SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution under UGC, New Delhi)

B.Tech. - III Year –II Semester

L	T	P	C
3	0	0	3

(R22CSD3213) BIG DATA ANALYTICS

Course Objectives:

1. The purpose of this course is to provide the students with knowledge of Big data Analytics principles and techniques.
2. This course is also designed to give an exposure to the frontiers of Big data Analytics

Courses Outcomes:

1. Ability to explain the foundations, definitions, and challenges of Big Data and various Analytical tools.
2. Ability to program using HADOOP and Map reduce, NOSQL
3. Ability to understand the importance of Big Data in social media and Mining.
4. Understand Supervised and unsupervised Learning.
5. Learn the basics of data serialization.
6. Learn about Mobile analytics.

UNIT - I

Introduction to Big Data: Big Data and its Importance – Four V's of Big Data – Drivers for Big Data – Introduction to Big Data Analytics – Big Data Analytics applications.

UNIT - II

Big Data Technologies: Hadoop's Parallel World – Data discovery – Open-source technology for Big Data Analytics–cloud and Big Data–Predictive Analytics–Mobile Business Intelligence and Big Data

UNIT - III

Introduction Hadoop : Big Data–Apache Hardtop & Hadoop Eco System–Moving Data in and out of Hadoop – Understanding inputs and outputs of Map Reduce – Data Serialization.

UNIT - IV

Hadoop Architecture: Hadoop: RDBMS Vs Hadoop, Hadoop Overview, Hadoop distributors, HDFS, HDFS Daemons, Anatomy of File Write and Read., Name Node, Secondary Name Node, and Data Node, HDFS Architecture, Hadoop Configuration, Map Reduce Framework, Role of HBase in Big Data processing, HIVE,PIG.

UNIT - V

Data Analytics with R , MachineLearning: Introduction,SupervisedLearning,UnsupervisedLearning, Collaborative Filtering, Social Media Analytics, Mobile Analytics, Big Data Analytics with Big R.

TEXT BOOKS:

1. Big Data Analytics, Seema Acharya, Subhasini Chellappan, Wiley2015.
2. Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Business, Michael Minelli, Michehe Chambers, 1st Edition, Ambiga Dhiraj, Wiely CIO Series, 2013.
3. Hadoop: The Definitive Guide, Tom White, 3rd Edition, O"Reilly Media,2012.
4. Big Data Analytics: Disruptive Technologies for Changing the Game, Arvind Sathi, 1st Edition, IBM Corporation,2012.

REFERENCE BOOKS:

1. Big Data and Business Analytics, Jay Liebowitz, Auerbach Publications, CRC press(2013)
2. Using R to Unlock the Value of Big Data: Big Data Analytics with Oracle R Enterprise and Oracle R Connector for Hadoop, Tom Plunkett, Mark Hornick, McGraw-Hill/Osborne Media (2013), Oracle press.
3. Professional Hadoop Solutions, Boris lublinsky, Kevin t. Smith, Alexey Yakubovich, Wiley, ISBN: 9788126551071, 2015.
4. Understanding Big data, Chris Eaton, Dirk deroos et al. McGraw Hill,2012.
5. Intelligent Data Analysis, Michael Berthold, David J. Hand, Springer,2007.
6. Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics, Bill Franks, 1st Edition, Wiley and SAS Business Series,2012.

UNIT-1

Introduction to Big Data: Big Data and its Importance – Four V‟s of Big Data – Drivers for Big Data – Introduction to Big Data Analytics – Big Data Analytics applications.

Big Data:

Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. It is not a single technique or a tool rather it involves many areas of business and technology.

What comes under Big data:

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

- **Black Box Data:** It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- **Social Media Data:** Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
- **Stock Exchange Data:** The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.
- **Power Grid Data:** The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data:** Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data:** Search engines retrieve lots of data from different databases.

The History of Big Data

- Although the concept of big data itself is relatively new, the origins of large data sets go back to the 1960s and '70s when the world of data was just getting started with the first data centers and the development of the relational database.
- Around 2005, people began to realize just how much data users generated through Facebook, YouTube, and other online services. Hadoop (an open-source framework created specifically to store and analyze big data sets) was developed that same year. NoSQL also began to gain popularity during this time.
- The development of open-source frameworks, such as Hadoop (and more recently, Spark) was essential for the growth of big data because they make big data easier to work with and cheaper to store. In the years since then, the volume of big data has skyrocketed. Users are still generating huge amounts of data—but it's not just humans who are doing it.
- With the advent of the Internet of Things (IoT), more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. The emergence of machine learning has produced still more data.
- While big data has come far, its usefulness is only just beginning. Cloud computing has expanded big data possibilities even further. The cloud offers truly elastic scalability, where developers can simply spin up ad hoc clusters to test a subset of data.

Benefits of Big Data:

- Big data makes it possible for you to gain more complete answers because you have more information.
- Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.
- Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.
- Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.
- More complete answers mean more confidence in the data—which means a completely different approach to tackling problems.

Characteristics of Big Data (Four V's of Big Data):

Big data can be described by the following characteristics:

- **Volume:** The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.
- **Variety:** The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.
- **Velocity:** In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.
- **Veracity:** The trustworthiness and accuracy of the data. It ensures that the data is reliable enough to be used for analysis and decision-making.

The data in it will be of three types.

- Structured data: Relational data (like SQL Tables).
- Semi Structured data: XML data, JSON Data.
- Unstructured data: Word, PDF, Text, Media Logs.

Big Data Challenges:

The major challenges associated with big data are as follows:

- Storage
- Searching
- Sharing
- Transfer
- Analysis
- Presentation

Big Data Solutions:

Google solved this problem using an algorithm called MapReduce. This algorithm divides the task into small parts and assigns them too many computers and collects the results from them which when integrated, form the result dataset.

Using the solution provided by Google, Doug Cutting and his team developed an Open Source Project called HADOOP. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.

Drivers of BigData:

Big Data is growing rapidly across industries due to technological, social, and economic changes. There are three major drivers behind the growth of Big Data:

1. Sophisticated Consumers
2. Automation
3. Monetization

1. Sophisticated Consumers

- Modern consumers are highly connected, informed, and digitally active.
- Unlike earlier times when customers depended only on company advertisements, today they compare products online, read peer reviews, watch demonstrations, and share their experiences through social media platforms.
- Every activity such as posting reviews, rating products, writing blogs, uploading images, and commenting on services generates massive amounts of unstructured data.
- Social networks also create opinion leaders whose decisions influence many followers. If a brand receives negative feedback from influential users, customer churn can spread rapidly.
- Organizations analyze this social and behavioral data using sentiment analysis and network analysis to improve products, retain customers, and design personalized marketing strategies. Thus, sophisticated consumers are a primary source of Big Data generation.

2. Automation

- Automation refers to digital systems that automatically capture and store data during every interaction.
- Earlier, data collection was manual and limited, but now websites, mobile applications, telecom networks, sensors, machines, and smart devices record data continuously without human intervention.
- For example, clickstream data tracks website navigation, telecom systems record call detail records, smart meters monitor electricity usage, and GPS devices collect location information.
- Automation significantly increases the volume, velocity, and variety of data. It enables real-time monitoring, performance tracking, and detailed behavioral analysis.
- Because every digital action leaves a data footprint, automation is a major technological driver behind the growth of Big Data.

3. Monetization

- Monetization means generating revenue from data. Organizations collect, analyze, and utilize data not only for operational efficiency but also for profit generation.
- One major example is online advertising, where user browsing behavior and search patterns are analyzed to display personalized advertisements through real-time bidding systems.
- Companies may also create data marketplaces where anonymized customer data is packaged and sold. Location-based promotions, targeted marketing campaigns, and predictive analytics models further increase business revenue.
- Since data has become a valuable economic asset, companies continuously invest in data collection, storage, and analytics technologies. This financial incentive accelerates Big Data growth.

Some other business drivers are at the core of this success and explain why Big Data has quickly risen to become one of the most coveted topics in the industry. Six main business drivers can be identified as below:

1. The digitization of society

Big Data is largely consumer driven and consumer oriented. Most of the data in the world is generated by consumers, who are nowadays 'always-on'. Most people now spend 4-6 hours per day consuming and generating data through a variety of devices and (social) applications. With every click, swipe or message, new data is created in a database somewhere around the world. Because everyone now has a smartphone in their pocket, the data creation sums to incomprehensible amounts.

2. The plummeting of technology computing

Technology related to collecting and processing massive quantities of diverse (high variety) data has become increasingly more affordable. The costs of data storage and processors keep declining, making it possible for small businesses and individuals to become involved with Big Data.

Besides the plummeting of the storage costs, a second key contributing factor to the affordability of Big Data has been the development of open source Big Data software frameworks.

3. Connectivity through cloud computing

Cloud computing environments (where data is remotely stored in distributed storage systems) have made it possible to quickly scale up or scale down IT infrastructure and facilitate a pay-as-you-go model. This means that organizations that want to process massive quantities of data (and thus have large storage and processing requirements) do not have to invest in large quantities of IT infrastructure. Instead, they can license the storage and processing capacity they need and only pay for the amounts they actually used. As a result, most of Big Data solutions leverage the possibilities of cloud computing to deliver their solutions to enterprises.

4. Increased knowledge about data science

The demand for data scientists (and similar job titles) has increased tremendously and many people have actively become engaged in the domain of data science. As a result, the knowledge and education about data science has greatly professionalized and more information becomes available every day. While statistics and data analysis mostly remained an academic field previously, it is quickly becoming a popular subject among students and the working population.

5. Social media applications

Social media data provides insights into the behaviors, preferences and opinions of 'the public' on a scale that has never been known before. Due to this, it is immensely valuable to anyone who is able to derive meaning from these large quantities of data. Social media data can be used to identify customer preferences for product development, target new customers for future purchases, or even target potential voters in elections. Social media data might even be considered one of the most important business drivers of Big Data.

6. The upcoming Internet-of-Things (IoT)

The last major Big Data business driver is the rapid rise of the Internet of Things. The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and network connectivity which enables these objects to connect and exchange data. It is increasingly gaining popularity as consumer goods providers start including 'smart' sensors in household appliances. Examples of these devices include thermostats, smoke detectors, televisions, audio systems and even smart refrigerators.

What is Big-Data Analytics?

Big Data Analytics is all about crunching massive amounts of information to uncover hidden trends, patterns, and relationships. It's like sifting through a giant mountain of data to find the gold nuggets of insight.

Here's a breakdown of what it involves:

- **Collecting Data:** Such data is coming from various sources such as social media, web traffic, sensors and customer reviews.
- **Cleaning the Data:** Imagine having to assess a pile of rocks that included some gold pieces in it. You would have to clean the dirt and the debris first. When data is being cleaned, mistakes must be fixed, duplicates must be removed and the data must be formatted properly.
- **Analyzing the Data:** It is here that the wizardry takes place. Data analysts employ powerful tools and techniques to discover patterns and trends. It is the same thing as looking for a specific pattern in all those rocks that you sorted through.

How does big data analytics work?

Big Data Analytics is a powerful tool which helps to find the potential of large and complex datasets. To get better understanding, let's break it down into key steps:

- **Data Collection:** Data is the core of Big Data Analytics. It is the gathering of data from different sources such as the customers' comments, surveys, sensors, social media, and so on. The primary aim of data collection is to compile as much accurate data as possible. The more data, the more insights.
- **Data Cleaning (Data Preprocessing):** The next step is to process this information. It often requires some cleaning. This entails the replacement of missing data, the correction of inaccuracies, and the removal of duplicates. It is like sifting through a treasure trove, separating the rocks and debris and leaving only the valuable gems behind.
- **Data Processing:** After that we will be working on the data processing. This process contains such important stages as writing, structuring, and formatting of data in a way it will be usable for the analysis. It is like a chef who is gathering the ingredients before cooking. Data processing turns the data into a format suited for analytics tools to process.
- **Data Analysis:** Data analysis is being done by means of statistical, mathematical, and machine learning methods to get out the most important findings from the processed data**. For example,** it can uncover customer preferences, market trends, or patterns in healthcare data.
- **Data Visualization:** Data analysis usually is presented in visual form, for illustration – charts, graphs and interactive dashboards. The visualizations provided a way to simplify the large amounts of data and allowed for decision makers to quickly detect patterns and trends.
- **Data Storage and Management:** The stored and managed analyzed data is of utmost importance. It is like digital scrapbooking. May be you would want to go back to those lessons in the long run, therefore, how you store them has great importance. Moreover, data protection and adherence to regulations are the key issues to be addressed during this crucial stage.
- **Continuous Learning and Improvement:** Big data analytics is a continuous process of collecting, cleaning, and analyzing data to uncover hidden insights. It helps businesses make better decisions and gain a competitive edge.

Types of Big Data Analytics

Big Data Analytics comes in many different types, each serving a different purpose:

1. **Descriptive Analytics:** This type helps us understand past events. In social media, it shows performance metrics, like the number of likes on a post.

2. **Diagnostic Analytics:** In Diagnostic analytics delves deeper to uncover the reasons behind past events. In healthcare, it identifies the causes of high patient re-admissions.

3. **Predictive Analytics:** Predictive analytics forecasts future events based on past data. Weather forecasting, for example, predicts tomorrow's weather by analyzing historical patterns.

4. **Prescriptive Analytics:** However, this category not only predicts results but also offers recommendations for action to achieve the best results. In e-commerce, it may suggest the best price for a product to achieve the highest possible profit.

5. **Real-time Analytics:** The key function of real-time analytics is data processing in real time. It swiftly allows traders to make decisions based on real-time market events.

6. **Spatial Analytics:** Spatial analytics is about the location data. In urban management, it optimizes traffic flow from the data unde the sensors and cameras to minimize the traffic jam.

7. **Text Analytics:** Text analytics delves into the unstructured data of text. In the hotel business, it can use the guest reviews to enhance services and guest satisfaction.

Big Data Analytics Technologies and Tools

Big Data Analytics relies on various technologies and tools that might sound complex, let's simplify them:

- **Hadoop:** Imagine Hadoop as an enormous digital warehouse. It's used by companies like Amazon to store tons of data efficiently. For instance, when Amazon suggests products you might like, it's because Hadoop helps manage your shopping history.
- **Spark:** Think of Spark as the super-fast data chef. Netflix uses it to quickly analyze what you watch and recommend your next binge-worthy show.
- **NoSQL Databases:** NoSQL databases, like MongoDB, are like digital filing cabinets that Airbnb uses to store your booking details and user data. These databases are famous because of their quick and flexible, so the platform can provide you with the right information when you need it.
- **Tableau:** Tableau is like an artist that turns data into beautiful pictures. The World Bank uses it to create interactive charts and graphs that help people understand complex economic data.
- **Python and R:** Python and R are like magic tools for data scientists. They use these languages to solve tricky problems. For example, Kaggle uses them to predict things like house prices based on past data.
- **Machine Learning Frameworks (e.g., TensorFlow):** In Machine learning frameworks are the tools who make predictions. Airbnb uses TensorFlow to predict which properties are most likely to be booked in certain areas. It helps hosts make smart decisions about pricing and availability.

These tools and technologies are the building blocks of Big Data Analytics and helps organizations gather, process, understand, and visualize data, making it easier for them to make decisions based on information.

Benefits of Big Data Analytics

Big Data Analytics offers a host of real-world advantages, and let's understand with examples:

1. **Informed Decisions:** Imagine a store like Walmart. Big Data Analytics helps them make smart choices about what products to stock. This not only reduces waste but also keeps customers happy and profits high.
2. **Enhanced Customer Experiences:** Think about Amazon. Big Data Analytics is what makes those product suggestions so accurate. It's like having a personal shopper who knows your taste and helps you find what you want.
3. **Fraud Detection:** Credit card companies, like MasterCard, use Big Data Analytics to catch and stop fraudulent transactions. It's like having a guardian that watches over your money and keeps it safe.
4. **Optimized Logistics:** FedEx, for example, uses Big Data Analytics to deliver your packages faster and with less impact on the environment. It's like taking the fastest route to your destination while also being kind to the planet.

Challenges of Big data analytics

While Big Data Analytics offers incredible benefits, it also comes with its set of challenges:

- **Data Overload:** Consider Twitter, where approximately 6,000 tweets are posted every second. The challenge is sifting through this avalanche of data to find valuable insights.
- **Data Quality:** If the input data is inaccurate or incomplete, the insights generated by Big Data Analytics can be flawed. For example, incorrect sensor readings could lead to wrong conclusions in weather forecasting.
- **Privacy Concerns:** With the vast amount of personal data used, like in Facebook's ad targeting, there's a fine line between providing personalized experiences and infringing on privacy.
- **Security Risks:** With cyber threats increasing, safeguarding sensitive data becomes crucial. For instance, banks use Big Data Analytics to detect fraudulent activities, but they must also protect this information from breaches.
- **Costs:** Implementing and maintaining Big Data Analytics systems can be expensive. Airlines like Delta use analytics to optimize flight schedules, but they need to ensure that the benefits outweigh the costs.

Big Data Analytics Applications:

Big Data Analytics has a significant impact in various sectors:

1. Social Media Command Center

A Social Media Command Center is a centralized analytics platform that monitors social networking platforms in real time to track customer opinions, complaints, brand mentions, and emerging trends.

Organizations collect massive volumes of unstructured data such as tweets, comments, blog posts, and online reviews, and apply sentiment analysis and text mining techniques to understand public perception.

This system helps companies detect service outages, product dissatisfaction, or viral issues quickly, enabling faster response and crisis management. By identifying

influencers and tracking brand reputation continuously, businesses can protect their image, improve customer engagement, and gain competitive insights.

2. Product Knowledge Hub

A Product Knowledge Hub integrates structured and unstructured product-related information from multiple sources such as manuals, supplier databases, websites, call center logs, and online forums into a unified searchable repository. Big Data technologies help in collecting, indexing, organizing, and retrieving large volumes of technical and customer-generated content efficiently.

This improves customer support by enabling faster issue resolution, enhances self-service portals, and reduces dependency on call centers. By centralizing knowledge, organizations ensure consistency in product information and improve overall operational efficiency.

3. Infrastructure & Operations Studies

Big Data Analytics is widely used to improve infrastructure planning and operational efficiency by analyzing data from sensors, GPS devices, network logs, and surveillance systems. Governments and enterprises use analytics to monitor traffic flow, optimize public transportation, manage utilities, detect bottlenecks, and predict maintenance requirements.

By analyzing large-scale operational data in real time, organizations can allocate resources more effectively, reduce downtime, lower operational costs, and enhance service quality. This application plays a crucial role in smart city development and enterprise performance optimization.

4. Product Selection, Design & Engineering

Big Data enables organizations to analyze product usage patterns, sensor logs, customer feedback, and failure reports to improve product design and engineering decisions. By studying real-world performance data, companies can identify frequently used features, eliminate rarely used components, detect defects early, and enhance product reliability.

Predictive analytics helps in forecasting product performance and guiding design modifications before large-scale production. This results in cost reduction, faster innovation cycles, improved product quality, and higher customer satisfaction.

5. Location-Based Services

Location-Based Services use geographic data collected from GPS systems, mobile applications, and telecom networks to deliver personalized services and targeted marketing. By analyzing customer location patterns, businesses can send real-time offers, recommend nearby stores, provide navigation assistance, and deliver geo-targeted advertisements.

This application enhances customer engagement by offering context-aware services and improving marketing effectiveness. However, it also raises privacy concerns, requiring organizations to implement secure data handling and consent mechanisms.

6. Micro-Segmentation & Next Best Action

Micro-segmentation involves dividing customers into highly specific groups based on behavioral patterns, purchase history, browsing habits, and interaction data rather than traditional demographic factors.

Using advanced analytics and machine learning, organizations identify fine-grained customer segments and determine the "Next Best Action" during customer interaction, such as recommending a product upgrade, offering a discount, or suggesting complementary services.

This real-time decision-making approach increases personalization, improves customer experience, enhances conversion rates, and maximizes revenue generation.

7. Online Advertising

Online advertising leverages Big Data to deliver highly targeted and personalized advertisements based on user browsing history, search queries, click behavior, and demographic information.

Real-time bidding systems analyze user profiles within milliseconds to determine which advertisement should be displayed. This data-driven approach increases advertisement relevance, improves return on investment (ROI), and enhances campaign effectiveness.

By continuously analyzing performance metrics, advertisers can optimize marketing strategies and achieve better engagement outcomes.

8. Improved Risk Management

Big Data Analytics significantly strengthens risk management by enabling real-time monitoring and predictive analysis across financial, operational, and security domains. Organizations use large volumes of transactional and behavioral data to detect fraud, assess creditworthiness, evaluate insurance claims, and identify cybersecurity threats.

Advanced techniques such as anomaly detection and predictive modeling help in identifying unusual patterns and preventing potential losses before they escalate.

This proactive risk management approach reduces financial exposure, ensures regulatory compliance, and enhances organizational stability.

UNIT-2

Big Data Technologies: Hadoop's Parallel World – Data discovery – Open-source technology for Big Data Analytics–cloud and Big Data–Predictive Analytics–Mobile Business Intelligence and Big Data

Hadoop's Parallel World:

"Hadoop's Parallel World" describes the way large-scale data processing is carried out simultaneously across many computers in a cluster. Instead of processing huge data on a single powerful machine, Apache Hadoop distributes the work among multiple nodes so that tasks are executed in parallel. This approach makes Big Data processing faster, scalable, and more reliable.

The original creators of Hadoop are Doug Cutting (used to be at Yahoo! now at Cloudera) and Mike Cafarella (now teaching at the University of Michigan in Ann Arbor). Doug and Mike were building a project called "Nutch" with the goal of creating a large Web index. They saw the MapReduce and GFS papers from Google, which were obviously super relevant to the problem Nutch was trying to solve. They integrated the concepts from MapReduce and GFS into Nutch; then later these two components were pulled out to form the genesis of the Hadoop project.

The name "Hadoop" itself comes from Doug 's son, he just made the word up for a yellow plush elephant toy that he has. Yahoo! hired Doug and invested significant resources into growing the Hadoop project, initially to store and index the Web for the purpose of Yahoo! Search. That said, the technology quickly mushroomed throughout the whole company as it proved to be a big hammer that can solve many problems.

In 2008, recognizing the huge potential of Hadoop to transform data management across multiple industries, Amr left Yahoo! to co-found Cloudera with Mike Olson and Jeff Hammerbacher. Doug Cutting followed in 2009.

1. Concept of Hadoop's Parallel World

In traditional computing systems, data is usually processed **sequentially**, meaning one task is completed before the next begins. When datasets become extremely large, sequential processing becomes slow and inefficient.

Hadoop introduces a **parallel computing model**, where:

- Data is divided into smaller blocks
- These blocks are distributed across several computers (nodes)
- Each node processes its portion of data **simultaneously**
- The results are then combined to generate the final output

This environment of multiple machines working together is called **Hadoop's Parallel World**.

2. Distributed Data Storage

The foundation of Hadoop's parallel processing is its storage system called **Hadoop Distributed File System (HDFS)**.

HDFS works by:

1. Breaking large files into smaller blocks
2. Storing these blocks across multiple machines in a cluster
3. Maintaining multiple copies of each block for reliability

Advantages:

- High availability
- Fault tolerance
- Efficient data management

If one machine fails, another machine containing the replicated data continues the processing.

3. Parallel Data Processing Using MapReduce

The main processing model used in Hadoop's parallel world is **MapReduce**.

Map Phase

- The input dataset is divided into smaller chunks.
- Each chunk is processed independently by different nodes.

Reduce Phase

- The results produced by the Map tasks are collected.
- These results are aggregated to produce the final output.

Because many nodes execute the Map tasks at the same time, Hadoop can process extremely large datasets very quickly.

4. Example of Hadoop's Parallel World

Consider analyzing **100 GB of web log data**.

Traditional System

- A single computer processes the entire dataset.
- Processing may take many hours.

Hadoop Parallel System

- The dataset is split into multiple blocks.
- Blocks are processed by many nodes simultaneously.
- Processing time is significantly reduced.

5. Key Characteristics of Hadoop's Parallel World

Distributed Computing

Processing is distributed across multiple machines.

Parallel Execution

Multiple tasks are executed at the same time.

Scalability

New nodes can be added to increase computing power.

Fault Tolerance

Data replication ensures system reliability even when hardware fails.

6. Benefits of Hadoop's Parallel Processing

1. **High Performance**
   Parallel execution speeds up data processing.
2. **Cost Efficiency**
   Uses inexpensive commodity hardware instead of expensive supercomputers.
3. **Scalability**
   Organizations can easily expand clusters by adding new nodes.
4. **Reliability**
   Data replication prevents data loss.

7. Importance in Big Data Analytics

Hadoop's parallel world enables organizations to process huge datasets generated from:

- Social media platforms
- Online transactions
- IoT devices
- Scientific research
- Web logs

Industries such as e-commerce, banking, healthcare, and telecommunications rely on Hadoop-based systems for large-scale analytics.

Data Discovery: Work the Way People's Minds Work

Data Discovery refers to the process of exploring and analyzing data in an intuitive and interactive way so that users can quickly find patterns, relationships, and insights without needing complex technical skills.

Traditional business intelligence systems required IT specialists to create reports, but modern data discovery tools allow business users themselves to explore data visually and interactively.

The idea behind the phrase "Work the Way People's Minds Work" is that analytics tools should match how humans naturally think, explore, and ask questions.

1. Meaning of Data Discovery

Data discovery is a process that enables users to:

- Explore large datasets
- Identify hidden patterns
- Discover relationships in data
- Generate insights interactively

Instead of using **predefined queries and preconfigured dashboards**, users can freely analyze data in a flexible way.

This approach gives users the **freedom and flexibility to work with Big Data**.

2. Shift from Traditional BI to Data Discovery

Traditional Business Intelligence

In the traditional BI approach:

1. Business users request reports from the IT department.
2. IT teams build queries and dashboards.
3. Reports are delivered to users.
4. If users need additional analysis, another request must be made.

This process is:

- Slow
- Rigid
- Highly dependent on IT

Data Discovery Approach

Data discovery introduces a **self-service analytics model**, where:

- Business users explore data directly
- Reports and dashboards are created interactively
- Users can drill down into data instantly

3. Companies Leading the Data Discovery Movement

Two companies that became very successful in this field:

- **Tableau Software**
- **QlikTech International**

These companies developed powerful tools that allow users to **visualize and explore data easily**.

Their software enables users to:

- Create charts and dashboards
- Filter and analyze data interactively
- Discover insights without programming knowledge

4. "Land and Expand" Business Model

Land

Initially, the software is introduced to a **small group of business users**.

Expand

If users find the tool useful, its usage gradually expands across the organization.

This approach is different from traditional BI vendors that tried to **sell large enterprise systems to IT departments first**.

5. Self-Service Analytics

Data discovery tools promote **self-service analytics**, meaning:

- Business users generate their own reports
- IT departments focus on managing infrastructure and data quality
- Users analyze data independently

This reduces the heavy dependency on IT teams.

6. Role of Visualization

Visualization plays an important role in data discovery.

Instead of analyzing complex spreadsheets, users can interpret data through:

- Charts
- Graphs
- Dashboards
- Interactive visualizations

These visual tools help users **quickly understand patterns and trends**.

7. Benefits of Data Discovery

The benefits include:

Faster Data Analysis

Users can analyze data instantly without waiting for IT.

Improved Decision Making

Interactive exploration leads to better insights.

Greater Flexibility

Users can ask new questions and explore data in different ways.

Better Understanding of Big Data

Visualization makes complex datasets easier to understand.

Open-Source Technology for Big Data Analytics:

Open-source technology plays a crucial role in Big Data analytics because it provides **powerful, flexible, and cost-effective tools** for processing and analyzing massive datasets. Open-source platforms allow organizations and researchers to **use, modify, and distribute software freely**, which accelerates innovation and development in the field of Big Data.

1. Cost Effectiveness

Most open-source tools are **free to use**, which reduces the cost of software licenses. Organizations can implement Big Data solutions without investing heavily in proprietary software.

2. Flexibility and Customization

Open-source tools allow developers to **modify the source code** according to their requirements. This flexibility makes it easier to customize analytics platforms for different industries such as healthcare, finance, and e-commerce.

3. Scalability

Many open-source Big Data tools are designed to run on **distributed computing environments**, enabling organizations to process extremely large datasets across multiple machines.

4. Community Support and Innovation

Open-source technologies are supported by **large developer communities** that continuously improve the tools, fix bugs, and introduce new features.

5. Rapid Development and Integration

Open-source platforms are built to integrate with other technologies, enabling organizations to build **complete Big Data ecosystems** quickly.

Various Open-Source Technologies Used in Big Data Analytics

1. Apache Hadoop

Apache Hadoop is one of the most popular open-source frameworks used for storing and processing large datasets.

Features:

- Distributed storage using **HDFS (Hadoop Distributed File System)**
- Parallel data processing using MapReduce
- High fault tolerance
- Scalable across thousands of nodes

Hadoop is widely used for **batch processing of Big Data**.

2. Apache Spark

Apache Spark is a fast and powerful Big Data processing engine designed to overcome the limitations of Hadoop MapReduce.

Features:

- In-memory data processing
- Faster than MapReduce
- Supports real-time analytics
- Supports machine learning and graph processing

Spark is widely used in **data analytics and machine learning applications**.

3. Apache Hive

Apache Hive is a data warehouse system built on top of Hadoop that allows users to query large datasets using SQL-like language called **HiveQL**.

Features:

- SQL-like querying
- Data summarization
- Easy integration with Hadoop ecosystem

It is useful for **data analysis and reporting**.

4. Apache Pig

Apache Pig provides a high-level scripting language called **Pig Latin** for processing large datasets in Hadoop.

Features:

- Simplifies complex data processing tasks
- Easy to write data transformation programs
- Works on top of Hadoop MapReduce

5. Apache HBase

Apache HBase is a distributed database that runs on top of Hadoop and is designed for **real-time read/write access to Big Data**.

Features:

- Column-oriented storage
- High scalability
- Suitable for large datasets

6. Apache Flume

Apache Flume is used to **collect, aggregate, and transfer large amounts of log data** into Hadoop.

Features:

- Reliable data ingestion
- Scalable architecture
- Used for streaming log data

7. Apache Kafka

Apache Kafka is a platform used for **real-time data streaming and processing**.

Features:

- High throughput
- Real-time data pipelines
- Fault tolerance

It is widely used for **real-time Big Data applications**.

Cloud and Big Data:

1. What is Big Data?

- **Big Data** refers to very large and complex datasets that cannot be handled efficiently using traditional database systems.
- Big Data is usually defined by the **5 V's**:
    1. **Volume** – Huge amount of data (terabytes, petabytes, exabytes).
    2. **Velocity** – Speed at which data is generated and processed.
    3. **Variety** – Different data types (structured, semi-structured, unstructured).
    4. **Veracity** – Data reliability and accuracy.
    5. **Value** – Extracting useful insights from data.

Examples of Big Data sources:

- Social media posts
- Online transactions
- Sensors and IoT devices
- Mobile applications
- Website logs

Handling such data requires powerful computing resources.

2. What is Cloud Computing?

Cloud Computing is the delivery of computing services over the internet, including:

- Storage
- Servers
- Databases
- Networking
- Software
- Analytics

Instead of buying expensive hardware, organizations can use cloud services on demand.
Common cloud service models:

Model	Description
IaaS (Infrastructure as a Service)	Provides virtual machines, storage, and networks
PaaS (Platform as a Service)	Provides development platforms
SaaS (Software as a Service)	Software delivered via internet

Examples of cloud providers:

- **Amazon Web Services (AWS)**
- **Microsoft Azure**
- **Google Cloud Platform**

3. Why Big Data Needs Cloud Computing

Big Data requires massive resources for storage and computation. Traditional systems have limitations such as:

- Limited storage capacity
- High infrastructure cost
- Difficulty scaling
- Complex maintenance

Cloud computing solves these problems.

Key Support from Cloud:

1. Massive Storage
2. High Processing Power
3. Scalability
4. Cost Efficiency
5. Distributed Computing

4. How Cloud Supports Big Data

1. Scalable Storage for Big Data

- Big Data requires enormous storage capacity.
- Cloud computing platforms provide **scalable and distributed storage systems** that allow organizations to store large datasets without maintaining physical hardware.

Example:

Cloud storage systems can store **petabytes of data** generated from social media, sensors, and business transactions.

2. High Processing Power

- Big Data analytics requires powerful computing resources.
- Cloud platforms provide **distributed computing frameworks** that process data in parallel across multiple servers.

Example tools commonly used with Big Data:

- Apache Hadoop
- Apache Spark

These tools run efficiently on cloud infrastructure.

3. Cost Efficiency

- Without cloud computing, organizations would need to purchase expensive hardware to process Big Data.
- Cloud computing offers **pay-as-you-use services**, which reduces infrastructure cost.

Advantages:

- No need to buy servers
- No maintenance cost
- Flexible resource allocation

4. Real-Time Data Processing

- Big Data applications often require **real-time data analysis**.
- Cloud platforms provide services that support **real-time streaming analytics**, enabling faster decision-making.

Example applications:

- Online recommendation systems
- Fraud detection
- Social media analytics

5. Data Accessibility

- Cloud computing allows Big Data platforms to be **accessed from anywhere through the internet**.

Benefits:

- Multiple users can access the data simultaneously
- Easy collaboration among teams
- Data sharing across organizations

6. Integration with Big Data Tools

- Cloud providers offer **built-in tools and services specifically designed for Big Data analytics**.

Examples include:

- Data warehousing
- Machine learning tools
- Data visualization services

These services simplify the process of analyzing large datasets.

7. Improved Reliability and Backup

Cloud systems store data across **multiple distributed servers**, ensuring:

- High availability
- Automatic backup
- Disaster recovery

This makes Big Data systems more reliable.

Predictive Analytics:

Predictive analytics has become one of the most important areas in modern data analytics. Organizations today collect massive amounts of data from various sources such as transactions, social media, sensors, and online activities. Instead of only analyzing past events, companies now want to predict future outcomes and trends. This growing need has brought predictive analytics into the spotlight, making it a key component of Big Data technologies.

1. Meaning of Predictive Analytics

Predictive analytics is a method of analyzing historical and current data using **statistical techniques, data mining, and machine learning algorithms** to predict future events or behaviors.

It answers questions such as:

- What will happen next?
- Which customers are likely to buy a product?
- Which transactions may be fraudulent?
- Which machine may fail soon?

Predictive analytics transforms raw data into **actionable insights** that help organizations make better decisions.

2. Why Predictive Analytics Became Important

Earlier business intelligence systems focused mainly on **descriptive analytics**, which explains what has already happened. However, businesses need more advanced insights to remain competitive.

Predictive analytics became important because it helps organizations to:

1. **Forecast future trends**
   Companies can anticipate demand, sales, and market behavior.
2. **Improve decision making**
   Managers can take proactive actions instead of reacting after problems occur.
3. **Reduce risks**
   Predictive models can identify potential risks such as fraud or equipment failure.
4. **Enhance customer experience**
   Businesses can predict customer preferences and offer personalized services.
5. **Increase operational efficiency**
   Predictive insights help optimize resources and processes.

3. Role of Big Data in Predictive Analytics

The growth of Big Data has significantly increased the effectiveness of predictive analytics. Large datasets provide more information for building accurate predictive models.

Big Data technologies allow organizations to:

- Process massive datasets
- Analyze real-time data streams
- Apply advanced machine learning models
- Generate accurate predictions

Technologies such as **Apache Hadoop** and **Apache Spark** are often used to process large datasets for predictive analytics.

4. Techniques Used in Predictive Analytics

Several analytical and machine learning techniques are used in predictive analytics.

- ***Regression Analysis***
  Used to predict continuous values such as sales forecasts or price predictions.
- ***Classification***
  Used to categorize data into different groups, such as identifying fraudulent transactions.
- ***Clustering***
  Groups similar data points together to discover patterns.
- ***Time Series Analysis***
  Used to analyze trends over time, such as predicting stock prices or weather patterns.
- ***Machine Learning Algorithms***
  Algorithms learn patterns from historical data and use them to make predictions.

5. Applications of Predictive Analytics

Predictive analytics is widely used across many industries.

1. ***Marketing***
   Companies analyze customer behavior to predict purchasing patterns and design targeted advertisements.
   Example: Streaming platforms like **Netflix** recommend movies based on users' viewing history.
2. ***Banking and Finance***
   Banks use predictive models to detect fraudulent transactions and assess credit risk.
3. **Healthcare**
   Hospitals predict disease outbreaks and identify high-risk patients for early treatment.
4. **Retail**
   Retailers forecast product demand and manage inventory effectively.
5. **Manufacturing**
   Predictive analytics helps detect machine failures before they occur, reducing downtime.

6. Benefits of Predictive Analytics

Predictive analytics provides several advantages:

- Better strategic planning
- Improved customer satisfaction
- Reduced operational risks
- Increased revenue opportunities
- Faster and more accurate decision making

Organizations can transform raw data into valuable insights that support business growth.

7. Challenges of Predictive Analytics

Despite its advantages, predictive analytics also has some challenges:

- Need for high-quality data
- Complexity of analytical models
- Requirement of skilled data scientists
- Data privacy and security issues

Mobile Business Intelligence and Big Data:

Mobile Business Intelligence (Mobile BI) is an important development in Big Data analytics that allows users to access business data and analytics through mobile devices such as smartphones and tablets. With the rapid growth of mobile technology and Big Data, organizations can now analyze large datasets and access insights anytime and anywhere.

Mobile BI combines mobile computing, cloud computing, and Big Data analytics to deliver real-time business insights to managers, executives, and employees.

1. Meaning of Mobile Business Intelligence

- Mobile Business Intelligence refers to the **delivery of business intelligence and analytics information through mobile devices**. It allows users to view reports, dashboards, and analytics on mobile applications instead of traditional desktop systems.
- Mobile BI helps organizations make **faster decisions by providing real-time data access**.
- **Example:**

  A sales manager can view sales performance reports on a smartphone while traveling.

2. Relationship Between Mobile BI and Big Data

Big Data generates massive amounts of data from various sources such as:

- Social media
- Sensors
- Online transactions
- Mobile applications
- IoT devices

3. Components of Mobile BI

Mobile BI systems generally include the following components:

1. **Data Sources**

   Data is collected from different systems such as enterprise databases, websites, social media, and IoT devices.
2. **Big Data Storage**

   Large datasets are stored in distributed systems such as **Apache Hadoop**.
3. **Data Processing and Analytics**

   Big Data processing frameworks analyze the data using tools such as **Apache Spark**.
4. **BI Tools and Dashboards**

   Analytical tools generate reports, charts, and dashboards.
5. **Mobile Applications**

   Mobile devices display analytics results through mobile apps and web interfaces.

4. Features of Mobile Business Intelligence

Mobile BI provides several important features:

- **Real-Time Data Access**

  Users can access updated information instantly from anywhere.
- **Interactive Dashboards**

  Mobile dashboards allow users to explore data using touch interfaces.
- **Alerts and Notifications**

  Managers receive notifications when important events occur.

Example:

- Low inventory alerts
- Sales performance updates

- **Location-Based Analytics**
  Mobile devices can provide insights based on geographic location.

5. Advantages of Mobile BI in Big Data

- **Faster Decision Making**
  Managers can analyze data and make decisions immediately.
- **Increased Productivity**
  Employees can access reports without returning to the office.
- **Improved Collaboration**
  Teams can share data insights quickly through mobile platforms.
- **Real-Time Monitoring**
  Executives can monitor business performance continuously.
- **Better Customer Service**
  Employees can access customer data instantly while interacting with customers.

6. Applications of Mobile BI

Mobile BI is widely used in different industries.

- **Retail**
  Store managers monitor sales and inventory levels using mobile dashboards.
- **Banking**
  Bank managers analyze financial data and transactions through mobile devices.
- **Healthcare**
  Doctors can access patient records and analytics through mobile applications.
- **Logistics**
  Delivery companies track shipments and analyze routes using mobile analytics.
- **Sales and Marketing**
  Sales teams monitor customer interactions and performance metrics on mobile devices.

7. Challenges of Mobile BI

Despite its advantages, Mobile BI faces some challenges.

- **Security Risks**
  Sensitive business data may be exposed if mobile devices are lost or hacked.
- **Device Compatibility**
  Different mobile devices and operating systems require different applications.
- **Limited Screen Size**
  Displaying complex dashboards on small screens can be difficult.
- **Network Dependency**
  Mobile BI depends on internet connectivity.

8. Future of Mobile BI with Big Data

Mobile BI is expected to grow rapidly due to:

- Increasing use of smartphones
- Growth of Big Data analytics
- Development of cloud computing
- Expansion of IoT devices

UNIT-3

Introduction Hadoop: Big Data–Apache Hadoop & Hadoop Eco System–Moving Data in and out of Hadoop – Understanding inputs and outputs of Map Reduce – Data Serialization.

Big Data Apache Hadoop & Hadoop Eco System:

The Apache Hadoop ecosystem refers to the collection of tools and technologies that work together with Hadoop to store, process, manage, and analyze large volumes of Big Data. Hadoop itself mainly provides distributed storage and distributed processing, while the ecosystem tools add functionalities such as querying, data ingestion, workflow management, and machine learning.

1. Core Components of Hadoop

1. HDFS

It stands for "Hadoop Distributed File System" and is the storage unit in Hadoop. The storage of Big Data is distributed across many computers and stored in blocks. HDFS creates copies of data stored on multiple systems using the replication method. This prevents data loss if one of the nodes fails and makes HDFS fault tolerant.

It comprises the following:

a. Storage Nodes in HDFS

The Hadoop Distributed File System (HDFS) stores data in a cluster of physical workstations or servers called storage nodes. These nodes are in charge of holding and managing the actual data blocks that make up the files kept in the cluster.

- **NameNode (Master):** The NameNode is the master in the Hadoop cluster that guides the DataNode (Slaves). It stores the metadata, which contains transaction logs, file names, size, location information, etc. It assists in locating the nearest DataNode for more rapid communication and directs them on activities such as deleting, creating, and replicating.
- **DataNode (Slave):** DataNodes serve as Slave nodes. They are used to store data in a Hadoop cluster and range in number from one to 500 or more. An increased number of DataNodes allows the Hadoop cluster to store more data.
- **Secondary NameNode:** It is not a backup, but a buffer and standby for the NameNode as it stores intermediate updates when the NameNode is inactive.

b. File Block in HDFS

Blocks are the storage units for data in HDFS. The size of one block is 128 megabytes (MB) by default and can be changed manually. For example, to store 550 MB of data, HDFS partitions the 550 MB of data into blocks, with the first four blocks containing 128 MB each, and the remaining 38 megabytes in the fifth block. This process is performed across the data nodes in the cluster.

2. MapReduce

The Java programming language models **MapReduce**, the data processing software unit. It performs distributive and parallel processing of data. This makes Hadoop a fast-working software. MapReduce's approach differs from traditional processing methods, where

all data types were processed on a single unit. It produces the resulting output, which is aggregated for better performance.

How do the MapReduce tasks (Map and Reduce) work?

The big data Input is passed to the *map ()* function, and its output is passed to the *reduce()* function. Later, the final output is received. MapReduce is divided into two tasks: The map task and the reduce task.

a. Map Task

The input data blocks are divided into key-value pairs known as Tuples by the *map()* function. The Map task involves:

- ***RecordReader*:** It breaks the record and provides key-value pairs in a *map()* function. In key-value pairs, the 'key' refers to the record's location information, and the 'value' refers to the data that goes with it.
- ***map()* Function:** This is a user-defined function that processes the key-value pairs (Tuples) from the record reader.
- **Combiner**: In the Map workflow, its function is to group the data. Its use is optional.

b. Reduce Task

The key-value pairs serve as inputs to the *reduce()* function. Upon receiving broken Tuples or key-value pairs, it combines them based on their Key value and performs the following operations:

- **Shuffle and Sort:** The Reduce () function begins with this step of shuffling and sorting. It shuffles and sorts the data based on key-value pairs.
- **Reduce:** It aggregates them into groups of similar key-value pairs.
- **OutputFormat**: After all operations are completed, key-value pairs are inserted into the file using the record writer, starting on a new line and separated by spaces.

3. YARN (Yet Another Resource Negotiator)

The acronym YARN stands for "Yet Another Resource Negotiator". It is a resource management and job scheduling unit. This Hadoop component manages the resources to run the processed data effectively. The data processing results of MapReduce are run on the Hadoop cluster simultaneously, and each of them needs some resources to complete the task. This is done with the help of a set of resources such as RAM, network bandwidth, and the CPU.

To process job requests and manage cluster resources in Hadoop, it consists of the following components:

a. **Resource Manager-** The resource manager serves as the master and assigns resources.
b. **Node Manager-** The node manager aids the resource manager and is responsible for handling the nodes and resource usage in the nodes.
c. **Application Manager-** The application manager is responsible for requesting containers from the node manager. Once the node manager gets the resources, he sends them to the resource manager.
d. **Container-** It is a collection of physical resources that conduct the actual processing of data.

4. Hadoop Common

Hadoop Common, also known as Common Utilities, includes core Java libraries and scripts required by all components in a Hadoop ecosystem such as HDFS, YARN and MapReduce.

These libraries offer core functionalities such as:

- File system and I/O operations
- Configuration and logging
- Security and authentication
- Network communication

Open Source Tools for Hadoop

1. Hive

- **Apache Hive** is a **data warehouse tool** for Hadoop.
- Provides SQL-like language called **HiveQL**.
- Used for querying and analyzing large datasets.

Example query:

```
SELECT * FROM sales WHERE amount > 1000;
```

2. Pig

- **Apache Pig** is a high-level platform for analyzing large data sets.
- Uses **Pig Latin** scripting language.
- Simplifies writing MapReduce programs.

3. HBase

- **Apache HBase** is a **distributed NoSQL database** built on Hadoop.
- Provides **real-time read/write access** to large datasets.
- Suitable for random data access.

4. Sqoop

- **Apache Sqoop** transfers data between **Hadoop and relational databases** like MySQL or PostgreSQL.

Uses:

- Import data from RDBMS to Hadoop
- Export data from Hadoop to RDBMS

5. Flume

- **Apache Flume** collects and moves **large amounts of log data** into Hadoop.

Example:

- Website log data → Hadoop storage.

6. Oozie

- **Apache Oozie** is a **workflow scheduler** for Hadoop jobs.
- Manages and coordinates tasks such as MapReduce, Hive, and Pig jobs.

7. Spark

- **Apache Spark** is a fast data processing engine.
- Processes data **much faster than MapReduce** using in-memory computing.

8. ZooKeeper

- **Apache ZooKeeper** manages synchronization and coordination between distributed systems.

Moving Data in and out of Hadoop:

Apache Hadoop is widely used for storing and processing large volumes of data. However, data must first be imported into Hadoop from external systems and sometimes exported back to other systems after processing.

The process of transferring data between Hadoop and external systems is called moving data in and out of Hadoop.

This data movement is usually performed using tools from the Hadoop ecosystem such as Apache Sqoop and Apache Flume.

1. Moving Data into Hadoop

Moving data into Hadoop means **importing data from external sources into Hadoop storage such as HDFS or HBase**.

Data Sources

Common data sources include:

- Relational Databases (MySQL, Oracle)
- Log files
- Social media streams
- Web server logs
- Sensor data
- Enterprise applications

Methods for Importing Data

1. Using Apache Sqoop

Apache Sqoop is used to transfer bulk data between **relational databases and Hadoop**.

Features

- Imports structured data into HDFS
- Supports parallel data transfer
- Works with databases like MySQL, Oracle, and PostgreSQL.

Example

Import data from MySQL to HDFS.

```
sqoop import \
--connect jdbc:mysql://localhost/employees \
--username root \
--table emp \
--target-dir /hadoop/empdata
```

2. Using Apache Flume

Apache Flume is used for collecting and transferring **large amounts of streaming data** such as log files.

Features

- Real-time data ingestion
- Reliable and scalable
- Used for log monitoring.

Example

Website logs → Flume → HDFS

3. Using HDFS Commands

Data can also be directly uploaded into Hadoop using **HDFS commands**.

Example:

```
hdfs dfs -put file.txt /user/data
```

This command copies a file from the **local file system to HDFS**.

2. Moving Data Out of Hadoop

Sometimes processed data in Hadoop must be **exported to external systems**.

Methods for Exporting Data

1. Using Apache Sqoop

Apache Sqoop can export processed Hadoop data back to relational databases.

Example

```
sqoop export \
--connect jdbc:mysql://localhost/sales \
--username root \
--table sales_data \
--export-dir /hadoop/output
```

2. Using HDFS Commands

Files can be copied from Hadoop to the local system.

Example:

```
hdfs dfs -get /user/data/output.txt
```

This command downloads files from **HDFS to the local system**.

3. Using Hive Queries

Data stored in Hadoop can be exported using **Apache Hive** queries.

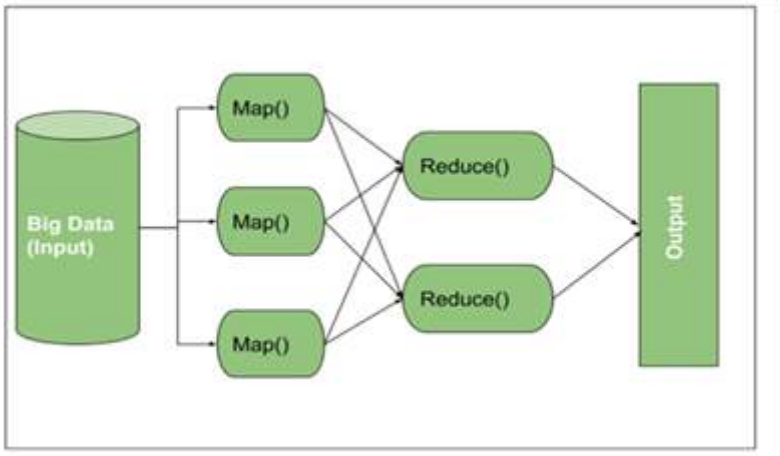
Example:

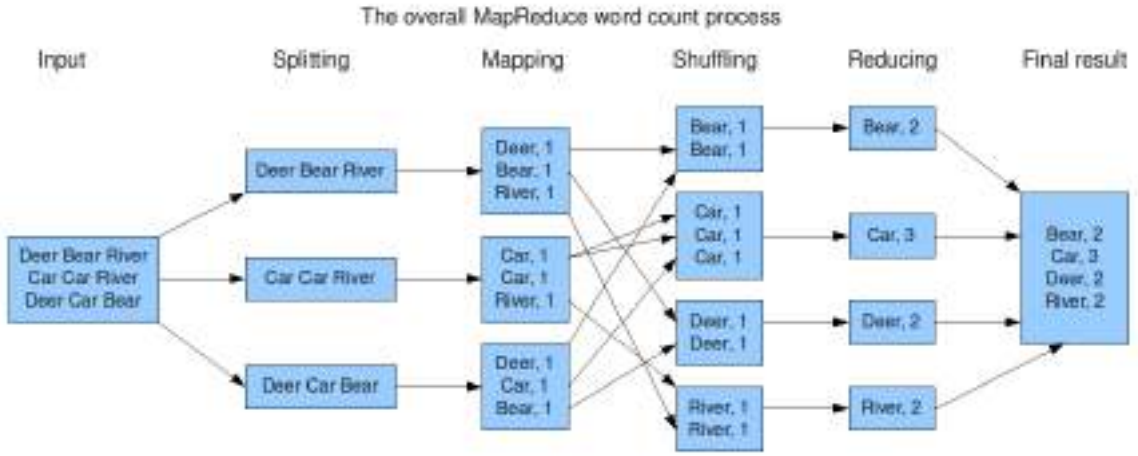
```
INSERT OVERWRITE DIRECTORY '/export/data'
SELECT * FROM sales;
```

Understanding inputs and outputs of Map Reduce:

MapReduce is the processing engine of Hadoop. While HDFS is responsible for storing massive amounts of data, MapReduce handles the actual computation and analysis. It provides a simple yet powerful programming model that allows developers to process large datasets in a distributed and parallel manner. It is a two-phase data processing model in Hadoop:

- **Map Phase:** Breaks the data into smaller chunks, processes them, and generates intermediate (key, value) pairs.
- **Reduce Phase:** Aggregates and merges the intermediate results to produce the final output.





How MapReduce Works

Example Input File: sample.txt

```
Hello I am GeeksforGeeks
How can I help you
How can I assist you
Are you an engineer
Are you looking for coding
Are you looking for interview questions
what are you doing these days
what are your strengths
```

When stored in HDFS, this file is divided into input splits (e.g., first.txt, second.txt, etc.). Each split is assigned to a Mapper for processing.

Step 1: Input Splitting & Record Reader

- The input file is divided into splits.
- Each split is converted into (key, value) pairs using a RecordReader.
- In TextInputFormat (default), the key is the byte offset of the line and the value is the line itself.

Example:

```
(0, "Hello I am GeeksforGeeks")
(26, "How can I help you")
```

File Formats in Hadoop

The way a RecordReader converts text into (key, value) pairs depends on the input file format. Hadoop provides several built-in formats:

- *TextInputFormat(Default):* Each line becomes (byte offset, line).
- *KeyValueTextInputFormat:* Splits each line into (key, value) pairs using a delimiter.
- *SequenceFileInputFormat:* Stores data in a binary (key, value) format (efficient for data exchange between MapReduce jobs).
- *SequenceFileAsTextInputFormat:* Reads binary sequence files but presents keys and values as text.

By default, Hadoop uses *TextInputFormat*.

Step 2: Map Phase

Each Mapper processes its assigned (key, value) pair and generates intermediate data.

```
For (0, "Hello I am GeeksforGeeks"):
(Hello, 1)
(I, 1)
(am, 1)
(GeeksforGeeks, 1)
For (26, "How can I help you"):
(How, 1)
(can, 1)
(I, 1)
(help, 1)
(you, 1)
```

All Mappers run in parallel, one per input split.

Step 3: Shuffling and Sorting

The Mapper outputs are not final. Before reducing:

- Shuffling groups all values of identical keys:

```
(How, [1,1])
(Are, [1,1,1])
(I, [1,1])
```

- Sorting arranges keys so that they can be processed sequentially.

This prepares data for the reducer.

Step 4: Reduce Phase

The Reducer aggregates values for each key:

```
(How, [1,1]) → (How, 2)
(Are, [1,1,1]) → (Are, 3)
(I, [1,1]) → (I, 2)
```

Final output (saved in result.output):

```
Hello - 1
I - 2
am - 1
GeeksforGeeks - 1
How - 2
Are - 3
are - 2
you - 3
what - 2
...
```

Advantages of MapReduce in Hadoop

- **Scalable:** Efficiently processes petabytes of data across thousands of nodes.
- **Parallel:** Executes tasks simultaneously on distributed data blocks.
- **Fault-Tolerant:** Recovers automatically by reassigning failed tasks.
- **Simple:** Developers only write Map and Reduce logic; Hadoop manages execution.

Data Serialization:

Data is serialized for two objectives −

- For persistent storage
- To transport the data over network

What is Serialization?

Serialization is the process of translating data structures or objects state into binary or textual form to transport the data over network or to store on some persistent storage. Once the data is transported over network or retrieved from the persistent storage, it needs to be deserialized again. Serialization is termed as marshalling and deserialization is termed as unmarshalling.

Serialization in Java:

- Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object.
- After a serialized object is written into a file, it can be read from the file and deserialized. That is, the type information and bytes that represent the object and its data can be used to recreate the object in memory.
- ***ObjectInputStream*** and ***ObjectOutputStream*** classes are used to serialize and deserialize an object respectively in Java.

Serialization in Hadoop

Generally in distributed systems like Hadoop, the concept of serialization is used for **Interprocess Communication** and **Persistent Storage**.

Interprocess Communication

- To establish the interprocess communication between the nodes connected in a network, RPC technique was used.
- RPC used internal serialization to convert the message into binary format before sending it to the remote node via network. At the other end the remote system deserializes the binary stream into the original message.
- The RPC serialization format is required to be as follows –
  - **Compact** − To make the best use of network bandwidth, which is the most scarce resource in a data center.
  - **Fast** − Since the communication between the nodes is crucial in distributed systems, the serialization and deserialization process should be quick, producing less overhead.
  - **Extensible** − Protocols change over time to meet new requirements, so it should be straightforward to evolve the protocol in a controlled manner for clients and servers.
  - **Interoperable** − The message format should support the nodes that are written in different languages.

Persistent Storage

Persistent Storage is a digital storage facility that does not lose its data with the loss of power supply. Files, folders, databases are the examples of persistent storage.

Writable Interface

This is the interface in Hadoop which provides methods for serialization and deserialization. The following table describes the methods –

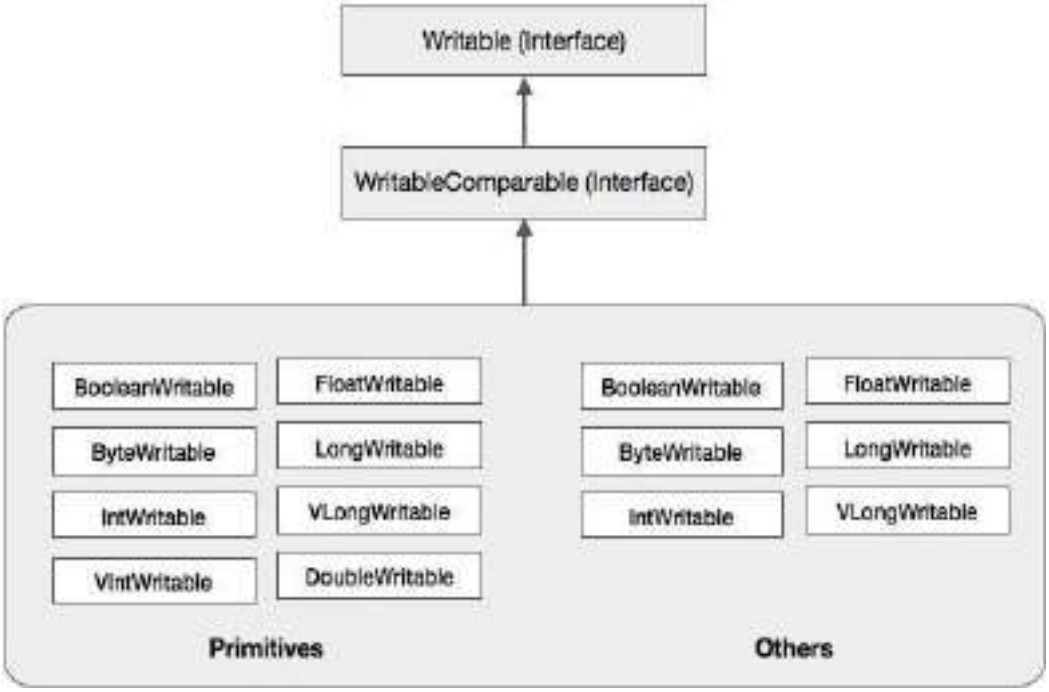
S.No.	Methods and Description
1	***void readFields(DataInput in)*** This method is used to deserialize the fields of the given object.
2	***void write(DataOutput out)*** This method is used to serialize the fields of the given object.

Writable Comparable Interface

It is the combination of Writable and Comparable interfaces. This interface inherits Writable interface of Hadoop as well as Comparable interface of Java. Therefore it provides methods for data serialization, deserialization, and comparison.

S.No.	Methods and Description
1	***int compareTo(class obj)*** This method compares current object with the given object obj.

In addition to these classes, Hadoop supports a number of wrapper classes that implement ***WritableComparable*** interface. Each class wraps a Java primitive type. The class hierarchy of Hadoop serialization is given below −



These classes are useful to serialize various types of data in Hadoop. For instance, let us consider the ***IntWritable*** class. Let us see how this class is used to serialize and deserialize the data in Hadoop.

IntWritable Class

This class implements ***Writable, Comparable,*** and ***WritableComparable*** interfaces. It wraps an integer data type in it. This class provides methods used to serialize and deserialize integer type of data.

Constructors

S.No.	Summary
1	*IntWritable()*
2	*IntWritable(int value)*

Methods

S.No.	Summary
1	***int get()*** Using this method you can get the integer value present in the current object.
2	***void readFields(DataInput in)*** This method is used to deserialize the data in the given ***DataInput*** object.
3	***void set(int value)*** This method is used to set the value of the current ***IntWritable*** object.
4	***void write(DataOutput out)*** This method is used to serialize the data in the current object to the given ***DataOutput*** object.

Serializing the Data in Hadoop

The procedure to serialize the integer type of data is discussed below.

- Instantiate ***IntWritable*** class by wrapping an integer value in it.
- Instantiate ***ByteArrayOutputStream*** class.

- Instantiate ***DataOutputStream*** class and pass the object of ***ByteArrayOutputStream*** class to it.
- Serialize the integer value in ***IntWritable*** object using ***write()*** method. This method needs an object of ***DataOutputStream*** class.
- The serialized data will be stored in the byte array object which is passed as parameter to the ***DataOutputStream*** class at the time of instantiation. Convert the data in the object to byte array.

Example:

```
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
class Serialization
{
	public byte[] serialize() throws IOException
	{
		IntWritable iw = new IntWritable(12);

		ByteArrayOutputStream bos = new ByteArrayOutputStream();
		DataOutputStream dos = new DataOutputStream(bos);

		intwritable.write(dos);

		byte[] byteArray = bos.toByteArray();

		dos.close();

		return(byteArray);
	}
	public static void main(String args[]) throws IOException
	{
		Serialization sobj= new Serialization();
		sobj.serialize();
		System.out.println();
	}
}
```

Deserializing the Data in Hadoop:

The procedure to deserialize the integer type of data is discussed below −

- Instantiate ***IntWritable*** class by wrapping an integer value in it.
- Instantiate ***ByteArrayOutputStream*** class.
- Instantiate ***DataOutputStream*** class and pass the object of ***ByteArrayOutputStream*** class to it.
- Deserialize the data in the object of ***DataInputStream*** using ***readFields()*** method of ***IntWritable*** class.
- The deserialized data will be stored in the object of ***IntWritable*** class. You can retrieve this data using ***get()*** method of this class.

Example:

```
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import org.apache.hadoop.io.IntWritable;
class Deserialization
{
	public void deserialize(byte[]  byteArray) throws Exception
	{
		IntWritable iw =new IntWritable();
		ByteArrayInputStream bis = new ByteArrayInputStream(byteArray);
		DataInputStream dis=new DataInputStream(bis);

		iw.readFields(dis);

		System.out.println((iw).get());
	}
	public static void main(String args[]) throws Exception
	{
		Deserialization dobj = new Deserialization();
		dobj.deserialize(new Serialization().serialize());
	}
}
```

BytesWritable Class:

A byte sequence that is usable as a key or value. It is resizable and distinguishes between the size of the sequence and the current capacity. The hash function is the front of the md5 of the buffer. The sort order is the same as memcmp.

Constructors:

Sno	Constructor
1	*BytesWritable()*
2	*BytesWritable(byte[] b)*
3	*BytesWritable(byte[] b,int length)*

Methods:

Modifier and Type	Method and Description
byte[]	***copyBytes****()* *Get a copy of the bytes that is exactly the length of the data.*
boolean	***equals****(****Object*** *right_obj)* *Are the two byte sequences equal?*
byte[]	***get****()* ***Deprecated.*** *Use* ***getBytes()*** *instead.*
byte[]	***getBytes****()* *Get the data backing the BytesWritable.*
int	***getCapacity****()* *Get the capacity, which is the maximum size that could handled without resizing the backing storage.*
int	***getLength****()* *Get the current size of the buffer.*
int	***getSize****()* ***Deprecated.*** *Use* ***getLength()*** *instead.*
int	***hashCode****()* *Return a hash of the bytes returned from {#getBytes()}.*

void	***readFields**(**DataInput** in)* *Deserialize the fields of this object from in.*
void	***set**(byte[] newData, int offset, int length)* *Set the value to a copy of the given byte range*
void	***set**(**BytesWritable** newData)* *Set the BytesWritable to the contents of the given newData.*
void	***setCapacity**(int new_cap)* *Change the capacity of the backing storage.*
void	***setSize**(int size)* *Change the size of the buffer.*
String	***toString**()* *Generate the stream of bytes as hex pairs separated by ' '.*
void	***write**(**DataOutput** out)* *Serialize the fields of this object to out.*

NullWritable Class:

Singleton Writable with no data.

Methods:

Modifier and Type	Method and Description
int	***compareTo**(**NullWritable** other)*
boolean	***equals**(**Object** other)*
*static **NullWritable***	***get**()* *Returns the single instance of this class.*
int	***hashCode**()*
void	***readFields**(**DataInput** in)* *Deserialize the fields of this object from in.*
String	***toString**()*
void	***write**(**DataOutput** out)* *Serialize the fields of this object to out.*

ObjectWritable Class:

A polymorphic Writable that writes an instance with its class name. Handles arrays, strings and primitive types without a Writable wrapper.

Constructors:

SNo	Constructor and Description
1	*ObjectWritable()*
2	*ObjectWritable(Class declaredClass, Object instance)*
3	*ObjectWritable(Object instance)*

Methods:

Modifier and Type	*Method and Description*
Object	***get**()* *Return the instance, or null if none.*
Configuration	***getConf**()* *Return the configuration used by this object.*
Class	***getDeclaredClass**()* *Return the class this is meant to be.*
*static **Class**<?>*	***loadClass**(**Configuration** conf, **String** className)* *Find and load the class with given name className by first finding it in the specified conf.*
void	***readFields**(**DataInput** in)* *Deserialize the fields of this object from in.*

static ***Object***	***readObject****(****DataInput*** *in,* ***Configuration*** *conf)* *Read a* ***Writable, String,*** *primitive type, or an array of the preceding.*
static ***Object***	***readObject****(****DataInput*** *in,* ***ObjectWritable*** *objectWritable,* ***Configuration*** *conf)* *Read a* ***Writable, String,*** *primitive type, or an array of the preceding.*
void	***set****(****Object*** *instance)* *Reset the instance.*
void	***setConf****(****Configuration*** *conf)* *Set the configuration to be used by this object.*
String	***toString****()*
void	***write****(****DataOutput*** *out)* *Serialize the fields of this object to out.*
static void	***writeObject****(****DataOutput*** *out,* ***Object*** *instance,* ***Class*** *declaredClass,* ***Configuration*** *conf)* *Write a* ***Writable, String,*** *primitive type, or an array of the preceding.*
static void	***writeObject****(****DataOutput*** *out,* ***Object*** *instance,* ***Class*** *declaredClass,* ***Configuration*** *conf, boolean allowCompactArrays)* *Write a* ***Writable, String,*** *primitive type, or an array of the preceding.*

UNIT-4

Hadoop Architecture: Hadoop: RDBMS Vs Hadoop, Hadoop Overview, Hadoop distributors, HDFS, HDFS Daemons, Anatomy of File Write and Read., Name Node, Secondary Name Node, and Data Node, HDFS Architecture, Hadoop Configuration, Map Reduce Framework, Role of HBase in Big Data processing, HIVE,PIG.

HADOOP

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines each offering local computation and storage.

At its core, Hadoop has two major layers namely:

(a) Processing/Computation layer (MapReduce) and

(b) Storage layer (Hadoop Distributed File System).

MapReduce Framework:

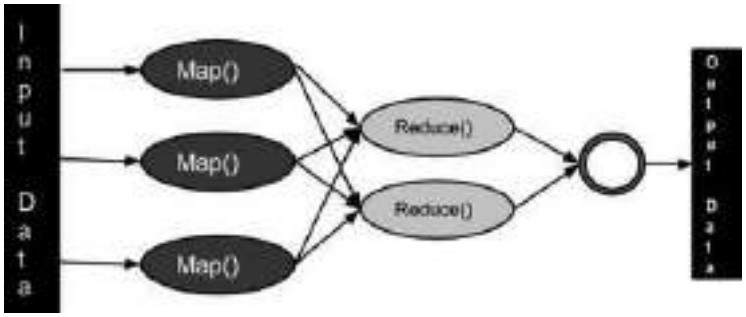
A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage**: The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage**: This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (Input) <k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3>(Output).

	Input	**Output**
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Hadoop Distributed File System (HDFS):

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low-cost hardware.

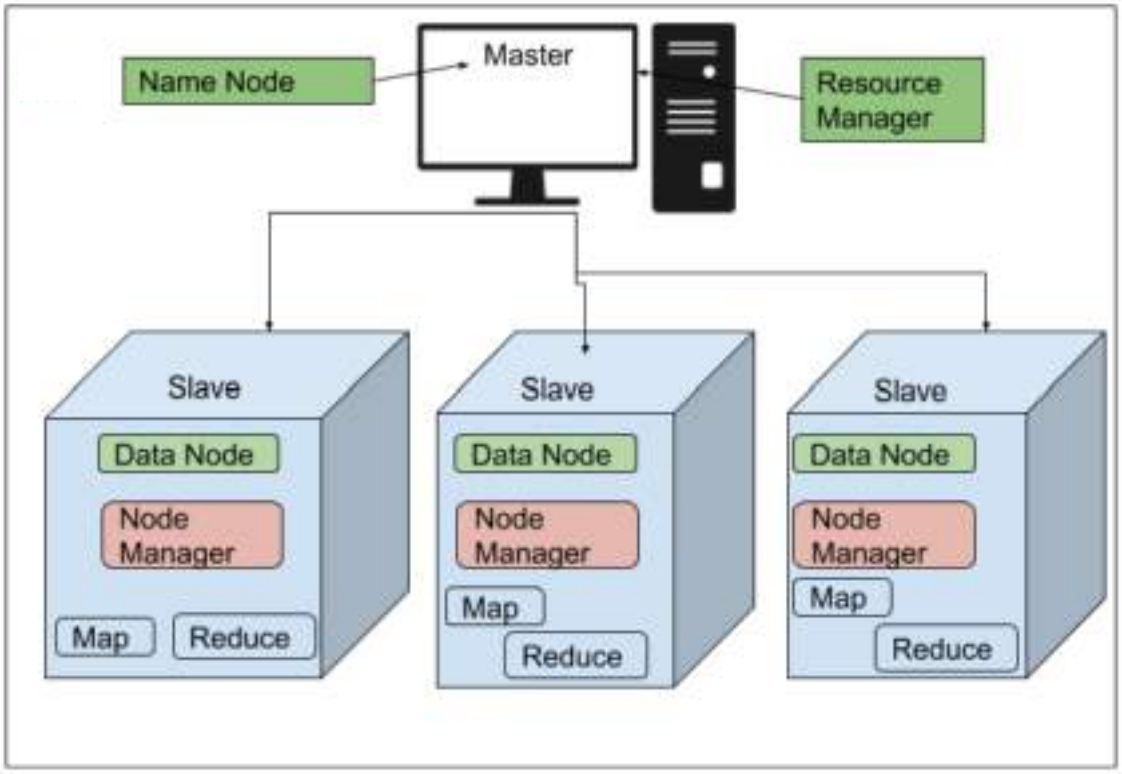
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of NameNode and DataNode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

HDFS Architecture:

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

NameNode

The NameNode is the commodity hardware that contains the GNU/Linux operating system and the NameNode software. It is a software that can be run on commodity hardware. The system having the NameNode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

DataNode

The DataNode is a commodity hardware having the GNU/Linux operating system and DataNode software. For every node (Commodity hardware/System) in a cluster, there will be a DataNode. These nodes manage the data storage of their system.

- DataNodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the NameNode.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

Goals of HDFS

- **Fault detection and recovery**: Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.
- **Huge datasets**: HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
- **Hardware at data**: A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

Job Tracker and Task Tracker:

- The primary function of the job tracker is resource management (managing the task trackers), tracking resource availability and task life cycle management (tracking its progress, fault tolerance etc.)
- The task tracker has a simple function of following the orders of the job tracker and updating the job tracker with its progress status periodically.
- The task tracker is pre-configured with a number of slots indicating the number of tasks it can accept.
- When the job tracker tries to schedule a task, it looks for an empty slot in the task tracker running on the same server which hosts the DataNode where the data for that task resides. If not found, it looks for the machine in the same rack. There is no consideration of system load during this allocation.
- HDFS is rack aware in the sense that the NameNode and the job tracker obtain a list of rack ids corresponding to each of the slave nodes (data nodes) and creates a mapping between the IP address and the rack id.
- HDFS uses this knowledge to replicate data across different racks so that data is not lost in the event of a complete rack power outage or switch failure.

Job Performance - Hadoop does speculative execution where if a machine is slow in the cluster and the map/reduce tasks running on this machine are holding on to the entire map/reduce phase, then it runs redundant jobs on other machines to process the same task and whichever task gets completed first reports back to the job tracker and results from the same are carried forward into the next phase.

Fault Tolerance –

- The task tracker spawns different JVM processes to ensure that process failures do not bring down the task tracker.
- The task tracker keeps sending heartbeat messages to the job tracker to say that it is alive and to keep it updated with the number of empty slots available for running more tasks.
- From version 0.21 of Hadoop, the job tracker does some checkpointing of its work in the filesystem. Whenever, it starts up it checks what was it upto till the last CP and resumes any incomplete jobs. Earlier, if the job tracker went down, all the active job information used to get lost.
- The status and information about the job tracker and the task tracker is exposed via jetty onto a web interface.

HDFS basic Command-line file operations

1. Create a directory in HDFS at given path(s):
   Command: hadoop fs -mkdir <paths>

2. List the contents of a directory:
   Command: hadoop fs -ls <args>

3. Upload and download a file in HDFS:
   Upload:
   Command: hadoop fs -put <localsrc> <HDFS_dest_path>
   Download:
   Command: hadoop fs -get <HDFS_src> <localdst>

4. See contents of a file:
   Command: hadoop fs -cat <path[filename]>

5. Copy a file from source to destination:
   Command: hadoop fs -cp <source> <dest>

6. Copy a file from/To Local file system to HDFS:
   Command: hadoop fs -copyFromLocal <localsrc> URI
   Command: hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localsrc>

7. Move file from source to destination:
   Command: hadoop fs -mv <src> <dest>

8. Remove a file or directory in HDFS:
   Remove files specified as argument. Delete directory only when it is empty.
   Command: hadoop fs -rm <arg>
   Recursive version of delete
   Command: hadoop fs -rmr <arg>

9. Display last few lines of a file:
   Command: hadoop fs -tail <path[filename]>

10. Display the aggregate length of a file:
    Command: hadoop fs -du <path>

11. Getting help:
    Command: hadoop fs -help

Adding files and directories:

- Creating a directory
  Command: hadoop fs -mkdir input/

- Copying the files from localfile system to HDFS
  Command: hadoop fs -put inp/file01 input/

Retrieving files:

Command: hadoop fs -get input/file01 localfs

Deleting files and directories:

Command: hadoop fs -rmr input/file01

Using HDFS in MapReduce

The HDFS is a powerful companion to Hadoop MapReduce. By setting the fs.default.name configuration option to point to the NameNode (as was done above), Hadoop MapReduce jobs will automatically draw their input files from HDFS. Using the regular FileInputFormat subclasses, Hadoop will automatically draw its input data sources from file paths within HDFS, and will distribute the work over the cluster in an intelligent fashion to exploit block locality where possible.

Configure XML files:

The HDFS configuration is located in a set of XML files in the Hadoop configuration directory; `conf/` under the main Hadoop install directory (where you unzipped Hadoop to). The `conf/hadoop-defaults.xml` file contains default values for every parameter in Hadoop. This file is considered read-only. You override this configuration by setting new values in `conf/hadoop-site.xml`. This file should be replicated consistently across all machines in the cluster. (It is also possible, though not advisable, to host it on NFS.)

Configuration settings are a set of key-value pairs of the format:

```
<property>
  <name>property-name</name>
  <value>property-value</value>
</property>
```

Adding the line `<final>true</final>` inside the `property` body will prevent properties from being overridden by user applications. This is useful for most system-wide configuration options.

fs.default.name - This is the URI (protocol specifier, hostname, and port) that describes the NameNode for the cluster. Each node in the system on which Hadoop is expected to operate needs to know the address of the NameNode. The DataNode instances will register with this NameNode, and make their data available through it. Individual client programs will connect to this address to retrieve the locations of actual file blocks.

dfs.data.dir - This is the path on the local file system in which the DataNode instance should store its data. It is not necessary that all DataNode instances store their data under the same local path prefix, as they will all be on separate machines; it is acceptable that these machines are heterogeneous. However, it will simplify configuration if this directory is standardized throughout the system. By default, Hadoop will place this under `/tmp`. This is fine for testing purposes, but is an easy way to lose actual data in a production system, and thus must be overridden.

dfs.name.dir - This is the path on the local file system of the NameNode instance where the NameNode metadata is stored. It is only used by the NameNode instance to find its information, and does not exist on the DataNodes. The caveat above about `/tmp` applies to this as well; this setting must be overridden in a production system.

dfs.replication: This is the default replication factor for each block of data in the file system. For a production cluster, this should usually be left at its default value of 3. (You are free to increase your replication factor, though this may be unnecessary and use more space than is required. Fewer than three replicas impact the high availability of information, and possibly the reliability of its storage.)

Running of Hadoop:

Hadoop can run on three modes

a) Standalone mode
b) Pseudo mode
c) Fully distributed mode

The software requirements for hadoop installation are

- Linux Operating System
- Java Development Kit
- Hadoop framework
- Secured shell

A) STANDALONE MODE:

- *Installation of Java*

  Command: *sudo apt update*

  Command: *sudo apt install default-jdk*

- *Download and extract Hadoop*

  Download Hadoop 2.7.0 from following link

  https://archive.apache.org/dist/hadoop/common/hadoop-2.7.0/hadoop-2.7.0.tar.gz

  Extract downloaded zip file

  Right click on file and click Extract Here

  Move Hadoop folder into /usr/lib

  Command: *sudo mv hadoop-2.7.0 /usr/lib/hadoop*

- *Set the path for java and hadoop*

  Command: *sudo gedit $HOME/.bashrc*

  export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

  export PATH=$PATH:$JAVA_HOME/bin

  export HADOOP_COMMON_HOME=/usr/lib/hadoop
  export HADOOP_MAPRED_HOME=/usr/lib/hadoop
  export PATH=$PATH: /usr/lib/hadoop/bin

  export PATH=$PATH: /usr/lib/hadoop/Sbin

- *Checking of java and hadoop*

  Command: *java --version*

  Command: *hadoop version*

B) PSEUDO Distributed MODE:

Hadoop single node cluster runs on single machine. The namenodes and datanodes are performing on the one machine. The installation and configuration steps as given below:

- *Installation of secured shell*

  Command: *sudo apt-get install openssh-server*

- *Create a ssh key for passwordless ssh configuration*
  Command: *ssh-keygen -t rsa –P ""*

- *Moving the key to authorized key*
  Command: *cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys*

  */**********RESTART THE COMPUTER**********/*

- *Checking of secured shell login*
  Command: *ssh localhost*

- *Add JAVA_HOME directory in hadoop-env.sh file*
  Command: *sudo gedit /usr/lib/hadoop/conf/hadoop-env.sh*

  export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

- *Creating namenode and datanode directories for hadoop*
  Command: *sudo mkdir -p /usr/lib/hadoop/dfs/namenode*
  Command: *sudo mkdir -p /usr/lib/hadoop/dfs/datanode*

- *Change permissions for namenode and datanode directories for hadoop*
  Command: *sudo chown -R sriindu:sriindu /usr/lib/hadoop/dfs/namenode*
  Command: *sudo chown -R sriindu:sriindu /usr/lib/hadoop/dfs/datanode*

- *Configure core-site.xml*
  Command: *sudo gedit /usr/lib/hadoop/etc/hadoop/core-site.xml*

```
<property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:8020</value>
</property>
```

- *Configure hdfs-site.xml*
  Command: *sudo gedit /usr/lib/hadoop/etc/hadoop/hdfs-site.xml*

```
<property>
        <name>dfs.replication</name>
        <value>1</value>
</property>

<property>
        <name>dfs.permissions</name>
        <value>false</value>
</property>

<property>
        <name>dfs.namenode.name.dir</name>
        <value>/usr/lib/hadoop/dfs/namenode</value>
</property>

<property>
        <name>dfs.datanode.data.dir</name>
        <value>/usr/lib/hadoop/dfs/datanode</value>
</property>
```

- *Configure mapred-site.xml*
  Command: *sudo gedit /usr/lib/hadoop/etc/hadoop /mapred-site.xml*
  <property>
  <name>mapred.job.tracker</name>
  <value>localhost:8021</value>
  </property>

- *Format the name node*
  Command: *hadoop namenode -format*

- *Start the namenode, datanode*
  Command: *start-dfs.sh*

- *Start the Node Manager and Resource Manager*
  Command: *start-mapred.sh*

- *To check if Hadoop started correctly*
  Command: *jps*
  NameNode
  SecondaryNamenode
  DataNode
  NodeManager
  ResourceManager
  jps

HBase

Apache HBase is a distributed, scalable, NoSQL database built on top of Apache Hadoop. It is designed to store and manage very large amounts of structured and semi-structured data across many servers.

HBase is modeled after Google's **Bigtable** and is suitable for applications requiring:

- Huge datasets
- Fast read/write access
- Real-time data processing
- Distributed storage

Features of HBase

- Column-oriented database
- Distributed and scalable
- Supports billions of rows and millions of columns
- Fault tolerant using Hadoop Distributed File System (HDFS)
- Real-time read/write access
- Automatic sharding using regions
- High availability

Architecture of HBase

Main components:

1. **HMaster**
   - Manages the HBase cluster
   - Handles region allocation and load balancing

2. **Region Server**
    - Stores and manages table regions
    - Handles read/write requests
3. **ZooKeeper**
    - Coordinates distributed services
    - Maintains server status
4. **HDFS**
    - Stores actual HBase data files

Role of HBase in Hadoop

HBase plays an important role in the Hadoop ecosystem by providing **real-time database capabilities**.

Why Hadoop Alone is Not Enough

Apache Hadoop mainly uses HDFS, which is excellent for:

- Batch processing
- Large-scale storage
- Sequential data access

But Hadoop HDFS has limitations:

- Slow random access
- Not suitable for real-time applications
- Cannot efficiently update records

Role of HBase

1. Real-Time Data Access

HBase allows:

- Fast random reads
- Fast writes
- Real-time querying

Example:

- Social media feeds
- Banking transactions
- Sensor data

2. Works on Top of HDFS

HBase stores data in:

- HDFS blocks
- Distributed nodes

This provides:

- Reliability
- Fault tolerance
- Scalability

3. Handles Big Data Efficiently

HBase can store:

- Terabytes
- Petabytes of data

across many commodity servers.

4. Column-Oriented Storage

Unlike traditional relational databases:

RDBMS	HBase
Row-based	Column-family based
Fixed schema	Flexible schema
SQL	NoSQL

This improves performance for analytical workloads.

5. Integration with Hadoop Ecosystem

HBase integrates with:

- Apache Hive
- Apache Pig
- Apache Spark
- Apache MapReduce

for large-scale analytics.

Advantages of HBase

- Horizontally scalable
- Real-time processing
- High throughput
- Automatic failover
- Suitable for sparse data

Limitations of HBase

- No support for complex joins
- Limited SQL support
- Complex configuration
- Not suitable for transactional systems like banking ERP

Applications of HBase

- Social media analytics
- IoT sensor storage
- Log processing
- Recommendation systems
- Fraud detection
- Time-series data

Installation of HBase:

- Download HBase from following Link: https://archive.apache.org/dist/hbase/2.5.8/hbase-2.5.8-bin.tar.gz

Name	Last modified	Size	Description
Parent Directory		-	
CHANGES.md	2024-03-01 00:23	1.1M	
RELEASENOTES.md	2024-03-01 00:23	680K	
api_compare_2.5.7_to_2.5.8RC0.html	2024-03-01 00:23	13K	
hbase-2.5.8-bin.tar.gz	2024-03-01 00:23	308M	
hbase-2.5.8-bin.tar.gz.asc	2024-03-01 00:23	833	
hbase-2.5.8-bin.tar.gz.sha512	2024-03-01 00:23	216	
hbase-2.5.8-client-bin.tar.gz	2024-03-01 00:23	296M	
hbase-2.5.8-client-bin.tar.gz.asc	2024-03-01 00:23	833	
hbase-2.5.8-client-bin.tar.gz.sha512	2024-03-01 00:23	268	
hbase-2.5.8-hadoop3-bin.tar.gz	2024-03-01 00:23	321M	
hbase-2.5.8-hadoop3-bin.tar.gz.asc	2024-03-01 00:23	833	
hbase-2.5.8-hadoop3-bin.tar.gz.sha512	2024-03-01 00:23	272	
hbase-2.5.8-hadoop3-client-bin.tar.gz	2024-03-01 00:23	309M	
hbase-2.5.8-hadoop3-client-bin.tar.gz.asc	2024-03-01 00:23	833	
hbase-2.5.8-hadoop3-client-bin.tar.gz.sha512	2024-03-01 00:23	300	
hbase-2.5.8-src.tar.gz	2024-03-01 00:23	50M	
hbase-2.5.8-src.tar.gz.asc	2024-03-01 00:23	833	
hbase-2.5.8-src.tar.gz.sha512	2024-03-01 00:23	216	

- Extract this file using right-click -> 'Extract here'.
- Now, move this folder to /usr/lib using following
- command:

```
$ sudo mv Downloads/hbase-2.5.5  /usr/lib/hbase
```

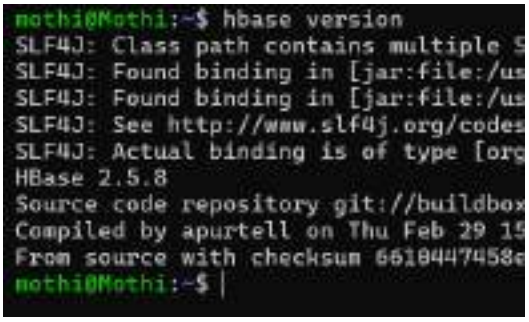
- Open the bashrc file:

```
$ sudo gedit ~/.bashrc
```

- Go to end of the file and add following lines.

```
export HBASE_HOME=/usr/lib/hbase
export PATH=$PATH:$HBASE_HOME/bin
```

- Close and open terminal again



```
mothi@Mothi:~$ hbase version
SLF4J: Class path contains multiple S
SLF4J: Found binding in [jar:file:/us
SLF4J: Found binding in [jar:file:/us
SLF4J: See http://www.slf4j.org/codes
SLF4J: Actual binding is of type [org
HBase 2.5.8
Source code repository git://buildbox
Compiled by apurtell on Thu Feb 29 15
From source with checksum 6610447458e
mothi@Mothi:~$
```

- **Create Directories and change permissions for "zookeeper" and "data"**

```
$ sudo mkdir -p /usr/lib/hbase/zookeeper
$ sudo mkdir -p /usr/lib/hbase/data
$ sudo chown -R sriindu:sriindu /usr/lib/hbase/zookeeper
$ sudo chown -R sriindu:sriindu /usr/lib/hbase/data
```

- **Edit hbase-env.sh file:**

Command: `sudo gedit /usr/lib/hbase/conf/hbase-env.sh`

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

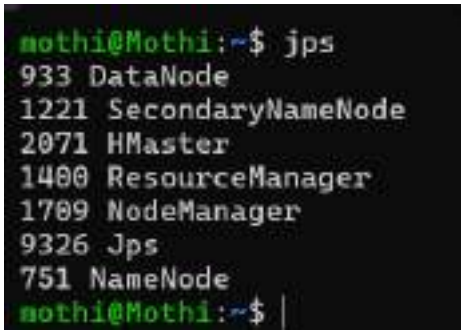
- **Edit hbase-site.xml file:**

Command: `sudo gedit /usr/lib/hbase/conf/hbase-site.xml`

```
<property>
    <name>hbase.unsafe.stream.capability.enforce</name>
    <value>false</value>
  </property>

  <property>
    <name>hbase.rootdir</name>
    <value>file:///usr/lib/hbase/data</value>
  </property>

  <property>
    <name>hbase.cluster.distributed</name>
    <value>false</value>
  </property>

  <property>
    <name>hbase.tmp.dir</name>
    <value>/usr/lib/hbase/tmp</value>
  </property>

  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/usr/lib/hbase/zookeeper</value>
  </property>
```

- Start Hadoop and Hbase

```
$ start-dfs.sh
$ start-mapred.sh
$ start-hbase.sh
$ jps
```



Store and retrieve data in HBase:

- Open Hbase using following command:

  Command: `hbase shell`

- Create student table

  Query: `create 'students','info';`

- Insert data into student table

  Query: `put 'students', '1', 'info:roll', '101';`

Query: *put 'students', '1', 'info:name', 'mothi';*
Query: *put 'students', '1', 'info:department', 'aids';*
Query: *put 'students', '1', 'info:marks', '86';*

Query: *put 'students', '2', 'info:roll', '102';*
Query: *put 'students', '2', 'info:name', 'ravi';*
Query: *put 'students', '2', 'info:department', 'aids';*
Query: *put 'students', '2', 'info:marks', '92';*

- Retrieve data from student table
  Query: *scan 'students';*

- Update data in student table
  Query: *put 'students', '2', 'info:marks', '95';*

- Drop student table
  Query: *disable 'student'*
  Query: *drop 'student'*

What is HIVE?

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive

These are the following features of Hive:

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.

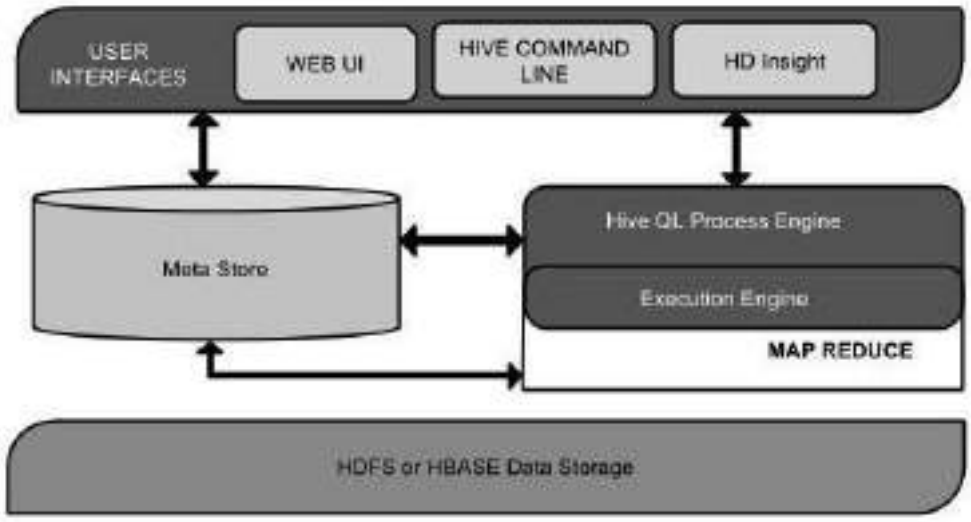
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

Limitations of Hive

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Architecture of Hive

The following component diagram depicts the architecture of Hive:

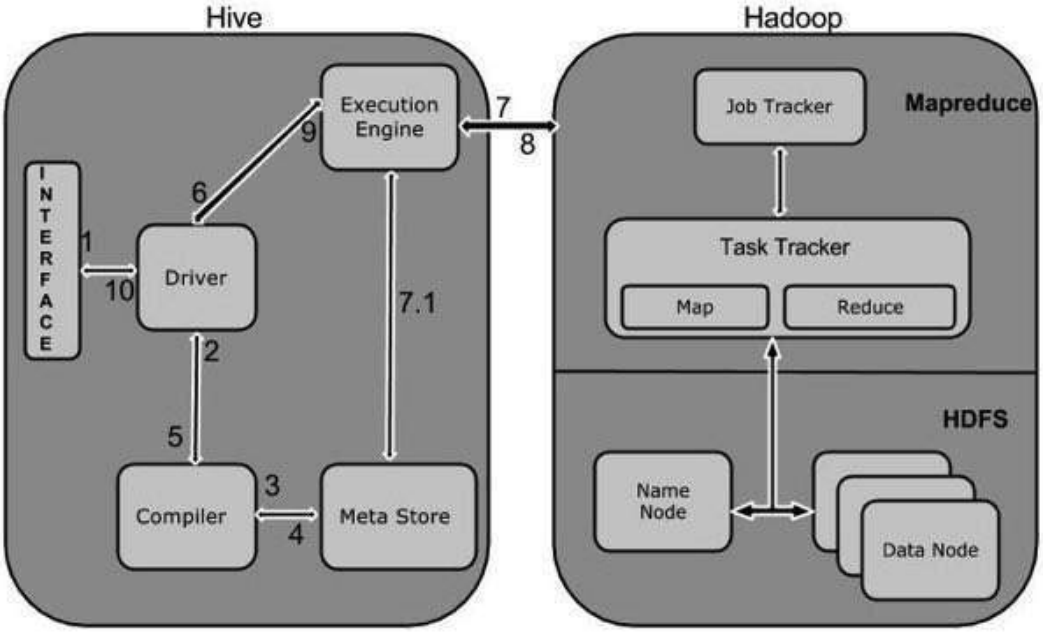


This component diagram contains different units. The following table describes each unit:

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Working of Hive

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.
5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job** Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result**

	The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

What is Apache Pig?

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Apache Pig.

To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using **Apache Pig**, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Why Do We Need Apache Pig?

Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.

- Using **Pig Latin**, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses **multi-query approach**, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- Pig Latin is **SQL-like language** and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

Features of Pig

Apache Pig comes with the following features −

- **Rich set of operators** − It provides many operators to perform operations like join, sort, filer, etc.
- **Ease of programming** − Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.
- **Optimization opportunities** − The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
- **Extensibility** − Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF's** − Pig provides the facility to create **User-defined Functions** in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data** − Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Apache Pig Vs MapReduce

Listed below are the major differences between Apache Pig and MapReduce.

Apache Pig	MapReduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid.
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce.
Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

Apache Pig Vs SQL

Listed below are the major differences between Apache Pig and SQL.

Pig	SQL
Pig Latin is a **procedural** language.	SQL is a **declarative** language.
In Apache Pig, **schema** is optional. We can store data without designing a schema (values are stored as $01, $02 etc.)	Schema is mandatory in SQL.
The data model in Apache Pig is **nested relational**.	The data model used in SQL **is flat relational**.
Apache Pig provides limited opportunity for **Query optimization**.	There is more opportunity for query optimization in SQL.

In addition to above differences, Apache Pig Latin −

- Allows splits in the pipeline.
- Allows developers to store data anywhere in the pipeline.
- Declares execution plans.
- Provides operators to perform ETL (Extract, Transform, and Load) functions.

Apache Pig Vs Hive

Both Apache Pig and Hive are used to create MapReduce jobs. And in some cases, Hive operates on HDFS in a similar way Apache Pig does. In the following table, we have listed a few significant points that set Apache Pig apart from Hive.

Apache Pig	Hive
Apache Pig uses a language called **Pig Latin**. It was originally created at **Yahoo**.	Hive uses a language called **HiveQL**. It was originally created at **Facebook**.
Pig Latin is a data flow language.	HiveQL is a query processing language.
Pig Latin is a procedural language and it fits in pipeline paradigm.	HiveQL is a declarative language.
Apache Pig can handle structured, unstructured, and semi-structured data.	Hive is mostly for structured data.

Applications of Apache Pig

Apache Pig is generally used by data scientists for performing tasks involving ad-hoc processing and quick prototyping. Apache Pig is used −

- To process huge data sources such as web logs.
- To perform data processing for search platforms.
- To process time sensitive data loads.

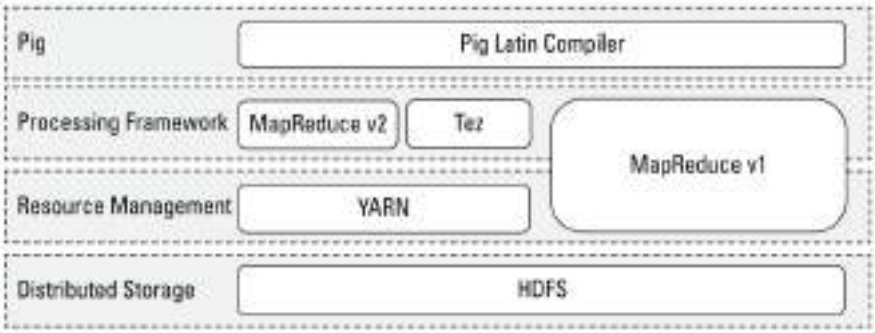
Admiring Pig Architecture:

Pig is made up of two components:

a) **The language itself:** The programming language for Pig is known as Pig Latin, a high-level language that allows you to write data processing and analysis programs.

b) **The Pig Latin compiler:** The Pig Latin compiler converts the Pig Latin code into executable code. The executable code is either in the form of MapReduce jobs or it can spawn a process where a virtual Hadoop instance is created to run the Pig code on a single node.

The sequence of MapReduce programs enables Pig programs to do data processing and analysis in parallel, leveraging Hadoop MapReduce and HDFS. Running the Pig job in the virtual Hadoop instance is a useful strategy for testing your Pig scripts. Figure 1 shows how Pig relates to the Hadoop ecosystem.



Pig programs can run on MapReduce v1 or MapReduce v2 without any code changes, regardless of what mode your cluster is running. However, Pig scripts can also run using the Tez API instead. Apache Tez provides a more efficient execution framework than MapReduce. YARN enables application frameworks other than MapReduce (like Tez) to run on Hadoop. Hive can also run against the Tez framework.

Apache Pig Detailed Architecture:

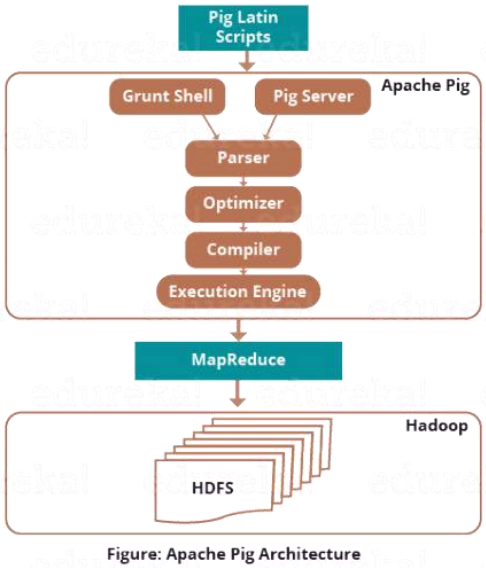


Figure: Apache Pig Architecture

Pig Latin Scripts

Initially as illustrated in the above image, we submit Pig scripts to the Apache Pig execution environment which can be written in Pig Latin using built-in operators.

There are three ways to execute the Pig script:

- ***Grunt Shell*:** This is Pig's interactive shell provided to execute all Pig Scripts.
- ***Script File*:** Write all the Pig commands in a script file and execute the Pig script file. This is executed by the Pig Server.
- ***Embedded Script*:** If some functions are unavailable in built-in operators, we can programmatically create User Defined Functions to bring that functionalities using other languages like Java, Python, Ruby, etc. and embed it in Pig Latin Script file**.** Then, execute that script file.

Parser

From the above image you can see, after passing through Grunt or Pig Server, Pig Scripts are passed to the Parser. The Parser does type checking and checks the syntax of the script. The parser outputs a DAG (directed acyclic graph). DAG represents the Pig Latin statements and logical operators. The logical operators are represented as the nodes and the data flows are represented as edges.

Optimizer

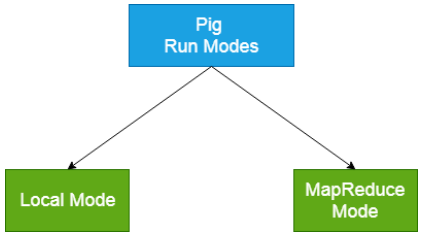
Then the DAG is submitted to the optimizer. The Optimizer performs the optimization activities like split, merge, transform, and reorder operators etc. This optimizer provides the automatic optimization feature to Apache Pig. The optimizer basically aims to reduce the amount of data in the pipeline at any instance of time while processing the extracted data, and for that it performs functions like:

- *PushUpFilter*: If there are multiple conditions in the filter and the filter can be split, Pig splits the conditions and pushes up each condition separately. Selecting these conditions earlier, helps in reducing the number of records remaining in the pipeline.
- *PushDownForEachFlatten*: Applying flatten, which produces a cross product between a complex type such as a tuple or a bag and the other fields in the record, as late as possible in the plan. This keeps the number of records low in the pipeline.
- *ColumnPruner*: Omitting columns that are never used or no longer needed, reducing the size of the record. This can be applied after each operator, so that fields can be pruned as aggressively as possible.
- *MapKeyPruner*: Omitting map keys that are never used, reducing the size of the record.
- *LimitOptimizer*: If the limit operator is immediately applied after a load or sort operator, Pig converts the load or sort operator into a limit-sensitive implementation, which does not require processing the whole data set. Applying the limit earlier, reduces the number of records.

This is just a flavor of the optimization process. Over that it also performs **Join**, **Order By** and **Group By** functions.

Apache Pig Running Modes:

Apache Pig executes in two modes: Local Mode and MapReduce Mode.



Local Mode

- It executes in a single JVM and is used for development experimenting and prototyping.
- Here, files are installed and run using localhost.
- The local mode works on a local file system. The input and output data stored in the local file system.

The command for local mode grunt shell:

$ pig-x local

MapReduce Mode

- The MapReduce mode is also known as Hadoop Mode.
- It is the default mode.
- In this Pig renders Pig Latin into MapReduce jobs and executes them on the cluster.
- It can be executed against semi-distributed or fully distributed Hadoop installation.
- Here, the input and output data are present on HDFS.

The command for Map reduce mode:

$ pig

or

$ pig -x mapreduce

Installation of Pig

- Download the tar.gz file of Apache Pig from here: *https://downloads.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz*

- Extract this file using right-click -> 'Extract here'.
- Now, move this folder to /usr/lib using following
- command:

  $ sudo mv Downloads/pig-0.16.0 /usr/lib/pig

- Open the bashrc file:

  $ sudo gedit ~/.bashrc

- Go to end of the file and add following lines.

  export PIG_HOME=/usr/lib/pig

```
export PATH=$PATH:$PIG_HOME/bin
```

- Close and open terminal again

```
mothi@Mothi:~$ pig -version
Apache Pig version 0.16.0 (r1746530)
compiled Jun 01 2016, 23:10:49
mothi@Mothi:~$
```

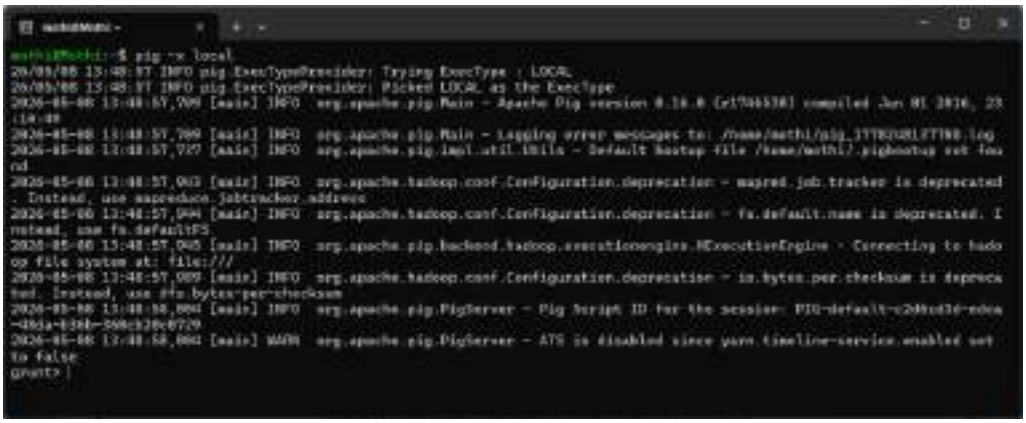
Start the Pig

- Start the pig in local mode:

```
pig -x local
```

- Start the pig in mapreduce mode (needs hadoop datanode started):

```
pig -x mapreduce
```



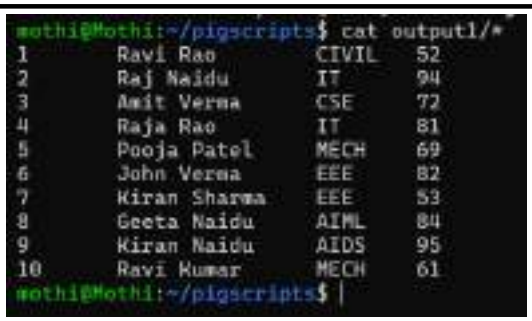
Sample Student dataset

ROLL	NAME	DEPARTMENT	MARKS
1	Ravi Rao	CIVIL	52
2	Raj Naidu	IT	74
3	Amit Verma	CSE	72
4	Raj Rao	IT	67
5	Pooja Patel	MECH	69
6	John Verma	EEE	62
7	Kiran Sharma	EEE	53
8	Geeta Naidu	AIML	54
9	Kiran Naidu	AIDS	57
10	Ravi Kumar	MECH	61

Programs:

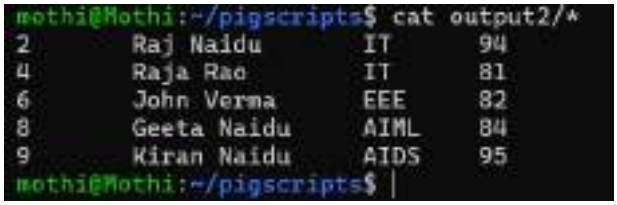
1. **a) Load the student data using Pig.**
   b) Display all records.
   c) Store filtered results into output directory.

```
$gedit 1.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        dump students;
        store students into 'output1';
$pig -x local 1.pig
```



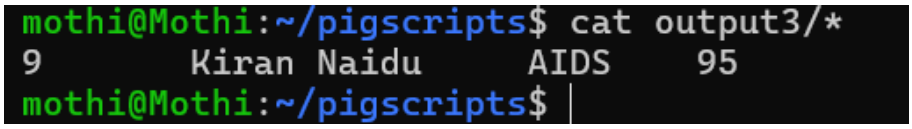
2. a) Load the student data using Pig.
b) Filter students with marks > 80.
c) Store filtered results into output directory.

```
$gedit 2.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        marks_greater_than_80 = FILTER students by marks>80;
        store marks_greater_than_80 into 'output2';
$pig -x local 2.pig
```



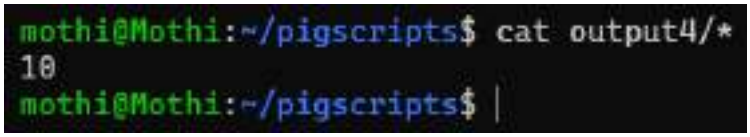
3. a) Load the student data using Pig.
b) Find students from AIDS department.
c) Store filtered results into output directory.

```
$gedit 3.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        students_aids = FILTER students by department matches 'aids';
        store students_aids into 'output3';
$pig -x local 3.pig
```



4. a) Load the student data using Pig.
b) Count total number of students.
c) Store filtered results into output directory.

```
$gedit 4.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        grouped = GROUP students ALL;
        students_count = FOREACH grouped GENERATE COUNT(students);
        store students_count into 'output4';
$pig -x local 4.pig
```



5. a) Load the student data using Pig.
b) Find average marks.
c) Store filtered results into output directory.

```
$gedit 5.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        grouped = GROUP students ALL;
        marks_avg = FOREACH grouped GENERATE AVG(students.marks);
        store marks_avg into 'output5';
$pig -x local 5.pig
```

```
mothi@Mothi:~/pigscripts$ cat output5/*
74.3
mothi@Mothi:~/pigscripts$
```

6. a) Load the student data using Pig.
b) Sort students by marks in descending order.
c) Store filtered results into output directory.

```
$gedit 6.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        marks_ord = ORDER students BY marks DESC;
        store marks_ord into 'output6';
$pig -x local 6.pig
```

```
mothi@Mothi:~/pigscripts$ cat output6/*
9       Kiran Naidu     AIDS    95
2       Raj Naidu       IT      94
8       Geeta Naidu     AIML    84
6       John Verma      EEE     82
4       Raja Rao        IT      81
3       Amit Verma      CSE     72
5       Pooja Patel     MECH    69
10      Ravi Kumar      MECH    61
7       Kiran Sharma    EEE     53
1       Ravi Rao        CIVIL   52
mothi@Mothi:~/pigscripts$
```

7. a) Load the student data using Pig.
b) Group students by department.
c) Store filtered results into output directory.

```
$gedit 7.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        dept_grp = GROUP students BY department;
        store dept_grp into 'output7';
$pig -x local 7.pig
```

```
mothi@Mothi:~/pigscripts$ cat output7/*
IT      {(4,Raja Rao,IT,81),(2,Raj Naidu,IT,94)}
CSE     {(3,Amit Verma,CSE,72)}
EEE     {(7,Kiran Sharma,EEE,53),(6,John Verma,EEE,82)}
AIDS    {(9,Kiran Naidu,AIDS,95)}
AIML    {(8,Geeta Naidu,AIML,84)}
MECH    {(10,Ravi Kumar,MECH,61),(5,Pooja Patel,MECH,69)}
CIVIL   {(1,Ravi Rao,CIVIL,52)}
mothi@Mothi:~/pigscripts$
```

8. a) Load the student data using Pig.
b) Find maximum marks in each department.
c) Store filtered results into output directory.

```
$gedit 8.pig
        students = LOAD 'studentmarks.csv' USING PigStorage(',') as (roll:int,
        name:chararray, department:chararray, marks:int);
        dept_grp = GROUP students BY department;
        max_marks_per_dept = FOREACH dept_grp GENERATE group AS department,
        MAX(students.marks) AS max_marks;
        store max_marks_per_dept into 'output8';
$pig -x local 8.pig
```

UNIT-5

Data Analytics with R, Machine Learning: Introduction, Supervised Learning, Unsupervised Learning, Collaborative Filtering, Social Media Analytics, Mobile Analytics, Big Data Analytics with Big R.

Machine Learning:

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that enables computers to learn from data and make decisions or predictions without being explicitly programmed.

Instead of writing fixed rules, ML systems analyze patterns in data and improve their performance automatically over time.

Applications of Machine Learning

- Email spam filtering
- Recommendation systems
- Face recognition
- Voice assistants
- Fraud detection
- Medical diagnosis
- Social media analysis
- Mobile app analytics

Types of Machine Learning

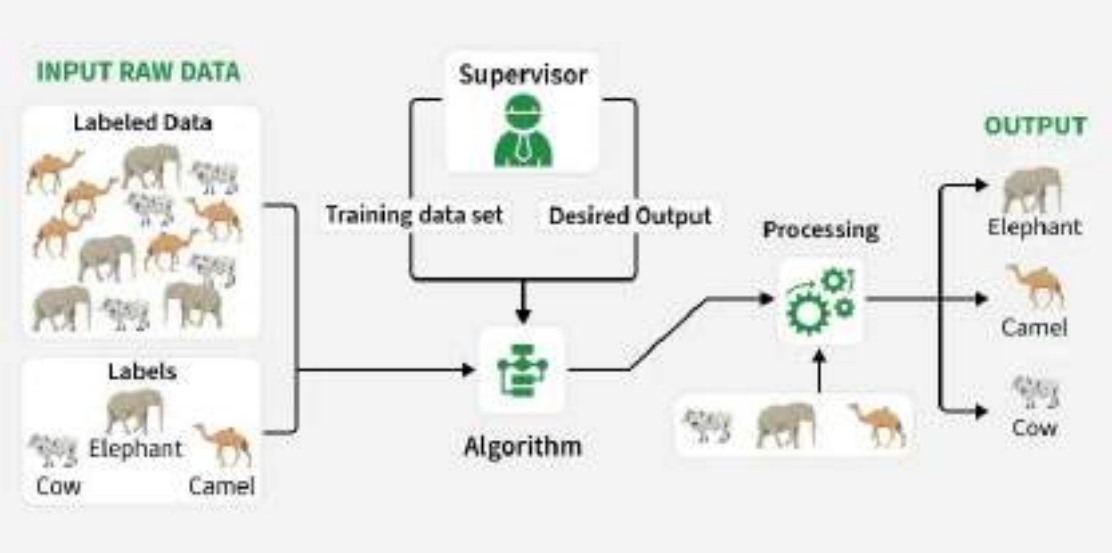
Machine Learning is mainly divided into:

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

1. Supervised Learning:

Supervised learning algorithms are generally categorized into two main types:

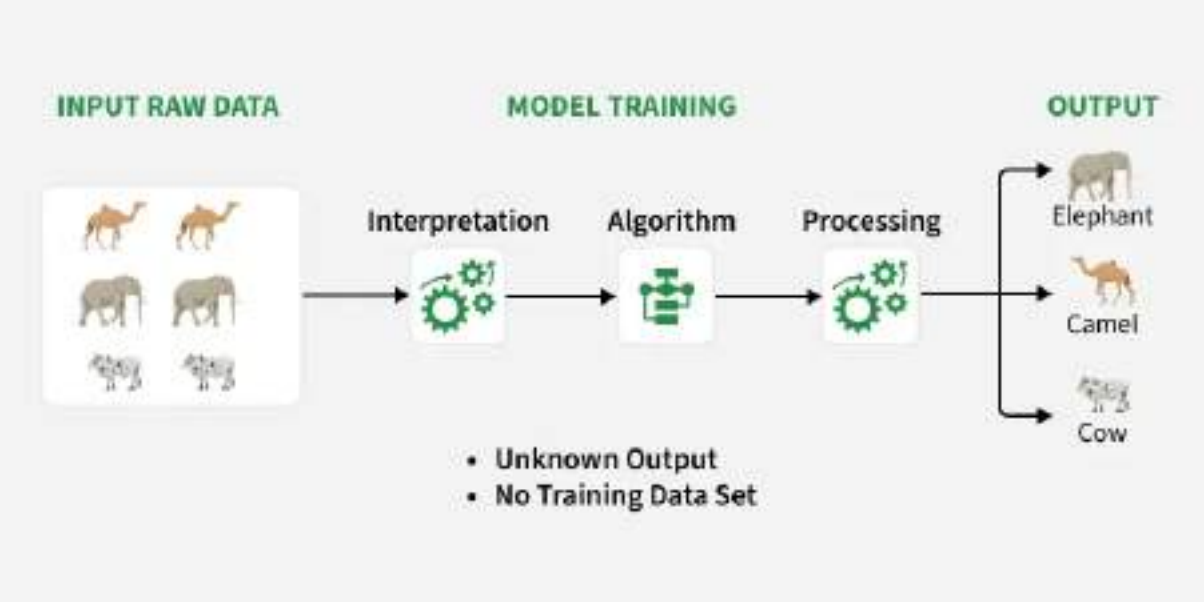
- Classification: where the goal is to predict discrete labels or categories
- Regression: where the aim is to predict continuous numerical values.



2. Unsupervised learning

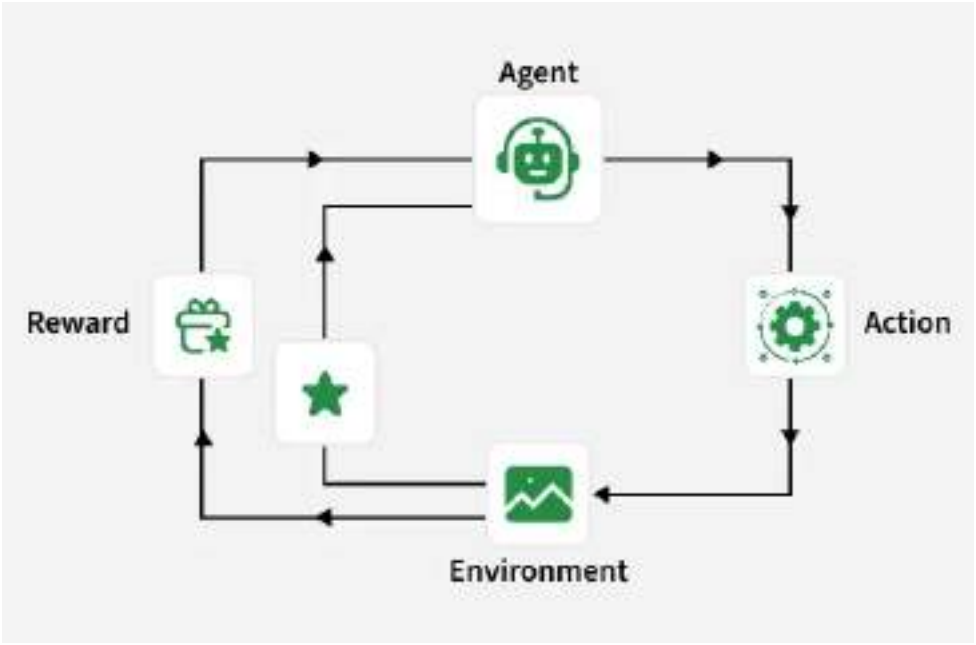
Unsupervised learning are again divided into three main categories based on their purpose:

- Clustering
- Association Rule Mining
- Dimensionality Reduction



3. Reinforcement Learning

Reinforcement learning interacts with environment and learn from them based on rewards.



Collaborative Filtering

A recommendation system analyses user behaviour and past activity to understand preferences and suggest relevant content, products or ideas. By tracking what users watch, click or interact with, it identifies patterns and continuously improves recommendations to enhance user experience and engagement.

- Tracks user activity like views, clicks and interactions
- Identifies patterns in user preferences
- Predicts likes and dislikes based on past behavior
- Recommends similar or relevant content
- Continuously updates suggestions with new data
- Improves personalization and user engagement

Types of Collaborative Filtering Techniques

- **Memory-Based:** Uses user-item data directly to make recommendations
- **Model-Based:** Builds predictive models using machine learning
- **Hybrid:** Combines multiple approaches for better results
- **Deep Learning:** Uses neural networks for more advanced recommendations

Measuring Similarity in Collaborative Filtering

Collaborative filtering works by comparing user preferences and identifying similarities in their ratings. Based on these similarities, the system predicts what a user might like or dislike and recommends items accordingly. Example:

Users	Movie 1	Movie 2	Movie 3	Movie 4
Users 1	5	4		5
Users 2	4		3	
Users 3		1		2
Users 4	1	2		

- User 1 and User 2 have nearly similar ratings (both liked Movie 1), showing similar preferences
- Based on this, Movie 3 (liked by User 2) can be recommended to User 1
- Similarly, Movie 4 (liked by User 1) can be recommended to User 2
- User 1 and User 3 have opposite tastes, so their recommendations will differ
- User 3 and User 4 share similar low ratings for Movie 2
- Since User 3 disliked Movie 4, it can be predicted that User 4 may also dislike Movie 4

Cosine Similarity in Collaborative Filtering

Cosine similarity measures how similar two users are based on their ratings. A higher cosine value means users have similar preferences, while a lower value means they are different. Missing values are often treated as 0 to simplify calculations.

- Measures similarity between users using their rating patterns
- Higher cosine value means more similar users
- Lower cosine value means less similar users
- Missing ratings can be filled with 0 for easy calculation
- Helps in recommending items liked by similar users

$$\text{similarity} = \frac{A \cdot B}{|A| \times |B|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Rounding the Data

Rounding is used to simplify rating data by converting it into binary values. Ratings below 3 are set to 0 (dislike), and ratings 3 or above are set to 1 (like). This makes comparison between users faster and easier.

- Converts complex ratings into simple 0 and 1 values
- Improves readability of the data
- Makes similarity comparison more efficient
- Helps clearly identify similar user groups

Example:

Users	Movie 1	Movie 2	Movie 3	Movie 4
Users 1	1	1		1
Users 2	1		1	
Users 3		0		0
Users 4	0	0		

- After rounding, User 1 and User 2 show similar patterns (more 1s) which means similar preferences
- User 3 and User 4 show similar patterns (more 0s) which means similar dislikes

Normalizing Rating

Normalization adjusts user ratings by subtracting each user's average rating from their given ratings. This converts values into positive and negative scores, making it easier to compare user preferences fairly.

- Subtracts the user's average rating from each rating
- Produces positive (above average) and negative (below average) values
- Removes bias of users who rate consistently high or low
- Helps group users with similar rating patterns
- Improves accuracy of recommendations by better similarity detection

Advantages

- Unlike content-based systems, it does not rely on limited item features making it more flexible in different use cases
- It can handle a wide variety of data since it learns from user interactions instead of predefined content
- It provides strong personalization by recommending items based on similar users' preferences
- It adapts easily to changes in user behavior over time, improving recommendations continuously
- It performs well when large amounts of user data are available, increasing accuracy and relevance

Challenges

- As the number of users and items increases, computation and storage requirements grow significantly, making the system slower
- Scalability becomes a major issue with large datasets, affecting performance and accuracy
- Relies heavily on historical data, so it may struggle when there is limited or new user data
- Tends to recommend similar types of items repeatedly, reducing diversity in recommendations
- May not capture changing interests instantly if recent data is limited

Social Media Analytics:

Social Media Analytics in Big Data refers to the process of collecting, processing, analyzing, and interpreting huge volumes of data generated from social media platforms to extract meaningful insights and support decision-making.

Social media platforms generate massive amounts of structured and unstructured data every second in the form of:

- Posts
- Comments
- Likes
- Shares
- Videos
- Images
- Hashtags
- User interactions

This large-scale data is analyzed using Big Data technologies and analytics tools.

Social media has become one of the largest sources of Big Data. Platforms such as:

- Facebook
- Instagram
- X
- YouTube
- LinkedIn

produce terabytes of data daily.

Organizations use Social Media Analytics to:

- Understand customer behavior
- Analyze market trends
- Improve business strategies
- Monitor brand reputation
- Detect public sentiment

Sources of Social Media Data

1. User Generated Content

- Posts
- Tweets
- Status updates
- Comments

2. Multimedia Content

- Images
- Videos
- Live streams

3. Interaction Data

- Likes
- Shares
- Followers
- Reactions

4. Location-Based Data

- Check-ins
- Geo-tags
- GPS data

Architecture of Social Media Analytics

Step 1: Data Collection

Data is collected using:

- APIs
- Web scraping
- Streaming services

Examples:

- Apache Kafka
- Apache Flume

Step 2: Data Storage

Large volumes of data are stored using Big Data storage systems such as:

- Hadoop Distributed File System (HDFS)
- HBase
- MongoDB

Step 3: Data Processing

Processing frameworks analyze massive datasets.

Technologies used:

- Apache Hadoop
- Apache Spark
- MapReduce

Step 4: Data Analysis

Different analytics methods are applied:

- Sentiment analysis
- Trend analysis
- Predictive analytics
- Network analysis

Step 5: Visualization

Results are displayed using dashboards and charts.

Visualization tools:

- Tableau
- Power BI
- R

Types of Social Media Analytics

1. Sentiment Analysis

Determines whether user opinions are:

- Positive
- Negative
- Neutral

Example:

Analyzing customer opinions about a product launch.

2. Trend Analysis

Identifies trending:

- Topics
- Hashtags
- Events

Example:

Tracking viral hashtags during elections or sports events.

3. Network Analysis

Studies relationships between users and communities.

Applications:

- Influencer identification
- Community detection

4. Predictive Analytics

Uses historical social media data to predict future outcomes.

Example:

Predicting customer purchasing behavior.

5. Content Analysis

Analyzes text, images, and videos for patterns and engagement.

Applications of Social Media Analytics

Business and Marketing

- Brand monitoring
- Customer feedback analysis
- Product improvement
- Advertisement targeting

Healthcare

- Disease outbreak monitoring
- Public health awareness analysis

Education

- Student sentiment analysis
- Online learning feedback

Politics

- Election campaign analysis
- Public opinion tracking

Entertainment

- Movie popularity analysis
- Audience engagement measurement

Advantages

- Real-time insights
- Better decision-making
- Improved customer engagement
- Market trend detection
- Competitive analysis

Mobile Analytics:

Mobile Analytics in Big Data refers to the collection, processing, analysis, and interpretation of large volumes of data generated from mobile devices and mobile applications.

Mobile devices such as smartphones and tablets continuously generate massive amounts of data through:

- Mobile apps
- Internet usage
- GPS location
- User interactions
- Sensors
- Transactions

With the rapid growth of smartphones and mobile applications, mobile data has become one of the largest sources of Big Data.

Popular mobile platforms include:

- Android
- iOS

Organizations use Mobile Analytics to:

- Understand user behavior
- Improve app performance
- Increase customer engagement
- Deliver personalized services
- Optimize mobile marketing

Sources of Mobile Data

1. Mobile Applications

- App usage
- Click events
- Session duration
- Screen navigation

2. Device Sensors

- GPS
- Accelerometer
- Gyroscope
- Camera
- Microphone

3. Mobile Transactions

- Online payments
- Banking transactions
- E-commerce purchases

4. Communication Data

- Calls
- SMS
- Notifications
- Chat messages

5. Social Media Mobile Usage

Data generated through mobile access to social media platforms.

Architecture of Mobile Analytics

Step 1: Data Collection

Data is collected from:

- Mobile applications
- APIs
- Sensors
- User interactions

Tools used:

- Firebase
- Google Analytics
- Flurry Analytics

Step 2: Data Transmission

Collected data is transmitted through:

- Mobile networks
- Wi-Fi
- Cloud services

Streaming tools:

- Apache Kafka
- Apache Flume

Step 3: Data Storage

Large-scale mobile data is stored using Big Data storage systems.

Technologies used:

- Hadoop Distributed File System
- MongoDB
- HBase

Step 4: Data Processing

Processing frameworks analyze large datasets efficiently.

Technologies:

- Apache Hadoop
- Apache Spark
- MapReduce

Step 5: Data Analysis and Visualization

Analytics results are presented using dashboards and charts.

Visualization tools:

- Power BI
- Tableau
- R

Types of Mobile Analytics

1. Usage Analytics

Tracks:

- Number of users
- Session duration
- Screen views
- Feature usage

2. Performance Analytics

Measures:

- App speed
- Crash reports
- Battery usage
- Network performance

3. Behavioral Analytics

Analyzes:

- User behavior
- Navigation patterns
- Purchase behavior

4. Location Analytics

Uses GPS and location data to provide:

- Navigation services
- Location-based advertisements
- Delivery tracking

5. Predictive Analytics

Uses historical mobile data to predict:

- User retention
- Customer churn
- Future sales

Applications of Mobile Analytics

Business

- Customer behavior analysis
- Personalized marketing
- Sales prediction

Healthcare

- Mobile health monitoring
- Fitness tracking
- Remote patient care

Education

- Mobile learning analysis
- Student engagement tracking

Transportation

- Traffic monitoring
- Vehicle tracking
- Route optimization

E-Commerce

- Product recommendations
- Customer purchase analysis

Advantages of Mobile Analytics

- Real-time insights
- Better customer experience
- Improved app performance
- Personalized services
- Better marketing strategies
- Increased business revenue

Challenges in Mobile Analytics

Challenge	Description
Privacy Issues	Protecting user personal data
Data Security	Preventing unauthorized access
Scalability	Handling large-scale mobile data
Battery Consumption	Analytics may consume device power
Data Integration	Combining data from multiple sources

Big Data Analytics with R:

Big Data Analytics with R refers to the process of analyzing large and complex datasets using the R programming language and Big Data technologies.

R is an open-source programming language mainly used for:

- Statistical analysis
- Data visualization
- Machine learning
- Predictive analytics
- Big Data processing

It provides powerful libraries and tools to handle structured and unstructured data efficiently.

In the modern digital world, organizations generate huge volumes of data from:

- Social media
- Banking systems
- Mobile devices
- E-commerce websites
- Healthcare systems
- IoT devices

Traditional tools cannot efficiently process such massive datasets. Big Data Analytics with R helps organizations:

- Analyze huge datasets
- Discover hidden patterns
- Predict future trends
- Support business decision-making

Advantages of R

1. Open Source

R is free and widely supported by the developer community.

2. Strong Statistical Support

Provides advanced statistical and mathematical functions.

3. Rich Visualization

Creates graphs, charts, dashboards, and reports.

4. Machine Learning Support

Supports predictive analytics and AI algorithms.

5. Big Data Integration

Can integrate with Big Data frameworks such as:

- Apache Hadoop
- Apache Spark
- HBase

Architecture of Big Data Analytics with R

Step 1: Data Collection

Data is collected from various sources:

- Databases
- Social media
- Sensors
- Websites
- Mobile applications

Step 2: Data Storage

Large datasets are stored using distributed storage systems:

- Hadoop Distributed File System
- MongoDB
- HBase

Step 3: Data Processing

Data is processed using:

- Apache Hadoop
- Apache Spark
- MapReduce programming

R can connect with these systems for analytics.

Step 4: Data Analysis

R performs:

- Statistical analysis
- Data mining
- Machine learning
- Predictive analytics

Step 5: Data Visualization

Results are represented using:

- Graphs
- Charts
- Dashboards

Important R Packages for Big Data Analytics

Package	Purpose
dplyr	Data manipulation
ggplot2	Data visualization
data.table	Fast data processing
caret	Machine learning
randomForest	Classification and prediction
tidyr	Data cleaning
sparklyr	Integration with Spark
RHadoop	Hadoop integration

Big Data Analytics Techniques in R

1. Descriptive Analytics

Analyzes historical data to understand past trends.

Example:

- Sales reports
- Website traffic analysis

2. Predictive Analytics

Predicts future outcomes using machine learning models.

Example:

- Sales forecasting
- Customer churn prediction

3. Prescriptive Analytics

Suggests actions based on analysis results.

Example:

- Product recommendation systems

4. Text Analytics

Analyzes text data from:

- Social media
- Reviews
- Comments

5. Sentiment Analysis

Determines whether opinions are:

- Positive
- Negative
- Neutral

Integration of R with Big Data Technologies

R with Hadoop

R works with Apache Hadoop using:

- RHadoop
- Hadoop Streaming

Benefits:

- Distributed data processing
- Large-scale analytics

R with Spark

R integrates with Apache Spark using:

- sparklyr
- SparkR

Benefits:

- Faster in-memory processing
- Real-time analytics

Applications of Big Data Analytics with R

Healthcare

- Disease prediction
- Patient analytics
- Medical research

Banking and Finance

- Fraud detection
- Risk analysis
- Stock market prediction

Social Media

- Sentiment analysis
- Trend analysis
- Customer behavior analysis

E-Commerce

- Product recommendations
- Customer segmentation
- Sales forecasting

Education

- Student performance analysis
- Learning analytics

Statistical Analysis of Bigdata:

Sample Dataset

Example: Student Performance Dataset

ID	Name	Branch	Marks	Attendance	Placement
1	Rahul	CSE	85	90	Yes
2	Sneha	ECE	78	88	No
3	Arjun	IT	92	95	Yes

Save as: **students.csv**

Procedure

Step 1: Install Required Packages

```
install.packages("ggplot2")
install.packages("dplyr")
```

Step 2: Load Libraries

```
library(ggplot2)
library(dplyr)
```

Step 3: Import Dataset

```
data <- read.csv("students.csv")
```

Step 4: View Dataset Structure

```
str(data)
summary(data)
```

```
> str(data)
'data.frame':   500 obs. of  6 variables:
 $ ID        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Name      : chr  "Karina" "Jessica" "Mitchell" "Jacob" ...
 $ Branch    : chr  "IT" "MECH" "ECE" "ECE" ...
 $ Marks     : int  93 95 54 45 84 92 65 81 81 80 ...
 $ Attendance: int  73 68 70 61 74 92 96 81 90 99 ...
 $ Placement : chr  "Yes" "Yes" "No" "Yes" ...
> summary(data)
       ID            Name              Branch              Marks        Attendance      Placement
 Min.   :  1.0   Length:500         Length:500         Min.   : 40.00   Min.   : 60.00   Length:500
 1st Qu.:125.8   Class :character   Class :character   1st Qu.: 56.00   1st Qu.: 69.00   Class :character
 Median :250.5   Mode  :character   Mode  :character   Median : 70.00   Median : 79.00   Mode  :character
 Mean   :250.5                                         Mean   : 70.63   Mean   : 79.90
 3rd Qu.:375.2                                         3rd Qu.: 86.00   3rd Qu.: 90.25
 Max.   :500.0                                         Max.   :100.00   Max.   :100.00
```

Step 5: Statistical Analysis

```
> mean(data$Marks)
[1] 70.632
> median(data$Marks)
[1] 70
> sd(data$Marks)
[1] 17.82583
> var(data$Marks)
[1] 317.7601
> max(data$Marks)
[1] 100
> min(data$Marks)
[1] 40
```

Data visualization of social media data

Sample Dataset

Create a CSV file named social_media.csv

PostID	Platform	Likes	Comments	Shares	Followers
1	Instagram	120	35	20	5000
2	Facebook	95	18	12	4200
3	Twitter	80	15	25	3900
4	Instagram	150	40	30	5500
5	YouTube	300	85	60	10000

Save as: **social_media.csv**

Procedure

Step 1: Install Required Packages

```
install.packages("ggplot2")
install.packages("dplyr")
```

Step 2: Load Libraries

```
library(ggplot2)
library(dplyr)
```

Step 3: Import Dataset

```
data <- read.csv("social_media.csv")
```

Step 4: View Dataset Information

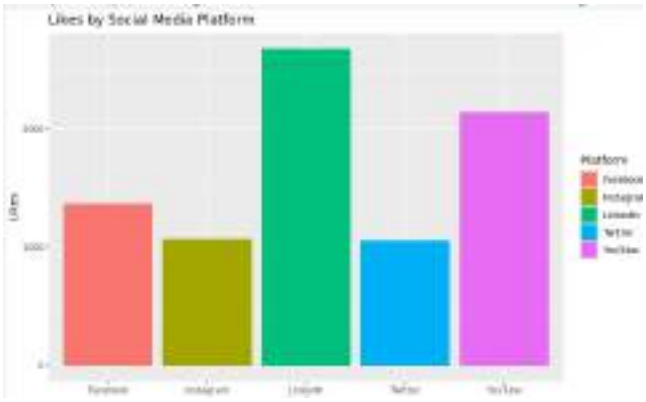
```
str(data)
summary(data)
```

```
> str(data)
'data.frame':   20 obs. of  7 variables:
 $ PostID     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Platform   : chr  "YouTube" "Facebook" "LinkedIn" "Facebook" ...
 $ ContentType: chr  "Post" "Story" "Story" "Story" ...
 $ Likes      : int  313 465 845 987 193 301 193 52 378 650 ...
 $ Comments   : int  200 30 185 76 104 118 276 270 91 225 ...
 $ Shares     : int  40 90 57 8 55 157 97 82 52 56 ...
 $ Followers  : int  17279 47332 27765 31067 45782 11868 1197 13183 59643 22824 ...
> summary(data)
     PostID        Platform         ContentType           Likes          Comments         Shares         Followers
 Min.   : 1.00   Length:20          Length:20          Min.   : 32.0   Min.   : 24.0   Min.   :  8.00   Min.   : 1197
 1st Qu.: 5.75   Class :character   Class :character   1st Qu.:193.0   1st Qu.:111.2   1st Qu.: 37.25   1st Qu.:13278
 Median :10.50   Mode  :character   Mode  :character   Median :349.0   Median :154.5   Median : 58.50   Median :28238
 Mean   :10.50                                         Mean   :415.7   Mean   :172.0   Mean   : 69.60   Mean   :25938
 3rd Qu.:15.25                                         3rd Qu.:527.2   3rd Qu.:227.5   3rd Qu.: 97.25   3rd Qu.:36654
 Max.   :20.00                                         Max.   :1042.0   Max.   :278.0   Max.   :149.00   Max.   :47565
```

Data Visualization Techniques

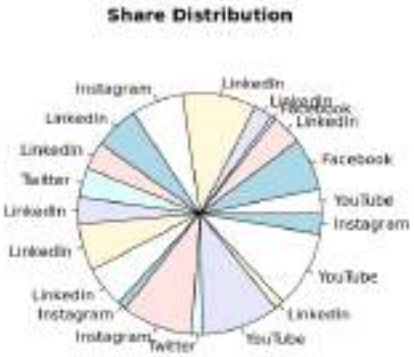
1. Bar Chart – Likes by Platform

```
ggplot(data, aes(x=Platform, y=Likes, fill=Platform)) +
  geom_bar(stat="identity") +
  ggtitle("Likes by Social Media Platform")
```



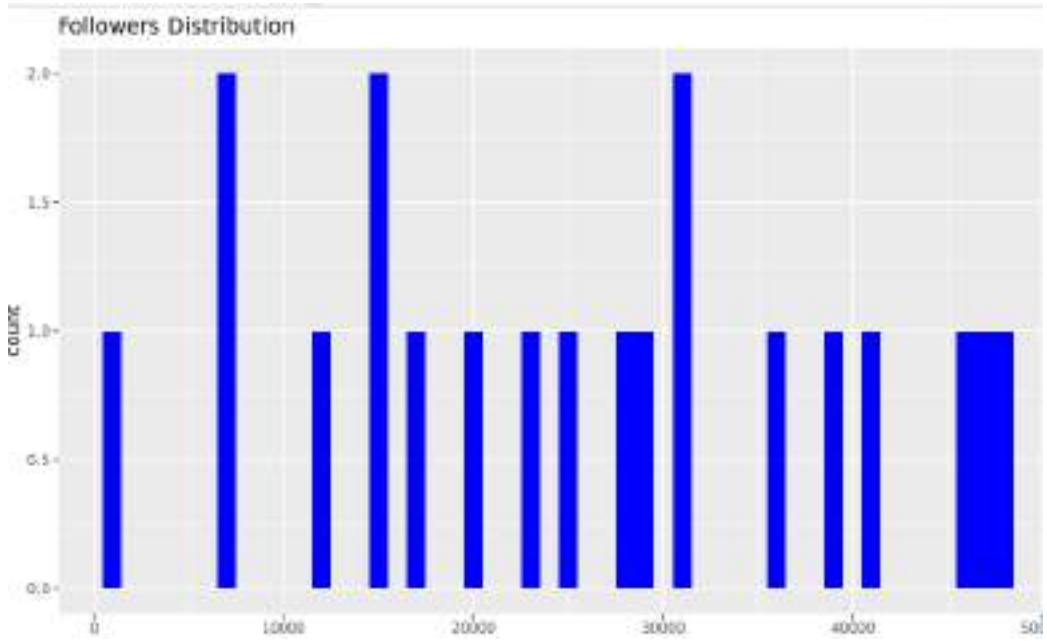
2. Pie Chart – Share Distribution

```
shares <- data$Shares
labels <- data$Platform

pie(shares,
    labels=labels,
    main="Share Distribution")
```



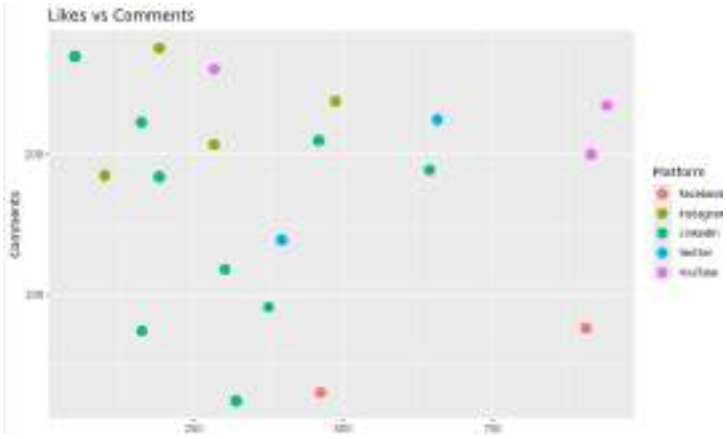
3. Histogram – Followers Count

```
ggplot(data, aes(x=Followers)) +
  geom_histogram(binwidth=1000, fill="blue") +
  ggtitle("Followers Distribution")
```



4. Scatter Plot – Likes vs Comments

```
ggplot(data,
       aes(x=Likes, y=Comments, color=Platform)) +
  geom_point(size=4) +
  ggtitle("Likes vs Comments")
```



5. Line Chart – Followers Trend

```
ggplot(data,
       aes(x=PostID, y=Followers, color=Platform)) +
  geom_line() +
  geom_point(size=3) +
  ggtitle("Followers Trend")
```