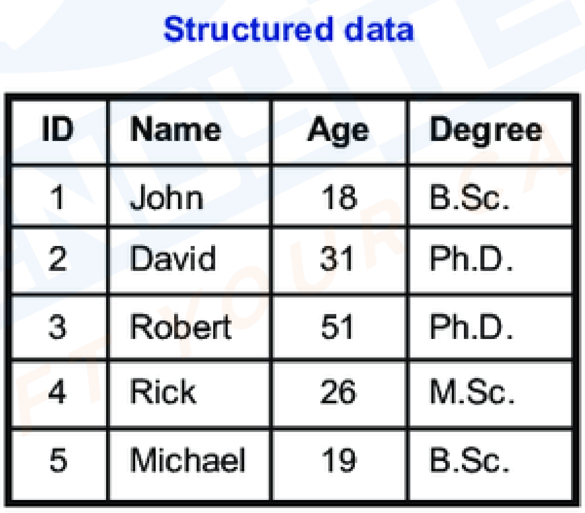
What is Big Data?

There are several informal definitions for describing Big Data, like - Data that cannot be stored/processed on a single computing system. However, IBM provides a formal definition for Big Data.
“**Data that is characterised by 3V’s / 5V’s is considered to be Big Data**.”
These V’s include -

1. **Volume**
   Data that is so huge that it cannot be stored / processed on a single conventional system is considered as Big Data. Ex- Data collected on Social Media / Data from sensors which are in several petabytes.

2. **Variety**
   Data can be coming in different formats and we have to handle it.

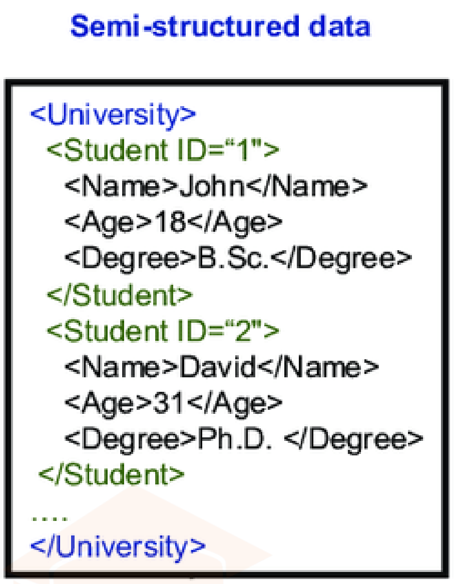
   ->**Structured** - Data in such formats have some schema associated with it. Ex- RDBMS / Database tables, where the data is stored in tables in the form of rows and columns (Employee table in a MySQL DB)



Structured data

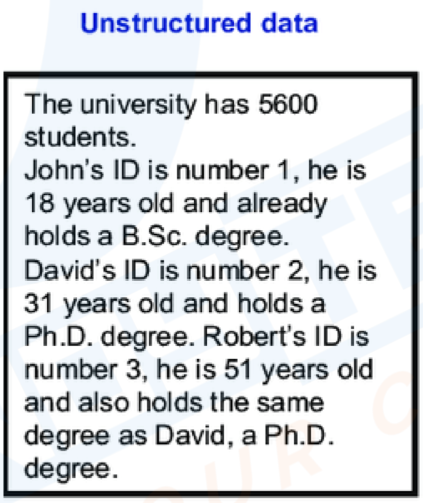
ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

   ->**Semi-Structured** - Data in such formats have some partial schema associated with it and are not stored in the form of rows & columns. Ex- CSV, JSON, XML


Semi-structured data

```
<University>
 <Student ID="1">
   <Name>John</Name>
   <Age>18</Age>
   <Degree>B.Sc.</Degree>
 </Student>
 <Student ID="2">
   <Name>David</Name>
   <Age>31</Age>
   <Degree>Ph.D. </Degree>
</Student>
....
</University>
```

->**Unstructured** - Data in such formats have no schema associated with it. Ex- Image, Log files, Videos.


Unstructured data

The university has 5600 students.
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

3. **Velocity**
   The speed at which new data is coming in. Let's consider Amazon is running a sale and has to present the sales that have happened in the last 1 hour to its Business Development team. System should be able to accommodate the new huge volumes of data coming in and generate some insights for Business decisions. Ex- How many iphones were sold in last 1 hr

4. **Veracity**
   Is the quality or correctness of Data. Data cannot always be in the right format and we should be able to handle this not so high quality data.

5. **Value**
   Should be able to generate some value out of the data collected by processing it for making Business Decisions.

Why is there a need to learn a new technology stack to handle Big Data?
The traditional / conventional technologies are not capable of handling Big Data.
To understand why, lets first know what is -

Monolithic Vs Distributed System

Monolithic System - One Big System holding a lot of resources and thereby high computing power.

What is a Resource ?
Compute - CPU cores (Ex- 4 CPU core)
Memory - RAM (Ex- 8GB RAM)
Storage - Hard Disk (Ex- 1TB Hard Disk)

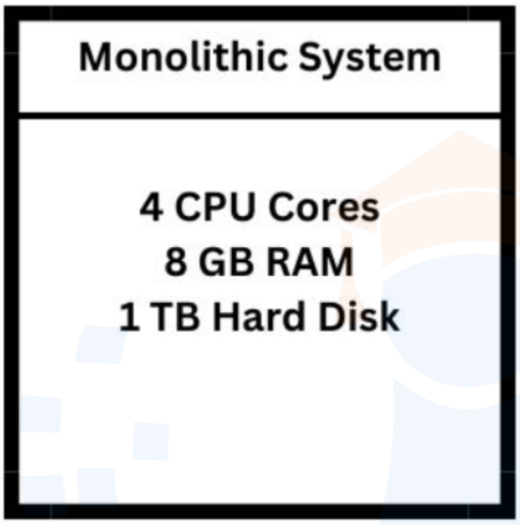
Consider you have, X Resource ->(giving) Y Performance
We assume that 2X Resources ->(Should Give) 2Y Performance
-But after a certain point, due to hardware limitation, even after increasing the resources, equivalent performance benefits are not achieved as per assumption.
-Performance does increase but not in the same ratio as increasing the resources/ amount spent.
-Monolithic is not a scalable system as the performance does not increase in the same ratio as resources.
-Monolithic follows Vertical Scaling of increasing the computing power of one Machine only by increasing the resources of that system.

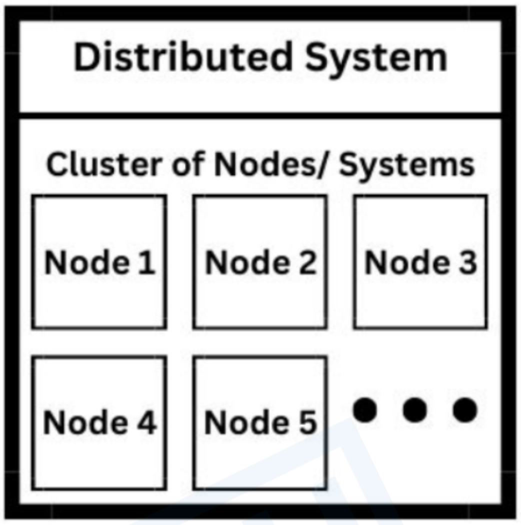
Distributed System - Is a cluster of systems grouped together with each holding its own resources. Computing power of such a system is a sum of computing power of all the nodes in the cluster.
Ex - 5 Node cluster, with each Node/ System holding 4CPU core, 8GB RAM and 1TB Hard Disk

-In case you need to scale up such a system, you just need to bring in more Nodes/ Systems.

-Distributed System follows Horizontal Scaling that involves increasing the number of Nodes/Systems. Such a scaling is True Scaling because if you double the resources, the performance will also double proportionately.

-All of the Big data Systems are based on Distributed architecture and not Monolithic in order to achieve True Scaling.





In order to design a good Big Data System, the following 3 things need to be considered.

1. **Storage -** Where will such huge data be stored as traditional systems cannot handle such data. That is we need to have Distributed Storage.
2. **Processing / Computation** - Traditional systems will work when the data resides on a single machine where it has to be processed. However, Big Data is stored / distributed across the cluster on multiple nodes. Therefore, the traditional programming approach doesn't work and we need Distributed Processing.
3. **Scalability** - The system should accommodate the growing demands. Therefore, a scalable model is required. A Distributed Storage and Processing System by default supports Scalability.

3 factors for designing a Big Data System

-Where is the data going to be stored

-How is the data going to be processed

-Is the system Scalable.

Overview of Hadoop

What is Hadoop?

Hadoop was the first framework designed to solve Big Data problems. It is a framework because It is not just a single tool but a combination / ecosystem of several tools and technologies to solve Big Data problems.

Evolution of Hadoop

2007 : Hadoop 1.0
2012 : Hadoop 2.0
Current version : Hadoop 3.0

3 Core Components of Hadoop

HDFS | MapReduce | YARN

Hadoop Distributed File System (HDFS) - Is a Distributed Storage where the data spans across multiple systems.

MapReduce - Is Distributed Processing. Since the Data is stored across multiple nodes, conventional programming models cannot process such data. Therefore MapReduce to process the data stored using Distributed Storage.

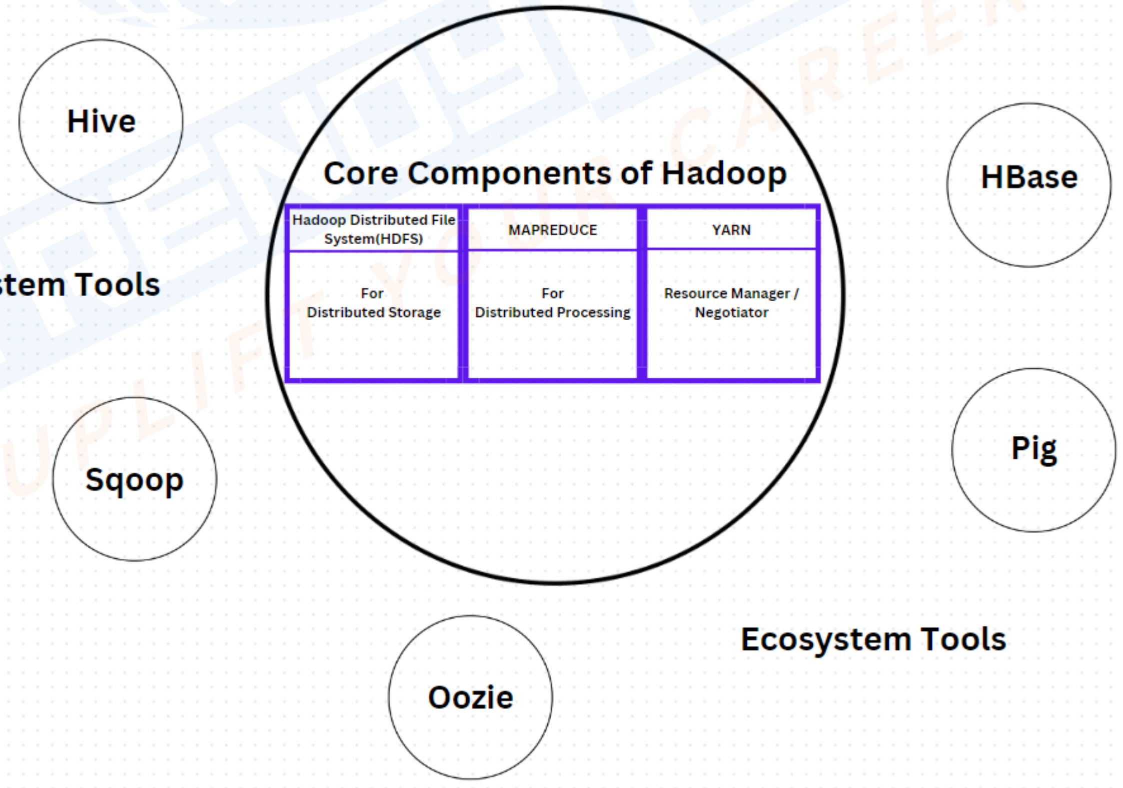
YARN - Is a Resource Manager that negotiates for resources to allocate for the processes.
Let say you have a 20 node hadoop cluster with many users requesting for resources. Then, a resource manager is required to manage and allocate resources as per the request.

Core Components of Hadoop

Hadoop Distributed File System(HDFS)	MAPREDUCE	YARN
For Distributed Storage	For Distributed Processing	Resource Manager / Negotiator

-*Writing MapReduce code is hard as it involves coding in Java, which would require many lines of code even to perform a small task. Therefore, MapReduce is obsolete at the moment and there are other alternatives that make things easier.*

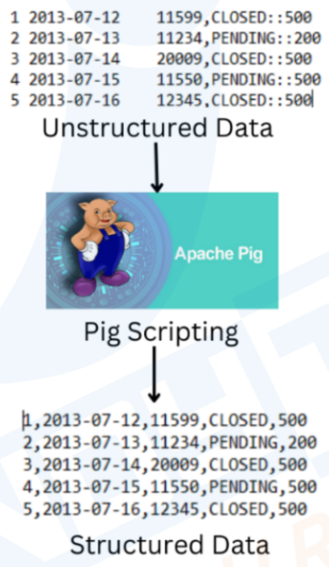
Hadoop Core & Ecosystem technologies


Hive
Core Components of Hadoop
HBase
Hadoop Distributed File System(HDFS)
MAPREDUCE
YARN
For Distributed Storage
For Distributed Processing
Resource Manager / Negotiator
Ecosystem Tools
Sqoop
Pig
Oozie
Ecosystem Tools

Apart from the Core components of Hadoop, the Ecosystem technologies come into picture to make things easier for processing.

Sqoop - is used for Data Movement. Suppose you have data in a Relational Database and you want to bring it to HDFS / Vice-Versa then the ecosystem tool called Sqoop comes handy. Sqoop is a MapReduce under the hood. Cloud based alternative for Sqoop is Azure Data Factory(ADF).

Pig - Mainly used to Clean the Data. It is a scripting Language. Pig also uses MapReduce under the hood.



```
1 2013-07-12     11599,CLOSED::500
2 2013-07-13     11234,PENDING::200
3 2013-07-14     20009,CLOSED::500
4 2013-07-15     11550,PENDING::500
5 2013-07-16     12345,CLOSED::500
```

```
1,2013-07-12,11599,CLOSED,500
2,2013-07-13,11234,PENDING,200
3,2013-07-14,20009,CLOSED,500
4,2013-07-15,11550,PENDING,500
5,2013-07-16,12345,CLOSED,500
```

Hive - Provides a SQL kind of interface to query the data. Hive is also a MapReduce underneath.

Oozie - It is a workflow scheduler.
Say we need to define a workflow with 3 MR jobs. Suppose we need M1 and M2 to run in parallel and then M3 should run, such workflows can be scheduled using Oozie. Cloud based alternative for Oozie is Azure Data Factory.

HBase - Is a NoSQL database. NoSQL is used in cases where very quick search of records from the data is required. Random access of records instantly is possible with Hbase. Cloud alternative for Hbase is CosmosDB

The Big Data Era Started with Hadoop but it is slightly losing its position. HDFS and YARN are still in use but MapReduce is Obsolete.

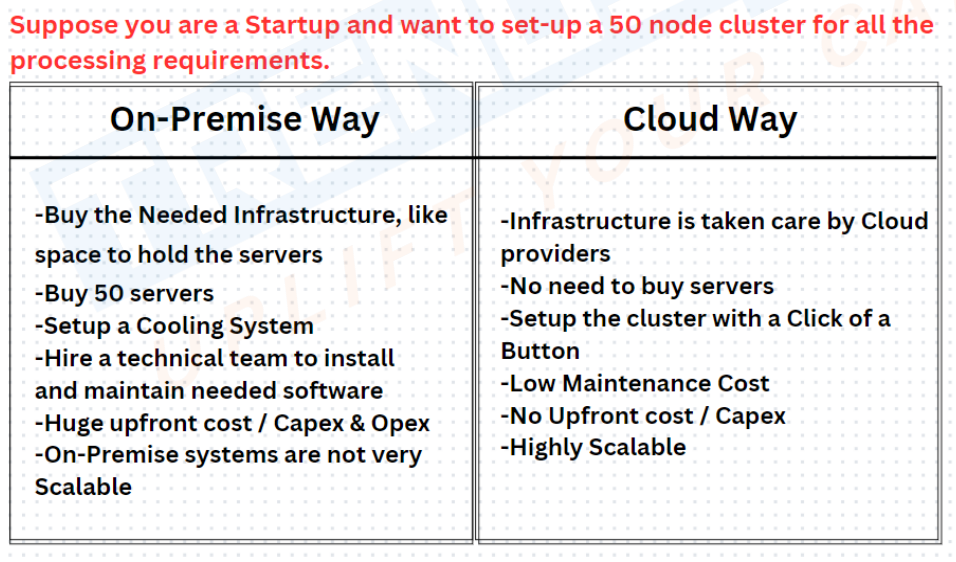
Challenges with Hadoop

1. **MapReduce** is very slow and hard. It is an overkill as one has to write large no.of lines of code to achieve even a simple task.
2. **Need to learn Different components to achieve different tasks** and each component has its own big learning curve. Like for data cleaning, we need to learn pig. For data querying, we need to learn Hive and many more.

==============================

Cloud and its Advantages

On-Premise Vs Cloud


Suppose you are a Startup and want to set-up a 50 node cluster for all the processing requirements.

On-Premise Way	Cloud Way
-Buy the Needed Infrastructure, like space to hold the servers -Buy 50 servers -Setup a Cooling System -Hire a technical team to install and maintain needed software -Huge upfront cost / Capex & Opex -On-Premise systems are not very Scalable	-Infrastructure is taken care by Cloud providers -No need to buy servers -Setup the cluster with a Click of a Button -Low Maintenance Cost -No Upfront cost / Capex -Highly Scalable

Advantages of Cloud

Scalable - System should be easily upscaled or downscaled as per the requirement.
CapEx Vs OpEx - With cloud, you will only have to pay for the services as per your usage (i.e., minimal operational Expenditure) unlike as in On-premise systems that needs huge CapEx (Capital Expenditure)
Agility - To set up an On-Premise cluster, it would need months to bring it up. However, in the case of cloud, the same can be setup in a few minutes with just a click of a button which is more of an agile approach.
GeoDistribution - A latency or network lag will be experienced if a user is requesting for a service in India but the server is placed in the US. This can be solved by having servers located in multiple locations to distribute the traffic as per the location from where the request is coming in.
Disaster Recovery - As the systems are spread all across the world, even if a system goes down in one location, a backup node can serve the request.
Cost Effective - You will pay only as per your usage and no upfront capital expenditure is involved.

Cloud Types

1. **Public Cloud -** If the data at hand is not sensitive / confidential, then public clouds like Amazon's AWS, Microsoft's Azure / Google's GCP can be used to store and process such data.
2. **Private Cloud -** If the data is very confidential (like Banking Domain Data) then private clouds are used by such organisations where they have their own cloud setup that is not shared by any other organisation.
3. **Hybrid Cloud -** Is a combination of Private and Public cloud. If the company has both generic and confidential data then they can use Hybrid cloud. All the confidential information goes to the private cloud and all the generic data goes to the public cloud.

On-Premise Vs Private Cloud :
In the case of On-Premise systems, there is a need to setup everything from scratch right from procuring resources, maintenance… However, with Private Cloud, there are still advantages of cloud way of handling

things but it is meant only for a specific company or organisation. Private cloud is basically an On-Premise setup on cloud.

==================================

What is Apache Spark?

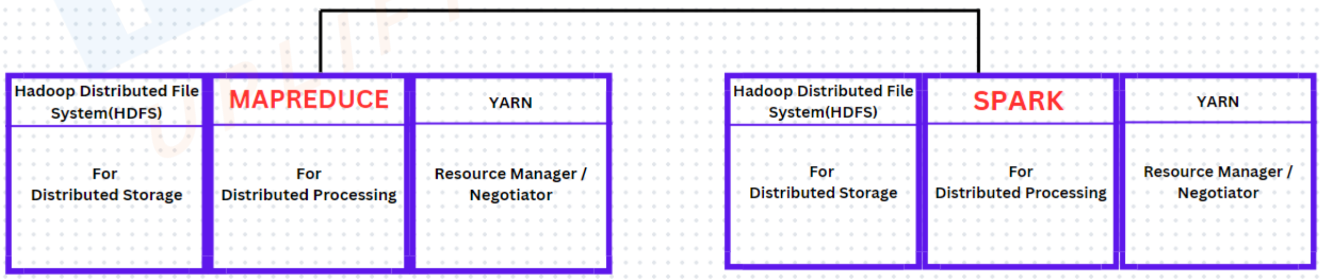
As learnt earlier, MapReduce was a bottleneck as it was slow and needed a lot of coding effort even to achieve a simple task.

This is where Apache Spark comes to rescue.
Apache Spark is a
-General Purpose
-In Memory
-Compute Engine

Is Apache Spark a replacement for Hadoop?
Explanation:
Hadoop provides 3 components

Storage (HDFS)
Compute (MR)
Resource Management (YARN)

Spark can act as a replacement for MapReduce and not Hadoop.
Spark is like a plug and play compute engine, it needs a Storage and a Resource Manager to function and is not bound to HDFS and YARN only.


Hadoop Distributed File System(HDFS)
For Distributed Storage
MAPREDUCE
For Distributed Processing
YARN
Resource Manager / Negotiator
Hadoop Distributed File System(HDFS)
For Distributed Storage
SPARK
For Distributed Processing
YARN
Resource Manager / Negotiator

Spark needs 2 components to work with

1. **Storage -** Ex: HDFS | Amazon S3 | Azure ADLS Gen2 | Google Cloud Storage | even local storage | ..
2. **Resource Manager -** Ex: YARN | Mesos | Kubernetes

Spark in 10x to 100x faster than traditional Mapreduce as it stores and processes the data In Memory

In which language do developers write Spark code?

PYTHON | SCALA | JAVA | R

(Spark itself is written in Scala)

Spark with python is called PySpark

==================================

Database Vs Data Warehouse Vs Data Lake

Database

->It deals with day to day transactional data

->Meant for OLTP (Online Transactional processing)

->Databases majorly deal with structured data, which has a rigid structure associated in the form of rows and columns.

->It can hold only recent data for better performance. It cannot hold historical data of years.

->Best use case for Databases - Online Banking Transactions

->Example of Databases - Oracle, MySQL,....

->It follows Schema on Write - While writing the data, the schema will be validated i.e., whether the data is in the same format as the structure / schema of the table in the Database. If not, it will throw an error.

->Cost to store the data in the database is high.

Data Warehouse (DWH)

-> Such a system is mainly used for analytical processing and not day to day operations.

-> It deals with a lot of historical data of many years to find insights.

-> Running complex queries on the database with an intent to do analysis, then the day to day transaction services become slower.

-> Data is brought in from multiple sources to the Data Warehouse.

-> Data warehouse also deals with structured data and follows Schema on Write for validating the data.

-> Storage cost is high but relatively lesser than Database

-> Example of Data Warehouse - Teradata.

-> Data Warehouse follow an ETL process -

Suppose your data is in the Database and you have to bring it to the Data Warehouse for Analysis purposes. Then, the data needs to go through the following

1.Extract - Collect/Extract the data from various sources

2.Transform - Do some complex processing and convert the data into a desired form. (Like adding and or removing the columns, etc)

3.Load - load the data to the data warehouse.

Challenges of Data Warehouses

- Transform is a complex process and needs a lot of time.
- Also, while loading we are not aware of what transformations are needed. However, it becomes compulsory to perform the transformation even before loading that reduces the flexibility in dealing with Data.

Therefore, Data Lake came to the rescue to overcome these Challenges !

Data Lake -

-> Data Lake is also meant to get insights from huge volumes of data just like Data Warehouse.

However, in case of Data Lake, data can be present in its raw form (either Structured / Unstructured)

Ex- Log Files, these files can be present in their raw forms in the Data Lake.

For this, i.e., to have the data in its raw form, Data Lake follows the

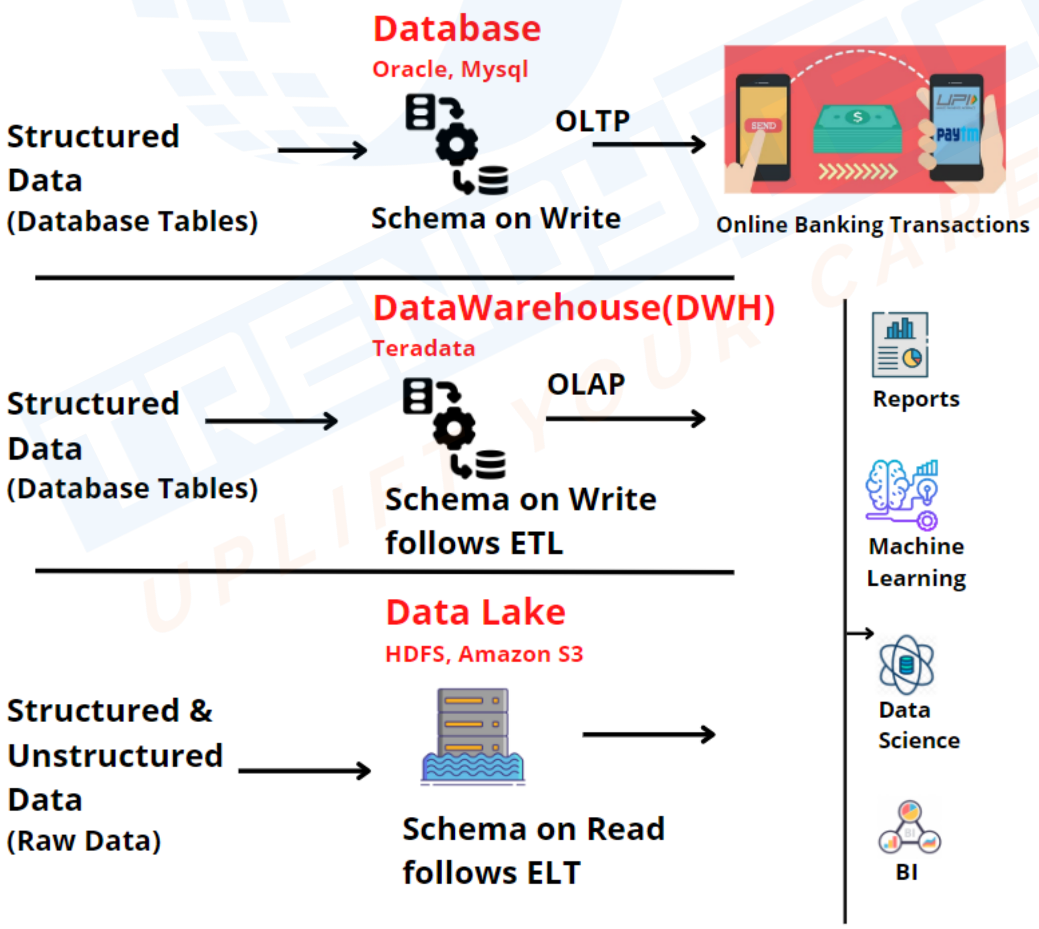
ELT process - Extract, Load & Transform

(Suppose we have a log file, we first Extract it, Load it to the Data Lake and then think of transformation before using as per the requirement, we think of transformation only when required after loading)

->Examples of Data Lakes which can handle Raw Data in its original form are HDFS, Amazon S3,..

->Cost Effective - Storage cost is very low due to which years of historical data can be stored without worrying about the cost

->Follows Schema on Read - Suppose that we have an Employee.csv file, we attach some schema over this data file to visualise the data. The structure is associated only when the data needs to be visualised.

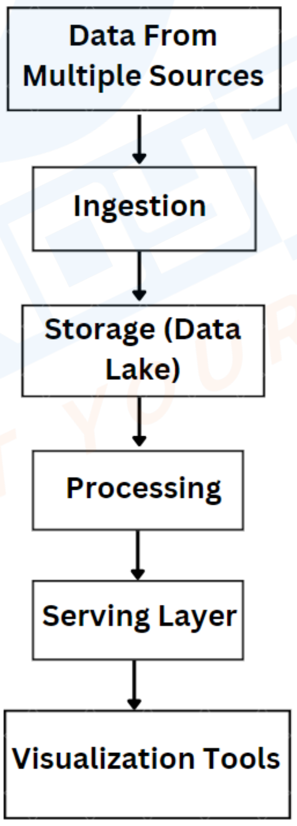

Database
Oracle, Mysql
Structured Data (Database Tables)
OLTP
Schema on Write
Online Banking Transactions
DataWarehouse(DWH)
Teradata
Structured Data (Database Tables)
OLAP
Schema on Write follows ETL
Data Lake
HDFS, Amazon S3
Structured & Unstructured Data (Raw Data)
Schema on Read follows ELT
Reports
Machine Learning
Data Science
BI

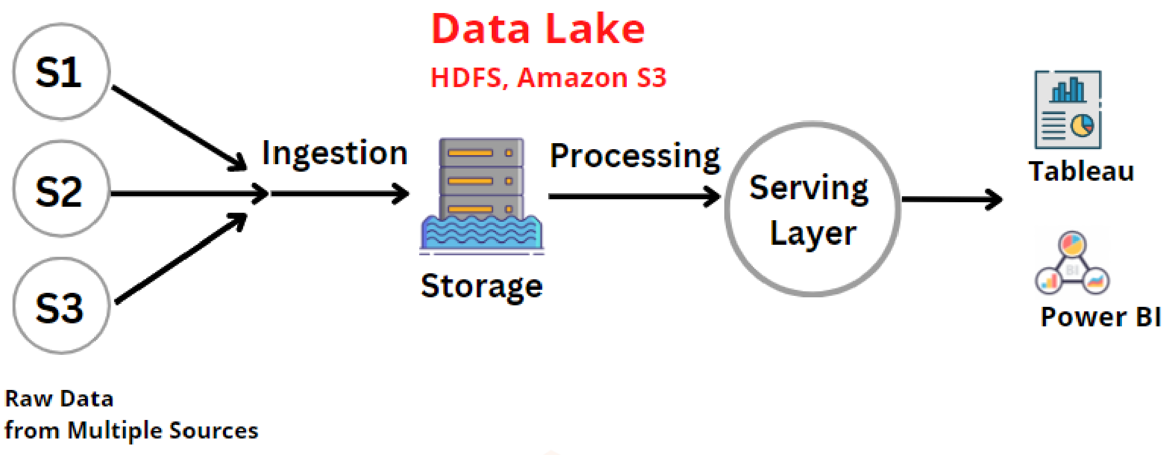
Big Data - The Big Picture

Data Engineering Flow -

Consider we are getting data from multiple sources like database, data warehouse, sensor data, social media data, etc.

->Firstly, You want to bring all the data from different sources to the storage (i.e., Datalake here)
->Secondly, we need an Ingestion framework to ingest the data from the different sources and bring it to storage
->Thirdly, we will be processing the data after it has been ingested to the storage. Here we are following an ELT process, where we load the data first and then perform various transformations(like cleaning, aggregation, joins,..) as per the requirement.
->Fourthly, the processed results will be put into the serving layer to which the visualisations tools(Like Tableau, PowerBI) are connected for viewing the results graphically.





Data Pipeline on Hadoop

1. **Data Pipeline workflow Visualisation for On-premise :**

Database(source) -> Sqoop(Ingestion) -> HDFS(storage) -> MapReduce/Spark(processing) -> Hive(Serving)
[Employee table in RDBMS - > Sqoop import -> CSV File -> Processing -> DataWarehouse]
Data Warehouse / Database Vs NoSQL in Serving Layer -
Database/ Data warehouse is used when there is a need to do Reporting.
Vs
NoSQL is used when building Custom UIs where fast retrieval is required.

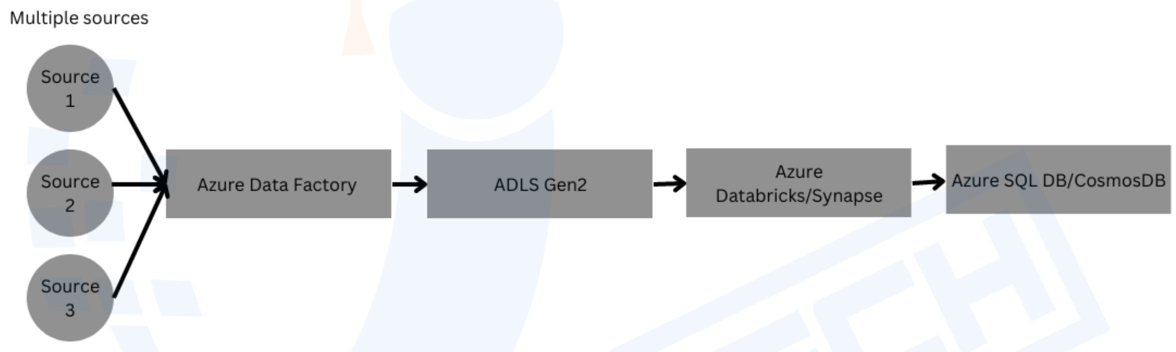
MySQL RDBMS -> Sqoop -> HDFS -> Mapreduce/Spark -> Hive/MySQl -> Tableau

MySQL RDBMS -> Sqoop -> HDFS -> Mapreduce/Spark -> Hbase -> Custom UI

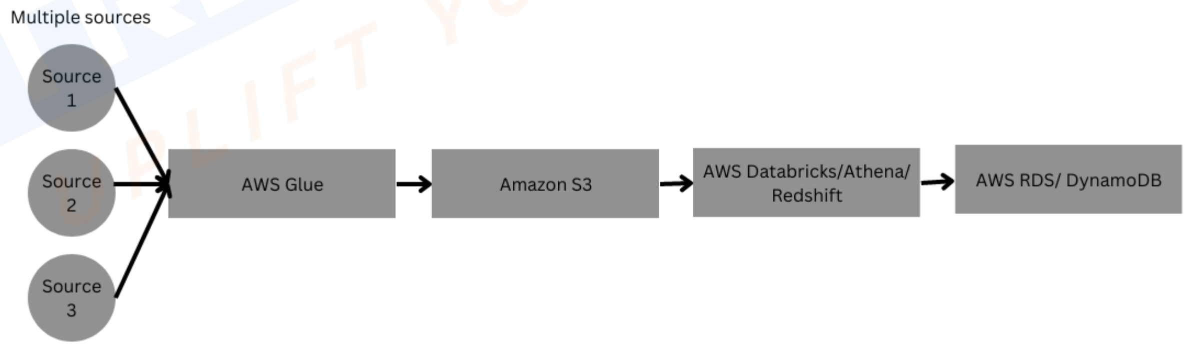
2. Data Pipeline workflow Visualisation for Cloud :

Multiple Sources -> Azure Data Factory-ADF(Ingestion) -> ADLS Gen 2(storage) -> Azure Databricks / Synapse(processing) -> Azure SQL / Cosmos DB(serving)

End-to-End Big data pipeline in Azure



End-to-End Big data pipeline in AWS



Phases / Stages of Data Pipeline	Example Data Pipeline Workflow tools and technologies for On-Premise and Cloud			
		On-Premise (Hadoop)	Azure Cloud	AWS Cloud
	Source	MySQL Database table	Multiple sources	Multiple sources
	Ingestion	Sqoop	ADF	AWS GLUE
	Storage	HDFS	ADLS Gen2	Amazon S3
	Processing	MapReduce / SPARK	Azure Databricks / Synapse	Athena / Redshift
	Serving layer	HBase	Azure SQL / Cosmos DB	AWS RDS / Dynamo DB

Categories of Computation

Serverless computing

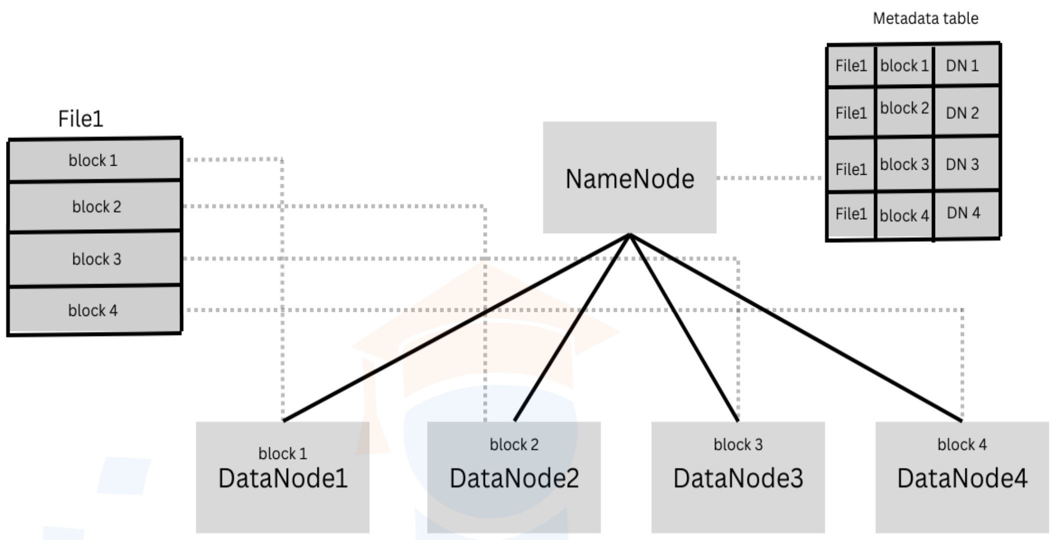
- Resources are not dedicated, hence no guaranteed performance.
- Less expensive than serverful computing.
- Good for scheduled jobs that are not particular about the time in which it gets finished.
- Eg: Athena, Synapse Serverless

Serverful computing

- Guaranteed performance since resources are dedicated.
- More expensive than serverless computing.
- Good for ad-hoc jobs that require immediate execution.
- Not suitable for scheduled jobs that can tolerate some delay.
- Eg: Redshift, Synapse Dedicated Pool

==============================

HDFS Architecture



Name Node -

Name node is the Master node that acts like an index to where the actual data is retained. It holds the Metadata table that keeps track of what is stored where.

Request from a Client (to read a file) -> Namenode -> Name node provides the actual location details of the File -> Client uses this data to fetch the file from the Data Node for reading.

How does Namenode know if Datanode is alive?

- Every Datanode sends a heartbeat to Namenode every 3 seconds indicating that it's alive.
- If the Namenode does not receive a heartbeat for 30 seconds, then Namenode assumes Datanode is dead.

What if a Namenode fails?

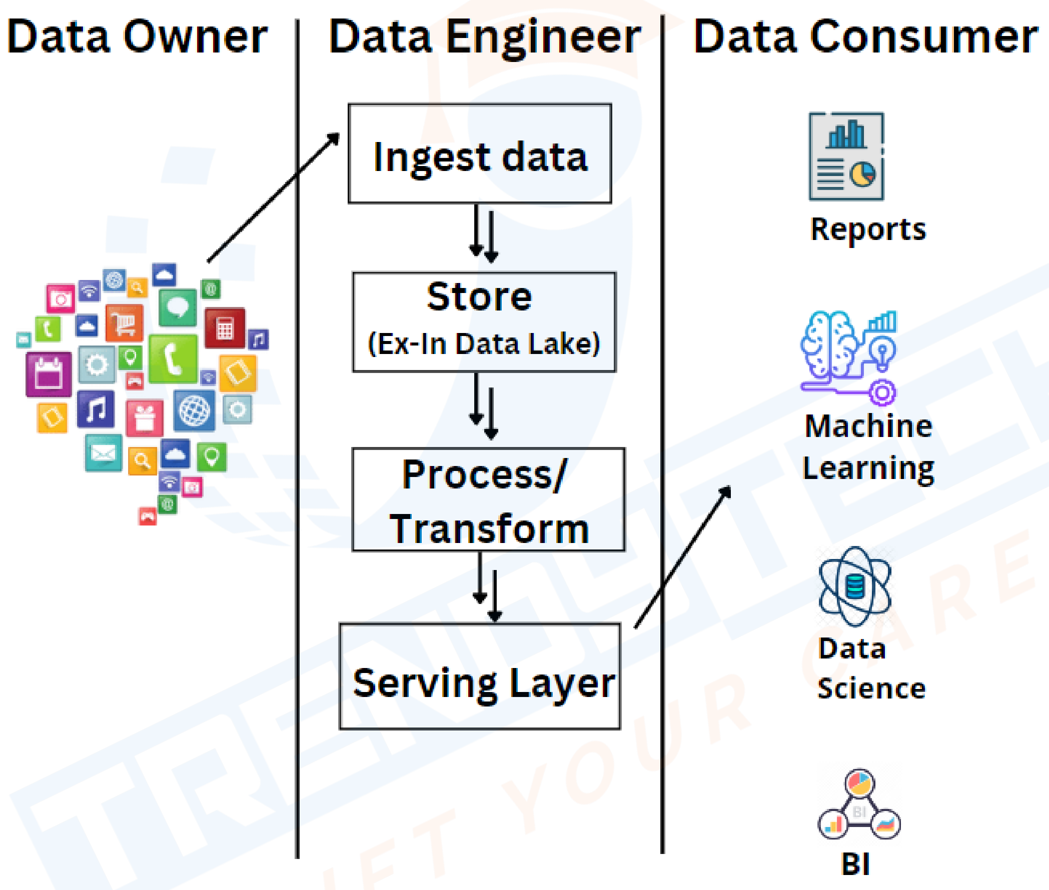
- Namenode is a good quality hardware and chances of going down is very less.
- But if it goes down, we have a secondary Name node to tackle this situation.

Role of Data Engineers

Understanding the need of Data Engineers

-Data engineers act as a glue between Data Owners and the Data Consumers. They are responsible for collecting the Data from multiple sources and transforming it into a format that can be served to the Data Consumers.

-Data engineers are responsible for providing the data in the form the Data Consumer requests for.



Traditional ways of processing Data and its Challenges

-Earlier Data was ingested and some transformations were applied using ETL tools like Informatica and Talend before it was loaded into Data Warehouses where further processing was carried out.

-Challenges of such ETL systems is that Data warehouses were very costly as they required specialised servers.