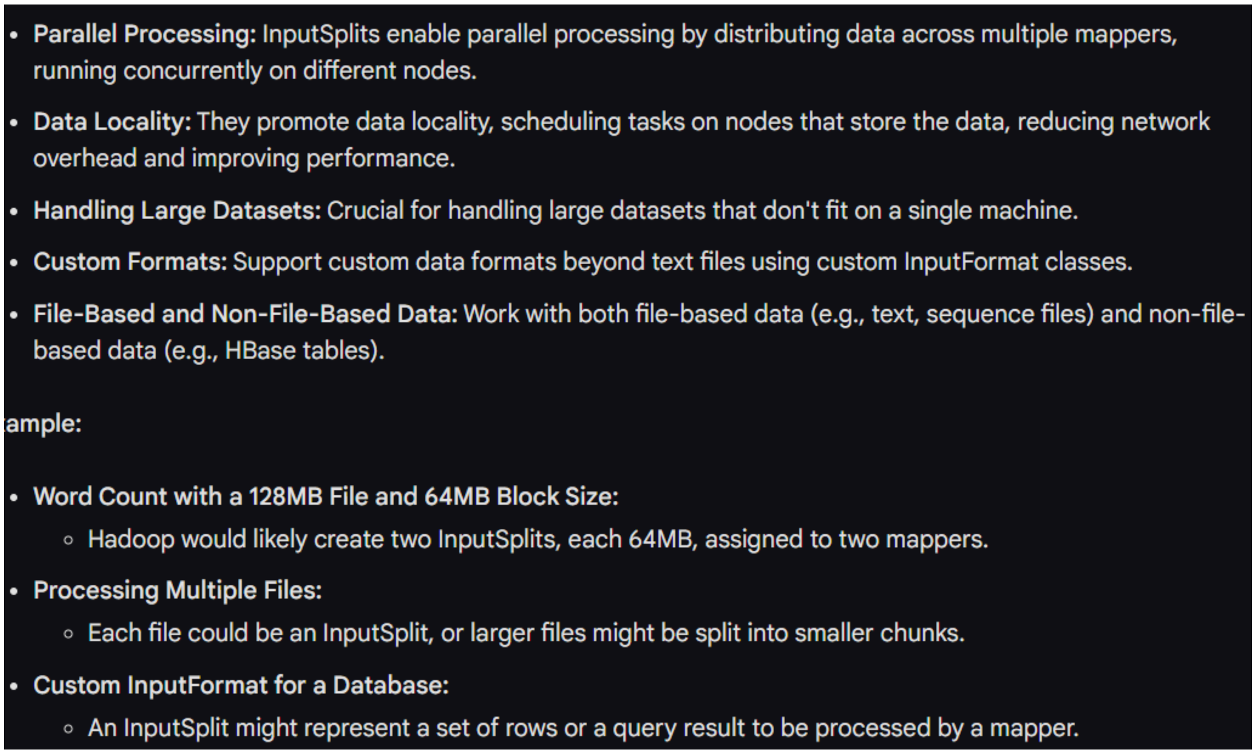


What is an InputSplit?

- In Hadoop MapReduce, an InputSplit is a logical representation of a chunk of input data that is assigned to a single map task for processing.
- It acts as a unit of work for a mapper.
- It's typically a subset of a larger input file, but it can also encompass multiple files or even non-file-based data sources.

Key Characteristics:

- **Size:** Measured in bytes, often configured to match HDFS block size for efficiency.
- **Location:** Stores the list of hosts (data nodes) where the split's data is located to facilitate data locality.



- **Parallel Processing:** InputSplits enable parallel processing by distributing data across multiple mappers, running concurrently on different nodes.
- **Data Locality:** They promote data locality, scheduling tasks on nodes that store the data, reducing network overhead and improving performance.
- **Handling Large Datasets:** Crucial for handling large datasets that don't fit on a single machine.
- **Custom Formats:** Support custom data formats beyond text files using custom InputFormat classes.
- **File-Based and Non-File-Based Data:** Work with both file-based data (e.g., text, sequence files) and non-file-based data (e.g., HBase tables).

ample:

- **Word Count with a 128MB File and 64MB Block Size:**
  - Hadoop would likely create two InputSplits, each 64MB, assigned to two mappers.
- **Processing Multiple Files:**
  - Each file could be an InputSplit, or larger files might be split into smaller chunks.
- **Custom InputFormat for a Database:**
  - An InputSplit might represent a set of rows or a query result to be processed by a mapper.