**RDBMS drawback –** can handle only structure data, limited capacity, costly however transactions RDBMS is only used even now becos of ACID properities

**Data warehouse** –

Also called OLAP(Online analytical platform)

Data from sources is stored in a place called data warehouse.

Understands SQL. Uses ETL(extract, transform and load) e.g., informatica tools to fetch from different sources, applies some schema and load data in warehouses.

Can run BA tools on these

**Drawback – 1.** runs on a single machine and there’s limit on how much amount of data is pulled

2. its not real-time as ETL jobs are run during night times it’s not as quick to provide updates

3. costly maintaining warehouses

**Hadoop is platform**

MapReduce is used to analyze data. Mapreduce are widely written in java but can also be written in python

open source, owned by Apache so if any issues comes no one would fix them

Commercial distribution of Hadoop was 1st provided by Cloudera and support is provided

Hortonworks is other distribution of Hadoop and both provide certifications

Mapr another commercial

Distributed computing is used in Hadoop

Has master-slave architecture

**Components**

1. **HDFS –** storage(Hadoop distributed file system)

Master – Namenode

Slave – Data nodes

* Can store any file
* Cant be edit data, appending is possible
* To connect to Hadoop cluster, need to connect to some gateway which inturn connects to cluster’s namenode, from there any file can be uploaded by dividing it to smaller chunks of cluster block size(data is stored in this size formats) automatically and name node assign different datanode store locations
* By default, in Hadoop each block is replicated 3 times i.e., in 3 different clusters
* Namenode stores metadata i.e., which file is stored in which clusters, if name node crashes we wnt be able to access data so there’s standby name node(for disaster recovery)

Hadoop releases – Hadoop 1,2,3

**Oozie –**

**Rack awareness –** Racks are units where different nodes resides, if we ae enabling rack awareness, Hadoop makes sure to store data in 2 data nodes within same rack and one replica in different rack

the concept to choose a nearby data node (closest to the client which has raised the Read/Write request), thereby reducing the network traffic

**Federation**(multiple active name nodes)There can be multiple name nodes, if data nodes exceeds 5000, then single name node can’t handle it

-Avro file format used in Hadoop to store data

In Hadoop, ETL is not possible becos transforming huge amount data is not possible, we do ELP(Extract, load n transform).

**Sqoop** is used data from db stores to Hadoop

**Flume** –used for unstructured data, tool used to point to point delivery tool, source can be anything to Hadoop. Js dumps the data, doesn’t store it anywhere

* Disadvantage of flume is if user wants information like if someone clicks on their website, they want a msg to be sent which is not possible by flume.

**Kafka** - it runs it own cluster, flume sends data to kafka & it can store up to a week by default. Any1 can access it, instead of directly pushing data from flume to Hadoop,

**Flume ->Kafka ->Hadoop (one copy)**

Kafka can directly get data from different sources but it makes changes while pulling, flume is pull based it wnt make any changes so flume is used to get data. Also Kafka producer has to be installed in the systems

To Analyse\realtime processing data another copy from Kafka is sent to Spark streaming(utility- lib in spark)

Other such tools are Flink,storm

-we can make SQL data from Sqoop also to be pushed to Kafka n then to spark streaming for processing but we usually send it to Hive(data warehouse) where it can be analysed

1. **MapReduce –** Processing

* To run batch in Hadoop MApreduce is used
* PIG – scripting tool in Hadoop not used now using of spark
* Hive – allows u to write SQL on top of Hadoop, hive in turn converts SQL queries to map reduce programs
* SPArk –(spark streaming is part of spark) in memory execution system and is faster.
* Spark also has spark SQL library reads data from Hive and can run.
* Drawback is Hadoop loads everything sequentially & runs on entire data for simple queries where 1 row is involved it isn’t useful
* IMPALA/HAWQ,LLAP/PRESTO/DRILL/PHEONIX – it uses metadata information and directs fetches only that particular data and runs them, much faster (then why not these instead of hive)but they are in memory executions hence not reliable, if system crashes they wnt run unlike hive (where multiple mapreduce run and finish execution even though system crashes)
* All the above including mapreduce are batch processes and run in Hadoop

**HBase –** realtime db of Hadoop, is installed on HDFS but randomly reads is possible in Hadoop(it a noSQL db). Drawback – it has it’s own language. Phoenix is sql on hbase . phoenix is not yet released once its released even transactional things can be handled by Hadoop using Hbase n phoenix

**MapReduce –** basic idea divide n conquer

-It’s a framework

- a mapreduce program has – mapper, reducer programs, mapper runs on different data nodes where our required data is available. once all mapper are run, reducer collects all outputs and produce final output. There can be multiple reducers based on num of mappers usually its 10% of mappers.

-mappers are independent

Output of mapper program is a key, value pair

- ex: in word count, each line is processed and distinct word with its count is made as a key, value pair

-MapReduce is slow, becos it stores each processed data to hard disk

- once all mappers are executed, another data node is picked and shuffle and sort is run where all key value pairs are sorted and if it has same key, values are appended(e.g., in word count – (arya, 1,1,1,1,1)

- reducer programmer- ‘ll count the word occurrences based on word

**Mapper -> Shuffle & sort -> Reducer**

* If there are multiple reducers, hash partitioning is used for distributing data between different reducers. It calculates hash of key and try to distribute evenly. It cant give half of the value to one reducer
* We can customize hash partitioning(there’s class in java custom partitioner) so that there won’t be much load on single system.

Spark also relies on similar idea, but it doesn’t push to hard disk instead stores in memory and gives result in one go. MapReduce in Hadoop is written in java and its very difficult its too complicated. In spark, programs are written in Scala(can be learnt in a day) or python

* Key is usually a string and value is integer writable. If MapReduce is written in java, serialization is done by MapReduce to load data from mapper to shuffler, we use wrapper classes
* HDFS is a file system on linux file system, u should have data on hdfs and program on linux to run it
* HDFS commands are similar to linux(most of them) but are appended with hdfs dfs before them

Like – hdfs dfs ls

Mapreduce has 3 parts – mapper, reducer and driver(controls the program execution) main comes in driver

Hadoop configuration is written in xml files in each node to understand whether it’s a name, data or yarn node.

-Configuration code tries to understand what type of node is it

-Before mapreduce, input split runs automatically , purpose is if a line or a single record is broken in different nodes as processing data is done in RAM it can easily add the rest broken line before processing

-Mapper input also should be in key value pair, framework automatically converts it like based on character position it assigns a key for each line.(this key is however discarded in mapper as its randomnly assigned n has no meaning)

-we can nvr decide on num of mappers running, it’s based on data

Usually mapper reducer are not run directly, they would be assigned to oozie for scheduled run and driver is already available in it

1. **YARN –** Resourcemanager

It’s a facilitator, program to be designed to allocate resources

By default only 1 reducer is run , however the programmer in code can vary it and also the cluster admin also can set it in YARN(then every mapreduce would call this num of reducers)

Resource manager (part of yarn) would hit with the jar file during execution 1st, it would contact a slave machine and launches app master it decides on how to run the program. It tries to contact the namenode to find where data is. Mapper is given to these data nodes after JVM are launched

Resource manager has an ui where u can see hw many applications are running. It can run any framework not only mapper and reducer

Dynamic resource managing is done by YARN

If the node is busy running other programs, then application manager knowing goes to replica of the data in another node

Usually resource manager and namenode would be on same machine but if the cluster is huge it can be on different machines

**Hadoop by Subramanyam**

parallel processing

low cost storage

hadoop architecture - Any hadoop cluster has few components - data node, name node(meta data of the data in data nodes), secondary name node

replication - number of copies of blocks stored in data nodes

hadoop ->HDFS(hadoop distributed files ystem) which internally use linux file system -- has name node & secondary & data nodes

Name node has 2 files --> FS image -

Edit Log - initially meta data would be stored in edit log & on a regular intervals, contents of it would be merged with FS Image

If Name node is down for some reason,

secondary name node -- merges Fs image & edit log contents & empties edit log

Processing - would be taken care by Resource manager is responsible for allocating resources for processing

every data node has a name node & r responsible to perform task

in earlier versions everything has to be written in mapreduce

moved from mapreduce to YARN -supports different models

hive - need to write something like sql like query where pig compiler would converts t o map reduce program

pig - supports pig latin - inefficiencies - didn't survive

pig & hive intermediate languages to convert to mapreduce

spark -

however still few advanced tasks cant be achieved using these n we need to use mapreduce

hbase - is a comular storage used to storage hadoop data in columar basis. can fetch individual columns fastly

Sqoop (sql to hadoop)- rdbms is used for front end application -> transferring structured data into hadoop

flume - used to bring unstructured data into hadoop

zoo keeper - coordination, software used to maintain different componenets of cluster

oozie/falcon - to create workflow & can be submitted to run( to schedule jobs )

Ambari/Cloudera manager - help to monitor & manage cluster

**Hadoop 1.0**

When jobs are submitted, would be split into multiple jobs by job tracker

Task Tracker –

Drawbacks – every program has to be written in the form of map reduce

There’s no backup for name node

**Hadoop 2.0**

**There’s backup namenode or journal which would takeup in case of failure of name node**

**Shared edit log**

**Major change was intro of YARN which removes mandate of writing programs in map reduce form.**

**Configuration details(like replication factor) would be present in HDFS-site.xml**

**Rack awareness –**

**To copy files from one cluster to other, we can use ‘cp’ & ‘mv’ but wnt be effective as within cluster. So we use HDFS Distributed File copy.**

**Syntax** Hadoop distcp <source url> <dest url>

**MapReduce**

**Steps**

1, mapper

2. reducer program

3. driver program

4. create jar file

5.

6. run MR program

**TASK**

**Project**: Word Count in Hadoop using Python

**Objective**: Implement the classic word count example on Hadoop using Python.

**Tools/Languages:**

Hadoop (latest stable version)

Python

Hadoop Streaming utility

**Steps:**

**Hadoop Setup:**

Install and configure Hadoop on your local machine or use a cloud-based Hadoop environment.

Ensure HDFS and MapReduce are functioning correctly.

Prepare Your Data:

Select a text dataset (e.g., a collection of articles, books, etc.).

Upload the dataset to HDFS using hdfs dfs -put.

Writing the Mapper and Reducer in Python:

Mapper (mapper.py): This script will read text input from stdin, break it into words, and output each word with a count of 1.

Reducer (reducer.py): This script will read the output from the mapper, aggregate the word counts, and output the final count of each word.

Running the Hadoop Streaming Job:

Use Hadoop Streaming to run your Python scripts as mapper and reducer.

Command example:

hadoop jar /path/to/hadoop-streaming.jar \

-file mapper.py -mapper mapper.py \

-file reducer.py -reducer reducer.py \

-input /path/to/input -output /path/to/output

Replace /path/to/... with the actual paths to your scripts and data.

Checking the Results:

After the job completes, inspect the output in HDFS.

Use hdfs dfs -cat /path/to/output/part-\* to view the results.

~

Debugging and Analysis:

If the results are not as expected, check the logs for errors.

Modify and optimize your scripts as needed.

Azure 🡪 Resource group 🡪 Hadoop\_Setup(created)

Cluster name – Hadoop

Cluster login Username – admin

Password – Bapuji@1964

Ssh username – sshuser

Password – same

**Hadoop Setup on AWS**

1. Create ec2 instance

Num of instances – 4 🡪 1 for name node, secondary name node & 2 data nodes

Select Ubuntu server

Make sure to select any subnet(else all nodes might not be created under same availability zone)

Make sure to allow all incoming traffic

Once instances are created, rename them (name node, secondary name node & 2 data nodes)

1. Name node should contain the keypair file used to create the rest nodes

Scp -I <keyvaluepairfile>.pem <keyvaluepairfile>[.pem@<publicDNSaddress](mailto:.pem@%3cpublicDNSaddress) of name node>

1. Open 4 command line & connect to each instance
2. Edit /etc/hosts file for all nodes, remove 1st file & replace it with private ip address space followed by public ipv4 DNS

Sudo vi /etc/hosts

Ex: 172.31.9.137 ec2-3-129-210-142.us-east-2.compute.amazonaws.com

1. Update the software

sudo apt-get update && sudo apt-get -y dist-upgrade

1. Install Java in all nodes

sudo apt-get -y install openjdk-8-jdk-headless