**PySpark**

**Apache Spark + Python = PySpark**



**+**

**=**

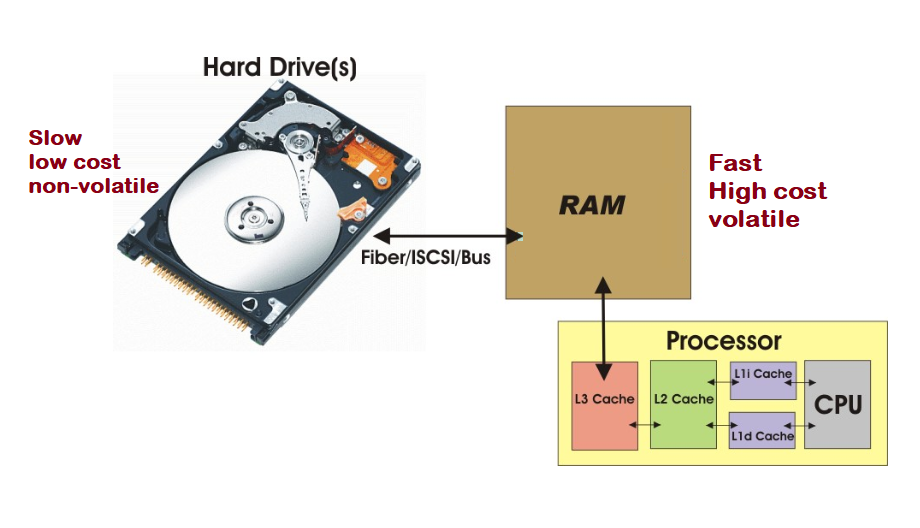
**=**

In **Machine Learning**, to process the data we use **Pandas**.

Pandas in not enough when we have large dataset.

If we are working on large datasets, then we need something else. So, here **PySpark** will rescue us.

**Data Flow from Storage to CPU (Processor)**



This is the very basics on the computer field.

This above diagram structure exists in every system like, Desktop or Laptop or All-in-one’s.

1st we have **Hard Drive** (it could be a physical hard drive, external hard drive, pen drive, CD-drive)

2nd we have **RAM**.

3rd we have **Processor** (which we call it as CPU)

**Hard Drives** are: Slow

Low Cost

Non-Volatile

**RAM** are: Fast

High Cost

Volatile: Volatile means whenever we shut down the computer, the contents will get destroyed.

Each time we execute any operation, that means we are giving instructions to the processor or CPU to process the data and that data is in the hard disk.

So, here data flow from hard disk to RAM, then CPU performs the operation on the data in the RAM (It could be any operation Read or Write).

So, Processor/CPU cannot perform any operation on the data which is in the hard disk. The data should be in the RAM because there are many contents and processor will take much time to read all the content to search all the content by using indexing.

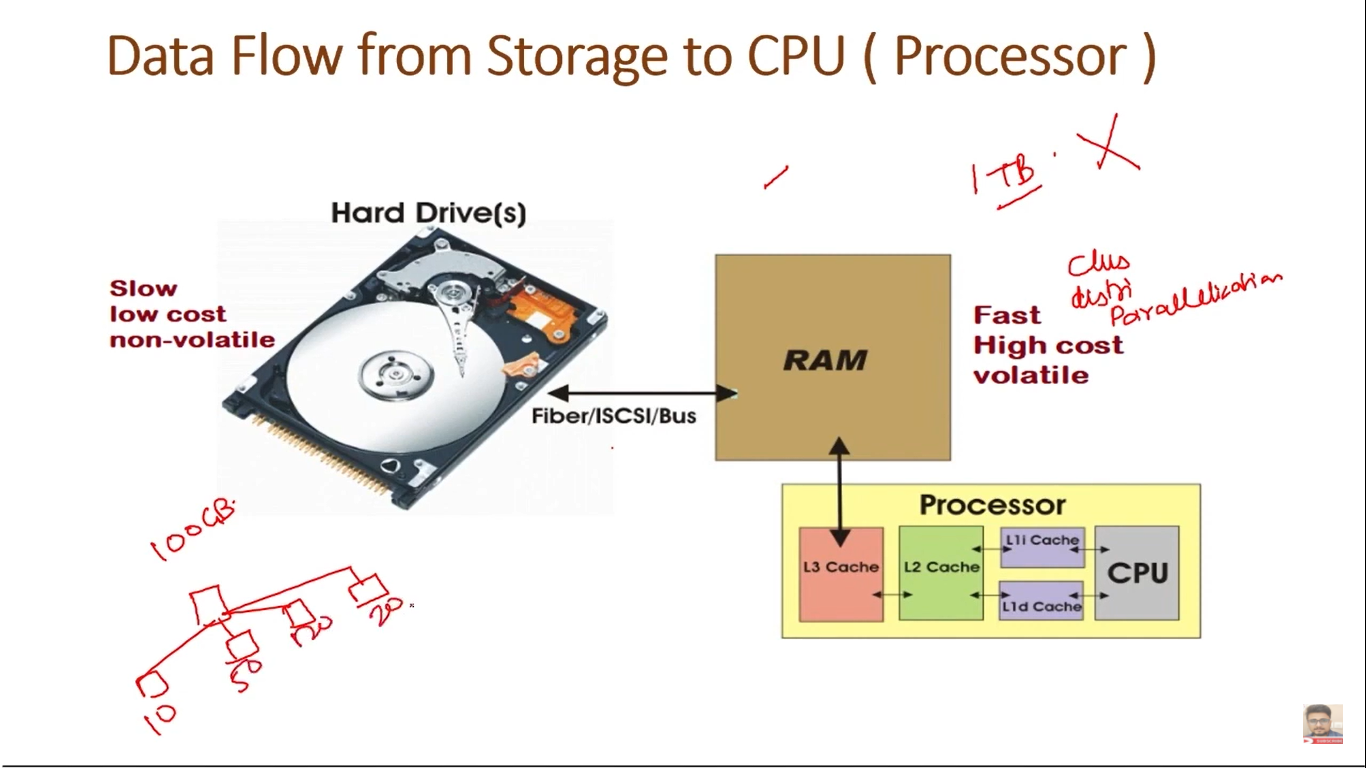
Processor always execute the operations on the data which is in the RAM.

The size of the file whether it is PDF, Excel, Movie file, Text File, must be less than the size of RAM. If the capacity of the RAM is larger than the size of the data, then there won’t be any problem because the RAM would be able to have the data inside its memory.

Whereas, if the capacity of the RAM is lesser than the size of the data, then it will not work.

So, to make this work, we can’t unnecessarily increase the RAM to 20GB, 50GB, or 1TB. This is not a normal/practical approach.

So, at this point we have to use something like **Cluster Environment**, **Distributed Environment** or **Parallelization**, so that we can distribute our work to different-different systems.



- Suppose this is a **Cluster**.

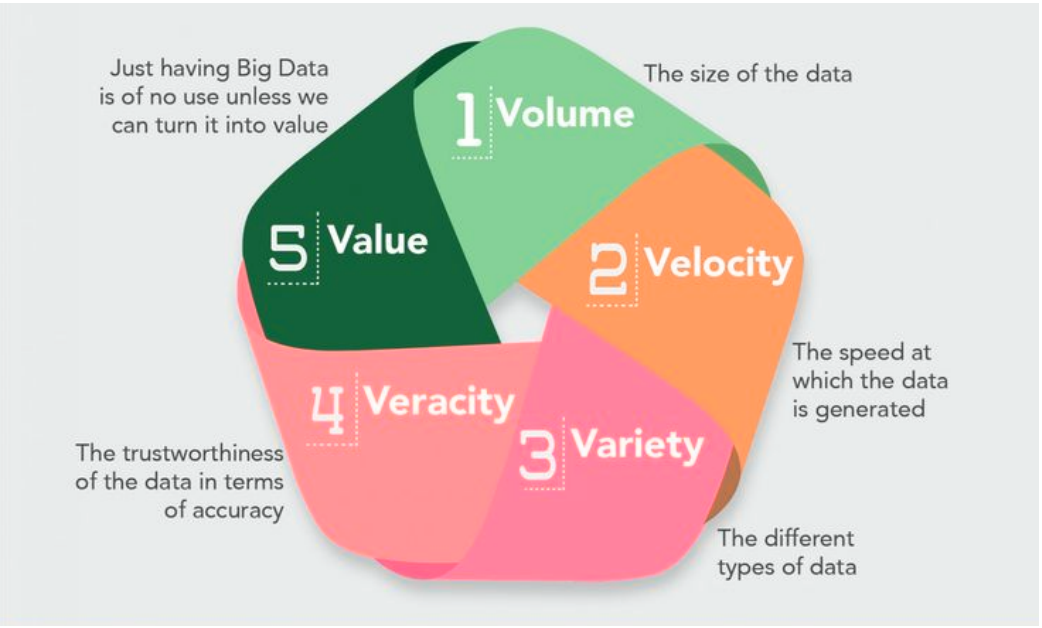
- We have 100GB of data.

- We can distribute it into 10GB, 50GB, 20GB, 20GB to each system.

- Now, it will take less time to read the data like, 10GB data, from the system instead of reading it from 100GB system.

- These data we call it as **Big Data**.

**5v of Big Data**



These are the classification parameter on which basis we classify the data as a Big Data.

Earlier we used to have only 3 things: **Volume**, **Velocity** and **Variety**.

We are considering these 3 terms to classify that data into Big Data.

But now we are using 5 terms.

1. **Volume:** The size of the data is high. So that we can classify as Big Data.

When our system is not compatible enough to store the data or to process the data, so we will classify that data as Big Data.

1. **Velocity:** Whenever data is getting increased, it is getting increased at very high speed, so we will say it as Big Data.
2. **Variety:** When we have variety of data like Structured data or Unstructured data, so we will consider it as Big Data.
3. **Veracity:** This is classified as a trustworthiness of the data in terms of accuracy. So, it refers to consistency and uncertainty in the data. That data which is available can sometimes get messy in the quality. In that scenario, the quality and accuracy are difficult to control. We will classify that data as Big Data.
4. **Value:** If we are doing analysis and we are performing some operations, so we are doing these things to extract some value from the data. So, if we are extracting some value, this is valuable data. So, we will classify that data as Big Data.

**Big data is the Problem, Hadoop is the solution**

People compare Big Data with Hadoop. It should not be like that.

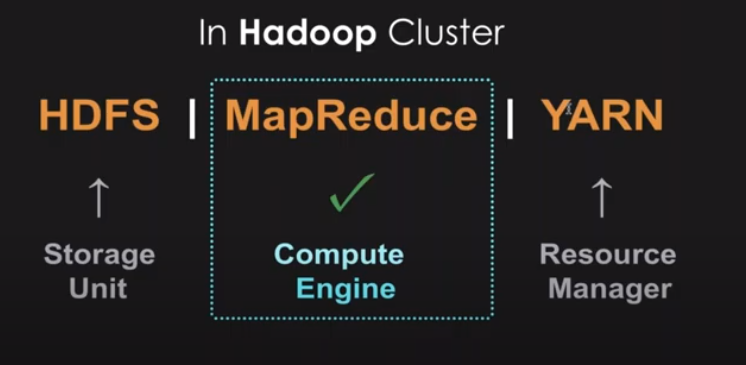
**Big Data is the problem**.

Let say, we have a Big Data and the volume of it are getting increased. Then we have the Velocity. A very high Velocity at which the data is getting generated each and every second.

Ex: Suppose 100MB or 1GB of data are getting generated every second then that is Big Data.

So, Big Data is the problem we have.

So, **to overcome that problem Hadoop is the solution** to that.



This is a picture of a Hadoop cluster.

In Hadoop Cluster, there are mainly 3 components.

i) HDFS ii) MapReduce iii) YARN

**i) HDFS (Hadoop Distributed File System)**

HDFS is just like a storage unit. This is not a database.

This is a file system. It is just showing in which format the data are getting saved.

It is a file system which provides high throughput test, to different applications which are running in the cluster.

I just partitioning the data in many machines.

So, it will just divide the data and it will store into partitions such as, Part-1, Part-2, Part-3, etc and it will depend on the size of the data.

**ii) MapReduce**

MapReduce is just a Compute Engine.

It is divided into 2 things: **Map** and **Reduce**.

**Map:** It is an operation which is performed in the parallel on the small portions of the dataset.

**Reduce:** It would combine all the results which are extracted from the Map.

**iii) YARN**

It is a Resource Manager.

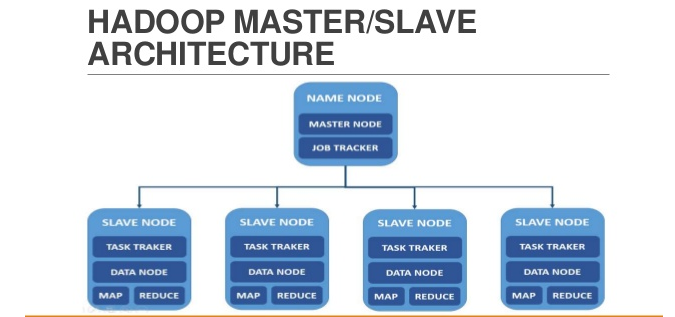
It is yet another resource negotiator.

YARN is present from the 2nd version of Hadoop. In the 1st version of Hadoop, it was not there.

In Hadoop, we have 2 things:

1. Resource Manager

2. Node Manager



Hadoop works on **Master** and **Slave**.

So, **1st** one is **Master** and **below 4** are **Slaves**.

So, in the **1st** one, we will **run** **Resource Manager**.

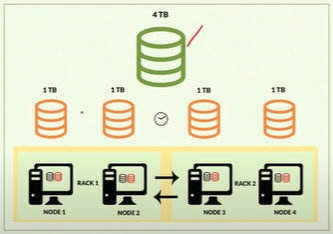
And, in the **below 4 slaves**, we will **run Node Manager**.

So, this is a Hadoop Cluster, how its look like.

Whatever the task will come, it will come first to **Resource Manager**. Then it will assign tasks to different Slaves.

It will find what data are in which node. So, all the below nodes will save the partitioned data. Ex: 10GB each node will carry.

**Distributed Clustered Environment**



- This is the same structure which we discussed.

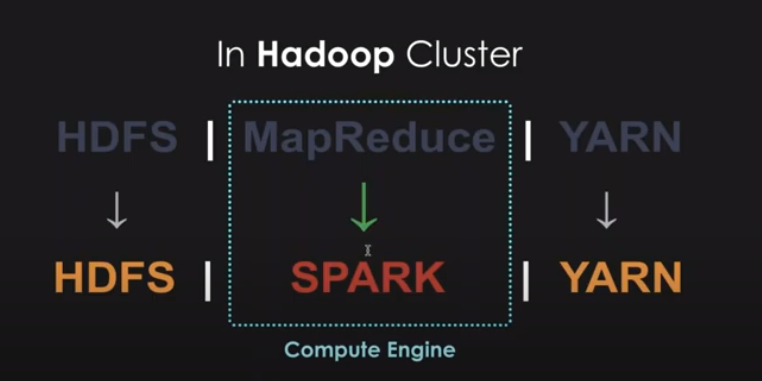
- We have 4TB of file and we have divided it into 1TB each and it will store into each Node.

- This is a Hadoop Cluster.











After Hadoop, Apache Spark got released.

Let’s discuss, at which point Apache Spark gets fit in the picture as compared to Hadoop.

So, in Hadoop cluster, we have replaced MapReduce with Apache Spark because MapReduce has some issues. It has some drawbacks.

**Note:** We are not replacing **Apache Spark** with **Hadoop**. We are just replacing **Apache Spark** with **MapReduce**.

So, the final cluster will be like,

**HDFS 🡪 Spark 🡪 YARN**

**Note:** If we are comparing **Hadoop** with **Spark**, then it is not a valid comparison. But, if we are comparing **Spark** with **MapReduce**, then it is a valid comparison.

**Note:** It is not necessary that we have to use HDFS which is by Hadoop. **If we do not want to use HDFS**, then we can use local data like, CSV file or Excel file or any other kind of file apart from HDFS and we can transfer that file to Spark.

**For YARN**, we can use Spark’s internal Resource Manager, or we can use MESOS or KUBERNETES.

So, this is a brief understand of how Spark is different from MapReduce.

**Note:** If we are using HDFS and YARN in our Spark Cluster, then we will say it as **Distributed Environment**.

Whereas, if we are not using HDFS and YARN in our Spark Cluster, then we can say it as **Standalone**.

So, **Spark can be deployed in 2 modes** i.e., ***Standalone*** and ***Distributed***.



This is the typical process which follows:

1. We have to define the Source of the data.

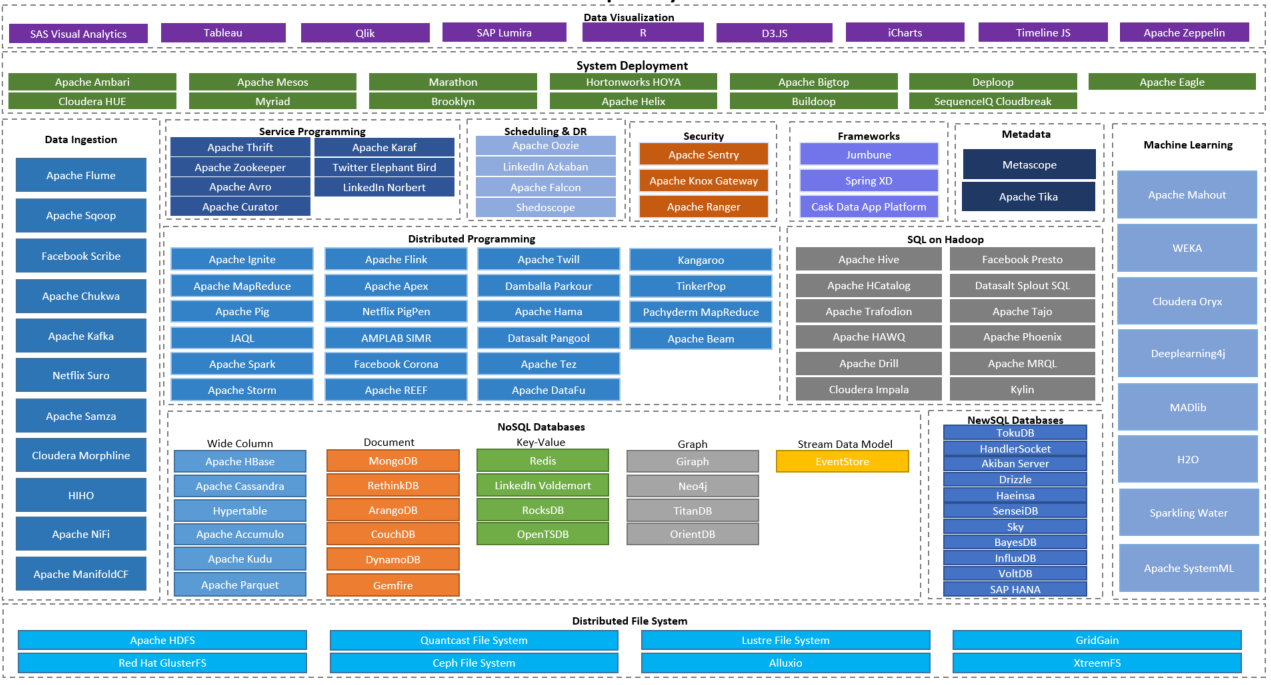
2. Then we have to Ingest. We have to integrate the data and have to store in one place.

3. Then we process the operation in the data. Here, it comes the MapReduce or Spark.

4. Then we store the data in some format.

5. Then we serve the data to the user or any application which requires.

**Hadoop Ecosystem**



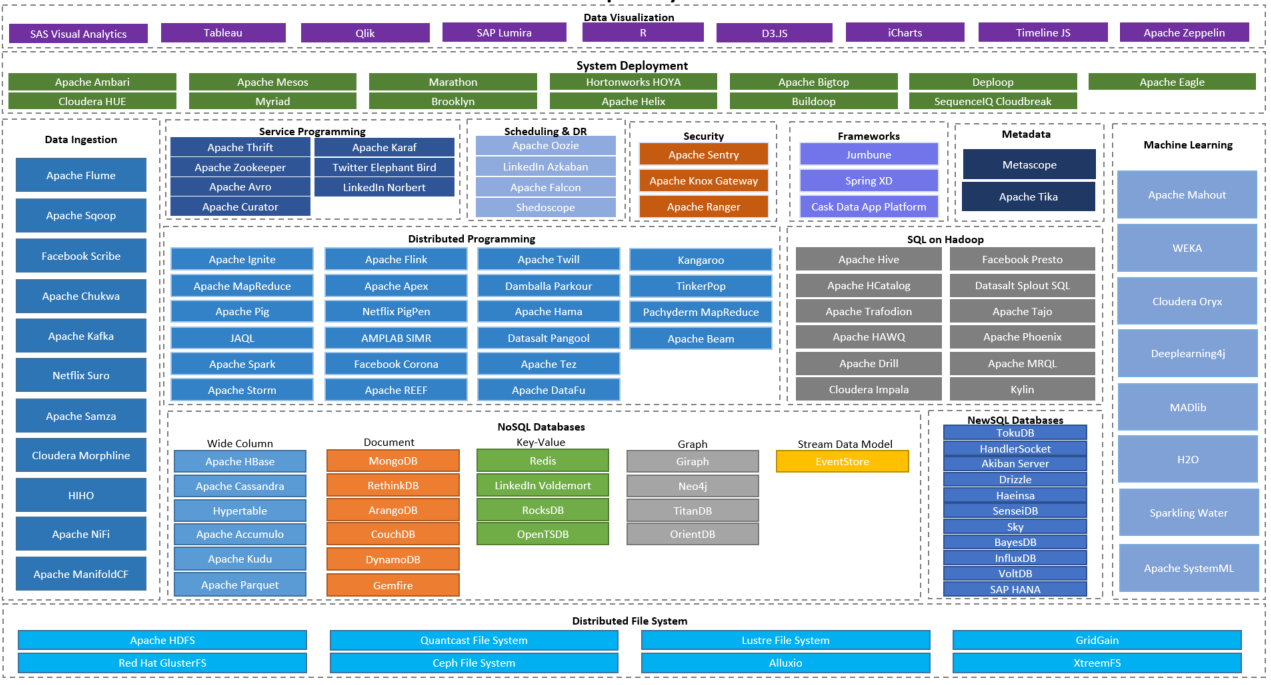
We have here:

**Distributed File System:** We can use so many file systems. The default is Apache HDFS for Hadoop. And we can use all other systems.

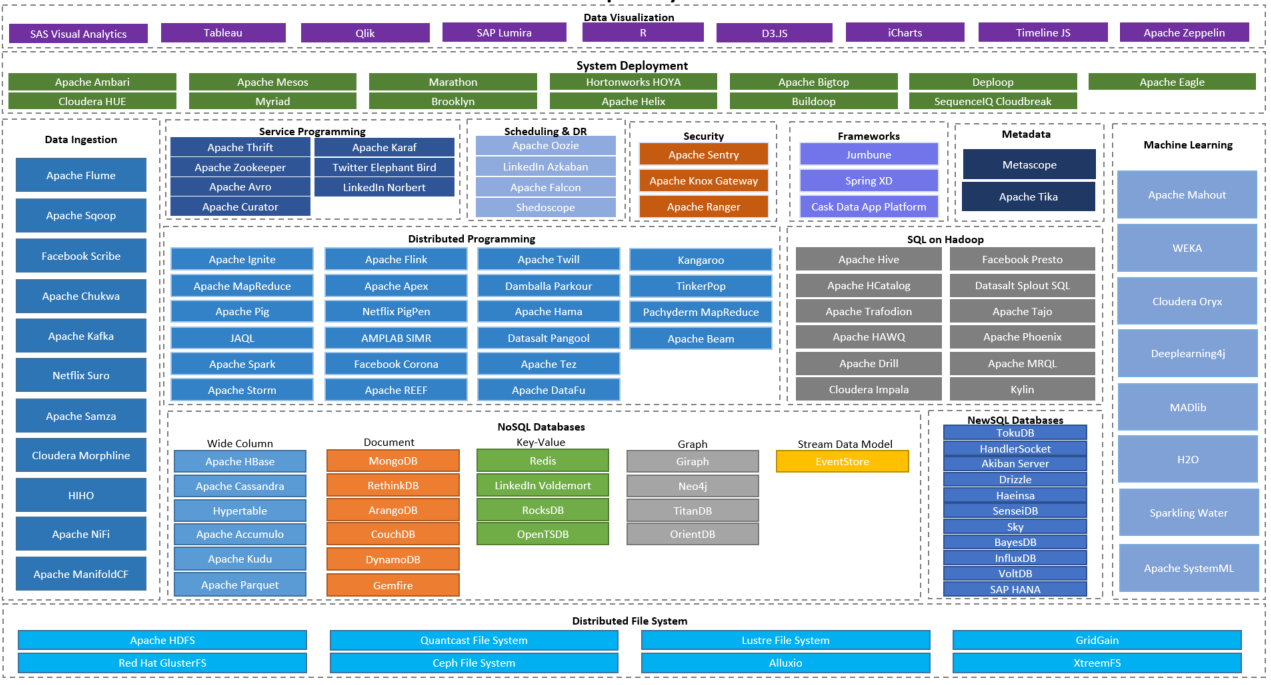
Then we have to use some database.

We can use **NoSQL Databases** or **New SQL Databases**

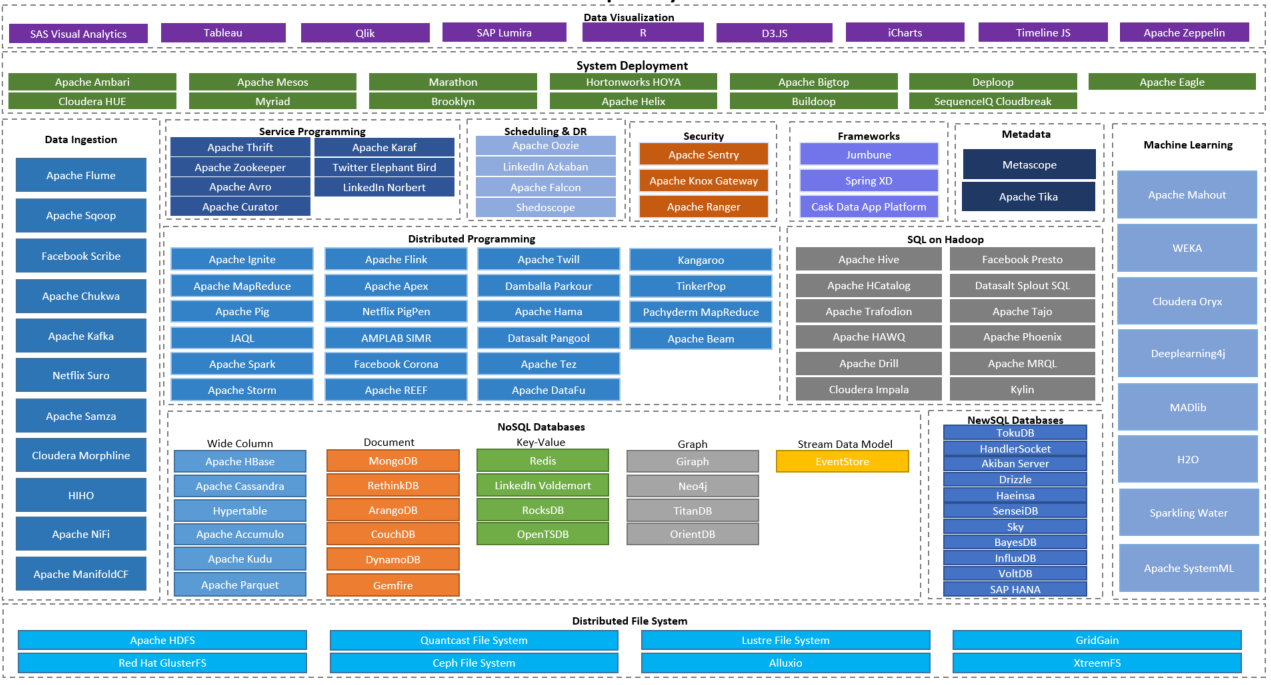
In **NoSQL Database,** we have

****

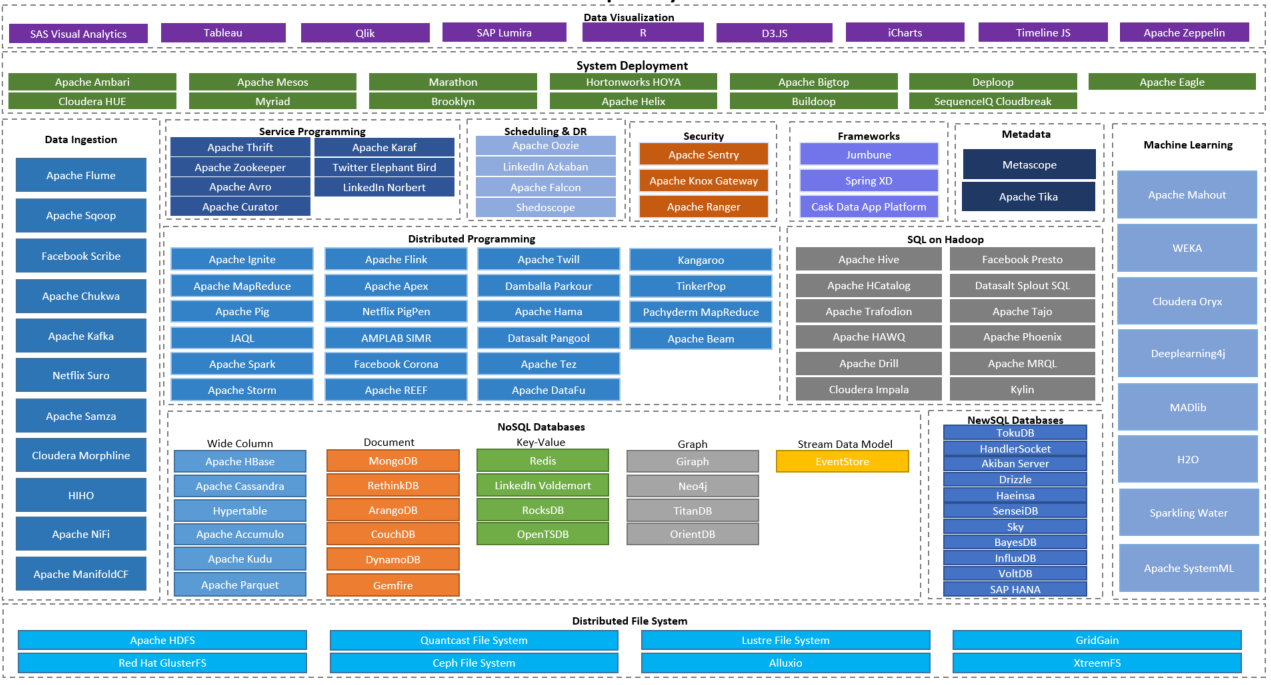
&, in **SQL Database,** we have

****

**Data Ingestion:** This will collect all the data.

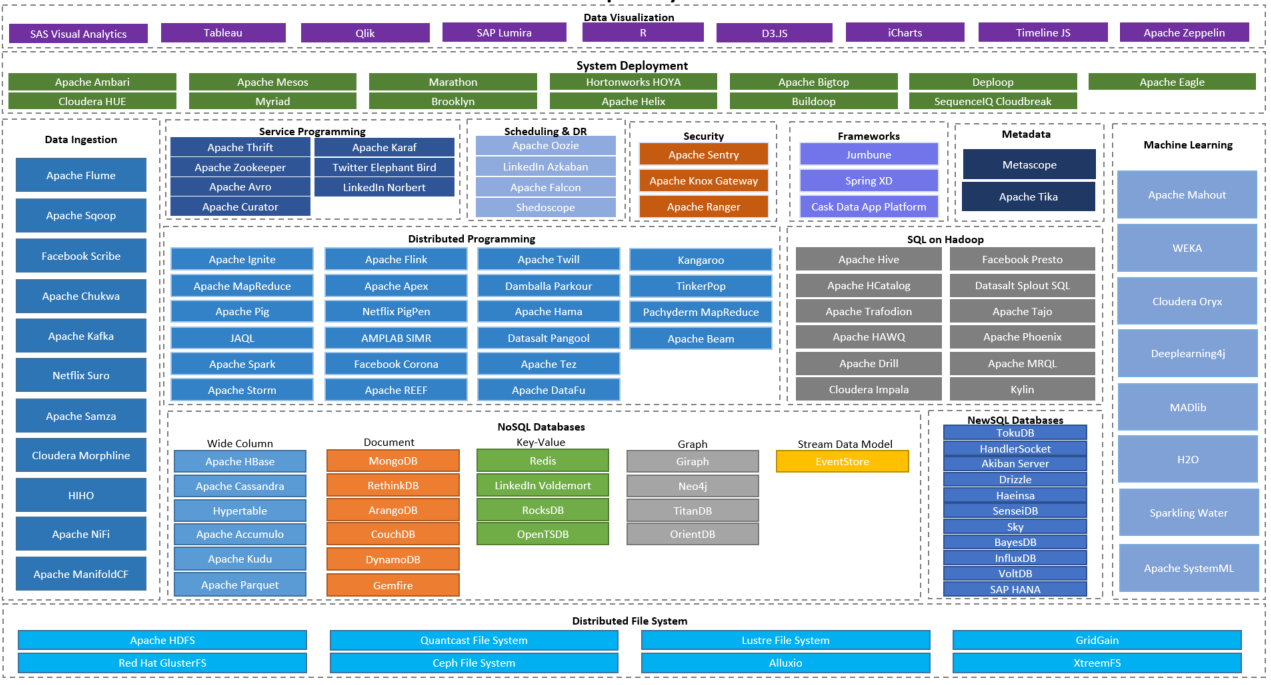
****

**Distributed Programming**

****

Here, we have Apache Spark.

**SQL on Hadoop**

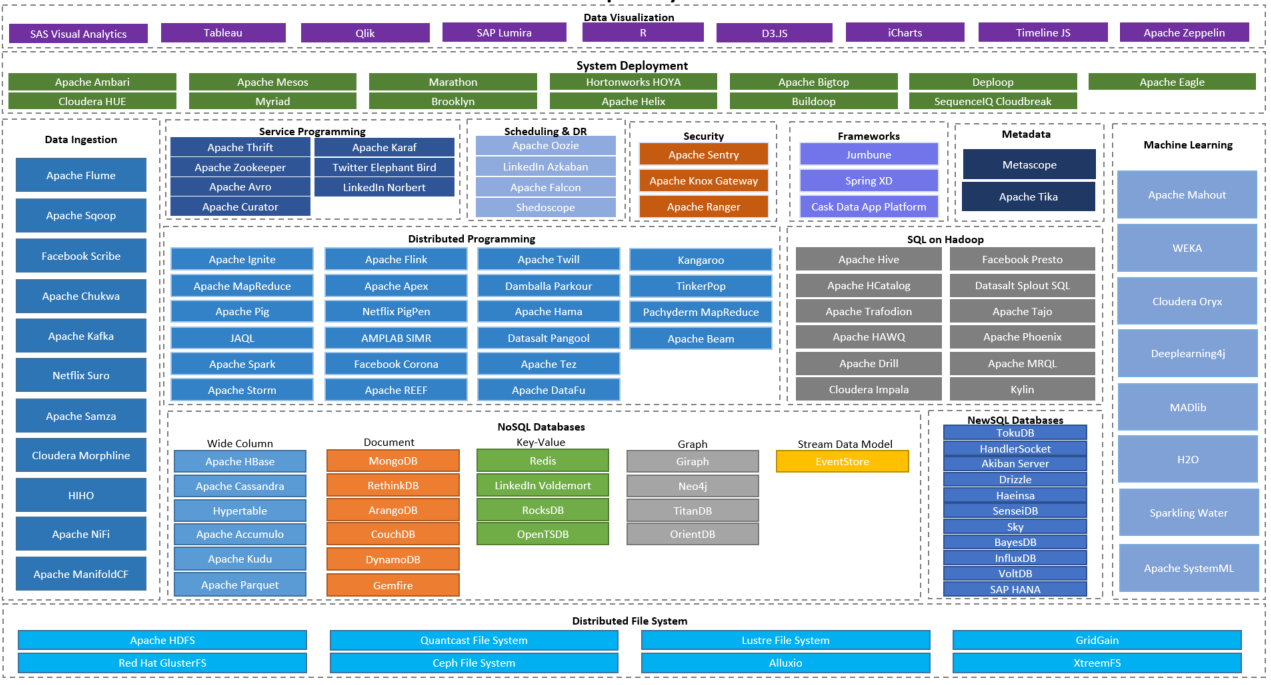
****

If we want to use SQL queries we do this because Hadoop is Java base.

If we have to write any code in Hadoop or MapReduce, then it would be in Java.

**Machine Learning**

If we want to use Machine Learning in Hadoop, then we have to use below frameworks. All we have to connect with the Hadoop. So, we have to use many connectors.



**Security:** If we want to do security, then we can use this.

**Scheduling and DR:** If we have to schedule any job then we have to use Oozie.

**Data Visualization:** If we want to do any Data Visualization then we can use these many frameworks.

**Service Programming**

**Frameworks**

**Metadata**

**System Deployment**

**Imp Note:**

- So, what we can conclude here is, Hadoop Ecosystem is much complex.

- Hadoop is only for **batch processing**.

- If we want to use Hadoop for **stream processing,** then there is no inbuilt support in Hadoop. So, we have to use **Kafka,** or we have to use **Flume**.

- If we want to schedule anything then we have to use Oozie.

- If we want to use Machine Learning libraries, then we have to use these many libraries which is available in the screenshot.

- If I have to use SQL, then we have to use given SQL options in the screenshot.

- Hadoop has only Java support.

- Hadoop has no support for other languages.

- If I want to migrate the data or database transfer, then we have to use **Sqoop**.

So, here we have **Spark** because **Spark have inbuilt everything**.

- It has Machine Learning libraries

- It has SQL.

- It has real time data processing support.

So, that’s why **Spark** is getting used comparatively to Hadoop.