← Big Data & How it Started

COBOL → For storing data as files & processing it
(Programming language)

↓

RDBMS Systems → This RDBMS offers us three things:
(MySQL, Oracle, SQL Server)

1. SQL – Data Query language
2. PL/SQL – This is for task which you couldn't perform using SQL
3. Interface for interacting with other programming languages such as JDBC & ODBC

As time evolved, Unstructured data (text, doc, image, video, etc) came into the world, which RDBMS wasn't able to handle.

3-categories of Data; were developed with time:

Tabular data — Structured Data — — — — RDBMS could handle this

↓

NO tabular format — Semi-Structured Data (JSON/XML)

↓

No definite pattern — Un-structured Data

Modern Data Processing need to handle all these problems:

1] Variety :- We need a method to store & process all 3-data categories

2] Volume :- Able to handle large volumes of data

3] Velocity : Modern data is generated at a very high speed. We need to process data at a faster rate.

These 3 problem combined are also known as Big Data Problem.

The Big Data problem was defined using the 3V's of big data – Variety, Volume and Velocity.

We have RDBMS developed overtime, however RDBMS wasn't able to handle Big Data Problems.

So, industry needed a Big Data Management Platform to do this.

- Two Approaches of Big Data Solution:

1] Monolithic Approach :- It designs a one large robust system.
Massive Resources: CPU, RAM, Disk

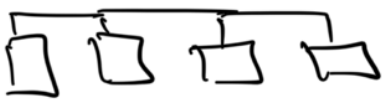
2] Distributed Approach :- In this we take many smaller systems and bring them together to solve larger problem
Cluster. Resource Pool: CPU, RAM, Disk

Evaluation Criteria for Both Approach:

1] Scalability - Both systems are scalable. But you may have to call the vendor to increase capacity of CPU, RAM, Disk.

In Monolithic, it is called vertical Scalability, as you increase size of one single system. (More time because to need to coordinate with vendor).

- In Distributed, it is like adding one more CPU, so it is horizontal scalability


- Existing


- New

(Less time, because you can easily buy new system.)

2] Fault Tolerance and HA :

- Monolithic Approach cannot handle Hardware Failure. (Primary/Secondary)
- In Distributed Approach, if a computer fails in a cluster, other computers can handle it. So, distributed system can handle failures. (Multifold)

3] Cost - Effectiveness :-

- Monolithic : It is Expensive
- Distibuted : Economical (As you can start with less computers & increase as per need)

So, from above analysis we can conclude that Distributed Approach is better.

So, Hadoop came as a distributed Big Data processing platform

- Hadoop :-

Hadoop platform was designed & developed in layers

Core Platform layer offers 3 capabilities:

- YARN - Cluster Operating System
  (It allows us to use clusters of computer as a single large computer)
- HDFS - Distributed Storage
- Map/Reduce - Distributed Computing
  (Hadoop allows us to write data processing application in Java and run it on the cluster)

** STORY OF HADOOP **

⇒ What is Hadoop?

Hadoop is a distributed data processing platform that offers the following core capabilities:
- YARN - Cluster Resource Manager
- HDFS - Distributed Storage
- Map/Reduce - Distributed Computing

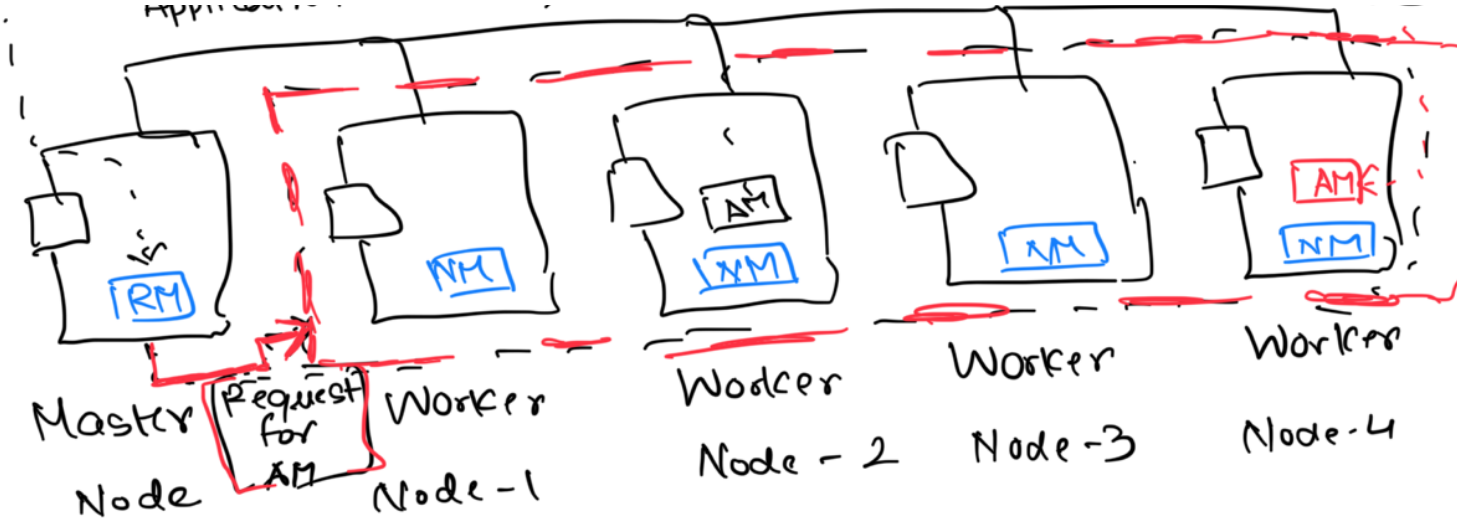
1) YARN :-

Let's assume we have 5 computers:

Client (Submitted Application to Master)

AM for app-2

(AM of app-1) AM container runs your application code



- You submitted Java application to Master node
- Now RM will run this app on cluster. So for this RM will request one of NM to start a resource container & run an AM in the container

What is container?

- It is a set of resources that includes memory & CPU

Example: A Yarn container may be a set of 2 CPU cores & 4GB of Memory.

Now, let's assume we submit another application: The RM will repeat the process. It will ask another NM to allocate a new AM container.

The point is each app on YARN runs inside a different AM container. 1 app running → 1 AM container

in parallel

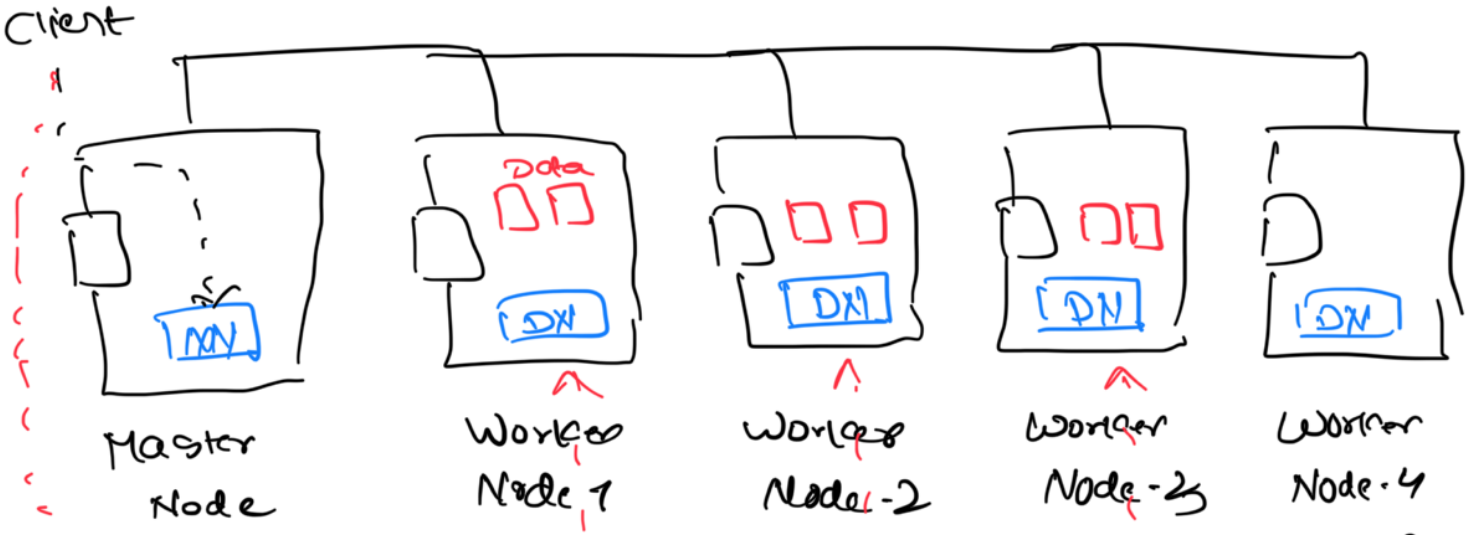
2] HDFS:

It allows you to save and retrieve data files in the Hadoop cluster

2 components:

1] Name Node (NN)

2] Data Node (DN)



- Hadoop install NN on master node & DN on workers node
- The name node with the data node service forms the HDFS.
- The primary purpose of HDFS is to allow us to save files on this Hadoop cluster and read them whenever required

But how this works?

... want to copy a large data

Let's assume you want to copy [illegible]
file on your Hadoop cluster.

- so you will initiate the file copy command.
- The file copy request will go to NN
- The NN will redirect the copy command to one or more data nodes (DN)
- Let's assume you get redirection for 3 DN
- The file copy command will split the file into smaller parts and write those parts on the 3 data nodes.

  It means the file copy command will break your file into smaller parts. Each part is known as block.

  The typical block size is 128 MB.

  So, file copy command will write some blocks to DN-1, some blocks will go to DN-2 and remaining blocks to DN-3

So, your Data file break into blocks, and Hadoop will distribute these blocks on data nodes.

The NN node facilitates this process and keeps track of all the file metadata.

File Metadata Stored at NN:

- File name
  - Directory location (In which directory you create file
  - File size (total size of file)
  - File blocks, block ID, block sequence, block location (How the HDFS breaks the files into blocks, & how many blocks are there?)

⇒ Now let's come back to Read operation

- You will initiate read operation when you want to read a file. The request goes to NN node.
- The NN owns all the information for reassembling the file from the blocks stored at DN.
- So NN will redirect the read operation to target DN.
- The read API will receive data block from the DN and reassemble the file using the metadata provided by NN.

3] Map/Reduce – Distributed computing

What is Map Reduce?

⇒ It is a – Programming Model

– Programming Framework

A programming model is a technique of solving problems.

The M/R framework is a set of APIs and services that allow you to apply the map-reduce programming model

Programming Model in detail :-

Suppose you have a CSV file. The size of file is 20TB. I asked you to count the number of lines in the file.

How will you do that?

– The simple solution will be this :-

open the file & count lines

```
open file as f_hd
    for each t_line in f_hd.get_line()
        n_count = n_count + 1
close f_hd
print n_count
```

But we have small problem – The file size is 20 TB

So, can you find a machine to store a 20TB file and run your line count program on this file?

– It is hard to find such machines. Even if you find a high-end server machine, your program will take hours or days to count the lines.

It takes 3-4 hours to read one TB of data from disk.

So, we have two problems here:

- Storage capacity problem
- Processing time problem

Hadoop offered a solution to both problem.

- You can use Hadoop cluster to store the file.

Let's assume you have 21-node Hadoop cluster.

One node - Master & other nodes workers.

Name Node - Master & Data node on workers.

YARN runs a RM on master & NM on workers

So we have those services running on the cluster.

Now, let's talk about resources

Each node comes with four hard disks of 2TB

So, single node storage capacity is 8TB. Overall Hadoop cluster storage capacity is 160 TB.

Each node also comes with 16 CPU & 64 GB of memory. So, total cluster capacity is 160 CPU Cores & 128 GB of memory.

- You can copy to your 20TB file on cluster using HDFS. HDFS will breaks the file into small blocks & spread them across cluster

So, storage problem is taken care by the Hadoop cluster.

What about processing time problem?

⇒ That's where map-reduce model comes into play.

Distributed solution Pseudocode:

```
def map(file_block):
    open file_block as fb_hd
    for each t_line in fb_hd.getline():
        n_count = n_count + 1
    close fb_hd
    return n_count
```

```
def reduce(list_counts):
    For each cnt in list_counts
        total_count = total_count + cnt
    print total_count
```

The difference between new logic & old logic is that, the new logic opens the file block and counts the line, whereas the old logic opens the file & counts the lines.

Now, let's see how it runs:

- I can run the map function on all the data nodes in parallel. This map function will open

each block on DN & count the lines.

Basically, I am counting lines on 14 DN in parallel. Everything runs at the same time. And, I will get the line counts in 1/14th of the time compared to doing it in a single machine.

- Now, I will use reduce_function to get the total count. I will start reduce function at one node.

So map-reduce is a programming model to break your application logic in two parts.

⇒ Now, you may think how many problem we can solve using Map Reduce? I mean, we can do line count. But can we do other complex data processing work?

→ Yes we can do it.. Hadoop offers a Hive database as an additional component. Hive allows you to write SQL expressions. However, Hive SQL engine translates all your SQL expressions into Map Reduce programs.

Therefore we use Map-Reduce to develop tools & solve problems.

We rarely do not use the raw Map-Reduce on Hadoop.

However we do not use map reduce,
We use high level solutions such as Hive SQL, Spark SQL, etc. and develop data processing applications.

⇒ Now, let's talk about Map-Reduce Framework:-

I will submit my code to YARN RM. who will request for AM container to run my code on a worker node. But my program, has 2 functions i.e. map & reduce function.

So, running my program will trigger map-reduce framework, and M/R Framework will take control of my problem & start running in the AM container.

- Now M/R framework will request RM to start 14 more containers. The RM will pass request to NM, & they will start 14 new container.

  So now, one new container is running on 14 workers. And these new containers will run my map function.

So, the Hadoop M/R Framework ran our map function on all data nodes.

Now M/R framework runs Reduce

Same way, M/R framework runs ... function & give us the output.

⟹ Hadoop as a platform and Hive as a Hadoop database became very popular.

But we still have other problems which needed improvement:

1] Performance :- Hive SQL queries performed slower than RDBMS SQL queries.

2] Ease of development :- Only a few skilled engineers were writing M/R code.

3] Language support : Hadoop M/R was only available in Java.

The popularity of cloud platforms realized the other two problems.

4] Storage : For storage in Hadoop you buy new computers. Initially it was cheap option. But when cloud started offering storage at less price, buying computers

seems expensive

5] Resource Management :- YARN

So, the point is Hadoop left a lot of scope for improvements.

And that's where Apache Spark came into existence as an improvement over Hadoop M/R

Apache Spark advantages over Hadoop :

1) Performance :- 10 to 100 times faster than Hadoop MR

2) Ease of development :- - Spark SQL
- High performance SQL engine

3) Language Support :- Java, Scala, Python & R

4) Storage :- HDFS storage
Cloud Storage

5) Resource Management :- YARN, Mesos, kubernetes.

improvement over

Spark started as an improvement over Hadoop MR.

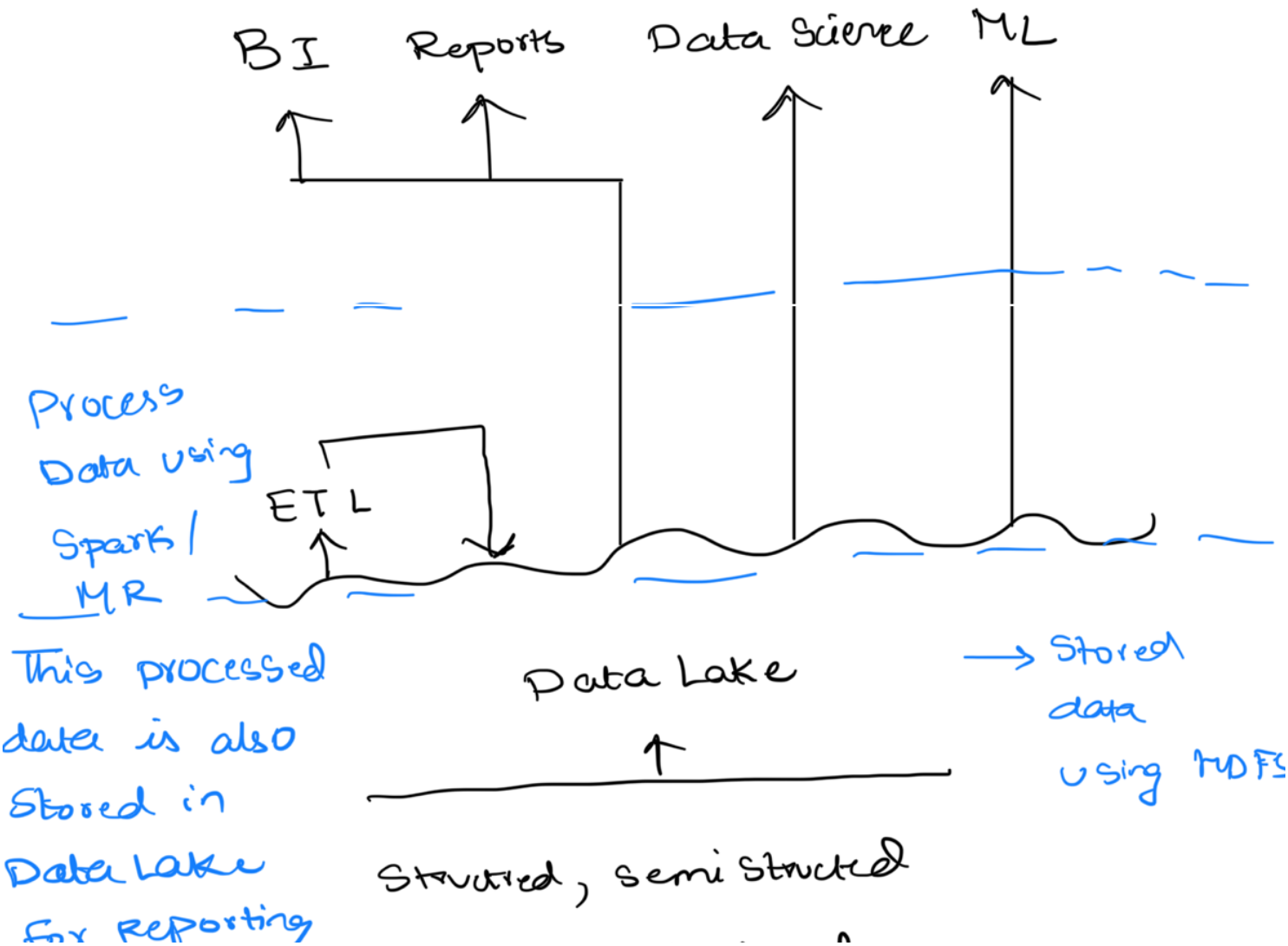
But overtime it become an independent system.

Now can use Spark in two kinds of setup:

1. With Hadoop (Data Lake)
2. Without Hadoop (Lakehouse)

What is Data Lake & How it works ?

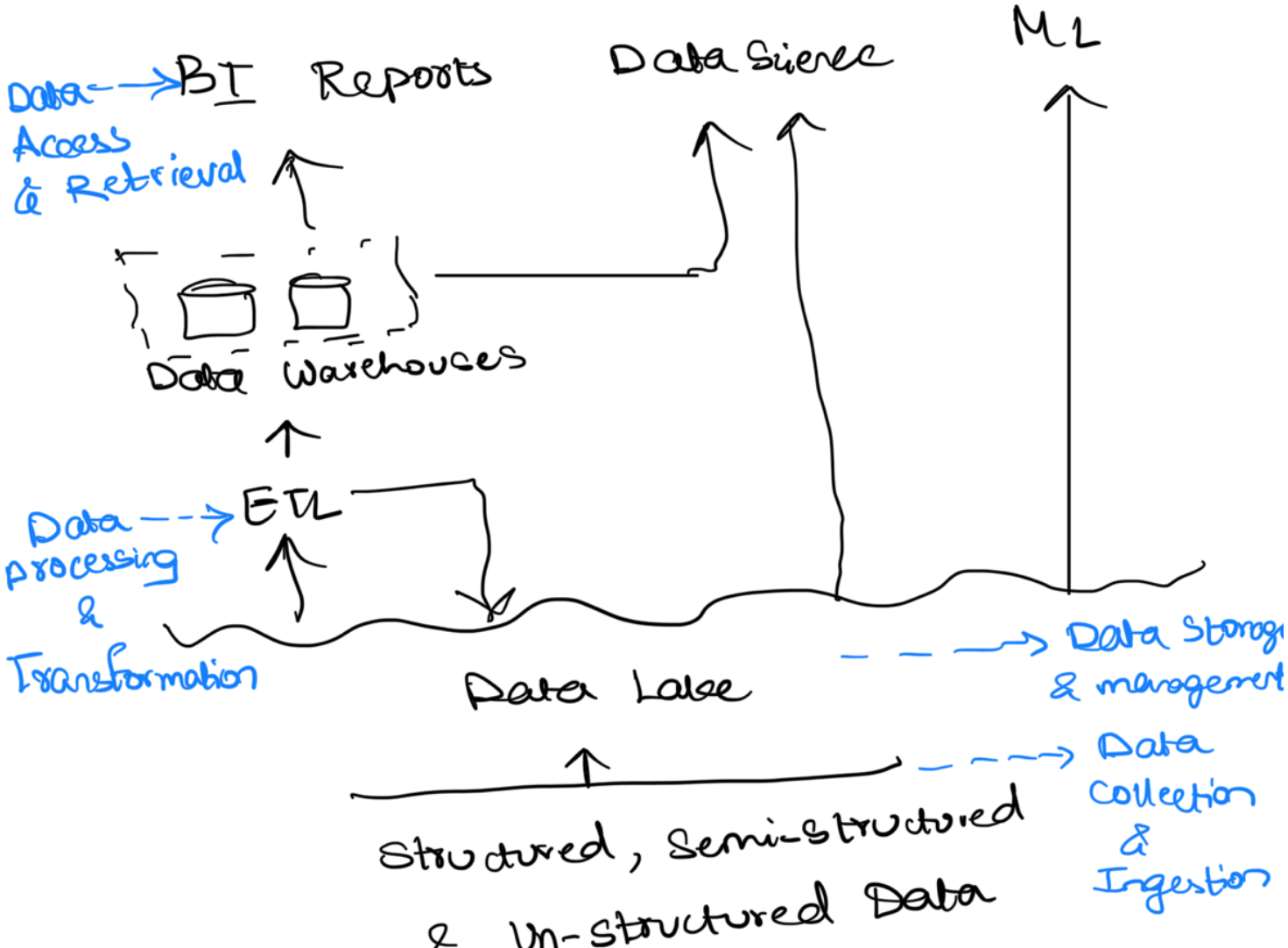
Data Lake :-



& Unstructured

However, Data Lake technology missed two critical features: 1) Transaction & Consistency
2) Reporting Performance

So, we started adapting different architecture for Data Lake



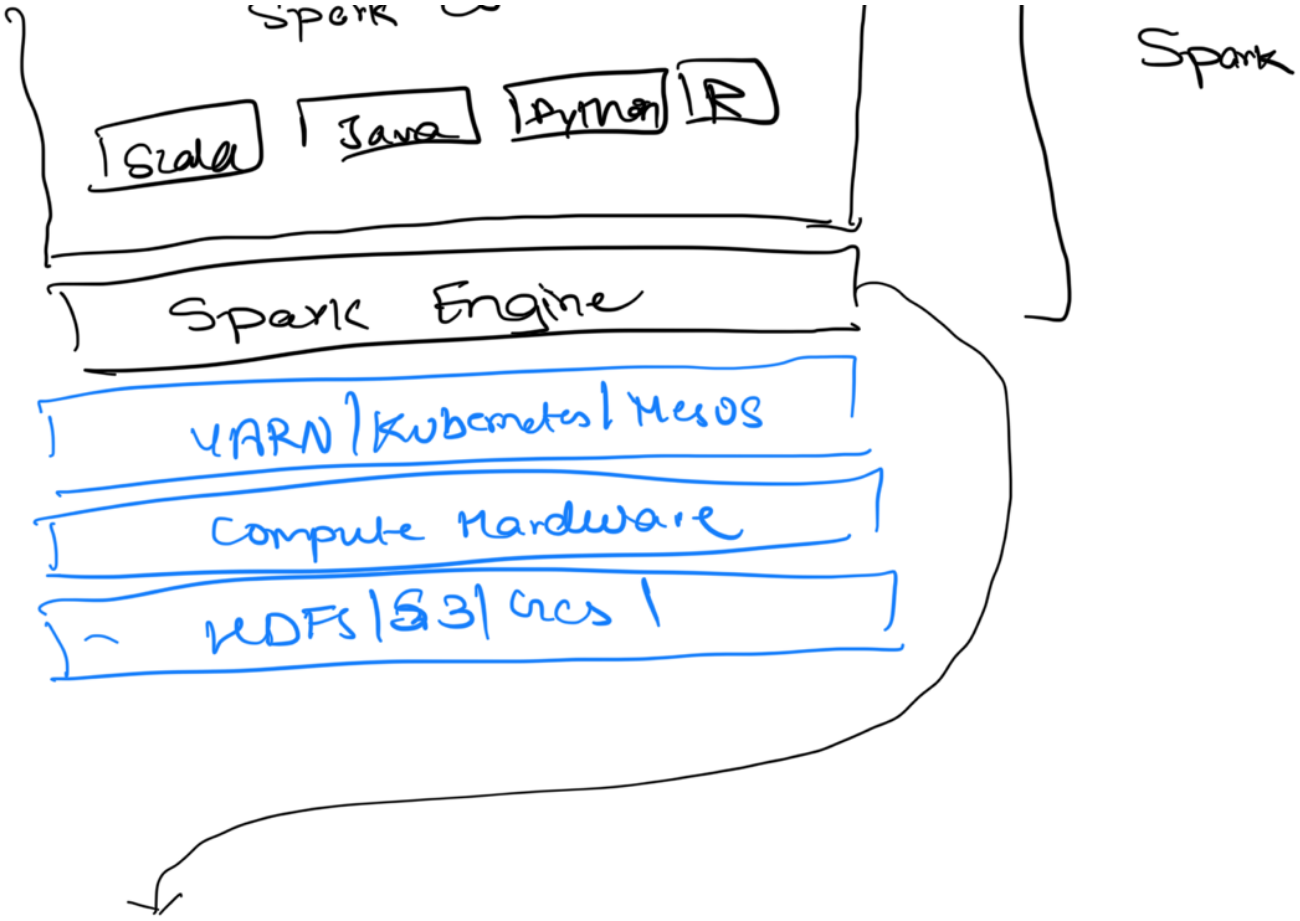
Data Storage & Management :- HDFS, Amazon S3, Google Cloud, Azure Blob

Data Processing & Transformation (ETL) :

1. Initial data quality check
2. Transforming & Preparing data
3. Correlating, Aggregating, Analysing and extracting business insights.
4. Applying ML models.

(Apache Spark falls in this data phase)

⇒ Introducing Apache Spark



This part
handles
data processing
workload
(It breaks your
data processing
task into
smaller tasks,
scheduling those
task on cluster,
managing &
monitoring those
task, etc)