UNIT-1

Syllabus

1.Definition and Characteristics of Big Data

- **Volume, Velocity, Variety, Veracity, Value**

2.Types of Analytics

- **Descriptive, Diagnostic, Predictive, Prescriptive**

3.Big Data Challenges and Opportunities

4.Hadoop Ecosystem Overview

- **HDFS, MapReduce, YARN**

5.NoSQL Databases

- **Key-Value, Columnar, Document, Graph Models**

6.Data Lake vs Data Warehouse

INTRODUCTION TO BIG DATA

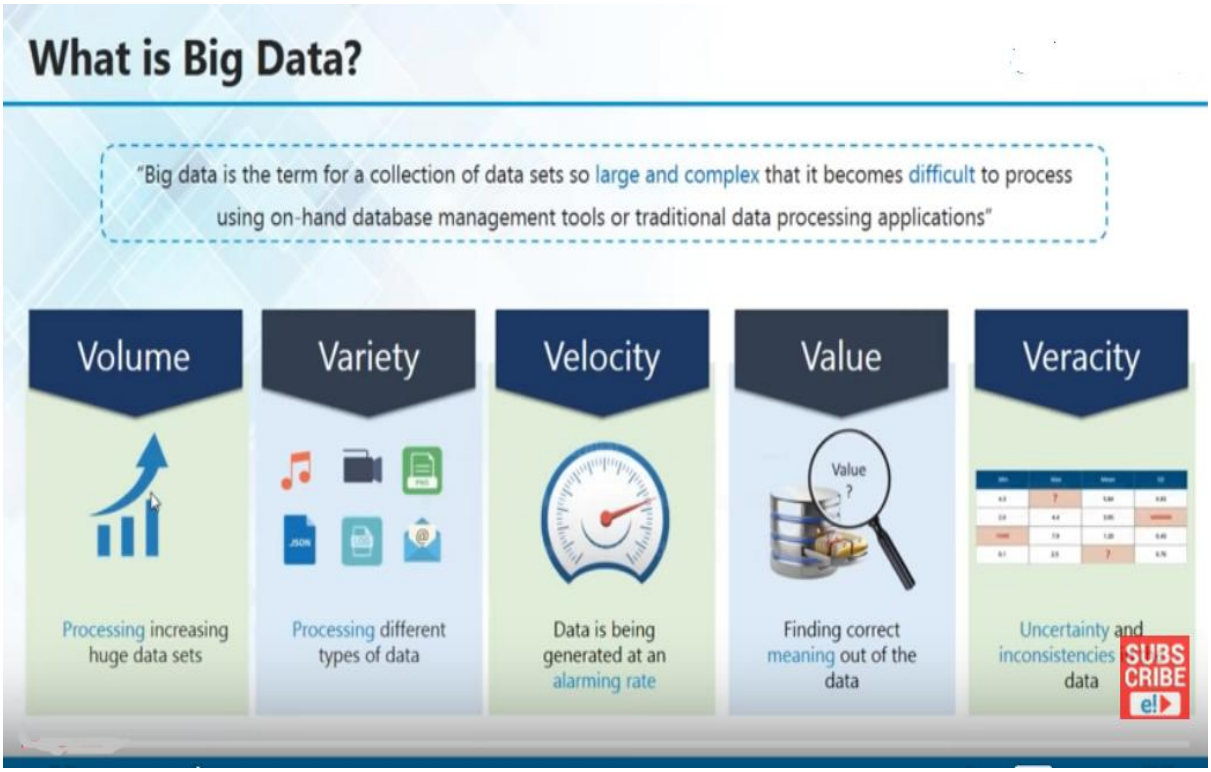
What is Data?

- The quantities, characters, or symbols on which operations are performed by a computer.

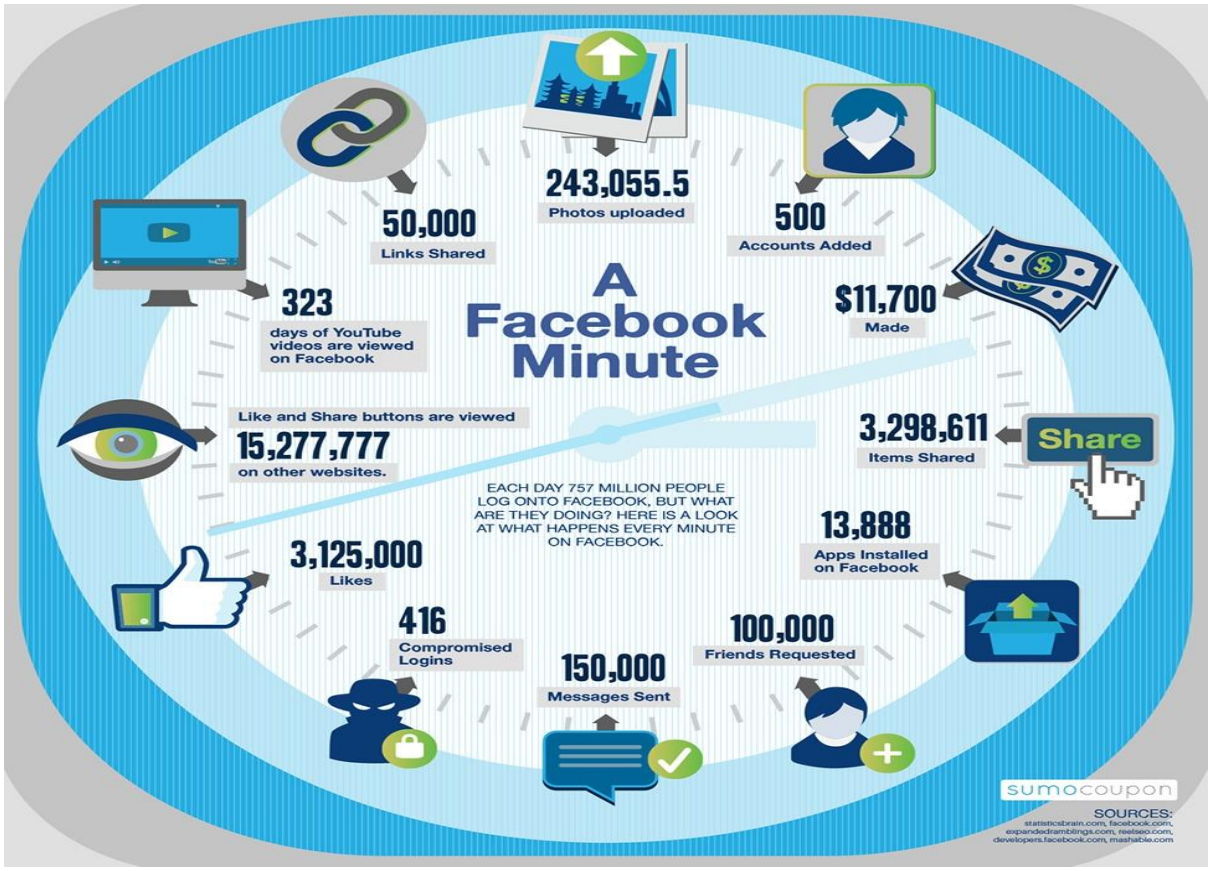
What is Big Data?

- Big Data is a collection of data that is huge in volume, yet growing exponentially with time

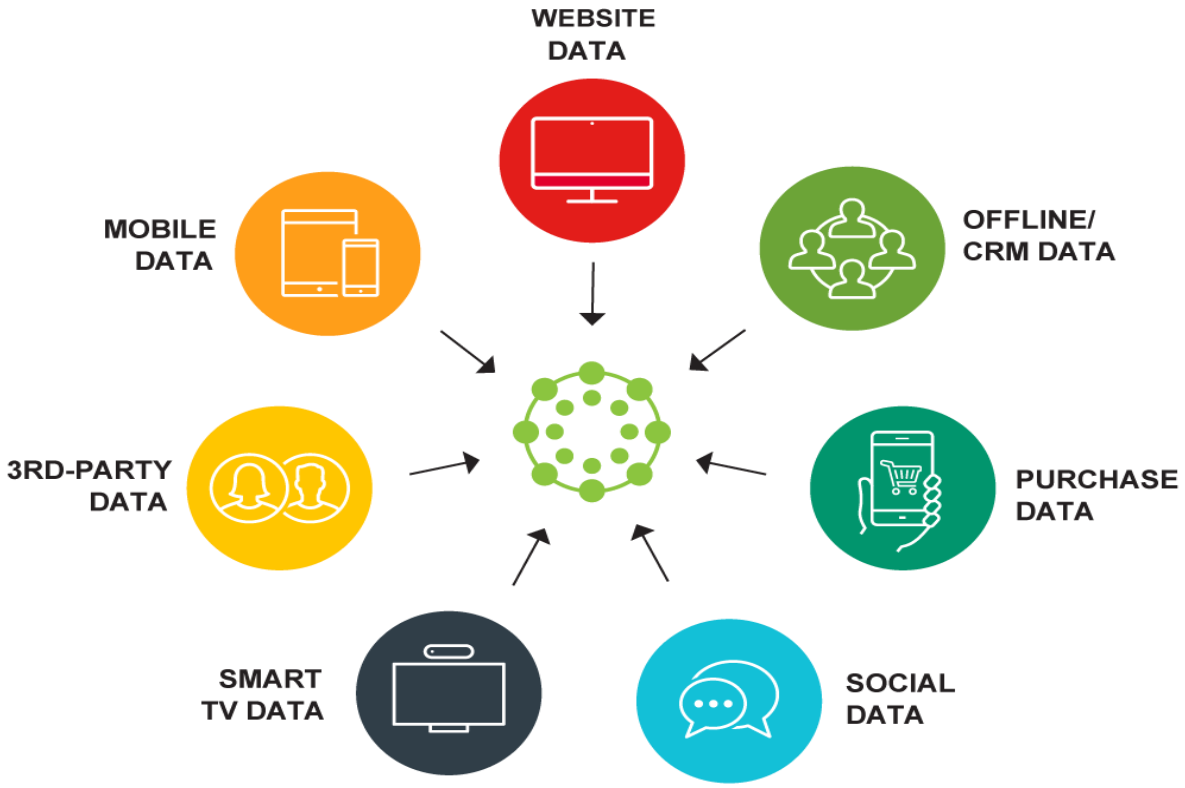
OR



What makes data as "Big" Data?

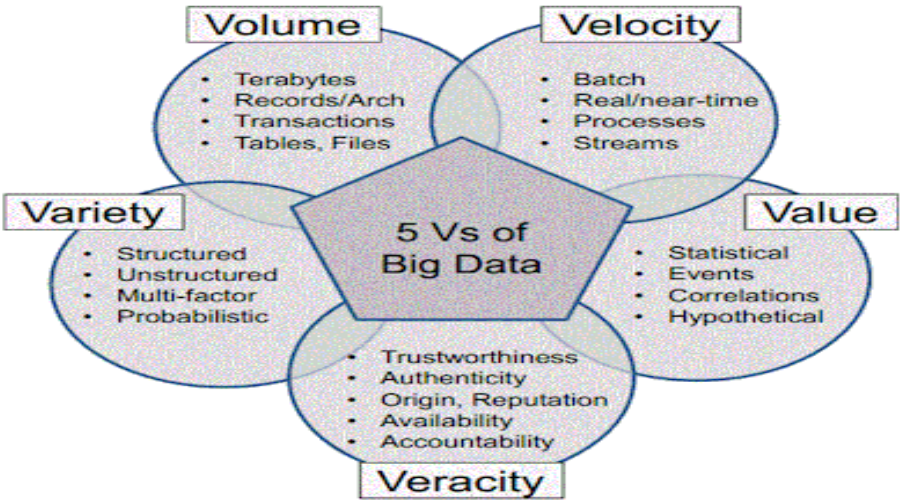


Big Data Sources



Characteristics of BigData

1) **Volume**
2) **Velocity**
3) **Variety**
4) **Veracity**
5) **Value**



1)Volume

- ❑ refers to the vast amounts of data generated every second.
- ❑ we are not talking about Terabytes but zetabytes.
- ❑ If we take all the data generated in the world between the beginning of time and 2008, the same amount of data will soon be generated every minute.
- ❑ This makes most data sets too large to store and analyze using Traditional database technology.
- ❑ **Example:** Amazon handles 15 million customer click stream user data per day to recommend products.

Typical data sources that are responsible for generating high data volumes can include:

- online transactions, such as point-of-sale and banking
- sensors, such as GPS sensors, RFIDs, smart meters and telematics

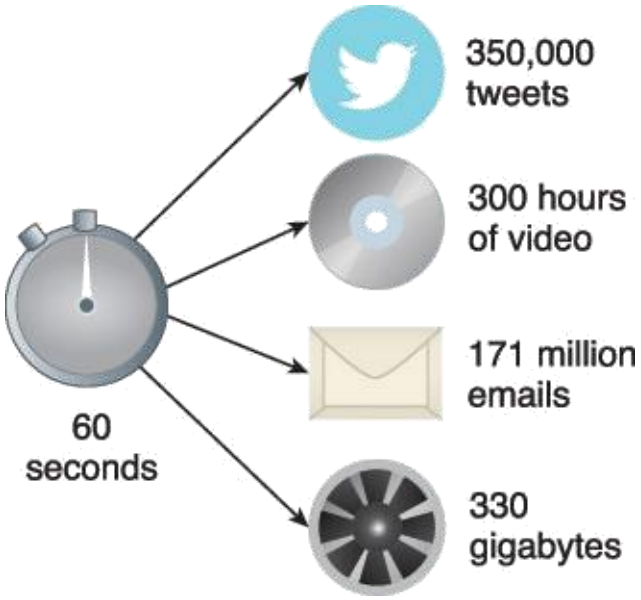
social media, such as Facebook and Twitter

2)Velocity

- ❑ Refers to the speed at which new data is generated and the speed at which data moves around. Just think of social media messages going viral in seconds

.

Example: 72 hours of video are uploaded to YouTube every minute this is the velocity. Extremely high velocity of data is another major characteristic of big data.



3)Variety

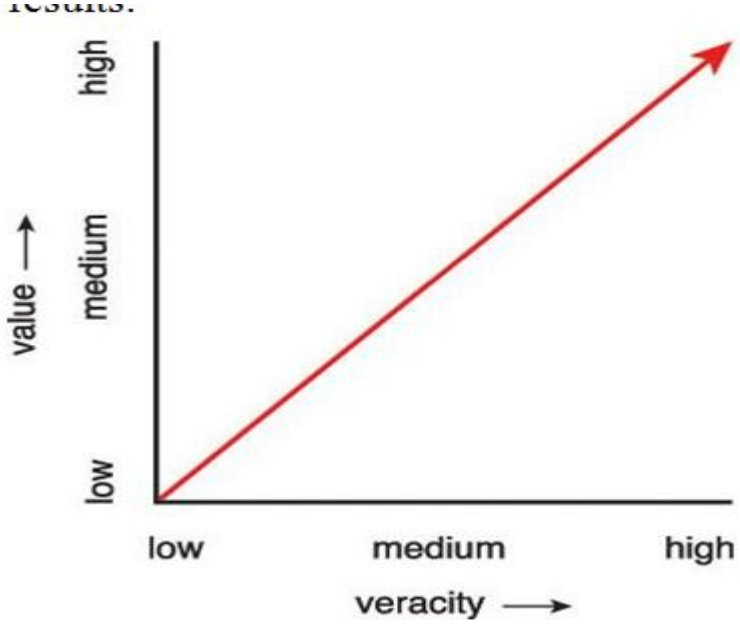
- Refers to the different types of data generated from different sources.
- It can be structured, semi-structured or unstructured.eg: data generated by sensors, electronic devices, social media etc..
- In the past we only focused on structured data that neatly fitted into tables or relational databases, such as financial data.
- It can be structured, semi-structured or unstructured.

4)Veracity

- Data should be accurate and reliable. Veracity refers to the quality or fidelity of data

5)Value

- Value is defined as the usefulness of data for an enterprise.
- The value characteristic is intuitively related to the veracity characteristic in that the higher the data fidelity, the more value it holds for the business.



Types of Data

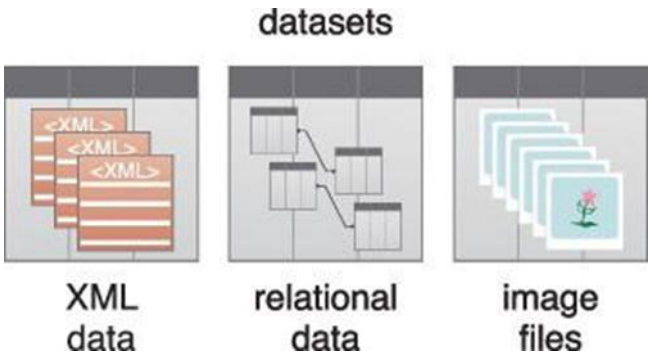
1) **Structured Data**
2) **Semi-structured Data**
3) **Unstructured Data**

1) Structured Data: data that has both Data Model and schema, eg : oracle Data
2) Semi-Structured: data that has only schema, eg : XML data
3) Unstructured: Data that doesn't have Data Model and schema, eg: face book data

Types of Analytics

What is Dataset

- ❑ Collection or group of related data are generally refer to as Dataset.
- ❑ Each group(datum) shares the same set of attributes as others in the same dataset.
- ❑ Some of the examples are:
  - ✓ Tweets stored in a flat file
  - ✓ a collection of image files in a directory.
  - ✓ an extract of rows from a database table stored in a CSV formatted file
  - ✓ historical weather observations that are stored as XML files



Datasets can be found in many different formats.

What is Data Analysis

- ✓ **Data analysis is the process of examining data to find facts, relationships, patterns, insights and/or trends.**
- ✓ **The overall goal of data analysis is to support better decision-making.**
- ✓ A simple data analysis example is the analysis of ice cream sales data in order to determine how the number of ice cream cones sold is related to the daily temperature.
- ✓ The results of such an analysis would support decisions related to how much ice cream a store should order in relation to weather forecast information.
- ✓ Following figure shows the symbol used to represent data analysis.

What is Data Analytics

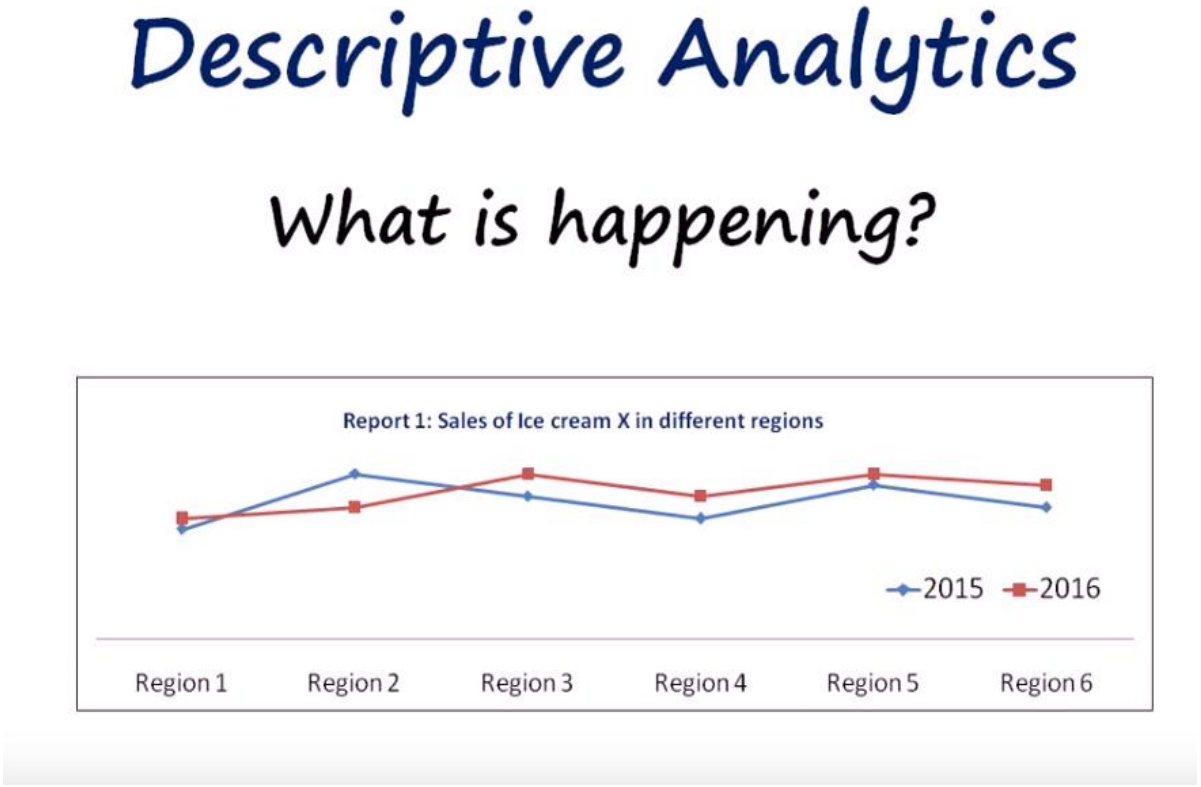
- ✓ Data analytics is a broader term that encompasses data analysis.
- ✓ Data analytics is a discipline that includes the management of the complete data lifecycle, which encompasses collecting, cleansing, organizing, storing, analyzing and governing data.
- ✓ Following figure shows the symbol used to represent analytics.

There are four general categories of analytics that are distinguished by the results they produce:

1. Descriptive Analytics
2. Diagnostic Analytics
3. Predictive Analytics
4. Prescriptive Analytics

1.Descriptive Analytics

Descriptive Analytics gives you information on what is happening. Let us see a particular Example.

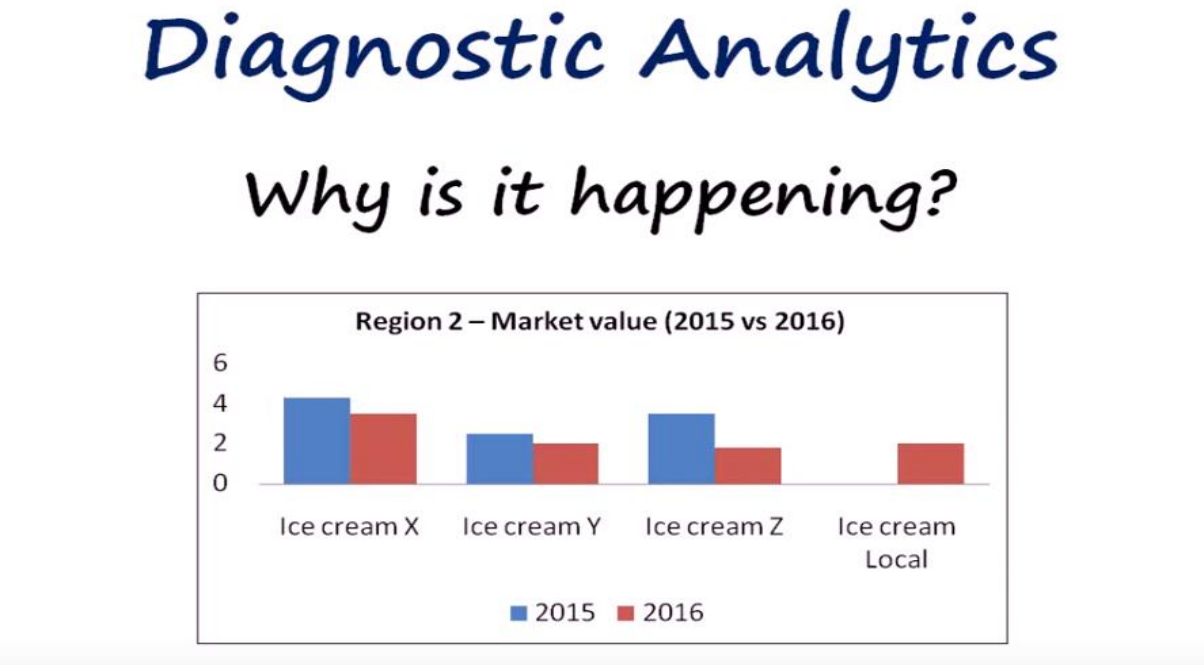


- ✓ This report provides information about the sales of ice cream **X** across different regions and compares these sales between the years **2015** and **2016**.
- ✓ This is an example of **descriptive analytics**, which focuses on summarizing historical data to explain **what has happened**.
- ✓ From the chart, we can infer that **sales increased in all regions except Region 2**, where a noticeable drop occurred from 2015 to 2016.

- ✓ This level of insight helps us understand past performance: now that we know sales have declined in Region 2, further analysis would be required to explore **why** this happened and how it can be addressed.

2.Diagnostic Analytics

- ✓ Information on why is it happening . so our job is to deep dive into region 2 and see what exactly is going wrong for us so this is a typical report for the market share of region 2



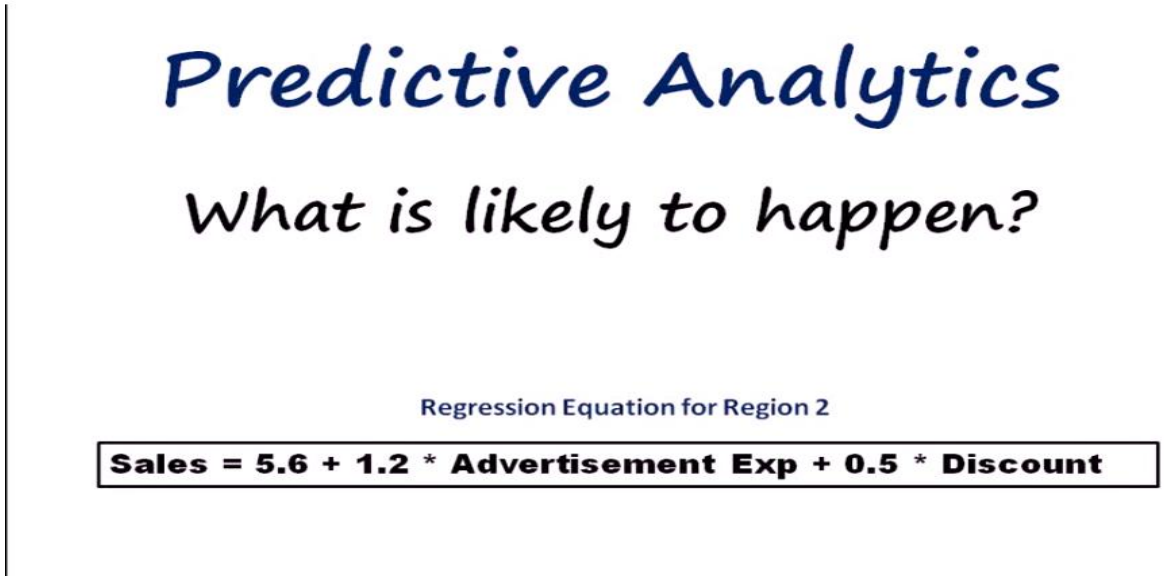
- ✓ From the graph , it is clear that the sales of ice cream brands x, y and z is dropped from 2015 to 2016 and it is because the local brand ice cream has come up in 2016.
- ✓ In the first step we found **what is happening** and here **we found why it is happening**.
- ✓ here we have diagnosed the reason of sales drop in region 2

3.Predictive Analytics

- ✓ Once we identified the reason of happenings , then we can take measures to improve things i.e what should be done to make things happen correct.
- ✓ In our example, first we identified region 2 has less sales in 2016 next, we found the reason of why region 2 sales got down and now it

is the time to find the solution to increase the sales and that is called predictive data analytics

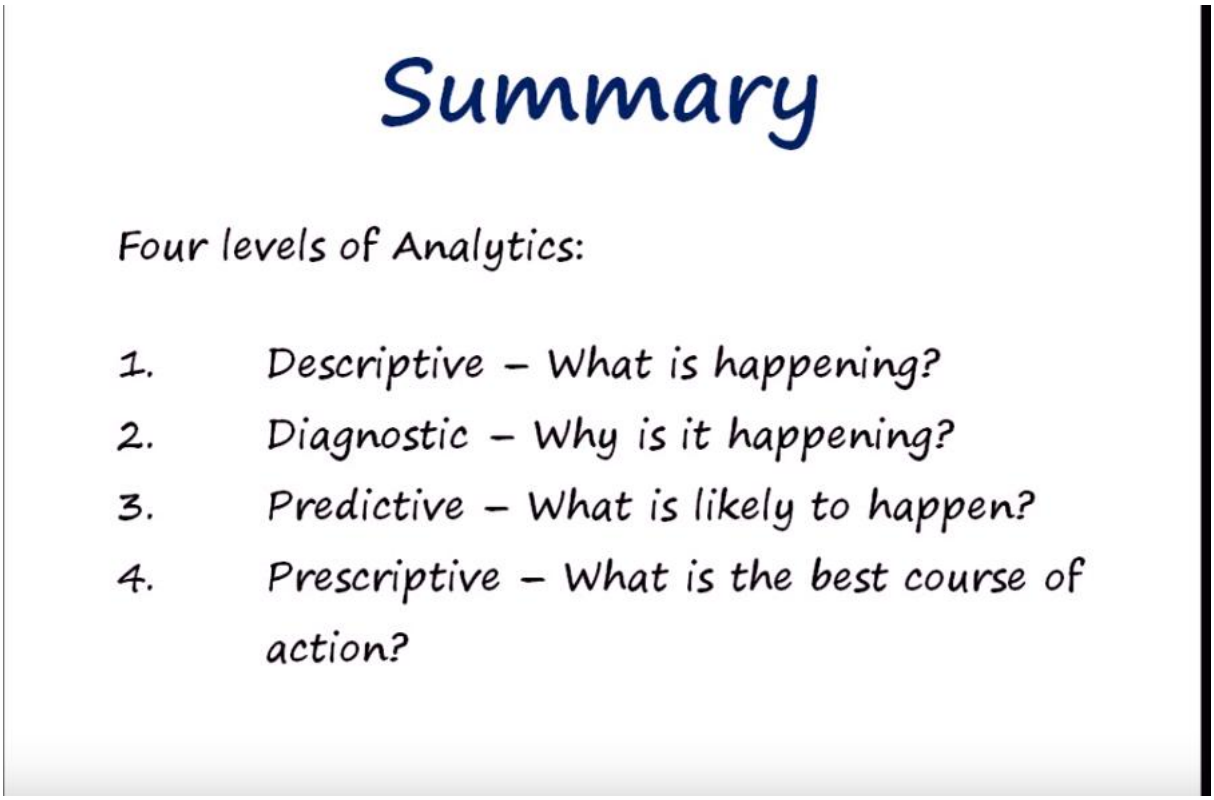
- ✓ For this example, we can build a **regression model** specifically for Region 2 to analyze the various factors that may have influenced its sales.
- ✓ By including variables such as **advertising expenditure**, **discounts**, **seasonality**, or other regional factors, the model can help us understand how each factor affects sales.
- ✓ Once the model is built, we can modify these inputs—for instance, increasing advertisement spending or adjusting discount rates—to **predict the impact of these changes on future sales**.
- ✓ This is where **predictive analytics** becomes useful: it helps us estimate what is **likely to happen** in the future based on historical patterns.
- ✓ At this stage, since we already know what happened and why it happened, our next task is to determine **what will happen next**, which is the core objective of predictive analytics.



4.Precriptive Analytics

- ✓ Our next step is to move to next topic i. e Prescriptive analysis there you suggest the best course of action to solve the problem.
- ✓ for this particular example we can give solutions like increase the advertising expense by 10% or give 5 % discount for two months
- ✓ if we are able to find out using the regression equation that these factors like advertising expense or discount has an impact on sales

- ✓ so these are the four different levels of business analytics .
- ✓ Now lets summarize what we have just seen



Big Challenges with Big Data

Challenges of Big Data

The challenges of Big Data are the real implementation hurdles that require immediate attention and need to be addressed to avoid the technology's failure. If not properly handled, these challenges can lead to inefficient data management, poor decision-making, and missed opportunities. Let's discuss some of the most critical challenges related to Big Data.

Data Volume: Managing and Storing Massive Amounts of Data

- **Challenge:** The most apparent challenge with Big Data is the sheer volume of data being generated. Organizations are now dealing with petabytes or even exabytes of data, making traditional storage solutions inadequate. This vast amount of data requires advanced storage infrastructure, which can be costly and complex to maintain.
- **Solution:** Adopting scalable cloud storage solutions, such as ***Amazon S3***, ***Google Cloud Storage***, ***or Microsoft Azure***, can help manage large volumes of data. These platforms offer flexible storage options that can grow with your data needs.

Additionally, implementing data compression and deduplication techniques can reduce storage costs and optimize the use of available storage space.

Data Variety: Handling Diverse Data Types

- **Challenge:** Big Data encompasses a wide variety of data types, including structured data (e.g., databases), semi-structured data (e.g., XML, JSON), and unstructured data (e.g., text, images, videos). The diversity of data types can make it difficult to integrate, analyze, and extract meaningful insights.
- **Solution:** To address the challenge of data variety, organizations can employ data integration platforms and tools like Apache Nifi, Talend, or Informatica. These tools help in consolidating disparate data sources into a unified data model. Moreover, adopting schema-on-read approaches, as opposed to traditional schema-on-write, allows for more flexibility in handling diverse data types.

Data Velocity: Processing Data in Real-Time

- **Challenge:** The speed at which data is generated and needs to be processed is another significant challenge. For instance, IoT devices, social media platforms, and financial markets produce data streams that require real-time or near-real-time processing. Delays in processing can lead to missed opportunities and inefficiencies.
- **Solution:** To handle high-velocity data, organizations can implement real-time data processing frameworks such as Apache Kafka, Apache Flink, or Apache Storm. These frameworks are designed to handle high-throughput, low-latency data processing, enabling businesses to react to events as they happen. Additionally, leveraging edge computing can help process data closer to its source, reducing latency and improving real-time decision-making.

Data Veracity: Ensuring Data Quality and Accuracy

- **Challenge:** With Big Data, ensuring the quality, accuracy, and reliability of data—referred to as data veracity—becomes increasingly difficult. Inaccurate or low-quality data can lead to misleading insights and poor decision-making. Data veracity issues can arise from various sources, including data entry errors, inconsistencies, and incomplete data.
- **Solution:** Implementing robust data governance frameworks is crucial for maintaining data veracity. This includes establishing data quality standards, performing regular data audits, and employing data cleansing techniques. Tools like **Trifacta, Talend Data Quality, and Apache Griffin** can help automate and streamline data quality management processes.

Data Security and Privacy: Protecting Sensitive Information

- **Challenge:** As organizations collect and store more data, they face increasing risks related to data security and privacy. High-profile data breaches and growing

concerns over data privacy regulations, such as **GDPR** and **CCPA**, highlight the importance of safeguarding sensitive information.

- **Solution:** To mitigate security and privacy risks, organizations must adopt comprehensive data protection strategies. This includes implementing encryption, access controls, and regular security audits. Additionally, organizations should stay informed about evolving data privacy regulations and ensure compliance by adopting privacy-by-design principles in their data management processes.

Data Integration: Combining Data from Multiple Sources

- **Challenge:** Integrating data from various sources, especially when dealing with legacy systems, can be a daunting task. Data silos, where data is stored in separate systems without easy access, further complicate the integration process, leading to inefficiencies and incomplete analysis.

- **Solution:** Data integration platforms like Apache Camel, **MuleSoft**, and **IBM DataStage** can help streamline the process of integrating data from multiple sources. Adopting a microservices architecture can also facilitate easier data integration by breaking down monolithic applications into smaller, more manageable services that can be integrated more easily.

Data Analytics: Extracting Valuable Insights

- **Challenge:** The ultimate goal of Big Data is to derive actionable insights, but the complexity of analyzing large, diverse datasets can be overwhelming. Traditional analytical tools may struggle to scale, and the lack of skilled data scientists can further hinder the ability to extract meaningful insights.

- **Solution:** Organizations should invest in advanced analytics platforms like Apache Spark, Hadoop, or Google BigQuery, which are designed to handle large-scale data processing and analysis. Additionally, fostering a culture of data literacy and providing training for employees can help bridge the skills gap and empower teams to effectively analyze Big Data.

Data Governance: Establishing Policies and Standards

- **Challenge:** As data becomes a critical asset, establishing effective data governance becomes essential. However, many organizations struggle with creating and enforcing policies and standards for data management, leading to issues with data consistency, quality, and compliance.

- **Solution:** Implementing a formal data governance framework is key to overcoming this challenge. This framework should define roles and responsibilities, establish data stewardship programs, and enforce data management policies. Tools like **Collibra, Alation, and Informatica's** data governance suite can assist in creating and maintaining a robust data governance strategy.

Big Data Opportunities

- **Better Decision-Making:**
    - Enables organizations to base decisions on real-time data rather than intuition.
    - Improves strategic planning and reduces uncertainty in business operations.
- **Customer Insights:**
    - Analyzes buying patterns, preferences, and feedback.
    - Helps companies segment customers and target them more effectively.
- **Personalization:**
    - Recommends personalized products, content, and services (e.g., Netflix, Amazon).
    - Enhances user satisfaction by tailoring experiences to individual needs.
- **Predictive Analytics:**
    - Predicts demand, sales trends, equipment failures, and market movements.
    - Supports proactive decision-making instead of reactive responses.
- **Fraud Detection:**
    - Identifies abnormal transactions, login patterns, and suspicious activities.
    - Helps financial institutions and e-commerce platforms prevent fraud.
- **Improved Healthcare:**
    - Enhances early disease detection using medical records and patient data.
    - Supports personalized treatment plans and remote monitoring systems.

NoSQL Databases

NoSQL Databases are designed to store and manage **large volumes of unstructured or semi-structured data**. Unlike traditional relational databases, they do not require a fixed schema and can scale easily.

Types of NoSQL Databases:

1. **Key-Value Stores**
    - Data is stored as **key-value pairs**.
    - Very fast for lookups using a key.
    - Example: Redis, DynamoDB

2. **Columnar (Column-Family) Stores**
    - Data is stored in **columns instead of rows**, optimized for analytical queries.
    - Example: Cassandra, HBase
3. **Document Stores**
    - Data is stored as **documents** (like JSON, XML).
    - Flexible schema, good for hierarchical data.
    - Example: MongoDB, CouchDB
4. **Graph Databases**
    - Data is stored as **nodes and edges** to represent relationships.
    - Ideal for social networks, recommendation engines.
    - Example: Neo4j, Amazon Neptune

Data Lake and Data Warehouse

Data Lake

- ✓ A **Data Lake** is a place where a company can store **all kinds of data**—whether it is structured, unstructured, or in any raw form—at a **low cost**.
- ✓ It is like an updated version of the old "landing zone" used in ETL and Data Warehouse systems. The main idea is that, unlike earlier times when companies had to choose only important data and store it in a structured way, now technology allows companies to **keep every piece of data** they create or purchase.
- ✓ The data does not need a fixed format or structure when stored. Later, this raw data can be analyzed to find useful insights.

Data Warehouse

- ✓ A **Data Warehouse** is basically a large database stored on the cloud or on a company's central server.
- ✓ It collects data from many different and unrelated sources to support analysis and help in business decision-making.
- ✓ A data warehouse is described as **subject-oriented**, **integrated**, **time-variant**, and **non-volatile**, which means it is organized by topics, combines data from multiple sources, keeps historical data, and does not change frequently.
- ✓ Its main purpose is to provide meaningful insights and support managers in making better decisions

Difference between Data Lake and Data Warehouse

Feature	Data Lake	Data Warehouse
Data Type	Stores **all types of data**: structured, semi-structured, unstructured	Stores **structured and processed data** only

Feature	Data Lake	Data Warehouse
Schema	Schema-on-read (apply structure **when reading/using the data**)	Schema-on-write (structure **applied before storing**)
Cost	Low-cost storage, can handle huge volumes	More expensive, designed for optimized queries
Purpose	Exploration, advanced analytics, machine learning, and storing raw data	Reporting, business intelligence, and supporting decision-making
Flexibility	Very flexible; stores raw data without predefined rules	Less flexible; data must be cleaned and structured first
Speed of Ingest	Fast; can ingest large amounts of raw data quickly	Slower; requires ETL process to structure data before loading
Users	Data scientists, analysts, engineers for experimentation	Business analysts, managers for reporting and insights
Storage Locatio	Often on low-cost cloud storage (e.g., S3, Azure Blob)	On-premises servers or cloud-based data warehouse solutions

TEXT BOOKS:

1. Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid Dastjerdi – Wiley
2. Learning Spark: Lightning-Fast Big Data Analysis by Jules S. Damji et al. – O'Reilly Data Science and Big Data Analytics by EMC Education Services – Wiley

REFERENCE BOOKS:

1. Designing Data-Intensive Applications by Martin Kleppmann – O'Reilly
2. Machine Learning with Spark by Rajdeep Dua, Tathagata Das – Packt Publishing
3. Streaming Systems by Tyler Akidau – O‘Reilly Media.
4. 4. Artificial Intelligence for Big Data by Anand Deshpande

REFERENCE WEBSITE:

1. https://www.coursera.org/specializations/big-data – Coursera Big Data Specialization
2. https://spark.apache.org/docs/latest/ – Apache Spark Documentation
3. https://www.edx.org/course/big-data-analysis-with-python – edX

3. https://www.udacity.com/course/ai-for-business-leaders--nd088 – Udacity AI for Business
4. https://www.kaggle.com/learn/intro-to-machine-learning – Kaggle ML Tutorials
5. https://data-flair.training/blogs/apache-spark-tutorial/ – Spark Tutorials

Introduction to Apache Hadoop

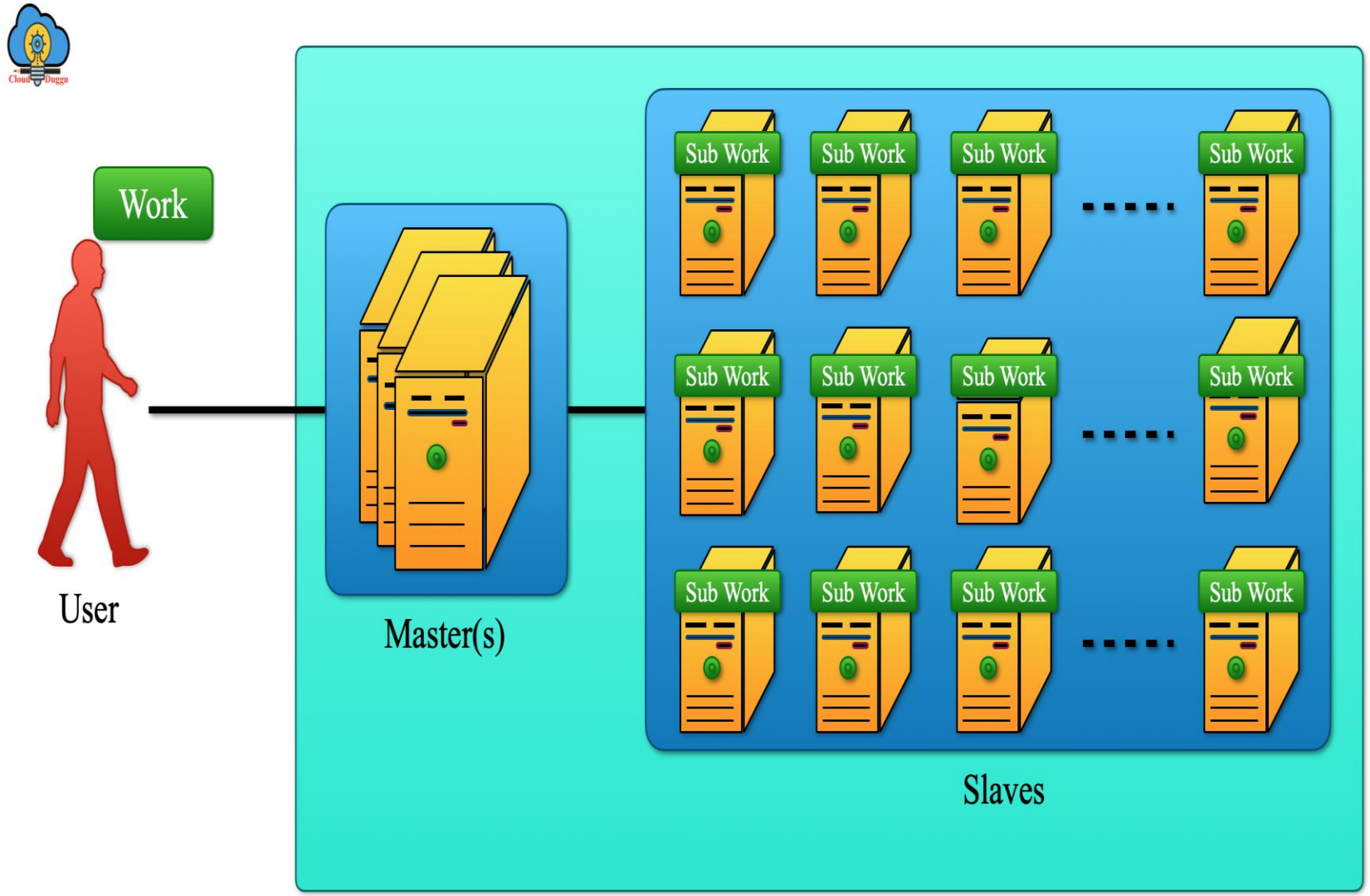
- The Core of Apache Hadoop consists of
  - Storage part
    - ***Hadoop Distributed File System (HDFS),***
  - Processing part
    - ***MapReduce programming model.***

Referred by Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid Dastjerdi – Wiley

Hadoop Technology

- Hadoop is open source tool from the Apache Software Foundation.
- Hadoop codes are written by Yahoo, IBM, Cloudera etc…
- Hadoop provides parallel processing through different commudity h/w simultaneously.

Hadoop Cluster



Hadoop Cluster

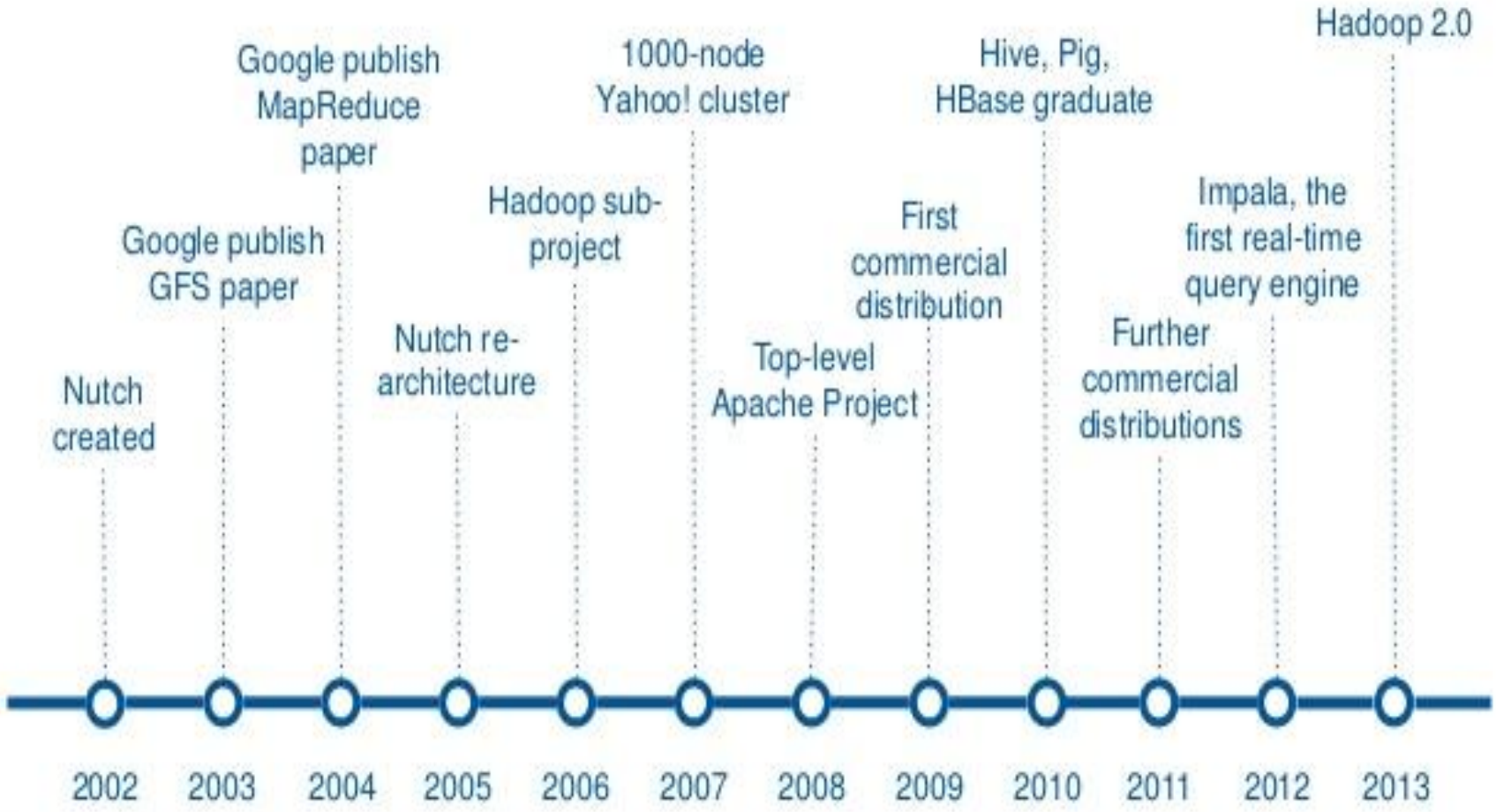
Referred by Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid

What is Hadoop?

- Framework for Large-Scale Data Processing
- Inspired by Google's Architecture:
  - **Google File System (GFS)**
  - **MapReduce**
- Open-source Apache project
  - **Nutch search engine project**
  - **Apache Incubator**
- Written in Java and shell scripts

Who use Hadoop?

A Brief History of Hadoop



Referred by Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid

Brief History of Hadoop

In 2002, **Doug Cutting and Mike Cafarella** started to work on Apache Nutch project.

Nutch is a open source web crawler software project

After a lot of Research on Nutch, they concluded that this project is very expensive

0.5 million dollars for hardware , Running cost/month $30,000 approximately

In 2003, They came across Google's white paper on GFS(Google File System)

This paper described the architecture of Google's distributed file system, for storing the large data sets.

But it was just the half solution to their problem

In 2004, Google published one more paper on the Mapreduce Technique

This paper was another half solution and now they had complete solution for their Nutch Project

Referred by Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid

Brief History of Hadoop

So, they started implementing Google's Techniques (GFS &Mapreduce) as open-source in the Apache Nutch Project

In 2005, Cutting found that nutch is limited to only 20-to-40 node clusters

He soon realized 2 problems

Nutch wouldn't achieve its potential until it can reliably on the larger clusters

And that was looking impossible with just two people

Doug Cutting joined Yahoo along with Nutch Project

He wanted to provide the world with an open-source,reliable,scable,computing framework, with the help of yahoo

So at yahoo first, he separates the distributed computing parts from Nutch and formed a new project "Hadoop"

In 2007, Yahoo successfully tested Hadoop on a 1000node cluster and Started using it

In 2008, In January, Yahoo released Hadoop as an open source project to AFS(Apache Software Foundation)

Referred by Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid

Brief History of Hadoop

In July, Apache Software Foundation successfully tested a 4000 node cluster with Hadoop

In 2009, Hadoop was successfully tested to sort a PB (Petabyte) of data in less than 17 hours

Doug Cutting left the Yahoo and joined Cloudera to fulfill the challenge of spreading Hadoop to other industries

In December 2011

Apache Software Foundation released Apache Hadoop version 1.0

In August 2013

Version 2.0.6 was available

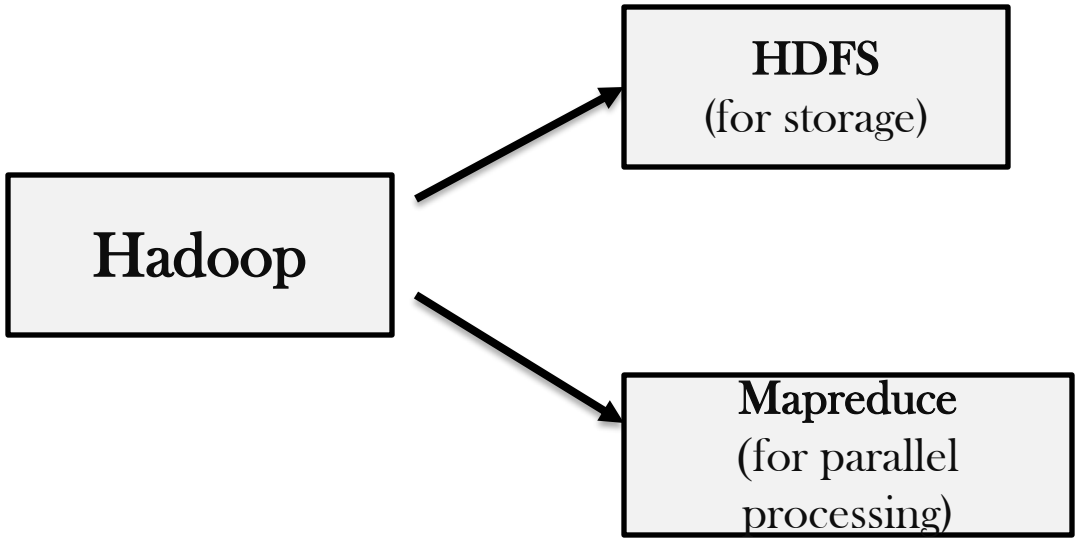
In December 2017

Apache Hadoop version 3.0 released which is currently we are having

Referred by Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid

Understanding Hadoop Features

- Doug cutting" and "mike cafarella" introduced Hadoop in 2006.
- Hadoop is a open source framework for "storing and processing" huge dataset.
- Hadoop is a open source tool that allows to store and process big data in a distributed environment across cluster of computer using simple programming model for processing.
- Core components of Hadoop are: "HDFS" and "Mapreduce"



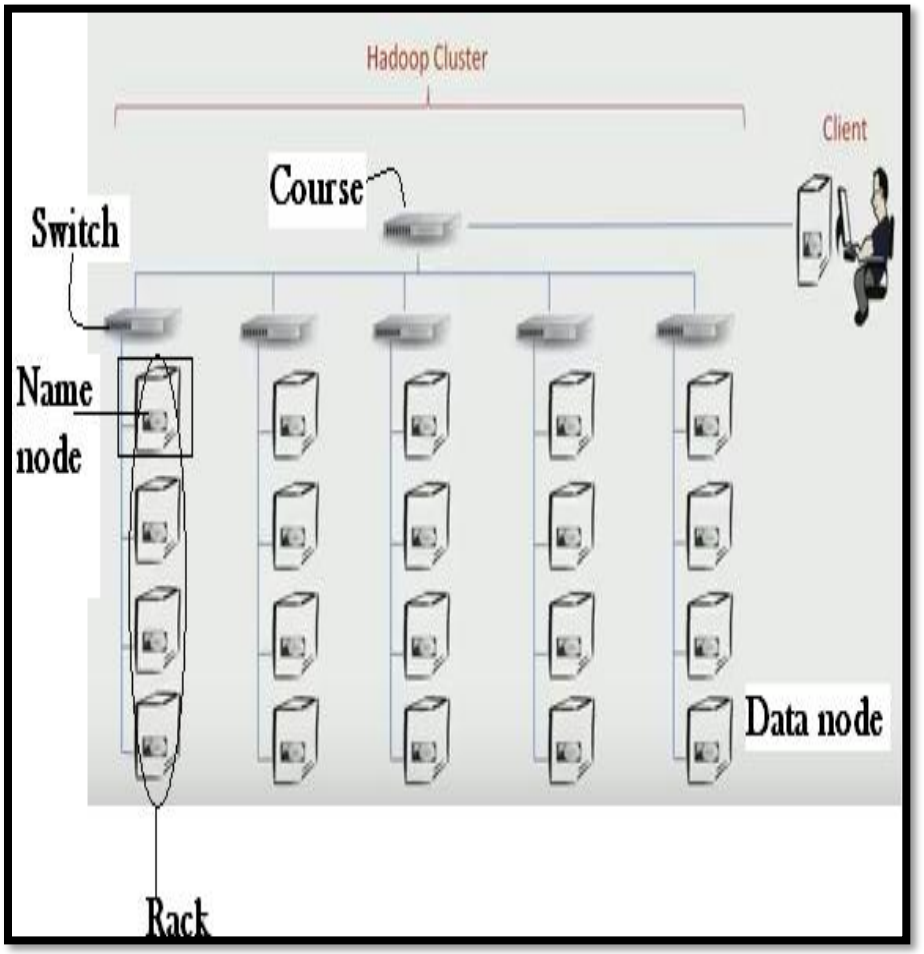
- The name “Hadoop” is not an acronym, it is madeup name. The projects creator “Doug cutting” explains how the name came about.
- The name, my kid gave to the stuffed yellow elephant. Short, relatively easy to pronounce, meaningless and not used else anywhere. Those are my naming criteria. Kids are good at generating search names

6.1 UNDERSTANDING HDFS

DISTRIBUTED STORAGE

Data is stored in a Distributed Manner

- Instead of relying on a single large machine , HDFS uses a network of several smaller machine and add them to a cluster this approach is called Horizontal scaling.
- So, we use a software that combines the storage capacity of the entire network into a single large system i.e network based file system and that is HDFS.



Hadoop client will send a request to name node that it want to create a file along **with target directory name and file name**

By receiving the request the **name node will perform various checks like is the file already exist and client has right permission to create a file.**

Name node does all this because it maintains an image of entire HDFS namespace into memory we call **it file system image.**

If all test pass the name node will **create an entry for new file and return success to client.**

Now, empty file is created.

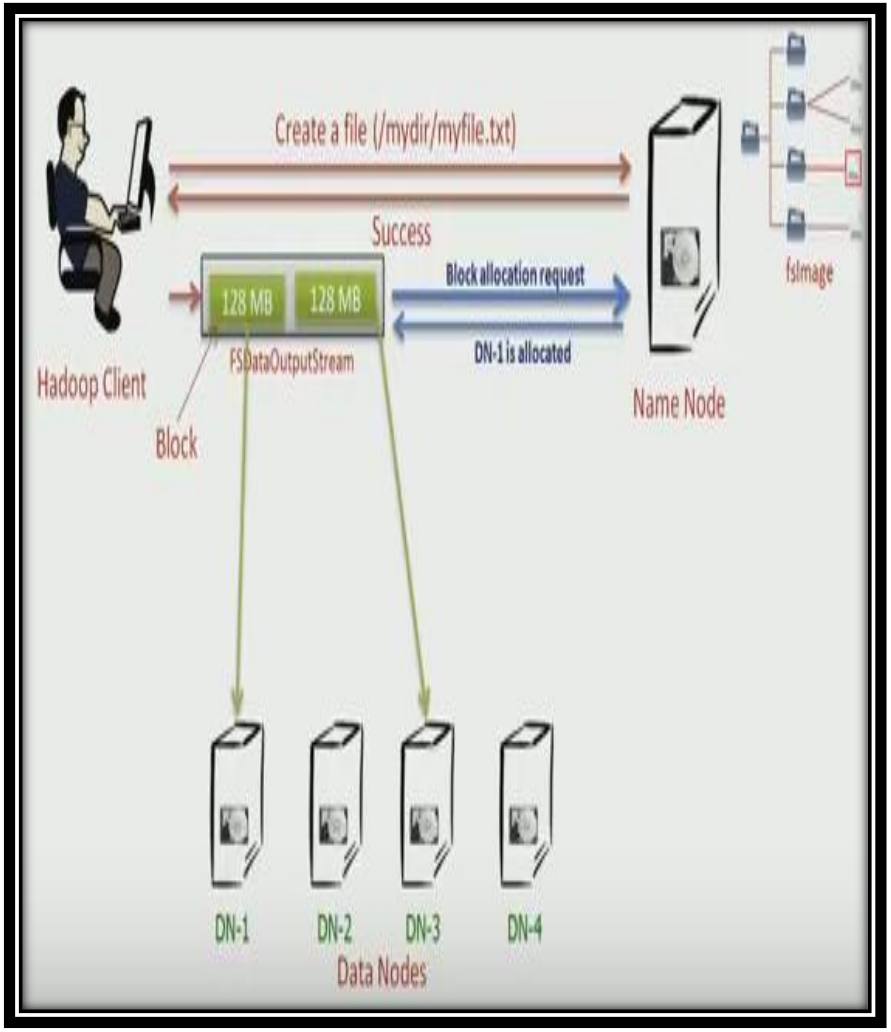
The client start writing **fsdata output stream.**

This stream internally buffers data until you accumulate the reasonable amount of data with default size **128 MB as a block**.

Each block send **block allocation request** to name node asking for location for storing that block name node will allocate a data node and send back data node name to the streamer.

Now streamer will send the **data to the data node.**

If file is large it creates a new block allocation and does the same work.



SCALABILITY

- As HDFS stores the large size data over multiple nodes, so when the requirement of data storing is increased or decreased the number of nodes can be scaled up or scaled down in a cluster.
- The vertical and Horizontal scalability is the 2 different mechanisms available to provide scalability in the cluster.
- Vertical scalability means adding the resource like Disk space, RAM on the existing node of the cluster.
- On another hand in Horizontal scaling, we increase the number of nodes in the cluster and it is more preferable since we can have hundreds of nodes in a cluster.

SCALABILITY

Scalability

- The primary benefit of Hadoop is its **Scalability.**
- One can easily scale the cluster by adding more nodes.
- There are two types of Scalability in Hadoop: **Vertical and Horizontal**

Vertical scalability

- It is also referred as "**scale up".** In vertical scaling, you can increase the **hardware capacity of the individual machine.**
- In other words, you can add more RAM or CPU to your existing system to make it more robust and powerful.

Horizontal scalability

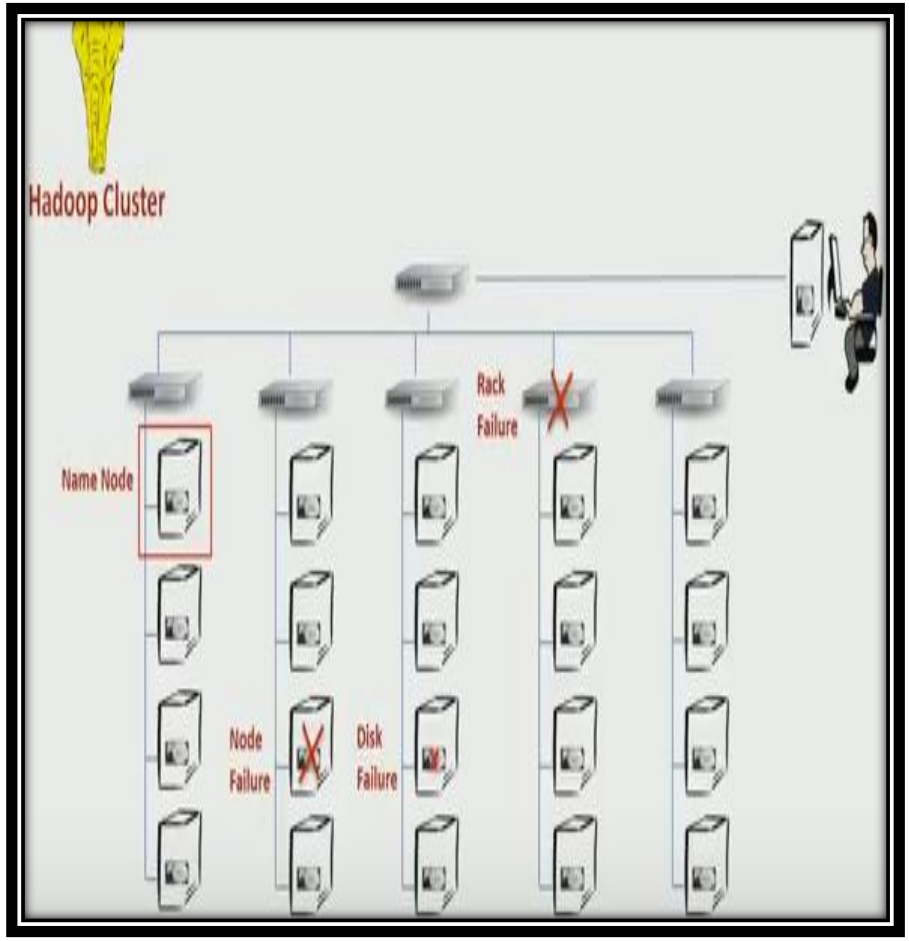
- It is also referred as "**scale out"** is basically the addition of more machines or setting up the cluster.
- In horizontal scaling instead of increasing hardware capacity of individual machine **you add more nodes to existing cluster**

FAULT TORELANCE

- HDFS is **highly fault-tolerant and reliable**.
- HDFS creates **replicas of file blocks** depending on the replication factor and stores them on different machines.
- If any of the machines containing data blocks fail, other Data Nodes containing the replicas of that data blocks are available.
- ***Thus ensuring no loss of data and makes the system reliable even in unfavorable conditions.***

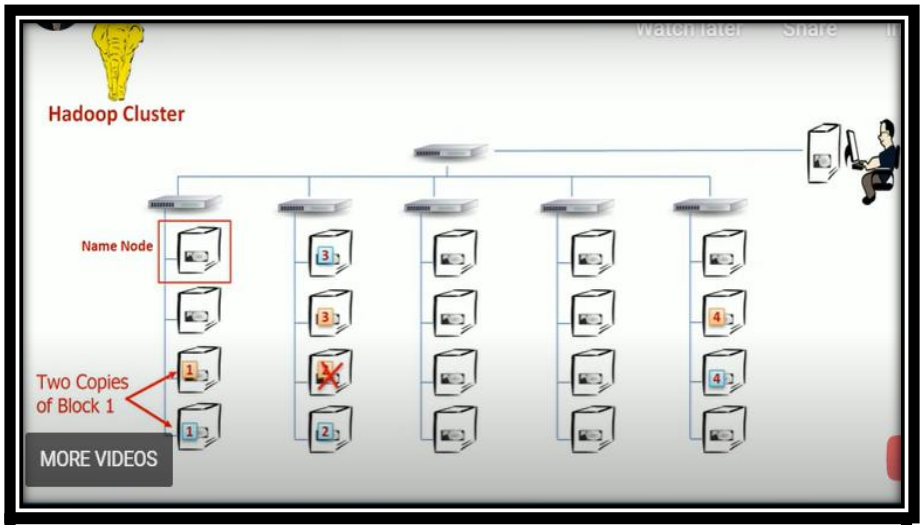
Types of Failures

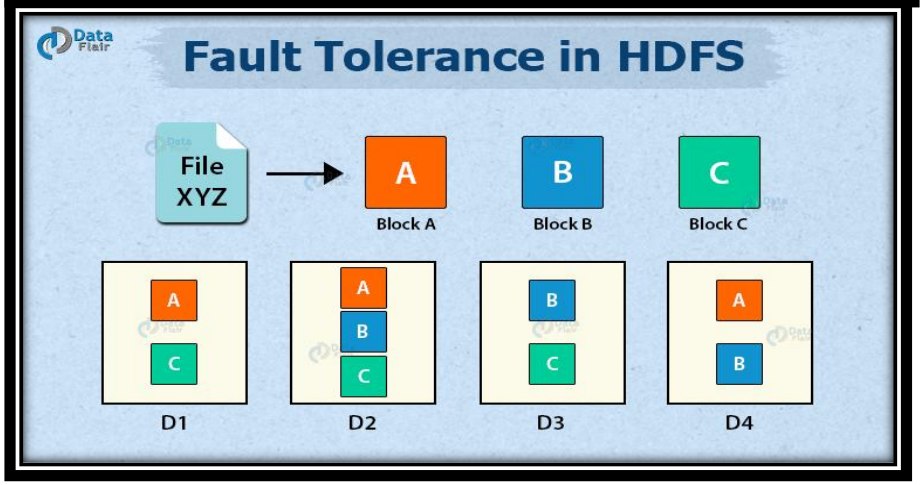
- Different Types of Failures that can happen in Hadoop Cluster are
  - ✓ Disk Failure
  - ✓ Node Failure
  - ✓ Rack Failure
  - ✓ Name Node Failure
- Replication helps to protect Data from **Disk and Node Failure**



FAULT TOLERANCE

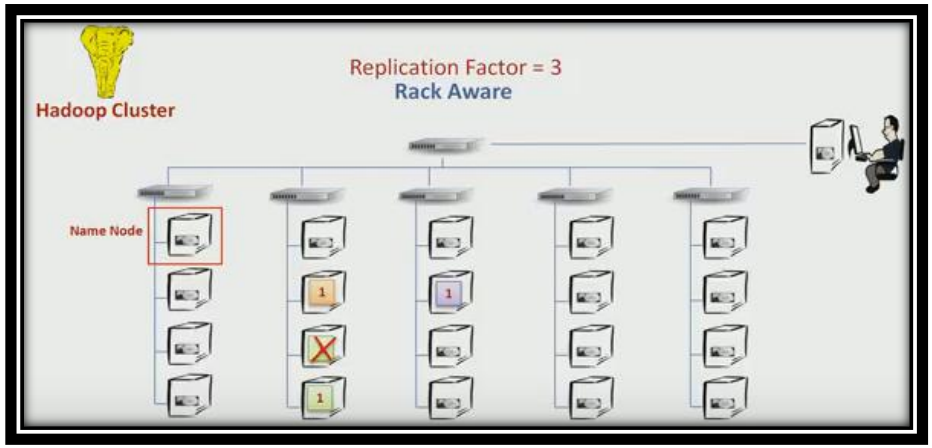
- **Fault tolerance refers to the ability of the system to work or operate even in case of unfavorable conditions (like components failure).**
- HDFS is **highly fault-tolerant**. it handles faults by the process of **replica creation**.
- It creates a replica of users' data on different machines in the HDFS cluster.
- So whenever if any machine in the cluster goes down, then data is accessible from other machines in which the same copy of data was created.
- We can set this replication on file to file basic, (by default its value is 3)
- if we set replication value as 3, HDFS will automatically makes 3 copies of each block for a file.
- Hadoop will also ensure that it keeps these three copies on rack Aware by placing atleast one node in different rack, so that if a complete rack get failed the data can be backed up easily.

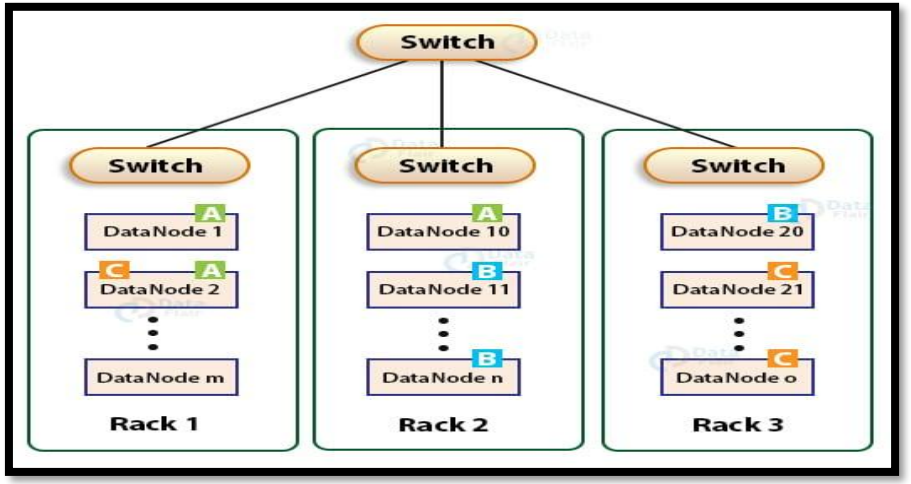




Replica placement via Rack awareness in Hadoop

- HDFS stores replicas of data blocks of a file to provide **fault tolerance and high availability**.
- ***If we store replicas on different nodes on the same rack, then it improves the network bandwidth, but if the rack fails (rarely happens), then there will be no copy of data on another rack.***
- Again, if we store replicas on unique racks, then due to the transfer of blocks to multiple racks while writes increase the cost of writes.
- Therefore, **NameNode on multiple rack cluster maintains block replication by using inbuilt Rack awareness policies which says:**
  - ✓ **Not more than one replica be placed on one node.**
  - ✓ **Not more than two replicas are placed on the same rack.**
- For the common case where the replication factor is three, the block replication policy put the first replica on the local rack, a second replica on the different DataNode on the same rack, and a third replica on the different rack.





HIGH AVAILABILITY

- The High availability feature of Hadoop ensures the availability of data even during NameNode or DataNode failure.
- Since HDFS creates replicas of data blocks, if any of the DataNodes goes down, the user can access his data from the other DataNodes containing a copy of the same data block.
- Also, if the active NameNode goes down, the passive node takes the responsibility of the active NameNode. Thus, data will be available and accessible to the user even during a machine crash.

HIGH AVAILABILITY

Although Rack, node and disk failure happens, these faults doesn't bring the entire system down your Hadoop cluster remains available, which shows Hadoop is rich in high availability.

What happens when the name node fails?

- We know that name node maintains the file system namespace.
- It also manages the file to block mapping.
- Every client interaction start with name node, so if name node fails we can't use Hadoop cluster.
- Name node is a single point of failure in cluster.
- To achieve high availability we need to protect it against name node failure

How to protect against name node(NN) failure?

- The solution is to back up the namespace content (HDFS namespace information)

HDFS namespace information:

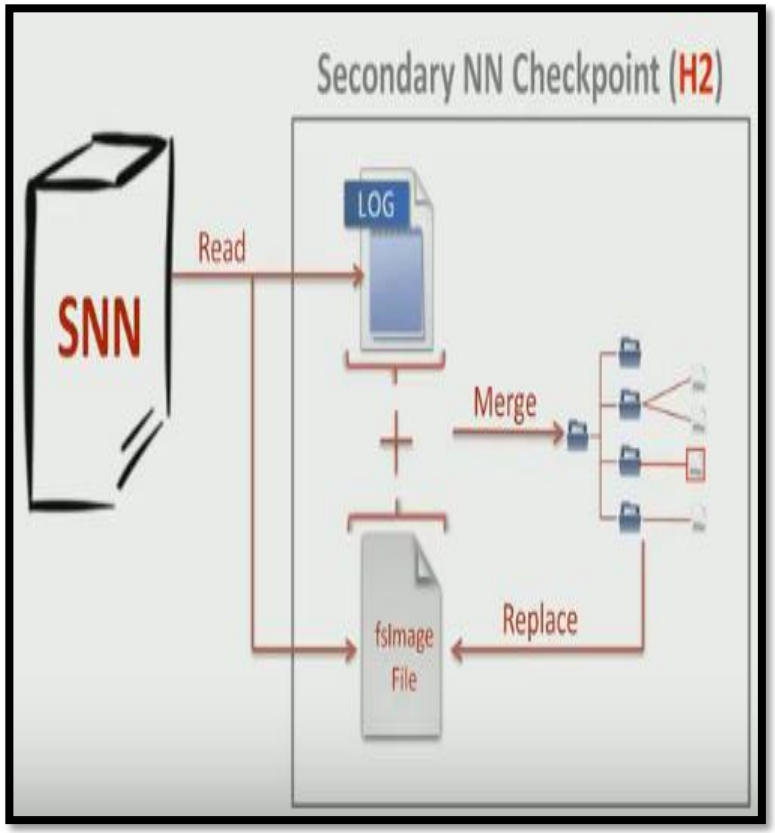
- All the information that name node maintains, should be continuously backed up at some other place.
- so that in the case of failure you have all necessary information to start a new name node on different machine.

SECONDARY NAME NODE

- Secondary NN differs from standby NN.
- Standby NN is a backup for Active NN, whereas secondary is not for that.
- Whenever we restart the name node we will lose the fsImage because it is an in-memory image (In-memory fsImage is the latest and updated picture of Hadoop file system namespace).
- Of course we have editlog to create new fsImage, but it may take more time to read the editlog because the log keeps growing bigger & bigger everyday this directly impact the restart time for a name node which may take hours of time.
- So to solve this problem we came with secondary NN.

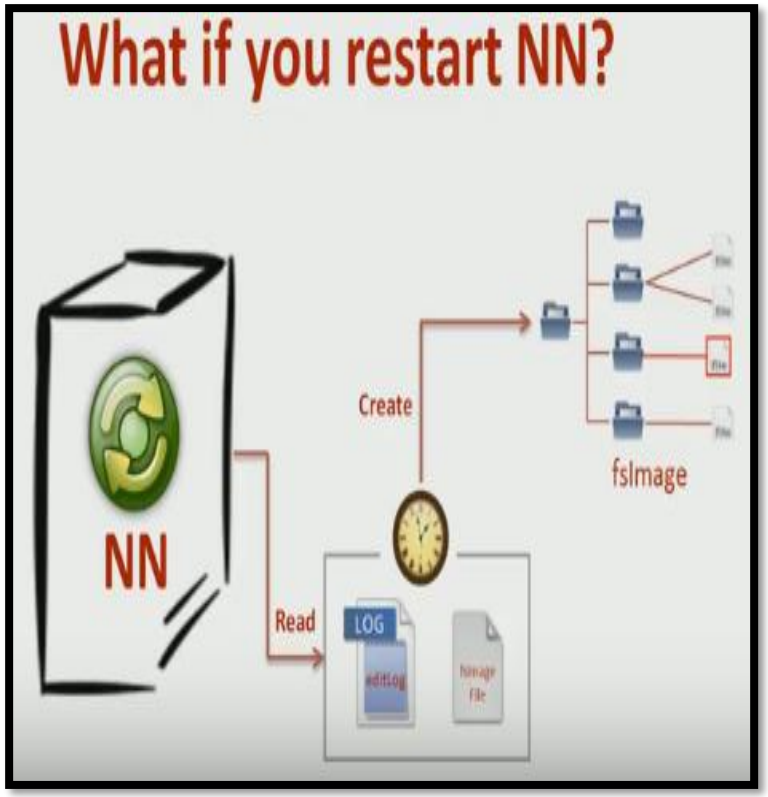
Secondary Name Node

- Secondary NN performs a **checkpoint activity every hour**.
- During the check point, the secondary NN will read the editlog and create the **latest file system image and save it to disk.**
- This state is exactly same as the In-memory fsImage. We call it on-disk fsimage.
- Once we have this on-disk fsImage the secondary NN truncate the edit log because all the changes are already applied, After an hour secondary NN will read the on-disk fsImage and apply all the changes from the editlog that we accumulated during the last 1 hour.
- It will replace the code on-disk fsImage with the new one and truncate the edit log once again. So the checkpoint activity is noting but the merging of an ondisk fsImage and editlog.



Secondary Name Node

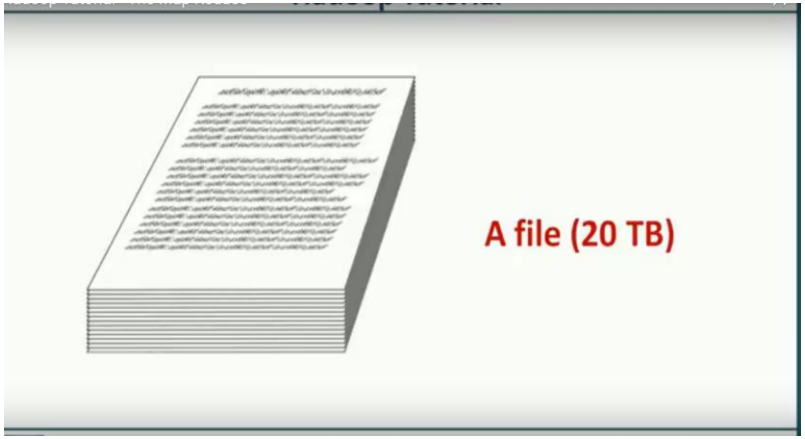
- The checkpoint doesn't take much time because we have just an hour old editlog to apply to the ondisk fsImage.
- The time to read fsImage is short because the fsImage is small compared to the editlog.
- The editlog is like a general ledger that records every transaction.
- The fsImage is like a balance sheet that shows the final state.
- So, the fsImage is small. In case of restart the name node also performs the same checkpoint activity that it can finish in a short time.
- Therefore the whole purpose of secondary NN and checkpoint is to minimize and restart time for the name node.



DATA LOCALITY

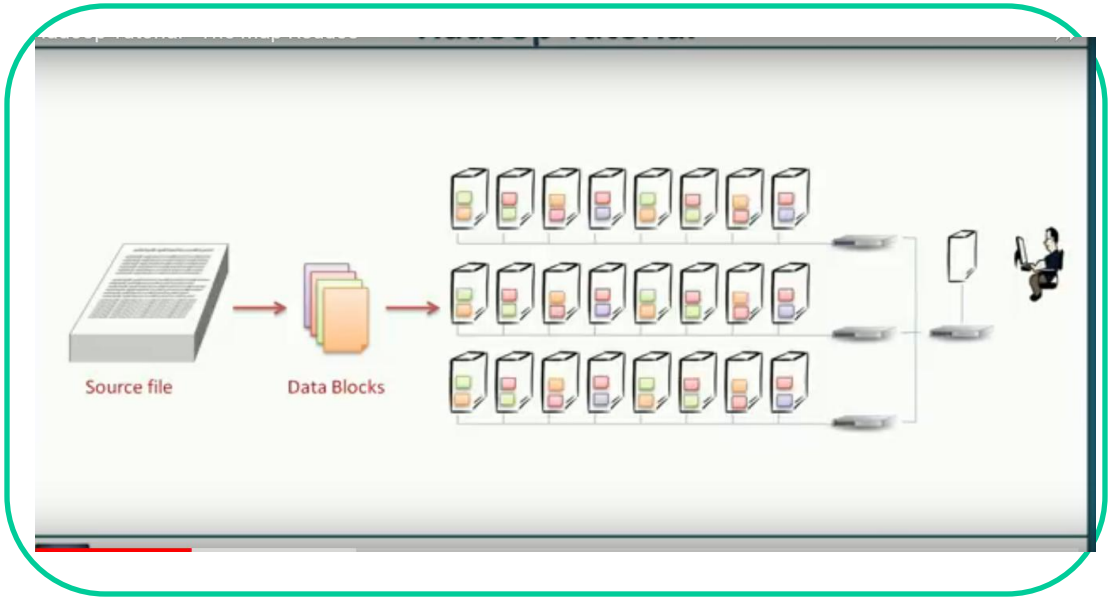
- In Hadoop, Data locality is the process of moving the computation close to where the actual data resides on the node, instead of moving large data to computation.

- This minimizes network congestion and increases the overall throughput of the system. .

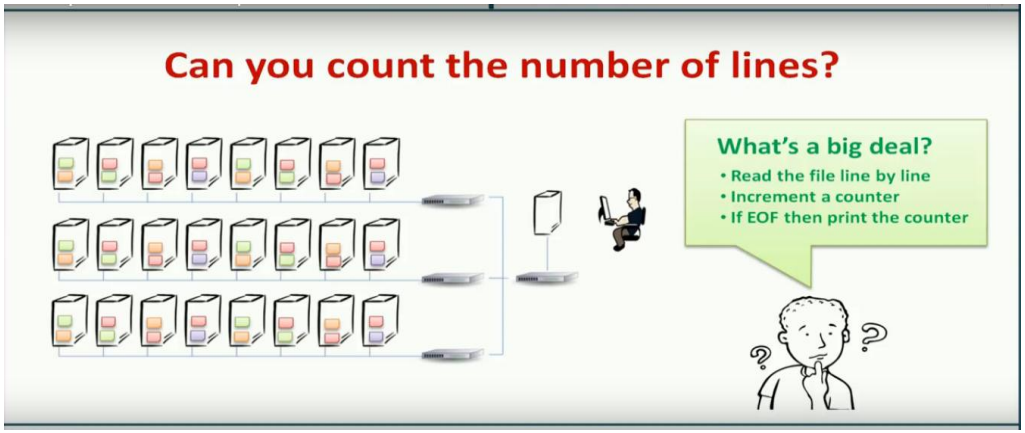


Lets us take a file of 20TB. We can't store the file in single machine .

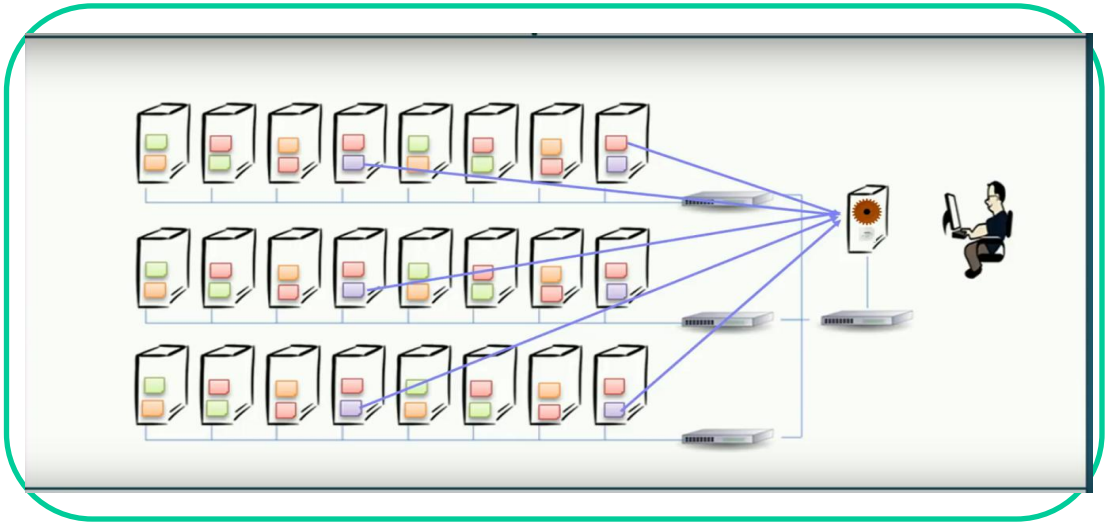
So we decided to store it in Hadoop Cluster



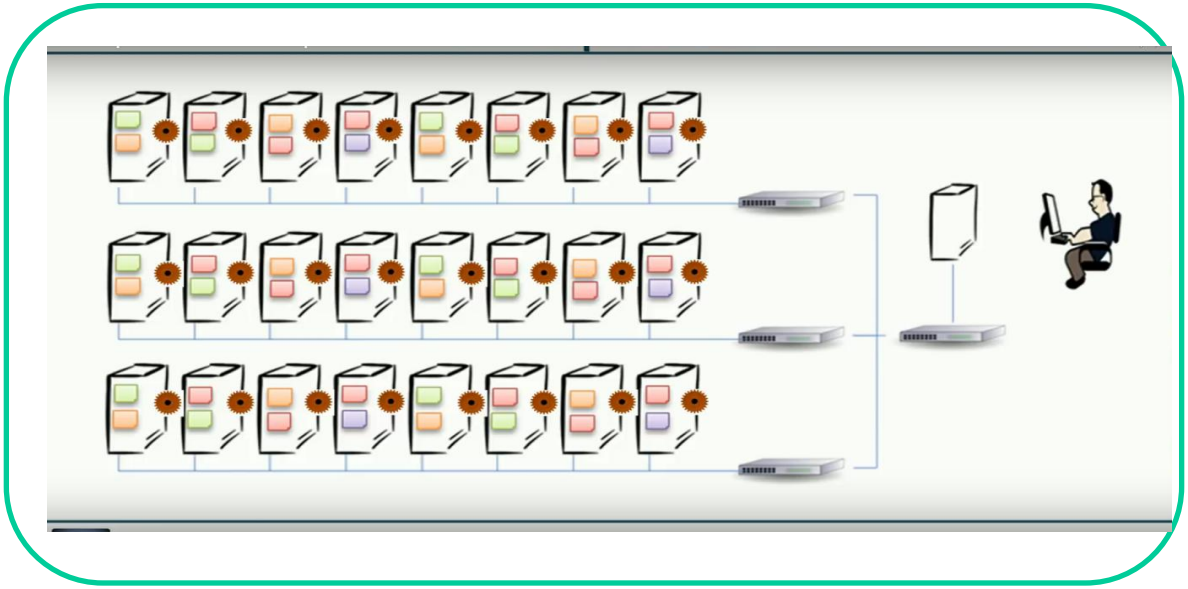
Divide the file into blocks . The Default Block size is 128 MB. Then store it in HDFS



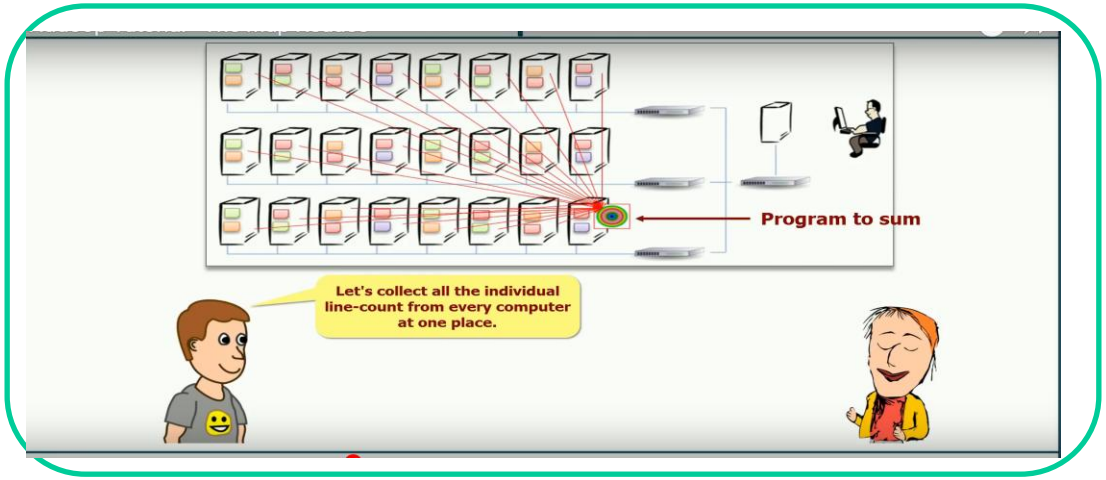
One way to count number of lines is to read a line and then increment the counter . Do the process until the EOF is Reached.



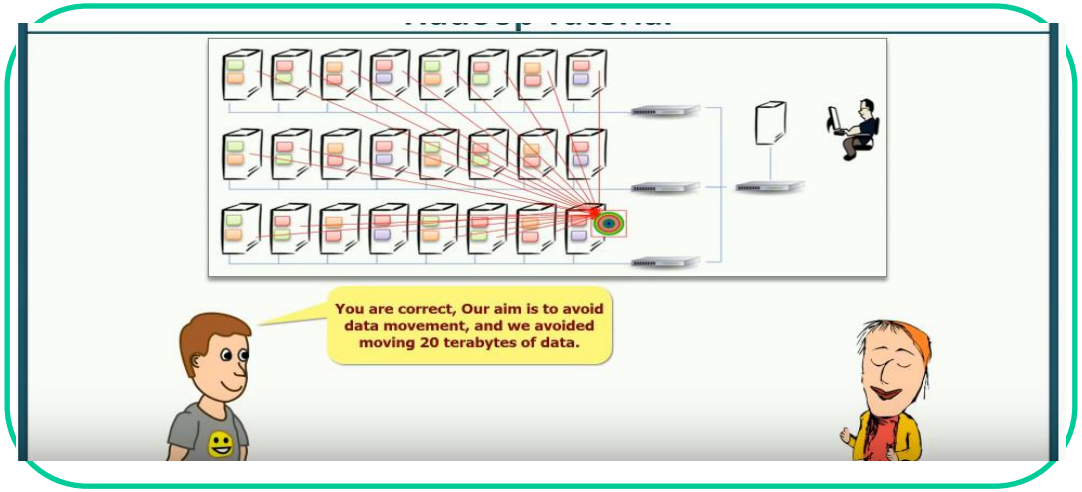
Bring all the Data to a single machine and execute the Java Program that counts the number of lines. This Process takes more Band width and more processing time .



Instead of Bringing data to processing , send processing to data and that is parallel processing



Collect all the Individual Line-count from every computer at one Place

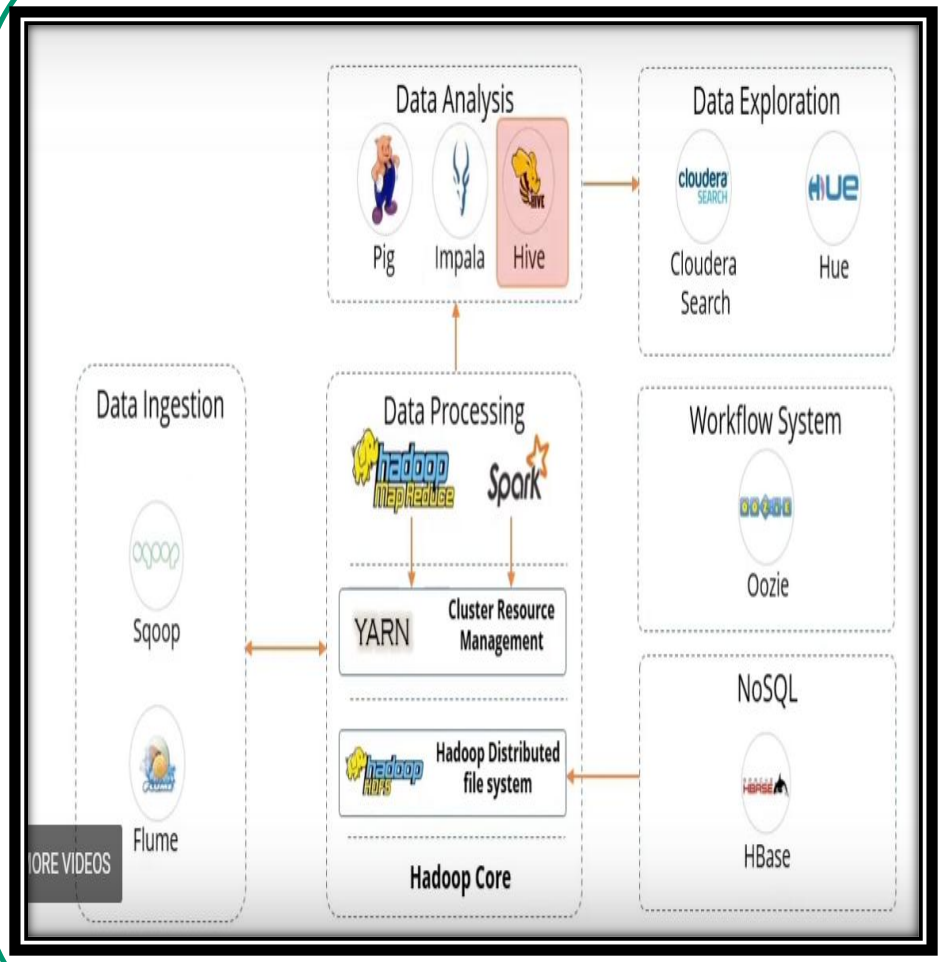


Our aim is to avoid Data Movement and has avoided moving 20 terabytes of Data

Apache Hadoop and Hadoop Ecosystem

- Hadoop is a Big Data Processing Tool (or)
- Hadoop is s a open source framework that allows to store and process big data in a distributed environment across clusters of computer using simple programming models
- Two major components of Hadoop are
  1) **Hadoop Distributed File System** **(For Distributed Storage)**
  2) **Map Reduce** **(For Parallel Processing)**
- Besides these two components, many components have been introduced to deal with Big Data
- And these components are collectively called as Hadoop Ecosystem

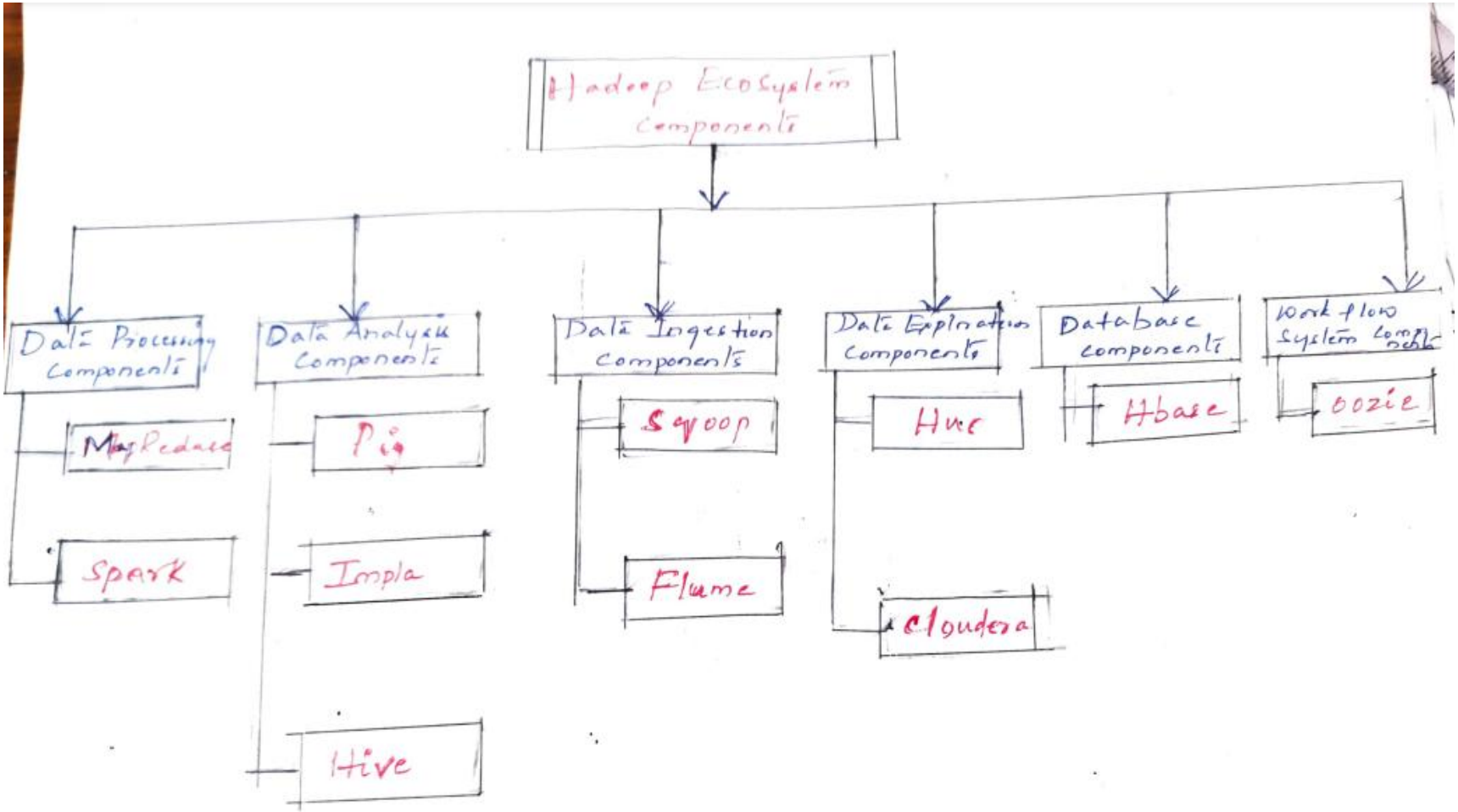
Apache Hadoop and Hadoop Ecosystem



Hadoop Ecosystem components are categorized into

1) Data Processing Components
2) **Data Analysis Components**
3) Data Ingestion Components
4) **Data Exploration Components**
5) Data Base Components
6) **Work flow system Components**

Apache Hadoop and Hadoop Ecosystem



Apache Hadoop and Hadoop Ecosystem

1) Data Processing Components

1.1) Map Reduce

- ✓ It is a parallel programming model for processing large amounts of structured, semi-structured, and unstructured data on large clusters of commodity hardware.

1.2) Spark

- ✓ Spark is an open-source distributed engine for processing and analzing huge volumes of real time data
- ✓ Provides 100 times faster performance as compared to MapReduce
- ✓ Provides in-memory computation of data
- ✓ Used to process and analyze real time streaming data such as stock market and banking data
- ✓ Supports Machine Learning, Business Intelligence, Streaming and Batch Processing

2) Data Analysis Components

2.1) Pig

- ✓ Pig is used to analyze data in Hadoop.
- ✓ It Provides a high level data processing language to perform numerous operations on the data.
- ✓ Language used in Pig is Pig Latin
- ✓ Pig Includes Pig Latin, a scripting language.
- ✓ Pig translates Pig Latin Scripts into MapReduce, which can ten process data in the HDFS cluster.
- ✓ Less Number of code is required to write in Pig when compared to MapReduce i.e 10 lines of pig latin script is around 200 lines of map Reduce job

2.2) Impla

- ✓ Impla is the highest performing SQL Engine which provides the fastest way to access data that I stored in Hadoop Cluster.
- ✓ Impla is Ideal for Interactive Analysis
- ✓ The Latency is very low and it is in measured in milli seconds

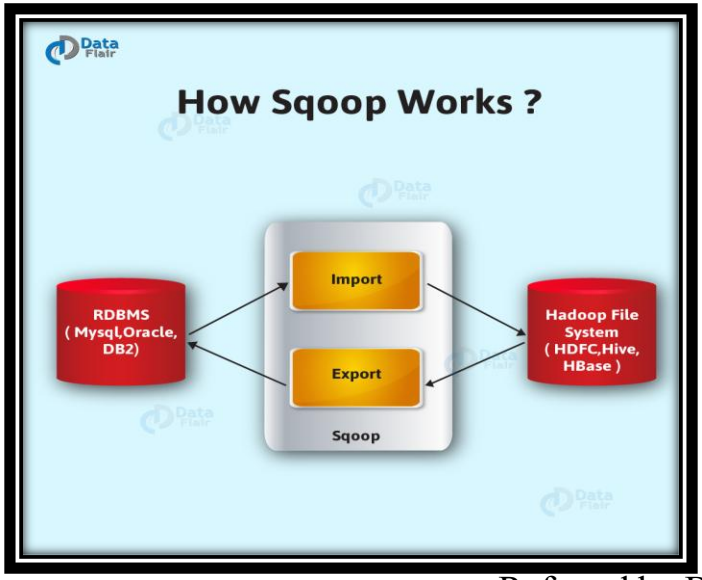
2.3) Hive

- ✓ The Hadoop ecosystem component, Apache Hive, is an open source data warehouse system for querying and analyzing large datasets stored in Hadoop files. (acts like a OLTP Tool)
- ✓ Hive do three main functions: *data summarization, query, and analysis*.
- ✓ Hive use language called HiveQL (HQL), which is similar to SQL. HiveQL automatically translates SQL-like queries into MapReduce jobs

3) **Data Ingestion**

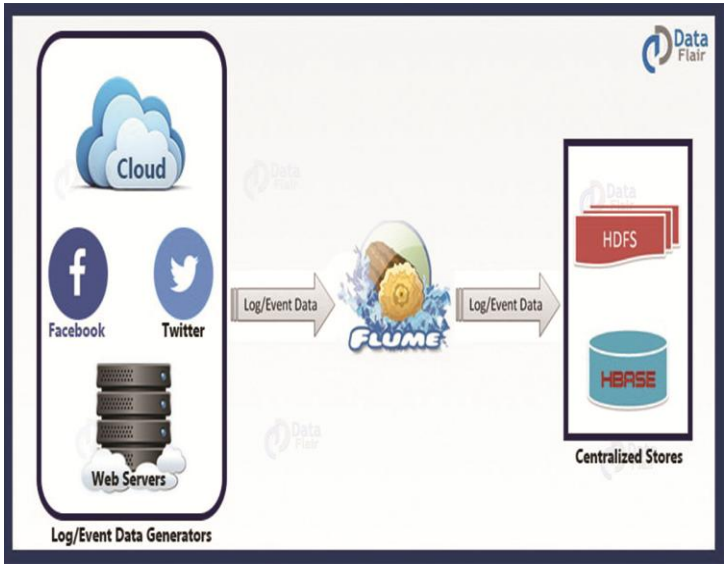
3.1) Sqoop

- ✓ Sqoop is a tool designed to transfer data between Hadoop and Relational Database Servers
- ✓ It is used to import data from relational databases such as Oracle, MYSQL to HDFS and Export data from HDFS to Relational databases



3.2) Flume

- ✓ To Ingests online streaming data from social media, log files, web server into HDFS
- ✓ **Flume** efficiently collects, aggregate and moves a large amount of data from its origin and sending it back to HDFS.



Apache Hadoop and Hadoop Ecosystem

4)Data Exploration

4.1) Hue

- ✓ Hue is an Acronym for Hadoop User Experience
- ✓ It provides SQL Editors fro Hive, Impla, MySQL, Oracle, PostgreSQL etc..
- ✓ Hue is an Open source web interface for analyzing data with Hadoop

4.2) Cloudera

- ✓ One of cloudera's near real time access products
- ✓ Users do not need SQL or Programming skills to use Cloudera Search
- ✓ Enables non-technical users to search and explore data stored in or ingested into Hadoop and HBase.

5) Data Base Components

5.1) Hbase

- ✓ Stores data in HDFS
- ✓ A NoSQL database or Non-relational database
- ✓ Mainly used when you need random. Real-time, read/write access to your Big Data.
- ✓ Provides support to high volume of data and high throughput

Limitations of Hadoop

- ❑ Hadoop can perform only batch processing, and data will be accessed only in a sequential manner. That means one has to search the entire dataset even for the simplest of jobs.
- ❑ A huge dataset when processed results in another huge data set, which should also be processed sequentially. At this point, a new solution is needed to access any point of data in a single unit of time (random access).

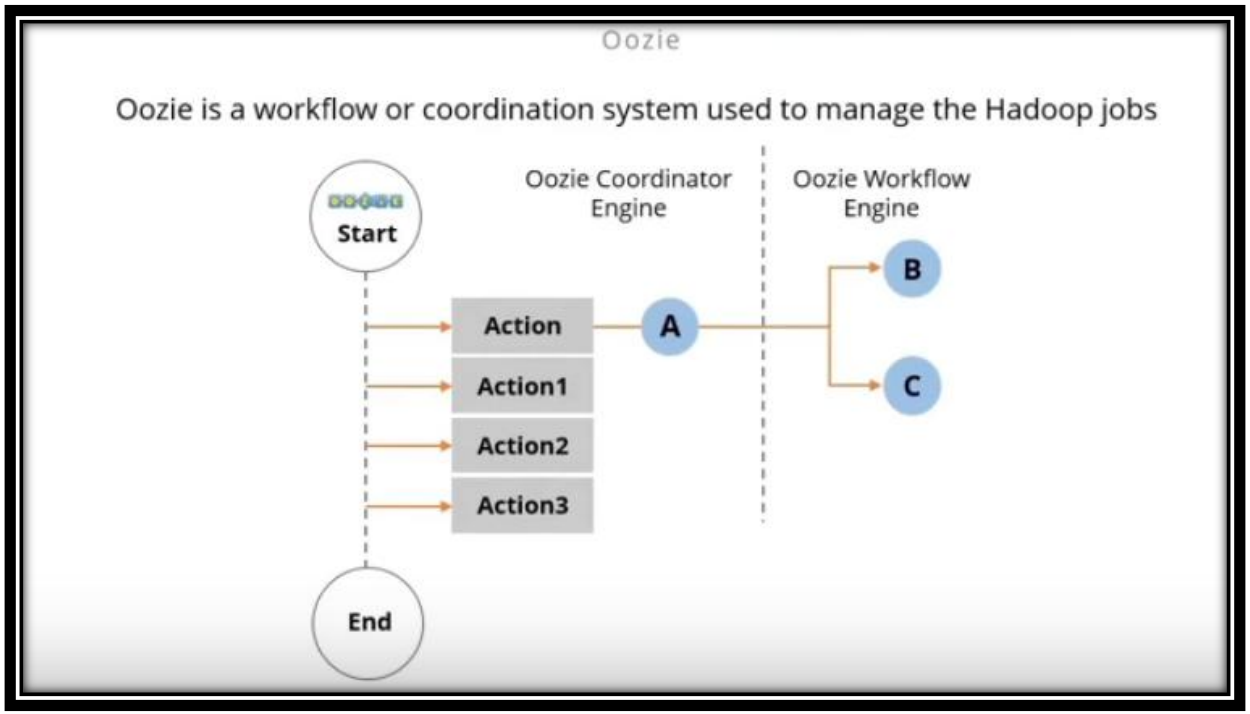
Hadoop Random Access Databases

- ❑ Hbase stores huge amounts of data and access the data in a random manner.
- ❑ HBase is a distributed column-oriented database built on top of the Hadoop file system.
- ❑ One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

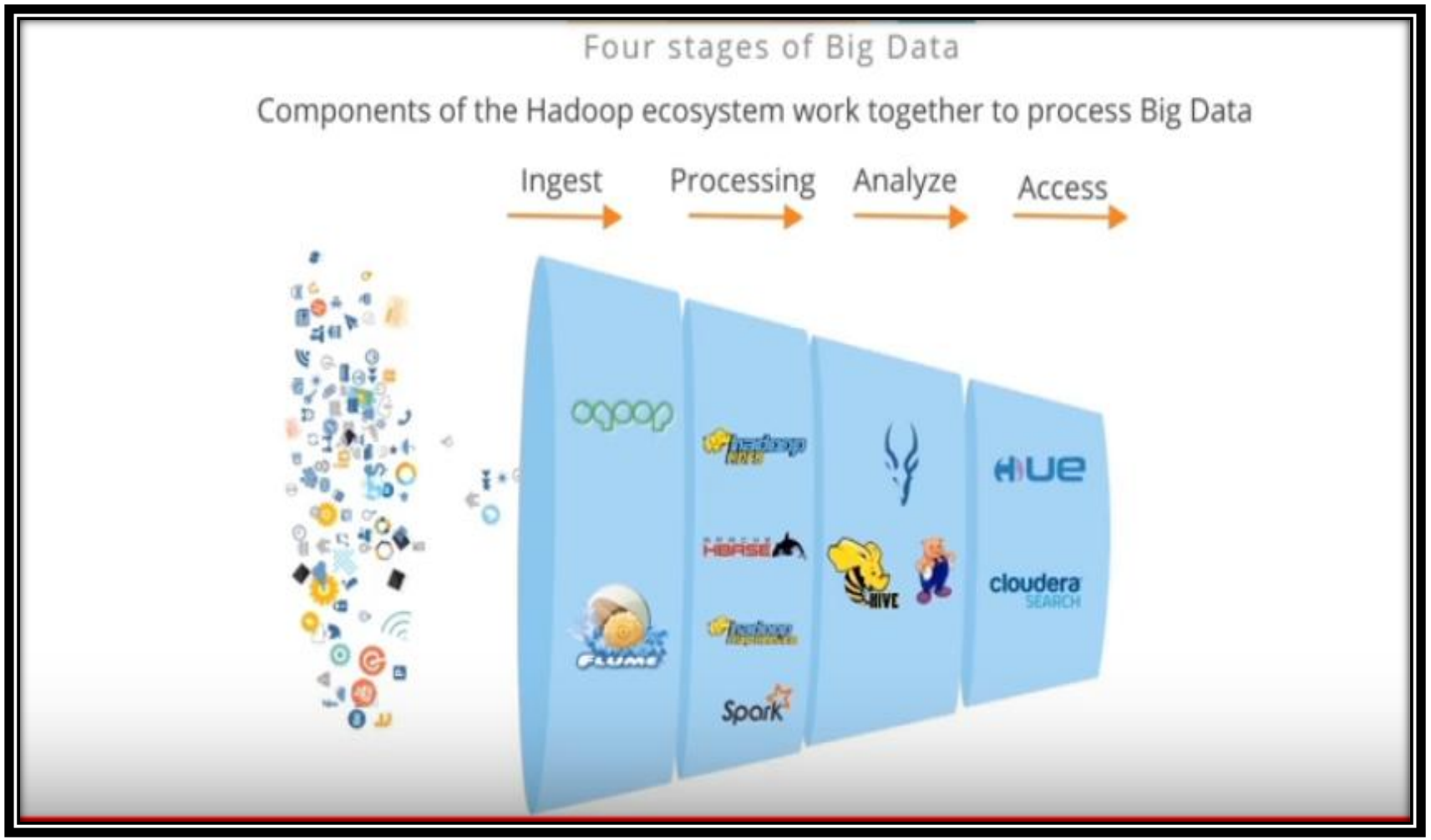
6) Work Fow System Components

6.1) Oozie

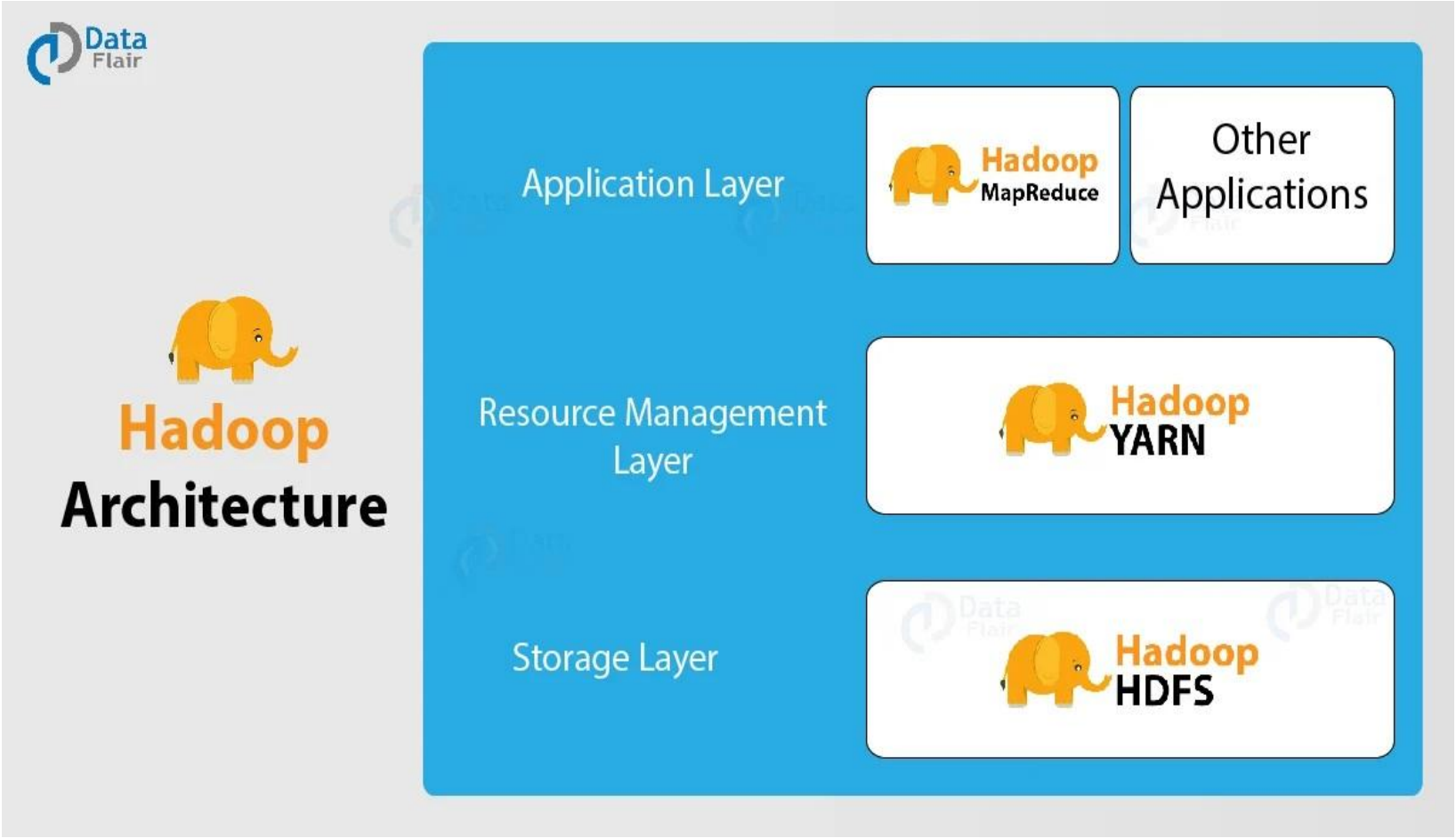
✓ Oozie is a workflow scheduler system to manage Hadoop Jobs



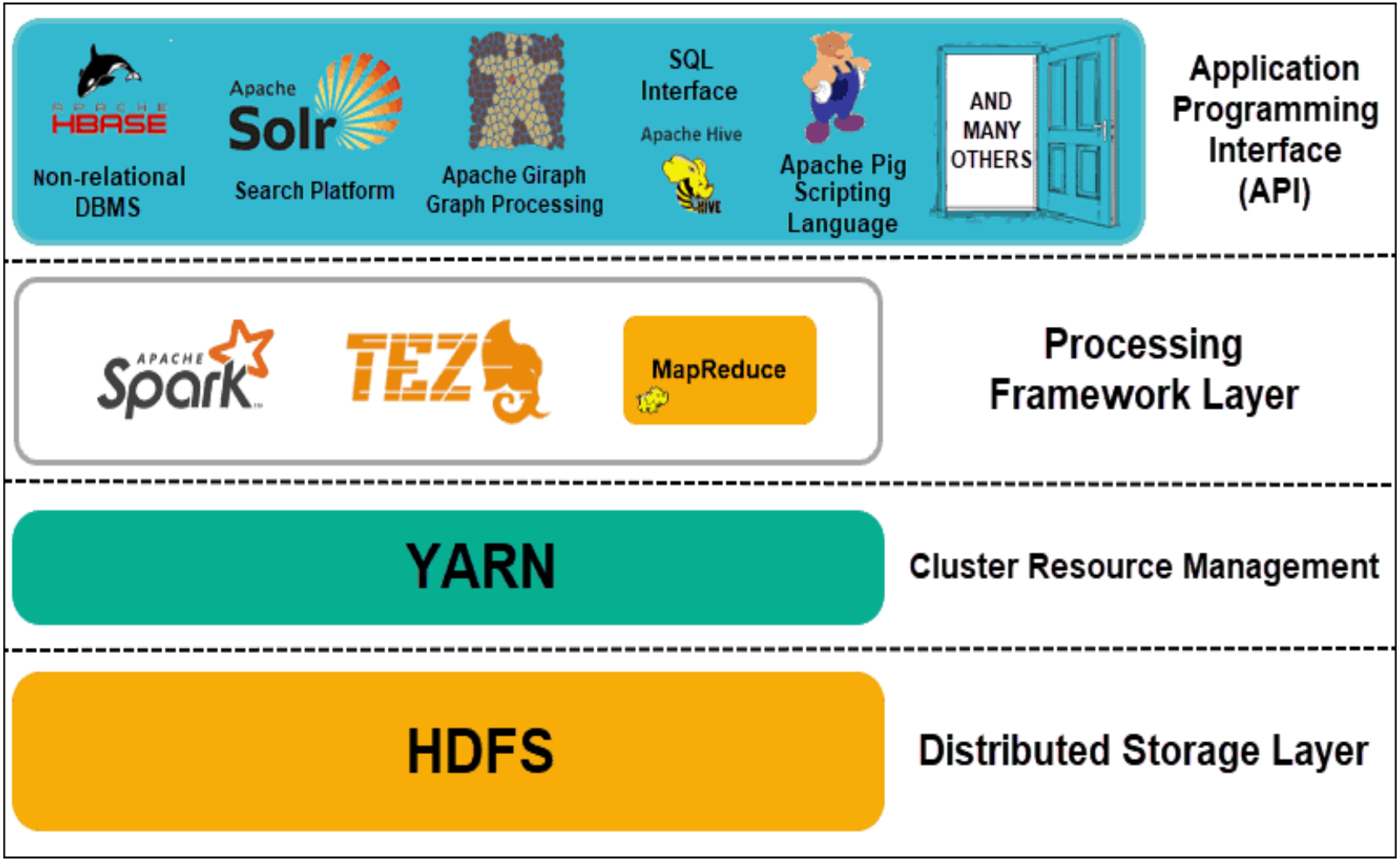
Apache Hadoop and Hadoop Ecosystem



Hadoop Architecture



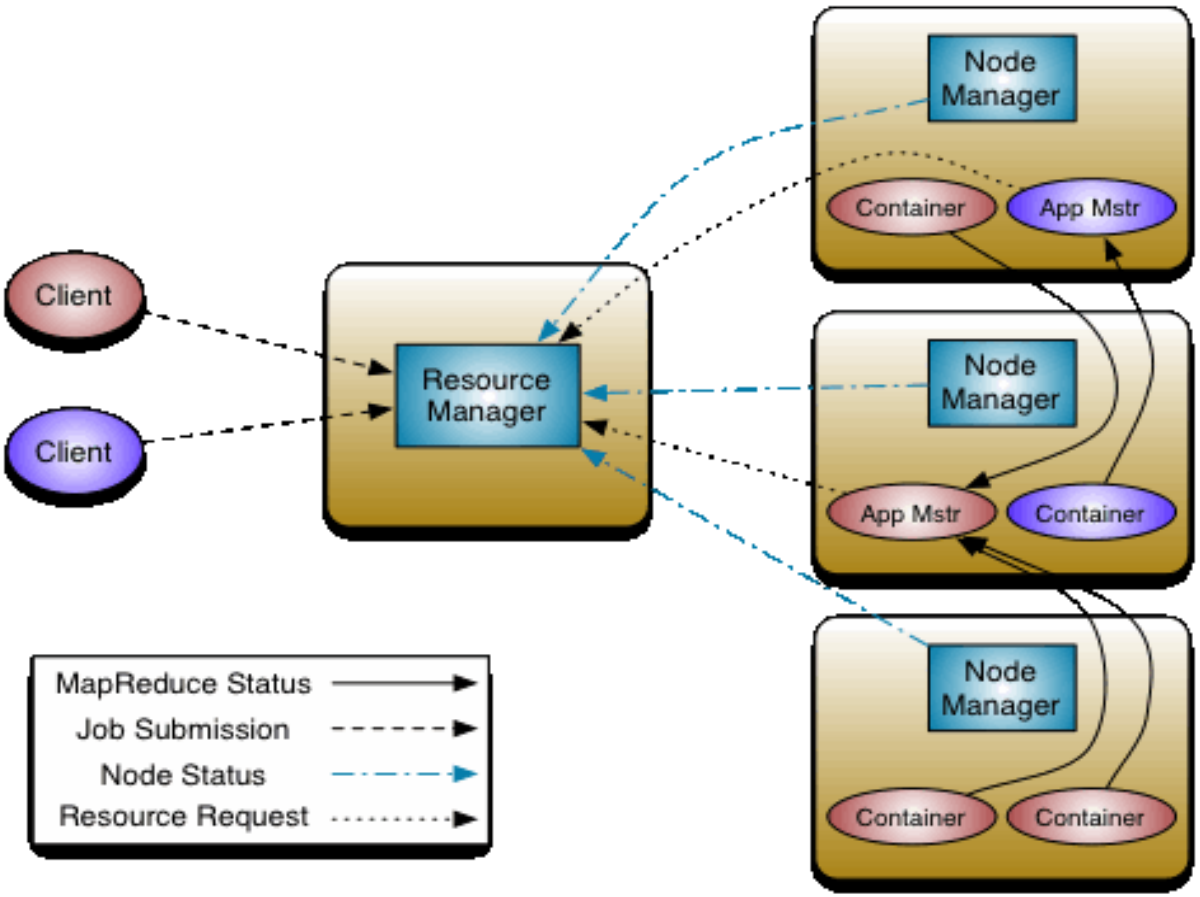
Hadoop Architecture



YARN – Yet Another Resource Negotiator

1. YARN (Yet Another Resource Negotiator) is a new component added in Hadoop 2.0.
2. YARN is a resource manager that was created by separating the **processing engine** and **resource management** capabilities of MapReduce.
3. YARN is often called the operating system of Hadoop because it is responsible for managing and monitoring workloads, maintaining a multi-tenant environment, implementing security controls, and managing high availability features of Hadoop.
4. Like an operating system on a server, YARN is designed to allow multiple, diverse user applications to run on a multi-tenant platform.
5. In Hadoop 1, users had the option of writing MapReduce programs in Java, in Python, Ruby or other scripting languages using streaming, or using Pig, a data transformation language.
6. YARN supports multiple processing models in addition to MapReduce.

YARN architecture:



The main components of YARN architecture include:

- **Client:** It submits map-reduce jobs.

Resource Manager: It is the master daemon of YARN and is responsible for resource assignment and management among all the applications.

It has two major components

- **Scheduler:** It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.
- **Application manager:** It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.

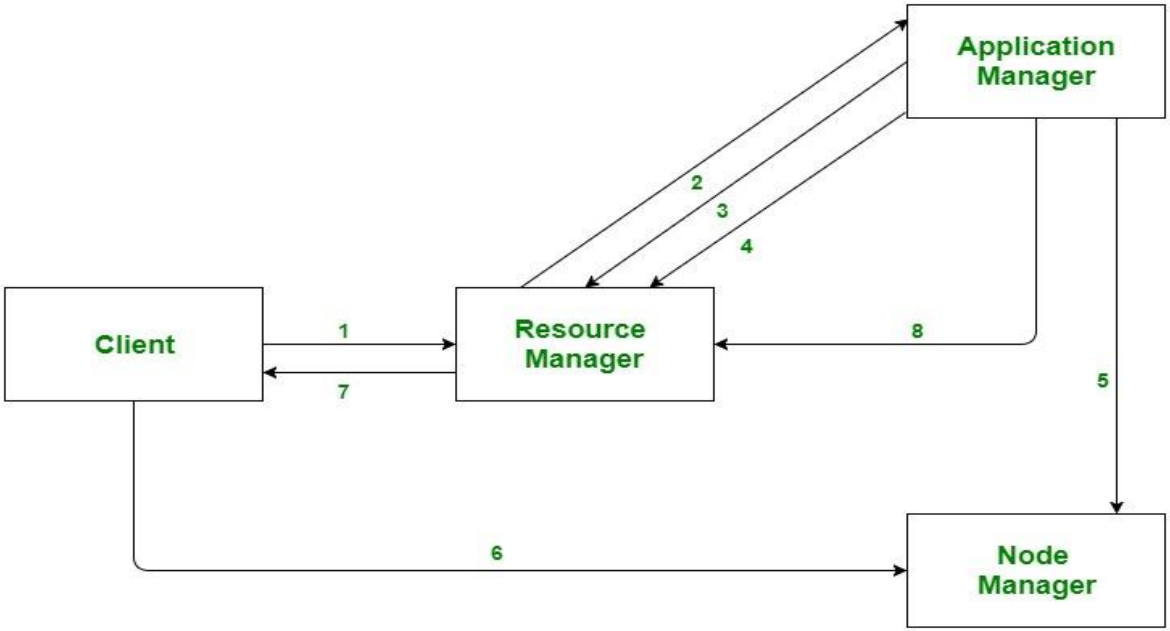
Node Manager: It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Resource Manager. It registers with the Resource Manager and sends heartbeats with the health status of the node.

Application Master: The application master is responsible for **negotiating resources** with the resource manager, **tracking the status and monitoring progress of a single application**.

The application master requests the container from the node manager by sending a Container Launch Context(CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.

Container: It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context(CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

Application workflow in Hadoop YARN:



1. Client submits an application
2. The Resource Manager allocates a container to start the Application Manager
3. The Application Manager registers itself with the Resource Manager
4. The Application Manager negotiates containers from the Resource Manager
5. The Application Manager notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application's status
8. Once the processing is complete, the Application Manager un-registers with the Resource Manager

UNIT II:

BIG DATA TOOLS AND FRAMEWORKS

1. **Apache Spark Architecture and RDDs**
2. **Spark SQL, Data Frames, and Datasets**
3. **Spark Streaming for Real-Time Analytics**
4. **Kafka for Data Ingestion and Message Queues**
5. **Hive, Pig, and Impala for Big Data Querying**
6. **Comparative Analysis of Hadoop vs. Spark**

What is Apache Spark

- Apache Spark is an open-source, distributed computing framework designed for big data processing and analytics.
- Designed to process large-scale datasets efficiently across multiple machines.
- Provides both batch and real-time (streaming) data processing.
- Runs programs up to 100x faster in memory and 10x faster on disk than traditional Hadoop MapReduce.

It offers high-level APIs in **Scala, Java, Python (PySpark), R, and SQL**

1.2Apache Spark Architecture and RDDS

Apache Spark follows a master–slave architecture with a driver program (master) and cluster workers (slaves). It uses a distributed execution engine to process large datasets across multiple nodes in a cluster.

Components of Spark Architecture

Driver Program (Master Node)

- The **entry point** of a Spark application.
- Runs the **main() function** of your program.
- Responsible for:
    - Maintaining **SparkContext** (connection to cluster).
    - Converting user code into a **Directed Acyclic Graph (DAG)**.
    - Splitting the DAG into **stages** and **tasks**.
    - Scheduling tasks on executors.
    - Collecting results back from executors.

Cluster Manager

- Manages resources (CPU, memory) across the cluster.
- Spark supports multiple cluster managers:
    - **Standalone** (built-in Spark manager)
    - **YARN** (Hadoop cluster manager)
    - **Apache Mesos**
    - **Kubernetes**

Worker Nodes

- The machines in the cluster that **run tasks assigned by the driver**.
- Each worker node hosts:

a) Executor

- A process launched for each Spark application.
- Runs tasks and stores data in memory/disk.
- Executors remain alive throughout the application (unlike MapReduce).

b) Task

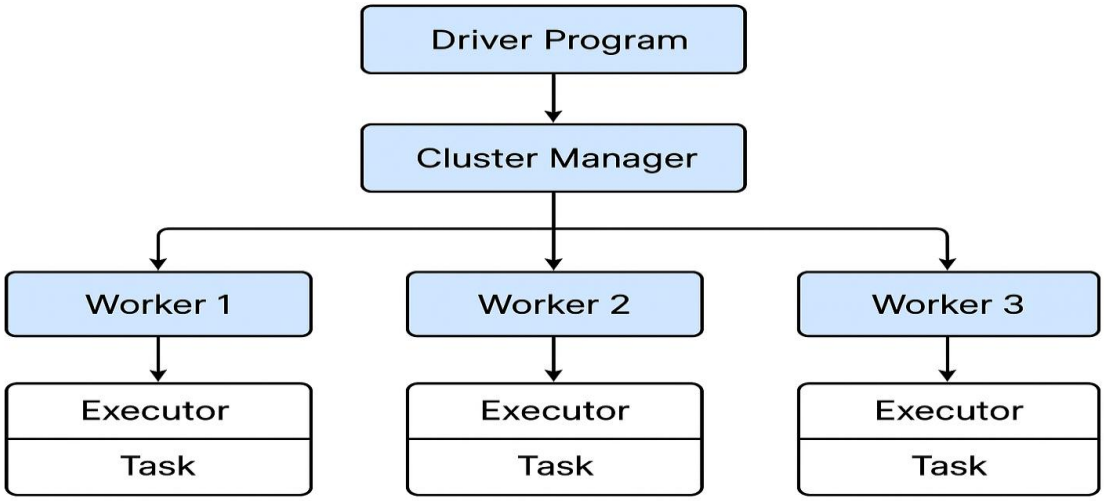
- The **smallest unit of execution** in Spark.
- One stage is divided into many tasks, distributed across executors.

RDD (Resilient Distributed Dataset)

- The fundamental abstraction in Spark.
- Immutable distributed collection of objects, partitioned across nodes.
- Provides **fault tolerance** using **lineage information** (recomputes lost partitions).

How Spark Executes a Job (Workflow)

1. **User Program**: You write Spark code (Scala, Python, Java, R, SQL).
2. **Driver Program**:
    - Initializes SparkContext.
    - Converts user code into a **DAG (Directed Acyclic Graph)** of stages.
3. **DAG Scheduler**:
    - Divides DAG into **stages**.
    - Each stage is split into **tasks** (based on data partitions).
4. **Task Scheduler**:
    - Assigns tasks to available executors on worker nodes.
5. **Cluster Manager**:
    - Provides resources (CPU, memory).
6. **Executors (on Workers)**:
    - Execute the tasks in parallel.
    - Store intermediate results in memory/disk.
7. **Driver**: Collects and combines the results.




```
from pyspark.sql import SparkSession
# Create Spark session
spark = SparkSession.builder.appName("EmpDF").getOrCreate()
emp_hist = map(lambda r: (int(r[0]), r[1], float(r[2]), r[3]),
            map(lambda x: x.split(","),
               ["101,alice,45000,nyc",
                "102,bob,55000,sfo",
                "103,charlie,60000,la",
                "104,diana,75000,chi",
                "105,edward,50000,hou"]))

emp_hist_rdd = spark.sparkContext.parallelize(list(emp_hist))
emp_new_rdd = spark.sparkContext.parallelize(list(emp_new))
print("Employee History DataFrame:")
emp_hist_df.show()
print("New Employee DataFrame:")
emp_new_df.show()
```

2.Spark SQL, Data Frames, and Datasets

Spark SQL

Definition:

Spark SQL is a module in Apache Spark designed for processing structured and semi-structured data. It enables users to run SQL queries on large-scale datasets and integrates seamlessly with Spark's DataFrame and Dataset APIs.

Key Features:

1. **SQL Queries on Spark Data:** Allows execution of standard SQL queries directly on Spark datasets.
2. **Multiple Data Sources:** Can read and query data from various sources such as Hive tables, JSON, Parquet, JDBC, and ORC.
3. **User-Defined Functions (UDFs):** Supports UDFs to extend SQL functionality with custom operations.
4. **Interoperability:** Works smoothly with Data Frames and Datasets, allowing easy conversion and optimized execution.
5. **Optimized Execution:** Uses the Catalyst optimizer for query optimization and the Tungsten engine for efficient memory and CPU management.

B. Data Frames

- **Definition:** A **Data Frame** is a distributed collection of data organized into **columns with names and types** (like a relational table).
- **Key Features:**
    1. High-level abstraction over **RDDs**.
    2. Supports **transformations** (select, filter, join, groupBy) and **actions** (show, count, collect).
    3. Optimized execution using **Catalyst optimizer** (query optimization) and **Tungsten engine** (memory and CPU optimizations).
    4. Works across **Python, Scala, Java, and R**.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("DFExample").getOrCreate()
data = [
   (1, "Alice", 20000),
   (2, "Bob", 30000),
   (3, "Charlie", 40000)
]
df = spark.createDataFrame(data, ["id", "name", "salary"])
df.show()
```

output:

```
+---+-------+------+
| id|  name |salary|
+---+-------+------+
|  1|  Alice| 20000|
|  2|    Bob| 30000|
|  3|Charlie| 40000|
+---+-------+------+
```

C. Datasets

- **Definition:** A **Dataset** is a distributed, **strongly-typed collection of objects**.
- **Key Features:**
    1. **Type-safe**: Errors can be caught at compile-time (Scala/Java).
    2. Combines **RDD advantages** (functional programming) and **DataFrame advantages** (optimized execution).
    3. Supports both **object-oriented** and **functional transformations**.
    4. Internally uses **DataFrame execution engine**, so it's optimized.

3. Spark Streaming for Real-Time Analytics

Definition:

Spark Streaming is a Spark module for processing live data streams in real-time. It allows continuous ingestion, processing, and analysis of data from sources such as Kafka, Flume.

Key Features

1. **Real-Time Data Processing**
    - Processes data as it arrives, rather than in large batches.
    - Supports low-latency computations, making it suitable for real-time dashboards, monitoring systems, and instant analytics.
    - Can handle high-volume, high-velocity streaming data from multiple sources simultaneously.

2. **Micro-Batching Model**
    - Spark Streaming divides incoming data into small batches (DStreams) and processes them sequentially.
    - Provides a hybrid approach, combining the scalability of batch processing with near real-time results.
    - Example: A Twitter stream can be processed in 1-second batches to count hashtags or mentions.

3. **Fault Tolerance**
    - Built on RDD lineage, which allows Spark to recompute lost data automatically in case of node failures.
    - Supports checkpointing for long-running streaming applications to ensure state recovery.
    - Ensures exactly-once processing semantics in certain sinks like Kafka and HDFS.

4. **Integration with Sources and Sinks**
    - Sources: Kafka, Flume, Kinesis, TCP sockets, HDFS, Amazon S3, and more.
    - Sinks (Outputs): HDFS, databases (MySQL, Cassandra), dashboards (Grafana, Kibana), Kafka topics, and file systems.
    - Enables flexible ingestion and output options, suitable for various use cases.

5. **Windowed Computations**
    - Supports sliding windows and tumbling windows for processing events over specific time intervals.
    - Useful for aggregations like moving averages, trend detection, or anomaly detection over recent events.
    - Example: Compute the sum of transactions in the last 10 minutes updated every 1 minute.

6. **High-Level APIs**
    - Supports transformations: map, flatMap, filter, reduceByKey, join.
    - Supports output operations: print(), saveAsTextFile(), foreachRDD().
    - Integration with Spark SQL allows running structured queries on streaming data (Structured Streaming)**.**

7. **Stateful Processing**

- Spark allows stateful operations, maintaining running counts or aggregations across batches.
- Useful for scenarios like session tracking, user activity analysis, and real-time metrics computation.
- Example: Counting clicks per user over a session period.

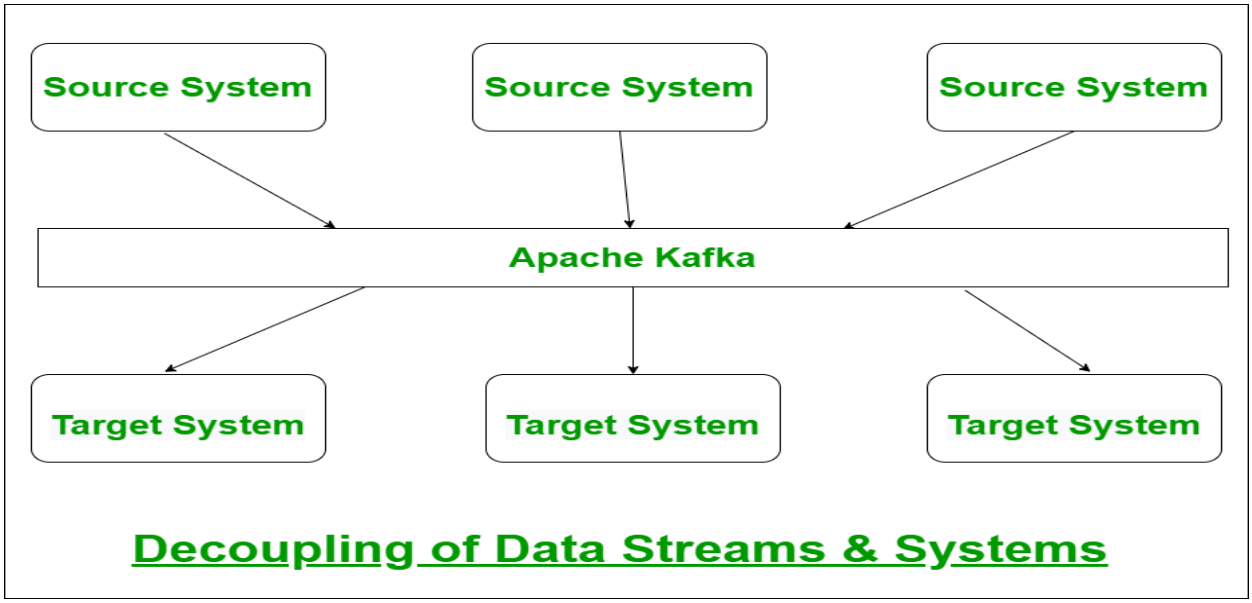
8. **Event-Time Processing and Watermarking**
   - Structured Streaming supports event-time processing, using the timestamp of data instead of arrival time.
   - Watermarks handle late-arriving data by specifying a threshold for lateness.
9. **Scalability and Performance**
   - Can process millions of records per second across a cluster of machines.
   - Scales horizontally by adding more worker nodes.
   - Supports backpressure handling, adjusting processing rate when sources produce data faster than it can be processed.
10. **Use Cases**
    - Real-time dashboards for monitoring IoT sensors.
    - Fraud detection in banking and credit card transactions.
    - Clickstream analytics for websites and e-commerce.
    - Social media sentiment analysis and trend detection.
    - Real-time recommendation engines.

4.Kafka for Data Ingestion and Message Queues

- Apache Kafka is a system that helps you send and receive large amounts of data very fast. It works in real time, meaning data moves immediately with almost no delay.
- Normally, if many systems need to share data, you must connect every system to every other system — which becomes complicated.
- Originally developed at **LinkedIn** and now under the maintenance of the Apache Software Foundation, Kafka is relied on by industry leaders such as Netflix, Uber, Walmart, and LinkedIn to process real-time data ingestion, streaming analytics, and event processing.
- Whether for user behaviour tracking, log aggregation, fraud detection, or fueling recommendation engines — Kafka scales with ease, provides millisecond-level performance, and guarantees data never goes missing.

What is Apache Kafka?

- Apache Kafka helps keep data-sending systems and data-receiving systems separate.
- The source systems only need to send their data to Kafka, and they do not need to know who will use it.
- Any system that wants the data can simply read it from Kafka. Because of this separation, the source and destination systems do not depend on each other.
- Kafka takes care of receiving the data and delivering it to whoever needs it.



- This method is not new—it is based on the publish–subscribe (pub-sub) model.
- But Apache Kafka is special because it can scale extremely well and handle a very large number of messages every second.
- The systems that send data to Kafka are called source systems. These could be things like website activity, pricing updates, financial transactions, or user actions.
- The systems that receive data from Kafka are called target systems, such as databases, analytics tools, email systems, or audit systems.

How Does Apache Kafka Work?

Apache Kafka is a distributed, high-performance platform for real-time data streaming and message processing. But if you're just starting out with it, you may be thinking how they works:

Think of Kafka as a large, really fast post office that receives messages (data) from various sources and sends them to their respective destinations.

- Producers send the messages (similar to people sending letters).
- Kafka brokers behave like postal staff – sorting, storing, and controlling the flow.
- Consumers get the messages (such as individuals opening their mailbox).
- Topics are similar to labeled bins or folders that categorize where messages go.
- Partitions assist in dividing the load so Kafka can process massive amounts of data in a timely manner.

Kafka Architecture

Kafka architecture is based on producer-subscriber model and follows distributed architecture, runs as cluster.

1. Kafka Producers

- These are the applications or systems that produce data into Kafka.
- For example, a mobile application producing user clicks, or an online store producing order information.
- Producers publish messages to a Kafka topic.

2. Kafka Topics

- A topic is a named stream of data.
- It groups messages by category. For example: orders, payments, or user-activity.
- Topics can be divided into partitions to enhance performance.

3. Kafka Partitions

- Partitions are similar to lanes in a highway. Every topic is assigned one or more partitions to deal with high-traffic volume.
- Messages within partitions are persisted in the same order they arrived

4. Kafka Brokers

- A Kafka broker is a server that continues the data and serves requests.
- Multiple brokers make up a Kafka cluster in big systems to share the load and ensure high availability.

5. Kafka Consumers

- Consumers are apps or services reading data from a Kafka topic.
- They subscribe to a topic and pull messages periodically.
- Consumers can operate as consumer groups for load sharing to avoid missing data.

Why Use Kafka for Data Ingestion?

- Apache Kafka is used for data ingestion because it can handle **millions of messages per second** and **scale easily** by adding more brokers.
- It is **reliable and fault-tolerant**, as messages are **replicated** across multiple servers. Kafka also **supports real-time analytics** and **decouples source and target systems**, so the systems sending and receiving data do not need to connect directly.
- This makes data pipelines simpler, faster, and more robust

Real-World Usage of Apache Kafka

- **Netflix** uses Kafka to apply a recommendation in real-time while you're watching TV shows.
- **Uber** uses Kafka to gather taxi user and trip data in real time and they will use it to compute and forecast demand and then they can compute the infamous search pricing in real-time to know how much to charge you for a ride in case there is a lot of high demand.
- **LinkedIn** uses Kafka to prevent spam and collect user interactions to make better connection recommendations in real time.
- Here is the example of how the food delivery apps work using kafka:
    - You place an order → this event is sent to Kafka (producer).
    - Kafka stores this in the orders topic, partitioned by order ID.
    - The delivery system (consumer) reads the message and assigns a driver.
    - The payment system (another consumer) reads the same message to process your payment.
    - Kafka keeps the data for a set time (e.g., 7 days) for audit or analytics

6.Comparative Analysis of Hadoop vs. Spark

Basis	Hadoop	Spark
Processing Speed & Performance	Hadoop's MapReduce model reads and writes from a disk, thus slowing down the processing speed.	Spark reduces the number of read/write cycles to disk and stores intermediate data in memory, hence faster-processing speed.
Usage	Hadoop is designed to handle batch processing efficiently.	Spark is designed to handle real-time data efficiently.
Latency	Hadoop is a high latency computing framework, which does not have an interactive mode.	Spark is a low latency computing and can process data interactively.
Data	With Hadoop MapReduce, a developer can only process data in batch mode only.	Spark can process real-time data, from real-time events like Twitter, and Facebook.
Cost	Hadoop is a cheaper option available while comparing it in terms of cost	Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost.
Algorithm Used	The PageRank algorithm is used in Hadoop.	Graph computation library called GraphX is used by Spark.

Basis	Hadoop	Spark
Fault Tolerance	Hadoop is a highly fault-tolerant system where Fault-tolerance achieved by replicating blocks of data. If a node goes down, the data can be found on another node	Fault-tolerance achieved by storing chain of transformations If data is lost, the chain of transformations can be recomputed on the original data
Security	Hadoop supports LDAP, ACLs, SLAs, etc and hence it is extremely secure.	Spark is not secure, it relies on the integration with Hadoop to achieve the necessary security level.
Machine Learning	Data fragments in Hadoop can be too large and can create bottlenecks. Thus, it is slower than Spark.	Spark is much faster as it uses MLib for computations and has in-memory processing.
Scalability	Hadoop is easily scalable by adding nodes and disk for storage. It supports tens of thousands of nodes.	It is quite difficult to scale as it relies on RAM for computations. It supports thousands of nodes in a cluster.
Language support	It uses Java or Python for MapReduce apps.	It uses Java, R, Scala, Python, or Spark SQL for the APIs.
User-friendliness	It is more difficult to use.	It is more user-friendly.

Basis	Hadoop	Spark
Resource Management	YARN is the most common option for resource management.	It has built-in tools for resource management.

TEXT BOOKS:

1. Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid Dastjerdi – Wiley
2. Learning Spark: Lightning-Fast Big Data Analysis by Jules S. Damji et al. – O'Reilly Data Science and Big Data Analytics by EMC Education Services – Wiley

REFERENCE BOOKS:

1. Designing Data-Intensive Applications by Martin Kleppmann – O'Reilly
2. Machine Learning with Spark by Rajdeep Dua, Tathagata Das – Packt Publishing
3. Streaming Systems by Tyler Akidau – O‘Reilly Media.
4. 4. Artificial Intelligence for Big Data by Anand Deshpande

REFERENCE WEBSITE:

1. https://www.coursera.org/specializations/big-data – Coursera Big Data Specialization
2. https://spark.apache.org/docs/latest/ – Apache Spark Documentation 3. https://www.edx.org/course/big-data-analysis-with-python – edX
3. https://www.udacity.com/course/ai-for-business-leaders--nd088 – Udacity AI for Business
4. https://www.kaggle.com/learn/intro-to-machine-learning – Kaggle ML Tutorials
5. https://data-flair.training/blogs/apache-spark-tutorial/ – Spark Tutorials

UNIT III

MACHINE LEARNING ON BIG DATA

Introduction to MLlib and Scikit-learn, Data Preprocessing for Big Data ML Pipelines, Supervised Learning: Classification and Regression on Large Datasets, Unsupervised Learning: Clustering and Dimensionality Reduction, Model Evaluation and Validation Techniques, Distributed Training and Optimization Techniques

Introduction to MLlib and Scikit-learn

MLlib (Machine Learning Library)

1. MLlib (Machine Learning Library) is the machine learning library of Apache Spark.
2. It provides scalable and distributed machine learning algorithms that can work efficiently on **large datasets (Big Data)**.
3. MLlib is a Spark-based library used for performing machine learning tasks such as classification, regression, clustering, and recommendation on big data.

Key Points about MLlib

- **MLlib is a component of Apache Spark**, used for machine learning tasks.
- It is specially **designed for Big Data processing**, handling very large datasets efficiently.
- MLlib supports **distributed computing across clusters**, allowing faster model training on multiple machines.
- It provides **ready-to-use machine learning algorithms and tools** such as classification, regression, and clustering.
- MLlib helps in building **complete machine learning pipelines**, including preprocessing, feature extraction, training, and evaluation.

Why MLlib is Important?

Traditional ML libraries struggle with very large datasets.

MLlib solves this by using Spark's distributed processing power, making it suitable for:

- Fraud detection
- Recommendation systems
- Large-scale predictive analytics
- Real-time ML applications

What is Scikit-learn?

Scikit-learn is one of the most popular **Python libraries for Machine Learning**.

It provides simple and efficient tools for:

- Data preprocessing

- Model training
- Model evaluation
- Machine learning algorithms

It is mainly used for **small to medium-sized datasets**.

Key Features of Scikit-learn

- **Scikit-learn is built on NumPy, SciPy, and Matplotlib**, making it efficient for numerical and scientific computing.
- It is **easy to learn and beginner-friendly**, with a simple and consistent API.
- It supports both **supervised learning** (classification, regression) and **unsupervised learning** (clustering, dimensionality reduction).
- It provides a wide range of **machine learning algorithms in a single package**, making model development faster.
- Scikit-learn is widely used in **research and real-world applications** for data analysis, prediction, and automation.

Data Preprocessing for Big Data ML Pipelines

Data preprocessing for Big Data refers to the process of cleaning, transforming, and organizing large-scale raw data into a suitable format for analysis and machine learning.

Since Big Data is often:

- Massive in size
- Unstructured
- Noisy and incomplete

preprocessing becomes an essential step before applying ML algorithms.

Why Preprocessing is Important in Big Data?

Big Data cannot be directly used for machine learning because it may contain:

- Missing values
- Duplicate records
- Inconsistent formats
- Outliers and noise
- Irrelevant features

Preprocessing helps improve:

- Data quality
- Model accuracy
- Training speed

- Scalability

Steps in Data Preprocessing for Big Data

1. Data Collection and Integration
2. Data Cleaning
3. Handling Missing Values
4. Data Transformation
5. Feature Scaling (Normalization/Standardization)
6. Encoding Categorical Data
7. Feature Selection
8. Dimensionality Reduction (PCA)
9. Data Sampling
10. Data Splitting (Train/Test/Validation)

1. Data Collection and Integration

Big Data is collected from multiple sources such as:

- Sensors
- Social media
- Transaction systems
- Logs and databases

Integration means combining data from different sources into one dataset.

Purpose: Create a unified dataset for analysis.

2. Data Cleaning

Big datasets often contain errors like:

- Duplicate records
- Wrong values
- Incomplete data
- Noise

Cleaning removes these issues.

Purpose: Improve data accuracy and consistency.

3. Handling Missing Values

Missing data is common in real-world datasets.

Methods:

- Remove rows with missing values
- Fill with mean/median/mode
- Use advanced imputation techniques

Purpose: Prevent incorrect model predictions.

4. Data Transformation

Raw data may not be in a usable format.

Transformation includes:

- Converting text into numbers
- Aggregating values
- Changing data types

Example: Converting dates into day/month/year.

Purpose: Make data suitable for ML algorithms.

5. Feature Scaling (Normalization / Standardization)

Features may have different ranges.

Example:

- Age: 20–60
- Salary: 20,000–1,00,000

Scaling methods:

- **Normalization** → values between 0 and 1
- **Standardization** → mean = 0, std = 1

Purpose: Helps models perform better and train faster.

6. Encoding Categorical Data

ML models only work with numerical values.

So categorical values like:

- Gender: Male/Female

- City: Delhi/Mumbai

are converted using:

- Label Encoding
- One-Hot Encoding

Purpose: Convert categories into machine-readable form.

7. Feature Selection

Big Data may contain many unnecessary features.

Feature selection removes irrelevant features.

Example: Removing ID columns that don't affect prediction.

Purpose:

- Reduce complexity
- Improve accuracy
- Speed up training

8. Dimensionality Reduction (PCA)

When there are too many features, training becomes slow.

PCA (Principal Component Analysis) reduces features while keeping important information.

Purpose:

- Reduce dataset size
- Avoid overfitting
- Improve performance

9. Data Sampling

Big datasets may be extremely large.

Sampling selects a smaller portion of data such as:

- **Random sampling:** Random sampling means selecting data points **completely at random** from the dataset. Every record has an **equal chance** of being chosen.
- **Stratified sampling:** Stratified sampling divides the dataset into **groups** based on categories and then samples from each group.This ensures all categories are properly represented

Purpose: Reduce computation while maintaining patterns.

10. Data Splitting (Train/Test/Validation)

Data is divided into:

- **Training set** → used to train the model

- **Validation set** → used to tune hyperparameters
- **Testing set** → used to check final performance

Common split: 80% train, 20% test

Purpose: Evaluate model properly and avoid bias.

What is an ML Pipeline?

An **ML Pipeline** is a structured sequence of steps used to build, train, and deploy a machine learning model efficiently.

It automates the complete workflow from raw data to prediction.

Why ML Pipelines are Needed?

Machine learning involves many repeated tasks such as:

- Data cleaning
- Feature engineering
- Model training
- Evaluation

ML pipelines help by providing:

- Automation
  Consistency
  Scalability
  Easy model deployment

Steps in ML Pipeline

1. Data Ingestion

Data ingestion is the first step where data is collected from different sources into ML systems.

- **Sources include:**
  - Databases
  - Sensors and IoT devices
  - Application logs
  - Streaming platforms (Kafka, Spark Streaming)

Purpose: Bring raw data into the ML system.

2. Data Preprocessing

Raw data is usually messy and cannot be directly used for ML models.

- **Includes:**
  - Data cleaning (removing duplicates, correcting errors)
  - Handling missing values
  - Encoding categorical values (Male/Female → 0/1)
  - Feature scaling (normalization/standardization)

Purpose: Improve data quality and prepare it for modeling.

3. Feature Engineering

Feature engineering is the process of selecting and creating important input features that improve model performance.

- **Includes:**
  - Feature selection (removing irrelevant features)
  - Feature extraction (creating new meaningful features)
  - Dimensionality reduction (PCA)

Purpose: Make the dataset more informative and reduce complexity.

4. Model Training

In this step, machine learning algorithms are applied to the training data to learn patterns.

- **Algorithms include:**
  - Logistic Regression
  - Decision Trees
  - Random Forest
  - K-Means clustering

Purpose: Build a predictive model from data.

5. Model Evaluation

After training, the model must be tested to check how well it performs.

- **Common evaluation metrics:**
  - Accuracy
  - Precision
  - Recall
  - RMSE (for regression)

Purpose: Ensure the model is reliable and accurate.

6. Model Deployment

Deployment means making the trained model available for real-world use.

- **Used for:**
  - Real-time predictions
  - Business decision systems
  - Production applications

Purpose: Apply the model to new incoming data.

Supervised Learning: Classification and Regression on Large datasets

Machine learning

- ✓ Machine learning is the branch of Artificial Intelligence that focuses on developing models and algorithms that let computers learn from data and improve from previous experience without being explicitly programmed for every task.
- ✓ In simple words, ML teaches the systems to think and understand like humans by learning from the data.

Types of Machine Learning

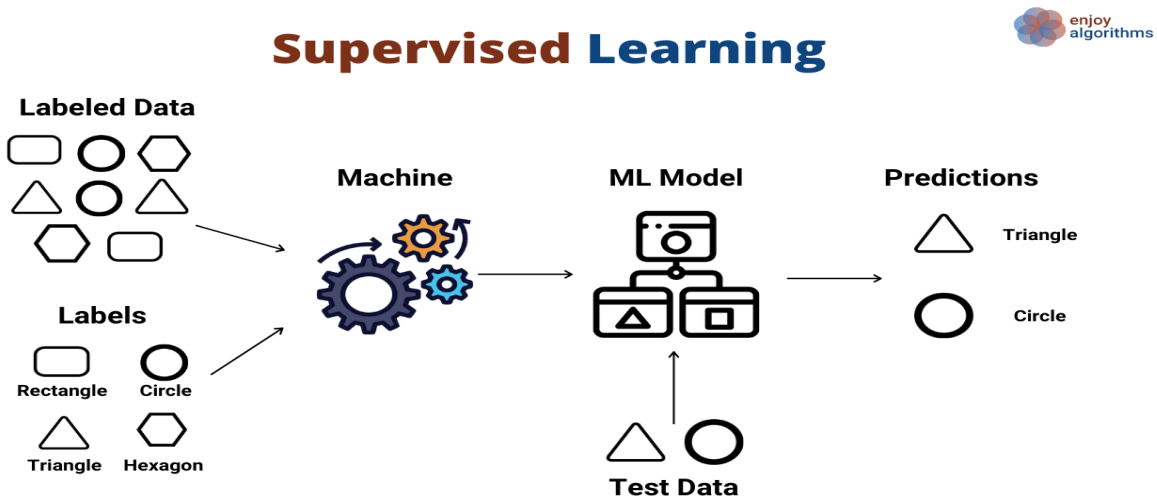
There are several types of machine learning, each with special characteristics and applications. Some of the main types of machine learning algorithms are as follows:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Additionally, there is a more specific category called semi-supervised learning, which combines elements of both supervised and unsupervised learning.

1. Supervised Learning

Supervised learning is defined as when a model gets trained on a **"Labelled Dataset"**. Labelled datasets have both input and output parameters. In **Supervised Learning** algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled.



There are two main categories of supervised learning that are mentioned below:

- ✓ Classification
- ✓ Regression

Classification

Classification deals with predicting **categorical** target variables, which represent discrete classes or labels. For instance, classifying emails as spam or not spam, or predicting whether a patient has a high risk of heart disease. Classification algorithms learn to map the input features to one of the predefined classes.

Here are some classification algorithms:

- **Logistic Regression**
- **Support Vector Machine**
- **Random Forest**
- **Decision Tree**
- **K-Nearest Neighbors (KNN)**
- **Naive Bayes**

Classification in Big Data

Classification is a supervised learning task where the model predicts a **category or class label** (such as spam/not spam, fraud/not fraud).

When datasets become very large (Big Data), traditional machine learning methods face limitations. Therefore, classification in Big Data requires the following:

1. Distributed Computing

Big Data is too large to fit on a single machine.

So, data and computation are distributed across multiple computers (nodes) in a cluster.

Benefit: Handles massive datasets efficiently.

Example: Apache Spark distributes data across cluster nodes.

2. Parallel Processing

Training classification models on large datasets takes a lot of time.

Parallel processing allows multiple parts of the dataset to be processed at the same time.

Benefit: Faster model training and prediction.

3. Scalable Algorithms

Big Data classification requires algorithms that can scale with increasing data size.

Scalable algorithms are designed to work efficiently even with:

- Millions of records
- High-dimensional features

Examples in Spark MLlib:

- Logistic Regression
- Random Forest
- Decision Trees

✅ Benefit: Reliable performance on large datasets.

Regression

Regression, on the other hand, deals with predicting **continuous** target variables, which represent numerical values. For example, predicting the price of a house based on its size, location, and amenities, or forecasting the sales of a product. Regression algorithms learn to map the input features to a continuous numerical value.

Here are some regression algorithms:

- Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression
- Decision tree

Regression in Big Data

When regression is applied to very large datasets (Big Data), it becomes more challenging because data may contain millions or billions of records.

Large-scale regression requires:

1. Efficient Data Preprocessing

Before training regression models, the data must be prepared properly:

- Cleaning noisy data
- Handling missing values
- Scaling numerical features
- Encoding categorical variables

Purpose: Improve accuracy and reduce computation errors.

2. Distributed Training

Big Data cannot be processed on a single machine.

Regression models must be trained using **distributed computing**, where:

- Data is divided across multiple nodes
- Computation happens in parallel

Purpose: Faster training on huge datasets.

3. Handling Millions of Records

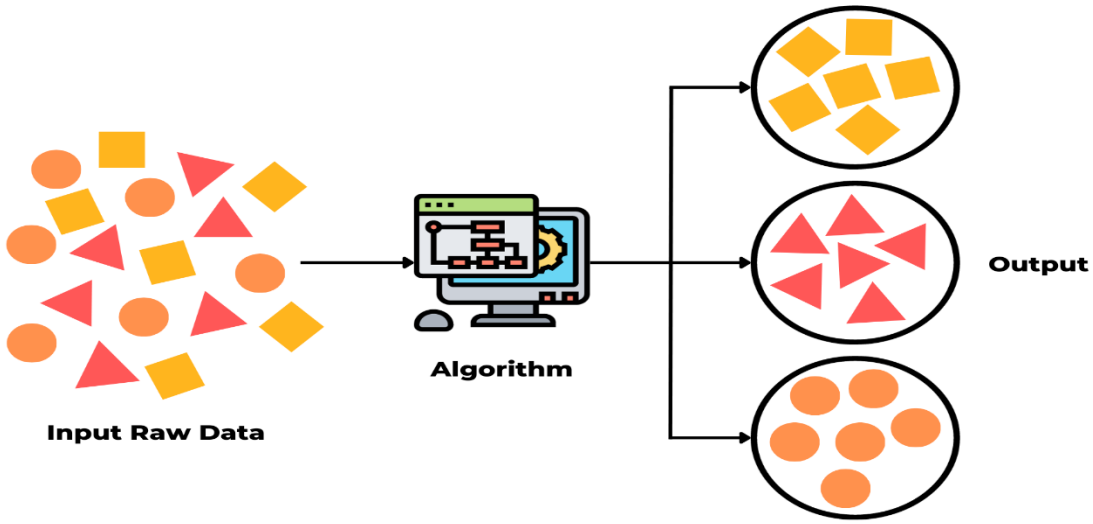
Regression on large datasets requires algorithms that can handle:

- Massive volume of data
- High-dimensional features
- Long training times

Scalable systems like Spark make this possible.

Unsupervised Learning:

- Unsupervised Learning Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabeled data.
- Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labeled target outputs.
- The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.



There are two main categories of unsupervised learning that are mentioned below:

- Clustering
- Association

Clustering

Clustering is the process of grouping similar data points into clusters.

Objects in the same cluster are more similar to each other than to those in other clusters.

Here are some clustering algorithms:

K-Means Clustering algorithm

Mean-shift algorithm

DBSCAN Algorithm

Principal Component Analysis

Independent Component Analysis

Clustering in Big Data

- Clustering is an unsupervised learning technique used to group similar data points into clusters.
- When datasets become very large (Big Data), traditional clustering methods face difficulties. Therefore, clustering in Big Data requires algorithms that are:

1. Scalable

- Big Data contains millions or billions of records.
- So clustering algorithms must be able to scale efficiently as data size increases.
- Benefit: Works well even with huge datasets.

2. Distributed

- Big datasets cannot be stored or processed on a single machine.
- Clustering must be performed using distributed computing, where data is divided across multiple nodes in a cluster.
- Example: Spark MLlib supports distributed K-Means clustering.
- Benefit: Handles large-scale data across clusters.

3. Fast

- Clustering on large datasets requires high computational power.
- Algorithms must run quickly using parallel processing to reduce training time.
- Benefit: Faster grouping and real-time analytics.

Dimensionality Reduction

Dimensionality Reduction is the process of reducing the number of input features (dimensions) in a dataset while preserving important information.

Big Data often contains too many features, making learning difficult.

Why Dimensionality Reduction?

- Reduces computation time
- Removes redundant features
- Avoids overfitting
- Improves visualization

Why Dimensionality Reduction is Needed in Big Data?

Big Data usually suffers from:

- Too many features
- High computation cost
- Slow model training
- Overfitting
- Difficulty in visualization

So dimensionality reduction helps by simplifying the dataset.

Importance in Big Data

Dimensionality reduction is important because it:

- Reduces storage and processing time
- Improves machine learning performance
- Removes redundant and irrelevant features
- Helps handle high-dimensional data efficiently

- Makes models faster and more scalable

Common Techniques Used for Dimensionality Reduction

Dimensionality reduction is used to reduce the number of features in large datasets while keeping important information.

1. PCA (Principal Component Analysis)

- The most widely used technique.
- Converts many correlated features into fewer uncorrelated components called **principal components**.
- Retains maximum variance (information).

Used for: High-dimensional numerical data

2. Feature Selection

- Selects only the most important features.
- Removes irrelevant or redundant columns.

Methods include:

- Chi-square test
- Information gain
- Feature importance (Random Forest)

Used for: Reducing dataset complexity

3. LDA (Linear Discriminant Analysis)

- Reduces dimensions while maximizing class separability.
- Mostly used in classification problems.

Used for: Supervised dimensionality reduction

Model Evaluation and Validation Techniques

Model Evaluation (Machine Learning)

Model Evaluation means checking **how well a machine learning model performs** on unseen (new) data.

It tells us:

- Is the model accurate?
- Is it overfitting?
- Will it work in real-world cases?

Why Model Evaluation is Needed?

Because a model may perform well on training data but fail on new data.

Example:

- Training Accuracy = 98%
- Testing Accuracy = 65%

👉 This is called **Overfitting**

So evaluation ensures:

- Generalization
- Reliability
- Best model selection

Common Evaluation Methods

1. Train–Test Split

Dataset is divided into:

- Training set (learns)
- Test set (evaluates)

Example:

- 80% Train
- 20% Test

What is cross-validation?

- ✓ **Cross-validation** is a technique for evaluating a machine learning model and testing its performance.
- ✓ CV is commonly used in applied ML tasks. It helps to compare and select an appropriate model for the specific predictive modeling problem.
- ✓ CV is easy to understand, easy to implement, and it tends to have a lower bias than other methods used to count the model's efficiency scores.
- ✓ There are a lot of different techniques that may be used to **cross-validate** a model. Still, all of them have a similar algorithm: ex: Hold out CV, K-fold CV etc.,

Hold-Out Cross-validation

- **Hold-out cross-validation** is the simplest and most common technique. You might not know that it is a **hold-out** method but you certainly use it every day.
- The algorithm of hold-out technique:
- Divide the dataset into two parts: the training set and the test set. Usually, 80% of the dataset goes to the training set and 20% to the test set but you may choose any splitting that suits you better.
- Train the model on the training set
- Validate on the test set
- Save the result of the validation

f

Why use cross validation

✓Helps **reduce overfitting** by ensuring the model performs well across different subsets of the data.

✓More **reliable** performance estimate compared to a single train/test split.

✓Useful when the dataset is **not very large**.

✓Allows you to compare models or configurations more **fairly**.

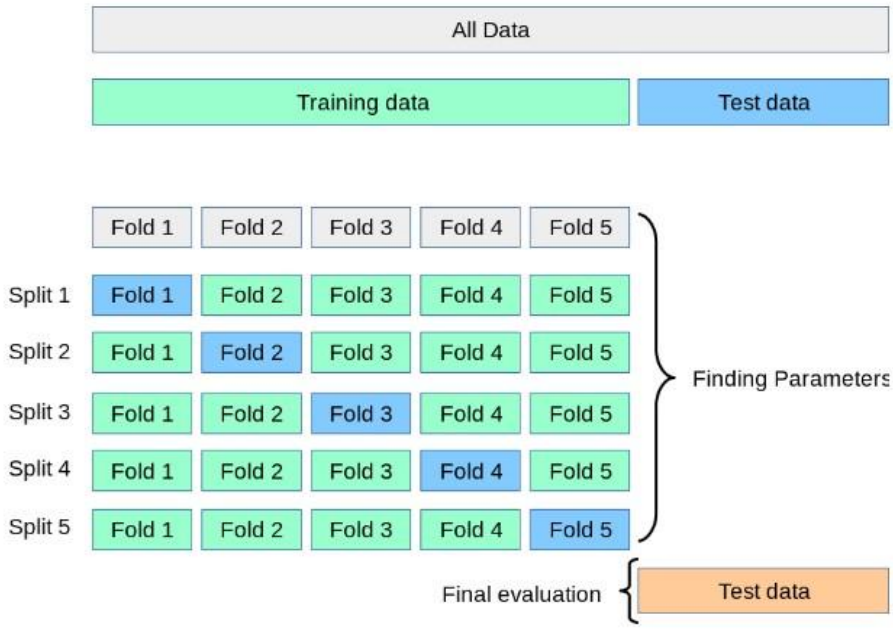
Algorithm

1. Divide the dataset into two parts: one for training, other for testing
2. Train the model on the training set
3. Validate the model on the test set
4. Repeat 1-3 steps a couple of times. This number depends on the C-V method that you are using.
5. End

k-Fold cross-validation

- **k-Fold cross-validation** is a technique that minimizes the disadvantages of the hold-out method. k-Fold introduces a new way of splitting the dataset which helps to overcome the "test only once bottleneck".
- The algorithm of the k-Fold technique:

1. Pick a number of folds – k. Usually, k is 5 or 10 but you can choose any number which is less than the dataset's length.
2. Split the dataset into k equal (if possible) parts (they are called folds)
3. Choose k – 1 folds as the training set. The remaining fold will be the test set
4. Train the model on the training set. On each iteration of cross-validation, you must train a new model independently of the model trained on the previous iteration
5. Validate on the test set
6. Save the result of the validation
7. Repeat steps 3 – 6 k times. Each time use the remaining fold as the test set. In the end, you should have validated the model on every fold that you have.
8. To get the final score average the results that you got on step 6.



Evaluation Metrics

A) Classification Metrics

1. Accuracy

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

.2Precision

$$Precision = \frac{TP}{TP + FP}$$

.3Recall

$$Recall = \frac{TP}{TP + FN}$$

.4F1 Score

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

B) Regression Metrics

1. MAE

$$MAE = \frac{1}{n}\sum |\, y - \hat{y}\, |$$

.2MSE

3. RMSE

$$RMSE = \sqrt{MSE}$$

.4R^2 Score

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

A) Classification Metrics (Example)

Actual Positive TP = 40 FN = 10

Actual Negative FP = 5 TN = 45

So:

TP = 40

TN = 45

FP = 5

FN = 10

Total = TP + TN + FP + FN

= 40 + 45 + 5 + 10

= 100

1. Accuracy

Accuracy = Correct Predictions / Total Predictions

Correct predictions = TP + TN = 40 + 45 = 85

Accuracy = 85 / 100 = 0.85 = 85%

2. Precision

Precision = TP / (TP + FP)

Precision = 40 / (40 + 5)

= 40 / 45

= 0.888

Precision ≈ 88.8%

3. Recall

Recall = TP / (TP + FN)

Recall = 40 / (40 + 10)

= 40 / 50

= 0.80

Recall = 80%

4. F1 Score

F1 = 2 × (Precision × Recall) / (Precision + Recall)

Precision = 0.888

Recall = 0.80

F1 = 2 × (0.888 × 0.80) / (0.888 + 0.80)

= 2 × 0.710 / 1.688

= 0.84

F1 Score ≈ 84%

B) Regression Metrics (Example)

Suppose we have:

Actual (y)	Predicted (ŷ)
10	12
20	18
30	33
40	39

Number of samples:

n = 4

1. MAE (Mean Absolute Error)

$$MAE = (1/n) \Sigma |y - \hat{y}|$$

Absolute errors:

y	$\hat{y}$	$\lvert y-\hat{y}\rvert$
10	12	2
20	18	2
30	33	3
40	39	1

Sum = 2 + 2 + 3 + 1 = 8

MAE = 8 / 4 = 2

2. **MSE (Mean Squared Error)**

$$MSE = (1/n)\ \Sigma\ (y - \hat{y})^2$$

Square errors:

Error	$Error^2$
2	4
2	4
3	9
1	1

Sum = 4 + 4 + 9 + 1 = 18

MSE = 18 / 4 = 4.5

3. RMSE

$$RMSE = \sqrt{MSE}$$

$$RMSE = \sqrt{4.5} = 2.12$$

4. R^2 Score (Simple Meaning)

R^2 tells how well model explains variance.

$R^2 = 1 \rightarrow$ Perfect model

$R^2 = 0 \rightarrow$ Poor model

$R^2 < 0 \rightarrow$ Worse than average

Metric	Example Result
Accuracy	85%
Precision	88.8%
Recall	80%
F1 Score	84%
MAE	2
MSE	4.5
RMSE	2.12

Techniques to Prevent Overfitting

Regularization

- L1 Regularization
- L2 Regularization

Early Stopping

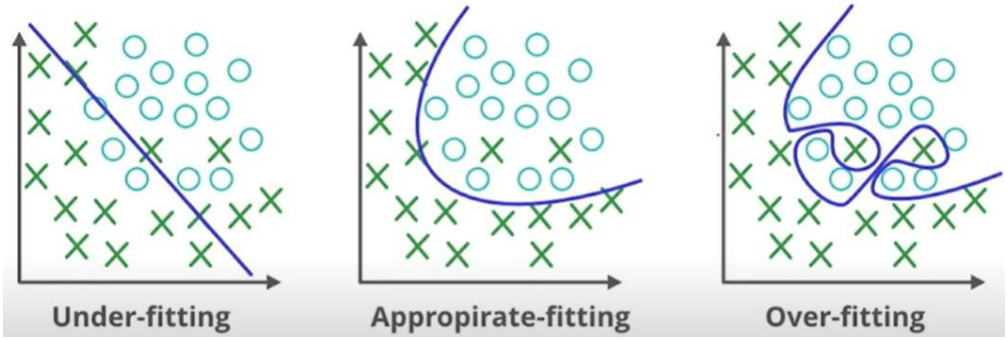
Stop training when validation error increases.

Dropout (Deep Learning)

Randomly drops neurons during training.

Regularization

- Overfitting and underfitting are common challenges in neural networks.
- Overfitting occurs when a model performs very well on training data but fails to generalize to new or unseen data, while underfitting happens when a model performs poorly on both training and validation datasets.



Regularization in deep Learning

- Regularization is a technique used to reduce errors by fitting the function appropriately on the given training set and avoiding overfitting.
- The commonly used techniques are:

1. Lasso Regularization – L1 regularization
2. Ridge Regularization – L2 regularization

Lasso Regularization

- A regression model which uses the L1 regularization technique is called LASSO (**Least Absolute Shrinkage and Selection Operator**) regression
- **L1 Regularization** is a technique used to **reduce overfitting** in neural networks by **adding a penalty** to the loss function during training. This penalty is the **sum of the absolute values** of the weights in the model.
- In each training step, the model tries to **minimize the total loss**.
- Because large weights increase the loss (due to the penalty), the optimizer **shrinks the weights**.
- Some weights become **exactly 0** — meaning those neurons or input features are **ignored**.
- This makes the model **simpler and more efficient**.

- Let's say the original loss function (like Mean Squared Error or Cross-Entropy) is:
- **Loss = L(y_true, y_pred)**
- With **L1 Regularization**, the new loss becomes:
-
- **Loss = L(y_true, y_pred) + λ * Σ |w|**
- **Where:**
- L(y_true, y_pred) = original loss (error between actual and predicted)
- λ (lambda) = regularization strength (a small positive number, like 0.01)
- Σ |w| = sum of absolute values of all weights in the network

Ridge Regression

- A regression model that uses the L2 regularization technique is called Ridge regression.
- **L2 Regularization** is a technique used to **prevent overfitting** by discouraging the model from having **large weight values**.
- It adds a **penalty** to the loss function based on the **square of the weights**.

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} w_i^2$$

- where
- 'm' number of feature
- 'n' number of examples
- Y_i actual target value
- $Y_i\hat{}$ predicted target value
- 'w' weight

$$\text{Loss} = L(y_{\text{true}}, y_{\text{pred}}) + \lambda \cdot \sum w^2$$

Where:

Term	Meaning
$L(y_{\text{true}}, y_{\text{pred}})$	Original loss (like MSE or Cross-Entropy)

Let's say:

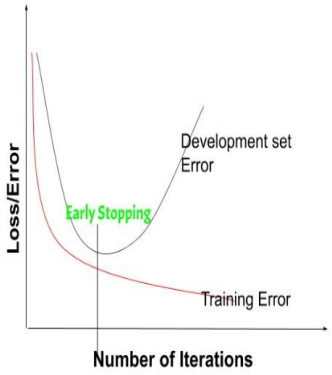
- Weights: $w = [2.0, -1.5, 0.0]$
- Then: $\sum w^2 = 2.0^2 + (-1.5)^2 + 0.0^2 = 4.0 + 2.25 + 0 = 6.25$
- Lambda (λ) = 0.01
- Original Loss (e.g., MSE) = 0.25

Then:

$$\text{Total Loss} = 0.25 + 0.01 \cdot 6.25 = 0.25 + 0.0625 = 0.3125$$

Early Stopping

- ❖ Early stopping is a smart trick to **stop training at the right time**, before overfitting happens
- ❖ A problem with training neural networks is in the choice of the number of training epochs to use.
- ❖ Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model.

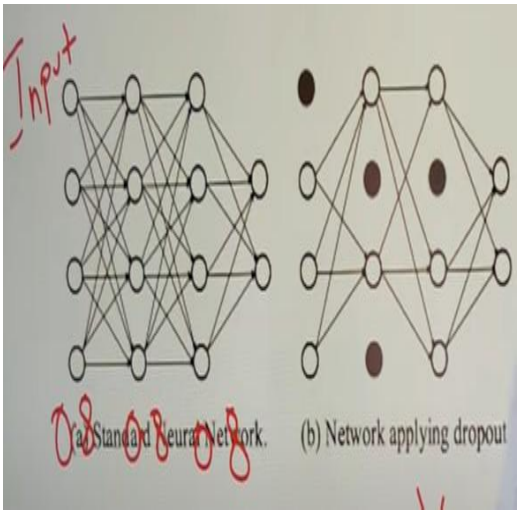


How It Works (Step-by-Step):

- Train your model with **many epochs** (say 100 or 200).
- After **each epoch**, check how the model performs on the **validation dataset**.
- Keep track of **validation error** (or loss).
- If the validation error **starts increasing**, it means the model is **starting to overfit**.
- Stop training at the point where validation error is **lowest**.

Drop Out Regularization

- ❖ Dropout regularization is a widely used technique in deep learning to prevent overfitting in neural networks.
- ❖ It works by randomly "dropping out," or setting to zero, a fraction of the units (i.e., neurons) in the neural network during training.
- ❖ This means that at each training iteration, certain neurons are temporarily removed from the network with a probability defined by the dropout rate.



TEXT BOOKS:

1. Big Data: Principles and Paradigms by Rajkumar Buyya, Rodrigo N. Calheiros, Amir Vahid Dastjerdi – Wiley
2. Learning Spark: Lightning-Fast Big Data Analysis by Jules S. Damji et al. – O'Reilly Data Science and Big Data Analytics by EMC Education Services – Wiley

REFERENCE BOOKS:

1. Designing Data-Intensive Applications by Martin Kleppmann – O'Reilly
2. Machine Learning with Spark by Rajdeep Dua, Tathagata Das – Packt Publishing
3. Streaming Systems by Tyler Akidau – O'Reilly Media.
4. 4. Artificial Intelligence for Big Data by Anand Deshpande

REFERENCE WEBSITE:

1. https://www.coursera.org/specializations/big-data – Coursera Big Data Specialization
2. https://spark.apache.org/docs/latest/ – Apache Spark Documentation 3. https://www.edx.org/course/big-data-analysis-with-python – edX
3. https://www.udacity.com/course/ai-for-business-leaders--nd088 – Udacity AI for Business
4. https://www.kaggle.com/learn/intro-to-machine-learning – Kaggle ML Tutorials
5. https://data-flair.training/blogs/apache-spark-tutorial/ – Spark Tutorials

UNIT IV

AI APPLICATIONS ON BIGDATA

Predictive Maintenance using Big Data & AI, Fraud Detection in Banking with Machine Learning, AI in Healthcare: Diagnosis, Genomics, Patient Monitoring, Retail and E- commerce Analytics, AI for Smart Cities and IoT Sensor Data Analysis, Evaluation of Real- Time AI Applications on Streaming Data.

Predictive Maintenance using Big Data & AI

- Predictive Maintenance is an advanced maintenance strategy that uses **Big Data analytics** and **Artificial Intelligence (AI)** to predict equipment failures before they happen.
- Instead of performing maintenance after breakdown (reactive) or at fixed intervals (preventive), predictive maintenance ensures maintenance is done **only when required**.

Role of Big Data in Predictive Maintenance

Modern industries generate huge amounts of machine data through:

- IoT sensors
- Embedded monitoring devices
- Industrial machines
- Production systems

This data is characterized by:

High Volume (large sensor readings)

High Velocity (real-time streaming)

High Variety (structured + unstructured data)

Big Data platforms like **Hadoop, Spark, Kafka** are used to store and process this continuous flow.

How AI is Used in Predictive Maintenance

AI models analyze historical and real-time machine data to detect patterns that indicate future failure.

AI Techniques Used

1.Machine Learning Models

1. Logistic Regression
2. Random Forest
3. Support Vector Machines (SVM)

2.Deep Learning Models

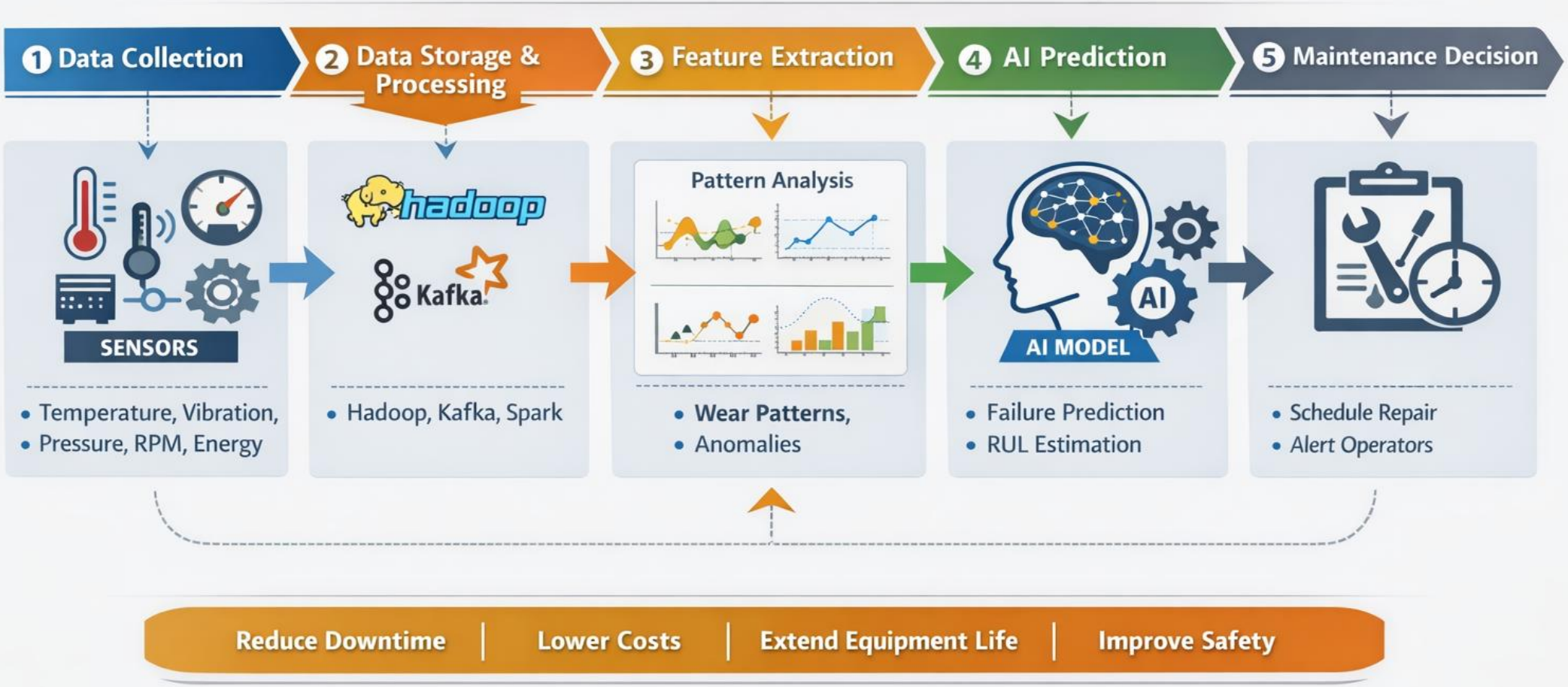
1. LSTM networks for time-series prediction
2. Autoencoders for anomaly detection

3.Anomaly Detection

1. Detect unusual vibration, temperature, or pressure changes.

Predictive Maintenance Workflow

Big Data & AI



Working Process of Predictive Maintenance

Predictive Maintenance uses **Big Data and AI** to monitor machines continuously and predict failures before they occur.

The complete working process is explained in the following steps:

Step 1: Data Collection

Industrial machines are equipped with **IoT sensors** that collect real-time operational parameters such as:

- Temperature
- Vibration
- Pressure
- RPM (Rotations per Minute)
- Energy consumption

This sensor data provides the raw input for predictive analysis.

Step 2: Data Storage and Processing

Since data is generated continuously and in large volumes, **Big Data platforms** are used to store and process it efficiently:

- Hadoop Distributed File System (HDFS) for large-scale storage
- Spark Streaming for real-time processing
- Kafka pipelines for streaming sensor data transfer
- These tools ensure scalable and fast handling of machine data.

Step 3: Feature Extraction

From the collected raw data, meaningful patterns and indicators are extracted, such as:

- Machine wear rate
- Sudden fluctuations in vibration or temperature
- Performance degradation over time

Feature extraction helps in identifying early warning signs of failure.

Step 4: AI Prediction

AI and Machine Learning models analyze the extracted features to predict:

- Remaining Useful Life (RUL) of equipment
- Probability of machine failure
- Maintenance scheduling requirements

Techniques like Random Forest, LSTM, and anomaly detection are commonly used.

Step 5: Maintenance Decision

Based on AI predictions, maintenance actions are automatically planned:

- Schedule repair before breakdown
- Send alerts to operators
- Replace components proactively

This prevents unexpected failures and improves operational efficiency.

Applications of Predictive Maintenance

Predictive Maintenance is widely used in many industries where continuous machine operation is critical.

Major applications include:

Manufacturing Industries (Smart Factories)

Used to monitor industrial machines and avoid sudden production stoppages.

Aircraft Engine Monitoring

Helps detect early faults in engines to ensure passenger safety and reduce maintenance costs.

Railway Systems

Applied for tracking the condition of train engines, tracks, and braking systems.

Power Plants and Turbines

Used for predicting failures in turbines, generators, and other heavy equipment.

Automotive Production Units

Ensures smooth functioning of robotic assembly lines and reduces equipment downtime.

Advantages of Predictive Maintenance

Predictive Maintenance provides several important benefits:

- **Reduces Unexpected Breakdowns**

Failures are predicted early, preventing sudden machine stoppage.

- **Minimizes Downtime**

Maintenance is planned efficiently without affecting production schedules.

- **Saves Maintenance Costs**

Repairs are done only when needed, avoiding unnecessary servicing.

- **Improves Machine Lifespan**

Early fault detection increases equipment durability and performance.

- **Enhances Safety and Productivity**

Prevents hazardous failures and ensures uninterrupted industrial operations.

Fraud Detection in Banking with Machine Learning

1. Introduction

Fraud detection in banking refers to identifying suspicious or illegal financial activities such as:

- Credit card fraud
- Online transaction fraud
- Identity theft
- Money laundering

With the growth of digital payments, fraud cases are increasing rapidly. Machine Learning (ML) helps banks detect fraud automatically and in real time.

2. Why Machine Learning for Fraud Detection?

Earlier, banks mainly used **traditional rule-based fraud detection systems**.

These systems worked with fixed rules such as:

- "Block transactions above ₹50,000"
- "Block transactions if the location changes suddenly"
- "Block if too many transactions occur in a short time"

Role of Machine Learning in Fraud Detection

Machine Learning overcomes these issues by learning patterns automatically from transaction data.
Machine Learning provides:

i. **Adaptive Learning**

   ML models continuously improve by learning from new fraud cases and changing fraud behaviors.

ii. **Pattern Recognition**

   ML can identify hidden and complex fraud patterns that are difficult to capture with manual rules.

iii. **Real-Time Fraud Detection**

   ML systems can analyze transactions instantly and detect fraud during the transaction process.

iv. **Reduced False Alarms**

   Unlike rigid rules, ML improves accuracy and reduces false positives (genuine transactions marked as fraud).

3. Working of Fraud Detection System

Step 1: Data Collection

Banks collect transaction data such as:

- Amount
- Time of transaction
- Merchant type
- Customer location
- Device/IP address
- Transaction history

Step 2: Data Preprocessing

Raw banking data needs cleaning:

- Handling missing values
- Normalization of transaction amount
- Encoding categorical data (merchant type, country)
- Feature scaling

Step 3: Feature Engineering

Important fraud indicators include:

- High transaction frequency
- Sudden large spending
- Transactions in unusual countries
- Multiple failed login attempts

Step 4: Model Training

ML algorithms learn fraud patterns using labeled transaction datasets.

4. Machine Learning Techniques Used

A) Supervised Learning Used when transactions are labeled as:

- Fraud (1)
- Genuine (0)
- Algorithms:
- Logistic Regression
- Decision Trees
- Random Forest
- Gradient Boosting (XGBoost, LightGBM)
- Neural Networks

B) Unsupervised Learning

Used when fraud labels are not available.

Algorithms:

- Clustering (K-Means, DBSCAN)
- Isolation Forest
- Autoencoders
- Goal: Detect anomalies in transaction behavior.

C) Deep Learning Models

Used for complex fraud patterns:

- LSTM (for sequential transaction data)
- CNN (for feature extraction)
- Autoencoders (for anomaly detection)

5. Challenges in Fraud Detection

Fraud detection in banking faces several challenges:

1.Imbalanced Data – Fraud cases are very rare compared to genuine transactions, making model training difficult.

2.False Positives – Genuine transactions may be wrongly flagged as fraud, affecting customer trust.

3.Changing Fraud Patterns – Fraudsters constantly evolve new techniques, so models must be updated regularly.

4.Real-Time Detection – Transactions must be analyzed instantly, requiring fast and efficient systems.

5.Data Privacy – Banking data is sensitive and must follow strict security and regulatory rules.

AI in Healthcare: Diagnosis, Genomics, and Patient Monitoring

Artificial Intelligence (AI) is revolutionizing the healthcare industry by improving disease detection, enabling personalized treatments, and supporting continuous patient care.

AI applications in healthcare mainly focus on diagnosis, genomics, and real-time patient monitoring.

1. AI in Healthcare: Diagnosis

AI is transforming medical diagnosis by helping doctors detect diseases faster and more accurately.

Key Uses

Medical Imaging

AI systems analyze medical images such as:

- X-rays
- MRI scans
- CT scans
- They help detect tumors, fractures, infections, and abnormalities.

Example:

AI-assisted breast cancer screening improves early tumor detection.

2. Disease Prediction

AI models predict diseases by analyzing patient health data and risk factors.
Common predictions include:

- Diabetes
- Heart disease
- Stroke risk

This supports preventive healthcare and early intervention.

3. Clinical Decision Support

AI provides treatment suggestions based on:

- Patient medical history
- Symptoms
- Previous clinical cases

This improves diagnostic accuracy and reduces errors.

Benefits

- Early detection of diseases
- Reduced human error
- Faster and more accurate diagnosis

2. AI in Genomics

Genomics is the study of genes and DNA sequences. AI helps analyze massive genomic datasets to understand diseases at a genetic level.

Applications of AI in Genomics

1. Gene Sequencing Analysis

AI identifies genetic mutations linked to diseases such as cancer or inherited disorders.

2. Personalized Medicine

AI enables treatments tailored to an individual's genetic profile, improving effectiveness.

3. Drug Discovery

AI predicts how genes respond to certain drugs, speeding up the development of new medicines.

Techniques Used

- Deep Learning for DNA pattern recognition
- Machine Learning classifiers for disease risk prediction

3. AI for Patient Monitoring

AI supports continuous monitoring of patients using wearable devices and real-time sensors.

Examples

- Smartwatches tracking heart rate
- Oxygen level monitoring devices
- ICU monitoring systems

Applications

1. Remote Patient Monitoring

Patients can be monitored from home, reducing hospital visits.

2. Early Warning Systems

AI can detect critical health conditions early, such as:

- Cardiac arrest risk
- Seizure prediction
- Sudden blood pressure changes

3. Chronic Disease Management

AI helps manage long-term illnesses like:

- Diabetes,Hypertension,Asthma

Retail and E-Commerce Analytics

- Retail and E-commerce Analytics refers to the application of **Big Data technologies and Artificial Intelligence (AI)** to analyze customer activities, enhance sales performance, and improve overall business decision-making in online and offline retail systems.
- It helps organizations understand customer preferences, optimize operations, and deliver personalized shopping experiences.

Key Areas of Retail and E-Commerce Analytics

1. Customer Segmentation

Customer segmentation involves dividing customers into different groups based on:

- Purchase history
- Age and gender
- Geographic location
- Interests and browsing behavior

This helps retailers provide customized offers and marketing strategies.

2. Recommendation Systems

Recommendation systems suggest products to users based on their preferences.

Examples include:

•Amazon product recommendations

•Flipkart personalized suggestions

AI models analyze customer behavior and purchase patterns to recommend relevant items.

3. Demand Forecasting

Demand forecasting predicts future product demand using:

- Historical sales data
- Seasonal trends
- Time-series analysis

It helps businesses maintain the right stock levels and avoid losses.

4. Dynamic Pricing

Dynamic pricing uses AI algorithms to adjust product prices automatically based on:

- Market competition
- Customer demand
- Festival seasons
- Stock availability

This improves revenue and competitiveness.

5. Churn Prediction

Churn prediction identifies customers who are likely to stop shopping or unsubscribe.

AI models detect churn based on:

- Reduced activity
- Negative feedback
- Purchase frequency changes

This helps companies retain customers through special offers.

6. Inventory and Supply Chain Optimization

Analytics helps optimize inventory management by preventing:

- Overstocking
- Understocking

Predictive models ensure smooth supply chain operations and timely delivery.

AI for Smart Cities and IoT Sensor Data Analysis

AI for Smart Cities refers to the use of **Artificial Intelligence (AI)** along with **Internet of Things (IoT) sensor networks** to improve the quality of urban life by enabling intelligent monitoring, automation, and efficient resource management.

Smart cities generate massive amounts of real-time data from IoT devices, and AI helps analyze this data to support better decision-making.

IoT Sensor Data in Smart Cities

Smart cities rely on different types of sensors that continuously collect data from the environment, such as:

- Traffic and vehicle sensors
- Smart CCTV surveillance cameras
- Air pollution and weather monitoring sensors
- Smart water and electricity meters
- Waste management sensors
- Public safety and emergency sensors

These sensors produce large-scale streaming data that must be processed efficiently.

Role of AI in Smart City Analytics

AI techniques are applied to IoT sensor data to extract useful insights and enable intelligent services.

Key AI Techniques Used

1. Machine Learning for Prediction

Machine learning models are used for forecasting future events such as:

- Traffic congestion prediction
- Energy consumption forecasting
- Water usage estimation

2. Deep Learning for Image and Video Analytics

Deep learning is widely used in smart surveillance systems for:

- Accident detection
- Crowd monitoring
- Vehicle number plate recognition
- Crime prevention

3. Anomaly Detection

AI-based anomaly detection identifies unusual patterns in sensor data, such as:

- Gas leakage alerts
- Sudden power failures
- Abnormal pollution levels
- Unauthorized activity detection

4. Edge AI Computing

In smart cities, AI is often deployed at the edge (near the sensors) to enable:

- Faster decision-making
- Reduced network delay
- Real-time processing

This is critical for emergency and safety applications.

Major Applications of AI in Smart Cities

AI-powered smart city systems are used in:

- **Smart traffic management and control**
- **Environmental pollution monitoring**
- **Energy-efficient smart grids**
- **Disaster prediction and emergency response**
- **Intelligent public transportation systems**
- **Smart waste collection and management**

Evaluation of Real-Time AI Applications on Streaming Data

Real-time AI applications operate on **continuous, fast-moving streaming data** rather than fixed historical datasets. These systems must generate predictions or decisions instantly while data is still arriving. Examples include:

- Fraud detection in banking transactions
- Predictive maintenance using IoT sensors
- Real-time recommendation systems
- Smart traffic and surveillance monitoring
- Healthcare monitoring with wearable devices

Evaluating such systems is more complex because streaming data is **dynamic, high-volume, and time-sensitive**.

Key Evaluation Metrics and Criteria

1.Accuracy Metrics

Measures prediction performance using Accuracy, Precision, Recall, F1-score, RMSE.

2.Latency (Response Time)

Time taken to generate output after data arrival. Low latency is crucial.

3.Throughput

Number of streaming events processed per second.

4.Scalability

Ability to handle increasing data volume and velocity efficiently.

5.Concept Drift Handling

Detects and adapts to changing data patterns over time.

6.Fault Tolerance & Reliability

Ensures continuous operation even during failures using checkpointing and recovery.

7.Resource Utilization

Evaluates CPU, memory, GPU, and network efficiency.

8.Data Quality Handling

Manages noisy, missing, duplicated, or out-of-order streaming data.

UNIT–5:ADVANCED TOPICS AND CASE STUDIES

Deep Learning on Big Data using TensorFlow on Spark, Explainable AI (XAI) in Big Data Environments, Ethical Issues and Data Governance in Big Data AI, Edge Computing and AI for Low Latency Applications, Case Study 1: AI-Powered Big Data in Healthcare, Case Study 2: Big Data AI Solution in Smart Manufacturing.

Deep Learning on Big Data using TensorFlow on Spark

1. Why Deep Learning with Big Data?

Deep Learning models (CNNs, RNNs, Transformers, etc.) require:

- Huge training datasets
- High computation power (GPUs/TPUs)
- Distributed processing

But traditional deep learning frameworks struggle when data becomes **too large** to fit on a single machine.
That's where **Spark + TensorFlow** comes in.

What is TensorFlow?

TensorFlow is Google's deep learning framework used for:

- Neural networks training
- GPU acceleration
- Deep learning applications
- **It supports:**
    - CNNs (Image processing)
    - RNNs/LSTMs (Sequence data)
    - Transformers (NLP)

4. Why Combine TensorFlow with Spark?

Spark Strength	TensorFlow Strength
Big Data distributed processing	Deep Learning model training
Handles huge datasets	Handles neural networks
Data preprocessing pipelines	Backpropagation + optimization

Together they enable:
Distributed Deep Learning on Big Data

How TensorFlow Works on Spark?

TensorFlow works with Apache Spark to enable **deep learning on large-scale Big Data** by combining Spark's distributed data processing power with TensorFlow's deep learning capabilities

Role of Spark

Spark helps in handling Big Data efficiently by:

- **Data Loading**: Reading massive datasets from sources like HDFS, Hive, S3, and Kafka.
- **Data Transformation**: Performing preprocessing tasks such as cleaning, feature extraction, and normalization.
- **Distributed Training Support**: Splitting data across cluster nodes and running training tasks in parallel for faster computation.

Role of TensorFlow

TensorFlow contributes by performing deep learning operations such as:

•**Neural Network Computation**: Building and executing deep learning models like CNNs, RNNs, and DNNs.

•**Gradient Updates**: Applying backpropagation and optimization algorithms to update model weights.

•**Model Building and Training**: Designing, training, and saving scalable deep learning models using GPUs/TPUs.

Architecture: TensorFlow + Spark

The architecture of **TensorFlow + Spark** enables scalable deep learning on Big Data by combining Spark's distributed data processing with TensorFlow's neural network training capabilities.

1. Data Stored in Big Data Sources

Large datasets are stored in distributed storage systems such as:

- **HDFS (Hadoop Distributed File System)**
- **Hive Data Warehouse**
- **Amazon S3**
- **Kafka Streams (Real-time data)**

2. Spark Reads and Preprocesses Data

Apache Spark is used for distributed data processing tasks like:

- **Data cleaning**
- **Feature extraction**
- **Normalization and transformation**

Spark efficiently handles massive datasets across cluster nodes.

3. Data Conversion to TensorFlow Format

After preprocessing, Spark converts the data into TensorFlow-supported formats such as:

- **TFRecords**
- **NumPy Arrays**

This ensures the data can be fed into deep learning models.

4. Distributed Training is Performed

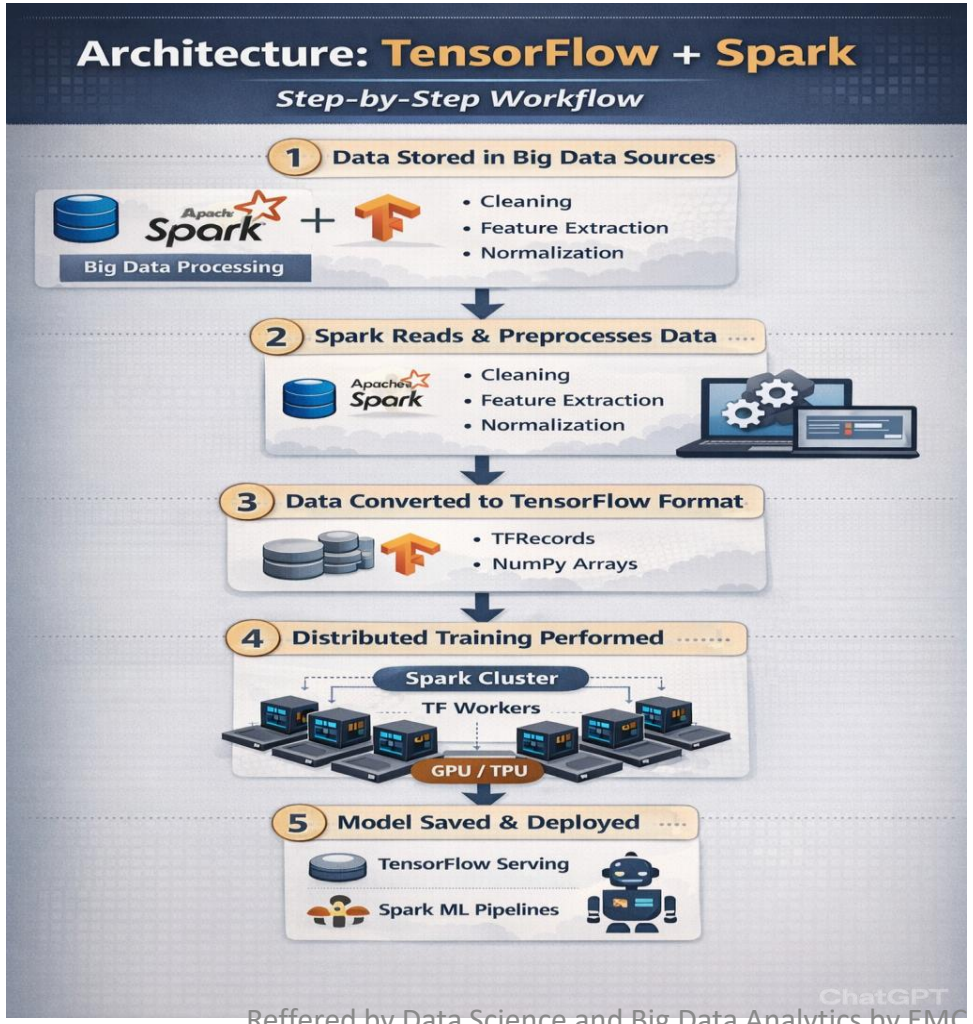
Training is executed in a distributed way where:

- Multiple Spark executors act as **TensorFlow workers**
- Each worker trains on a portion of the dataset
- Parallel training improves speed and scalability

5. Model Saving and Deployment

Once training is completed, the deep learning model is saved and deployed using:

- **TensorFlow Serving**
- Integration with Spark ML pipelines for production use



Explainable AI (XAI) in Big Data Environments

Explainable AI (XAI) refers to methods and techniques that make the decisions of Artificial Intelligence models **transparent, interpretable, and understandable** to humans.

In **Big Data environments**, AI models often work on:

- Massive datasets
- Distributed systems
- Complex deep learning architectures

So, explaining *why* a model made a decision becomes essential.

Why Explainability is Needed in Big Data AI?

Big Data AI models are often called **black-box models** because:

- They process huge volumes of data
- Use deep neural networks
- Produce outputs without clear reasoning

XAI helps to:

- Build trust
  Improve accountability
  Detect bias
  Meet legal requirements
  Support critical decision-making

Key Components of XAI in Big Data

1. Interpretability

Ability to understand the model structure and reasoning.

2. Transparency

Model decisions should be visible and auditable.

3. Trust and Fairness

Ensures AI decisions are unbiased and ethical.

5. XAI Techniques Used in Big Data

(a) Feature Importance Methods

Identify which input features contributed most.

Example:

- Income
- Transaction amount
- Location

(b) LIME (Local Interpretable Model-Agnostic Explanations)

Explains individual predictions locally.

Used for:

- Fraud detection
- Medical diagnosis

(c) SHAP (Shapley Additive Explanations)

Provides consistent explanations using game theory.

Works well for large-scale Big Data AI systems.

(d) Rule-Based Explanations

Generates human-readable rules:

"If transaction > ₹50,000 and location unusual → Fraud"

(e) Attention Visualization (Deep Learning)

Used in NLP and Transformers to show which parts of input mattered most.

6. Applications of XAI in Big Data

Healthcare

Explaining diagnosis predictions from patient Big Data.

Finance

Explaining loan approvals and fraud alerts.

Smart Cities

Explaining traffic prediction and surveillance AI decisions.

Manufacturing

Explaining predictive maintenance results.

7. Benefits of XAI in Big Data Environments

- Better trust in AI decisions
  Improved model debugging
  Compliance with regulations (GDPR, AI laws)
  Fairness and bias reduction
  Supports human-AI collaboration

Ethical Issues and Data Governance in Big Data AI

1. Introduction

Big Data AI systems use massive amounts of data to build intelligent models for:

- Healthcare
- Finance
- Smart cities
- Social media
- Manufacturing

However, using AI with Big Data raises serious **ethical concerns** and requires strong **data governance frameworks**.

Ethical Issues in Big Data AI

Ethical issues refer to problems related to fairness, privacy, trust, and responsible use of AI.

1. Privacy and Data Security

Big Data often contains sensitive information such as:

- personal details
- medical records
- financial transactions

Risks:

- Data leaks
- Unauthorized access
- Surveillance misuse

Example: Patient data being used without consent.

2. Bias and Discrimination

AI models trained on biased data can produce unfair results.

Examples:

- Loan rejection based on race/gender
- Hiring systems favoring certain groups
- Bias leads to unethical decision-making.

3. Lack of Transparency (Black-Box Models)

Deep learning models are complex and hard to interpret.

Problem:

- Users cannot understand why AI made a decision
- Reduces trust and accountability
- XAI is needed to improve explainability.

4. Accountability and Responsibility

When AI makes wrong decisions, it is unclear who is responsible:

- Developer?
- Organization?
- Data provider?

Example: Wrong medical diagnosis by AI system.

5. Misuse of AI and Surveillance

Big Data AI can be misused for:

- •Mass surveillance
- •Social manipulation
- •Fake news targeting

Ethical AI ensures responsible usage.

6. Data Ownership and Consent

Questions arise:

- •Who owns the collected data?
- •Was user consent taken?
- •Can data be reused for other purposes?

Proper consent is essential.

Data Governance in Big Data AI

Data Governance refers to the policies, standards, and practices that ensure data is:

- accurate
- secure
- ethical
- compliant

Key Elements of Data Governance

1.Data Quality Management

Ensures data is complete, consistent, and accurate. Poor data leads to wrong AI results.

2.Data Privacy and Protection

Uses encryption, access control, and anonymization to protect sensitive data.

3.Regulatory Compliance

Follows laws like GDPR, HIPAA, and Data Protection Acts to ensure legal data use.

4.Ethical AI Policies

Promotes fairness, transparency, accountability, and responsible AI deployment.

5.Audit and Monitoring

Continuously checks for bias, security breaches, and unethical AI decisions.

Edge Computing and AI for Low Latency Applications

1. Introduction

- **Edge Computing** refers to processing data **near the source of data generation** (edge devices) instead of sending everything to a centralized cloud.
- When combined with **Artificial Intelligence**, Edge Computing enables **fast, real-time intelligent decisions**.

2. Why Edge AI is Needed?

Traditional Cloud AI faces problems:

- High network delay
- Bandwidth limitations
- Privacy concerns
- Slow real-time response

Edge AI solves these by performing AI inference locally.

3. What is Low Latency?

Low latency refers to a **very small delay** between:

- Data input
- AI processing
- Output decision

It is essential for real-time applications.

Example: A self-driving car must detect an obstacle and brake instantly.

4. How Edge Computing + AI Works

Edge Computing with AI works by bringing intelligence closer to where data is produced, instead of relying completely on cloud servers.

Step-by-Step Process:

1.Data Generation at Edge Devices

Data is continuously produced by devices such as:

1. Sensors
2. CCTV cameras
3. Smartphones
4. IoT machines

2.Local Data Processing

Instead of sending raw data to the cloud, the edge device processes it locally.

3.AI Model Runs at the Edge

Pre-trained AI models are deployed on edge hardware to perform tasks like:

1. Object detection
2. Speech recognition
3. Fault prediction

Instant Decision Making

The AI system gives real-time output immediately, ensuring:

- Fast response
- Very low latency

Example: A car detects an obstacle and brakes instantly.

Reduced Cloud Dependency

Since most computation happens locally, the system does not depend heavily on internet connectivity.

Sending Only Important Data to Cloud

Only useful insights or summaries are transmitted to the cloud for:

- Storage
- Further analysis
- Model improvement

Efficient Bandwidth Usage

This reduces network traffic and saves bandwidth, especially in large IoT environments.

5. Applications of Edge AI for Low Latency

Autonomous Vehicles

Real-time obstacle detection and braking.

Healthcare Monitoring

Wearable devices detect abnormalities instantly.

Smart Manufacturing

Predictive maintenance with sensor data in real-time.

Smartphones and Voice Assistants

Face recognition and speech processing locally.

Smart Cities

Traffic monitoring and surveillance at the edge.