KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

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.

1. INTRODUCTION TO BIG DATA:

Big Data is a collection of data that is huge in volume, yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently. Big data is also a data but with huge size.

Example of Big Data:

- **New York Stock Exchange** is an example of Big Data that generates about one terabyte of new trade data per day.
- **Social Media**

  The statistic shows that 500+terabytes of new data get ingested into the databases of social media site Facebook, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.
- A **single Jet engine** can generate 10+terabytes of data in 30 minutes of flight time. With many thousand flights per day, generation of data reaches up to many Petabytes.

Types of Big Data

Following are the types of Big Data:

a) Structured
b) Unstructured
c) Semi-structured

a) Structured

Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. Over the period of time, talent in computer science has achieved greater success in developing techniques for working with such kind of data (where the format is well known in advance) and also deriving value out of it.

Examples of Structured Data: An 'Employee' table in a database is an example of Structured Data

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

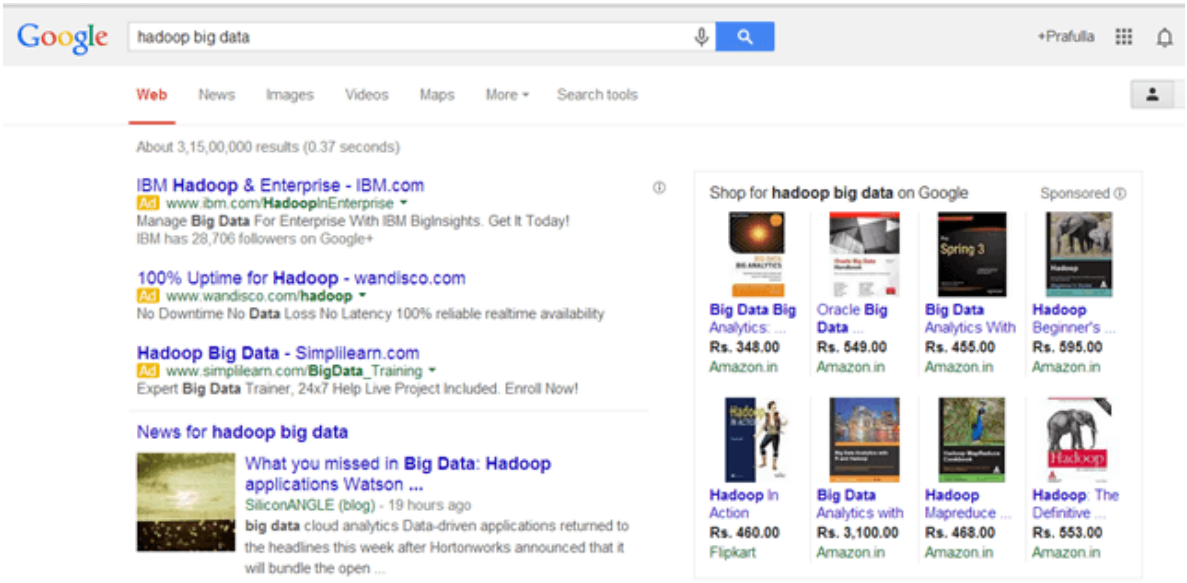
However, nowadays, we are foreseeing issues when a size of such data grows to a huge extent, typical sizes are being in the rage of multiple zettabytes. Looking at these figures one can easily understand why the name Big Data is given and imagine the challenges involved in its storage and processing.

b) *Unstructured*

Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it. A typical example of unstructured data is a heterogeneous data source containing a combination of simple text files, images, videos etc.

Now a days, organizations have wealth of data available with them but unfortunately, they don't know how to derive value out of it since this data is in its raw form or unstructured format.

Examples of Un-structured Data: The output returned by 'Google Search'



c) *Semi-structured*

Semi-structured data can contain both the forms of data. We can see semi-structured data as a structured in form but it is actually not defined with e.g. a table definition in relational DBMS. Example of semi-structured data is a data represented in an XML file.

Examples of Semi-structured Data: Personal data stored in an XML file-

```
<rec><name>Prashant
Rao</name><sex>Male</sex><age>35</age></rec>
<rec><name>Seema
R.</name><sex>Female</sex><age>41</age></rec>
<rec><name>Satish
Mane</name><sex>Male</sex><age>29</age></rec>
<rec><name>Subrato
Roy</name><sex>Male</sex><age>26</age></rec>
<rec><name>Jeremiah
J.</name><sex>Male</sex><age>35</age></rec>
```

2. BIG DATA AND ITS IMPORTANCE:

Big Data initiatives were rated as "extremely important" to 93% of companies. Leveraging a Big Data analytics solution helps organizations to unlock the strategic values and take full advantage of their assets.

It helps organizations:

- ✓ To understand Where, When and Why their customers buy
- ✓ Protect the company's client base with improved loyalty programs
- ✓ Seizing cross-selling and upselling opportunities
- ✓ Provide targeted promotional information
- ✓ Optimize Workforce planning and operations
- ✓ Improve inefficiencies in the company's supply chain
- ✓ Predict market trends
- ✓ Predict future needs
- ✓ Make companies more innovative and competitive
- ✓ It helps companies to discover new sources of revenue

Companies are using Big Data to know what their customers want, who are their best customers, why people choose different products. The more a company knows about its customers, the more competitive it becomes.

We can use it with Machine Learning for creating market strategies based on predictions about customers. Leveraging big data makes companies customer-centric.

Companies can use Historical and real-time data to assess evolving consumers' preferences. This consequently enables businesses to improve and update their marketing strategies which make companies more responsive to customer needs.

Importance of Big data

Big Data importance doesn't revolve around the amount of data a company has. Its importance lies in the fact that how the company utilizes the gathered data. Every company uses its collected data in its own way. More effectively the company uses its data, more rapidly it grows. The companies in the present market need to collect it and analyze it because:

1. Cost Savings: Big Data tools like Apache Hadoop, Spark, etc. bring cost-saving benefits to businesses when they have to store large amounts of data. These tools help organizations in identifying more effective ways of doing business.

2. Time-Saving: Real-time in-memory analytics helps companies to collect data from various sources. Tools like Hadoop help them to analyze data immediately thus helping in making quick decisions based on the learning.

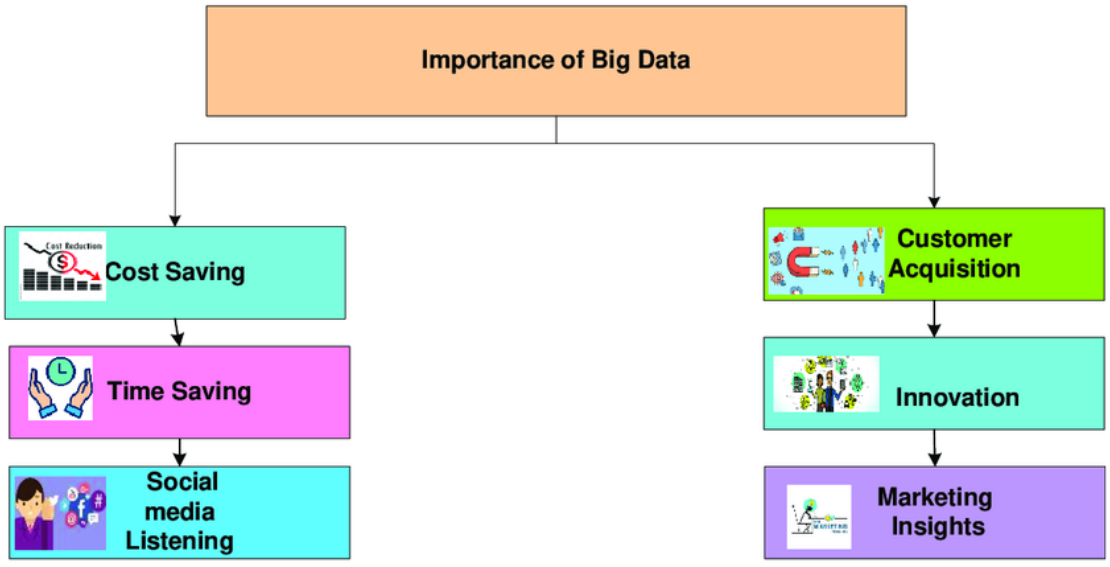
3. Understand the market conditions: Big Data analysis helps businesses to get a better understanding of market situations. For example, analysis of customer purchasing behavior helps companies to identify the products sold most and thus produces those products accordingly. This helps companies to get ahead of their competitors.

4. Social Media Listening: Companies can perform sentiment analysis using Big Data tools. These enable them to get feedback about their company, that is, who is saying what about the company. Companies can use Big data tools to improve their online presence.

5. Boost Customer Acquisition and Retention: Customers are a vital asset on which any business depends on. No single business can achieve its success without building a robust customer base. But even with a solid customer base, the companies can't ignore the competition in the market. If we don't know what our customers want then it will degrade companies' success. It will result in the loss of clientele which creates an adverse effect on business growth. Big data analytics helps businesses to identify customer related trends and patterns. Customer behavior analysis leads to a profitable business.

6. Solve Advertisers Problem and Offer Marketing Insights: Big data analytics shapes all business operations. It enables companies to fulfill customer expectations. Big data analytics helps in changing the company's product line. It ensures powerful marketing campaigns.

7. The driver of Innovations and Product Development: Big data makes companies capable to innovate and redevelop their products.



3. FOUR V'S OF BIG DATA (CHARACTERISTICS):

Big data can be described by the following characteristics:

i) Volume
ii) Variety
iii) Velocity
iv) Variability

(i) Volume – The name Big Data itself is related to a size which is enormous. Size of data plays a very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is

dependent upon the volume of data. Hence, **'Volume'** is one characteristic which needs to be considered while dealing with Big Data solutions.

(ii) Variety – The next aspect of Big Data is its **variety**. Variety refers to heterogeneous sources and the nature of data, both structured and unstructured. During earlier days, spreadsheets and databases were the only sources of data considered by most of the applications. Nowadays, data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. are also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analyzing data.

(iii) Velocity – The term **'velocity'** refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data. Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks, and social media sites, sensors, Mobile devices, etc. The flow of data is massive and continuous.

(iv) Variability – This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Advantages of Big Data Processing:

Ability to process Big Data in DBMS brings in multiple benefits, such as-

a) Businesses can utilize outside intelligence while taking decisions

   Access to social data from search engines and sites like facebook, twitter are enabling organizations to fine tune their business strategies.

b) Improved customer service

   Traditional customer feedback systems are getting replaced by new systems designed with Big Data technologies. In these new systems, Big Data and natural language processing technologies are being used to read and evaluate consumer responses.

c) Early identification of risk to the product/services, if any.

d) Better operational efficiency

   Big Data technologies can be used for creating a staging area or landing zone for new data before identifying what data should be moved to the data warehouse. In addition, such integration of Big Data technologies and data warehouse helps an organization to offload infrequently accessed data.

4. DRIVERS FOR BIG DATA:

Big Data has quickly risen to become one of the most desired topics in the industry. The main business drivers for such rising demand for Big Data Analytics are:

a. The digitization of society
b. The drop in technology costs
c. Connectivity through cloud computing

d. Increased knowledge about data science
e. Social media applications
f. The rise of Internet-of-Things(IoT)

Example: A number of companies that have Big Data at the core of their strategy like: Apple, Amazon, Facebook and Netflix have become very successful at the beginning of the 21st century.

5. INTRODUCTION TO BIG DATA ANALYTICS:

Big Data analytics is a process used to extract meaningful insights, such as hidden patterns, unknown correlations, market trends, and customer preferences. Big Data analytics provides various advantages—it can be used for better decision making, preventing fraudulent activities, among other things.

Importance:

In today's world, Big Data analytics is fueling everything we do online—in every industry.

Take the music streaming platform **Spotify**, for example. The company has nearly 96 million users that generate a tremendous amount of data every day. Through this information, the cloud-based platform automatically generates suggested songs—through a smart recommendation engine—based on likes, shares, search history, and more. What enables this is the techniques, tools, and frameworks that are a result of Big Data analytics.

If you are a Spotify user, then you must have come across the top recommendation section, which is based on your likes, past history, and other things. Utilizing a recommendation engine that leverages data filtering tools that collect data and then filter it using algorithms works. This is what Spotify does.

Uses and Examples of Big Data Analytics:

There are many different ways that Big Data analytics can be used in order to improve businesses and organizations. Here are some examples:

- ✓ Using analytics to understand customer behavior in order to optimize the customer experience
- ✓ Predicting future trends in order to make better business decisions
- ✓ Improving marketing campaigns by understanding what works and what doesn't
- ✓ Increasing operational efficiency by understanding where bottlenecks are and how to fix them
- ✓ Detecting fraud and other forms of misuse sooner

Advantages of Big Data Analytics:

1. Risk Management:

Use Case: Banco de Oro, a Phillippine banking company, uses Big Data analytics to identify fraudulent activities and discrepancies. The organization leverages it to narrow down a list of suspects or root causes of problems.

2. Product Development and Innovations

Use Case: Rolls-Royce, one of the largest manufacturers of jet engines for airlines and armed forces across the globe, uses Big Data analytics to analyze how efficient the engine designs are and if there is any need for improvements.

3. Quicker and Better Decision Making Within Organizations

Use Case: Starbucks uses Big Data analytics to make strategic decisions. For example, the company leverages it to decide if a particular location would be suitable for a new outlet or not. They will analyze several different factors, such as population, demographics, accessibility of the location, and more.

4. Improve Customer Experience

Use Case: Delta Air Lines uses Big Data analysis to improve customer experiences. They monitor tweets to find out their customers' experience regarding their journeys, delays, and so on. The airline identifies negative tweets and does what's necessary to remedy the situation. By publicly addressing these issues and offering solutions, it helps the airline build good customer relations.

6. BIG DATA ANALYTICS APPLICATIONS:

Here are some examples of the applications of big data analytics:

- ***Customer Acquisition and Retention:*** Customer information helps tremendously in marketing trends, through data-driven actions, to increase customer satisfaction. For example, personalization engines for Netflix, Amazon, and Spotify help with improved customer experiences and gaining customer loyalty.
- ***Targeted Ads:*** Personalized data about interaction patterns, order history, and product page viewing history can help immensely to create targeted ad campaigns for customers on a larger scale and at the individual level.
- ***Product Development:*** It can generate insights on development decisions, product viability, performance measurements, etc., and direct improvements that positively serve the customers.
- ***Price Optimization:*** Pricing models can be modeled and used by retailers with the help of diverse data sources to maximize revenues.
- ***Supply Chain and Channel Analytics:*** Predictive analytical models help with B2B supplier networks, preemptive replenishment, route optimizations, inventory management, and notification of potential delays in deliveries.
- ***Risk Management:*** It helps in the identification of new risks with the help of data patterns for the purpose of developing effective risk management strategies.
- ***Improved Decision-making:*** The insights that are extracted from the data can help enterprises make sound and quick decisions

Big Data Analytics Implementation in Major Sectors:

- ***Retail:*** The retail industry is actively deploying big data analytics. It is applying the techniques of data analytics to understand what the customers are buying and then offering products and services that are tailor-made for them. Today, it is all about having an omnichannel experience. Customers may make contact with a brand on one channel and then finally buy the product(s) through another channel, meanwhile going through more intermediary channels. The retailers will have to keep track of these customer journeys, and they must deploy

their marketing and advertising campaigns based on that, to improve the chances of increasing sales and lowering costs.

- ***Technology:*** Technology companies are heavily deploying big data analytics. They are finding out more about how customers interact with websites or apps and gather key information. Based on this, technology companies can optimize their sales, customer service, customer satisfaction, etc. This also helps them launch new products and services since we are living in a knowledge-intensive economy, and the companies in the technology sector are reaping the benefits of big data analytics.
- ***Healthcare:*** Healthcare is another industry that can benefit from big data analytics tools, techniques, and processes. Healthcare personnel can diagnose the health of their patients through various tests, run them through the computers, and look for telltale signs of anomalies, maladies, etc. It also helps in healthcare to improve patient care and increase the efficiency of the treatment and medication processes. Some diseases can be diagnosed before their onset so that measures can be taken in a preventive manner rather than a remedial manner.
- ***Manufacturing:*** Manufacturing is an industrial sector that is involved with developing physical goods. The life cycle of a manufacturing process can vary from product to product. Manufacturing systems are involved within the industry setup and across the manufacturing floor. There are a lot of technologies that are involved in manufacturing such as the Internet of Things (IoT), robotics, etc., but the backbone of all of these is firmly based on big data analytics. By using this, manufacturers can improve their yield, reduce the time to market, enhance the quality, optimize the supply chain and logistics processes, and build prototypes before the launch of products. It can help manufacturers through all these steps.
- ***Energy:*** Most oil and gas companies, which come under the energy sector, are extensive users of big data analytics. It is deployed when it comes to discovering oil and other natural resources. Tremendous amounts of big data go into finding out what the price of a barrel of oil will be, what the output should be, and if an oil well will be profitable or not. It is also deployed in finding out equipment failures, deploying predictive maintenance, and optimally using resources in order to reduce capital expenditure.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Big Data Analytics Tools:

a. ***Apache Spark:*** Spark is a framework for real-time data analytics, which is a part of the Hadoop ecosystem.
b. ***Python***: Python is one of the most versatile programming languages that is rapidly being deployed for various applications including machine learning.
c. ***SAS***: SAS is an advanced analytical tool that is used for working with large volumes of data and deriving valuable insights from it.
d. ***Hadoop***: Hadoop is the most popular big data framework that is deployed by a wide range of organizations from around the world for making sense of big data.
e. ***SQL***: SQL is used for working with relational database management systems.
f. ***Tableau***: Tableau is the most popular business intelligence tool that is deployed for the purpose of data visualization and business analytics.
g. ***Splunk***: Splunk is the tool of choice for parsing machine-generated data and deriving valuable business insights out of it.
h. ***R:*** R is the no. 1 programming language that is being used by data scientists for statistical computing and graphical applications alike.

Big Data Analytics Challenges: Big data analytics does not just come with wide-reaching benefits, it also comes with its own challenges:

a. ***Accessibility of Data:*** With larger volumes of data, storage and processing become a challenge. Big data should be maintained in such a way that it can be used by less-experienced data scientists and data analysts as well.
b. ***Data Quality Maintenance:*** With high volumes of data from disparate sources and in different formats, the proper management of data quality requires considerable time, effort, and resources.
c. ***Data Security:*** The complexity of big data systems poses unique challenges when it comes to security. It can be a complex undertaking to properly address such security concerns within complicated big data ecosystems.
d. ***Choosing the Right Tools:*** Choosing big data analytics tools from the wide range that is available in the market can be quite confusing. One should know how to select the best tool that aligns with user requirements and organizational infrastructure.
e. ***Supply-demand Gap in Skills:*** With a lack of data analytics skills in addition to the high cost of hiring experienced professionals, enterprises are finding it hard to meet the demand for skilled big data analytics professionals.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

UNIT – 2: Big Data Technologies: Hadoop's Parallel World – Data discovery – Open source technology for BigData Analytics – cloud and Big Data –Predictive Analytics – Mobile Business Intelligence and Big Data

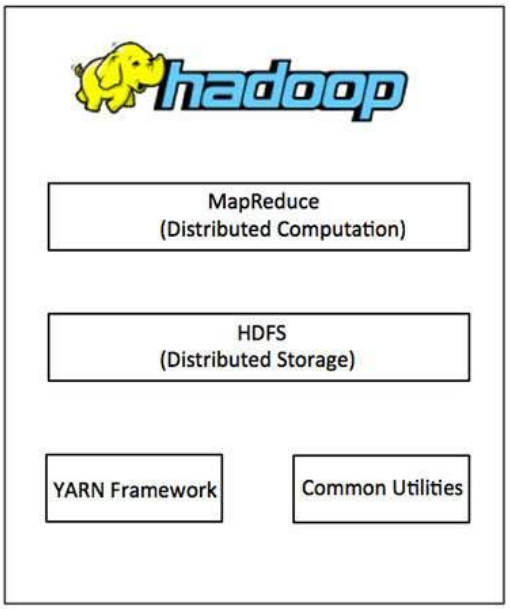
1. Big Data Technologies: Hadoop's Parallel World:

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.

Hadoop Architecture: At its core, Hadoop has two major layers namely −

a) Processing/Computation layer (MapReduce), and

b) Storage layer (Hadoop Distributed File System).

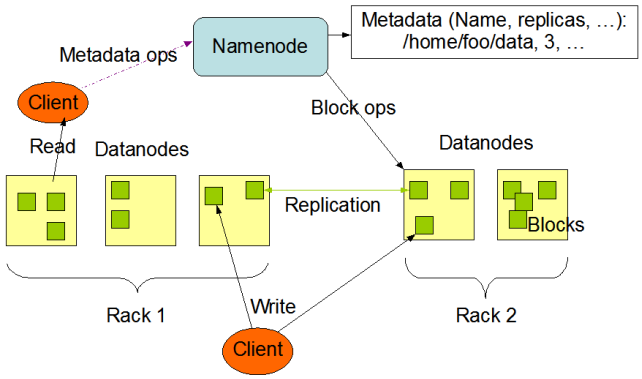


a) MapReduce: MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

b) ***Hadoop Distributed File System:*** The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

HDFS Architecture



Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules −

i) Hadoop Common − These are Java libraries and utilities required by other Hadoop modules.

ii) Hadoop YARN − This is a framework for job scheduling and cluster resource management.

How Does Hadoop Work?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs −

- ✓ Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
- ✓ These files are then distributed across various cluster nodes for further processing.
- ✓ HDFS, being on top of the local file system, supervises the processing.
- ✓ Blocks are replicated for handling hardware failure.
- ✓ Checking that the code was executed successfully.
- ✓ Performing the sort that takes place between the map and reduce stages.
- ✓ Sending the sorted data to a certain computer.
- ✓ Writing the debugging logs for each job.

Advantages of Hadoop:

- ➢ Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.

- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

2. Data discovery:

Data Discovery involves the collection and evaluation of data from various sources and is often used to understand trends and patterns in the data. It requires a progression of steps that organizations can use as a framework to understand their data. Data discovery, usually associated with ***business intelligence (BI),*** helps inform business decisions by bringing together disparate, siloed data sources to be analyzed. Having mounds of data is useless unless you find a way to extract insights from it. The data discovery process includes connecting multiple data sources, cleansing and preparing the data, sharing the data throughout the organization, and performing analysis to gain insights into business processes.

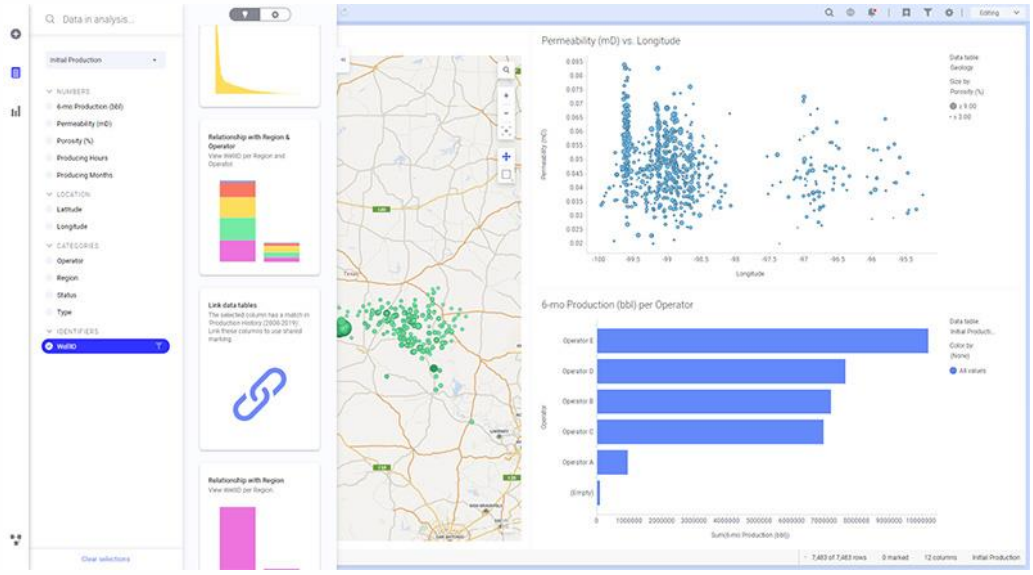
Today, nearly all businesses collect huge amounts of data on their customers, markets, suppliers, production processes, and more. Data flows in from online and traditional transactions systems, sensors, social media, mobile devices, and other diverse sources. As a result, decision makers are drowning in data but starving for insights. Insights are hidden within that data.

Data exploration and visual analytics is one approach that business data analysts use to uncover and investigate hidden but potentially useful insights in data. It is a methodology for digging into data looking for interesting relationships, trends, patterns, and anomalies requiring further exploration. Exploration and visual analytics enables the use of technology assisted analytical and pattern recognition software for visualization and drill-downs to turn data into knowledge and understanding.

Data discovery offers businesses a way to make their data clean, easily understandable, and user-friendly. A comprehensive solution should be able to be used by all members of the business. The main benefit of data discovery is the actionable insights that are uncovered in the data. These insights help users spot valuable opportunities before competitors without having to consult the IT organization. Visual data discovery can enhance this value, allowing line of business workers to find answers faster.

Today, companies are finding that the use of artificial intelligence (AI) is greatly enhancing the data discovery process. This process is also referred to as smart data discovery. In smart data discovery, AI can automatically discover data relationships and accelerate a company's analyses with AI-powered recommendations. The underlying AI suggestion engine uses sophisticated AI algorithms that run against any type of data without the user being aware that processing is happening in the background. The AI engine identifies potential relationships such as correlations

by employing trained learning algorithms. Leading analytics platforms utilizing AI offer recommended visualizations of related variables that users can choose to explore further.



There are several exciting directions for innovation in the area of AI-powered analytics including:

- AI techniques can be used to suggest data preparation steps such as normalization, missing data handling, string pattern recognitions, and others.
- Algorithms can be used to identify and draw attention to particular patterns or outliers in the data for groups of related variables.
- Time series analysis has distinct needs and techniques for pattern recognition, anomaly detection, and series relationships discovery.
- Behavioral data of expert users can be collected, analyzed, and used to influence recommended analysis actions.

AI suggestion engines and recommendations are increasingly used to augment analytics on an ever-expanding space of problems. This combination of human understanding with machine tirelessness enables business professionals to rapidly discover important relationships across vast amounts of data in time to take action.

Solving Business Problems with Data Discovery

Analysts are tasked with discovering insights in the massive amounts of data that businesses collect. Because it brings in data from so many different sources, data discovery enables businesses to use data in innovative ways. It helps users explore data in new and different ways and to find insights that were not apparent prior to data discovery. And, once new trends or patterns are made, data discovery makes it easy for users to drill down into the variables and come up with new questions and insights.

These insights can include identifying customer problems such as the following:

- Unexpected customer churn
- Customer relationship and management problems

- Subtle product issues such as returns and failures
- Price leakages due to excessive discounting
- Promotional failures
- Lost market share due to competitive actions such as aggressive pricing or a new product

Data discovery is enabling companies to capture a 360-degree view of their customers by compiling and assessing customer behavioral, transactional, and sentiment data across the many channels customers use to interact with companies.

Data discovery is invaluable in helping decision makers detect early warning signs about customer dissatisfaction.

Data discovery helps business leaders gain a more thorough understanding of how customers view the company.

Text, sentiment, social, and speech analytics can be used to identify what customers are saying about your company across a variety of interactions, including social media comments and contact center interactions. Key word searches against customer sentiment can help business leaders identify where potential product or service problems may be coming to the fore with multiple customers.

Data discovery tools also offer banks myriad opportunities to learn more about their customers and act on these insights. For instance, data discovery tools can help bankers determine which products a particular customer is using (e.g., checking, savings) and then determine based on that customer's income, lifecycle status, and other factors whether she might be a good candidate for a cross-sell or upsell offer (e.g., certificate of deposit).

With customer churn so high in financial services, bankers can also use data analysis and data discovery tools to determine the primary causes of customer defection among certain groups of customers and also to spot the warning signs when a customer is about to jump ship. Undetected and unaddressed, these problems can seriously undermine any business. Hence the urgency to find insights in the data and take action. With the right insights, companies can focus their efforts where they are needed to retain and delight customers rather than simply throwing customer-enticing tactics against the wall and see what sticks. Data discovery puts the power of big data into the hands of the everyday business user giving them the information that they need to make data-driven business decisions.

3. Open source technology for BigData Analytics:

Open-source big data analytics refers to the use of open-source software and tools for analyzing huge quantities of data in order to gather relevant and actionable information that an organization can use in order to further its business goals. The biggest player in open-source big data analytics is Apache's Hadoop – it is the most widely used software library for processing enormous data sets across a cluster of computers using a distributed process for parallelism.

Open-source big data analytics makes use of open-source software and tools in order to execute big data analytics by either using an entire software platform or various open-source tools for different tasks in the process of data analytics. Apache Hadoop is the most well-known system for big data analytics, but other components are required before a real analytics system can be put together.

Hadoop is the open-source implementation of the MapReduce algorithm pioneered by Google and Yahoo, so it is the basis of most analytics systems today. Many big data analytics tools make use of open source, including robust database systems such as the open-source MongoDB, a sophisticated and scalable NoSQL database very suited for big data applications, as well as others.

Open-source big data analytics services encompass:

- Data collection system
- Control center for administering and monitoring clusters
- Machine learning and data mining library
- Application coordination service
- Compute engine
- Execution framework

Top 10 Open-source Big Data Tools:

- **Hadoop**: analyzes large data sets, as the platform can send data to different servers.
- **Apache Spark:** It fills the gaps of Hadoop when it comes to data processing. It is also used for data analysis over other types of programs due to its ability to store large computations in memory.
- **Apache Cassandra:** It processes structured data sets.
- **MongoDB:** It is a document-oriented database is an ideal choice for businesses that need fast and real-time data for instant decisions.
- **HPCC (High-Performance Computing Cluster):** It is the competitor of Hadoop in the big data market. It is one of the open-source big data tools under the Apache 2.0 license. It delivers on a single platform, a single architecture, and a single programming language for data processing.
- **Apache Storm:** It is a free big data open-source computation system. It offers a distributed, real-time, fault-tolerant processing system.
- **Apache SAMOA (Scalable Advanced Massive Online Analysis):** It is used for mining big data streams with a special emphasis on machine learning enablement. It supports the ***Write Once Run Anywhere (WORA)*** architecture that allows seamless integration of multiple distributed stream processing engines into the framework.
- **Atlas.ti:** With this tool, we can access all available platforms from one place. It can be utilized for hybrid techniques and qualitative data analysis in academia, business, and user experience research.
- **Stats iQ:** It is used to automatically selects statistical tests. It is a large data tool that can quickly examine any data, and with Statwing, you can quickly make charts, discover relationships, and tidy up data.
- **CouchDB:** It uses JSON documents that can be browsed online or queried using JavaScript to store information. It enables fault-tolerant storage and distributed scaling.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

4. Cloud and Big Data:

a. **Big Data:** Big data refers to the data which is huge in size and also increasing rapidly with respect to time. Big data includes structured data, unstructured data as well as semi-structured data. Big data can not be stored and processed in traditional data management tools it needs specialized big data management tools. It refers to complex and large data sets having 5 V's volume, velocity, Veracity, Value and variety information assets. It includes data storage, data analysis, data mining and data visualization. Examples of the sources where big data is generated includes social media data, e-commerce data, weather station data, IoT Sensor data etc.

Characteristics of Big Data:

- Variety of Big data – Structured, unstructured, and semi structured data
- Velocity of Big data – Speed of data generation
- Volume of Big data – Huge volumes of data that is being generated
- Value of Big data – Extracting useful information and making it valuable
- Variability of Big data – Inconsistency which can be shown by the data at times.

Advantages of Big Data:

- Cost Savings
- Better decision-making
- Better Sales insights
- Increased Productivity
- Improved customer service.

Disadvantages of Big Data:

- Incompatible tools
- Security and Privacy Concerns
- Need for cultural change
- Rapid change in technology
- Specific hardware needs.

b. **Cloud Computing:** Cloud computing refers to the on demand availability of computing resources over internet. These resources includes servers, storage, databases, software, analytics, networking and intelligence over the Internet and all these resources can be used as per requirement of the customer. In cloud computing customers have to pay as per use. It is very flexible and can be resources can be scaled easily depending upon the requirement. Instead of buying any IT resources physically, all resources can be availed depending on the requirement from the cloud vendors. Cloud computing has three service models i.e Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Examples of cloud computing vendors who provides cloud computing services are Amazon Web Service (AWS), Microsoft Azure, Google Cloud Platform, IBM Cloud Services etc.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Characteristics of Cloud Computing:

- On-Demand availability
- Accessible through a network
- Elastic Scalability
- Pay as you go model
- Multi-tenancy and resource pooling.

Advantages of Cloud Computing:

- Back-up and restore data
- Improved collaboration
- Excellent accessibility
- Low maintenance cost
- On-Demand Self-service.

Disadvantages of Cloud Computing:

- Vendor lock-in
- Limited Control
- Security Concern
- Downtime due to various reason
- Requires good Internet connectivity.

Difference between Big Data and Cloud Computing:

S. No.	BIG DATA	CLOUD COMPUTING
1	Big data refers to the data which is huge in size and also increasing rapidly with respect to time.	Cloud computing refers to the on demand availability of computing resources over internet.
2	Big data includes structured data, unstructured data as well as semi-structured data.	Cloud Computing Services includes Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).
3	Volume of data, Velocity of data, Variety of data, Veracity of data, and Value of data are considered as the 5 most important characteristics of Big data.	On-Demand availability of IT resources, broad network access, resource pooling, elasticity and measured service are considered as the main characteristics of cloud computing.
4	The purpose of big data is to organizing the large volume of data and extracting the useful information from it and using that information for the improvement of business.	The purpose of cloud computing is to store and process data in cloud or availing remote IT services without physically installing any IT resources.
5	Distributed computing is used for analyzing the data and extracting the useful information.	Internet is used to get the cloud based services from different cloud vendors.
6	Big data management allows centralized platform, provision for backup and recovery and low maintenance cost.	Cloud computing services are cost effective, scalable and robust.
7	Some of the challenges of big data are variety of data, data storage and integration, data processing	Some of the challenges of cloud computing are availability, transformation, security concern, charging

	and resource management.	model.
8	Big data refers to huge volume of data, its management, and useful information extraction.	Cloud computing refers to remote IT resources and different internet service models.
9	Big data is used to describe huge volume of data and information.	Cloud computing is used to store data and information on remote servers and also processing the data using remote infrastructure.
10	Some of the sources where big data is generated includes social media data, e-commerce data, weather station data, IoT Sensor data etc.	Some of the cloud computing vendors who provides cloud computing services are Amazon Web Service (AWS), Microsoft Azure, Google Cloud Platform, IBM Cloud Services etc.

5. Predictive Analytics:

Big Data is frequently used to discuss Predictive Analytics. Predictive Analytics isn't a black-and-white notion or a stand-alone component of today's database management systems. It's, rather, a collection of data analysis tools and statistical methodologies. Thus, Big Data and business intelligence (BI) combine to bring about predictive analytics.

Predictive Analytics involves accumulating and analyzing historical data in order to predict future results. Connecting the dots between different departments, business processes, and forms of Big Data is made possible by combining multiple datasets. Examples of these future results include trends ("where a particular stock is likely to move?") and behavior traits ("what a particular customer is likely to purchase?"). To put it another way, Predictive Analytics is used to predict what will happen in a particular situation.

Many forward-thinking businesses, such as Google and Amazon, have recognized the value of Big Data Predictive Analytics in achieving a competitive advantage. These methods allow for the discovery of patterns and the improvement of optimization algorithms, among other things.

As Big Data technology evolves, businesses are turning to Predictive Analytics to help them enhance consumer engagement, streamline operations, and cut operational costs. The combination of real-time Big Data streams with Predictive Analytics— also known as "never-ending processing"— has the potential to give businesses a significant competitive advantage. Big Data Predictive Analytics is one way to use all of that data, obtain actionable new insights, and remain ahead of the competition

Examples of Predictive Analytics

- Customer Service
- Higher Education
- Insurance
- Supply Chain
- Software Testing

Predictive Analytics Techniques

- Decision Trees
- Neural Networks

- Text Analytics
- Regression Model

Predictive Analytics in Conjunction with Big Data: How They Work?

The heart of Predictive Analytics is that it is possible to 'model' most things. Underlying this idea is the notion that, between the data parameters, there is a cause and effect relationship, i.e., that as some data parameters change (cause), other data parameters will change in response (effect). In a nutshell, the following is a step-by-step procedure for using Predictive Analytics in businesses:

- ✓ Massive amounts of historical data are gathered or compiled.
- ✓ Certain statistical procedures, such as regression models, are used to analyze the data.
- ✓ The results of these assessments are then used to make forecasts about potential future events.
- ✓ These future predictions can then be used to help with decision-making, business process improvement, waste reduction, and more.

The final step would be to review the impact of Predictive Analytics on the process under consideration.

Big Data Predictive Analytics Processing: Predictive Analytics uses Big Data to find meaningful patterns to forecast future events, and evaluate the attractiveness of different solutions. Predictive Analytics can be used to analyze any form of unknown data from the past, present, or future. Using Big Data insights, Predictive Analytics gives businesses intelligence about the future

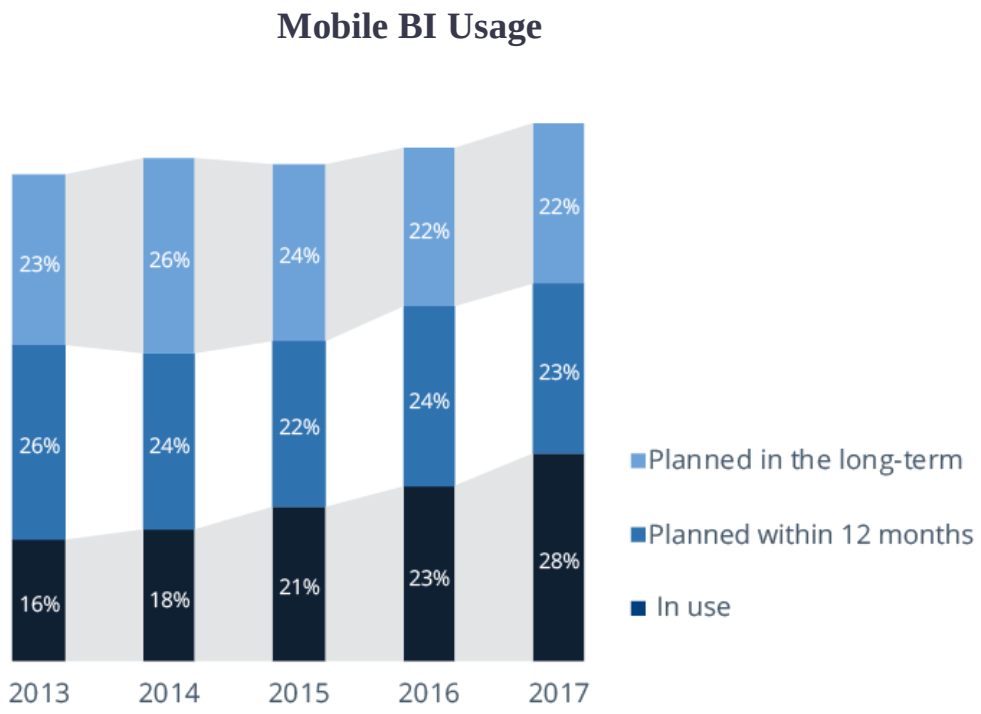
Comparisons between Big Data and Predictive Analytics:

The following are the top Big Data Predictive Analytics comparisons:

Big Data	Predictive Analytics
Big Data is concerned with the purification and interpretation of large amounts of data and can be applied to many business activities.	Predictive Analytics is a technique for predicting business and market events.
Big Data engines include built-in machine learning libraries, but integrating AI is still an R&D work for Data Engineers.	It deals with a platform based on mathematical calculations and probability.
The amount of data and the speed with which it is processed are enormous. It's not recommended to use Big Data platforms for small amounts of data because their performance is exponential.	The amount of data and the speed with which it is processed are both on the medium side. In terms of models and algorithms, very large and very small data sets can contribute to inaccurate predictions and discoveries.
Big Data comes with D3.js, Tableau, infogram, and other backend technology imports for Dashboards and Visualizations.	Predictive Analytics tools have built-in reporting integrations, such as Microsoft BI tools. So there's no need to get it from the source or a third-party seller.
The level of advancement for Big Data is high.	The level of advancement for Big Data is medium.
Big Data is a trendy topic right now. Everyone in the market wants to get into the Big Data business.	Predictive Analytics is popular, but it isn't the same as Big Data. It is dependent on the use cases and the type of organization that is putting it in place.
It's a tool for making data-driven decisions.	It is used in the assessment of risk and the forecasting of future results.
It's a best practice for handling large amounts of data.	It's a best practice for predicting the future with data.

6. Mobile Business Intelligence (B.I) and Big Data:

- The definition of mobile BI refers to the access and use of information via mobile devices. With the increasing use of mobile devices for business – not only in management positions – mobile BI is able to bring business intelligence and analytics closer to the user when done properly. Whether during a train journey, in the airport departure lounge or during a meeting break, information can be consumed almost anywhere and anytime with mobile BI.
- Mobile BI – driven by the success of mobile devices – was considered by many as a big wave in BI and analytics a few years ago. Nowadays, there is a level of disillusion in the market and users attach much less importance to this trend.



One of the major problems customers face when using mobile devices for information retrieval is the fact that mobile BI is no longer as simple as the pure display of BI content on a mobile device. Moreover, a mobile strategy has to be defined to cope with different suppliers and systems as well as private phones.

Besides attempts to standardize with the same supplier, companies are also concerned that solutions should have robust security features. These points have led many to the conclusion that a proper concept and strategy must be in place before supplying corporate information to mobile devices.

a) Benefits of mobile BI:

- ✓ The first major benefit is the ability for end users to access information in their mobile BI system **at any time** and **from any location**. This enables them to get data and analytics in 'real time', which **improves their daily operations** and means they can **react more quickly** to a wider range of events.

- ✓ The integration of mobile BI functions into operational business processes increases the penetration of BI within organizations and often brings benefits in the form of additional information.
- ✓ This **speeds up the decision-making** process by extending information and reducing the time spent searching for relevant information. With this **real-time access to data**, **operational efficiency** is improved and **organizational collaboration** is enforced.
- ✓ Overall, mobile BI brings about **greater availability of information**, faster reaction speed and **more efficient working**, as well as **improving internal communication** and **shortening workflows**.
- ✓ Finally, with the provision of proper mobile applications to all mobile device users, information can be used by people who previously did not use BI systems. This in turn leads to a **higher BI penetration** rate within companies.

b) Mobile BI technology:

- ✓ A variety of mobile devices can be used to display and actively work with information. Smartphones, tablets and wearables from brands such as Apple, Samsung, HTC and BlackBerry are the most common today.
- ✓ A significant difference between these types of device is obviously the size of the screen, which also affects mobile BI. For instance, tablets are comparable to small notebook computers, and are typically not subject to the extreme constraints of the small screen of a mobile phone.
- ✓ Thus, they offer more space to display content such as dashboards and reports, business data and KPIs compared to the smaller screen of mobile phones. Although BI applications can theoretically run on both tablets and mobile phones, they are not equally well suited to all types of BI. For example, interactive data visualizations require more screen space than displaying KPIs within a table.
- ✓ There are various ways to implement content on mobile devices. The most common we see in the marketplace are:
  - o Provision of PDF reports to a mobile device
  - o Website (HTML rendering), partly using proprietary technologies (Flash, Silverlight)
  - o HTML5 site
  - o Connection of a native application with HTML5 (hybrid application)
  - o Native application
- ✓ In principle, any BI vendor that can create a PDF or render in HTML (and almost all of them can) can say it supports mobile BI. Most mobile devices include a web browser that can access almost any web page to an acceptable degree of quality. The exception here is when proprietary technologies – which require additional software to display – are used.
- ✓ In the case mentioned, BI developers must check how their content renders on a mobile device when creating reports and other visualizations. This means designing their application specifically for mobile use. The main

advantage of this is its independence from device types (except when using proprietary technologies), since the content can be consumed on all devices.

- ✓ Another interesting trend among many software developers is the HTML5 client. BI content is displayed in the browser as previously described, but with several improvements. HTML5 enables Rich Internet Application (RIA) content to be projected across all types of mobile devices without relying on proprietary standards and without having to deal with their disadvantages.
- ✓ This technology is favored by software manufacturers, and not just because of its browser and operating system capabilities. The end user also benefits by being able to use it without having to install it. Unlike traditional HTML rendering, clients developed in HTML5 also provide some mobile-optimized navigation controls and functions such as zooming, pinching and double-tapping.
- ✓ In addition, HTML5 can be merged with the features of a native mobile application into a so-called "hybrid" form. This generally refers to a web application that can be downloaded as an app and installed on the device, but at its core includes a web viewer. For this reason, hybrids are often hard to distinguish from native apps. This hybrid category essentially supports more of the native features of the mobile device than a pure HTML5 client, but fewer than a native application.
- ✓ The "native" application type is the most expensive way for software manufacturers to support mobile BI because the software has to be tailored to the operating system (OS) of the mobile device. Native apps are typically downloaded and installed.
- ✓ The advantage of these products lies in their support of device-specific properties, such as the use of cache and navigation controls like "swipe" on the iPhone or iPad. Although the creation of native apps requires effort on the customer and vendor side, they enable interactive and enhanced use of analytics content.
- ✓ For instance, device functions such as voice recognition can be coupled with the software's natural language generation capabilities to query data ad hoc based on speech. Moreover, app developers are able to use sensors such as GPS to guide a customer to an article which is calculated to be potentially relevant to him. The more operational use and interaction with information that is required, the better the mobile OS support has to be.
- ✓ In general, the trend in mobile BI apps is veering towards knowledge generation rather than pure content consumption. Analysis and manipulation as well as input options for data are increasingly supported these days. Meanwhile, forecasts based on past data can be statistically calculated and directly reused on mobile devices.
- ✓ In our opinion, information should be updated as often as the reader needs it (sometimes even in real time). Especially in operational scenarios, decision-makers often have to react instantly to insights from data or changes in circumstances.

➢ ***Challenges of Mobile Business Intelligence:***

- ✓ Regardless of the delivery option, customers must consider usability. Displaying familiar content in a mobile browser (even as HTML5) does not necessarily mean that it can be used intuitively. Companies and app

developers should take great care in designing the user interface to ensure the app's acceptance, especially in operational scenarios where workers may not be accustomed to using BI

- ✓ When implementing a mobile BI solution, security and privacy may pose problems. Companies have to make sure they implement a strong security configuration in order to protect sensitive business and user data. Furthermore, the mobile strategy should be aligned with existing security procedures.
- ✓ Mobile devices can easily be hacked, lost or stolen. Using mobile BI may consequently put sensitive or confidential information at greater risk of being breached.
- ✓ Due to the limited screen size of mobile devices, the design of mobile BI applications presents new challenges to developers. Each device and browser works differently.

➢ ***Mobile BI tools:***

- ✓ The requirements and expectations of mobile BI are increasing so the selection of the right product is key to guaranteeing the maximum return on investment.
- ✓ Since providers are strongly oriented towards the most popular operating systems, they are also aligning their product portfolios for mobile BI.
- ✓ As well as vendors developing mobile BI solutions for their own BI software, there are more and more companies selling platform-independent systems. Most of them are specialist manufacturers who use the BI systems of popular providers such as IBM, SAP or Microsoft to extract information. Data is then integrated, processed and displayed using their own technologies. This type of mobile software is especially suitable for companies that use multiple BI systems.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

UNIT – 3: Introduction Hadoop: Big Data – Apache Hadoop & Hadoop Eco System – Moving Data in and out of Hadoop – Understanding inputs and outputs of MapReduce - Data Serialization

1. Introduction Hadoop:

Hadoop is an open-source software framework that is used for storing and processing large amounts of data in a distributed computing environment. It is designed to handle big data and is based on the MapReduce programming model, which allows for the parallel processing of large datasets.

Hadoop has two main components:

- HDFS (Hadoop Distributed File System): This is the storage component of Hadoop, which allows for the storage of large amounts of data across multiple machines. It is designed to work with commodity hardware, which makes it cost-effective.
- YARN (Yet Another Resource Negotiator): This is the resource management component of Hadoop, which manages the allocation of resources (such as CPU and memory) for processing the data stored in HDFS.
- Hadoop also includes several additional modules that provide additional functionality, such as Hive (a SQL-like query language), Pig (a high-level platform for creating MapReduce programs), and HBase (a non-relational, distributed database).
- Hadoop is commonly used in big data scenarios such as data warehousing, business intelligence, and machine learning. It's also used for data processing, data analysis, and data mining.

History of Hadoop:

Apache Software Foundation is the developers of Hadoop, and it's co-founders are **Doug Cutting** and **Mike Cafarella**. It's co-founder Doug Cutting named it on his son's toy elephant. In October 2003 the first paper release was Google File System. In January 2006, MapReduce development started on the Apache Nutch which consisted of around 6000 lines coding for it and around 5000 lines coding for HDFS. In April 2006 Hadoop 0.1.0 was released.

Hadoop is an open-source software framework for storing and processing big data. It was created by Apache Software Foundation in 2006, based on a white paper written by Google in 2003 that described the Google File System (GFS) and the MapReduce programming model. The Hadoop framework allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. It is used by many organizations, including Yahoo, Facebook, and IBM, for a variety of purposes such as data warehousing, log processing, and research. Hadoop has been widely adopted in the industry and has become a key technology for big data processing.

Features of hadoop:

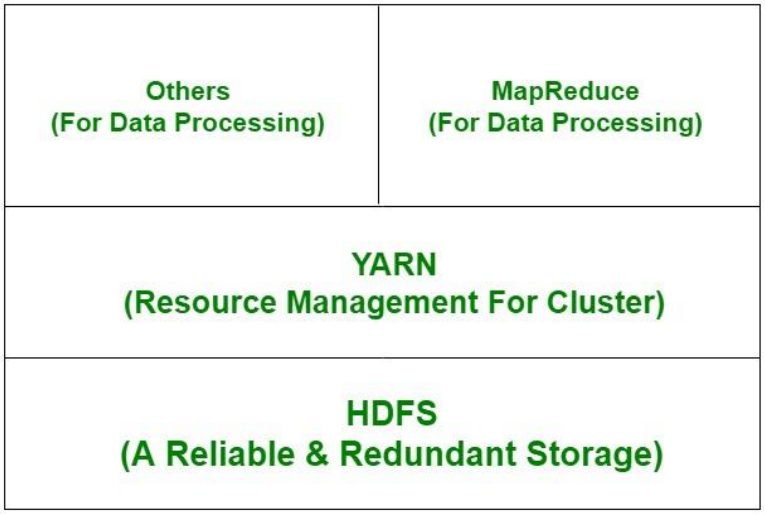
1. it is fault tolerance.
2. it is highly available.
3. it's programming is easy.

4. it have huge flexible storage.

5. it is low cost.

Hadoop has several key features that make it well-suited for big data processing:

- Distributed Storage: Hadoop stores large data sets across multiple machines, allowing for the storage and processing of extremely large amounts of data.
- Scalability: Hadoop can scale from a single server to thousands of machines, making it easy to add more capacity as needed.
- Fault-Tolerance: Hadoop is designed to be highly fault-tolerant, meaning it can continue to operate even in the presence of hardware failures.
- Data locality: Hadoop provides data locality feature, where the data is stored on the same node where it will be processed, this feature helps to reduce the network traffic and improve the performance
- High Availability: Hadoop provides High Availability feature, which helps to make sure that the data is always available and is not lost.
- Flexible Data Processing: Hadoop's MapReduce programming model allows for the processing of data in a distributed fashion, making it easy to implement a wide variety of data processing tasks.
- Data Integrity: Hadoop provides built-in checksum feature, which helps to ensure that the data stored is consistent and correct.
- Data Replication: Hadoop provides data replication feature, which helps to replicate the data across the cluster for fault tolerance.
- Data Compression: Hadoop provides built-in data compression feature, which helps to reduce the storage space and improve the performance.
- YARN: A resource management platform that allows multiple data processing engines like real-time streaming, batch processing, and interactive SQL, to run and process data stored in HDFS.



HDFS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Hadoop Distributed File System:

It has distributed file system known as HDFS and this HDFS splits files into blocks and sends them across various nodes in form of large clusters. Also in case of a node failure, the system operates and data transfer takes place between the nodes which are facilitated by HDFS.

Advantages of HDFS: It is inexpensive, immutable in nature, stores data reliably, ability to tolerate faults, scalable, block structured, can process a large amount of data simultaneously and many more. **Disadvantages of HDFS:** It's the biggest disadvantage is that it is not fit for small quantities of data. Also, it has issues related to potential stability, restrictive and rough in nature. Hadoop also supports a wide range of software packages such as Apache Flumes, Apache Oozie, Apache HBase, Apache Sqoop, Apache Spark, Apache Storm, Apache Pig, Apache Hive, Apache Phoenix, Cloudera Impala.

Some common frameworks of Hadoop:

1. Hive- It uses HiveQl for data structuring and for writing complicated MapReduce in HDFS.
2. Drill- It consists of user-defined functions and is used for data exploration.
3. Storm- It allows real-time processing and streaming of data.
4. Spark- It contains a Machine Learning Library(MLlib) for providing enhanced machine learning and is widely used for data processing. It also supports Java, Python, and Scala.
5. Pig- It has Pig Latin, a SQL-Like language and performs data transformation of unstructured data.
6. Tez- It reduces the complexities of Hive and Pig and helps in the running of their codes faster.

Hadoop framework is made up of the following modules:

1. Hadoop MapReduce- a MapReduce programming model for handling and processing large data.
2. Hadoop Distributed File System- distributed files in clusters among nodes.
3. Hadoop YARN- a platform which manages computing resources.
4. Hadoop Common- it contains packages and libraries which are used for other modules.

Advantages and Disadvantages of Hadoop:

Advantages:

- Ability to store a large amount of data.
- High flexibility.
- Cost effective.
- High computational power.
- Tasks are independent.
- Linear scaling.

Hadoop has several advantages that make it a popular choice for big data processing:

- Scalability: Hadoop can easily scale to handle large amounts of data by adding more nodes to the cluster.
- Cost-effective: Hadoop is designed to work with commodity hardware, which makes it a cost-effective option for storing and processing large amounts of data.

- Fault-tolerance: Hadoop's distributed architecture provides built-in fault-tolerance, which means that if one node in the cluster goes down, the data can still be processed by the other nodes.
- Flexibility: Hadoop can process structured, semi-structured, and unstructured data, which makes it a versatile option for a wide range of big data scenarios.
- Open-source: Hadoop is open-source software, which means that it is free to use and modify. This also allows developers to access the source code and make improvements or add new features.
- Large community: Hadoop has a large and active community of developers and users who contribute to the development of the software, provide support, and share best practices.
- Integration: Hadoop is designed to work with other big data technologies such as Spark, Storm, and Flink, which allows for integration with a wide range of data processing and analysis tools.

Disadvantages:

- Not very effective for small data.
- Hard cluster management.
- Has stability issues.
- Security concerns.
- Complexity: Hadoop can be complex to set up and maintain, especially for organizations without a dedicated team of experts.
- Latency: Hadoop is not well-suited for low-latency workloads and may not be the best choice for real-time data processing.
- Limited Support for Real-time Processing: Hadoop's batch-oriented nature makes it less suited for real-time streaming or interactive data processing use cases.
- Limited Support for Structured Data: Hadoop is designed to work with unstructured and semi-structured data, it is not well-suited for structured data processing
- Data Security: Hadoop does not provide built-in security features such as data encryption or user authentication, which can make it difficult to secure sensitive data.
- Limited Support for Ad-hoc Queries: Hadoop's MapReduce programming model is not well-suited for ad-hoc queries, making it difficult to perform exploratory data analysis.
- Limited Support for Graph and Machine Learning: Hadoop's core component HDFS and MapReduce are not well-suited for graph and machine learning workloads, specialized components like Apache Graph and Mahout are available but have some limitations.
- Cost: Hadoop can be expensive to set up and maintain, especially for organizations with large amounts of data.
- Data Loss: In the event of a hardware failure, the data stored in a single node may be lost permanently.
- Data Governance: Data Governance is a critical aspect of data management, Hadoop does not provide a built-in feature to manage data lineage, data quality, data cataloging, data lineage, and data audit.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

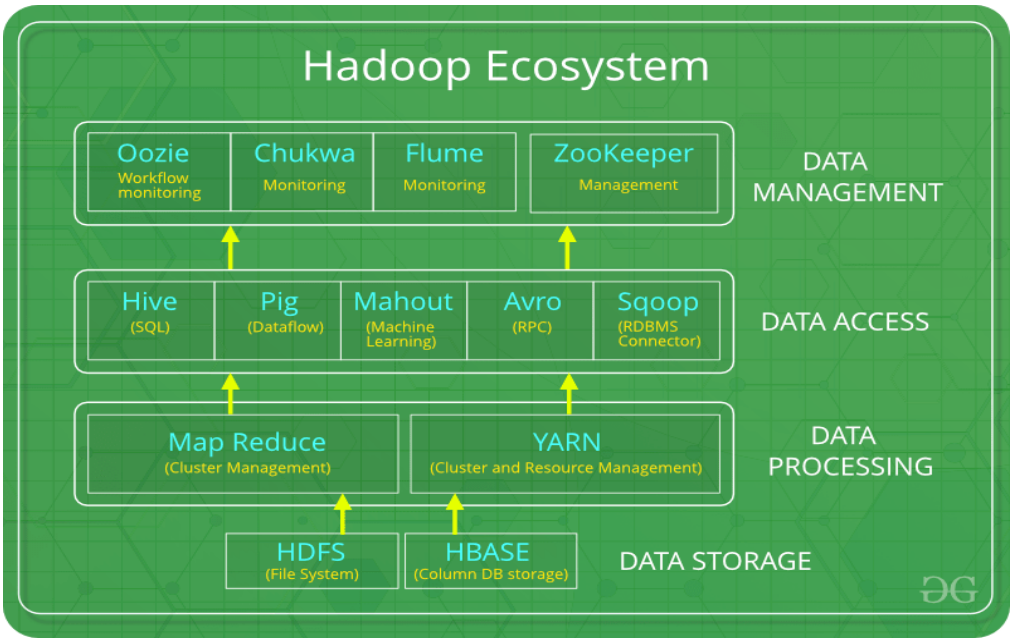
2. Big Data - Apache Hadoop & Hadoop Eco System:

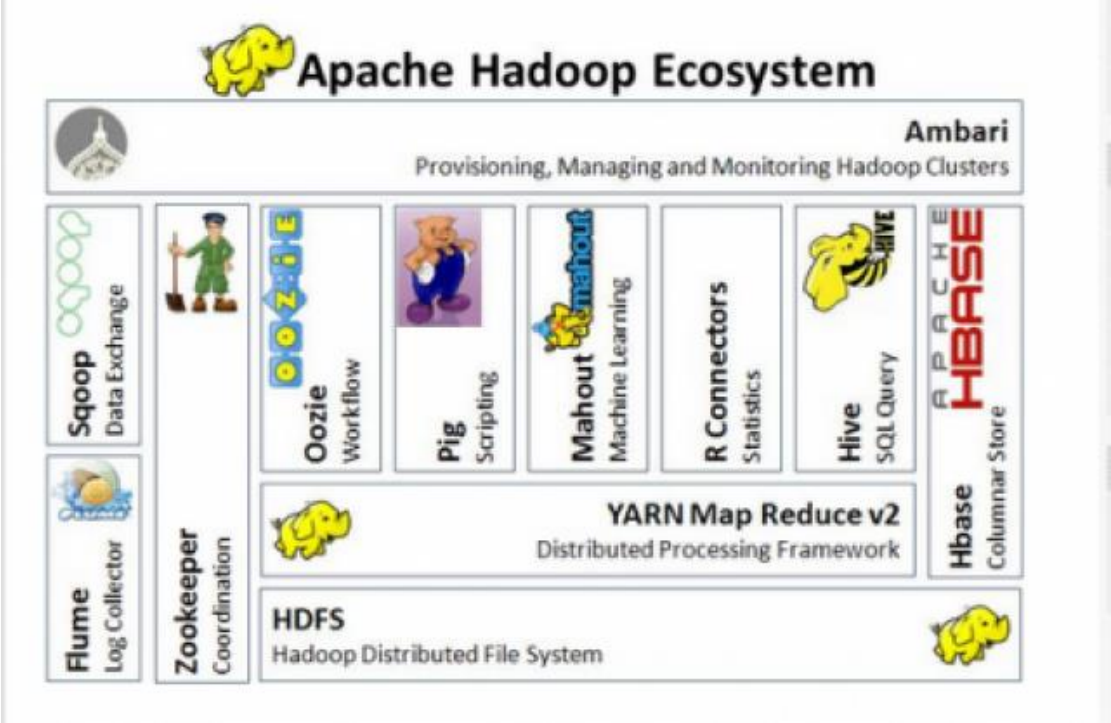
Apache Hadoop is an open source software framework for storage and large scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. It is licensed under the Apache License 2.0. Hadoop was created by Doug Cutting and Mike Cafarella in 2005. It was originally developed to support distribution for the Nutch search engine project. Doug, who was working at Yahoo! at the time and is now Chief Architect of Cloudera, named the project after his son's toy elephant. Cutting's son was 2 years old at the time and just beginning to talk. He called his beloved stuffed yellow elephant "Hadoop" (with the stress on the first syllable). Now 12, Doug's son often exclaims, "Why don't you say my name, and why don't I get royalties? I deserve to be famous for this!"

The Apache Hadoop framework is composed of the following modules

1. Hadoop Common: contains libraries and utilities needed by other Hadoop modules
2. Hadoop Distributed File System (HDFS): a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster
3. Hadoop YARN: a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications
4. Hadoop MapReduce: a programming model for large scale data processing

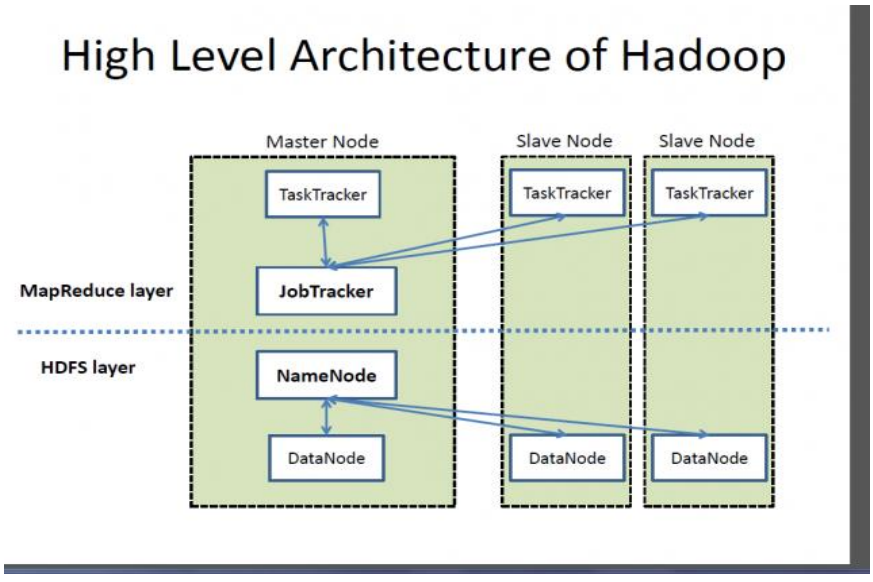
All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google's MapReduce and Google File System (GFS) papers. Beyond HDFS, YARN and MapReduce, the entire Apache Hadoop "platform" is now commonly considered to consist of a number of related projects as well: Apache Pig, Apache Hive, Apache HBase, and others.



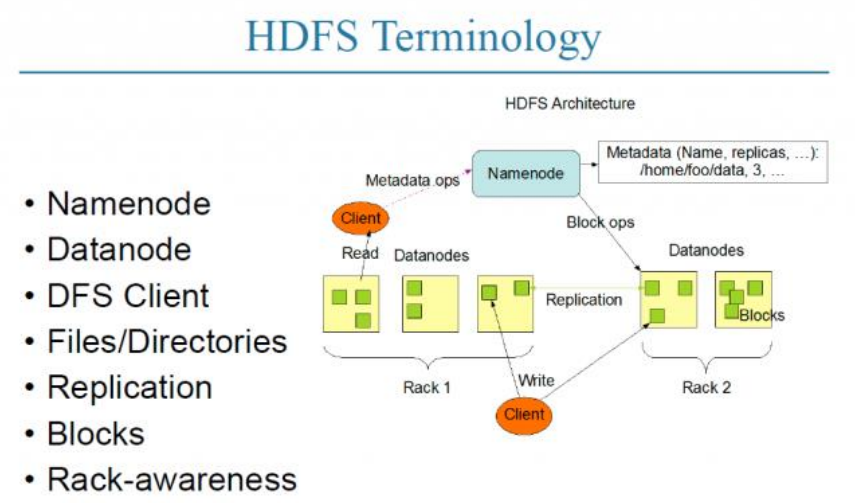


For the end-users, though MapReduce Java code is common, any programming language can be used with "Hadoop Streaming" to implement the "map" and "reduce" parts of the user's program. Apache Pig and Apache Hive, among other related projects, expose higher level user interfaces like Pig latin and a SQL variant respectively. The Hadoop framework itself is mostly written in the Java programming language, with some native code in C and command line utilities written as shell-scripts.

HDFS and MapReduce: There are two primary components at the core of Apache Hadoop 1.x: the Hadoop Distributed File System (HDFS) and the MapReduce parallel processing framework. These are both open source projects, inspired by technologies created inside Google.



Hadoop distributed file system: The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single namenode, and a cluster of datanodes form the HDFS cluster. The situation is typical because each node does not require a datanode to be present. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses the TCP/IP layer for communication. Clients use Remote procedure call (RPC) to communicate between each other.



HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence does not require RAID storage on hosts. With the default replication value, 3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high.

HDFS is not fully POSIX-compliant, because the requirements for a POSIX file-system differ from the target goals for a Hadoop application. The tradeoff of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append.

HDFS added the high-availability capabilities for release 2.x, allowing the main metadata server (the NameNode) to be failed over manually to a backup in the event of failure, automatic fail-over. The HDFS file system includes a so-called secondary namenode, which misleads some people into thinking that when the primary namenode goes offline, the secondary namenode takes over. In fact, the secondary namenode regularly connects with the primary namenode and builds snapshots of the primary namenode's directory information, which the system then saves to local or remote directories.

These checkpointed images can be used to restart a failed primary namenode without having to replay the entire journal of file-system actions, then to edit the log to create an up-to-date directory structure. Because the namenode is the single point for storage and management of metadata, it can become a bottleneck for supporting a huge number of files,

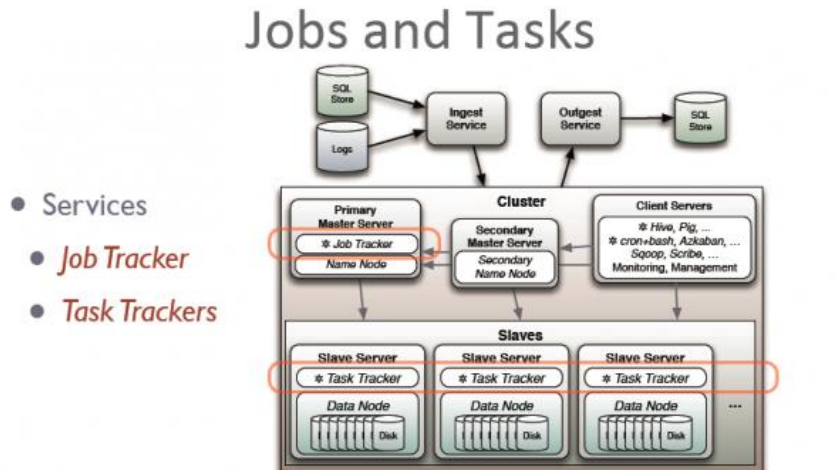
especially a large number of small files. HDFS Federation, a new addition, aims to tackle this problem to a certain extent by allowing multiple name-spaces served by separate namenodes.

An advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduce jobs to task trackers with an awareness of the data location. For example, if node A contains data (x, y, z) and node B contains data (a, b, c), the job tracker schedules node B to perform map or reduce tasks on (a,b,c) and node A would be scheduled to perform map or reduce tasks on (x,y,z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems, this advantage is not always available. This can have a significant impact on job-completion times, which has been demonstrated when running data-intensive jobs. HDFS was designed for mostly immutable files and may not be suitable for systems requiring concurrent write-operations.

Another limitation of HDFS is that it cannot be mounted directly by an existing operating system. Getting data into and out of the HDFS file system, an action that often needs to be performed before and after executing a job, can be inconvenient. A filesystem in Userspace (FUSE) virtual file system has been developed to address this problem, at least for Linux and some other Unix systems.

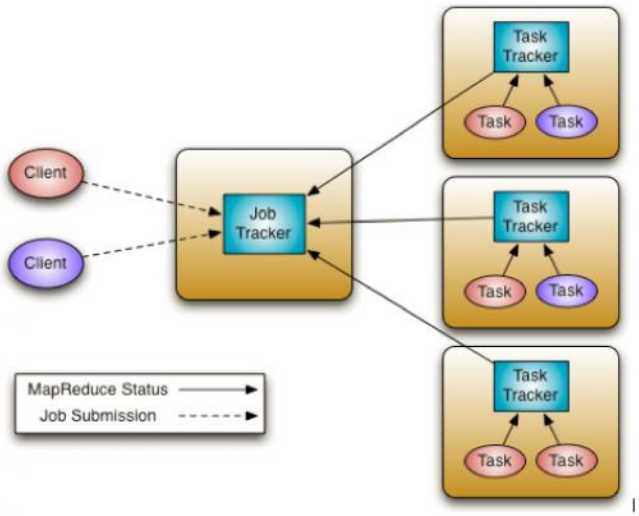
File access can be achieved through the native Java API, the Thrift API, to generate a client in the language of the users' choosing (C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, or OCaml), the command-line interface, or browsed through the HDFS-UI web app over HTTP.

JobTracker and TaskTracker: The MapReduce engine:



Above the file systems comes the MapReduce engine, which consists of one JobTracker, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible. With a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes in the same rack. This reduces network traffic on the main backbone network.

If a TaskTracker fails or times out, that part of the job is rescheduled. The TaskTracker on each node spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job crashes the JVM. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. The Job Tracker and TaskTracker status and information is exposed by Jetty and can be viewed from a web browser.



If the JobTracker failed on Hadoop 0.20 or earlier, all ongoing work was lost. Hadoop version 0.21 added some checkpointing to this process. The JobTracker records what it is up to in the file system. When a JobTracker starts up, it looks for any such data, so that it can restart work from where it left off.

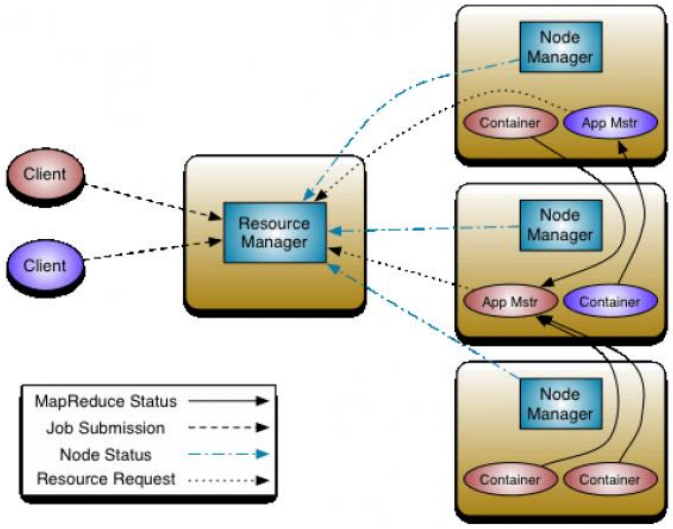
Known limitations of this approach in Hadoop 1.x: The allocation of work to TaskTrackers is very simple. Every TaskTracker has a number of available slots (such as "4 slots"). Every active map or reduce task takes up one slot. The Job Tracker allocates work to the tracker nearest to the data with an available slot. There is no consideration of the current system load of the allocated machine, and hence its actual availability.

If one TaskTracker is very slow, it can delay the entire MapReduce job—especially towards the end of a job, where everything can end up waiting for the slowest task. With speculative execution enabled, however, a single task can be executed on multiple slave nodes.

Apache Hadoop NextGen MapReduce (YARN): MapReduce has undergone a complete overhaul in hadoop-0.23 and we now have, what we call, MapReduce 2.0 (MRv2) or YARN. Apache™ Hadoop® YARN is a sub-project of Hadoop at the Apache Software Foundation introduced in Hadoop 2.0 that separates the resource management and processing components.

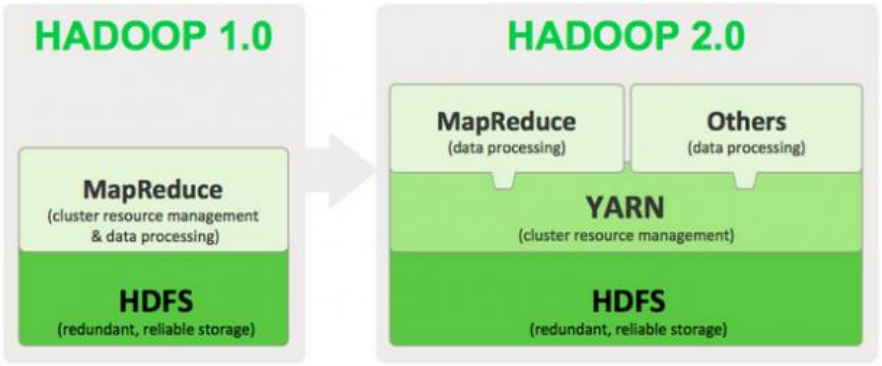
YARN was born of a need to enable a broader array of interaction patterns for data stored in HDFS beyond MapReduce. The YARN-based architecture of Hadoop 2.0 provides a more general processing platform that is not constrained to MapReduce.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE



The fundamental idea of MRv2 is to split up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate daemons. The idea is to have a global ResourceManager (RM) and per-application ApplicationMaster (AM). An application is either a single job in the classical sense of Map-Reduce jobs or a DAG of jobs.

The ResourceManager and per-node slave, the NodeManager (NM), form the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.



As part of Hadoop 2.0, YARN takes the resource management capabilities that were in MapReduce and packages them so they can be used by new engines. This also streamlines MapReduce to do what it does best, process data. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource management. Many organizations are already building applications on YARN in order to bring them IN to Hadoop.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

As part of Hadoop 2.0, YARN takes the resource management capabilities that were in MapReduce and packages them so they can be used by new engines. This also streamlines MapReduce to do what it does best, process data. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource management. Many organizations are already building applications on YARN in order to bring them IN to Hadoop. When enterprise data is made available in HDFS, it is important to have multiple ways to process that data. With Hadoop 2.0 and YARN organizations can use Hadoop for streaming, interactive and a world of other Hadoop based applications.

What YARN does: YARN enhances the power of a Hadoop compute cluster in the following ways:

1. Scalability: The processing power in data centers continues to grow quickly. Because YARN ResourceManager focuses exclusively on scheduling, it can manage those larger clusters much more easily.
2. Compatibility with MapReduce: Existing MapReduce applications and users can run on top of YARN without disruption to their existing processes.
3. Improved cluster utilization: The ResourceManager is a pure scheduler that optimizes cluster utilization according to criteria such as capacity guarantees, fairness, and SLAs. Also, unlike before, there are no named map and reduce slots, which helps to better utilize cluster resources.
4. Support for workloads other than MapReduce: Additional programming models such as graph processing and iterative modeling are now possible for data processing. These added models allow enterprises to realize near real-time processing and increased ROI on their Hadoop investments.
5. Agility: With MapReduce becoming a user-land library, it can evolve independently of the underlying resource manager layer and in a much more agile manner.

How YARN works

The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker/TaskTracker into separate entities:

1. a global ResourceManager
2. a per-application ApplicationMaster
3. a per-node slave NodeManager and
4. a per-application container running on a NodeManager

The ResourceManager and the NodeManager form the new, and generic, system for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is a framework-specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks. The ResourceManager has a scheduler, which is responsible for allocating resources to the various running applications, according to constraints such as queue capacities, user-limits etc. The scheduler performs its scheduling function based on the resource requirements of the applications. The NodeManager is the per-machine slave, which is responsible for launching the applications' containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager. Each ApplicationMaster has the responsibility of negotiating appropriate resource containers from the scheduler, tracking their status, and monitoring their progress. From the system perspective, the ApplicationMaster runs as a normal container.

3. Moving Data in and out of Hadoop:

Importing from MySQL to HDFS

The following is the canonical import job example sourced from

http://sqoop.apache.org/docs/1.99.2/Sqoop5MinutesDemo.html.

In Hue, this can be done in 3 easy steps:

- Environment
- CDH 4.4 or Hue 3.0.0
- MySQL 5.1

First, make sure that Sqoop2 is up and running and the Hue points to it in its hue.ini:

```
###########################################################################
\# Settings to configure Sqoop
###########################################################################
[sqoop]
\# Sqoop server URL
server_url=http://sqoop2.com:12000/sqoop
```

Troubleshooting

If the new job button is not appearing, Sqoop2 is probably not starting. Make sure the MySql or other DB connectors are in the /usr/lib/sqoop/lib directory of Sqoop2. Make sure you have these properties in the Sqoop2 Server configuration:

```
org.apache.sqoop.repository.schema.immutable=false
org.apache.sqoop.connector.autoupgrade=true
org.apache.sqoop.framework.autoupgrade=true
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

1. Create a Connection

In the Sqoop app, the connection manager is available from the "New Job" wizard. To get to the new job wizard, click on "New Job". There may be a list of connections available if a few have been created before. For the purposes of this demo, we'll go through the process of creating a new connection. Click "Add a new connection" and fill in the blanks with the data below. Then click save to return to the "New Job" wizard!

Connection Parameter	Value
Name	mysql-connection-demo
JDBC Driver Class	com.mysql.jdbc.Driver
JDBC Connection String	jdbc:mysql://hue-demo/demo
Username	demo
Password	demo

Connection form values.

2. Create a Job

After creating a connection, follow the wizard and fill in the blanks with the information below.

Job Wizard Parameter	Value
Name	mysql-import-job-demo
Type	IMPORT
Connection	mysql-connection-demo
Table name	test
Storage Type	HDFS
Output format	TEXT_FILE
Output directory	/tmp/mysql-import-job-demo

Job wizard form values.

3. Save and Submit the Job

At the end of the Job wizard, click "Save and Run"! The job should automagically start after that and the job dashboard will be displayed. As the job is running, a progress bar below the job listing will be dynamically updated. Links to the HDFS output via the File Browser and Map Reduce logs via Job Browser will be available on the left hand side of the job edit page.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

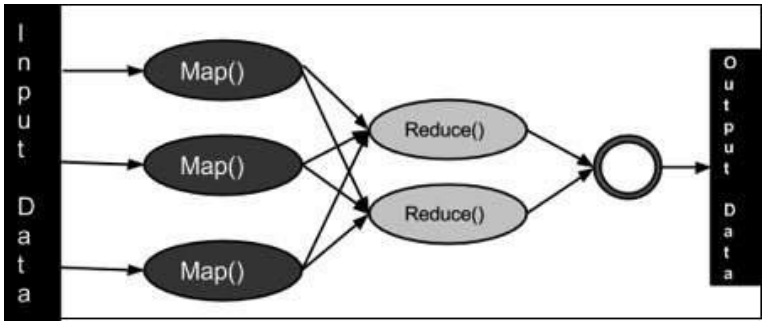
4. Understanding inputs and outputs of MapReduce:

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

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.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

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>)

Terminology

- **PayLoad** − Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** − Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** − Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** − Node where data is presented in advance before any processing takes place.
- **MasterNode** − Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** − Node where Map and Reduce program runs.
- **JobTracker** − Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** − Tracks the task and reports status to JobTracker.
- **Job** − A program is an execution of a Mapper and Reducer across a dataset.
- **Task** − An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** − A particular instance of an attempt to execute a task on a SlaveNode.

Example Scenario

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	0	40	39	39	45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written.

But, think of the data representing the electrical consumption of all the largescale industries of a particular state, since its formation.

When we write applications to process such bulk data,

- They will take a lot of time to execute.
- There will be a heavy network traffic when we move data from source to network server and so on.

To solve these problems, we have the MapReduce framework.

Input Data

The above data is saved as **sample.txt**and given as input. The input file looks as shown below.

1979 23 23 2 43 24 25 26 26 26 26 25 26 25

1980 26 27 28 28 28 30 31 31 31 30 30 30 29

1981 31 32 32 32 33 34 35 36 36 34 34 34 34

1984 39 38 39 39 39 41 42 43 40 39 38 38 40

1985 38 39 39 39 39 41 41 41 00 40 39 39 45

Example Program

Given below is the program to the sample data using MapReduce framework.

```
package hadoop;
import java.util.*;
import java.io.IOException;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class ProcessUnits
{
  //Mapper class
  public static class E_EMapper extends MapReduceBase implements
  Mapper<LongWritable ,/*Input key Type */
  Text,              /*Input value Type*/
  Text,              /*Output key Type*/
  IntWritable>       /*Output value Type*/
  {
    //Map function
    public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException
    {
       String line = value.toString();
       String lasttoken = null;
       StringTokenizer s = new StringTokenizer(line,"\t");
       String year = s.nextToken();
       while(s.hasMoreTokens())
       {
```

```
            lasttoken = s.nextToken();
          }
          int avgprice = Integer.parseInt(lasttoken);
          output.collect(new Text(year), new IntWritable(avgprice));
      }
}
//Reducer class
  public static class E_EReduce extends MapReduceBase implements Reducer< Text,
IntWritable, Text, IntWritable >
  {
    //Reduce function
     public void reduce( Text key, Iterator <IntWritable> values,
     OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException
     {
        int maxavg = 30;
        int val = Integer.MIN_VALUE;
        while (values.hasNext())
        {
          if((val = values.next().get())>maxavg)
          {
            output.collect(key, new IntWritable(val));
          }
        }
     }
  }
 //Main function
  public static void main(String args[])throws Exception
  {
     JobConf conf = new JobConf(ProcessUnits.class);
    conf.setJobName("max_eletricityunits");
     conf.setOutputKeyClass(Text.class);
     conf.setOutputValueClass(IntWritable.class);
     conf.setMapperClass(E_EMapper.class);
     conf.setCombinerClass(E_EReduce.class);
     conf.setReducerClass(E_EReduce.class);
     conf.setInputFormat(TextInputFormat.class);
     conf.setOutputFormat(TextOutputFormat.class);
     FileInputFormat.setInputPaths(conf, new Path(args[0]));
     FileOutputFormat.setOutputPath(conf, new Path(args[1]));
     JobClient.runJob(conf);
  }
```

Save the above program as **ProcessUnits.java.** The compilation and execution of the program is explained below.

Compilation and Execution of Process Units Program

Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop).

Follow the steps given below to compile and execute the above program.

Step 1

The following command is to create a directory to store the compiled java classes.

$ mkdir units

Step 2

Download **Hadoop-core-1.2.1.jar,** which is used to compile and execute the MapReduce program. Visit the following link mvnrepository.com to download the jar. Let us assume the downloaded folder is **/home/hadoop/.**

Step 3

The following commands are used for compiling the **ProcessUnits.java** program and creating a jar for the program.

$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java

$ jar -cvf units.jar -C units/ .

Step 4

The following command is used to create an input directory in HDFS.

$HADOOP_HOME/bin/hadoop fs -mkdir input_dir

Step 5

The following command is used to copy the input file named **sample.txt**in the input directory of HDFS.

$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir

Step 6

The following command is used to verify the files in the input directory.

$HADOOP_HOME/bin/hadoop fs -ls input_dir/

Step 7

The following command is used to run the Eleunit_max application by taking the input files from the input directory.

$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir

Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

INFO mapreduce.Job: Job job_1414748220717_0002

completed successfully

14/10/31 06:02:52

INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read = 61

FILE: Number of bytes written = 279400

FILE: Number of read operations = 0

FILE: Number of large read operations = 0

FILE: Number of write operations = 0

HDFS: Number of bytes read = 546

HDFS: Number of bytes written = 40

HDFS: Number of read operations = 9

HDFS: Number of large read operations = 0

HDFS: Number of write operations = 2 Job Counters

Launched map tasks = 2

Launched reduce tasks = 1

Data-local map tasks = 2

Total time spent by all maps in occupied slots (ms) = 146137

Total time spent by all reduces in occupied slots (ms) = 441

Total time spent by all map tasks (ms) = 14613

Total time spent by all reduce tasks (ms) = 44120

Total vcore-seconds taken by all map tasks = 146137

Total vcore-seconds taken by all reduce tasks = 44120

Total megabyte-seconds taken by all map tasks = 149644288

Total megabyte-seconds taken by all reduce tasks = 45178880

Map-Reduce Framework

Map input records = 5

Map output records = 5

Map output bytes = 45

Map output materialized bytes = 67

Input split bytes = 208

Combine input records = 5

Combine output records = 5

Reduce input groups = 5

Reduce shuffle bytes = 6

Reduce input records = 5

Reduce output records = 5

Spilled Records = 10

Shuffled Maps = 2

Failed Shuffles = 0

Merged Map outputs = 2

GC time elapsed (ms) = 948

CPU time spent (ms) = 5160

Physical memory (bytes) snapshot = 47749120

Virtual memory (bytes) snapshot = 2899349504

Total committed heap usage (bytes) = 277684224

File Output Format Counters

Bytes Written = 40

Step 8

The following command is used to verify the resultant files in the output folder.

$HADOOP_HOME/bin/hadoop fs -ls output_dir/

Step 9

The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000

Below is the output generated by the MapReduce program.

1981 34

1984 40

1985 45

Step 10

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs get output_dir /home/hadoop

Important Commands

All Hadoop commands are invoked by the **$HADOOP_HOME/bin/hadoop** command. Running the Hadoop script without any arguments prints the description for all commands.

Usage − hadoop [--config confdir] COMMAND

The following table lists the options available and their description.

S. No.	Option	Description
1	**namenode -format**	Formats the DFS filesystem.
2	**secondarynamenode**	Runs the DFS secondary namenode.
3	**namenode**	Runs the DFS namenode.
4	**datanode**	Runs a DFS datanode.
5	**dfsadmin**	Runs a DFS admin client.
6	**mradmin**	Runs a Map-Reduce admin client.
7	**fsck**	Runs a DFS filesystem checking utility.
8	**fs**	Runs a generic filesystem user client.
9	**balancer**	Runs a cluster balancing utility.
10	**oiv**	Applies the offline fsimage viewer to an fsimage.
11	**fetchdt**	Fetches a delegation token from the NameNode.
12	**jobtracker**	Runs the MapReduce job Tracker node.
13	**pipes**	Runs a Pipes job.
14	**tasktracker**	Runs a MapReduce task Tracker node.
15	**historyserver**	Runs job history servers as a standalone daemon.
16	**job**	Manipulates the MapReduce jobs.
17	**queue**	Gets information regarding JobQueues.
18	**version**	Prints the version.
19	**jar <jar>**	Runs a jar file.

20	**distcp <srcurl> <desturl>**	Copies file or directories recursively.
21	**distcp2 <srcurl> <desturl>**	DistCp version 2.
22	**archive -archiveName NAME -p <parent path> <src>* <dest>**	Creates a hadoop archive.
23	**classpath**	Prints the class path needed to get the Hadoop jar and the required libraries.
24	**daemonlog**	Get/Set the log level for each daemon

How to Interact with MapReduce Jobs

Usage − hadoop job [GENERIC_OPTIONS]

The following are the Generic Options available in a Hadoop job.

S. No.	GENERIC_OPTION	Description
1	**-submit <job-file>**	Submits the job.
2	**-status <job-id>**	Prints the map and reduce completion percentage and all job counters.
3	**-counter <job-id> <group-name> <countername>**	Prints the counter value.
4	**-kill <job-id>**	Kills the job.
5	**-events <job-id> <fromevent-#> <#-of-events>**	Prints the events' details received by jobtracker for the given range.
6	**-history [all] <jobOutputDir> - history < jobOutputDir>**	Prints job details, failed and killed tip details. More details about the job such as successful tasks and task attempts made for each task can be viewed by specifying the [all] option.
7	**-list[all]**	Displays all jobs. -list displays only jobs which are yet to complete.
8	**-kill-task <task-id>**	Kills the task. Killed tasks are NOT counted against failed attempts.
9	**-fail-task <task-id>**	Fails the task. Failed tasks are counted against failed attempts.
10	**-set-priority <job-id> <priority>**	Changes the priority of the job. Allowed priority values are VERY_HIGH, HIGH, NORMAL, LOW, VERY_LOW

To see the status of job

$ $HADOOP_HOME/bin/hadoop job -status <JOB-ID>

e.g.: $ $HADOOP_HOME/bin/hadoop job -status job_201310191043_0004

To see the history of job output-dir

$ $HADOOP_HOME/bin/hadoop job -history <DIR-NAME>

e.g. : $ $HADOOP_HOME/bin/hadoop job -history /user/expert/output

To kill the job

$ $HADOOP_HOME/bin/hadoop job -kill <JOB-ID>

e.g. : $ $HADOOP_HOME/bin/hadoop job -kill job_201310191043_0004

Input Formats: Hadoop can process many different types of data formats, from flat text files to databases. In this section, we explore the different formats available.

a). Controlling the maximum line length

If you are using one of the text input formats discussed here, you can set a maximum expected line length to safeguard against corrupted files. Corruption in a file can manifest itself as a very long line, which can cause out-ofmemory errors and then task failure. By setting mapreduce.input.linerecordreader.line.maxlength to a value in bytes that fits in memory (and is comfortably greater than the length of lines in your input data), you ensure that the record reader will skip the (long) corrupt lines without the task failing.

TextInputFormat – TextInputFormat is the default InputFormat. Each record is a line of input. The key, a LongWritable, is the byte offset within the file of the beginning of the line. The value is the contents of the line, excluding any line terminators.

KeyValueTextInputFormat – TextInputFormat's keys, being simply the offsets within the file, are not normally very useful. It is common for each line in a file to be a key-value pair, separated by a delimiter such as a tab character. For example, this is the kind of output produced by TextOut putFormat, Hadoop's default OutputFormat. To interpret such files correctly, KeyValueTextInputFormat is appropriate. You can specify the separator via the mapreduce.input.keyvaluelinerecordreader.key.value.separator property. It is a tab character by default.

NLineInputFormat – With TextInputFormat and KeyValueTextInputFormat, each mapper receives a variable number of lines of input. The number depends on the size of the split and the length of the lines. If you want your mappers to receive a fixed number of lines of input, then NLineInputFormat is the InputFormat to use. Like with TextInputFormat, the keys are the byte offsets within the file and the values are the lines themselves. N refers to the number of lines of input that each mapper receives. With N set to 1 (the default), each mapper receives exactly one line of input. The mapreduce.input.lineinputformat.linespermap property controls the value of N.

StreamInputFormat – Hadoop comes with a InputFormat for streaming which can be used outside streaming and can be used for processing XML documents. You can use it by setting your input format to StreamInputFormat and setting the stream.recordreader.class property to org.apache.hadoop.streaming.mapreduce.StreamXmlRecordReader. The reader is configured by setting job configuration properties to tell it the patterns for the start and end tags.

b). XmlInputFormat courtesy of the Lucene sub-project:

Mahout is been recommended in many blogging sites to be working for xml parsing and can be considered for xml reading. Link here XmlInputFormat.java

SequenceFileInputFormat – Hadoop MapReduce is not restricted to processing textual data. It has support for binary formats, too. Hadoop's sequence file format stores sequences of binary key-value pairs. Sequence files are well suited as a format for MapReduce data because they are splittable (they have sync points so that readers can synchronize with record boundaries from an arbitrary point in the file, such as the start of a split), they support compression as a part of the format, and they can store arbitrary types using a variety of serialization frameworks.

SequenceFileAsTextInputFormat – SequenceFileAsTextInputFormat is a variant of SequenceFileInputFormat that converts the sequence file's keys and values to Text objects. The conversion is performed by calling toString() on the keys and values. This format makes sequence files suitable input for Streaming

SequenceFileAsBinaryInputFormat – SequenceFileAsBinaryInputFormat is a variant of SequenceFileInputFormat that retrieves the sequence file's keys and values as opaque binary objects. They are encapsulated as BytesWritable objects, and the application is free to interpret the underlying byte array as it pleases.

FixedLengthInputFormat – FixedLengthInputFormat is for reading fixed-width binary records from a file, when the records are not separated by delimiters. The record size must be set via fixedlengthinputformat.record.length.

Multiple Inputs – Although the input to a MapReduce job may consist of multiple input files (constructed by a combination of file globs, filters, and plain paths), all of the input is interpreted by a single InputFormat and a single Mapper. What often happens, however, is that the data format evolves over time, so you have to write your mapper to cope with all of your legacy formats. Or you may have data sources that provide the same type of data but in different formats. This arises in the case of performing joins of different datasets . For instance, one might be tab-separated plain text, and the other a binary sequence file. Even if they are in the same format, they may have different representations, and therefore need to be parsed differently. These cases are handled elegantly by using the MultipleInputs class, which allows you to specify which InputFormat and Mapper to use on a per-path basis.

Database Input – DBInputFormat is an input format for reading data from a relational database, using JDBC. Because it doesn't have any sharding capabilities, you need to be careful not to overwhelm the database from which you are reading by running too many mappers. For this reason, it is best used for loading relatively small datasets, perhaps for joining with larger datasets from HDFS using MultipleInputs. The corresponding output format is DBOutputFormat, which is useful for dumping job outputs (of modest size) into a database.

Output Formats:Hadoop has output data formats that correspond to the input formats.

Text Output – The default output format, TextOutputFormat, writes records as lines of text. Its keys and values may be of any type, since TextOutputFormat turns them to strings by calling toString() on them. Each key-value pair is separated by a tab character, although that may be changed using the mapreduce.output.textoutputformat.separator property.

SequenceFileOutputFormat – its for writing binary Output. As the name indicates, SequenceFileOutputFormat writes sequence files for its output. This is a good choice of output if it forms the input to a further MapReduce job, since it is compact and is readily compressed.

SequenceFileAsBinaryOutputFormat – Is the counterpart to SequenceFileAsBinaryInputFormat writes keys and values in raw binary format into a sequence file container.

MapFileOutputFormat – MapFileOutputFormat writes map files as output. The keys in a MapFile must be added in order, so you need to ensure that your reducers emit keys in sorted order.

Multiple Outputs – Sometimes there is a need to have more control over the naming of the files or to produce multiple files per reducer. MapReduce comes with the MultipleOutputs class to help you do this.

Lazy Output – FileOutputFormat subclasses will create output (part-r-nnnnn) files, even if they are empty. Some applications prefer that empty files not be created, which is where Lazy OutputFormat helps. It is a wrapper output format that ensures that the output file is created only when the first record is emitted for a given partition. To use it, call its setOutputFormatClass() method with the JobConf and the underlying output format.

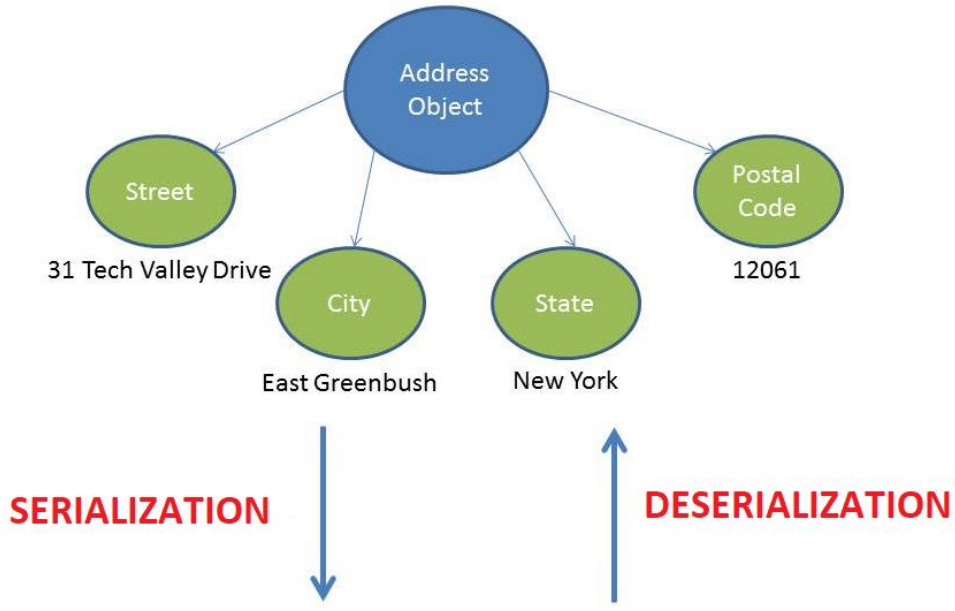
Database Output – The output formats for writing to relational databases and to HBase

5. Data Serialization:

Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices. Computer systems may vary in their hardware architecture, OS, addressing mechanisms. Internal binary representations of data also vary accordingly in every environment. Storing and exchanging data between such varying environments requires a platform-and-language-neutral data format that all systems understand.

Once the serialized data is transmitted from the source machine to the destination machine, the reverse process of creating objects from the byte sequence called ***deserialization*** is carried out. Reconstructed objects are clones of the original object. Choice of data serialization format for an application depends on factors such as data complexity, need for human readability, speed and storage space constraints. XML, JSON, BSON, YAML, MessagePack, and protobuf are some commonly used data serialization formats.

Working of Serialization and DeSerialization:



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Computer data is generally organized in data structures such as arrays, tables, trees, classes. When data structures need to be stored or transmitted to another location, such as across a network, they are serialized. For simple, linear data (number or string) there's nothing to do. Serialization becomes complex for nested data structures and object references. When objects are nested into multiple levels, such as in trees, it's collapsed into a series of bytes, and enough information (such as traversal order) is included to aid reconstruction of the original tree structure on the destination side.

When objects with pointer references to other member variables are serialized, the referenced objects are tracked and serialized, ensuring that the same object is not serialized more than once. However, all nested objects must be serializable too. Finally, the serialized data stream is persisted in a byte sequence using a standard format. ISO-8859-1 is a popular format for 1-byte representation of English characters and numerals. UTF-8 is the world standard for encoding multilingual, mathematical and scientific data; each character may take 1-4 bytes of data in Unicode.

Storage formats are a way to define how to store information in the file. Most of the time, assume this information is from the extension of the data. Both structured and unstructured data can store on HADOOP-enabled systems. Common Hdfs file formats are -

- Plain text storage
- Sequence files
- RC files
- AVRO
- Parquet

Why Storage Formats?

- File format must be handy to serve complex data structures
- HDFS enabled applications to find relevant data in a particular location and write back data to another location.
- Dataset is large
- Having schemas
- Having storage constraints

Why choose different File Formats? Proper selection of file format leads to -

- Faster read time
- Faster write time
- Splittable files (for partial data read)
- Schema evolution support (modifying dataset fields)
- Advance compression support
- Snappy compression leads to high speed and reasonable compression/decompression.
- File formats help to manage Diverse data.

How to enabling data serialization in it?

- Data serialization is a process that converts structure data manually back to the original form.

- Serialize to translate data structures into a stream of data. Transmit this stream of data over the network or store it in DB regardless of the system architecture.
- Isn't storing information in binary form or stream of bytes is the right approach.
- Serialization does the same but isn't dependent on architecture.
- Consider CSV files contains a comma (,) in between data, so while Deserialization, wrong outputs may occur. Now, if metadata is stored in XML form, a self- architected form of data storage, data can easily deserialize.

Why Data Serialization for Storage Formats?

- To process records faster (Time-bound).
- When proper data formats need to maintain and transmit over data without schema support on another end.
- Now when in the future, data without structure or format needs to process, complex Errors may occur.
- Serialization offers data validation over transmission.

Areas of Serialization for Storage Formats

To maintain the proper format of a data serialization system must have the following four properties -

- **Compact -** helps in the best use of network bandwidth
- **Fast -** reduces the performance overhead
- **Extensible -** can match new requirements
- **Inter-operable -** not language-specific

It has two areas -

1. ***Interprocess communication:*** When a client calls a function or subroutine from one pc to the pc in-network or server, that calling is a remote procedure call.
2. ***Persistent storage:*** It is better than java's inbuilt serialization as java serialization isn't compact. Serialization and Deserialization of data help maintain and manage corporate decisions for effective use of resources and data available in Data warehouse or any other database -writable - language specific to java.

Stacking of Storage Formats Hadoop

- Use splittable formats
- CSV files lack block compression; using CSV files in it may increase reading performance cost. Schema support is also within the limit.
- Use JSON records instead of JSON files. For better performance, experiments have to perform for each use case of JSON.
- AVRO file format and data serialization framework.
- Sequence files are complex in reading.
- Write operation is slower if RC (Row-Columnar) files are in use.
- Optimized RC (ORC) files are also the option to use but have less support.
- Parquet Files are using a columnar format to store data and process it.

Guide to Apache Hadoop File Format Configuration

- Its distribution information and architecture type (system information where Hadoop system is working and process jobs) must analyze thoroughly.
- Perform Schema evolution by defining the structure of your metadata.
- Processing requirements for file formats and configure the Hadoop system.
- Engine support for reading and write operations.
- Evaluate storage size at a production level its File formats selection.
- Text files are lightweight, but splitting those files and reading data leads to huge maps (MAPS needs to create a complicated way to achieve splitting of text (files data).
- Sequence files support block compression. A hive has SQL types, so not worthy of working with Hive.
- RCFILE has a high compression rate, but it takes more time to load data.
- ORC can reduce data size up to 75% and suitable with hive but increases CPU overhead. Serialization in ORC depends on data type (either integer or string).
- AVRO provides features like serialization and Deserialization with file format also. The core of AVRO is its schema. Supports dynamic and static types.
- Parquet provides the facility to partition data into both rows and columns but computationally slow on the right side.
- Avro is a data exchange and data serialization service system. It is based on schemas.
- Always store Schema in an AVRO file with data.
- Support Specific and Generic data.
- JSON defines AVRO schema (most languages have JSON libraries), and data format is binary, which determines the efficiency and makes it compact.
- Set the default schema for some mismatched or failed schema.

Apache Avro Solution Offerings: At the same time, a few questions arise that - Is it possible to handle any schema change

- Yes, in AVRO, schema changes like missing fields, changed/modified fields, and added/new areas are easy to maintain What will happen to past data when there is a change in schema.
- AVRO can read data with a new schema regardless of data generation time. Is there any problem that occurs while trying to convert AVRO data to other formats, if needed.
- There are some inbuilt specifications in its system while using a Hybrid system of data? In such a way, Apache spark can convert Avro files to parquet. Is it possible to manage MapReduce jobs.
- MapReduce jobs can efficiently achieve by using Avro file processing in MapReduce itself. What connection Handshake do.
- Connection handshake helps to exchange schemas between client and server during RPC.

How is AVRO different from other systems? (Thrift and protocol buffer)

- AVRO has Dynamic Schema stores for serialized values in binary format in a space-efficient way.
- JSON schema and data in the file.
- Parser No compilation directly by using the parser library.
- AVRO is a language-neutral system.

How to implement Avro?

- Have the schema of data format ready
- To read schema in program
- Compile using AVRO (by generating class)
- Direct read via Parser library
- Achieve serialization through serialization API (java)

By creating class - A schema fed to AVRO utility and then gets processes as a java file. Now write API methods to serialize data.

By parser library - Fed schema to AVRO utility and access serialization parser library to serialize data. Achieve Deserialization through deserialization API (java.

By generating class - Deserialize the object and instantiate DataFileReader class.

By parser library - Instantiate parser class.

Creating schema in Avro: There are three ways to define and create JSON schemas of AVRO -

- JSON String
- The JSON object
- JSON Array

There are some data type that needs to mention in schema -

- Primitive (common data types)
- Complex (when collective data elements need to be process and store)
- Primitive data types contain null, boolean, int, long, bytes, float, string, and double. In contrast, complex contains Records (attribute encapsulation), Enums, Arrays, Maps (associativity), Unions (multiple types for one field), and Fixed (deals with the size of data).
- Data types help to maintain sort order.
- Logical types (a super form of complex data types) are used to store individual data involved, i.e., time, date, timestamp, and decimal.

Apache Avro Use Cases

- Schema evolution.
- Apture is using AVRO for logging matrices data.
- RichRelevance is using binary formatted key/value pairs.

What is Apache Parquet?

Apache developed parquet, and it is a columnar storage format for the Hadoop ecosystem. It supports all data processing frameworks. Also, it supports all data models or programming languages -

- Provides a compressed and efficient representation of columnar data.
- Record stripping and assembly algorithm from Dremel paper.
- Compression schemes can be present at the per-column level.
- Metadata is for single-pass writing after data.
- The file format contains column-wise data, which is split into row groups.
- Associations with data are done in three types of metadata forms

Files -Includes associations of schema element, principal value, and row group, which connect further to column chunk.

1. **Column (chunks) -** Contains associations with KeyValue and connected with every ColumnChunk under RowGroup.
2. **Page Header -** Contains info of size and associations of PageDataHeader, indexPage.

Header and Dictionary PageHeader

Had a capability to reuse efficient encoding over primitive types utilizing Annotations that further can decode and interpret and Support many engines such as Hive, Impala, spark, etc.

Apache Parquet Capabilities

- Task-oriented serialization of data can be performed if some column-based decisions need to be managed.
- Data in 3 rows like r1(c1,c2,c3), r2(c1,c2,c3), r3(c1,c2,c3) will represent in columnar storage as 3 columns like c1(r1,r2,r3), c2(r1,r2,r3), c3(r1,r2,r3).
- Easy to split data partition in column way.
- As data is compressed and metadata is associated, modifications and new additions to schema may reflect changes immediately without interruption.
- The columnar format can store the same primitive types of values together, forming a nested data structure (using Definition and repetition levels).
- Columnar storage provides more homogeneous data so that the compression rate will be better for the Hadoop cluster.
- ProtoBuf model can help to describe the data structure of the columnar format using a schema.
- Encoding and decoding of column data make data more compact and compressed in nature.

Causes of error in Parquet.

- When any metadata is corrupt, then it will cause particular associative data to lose.
- While writing file metadata at the end of the file may also cause data loss if any error occurs.
- When the structure of the external table is changed, how can metadata information be updated?.

Apache Parquet Solution Offerings

- The approach with sync markers in RC and AVRO is usable here. Write file metadata at some Nth RowGroup, so during error, partially written data can be recovered.
- Easy to integrate with AVRO, thrift, etc. Models.
- When the PURGE option is not provided, managed tables can be retrieved from trash for a defined duration.

What is Apache Hive and working architecture?

- Data warehouse software that provides fast processing of massive amounts of data.
- It can use for both Batch and Streaming queries.
- Read, write and manage large datasets of distributed systems using SQL.
- Requires schema to process structured data, follows OLAP.
- Develop SQL typescripts for MapReduce operations, known as HQL.
- The execution engine uses the technique of MapReduce to process the query and to generate results similar to MapReduce.
- Require an external database server to configure Metastore (Apache Derby).
- Convert unstructured data into a proper structure to process using Hive.
- Apache Hive has a high level of abstraction compared to Hadoop MapReduce, and code efficiency is very less.

Apache Hive Working Architecture

- First of all, give a query to the driver to execute
- The driver then sends a request to the compiler and then requests for metadata from the Meta store
- That metadata is then processed back to the driver. Then execution code is given to the execution engine, which processed data from the Hadoop system containing job tracker and HDFS with name nodes.
- After MapReduce finishes its processing of tasks and jobs, the result is then sent back to drivers and displayed to the user interface (CLI, web UI, etc.).
- Hive organizes data into Databases (namespaces), tables (schemas), partitions (each unit defines its meaning), and clusters (known as Buckets).
- HIVE has user-defined Aggregations (UDAFs), user-defined table functions (UDTFs), and user-defined functions (UDFs) to expand Hive's SQL.

Query Optimization with Apache Hive

- Use effective schema with partitioning tables
- Use compression on the map and reduce outputs
- Bucketing for increased performance on I/O scans
- MapReduce jobs with parallelism
- Vectorization process can be implemented for processing batch of rows together.
- Hive has inbuilt support to SerDes and allows custom SerDes to be developed by users of Hive.

- Shared and Exclusive locks can be used for optimizing resource use, and concurrent writers and readers are supported.

Apache Hive Use Cases

- Amazon Elastic MapReduce.
- Yahoo is using Pig for data factory and Hive for data warehousing.

Apache Hive data types

- Column types (specific to columns only)
- Integral (TINYINT, SMALLINT, INT, and BIGINT)
- Strings (Varchar, Char)
- Timestamps (Date)
- Decimals (immutable arbitrary-precision as of JAVA)
- Literals (representation of fixed value)
- Floating
- Decimals (values higher than DOUBLE)
- NULL type
- Complex Types (Arrays, Maps, Structs, Union)
- Hive can also change the kind of data when megastore col. Type. Changes are set to false. If the conversion is successful, data will be displayed otherwise NULL.

Partitioning in Apache Hive

Partitioning is the technique to divide tables into parts based on columns data, the structure of the data, and the nature of the data based on its source of generation and storage -

- Partition keys help to identify partitions from a table
- Hive converts SQL queries into jobs of MapReduce to submit to the Hadoop cluster
- Created when data is inserted
- There are two different types of tables in the Hive - Internal (Managed)
- Manages data by default
- maintain Security at the Schema level
- Best for temporary data

External Tables

- The external warehouse directory needs to be mentioned when external tables are present.
- Apply security at the HDFS level
- When data is necessary even after the DROP statement
- Execution load is horizontal

- When data is stored internally in a Hadoop system, then Dropping a table causes deletion of metadata and data tables permanently

Static partitioning in Apache Hive

- When data file size is significant, then static partitions are preferred only.
- Big files are generally generated in HDFS.
- From filename, partition column values can be easily fetched
- Execute Load statement whenever Partition column value needs to specify.
- Insert each data file as an individual.
- When the partition is to be defined for one or two columns, only
- Alteration is possible on static partitions

Dynamic partitioning in Apache Hive

- Every row of data is read and then goes to a specific field in the table
- When ETL flow of data pipeline is necessary
- Partition column value needs not to be mention whenever LOAD statements are executed.
- Only single insert
- Loads data from a non-partitioned table
- It takes more time to load as data needs to be read row by row
- When the partition generation process isn't static
- Alteration isn't possible. Need to perform load operation again

What are the features of Apache Hive?

- Interactive shell with essential Linux commands support
- Beeline CLI, a JDBC client on SQLLine (used at HiveServer2)
- Use AvroSerde for Avro data in HIVE tables, HiveQL for ORC file formats
- Hive does not support some data types of AVRO, but AvroSerde automatically do the work for this
- Test cases of Various file formats in it.
- Test case outputs are directly or indirectly changed with test cases, data model usage, and hardware specification and configuration.

Following are some test cases (based on CERN's blog) of Kudu, AVRO, Parquet, HBase -

- Based on space utilization
- Kudu and Parquet serve the best compression size with Snappy and GZip algorithms.
- Based on the ingestion rate
- AVRO and Parquet are best as compared with the writing speed of others
- Based on Lookup latency of random data
- Kudu and Hbase are faster than others

- The scan rate of data
- AVRO has the best scan rate of data when non-key columns are in output Test Cases Output

Using the hybrid approach in its file system provides more efficient and Fast data processing with Structured or unstructured data for SerDes and Analytics on Data. The following are some useful points with some formats that use within the Hybrid approach.

- When used with snappy compression, parquet or kudu reduces the data volume by ten times than simple uncompressed serialization format.
- Random lookup operations case is best suitable for HBase or Kudu. Parquet is also fast, but resources use more.
- Analytic on data is the best use case for Parquet and Kudu
- HBase and Kudu can alter in-place data
- Avro is the quickest encoder for structured data. Best when all attributes of record are necessary.
- Parquet works seamlessly for scan and ingestion

SerDe in Apache Hive

Hive has some built-in SerDe for Avro, ORC, parquet, CSV, JsonSerDe, and Thrift. Custom support is also available for users. Hive engines automatically handle object conversion. Two types of processing are done for serialization and Deserialization of records.

Input Processing

- The Execution Engine uses the configured input format to read
- Then performs Deserialization
- that engine also holds object inspector
- Deserialized objects and inspectors are then passed to operators by the engine to manipulate records.

Output Processing

- Serialize method of Serde uses deserialized object and object inspector
- Then performs serialization on it in expected output format for writing

Applications of Data Serialization: Serialization allows a program to save the state of an object and recreate it when needed. Its common uses are:

- ***Persisting data onto files*** – happens mostly in language-neutral formats such as CSV or XML. However, most languages allow objects to be serialized directly into binary using APIs such as the Serializable interface in Java, fstream class in C++, or Pickle module in Python.
- ***Storing data into Databases*** – when program objects are converted into byte streams and then stored into DBs, such as in Java JDBC.
- ***Transferring data through the network*** – such as web applications and mobile apps passing on objects from client to server and vice versa.
- ***Remote Method Invocation (RMI)*** – by passing serialized objects as parameters to functions running on a remote machine as if invoked on a local machine. This data can be transmitted across domains through firewalls.

- ***Sharing data in a Distributed Object Model*** – when programs written in different languages (running on diverse platforms) need to share object data over a distributed network using frameworks such as COM and CORBA. However, SOAP, REST and other web services have replaced these applications now.

Text-based Data Serialization formats and their key features:

- ***XML (Extensible Markup Language) -*** Nested textual format. Human-readable and editable. Schema based validation. Used in metadata applications, web services data transfer, web publishing.
- ***CSV (Comma-Separated Values) -*** Table structure with delimiters. Human-readable textual data. Opens as spreadsheet or plaintext. Used as plaintext Database.
- ***JSON (JavaScript Object Notation) -*** Short syntax textual format with limited data types. Human-readable. Derived from JavaScript data formats. No need of a separate parser (like XML) since they map to JavaScript objects. Can be fetched with an XMLHttpRequest call. No direct support for DATE data type. All data is dynamically processed. Popular format for web API parameter passing. Mobile apps use this extensively for user interaction and database services.
- ***YAML (YAML Ain't Markup Language) -*** Lightweight text format. Human-readable. Supports comments and thus easily editable. Superset of JSON. Supports complex data types. Maps easily to native data structures. Used in configuration settings, document headers, Apps with need for MySQL style self-references in relational data.

Binary Data Serialization formats and their key features.

- ***BSON (Binary JSON) -*** Created and internally used by MongoDB. Binary format, not human-readable. Deals with attribute-value pairs like JSON. Includes datetime, bytearray and other data types not present in JSON. Used in web apps with rich media data types such as live video. Primary use is storage, not network communication.
- ***MessagePack*** - Designed for data to be transparently converted from/to JSON. Compressed binary format, not human-readable. Supports static typing. Supports RPC. Better JSON compatibility than BSON. Primary use is network communication, not storage. Used in apps with distributed file systems.
- ***protobuf (Protocol Buffers) -*** Created by Google. Binary message format that allows programmers to specify a schema for the data. Also includes a set of rules and tools to define and exchange these messages. Transparent data compression. Used in multi-platform applications due to easy interoperability between languages. Universal RPC framework. Used in performance-critical distributed applications.

--ooOoo--

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

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.

1. Hadoop Architecture:

Hadoop is a software framework that allows you to store and analyze large amounts of data. It was originally developed by Google to help them analyze large datasets and make sense of them. Today, it is used by many different companies to store, process, and analyze data. Hadoop is a highly scalable, open-source, distributed computing platform that allows you to store and process large amounts of data. It is used to store and analyze data from a variety of sources, including databases, web servers, and file systems.

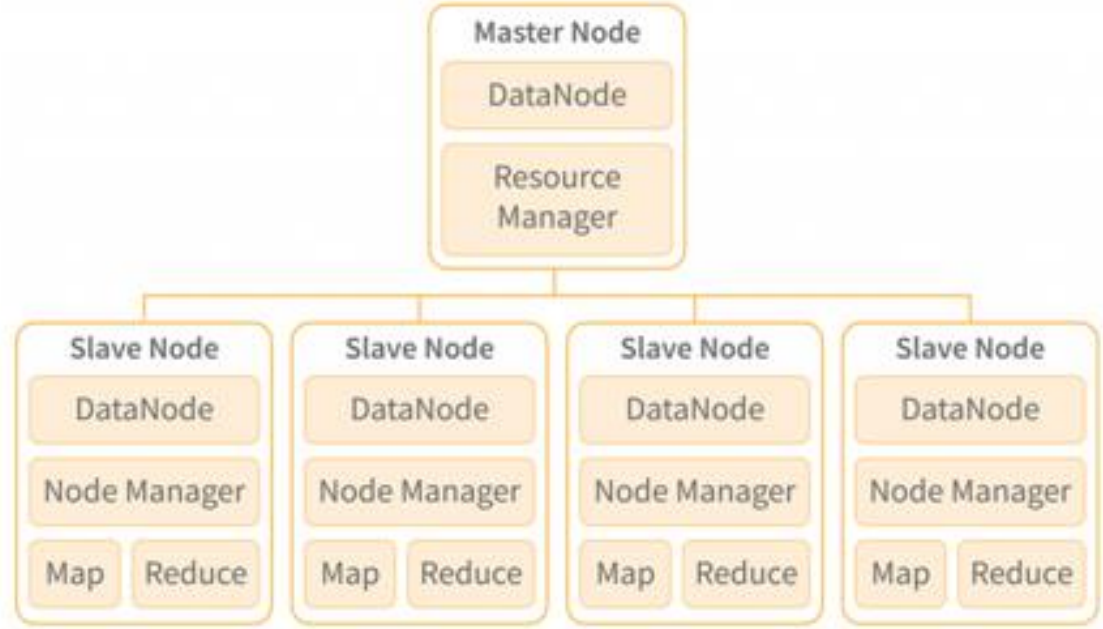
Hadoop is designed to be scalable by distributing the processing of data across a large number of computers. It also allows you to store and analyze data in a way that is faster than traditional methods. Hadoop is used to store and analyze data from a variety of sources, including databases, web servers, and file systems. It is designed to be scalable by distributing the processing of data across a large number of computers. It also allows you to store and analyze data in a way that is faster than traditional methods

1. HDFS

The master node keeps track of the status of all the data nodes. If a data node goes down, the master node takes over the processing of that block. The slave nodes process the data on their own. HDFS requires a high-speed Internet connection. It is usually best to have at least a 10 Mbps network connection. HDFS works on a time-based algorithm, which means that every block is processed in a predetermined time interval. This provides a high degree of scalability as all the nodes process the data in real time. HDFS is a great solution for data warehouse and business intelligence tasks. It is also a good solution for high-volume, high-frequency data analysis.

i) NameNode and DataNode:

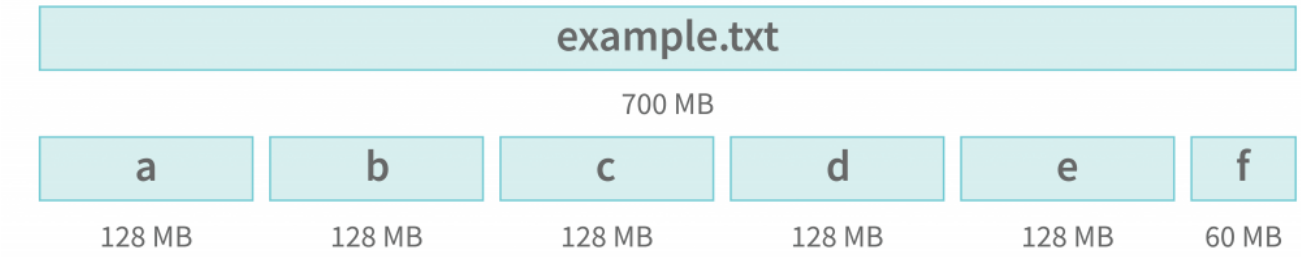
When a client connection is received from a master server, the NameNode running on the master server looks into both the local namespace and the master server's namespace to find the matching records. The NameNode running on the master server then executes the lookup and returns a list of records that match the query. DataNodes then get the records and start storing them. A data block is a record of data. Data nodes use the data blocks to store different types of data, such as text, images, videos, etc. NameNode maintains data node connections with clients based on the replication status of DataNodes. If a DataNode goes down, the client can still communicate with the NameNode. The client then gets the latest list of data blocks from the NameNode and communicates with the DataNode that has the newest data block.



Thus, DataNode is a compute-intensive task. It is therefore recommended to use the smallest possible DataNode. As DataNode stores data, it is recommended to choose a node that is close to the centre of the data. In a distributed system, all the nodes have to run the same version of Java, so it is recommended to use open-source Java. If there are multiple DataNodes, then they are expected to work in tandem to achieve a certain level of performance.

ii) Block in HDFS

A file stored in Hadoop has the default 128 MB or 256 MB block size.

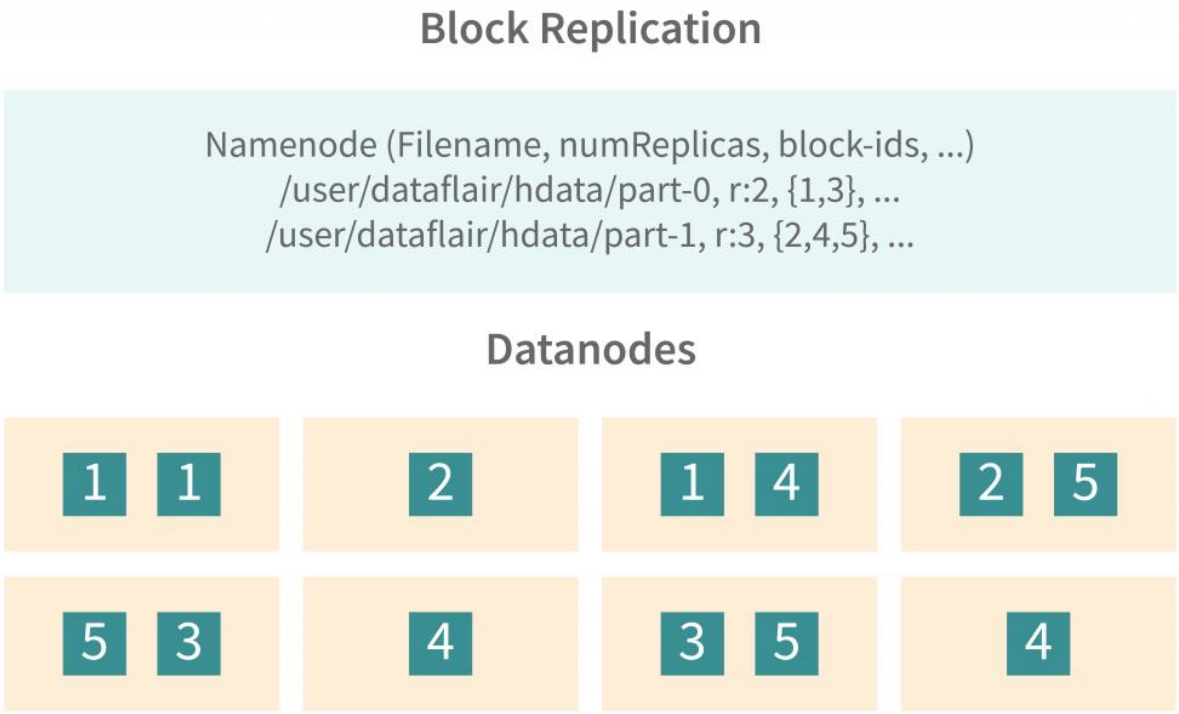


We have to ensure that the storage consumed by our application doesn't exceed the level of data storage. If the storage consumed by our application is too much, then we have to choose our block size to avoid excessive metadata growth. If you are using a standard block size of 16KB, then you are good to go. You won't even have to think about it. Our HDFS block size will be chosen automatically. However, if we are using a large block size, then we have to think about the metadata explosion. We can either keep the metadata as small as possible or keep it as large as possible. HDFS has the option of keeping the metadata as large as possible.

iii) Replication Management

When a node fails, the data stored on it is copied to another healthy node. This process is known as "replication". If a DataNode fails, the data stored on it is not lost. It is simply stored on another DataNode. This is a good thing because it helps in the high availability of data. When you are using a DataNode as the primary storage for your data, you must

keep in mind that it is a highly-available resource. If the primary storage for your data is a DataNode, you must make sure to have a proper backup and restoration mechanism in place.

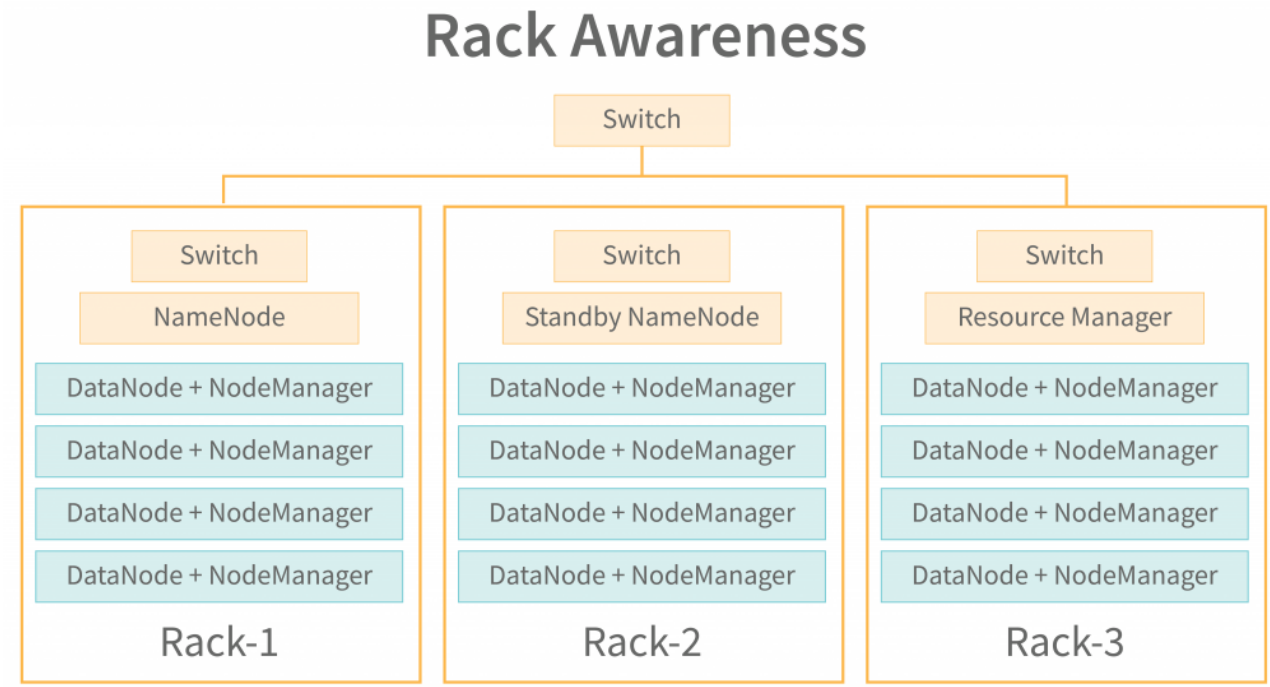


A given file can have a replication factor of 3 or 1, but it will require 3 times the storage if we keep using a replication factor of 3. The NameNode keeps track of each data node's block report and whenever a block is under-or over-replicated, it adds or removes replicas accordingly.

iv.Rack Awareness

When a block is deleted from a rack, the next available block will be placed on the rack. When a block is updated, the previous block is automatically updated in the same rack. This ensures data consistency across the storage network. In case of a fault in any of the data storage nodes, other storage nodes can be alerted through the rack awareness algorithm to take over the responsibility of the failed node. This helps in providing failover capability across the storage network.

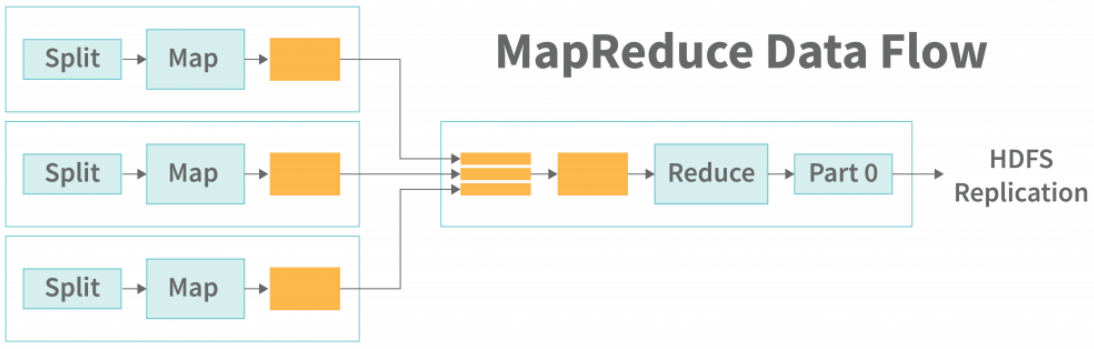
This helps in providing high availability of data storage. As data stored in HDFS is massive, it makes sense to use multiple storage nodes for high-speed data access. HDFS uses a distributed data store architecture to provide high-speed data access to its users. This distributed data store architecture allows for parallel processing of data in multiple data nodes. This parallel processing of data allows for high-speed storage of large data sets.



2. MapReduce

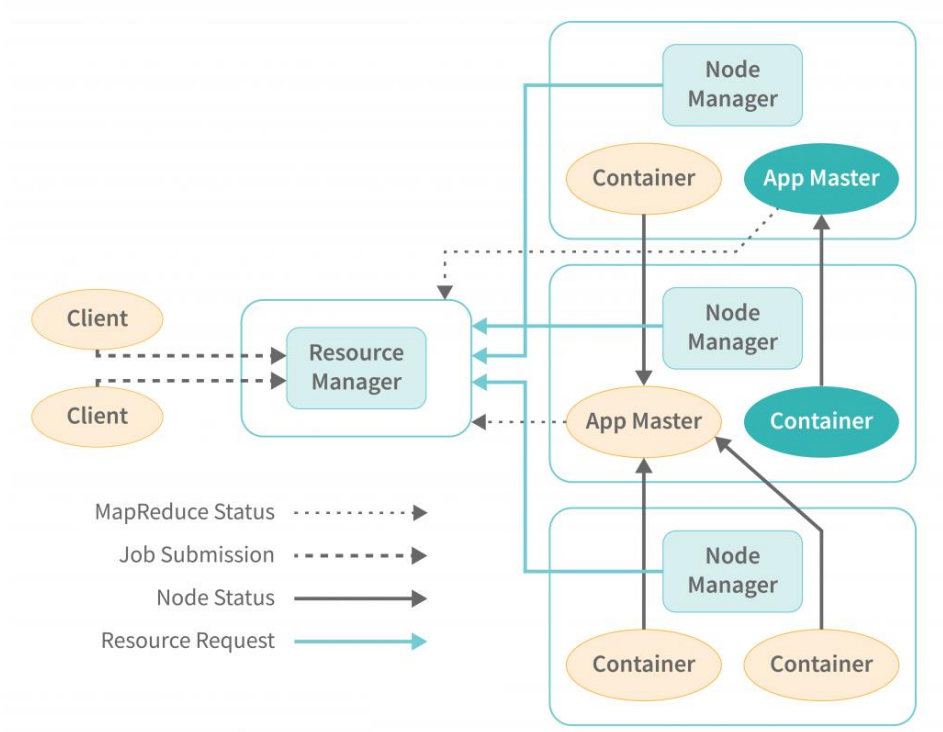
MapReduce is a data processing language and software framework that allows you to process large amounts of data in a reliable and efficient manner. It is a great fit for data-intensive, real-time and/or streaming applications. Basically, MapReduce allows you to partition your data and process items only when they are needed. A map-reduce job consists of several maps and reduces functions. Each map function generates, parses, transforms, and filters data before passing it on to the next function. The reduced function groups, aggregates, and partitioning of this intermediate data from the map functions.

The map task runs on the same node as the input source. Map tasks are responsible for generating summary statistics about data in the form of a report. The report can be viewed on a web browser or printed. The output of a map task is the same as that of a reduced task. The only difference is that the map task returns a result whereas the reduced task returns a data structure that is applicable for further analysis. A map task is usually repetitive and is triggered when the data volume on the source is greater than the volume of data that can be processed in a short period of time.



3. Yarn

YARN, also known as Yet Another Resource Negotiator, is a resource management and job scheduling/monitoring daemon in Hadoop. A resource manager in YARN isolates resource management from job scheduling and monitoring. A global Resource Manager oversees operations for the entire YARN network, including per-application ApplicationMaster. A job or a DAG of jobs may be defined as an application. The Resource Manager manages resources for all the competing applications in the YARN framework. The NodeManager monitors resource usage by the container and passes it on to ResourceManger. There are resources such as CPU, memory, disk, and connectivity, among others. To perform and monitor the application, the ApplcationMaster talks to the ResourceManager and the NodeManager to handle and manage resources.



Advantages of Hadoop Architecture

- The huge data sets that Hadoop can store and distribute across hundreds of inexpensive servers are also capable of processing a great deal of data. Unlike RDBMSs which can't function as a scaling storage platform, Hadoop enables businesses to run applications on thousands of nodes where many thousands of terabytes of data are being processed.
- A company's exploding data sets are a costly headache in traditional relational database management systems. This is because computing such enormous quantities of data is extremely costly. To cut costs, some companies have pasted data down-sample as they wished, assuming that certain types of data were more important than others. This

procedure could have worked in the short term, but it might have destroyed the entire raw data set as it was too costly to keep.

- A single data source (such as social media) can be accessed by a business using Hadoop, providing access to a large variety of data types (structured and unstructured). Businesses can use Hadoop to gain new business insights from data sources such as social media, and email communications. Hadoop can be used for a range of purposes, from log processing to data warehousing to fraud detection, among others.
- Hadoop's distributed file system "maps" data wherever it is located on a cluster, therefore reducing storage costs. It also makes data processing quicker because the tools for data processing are often on the same servers where the data is located. Big volumes of unstructured data can be processed effectively using Hadoop in minutes or hours, depending on how much data is involved.
- Hadoop's fault tolerance is a significant advantage. When data is transferred to individual nodes in the cluster, it is also duplicated to other nodes in the event of failure, so that when the node is down, there is another copy available to use.

Disadvantages of Hadoop Architecture

- A complex Hadoop application like security is hard to manage. Hardly any straightforward example can be found in the security model. If anyone is incapable of enabling it, your data may be at risk. Hadoop is also lacking encryption at the storage and network levels, despite its popularity as a government surveillance tool.
- The very nature of Hadoop makes it a risky endeavour to run it. Because of Java's widespread use and continued controversy, it has been extensively abused by cybercriminals and other miscreants.
- Big data is not primarily used by big corporations, however small data platforms such as Hadoop are not equipped for it. The Hadoop Distributed File System, because of its capacity, is unable to efficiently handle small files. For example, small quantities of data cannot be stored in the Hadoop Distributed File System.
- It is important for organizations to ensure that they are running the latest stable version of Hadoop or that they use a third-party vendor that can deal with issues like these.
- A possible solution is described in the article. Each of these platforms can improve the efficiency and reliability of data collection, aggregation, and integration by adding value to it. The main idea is that companies could be missing out on big benefits by using Hadoop alone.

2. Hadoop- RDBMS Vs Hadoop:

RDMS (Relational Database Management System): RDBMS is an information management system, which is based on a data model.In RDBMS tables are used for information storage. Each row of the table represents a record and column represents an attribute of data. Organization of data and their manipulation processes are different in RDBMS from other databases. RDBMS ensures ACID (atomicity, consistency, integrity, durability) properties

required for designing a database. The purpose of RDBMS is to store, manage, and retrieve data as quickly and reliably as possible.

Hadoop: It is an open-source software framework used for storing data and running applications on a group of commodity hardware. It has large storage capacity and high processing power. It can manage multiple concurrent processes at the same time. It is used in predictive analysis, data mining and machine learning. It can handle both structured and unstructured form of data. It is more flexible in storing, processing, and managing data than traditional RDBMS. Unlike traditional systems, Hadoop enables multiple analytical processes on the same data at the same time. It supports scalability very flexibly.

Below is a table of differences between RDBMS and Hadoop:

S. No.	RDBMS	Hadoop
1.	Traditional row-column based databases, basically used for data storage, manipulation and retrieval.	An open-source software used for storing data and running applications or processes concurrently.
2.	In this structured data is mostly processed.	In this both structured and unstructured data is processed.
3.	It is best suited for OLTP environment.	It is best suited for BIG data.
4.	It is less scalable than Hadoop.	It is highly scalable.
5.	Data normalization is required in RDBMS.	Data normalization is not required in Hadoop.
6.	It stores transformed and aggregated data.	It stores huge volume of data.
7.	It has no latency in response.	It has some latency in response.
8.	The data schema of RDBMS is static type.	The data schema of Hadoop is dynamic type.
9.	High data integrity available.	Low data integrity available than RDBMS.
10.	Cost is applicable for licensed software.	Free of cost, as it is an open source software.

3. Hadoop Overview- Hadoop distributors:

Hadoop is Apache software so it is freely available for download and use. But, this is very akin to Linux a few years back and Linux distributions like RedHat, Suse and Ubuntu. The software is free to download and use but distributions offer an easier to use bundle. The Hadoop Distributors offers the following:

- **Distributions provide easy to install mediums like RPMs**

  The Apache version of Hadoop is just TAR balls. Distros actually package it nicely into easy to install packages which make it easy for system administrators to manage effectively.

- **Distros package multiple components that work well together**

  The Hadoop ecosystem contains a lot of components (HBase, Pig, Hive, Zookeeper, etc.) which are being developed independently and have their own release schedules. Also, there are version dependencies among the components. For example version 0.92 of HBase needs a particular version of HDFS. Distros bundle versions of components that work well together. This provides a working Hadoop installation right out of the box.

- **Tested:** Distro makers strive to ensure good quality components.
- **Performance patches:** Sometimes, distros lead the way by including performance patches to the 'vanilla' versions.

- **Predictable upgrade path:** Distros have predictable product release road maps. This ensures they keep up with developments and bug fixes.
- **And most importantly . . SUPPORT:** Lot of distros come with support, which could be very valuable for a production critical cluster.

\The Top Hadoop Distributors are …

Distributor	Remarks	Free / Premium
Apache hadoop.apache.org	○ The Hadoop Source ○ No packaging except TAR balls ○ No extra tools	Completely free and open source
Cloudera www.cloudera.com	○ Oldest distro ○ Very polished ○ Comes with good tools to install and manage a Hadoop cluster	Free / Premium model (depending on cluster size)
HortonWorks www.hortonworks.com	○ Newer distro ○ Tracks Apache Hadoop closely ○ Comes with tools to manage and administer a cluster	Completely open source
MapR www.mapr.com	○ MapR has their own file system (alternative to HDFS) ○ Boasts higher performance ○ Nice set of tools to manage and administer a cluster ○ Does not suffer from Single Point of Failure ○ Offer some cool features like mirroring, snapshots, etc.	Free / Premium model
Intel hadoop.intel.com	○ Encryption support ○ Hardware acceleration added to some layers of stack to boost performance ○ Admin tools to deploy and manage Hadoop	Premium
Pivotal HD gopivotal.com	○ fast SQL on Hadoop ○ software only or appliance	Premium

Hadoop Distributors in the Cloud:

- **Hadoop clusters in the Cloud:** Hadoop clusters can be set up in any cloud service that offers suitable machines. However, in line with the cloud mantra 'only pay for what you use', Hadoop can be run 'on demand' in the cloud.
- **Amazon Elastic Map Reduce:** Amazon offers 'On Demand Hadoop', which means there is no permanent Hadoop cluster. A cluster is spun up to do a job and after that it is shut down - 'pay for usage'. Amazon offers a slightly customized version of Apache Hadoop and also offers MapR's distribution.
- **Google's Compute Engine:** Google offers MapR's Hadoop distribution in their Compute Engine Cloud.
- SkyTab Cloud: SkyTap offers deploy-able Hadoop templates

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

4. HDFS - HDFS Daemons:

HDFS:

With growing data velocity the data size easily outgrows the storage limit of a machine. A solution would be to store the data across a network of machines. Such filesystems are called distributed filesystems. Since data is stored across a network all the complications of a network come in.

This is where Hadoop comes in. It provides one of the most reliable filesystems. HDFS (Hadoop Distributed File System) is a unique design that provides storage for extremely large files with streaming data access pattern and it runs on commodity hardware. Let's elaborate the terms:

- *Extremely large files:* Here we are talking about the data in range of petabytes(1000 TB).
- *Streaming Data Access Pattern:* HDFS is designed on principle of write-once and read-many-times. Once data is written large portions of dataset can be processed any number times.
- *Commodity hardware:* Hardware that is inexpensive and easily available in the market. This is one of feature which specially distinguishes HDFS from other file system.

Nodes: Master-slave nodes typically forms the HDFS cluster.

- **NameNode(MasterNode):**
  - ✓ Manages all the slave nodes and assign work to them.
  - ✓ It executes filesystem namespace operations like opening, closing, renaming files and directories.
  - ✓ It should be deployed on reliable hardware which has the high config. not on commodity hardware.
- **DataNode(SlaveNode):**
  - ✓ Actual worker nodes, who do the actual work like reading, writing, processing etc.
  - ✓ They also perform creation, deletion, and replication upon instruction from the master.
  - ✓ They can be deployed on commodity hardware.

HDFS Daemons: Daemons are the processes running in background.

Daemons mean **Process**. Hadoop Daemons are a set of processes that run on Hadoop. Hadoop is a framework written in Java, so all these processes are Java Processes.

Apache Hadoop 2 consists of the following Daemons:

1. NameNode
2. DataNode
3. Secondary Name Node
4. Resource Manager
5. Node Manager

Namenode, Secondary NameNode, and Resource Manager work on a Master System while the Node Manager and DataNode work on the Slave machine.

1. NameNode

NameNode works on the Master System. The primary purpose of Namenode is to manage all the MetaData. Metadata is the list of files stored in HDFS(Hadoop Distributed File System). As we know the data is stored in the form of blocks in a Hadoop cluster. So the DataNode on which or the location at which that block of the file is stored is mentioned in MetaData. All information regarding the logs of the transactions happening in a Hadoop cluster (when or who read/wrote the data) will be stored in MetaData. MetaData is stored in the memory.

Features:

- It never stores the data that is present in the file.
- As Namenode works on the Master System, the Master system should have good processing power and more RAM than Slaves.
- It stores the information of DataNode such as their Block id's and Number of Blocks

How to start Name Node?

hadoop-daemon.sh start namenode

How to stop Name Node?

hadoop-daemon.sh stop namenode

```
dikshant@dikshant-Inspiron-5567:~$ hadoop-daemon.sh start namenode
starting namenode, logging to /home/dikshant/Documents/hadoop/logs/hadoop-dikshan
out
dikshant@dikshant-Inspiron-5567:~$ hadoop-daemon.sh stop namenode
stopping namenode
```

2. DataNode

DataNode works on the Slave system. The NameNode always instructs DataNode for storing the Data. DataNode is a program that runs on the slave system that serves the read/write request from the client. As the data is stored in this DataNode, they should possess high memory to store more Data.

How to start Data Node?

hadoop-daemon.sh start datanode

How to stop Data Node?

hadoop-daemon.sh stop datanode

```
dikshant@dikshant-Inspiron-5567:~$ hadoop-daemon.sh start datanode
starting datanode, logging to /home/dikshant/Documents/hadoop/logs/hadoop-dikshant-datanode-dikshant-
out
dikshant@dikshant-Inspiron-5567:~$ hadoop-daemon.sh stop datanode
stopping datanode
dikshant@dikshant-Inspiron-5567:~$
```

3. Secondary NameNode

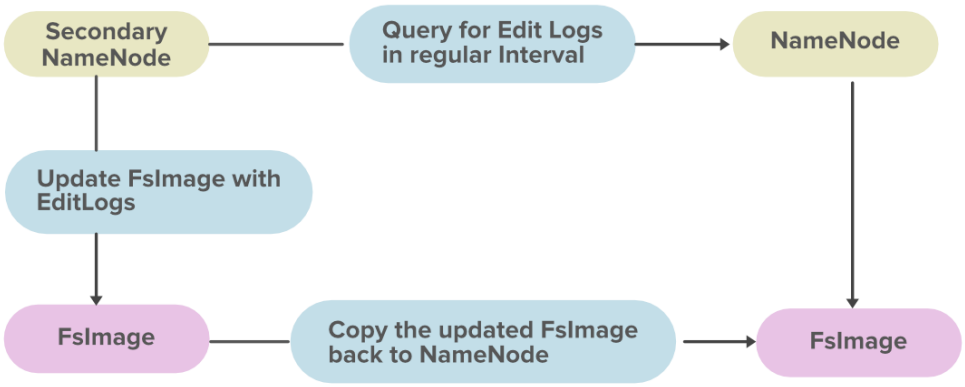
Secondary NameNode is used for taking the hourly backup of the data. In case the Hadoop cluster fails, or crashes, the secondary Namenode will take the hourly backup or checkpoints of that data and store this data into a file name *fsimage*. This file then gets transferred to a new system. A new MetaData is assigned to that new system and

a new Master is created with this MetaData, and the cluster is made to run again correctly. This is the benefit of Secondary Name Node. Now in Hadoop2, we have High-Availability and Federation features that minimize the importance of this Secondary Name Node in Hadoop2.

Major Function Of Secondary NameNode:

- It groups the Edit logs and Fsimage from NameNode together.
- It continuously reads the MetaData from the RAM of NameNode and writes into the Hard Disk.

As secondary NameNode keeps track of checkpoints in a Hadoop Distributed File System, it is also known as the checkpoint Node.



The Hadoop Daemon's	Port
Name Node	50070
Data Node	50075
Secondary Name Node	50090

These ports can be configured manually in *hdfs-site.xml* and *mapred-site.xml* files.

4. Resource Manager

Resource Manager is also known as the Global Master Daemon that works on the Master System. The Resource Manager Manages the resources for the applications that are running in a Hadoop Cluster. The Resource Manager Mainly consists of 2 things.

a. **ApplicationsManager**

b. **Scheduler**

An Application Manager is responsible for accepting the request for a client and also makes a memory resource on the Slaves in a Hadoop cluster to host the *Application Master*. The scheduler is utilized for providing resources for applications in a Hadoop cluster and for monitoring this application.

How to start ResourceManager? **yarn-daemon.sh start resourcemanager**

How to stop ResourceManager? **stop:yarn-daemon.sh stop resoucemnager**

```
dikshant@dikshant-Inspiron-5567:~$ yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/dikshant/Documents/hadoop/logs/yarn-dikshant-
spiron-5567.out
dikshant@dikshant-Inspiron-5567:~$ yarn-daemon.sh stop resourcemanager
stopping resourcemanager
```

5. Node Manager

The Node Manager works on the Slaves System that manages the memory resource within the Node and Memory Disk. Each Slave Node in a Hadoop cluster has a single NodeManager Daemon running in it. It also sends this monitoring information to the Resource Manager.

How to start Node Manager? **yarn-daemon.sh start nodemanager**

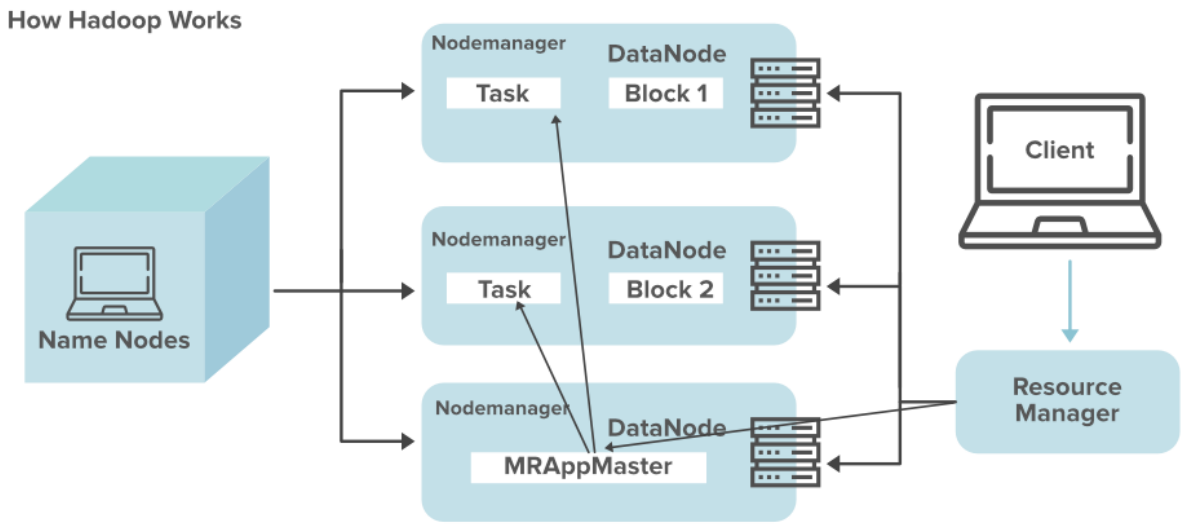
How to stop Node Manager? **yarn-daemon.sh stop nodemanager**

```
dikshant@dikshant-Inspiron-5567:~$ yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/dikshant/Documents/hadoop/logs/yarn-dikshant-no
567.out
dikshant@dikshant-Inspiron-5567:~$ yarn-daemon.sh stop nodemanager
stopping nodemanager
```

In a Hadoop cluster, Resource Manager and Node Manager can be tracked with the specific URLs, of type ***http://:port_number***

The Hadoop Daemon's	Port
ResourceManager	8088
NodeManager	8042

The below diagram shows how Hadoop works.



Terms related to HDFS:

- ✓ ***HeartBeat:*** It is the signal that datanode continuously sends to namenode. If namenode doesn't receive heartbeat from a datanode then it will consider it dead.
- ✓ ***Balancing***: If a datanode is crashed the blocks present on it will be gone too and the blocks will be under-replicated compared to the remaining blocks. Here master node(namenode) will give a signal to datanodes containing replicas of those lost blocks to replicate so that overall distribution of blocks is balanced.
- ✓ ***Replication:*** It is done by datanode.

Note: No two replicas of the same block are present on the same datanode.

Features:

- ✓ Distributed data storage.
- ✓ Blocks reduce seek time.
- ✓ The data is highly available as the same block is present at multiple datanodes.
- ✓ Even if multiple datanodes are down we can still do our work, thus making it highly reliable.
- ✓ High fault tolerance.

Limitations: Though HDFS provide many features there are some areas where it doesn't work well.

- ✓ ***Low latency data access:*** Applications that require low-latency access to data i.e in the range of milliseconds will not work well with HDFS, because HDFS is designed keeping in mind that we need high-throughput of data even at the cost of latency.
- ✓ ***Small file problem:*** Having lots of small files will result in lots of seeks and lots of movement from one datanode to another datanode to retrieve each small file, this whole process is a very inefficient data access pattern.

5. Anatomy of File Write and Read:

Big data is nothing but a collection of data sets that are large, complex, and which are difficult to store and process using available data management tools or traditional data processing applications. Hadoop is a framework (open source) for writing, running, storing, and processing large datasets in a parallel and distributed manner. It is a solution that is used to overcome the challenges faced by big data. Hadoop has two components:

a) HDFS (Hadoop Distributed File System)

b) YARN (Yet Another Resource Negotiator)

In this article, we focus on one of the components of Hadoop i.e., HDFS and the anatomy of file reading and file writing in HDFS. HDFS is a file system designed for storing very large files (files that are hundreds of megabytes, gigabytes, or terabytes in size) with streaming data access, running on clusters of commodity hardware(commonly available hardware that can be obtained from various vendors). In simple terms, the storage unit of Hadoop is called HDFS.

HDFS Characteristics:

- ✓ Fault-Tolerance

- ✓ Scalability
- ✓ Distributed Storage
- ✓ Reliability
- ✓ High availability
- ✓ Cost-effective
- ✓ High throughput

Building Blocks of Hadoop:

- ✓ Name Node
- ✓ Data Node
- ✓ Secondary Name Node (SNN)
- ✓ Job Tracker
- ✓ Task Tracker

Anatomy of File Read in HDFS: Let's get an idea of how data flows between the client interacting with HDFS, the name node, and the data nodes with the help of a diagram. Consider the figure:

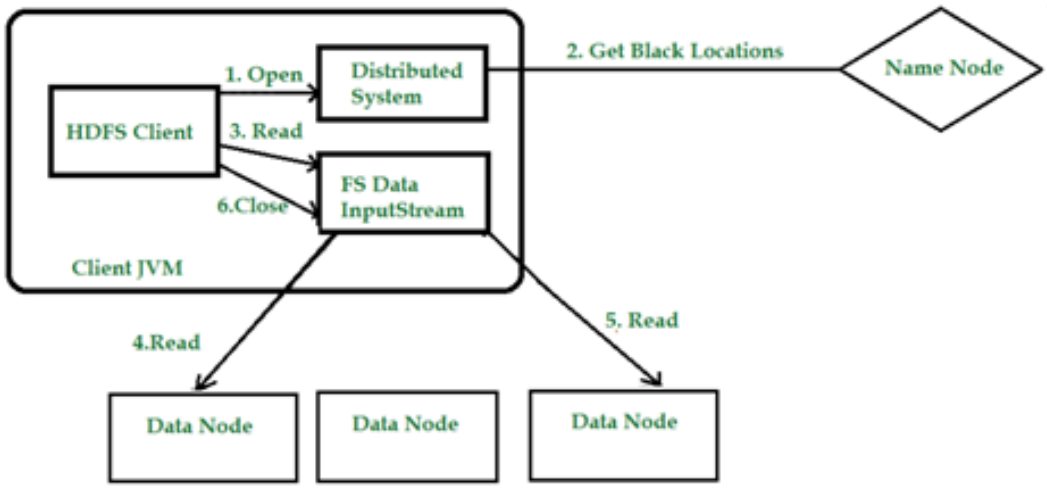


Fig: HDFS Read

- ***Step 1:*** The client opens the file it wishes to read by calling open() on the File System Object(which for HDFS is an instance of Distributed File System).
- ***Step 2:*** Distributed File System(DFS) calls the name node, using remote procedure calls (RPCs), to determine the locations of the first few blocks in the file. For each block, the name node returns the addresses of the data nodes that have a copy of that block. The DFS returns an FSDataInputStream to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the data node and name node I/O.
- ***Step 3:*** The client then calls read() on the stream. DFSInputStream, which has stored the info node addresses for the primary few blocks within the file, then connects to the primary (closest) data node for the primary block in the file.

- Step 4: Data is streamed from the data node back to the client, which calls read() repeatedly on the stream.
- ***Step 5:*** When the end of the block is reached, DFSInputStream will close the connection to the data node, then finds the best data node for the next block. This happens transparently to the client, which from its point of view is simply reading an endless stream. Blocks are read as, with the DFSInputStream opening new connections to data nodes because the client reads through the stream. It will also call the name node to retrieve the data node locations for the next batch of blocks as needed.
- ***Step 6:*** When the client has finished reading the file, a function is called, close() on the FSDataInputStream.

Anatomy of File Write in HDFS: Let's check out how files are written to HDFS. Consider the following figure to get a better understanding of the concept.

Note: HDFS follows the Write once Read many times model. In HDFS we cannot edit the files which are already stored in HDFS, but we can append data by reopening the files.

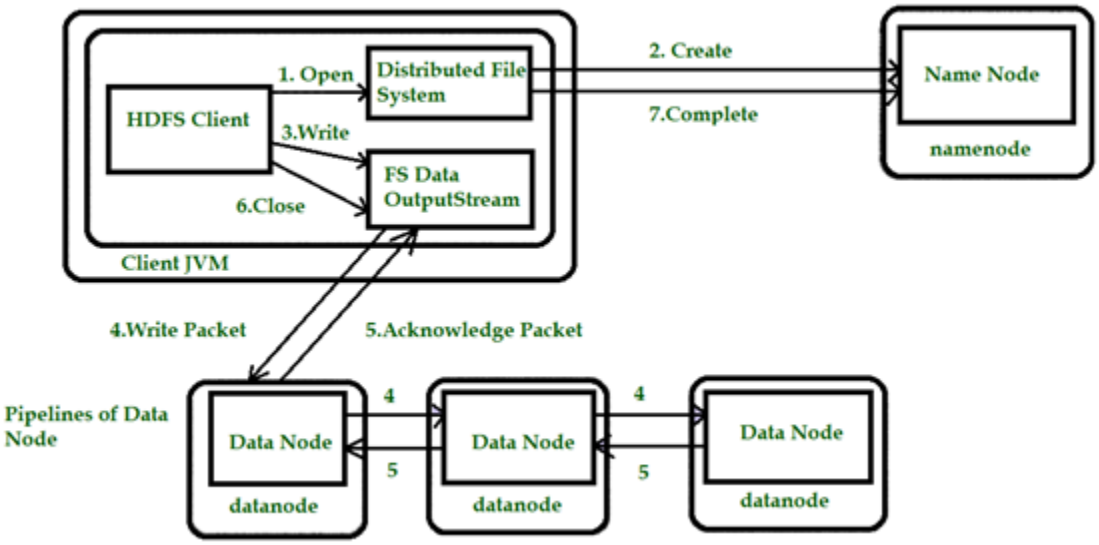


Fig: HDFS Write

- ***Step 1:*** The client creates the file by calling create() on DistributedFileSystem(DFS).
- ***Step 2:*** DFS makes an RPC call to the name node to create a new file in the file system's namespace, with no blocks associated with it. The name node performs various checks to make sure the file doesn't already exist and that the client has the right permissions to create the file. If these checks pass, the name node prepares a record of the new file; otherwise, the file can't be created and therefore the client is thrown an error i.e. IOException. The DFS returns an FSDataOutputStream for the client to start out writing data to.
- ***Step 3:*** Because the client writes data, the DFSOutputStream splits it into packets, which it writes to an indoor queue called the info queue. The data queue is consumed by the DataStreamer, which is liable for asking the name node to allocate new blocks by picking an inventory of suitable data nodes to store the replicas. The list of data

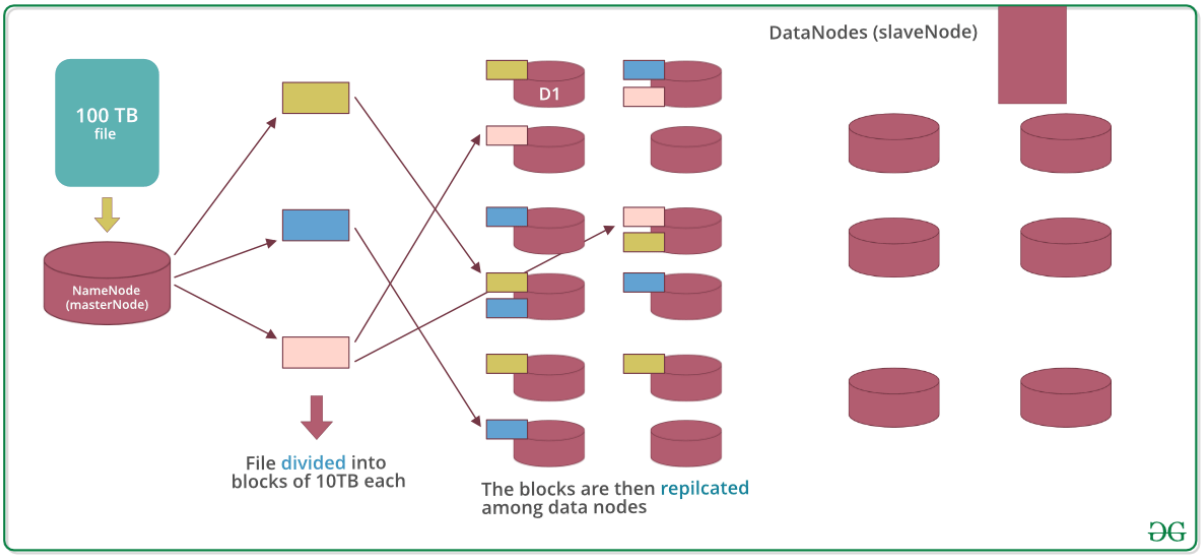
nodes forms a pipeline, and here we'll assume the replication level is three, so there are three nodes in the pipeline. The DataStreamer streams the packets to the primary data node within the pipeline, which stores each packet and forwards it to the second data node within the pipeline.

- ***Step 4:*** Similarly, the second data node stores the packet and forwards it to the third (and last) data node in the pipeline.
- ***Step 5:*** The DFSOutputStream sustains an internal queue of packets that are waiting to be acknowledged by data nodes, called an "ack queue".
- ***Step 6:*** This action sends up all the remaining packets to the data node pipeline and waits for acknowledgments before connecting to the name node to signal whether the file is complete or not.

HDFS follows Write Once Read Many models. So, we can't edit files that are already stored in HDFS, but we can include them by again reopening the file. This design allows HDFS to scale to a large number of concurrent clients because the data traffic is spread across all the data nodes in the cluster. Thus, it increases the availability, scalability, and throughput of the system.

6. Name Node, Secondary Name Node, and Data Node:

Data storage in HDFS: Now let's see how the data is stored in a distributed manner.



Lets assume that 100TB file is inserted, then masternode(namenode) will first divide the file into blocks of 10TB (default size is 128 MB in Hadoop 2.x and above). Then these blocks are stored across different datanodes(slavenode). Datanodes(slavenode)replicate the blocks among themselves and the information of what blocks they contain is sent to the master. Default replication factor is 3 means for each block 3 replicas are created (including itself). In hdfs.site.xml we can increase or decrease the replication factor i.e we can edit its configuration here.

Note: MasterNode has the record of everything, it knows the location and info of each and every single data nodes and the blocks they contain, i.e. nothing is done without the permission of masternode.

Namenodes:

- ✓ Run on the master node.
- ✓ Store metadata (data about data) like file path, the number of blocks, block Ids. etc.
- ✓ Require high amount of RAM.
- ✓ Store meta-data in RAM for fast retrieval i.e to reduce seek time. Though a persistent copy of it is kept on disk.

Secondary NameNode

Secondary NameNode keeps a copy of FSIMAGE. Periodically Secondary NameNode will get the copy of the FSIMAGE file and the temporary log file from the NameNode and apply the log file to the FSIMAGE file. There by bringing the FSIMAGE file current. This relieves the NameNode from worrying about merging the contents of FSIMAGE with the temporary log file.

Secondary NameNode however doesn't take over the functions of the NameNode if the NameNode encounters an issue. Secondary NameNode can be manually made the primary NameNode but it doesn't happen automatically. Secondary NameNode is also an old concept. Newer versions of Hadoop support High Availability capabilities with Quorum Journal Manager (QJM) or NFS (shared storage).

DataNodes:

- ✓ Run on slave nodes.
- ✓ Require high memory as data is actually stored here.

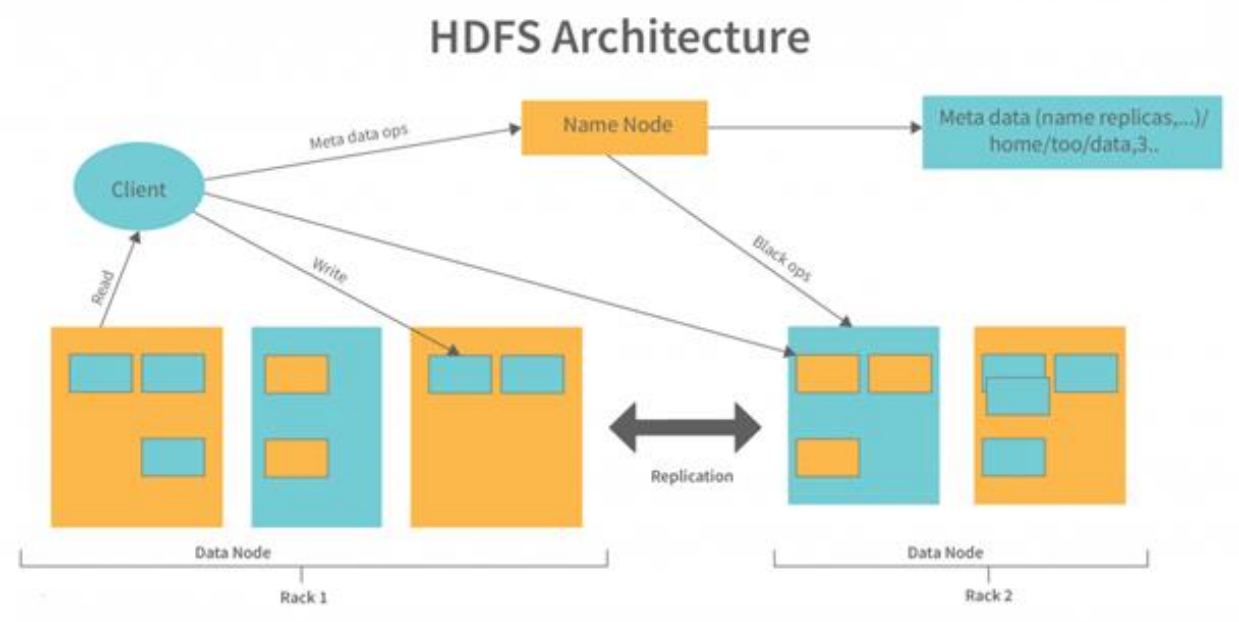
7. HDFS Architecture:

Hadoop is a software framework that enables distributed storage and processing of large data sets. It consists of several open source projects, including HDFS, MapReduce, and Yarn. While Hadoop can be used for different purposes, the two most common are Big Data analytics and NoSQL database management. HDFS stands for "Hadoop Distributed File System" and is a decentralized file system that stores data across multiple computers in a cluster. This makes it ideal for large-scale storage as it distributes the load across multiple machines so there's less pressure on each individual machine. MapReduce is a programming model that allows users to write code once and execute it across many servers. When combined with HDFS, MapReduce can be used to process massive data sets in parallel by dividing work up into smaller chunks and executing them simultaneously.

HDFS Architecture

HDFS is an Open source component of the Apache Software Foundation that manages data. HDFS has scalability, availability, and replication as key features. Name nodes, secondary name nodes, data nodes, checkpoint nodes, backup nodes, and blocks all make up the architecture of HDFS. HDFS is fault-tolerant and is replicated. Files are distributed across the cluster systems using the Name node and Data Nodes. The primary difference between Hadoop

and Apache HBase is that Apache HBase is a non-relational database and Apache Hadoop is a non-relational data store.



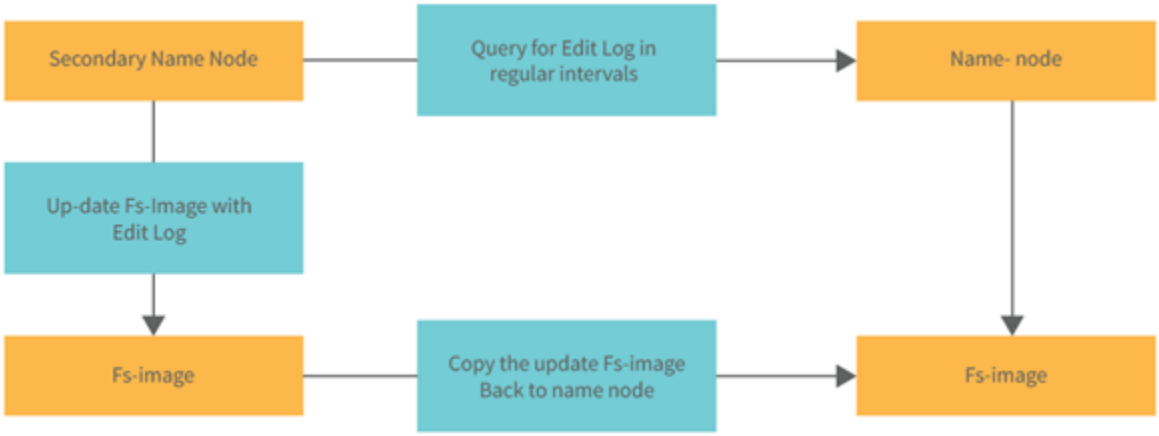
NameNode:

All the blocks on DataNodes are handled by NameNode, which is known as the master node. It performs the following functions:

1. Monitor and control all the DataNodes instances.
2. Permits the user to access a file.
3. Stores all of the block records on a DataNode instance.
4. EditLogs are committed to disk after every write operation to Name Node's data storage. The data is then replicated to all the other data nodes, including Data Node and Backup Data Node. In the event of a system failure, EditLogs can be manually recovered by Data Node.
5. All of the DataNodes' blocks must be alive in order for all of the blocks to be removed from the data nodes.
6. Therefore, every UpdateNode in a cluster is aware of every DataNode in the cluster, but only one of them is actively managing communication with all the DataNodes. Since every DataNode runs their own software, they are completely independent. Therefore, if a DataNode fails, the DataNode will be replaced by another DataNode. This means that the failure of a DataNode will not impact the rest of the cluster, since all the DataNodes are aware of every DataNode in the cluster.

There are two kinds of files in NameNode: FsImage files and EditLogs files:

1. **FsImage:** It contains all the details about a filesystem, including all the directories and files, in a hierarchical format. It is also called a file image because it resembles a photograph.
2. **EditLogs:** The EditLogs file keeps track of what modifications have been made to the files of the filesystem.



Secondary NameNode:

When NameNode runs out of disk space, a secondary NameNode is activated to perform a checkpoint. The secondary NameNode performs the following duties.

1. It stores all the transaction log data (from all the source databases) into one location so that when you want to replay it, it is at one single location. Once the data is stored, it is replicated across all the servers, either directly or via a distributed file system.
2. The information stored in the filesystem is replicated across all the cluster nodes and stored in all the data nodes. Data nodes store the data. The cluster nodes store the information about the cluster nodes. This information is called metadata. When a data node reads data from the cluster, it uses the metadata to determine where to send the data and what type of data it is. This metadata is also written to a hard drive. The cluster nodes will write this information if the cluster is restarted. The cluster will read this information and use it to determine where to send the data and what type of data it is.
3. The FsImage can be used to create a new replica of data, which can be used to scale up the data. If the new FsImage needs to be used to create a new replica, this replication will start with a new FsImage. There are some cases when it is necessary to recover from a failed FsImage. In this situation, a new FsImage must be created from an old one. The FsImage can be used to create backups of data. Data stored in the Hadoop cluster can be backed up and stored in another Hadoop cluster, or the data can be stored on a local file system.

DataNode:

Every slave machine that contains data organsises a DataNode. DataNode stores data in ext3 or ext4 file format on DataNodes. DataNodes do the following:

1. DataNodes store every data.
2. It handles all of the requested operations on files, such as reading file content and creating new data, as described above.
3. All the instructions are followed, including scrubbing data on DataNodes, establishing partnerships, and so on.

Checkpoint Node:

It establishes checkpoints at specified intervals to generate checkpoint nodes in FsImage and EditLogs from NameNode and joins them to produce a new image. Whenever you generate FsImage and EditLogs from NameNode and merge them to create a new image, checkpoint nodes in HDFS create a checkpoint and deliver it to the NameNode. The directory structure is always identical to that of the name node, so the checkpointed image is always available.

Backup Node:

Backup nodes are used to provide high availability of the data. In case one of the active NameNode or DataNodes goes down, the backup node can be promoted to active and the active node switched over to the backup node. Backup nodes are not used to recover from a failure of the active NameNode or DataNodes. Instead, you use a replica set of the data for that purpose. Data nodes are used to store the data and to create the FsImage and editsLogs files for replication. Data nodes connect to one or more replica sets of the data to create the FsImage and editsLogs files for replication. Data nodes are not used to provide high availability.

B*locks:*

This default size can be changed to any value between 32 and 128 megabytes, depending on the performance required. Data is written to the DataNodes every time a user makes a change, and new data is appended to the end of the DataNode. DataNodes are replicated to ensure data consistency and fault tolerance. If a Node fails, the system automatically recovers the data from a backup and replicates it across the remaining healthy Nodes. DataNodes do not store the data directly on the hard drives, instead using the HDFS file system. This architecture allows HDFS to scale horizontally as the number of users and data types increase. When the file size gets bigger, the block size gets bigger as well. When the file size becomes bigger than the block size, the larger data is placed in the next block. For example, if the data is 135 MB and the block size is 128 MB, two blocks will be created. The first block will be 128 MB, while the second block will be 135 MB. When the file size gets bigger than that, the larger data will be placed in the next block. This ensures that the most data will always be stored at the same block.

Features of HDFS: The following are the main advantages of HDFS

- HDFS can be configured to create multiple replicas for a particular file. If any one replica fails, the user can still access the data from other replicas. HDFS provides the option to configure automatic failover in case of a failure. So, in case of any hardware failure or an error, the user can get his data from another node where the data has been replicated. HDFS provides the facility to perform software failover. This is similar to automatic failover; however, it is performed at the data provider level. So, in case of any hardware failure or an error, the user can get his data from another node where the data has been replicated.
- Horizontal scalability means that the data stored on multiple nodes can be stored in a single file system. Vertical scalability means that data can be stored on multiple nodes. Data can be replicated to ensure data integrity. Replication occurs through the use of replication factors rather than the data itself. HDFS can store up to 5PB of data in a single cluster and handles the load by automatically choosing the best data node to store data on. Data can be read/updated

quickly as it is stored on multiple nodes. Data stored on multiple nodes through replication increases the reliability of data.

- Data is stored on HDFS, not on the local filesystem of your computer. In the event of a failure, the data is stored on a separate server, and can be accessed by the application running on your local computer. Data is replicated on multiple servers to ensure that even in the event of a server failure, your data is still accessible. Data can be accessed via a client tool such as the Java client, the Python client, or the CLI. Access to data is accessible via a wide variety of client tools. This makes it possible to access data from a wide variety of programming languages.

Replication Management in HDFS Architecture

HDFS is able to survive computer crashes and recover from data corruption. HDFS operates on the principle of duplicates, so in the event of failure, it can continue operating as long as there are replicas available. When working on the principle of replicas, data is duplicated and stored on different machines in the DHFS cluster. A replica of every block is stored on at least three DataNodes. HDFS uses a technique referred to as nameNode maintenance to maintain copies on multiple DataNodes. The nameNode keeps track of how many blocks have been under- or over-replicated, and subsequently adds or deletes copies accordingly.

Write Operation

The process continues until all DataNodes have received the data. After DataNodes receive a copy of the file, they send back the location of the last block they received. This enables the NameNode to reconstruct the file. After receiving the last block, the DataNodes notify the NameNode that the job is complete. The NameNode then replies with a complete file that can be used by the application. When a file is split into segments, it must be reassembled to return the file data to the application. Splitting a file into segments is a method that enables the NameNode to optimize its storage capacity. Splitting a file into segments also improves fault tolerance and availability. When the client receives a split file, the process is similar to that of a single file. The client divides the file into segments, which are then sent to the DataNode. DataNode 1 receives the segment A and passes it to DataNode 2 and so on.

Read Operation

The client then sends the file to the Replicator. The Replicator does not have a copy of the file and must read the data from another location. In the background, data is then sent to the DataNode. The DataNode only has metadata and must contact the other data nodes to receive the actual data. The data is then sent to the Replicator. The Replicator again does not have a copy of the file and must read the data from another location. Data is then sent to the Reducer. The Reducer does have a copy of the data, but a compressed version.

Advantages of HDFS Architecture:

1. It is a highly scalable data storage system. This makes it ideal for data-intensive applications like Hadoop and streaming analytics. Another major benefit of Hadoop is that it is easy to set up. This makes it ideal for non-technical users.

2. It is very easy to implement, yet very robust. There is a lot of flexibility you get with Hadoop. It is a fast and reliable file system.
3. This makes Hadoop a great fit for a wide range of data applications. The most common one is analytics. You can use Hadoop to process large amounts of data quickly, and then analyze it to find trends or make recommendations. The most common type of application that uses Hadoop analytics is data crunching.
4. You can increase the size of the cluster by adding more nodes or increase the size of the cluster by adding more nodes. If you have many clients that need to be stored on HDFS you can easily scale your cluster horizontally by adding more nodes to the cluster. To scale your cluster vertically, you can increase the size of the cluster. Once the size of the cluster is increased, it can serve more clients.
5. This can be done by setting up a centralized database, or by distributing data across a cluster of commodity personal computers, or a combination of both. The most common setup for this type of virtualization is to create a virtual machine on each of your servers.
6. Specialization reduces the overhead of data movement across the cluster and provides high availability of data.
7. Automatic data replication can be accomplished with a variety of technologies, including RAID, Hadoop, and database replication. Logging data and monitoring it for anomalies can also help to detect and respond to hardware and software failures.

Disadvantages of HDFS Architecture:

1. It is important to have a backup strategy in place. The cost of downtime can be extremely high, so it is important to keep things running smoothly. It is also recommended to have a security plan in place. If your company does not have a data backup plan, you are putting your company's data at risk.
2. The chances are that the data in one location is vulnerable to hacking. Imagine the fear of losing valuable data when a disaster strikes. To protect data, backup data to a remote location. In the event of a disaster, the data can be quickly restored to its original location.
3. This can be done manually or through a data migration process. Once the data is copied to the local environment, it can be accessed, analyzed, and used for any purpose.

8. <u>Hadoop Configuration:</u>

In the Hadoop stack, we are having the multiple services in it like HDFS, Yarn, Oozie, MapReduce, Spark, Atlas, Ranger, Zeppelin, Kafka, NiFi, Hive, HBase, etc. Every service is having its own functionality and working methodology. As we have said the working methodology is different than the configuration is also different for the different services. Before doing the Hadoop configuration, we need to take care of the operating system configuration also. In the Hadoop ecosystem, Hadoop configuration will come in the second part. In the primary part, we need to tune up the operating system configuration and make the operating system up to the mark. So, the operating system will able to handle the load of the Hadoop ecosystem.

Syntax of Hadoop Configuration:

As such, there is no specific syntax available for the Hadoop /. Generally, we are using the number of services on it. As per the requirement or need, we will install the Hadoop services and configure the parameters. In the Hadoop stack, most of the time we will do the configuration form the UI level only. But for the troubleshooting or some different configuration, we will also use the CLI.

1. Hadoop Configuration: HDFS

In the Hadoop environment, the Hadoop configuration command is very common. It is using very widely. It will help us to list out the number of files on the HDFS level.

Configuration Properties:

- **client.https.need-auth:** It will help to check whether SSL client certificate authentication is required or not for the client and server communication.
- **client.cached.conn.retry:** From the cache, the value will define the number of times the HDFS client will able to get a socket. If the number of socket try will be exceeded, the HDFS client will try to create new socket.
- **https.server.keystore.resource:** It will be the same resource file from which we will get the SSL server Keystore evidence will be extracted.
- **client.https.keystore.resource:** It will be the same resource file from which we will get the SSL server Keystore evidence will be extracted in terms of HTTPS communication.
- **datanode.https.address:** It will be the configuration parameter of the datanode secure HTTPS server address and port information.
- **namenode.https-address:** It will be the configuration parameter of the namenode secure https server address and port information.
- **qjournal.queued-edits.limit.mb:** In terms of the quorum journal edits, it will help to define the queue size. The value will be defined in MB.
- **qjournal.select-input-streams.timeout.ms:** In terms of the journal managers, it is the timeout value for accepting streams. The value would be in milliseconds.
- **qjournal.start-segment.timeout.mb:** This configuration value will help to define the quorum timeout. The value will be in milliseconds.

2. Hadoop Configuration: Yarn

The yarn is very important in terms of resource allocation to the jobs that are running in the Hadoop ecosystem.

Configuration Properties:

- **resource-types:** It will be the addition of the resources. We need to define by Comma separated value. It will not include the configuration parameters like memory (in Mb or GB) or vcores values.
- **resource-types.<resource>.units:** It will be the default unit for the specified resource type in the yarn config.
- **resource-types.<resource>.minimum-allocation:** We can set the value for the minimum request for the definite resource type.

- **resource-types.<resource>.maximum-allocation:** We can set the value for the maximum request for the definite resource type.
- **app.mapreduce.am.resource.mb:** It will help to configure the memory requested for the application master container. The value will be in MB. The defaults of the configuration are 1536.
- **app.mapreduce.am.resource.memory:** It will help to configure the memory requested for the application master container. The value will be in MB. The defaults of the configuration are 1536.
- **app.mapreduce.am.resource.memory-mb:** It will help to set the memory requested for the application master. The value will be in MB. The defaults of the configuration are 1536.
- **app.mapreduce.am.resource.cpu-vcores:** It will help to configure the CPU requested for the application master container to the value. The value will be in CPU count. The defaults of the configuration are 1.

3. Hadoop Configuration: Oozie

In Hadoop, we are using the Oozie service to schedule the Hadoop level job.

Configuration Properties:

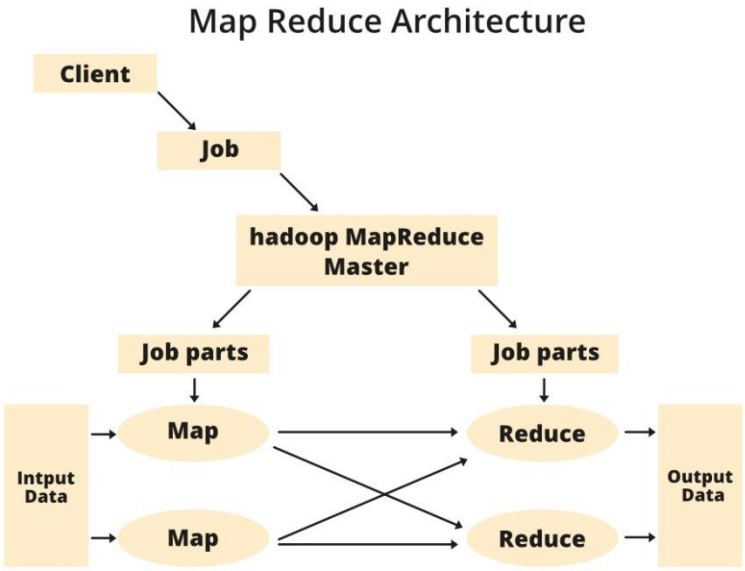
- **CATALINA_OPTS:** It will help to setup the tomcat server. It will help to run the oozie java configuration or properties. The oozie properties will provided in term of the variable. There is no default value for this configuration.
- **OOZIE_CONFIG_FILE:** With the help of this configuration property, we will load the oozie configuration file in the system. The value of this configuration is oozie-site.xml.
- **OOZIE_LOGS:** It will help to store the oozie logs information to the specific directory. While installing the oozie server, it will define the value by its own.
- **OOZIE_LOG4J_FILE:** With the help of this configuration property, we will load the oozie Log4J configuration file in the system. The value of this configuration is oozie-log4j.properties.
- **OOZIE_LOG4J_RELOAD:** It will help to reload the Log4J configuration file on a specific interval of time. The value will be in seconds. The default value is 10.
- **OOZIE_HTTP_PORT:** It will help to define the port for the Oozie server. The default port value is 11000.
- **OOZIE_ADMIN_PORT:** It will help to define the admin port for the Oozie server. The default port value is 11001.
- **OOZIE_HTTP_HOSTNAME:** It will help to define the hostname on which the Oozie server runs. As per the Hadoop architecture, we need to define the specific host or node for it.
- **OOZIE_BASE_URL:** It will help to define the base URL for call-back actions URLs to Oozie server. The default configuration value for the property is http://$ {oozie server host name} : $ {oozie server HTTP port} / oozie.

9. Map Reduce Framework:

MapReduce and HDFS are the two major components of Hadoop which makes it so powerful and efficient to use. MapReduce is a programming model used for efficient processing in parallel over large data-sets in a distributed

manner. The data is first split and then combined to produce the final result. The libraries for MapReduce is written in so many programming languages with various different-different optimizations. The purpose of MapReduce in Hadoop is to Map each of the jobs and then it will reduce it to equivalent tasks for providing less overhead over the cluster network and to reduce the processing power. The MapReduce task is mainly divided into two phases Map Phase and Reduce Phase.

MapReduce Architecture:



Components of MapReduce Architecture:

1. **Client:** The MapReduce client is the one who brings the Job to the MapReduce for processing. There can be multiple clients available that continuously send jobs for processing to the Hadoop MapReduce Manager.
2. **Job:** The MapReduce Job is the actual work that the client wanted to do which is comprised of so many smaller tasks that the client wants to process or execute.
3. **Hadoop MapReduce Master:** It divides the particular job into subsequent job-parts.
4. **Job-Parts:** The task or sub-jobs that are obtained after dividing the main job. The result of all the job-parts combined to produce the final output.
5. **Input Data:** The data set that is fed to the MapReduce for processing.
6. **Output Data:** The final result is obtained after the processing.

In **MapReduce**, we have a client. The client will submit the job of a particular size to the Hadoop MapReduce Master. Now, the MapReduce master will divide this job into further equivalent job-parts. These job-parts are then made available for the Map and Reduce Task. This Map and Reduce task will contain the program as per the requirement of the use-case that the particular company is solving. The developer writes their logic to fulfill the requirement that the industry requires. The input data which we are using is then fed to the Map Task and the Map will generate intermediate key-value pair as its output. The output of Map i.e. these key-value pairs are then fed to

the Reducer and the final output is stored on the HDFS. There can be n number of Map and Reduce tasks made available for processing the data as per the requirement. The algorithm for Map and Reduce is made with a very optimized way such that the time complexity or space complexity is minimum. The MapReduce task is mainly divided into **2 phases** i.e. Map phase and Reduce phase.

1. **Map:** As the name suggests its main use is to map the input data in key-value pairs. The input to the map may be a key-value pair where the key can be the id of some kind of address and value is the actual value that it keeps. The *Map()* function will be executed in its memory repository on each of these input key-value pairs and generates the intermediate key-value pair which works as input for the Reducer or *Reduce()* function.
2. **Reduce:** The intermediate key-value pairs that work as input for Reducer are shuffled and sort and send to the *Reduce()* function. Reducer aggregate or group the data based on its key-value pair as per the reducer algorithm written by the developer.

How Job tracker and the task tracker deal with MapReduce:

1. **Job Tracker:** The work of Job tracker is to manage all the resources and all the jobs across the cluster and also to schedule each map on the Task Tracker running on the same data node since there can be hundreds of data nodes available in the cluster.
2. **Task Tracker:** The Task Tracker can be considered as the actual slaves that are working on the instruction given by the Job Tracker. This Task Tracker is deployed on each of the nodes available in the cluster that executes the Map and Reduce task as instructed by Job Tracker.

   There is also one important component of MapReduce Architecture known as **Job History Server**. The Job History Server is a daemon process that saves and stores historical information about the task or application, like the logs which are generated during or after the job execution are stored on Job History Server.

10. Role of HBase in Big Data processing:

Hadoop uses distributed file system for storing big data, and MapReduce to process it. Hadoop excels in storing and processing of huge data of various formats such as arbitrary, semi-, or even unstructured.

Limitations of Hadoop

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

Hadoop Random Access Databases

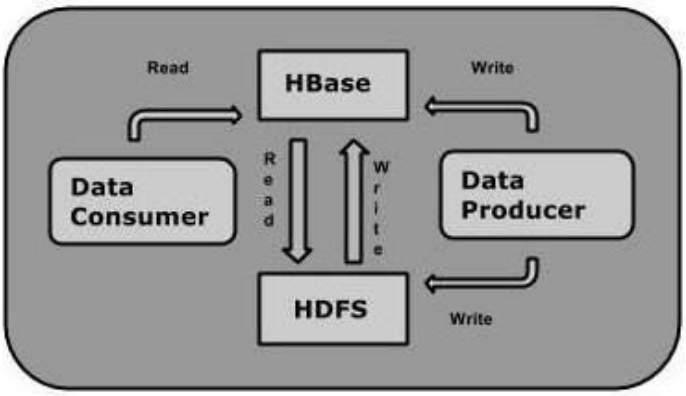
Applications such as HBase, Cassandra, couchDB, Dynamo, and MongoDB are some of the databases that store huge amounts of data and access the data in a random manner.

What is HBase?

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



HBase and HDFS

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

Storage Mechanism in HBase

HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

Given below is an example schema of table in HBase.

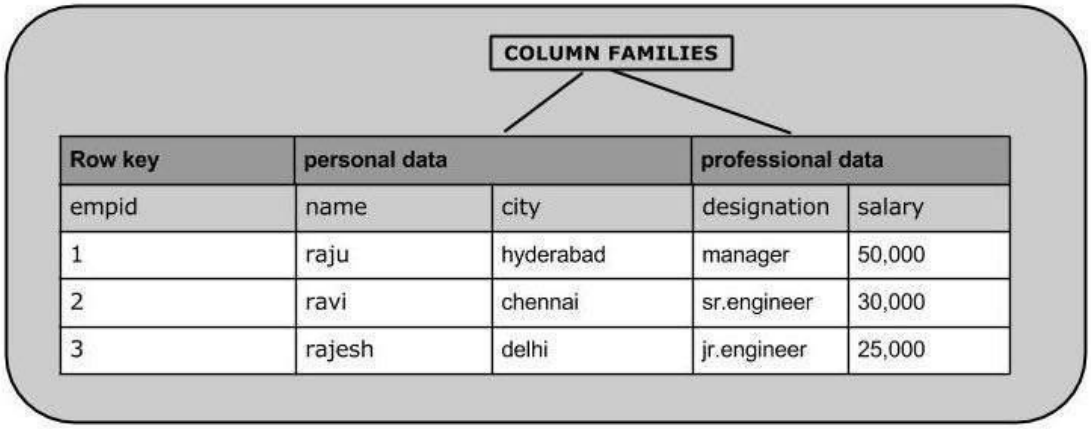
Rowid	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3	col1	col2	col3
1												
2												
3												

Column Oriented and Row Oriented

Column-oriented databases are those that store data tables as sections of columns of data, rather than as rows of data. Shortly, they will have column families.

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

The following image shows column families in a column-oriented database:



HBase and RDBMS

HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.

Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.

- It has easy java API for client.
- It provides data replication across clusters.

Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

Applications of HBase

- It is used whenever there is a need to write heavy applications.
- HBase is used whenever we need to provide fast random access to available data.
- Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

HBase History

Year	Event
Nov 2006	Google released the paper on BigTable.
Feb 2007	Initial HBase prototype was created as a Hadoop contribution.
Oct 2007	The first usable HBase along with Hadoop 0.15.0 was released.
Jan 2008	HBase became the sub project of Hadoop.
Oct 2008	HBase 0.18.1 was released.
Jan 2009	HBase 0.19.0 was released.
Sept 2009	HBase 0.20.0 was released.
May 2010	HBase became Apache top-level project.

11. HIVE:

Apache Hive is a data warehouse and an ETL tool which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop. It is built on top of Hadoop. It is a software project that provides data query and analysis. It facilitates reading, writing and handling wide datasets that stored in distributed storage and queried by Structure Query Language (SQL) syntax. It is not built for Online Transactional Processing (OLTP) workloads. It is frequently used for data warehousing tasks like data encapsulation, Ad-hoc Queries, and analysis of huge datasets. It is designed to enhance scalability, extensibility, performance, fault-tolerance and loose-coupling with its input formats.

Initially Hive is developed by Facebook and Amazon, Netflix and It delivers standard SQL functionality for analytics. Traditional SQL queries are written in the MapReduce Java API to execute SQL Application and SQL queries over distributed data. Hive provides portability as most data warehousing applications functions with SQL-based query languages like NoSQL.

Apache Hive is a data warehouse software project that is built on top of the Hadoop ecosystem. It provides an SQL-like interface to query and analyze large datasets stored in Hadoop's distributed file system (HDFS) or other compatible storage systems.

Hive uses a language called HiveQL, which is similar to SQL, to allow users to express data queries, transformations, and analyses in a familiar syntax. HiveQL statements are compiled into MapReduce jobs, which are then executed on the Hadoop cluster to process the data.

Hive includes many features that make it a useful tool for big data analysis, including support for partitioning, indexing, and user-defined functions (UDFs). It also provides a number of optimization techniques to improve query performance, such as predicate pushdown, column pruning, and query parallelization.

Hive can be used for a variety of data processing tasks, such as data warehousing, ETL (extract, transform, load) pipelines, and ad-hoc data analysis. It is widely used in the big data industry, especially in companies that have adopted the Hadoop ecosystem as their primary data processing platform.

Components of Hive:

- HCatalog – It is a Hive component and is a table as well as a store management layer for Hadoop. It enables user along with various data processing tools like Pig and MapReduce which enables to read and write on the grid easily.
- WebHCat – It provides a service which can be utilized by the user to run Hadoop MapReduce (or YARN), Pig, Hive tasks or function Hive metadata operations with an HTTP interface.

Modes of Hive:

- Local Mode – It is used, when the Hadoop is built under pseudo mode which has only one data node, when the data size is smaller in term of restricted to single local machine, and when processing will be faster on smaller datasets existing in the local machine.
- Map Reduce Mode – It is used, when Hadoop is built with multiple data nodes and data is divided across various nodes, it will function on huge datasets and query is executed parallelly, and to achieve enhanced performance in processing large datasets.

Characteristics of Hive: Databases and tables are built before loading the data.

- Hive as data warehouse is built to manage and query only structured data which is residing under tables.
- At the time of handling structured data, MapReduce lacks optimization and usability function such as UDFs whereas Hive framework have optimization and usability.
- Programming in Hadoop deals directly with the files. So, Hive can partition the data with directory structures to improve performance on certain queries.
- Hive is compatible for the various file formats which are TEXTFILE, SEQUENCEFILE, ORC, RCFILE, etc.
- Hive uses derby database in single user metadata storage and it uses MYSQL for multiple user Metadata or shared Metadata.

Features of Hive:

- ✓ It provides indexes, including bitmap indexes to accelerate the queries. Index type containing compaction and bitmap index as of 0.10.

- ✓ Metadata storage in a RDBMS, reduces the time to function semantic checks during query execution.
- ✓ Built in user-defined functions (UDFs) to manipulation of strings, dates, and other data-mining tools. Hive is reinforced to extend the UDF set to deal with the use-cases not reinforced by predefined functions.
- ✓ DEFLATE, BWT, snappy, etc are the algorithms to operation on compressed data which is stored in Hadoop Ecosystem.
- ✓ It stores schemas in a database and processes the data into the Hadoop File Distributed File System (HDFS).
- ✓ It is built for Online Analytical Processing (OLAP).
- ✓ It delivers various types of querying language which are frequently known as Hive Query Language (HVL or HiveQL).

Advantages:

a. Scalability: Apache Hive is designed to handle large volumes of data, making it a scalable solution for big data processing.
b. Familiar SQL-like interface: Hive uses a SQL-like language called HiveQL, which makes it easy for SQL users to learn and use.
c. Integration with Hadoop ecosystem: Hive integrates well with the Hadoop ecosystem, enabling users to process data using other Hadoop tools like Pig, MapReduce, and Spark.
d. Supports partitioning and bucketing: Hive supports partitioning and bucketing, which can improve query performance by limiting the amount of data scanned.
e. User-defined functions: Hive allows users to define their own functions, which can be used in HiveQL queries.

Disadvantages:

a. Limited real-time processing: Hive is designed for batch processing, which means it may not be the best tool for real-time data processing.
b. Slow performance: Hive can be slower than traditional relational databases because it is built on top of Hadoop, which is optimized for batch processing rather than interactive querying.
c. Steep learning curve: While Hive uses a SQL-like language, it still requires users to have knowledge of Hadoop and distributed computing, which can make it difficult for beginners to use.
d. Lack of support for transactions: Hive does not support transactions, which can make it difficult to maintain data consistency.
e. Limited flexibility: Hive is not as flexible as other data warehousing tools because it is designed to work specifically with Hadoop, which can limit its usability in other environments.

12. PIG:

Pig Represents Big Data as data flows. Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as *Pig Latin* which is used to develop the data analysis codes. First, to process the data

which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language. Internally *Pig Engine*(a component of Apache Pig) converted all these scripts into a specific map and reduce task. But these are not visible to the programmers in order to provide a high-level of abstraction. Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

Note: Pig Engine has two type of execution environment i.e. a *local execution environment* in a single JVM (used when dataset is small in size) and *distributed execution environment* in a Hadoop Cluster.

Need of Pig: One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development using the multi-query approach. Also, Pig is beneficial for programmers who are not from Java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin.

- It uses query approach which results in reducing the length of the code.
- Pig Latin is SQL like language.
- It provides many builtIn operators.
- It provides nested data types (tuples, bags, map).

Evolution of Pig: Earlier in 2006, Apache Pig was developed by Yahoo's researchers. At that time, the main idea to develop Pig was to execute the MapReduce jobs on extremely large datasets. In the year 2007, it moved to Apache Software Foundation (ASF) which makes it an open source project. The first version(*0.1*) of Pig came in the year 2008. The latest version of Apache Pig is *0.18* which came in the year 2017.

Features of Apache Pig:

- For performing several operations Apache Pig provides rich sets of operators like the filtering, joining, sorting, aggregation etc.
- Easy to learn, read and write. Especially for SQL-programmer, Apache Pig is a boon.
- Apache Pig is extensible so that you can make your own process and user-defined functions(UDFs) written in python, java or other programming languages .
- Join operation is easy in Apache Pig.
- Fewer lines of code.
- Apache Pig allows splits in the pipeline.
- By integrating with other components of the Apache Hadoop ecosystem, such as Apache Hive, Apache Spark, and Apache ZooKeeper, Apache Pig enables users to take advantage of these components' capabilities while transforming data.
- The data structure is multivalued, nested, and richer.
- Pig can handle the analysis of both structured and unstructured data.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Difference between Pig and MapReduce

Apache Pig	MapReduce
It is a scripting language.	It is a compiled programming language.
Abstraction is at higher level.	Abstraction is at lower level.
It have less line of code as compared to MapReduce.	Lines of code is more.
Less effort is needed for Apache Pig.	More development efforts are required for MapReduce.
Code efficiency is less as compared to MapReduce.	As compared to Pig efficiency of code is higher.
Pig provides built in functions for ordering, sorting and union.	Hard to perform data operations.
It allows nested data types like map, tuple and bag	It does not allow nested data types

Applications of Apache Pig:

- For exploring large datasets Pig Scripting is used.
- Provides the supports across large data-sets for Ad-hoc queries.
- In the prototyping of large data-sets processing algorithms.
- Required to process the time sensitive data loads.
- For collecting large amounts of datasets in form of search logs and web crawls.
- Used where the analytical insights are needed using the sampling.

Types of Data Models in Apache Pig: It consist of the 4 types of data models as follows:

- **Atom**: It is a atomic data value which is used to store as a string. The main use of this model is that it can be used as a number and as well as a string.
- **Tuple**: It is an ordered set of the fields.
- **Bag**: It is a collection of the tuples.
- **Map**: It is a set of key/value pairs.

--ooOoo--

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

UNIT – 5: Data Analytics with R Machine Learning: Introduction, Supervised Learning, Unsupervised Learning, Collaborative Filtering, Social Media Analytics, Mobile Analytics, Big Data Analytics with BigR.

1. Data Analytics with R Machine Learning:

Introduction: Data analytics and machine learning in big data using R can be a powerful combination for gaining insights and making predictions. R is a popular programming language for statistical computing and data analysis, and it offers several powerful libraries and packages for machine learning. When working with big data, it's important to consider the challenges of scale and efficiency. Here are some key steps to perform data analytics and machine learning in big data using R:

a. ***Data preprocessing:*** Big data often requires preprocessing steps to handle missing values, outliers, and feature engineering. R provides various packages like "dplyr" and "tidyverse" that can assist in data manipulation and transformation.

b. ***Distributed computing:*** R is primarily a single-node language, but there are packages like "SparkR" that enable distributed computing on big data frameworks like Apache Spark. This allows you to leverage the power of distributed computing to handle large datasets.

c. ***Feature selection and dimensionality reduction:*** With big data, you may have a large number of features, and not all of them are relevant. Feature selection techniques such as correlation analysis, backward elimination, or random forests can help identify the most important features. R provides packages like "caret" and "mlr" that offer various feature selection and dimensionality reduction algorithms.

d. ***Model training:*** R offers a wide range of machine learning algorithms, such as decision trees, random forests, support vector machines, and neural networks. Packages like "caret," "randomForest," "e1071," and "keras" provide implementations of these algorithms. When working with big data, it is important to use algorithms that are scalable and can handle large datasets efficiently.

e. ***Model evaluation and tuning:*** After training the models, you need to evaluate their performance and tune the hyperparameters. Techniques like cross-validation and grid search can help in model selection and hyperparameter tuning. R packages like "caret" and "mlr" provide functions for model evaluation and tuning.

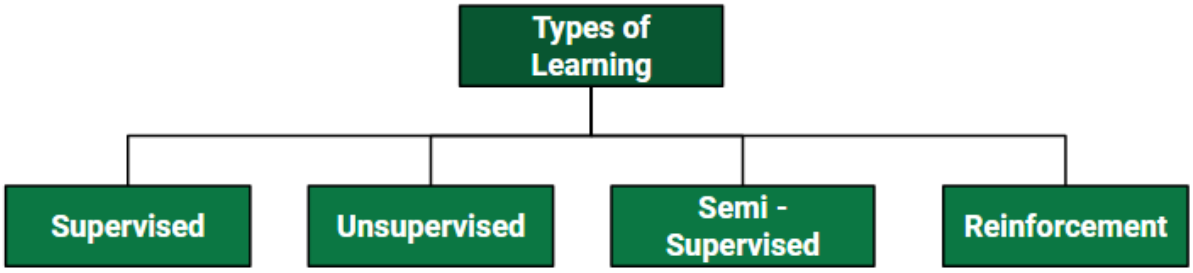
f. ***Deployment and productionization:*** Once you have a trained and optimized model, you can deploy it in production systems. R offers options to deploy models as APIs using packages like "plumber" or integrate them into other software systems.

It's worth noting that while R is a powerful language for data analytics and machine learning, it may not be the most performant option for big data processing. In some cases, using distributed computing frameworks like Apache Spark or switching to languages like Python or Scala might be more suitable for handling large-scale data. R can be a

valuable tool for data analytics and machine learning in big data, but it's essential to consider the specific requirements and challenges of your project and choose the appropriate tools and techniques accordingly.

2. Supervised Learning, Unsupervised Learning:

Machine Learning is the Field of study that gives computers the capability to learn without being explicitly programmed. I.e. Machine Learning (ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance. The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate. There are mainly **four** types of learning.



In this article let's discuss the two most important learning e.g Supervised and Unsupervised Learning in R programming.

R language is basically developed by statisticians to help other statisticians and developers faster and efficiently with the data. As of now, we know that machine learning is basically working with a large amount of data and statistics as a part of data science the use of R language is always recommended. Therefore, the R language is mostly becoming handy for those working with machine learning making tasks easier, faster, and innovative. Here are some top advantages of the R language to implement a machine learning algorithm in R programming.

Advantages to Implement Machine Learning Using R Language

- It provides good explanatory code. For example, if you are at the early stage of working with a machine learning project and you need to explain the work you do, it becomes easy to work with R language comparison to python language as it provides the proper statistical method to work with data with fewer lines of code.
- R language is perfect for data visualization. R language provides the best prototype to work with machine learning models.
- R language has the best tools and library packages to work with machine learning projects. Developers can use these packages to create the best pre-model, model, and post-model of the machine learning projects. Also, the packages for R are more advanced and extensive than python language which makes it the first choice to work with machine learning projects.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Supervised Learning

Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is learning in which we teach or train the machine using data that is well labeled which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data. Supervised learning is classified into two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as "Red" or "blue" or "disease" and "no disease".
- **Regression**: A regression problem is when the output variable is a real value, such as "dollars" or "weight".

Supervised learning deals with or learns with "labeled" data. This implies that some data is already tagged with the correct answer.

Types

- Regression
- Logistic Regression
- Classification
- Naïve Bayes Classifiers
- Decision Trees
- Support Vector Machine

Implementation in R

Let's implement one of the very popular Supervised Learning i.e Simple Linear Regression in R programming. Simple Linear Regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables. One variable denoted x is regarded as an independent variable and the other one denoted y is regarded as a dependent variable. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x). The basic syntax for regression analysis in R is:

→***Syntax:*** **lm(Y ~ model)**

where,

Y *is the object containing the dependent variable to be predicted and model is the formula for the chosen mathematical model.*

The command **lm()** provides the model's coefficients but no further statistical information. The following R code is used to implement **simple linear regression.**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Example:

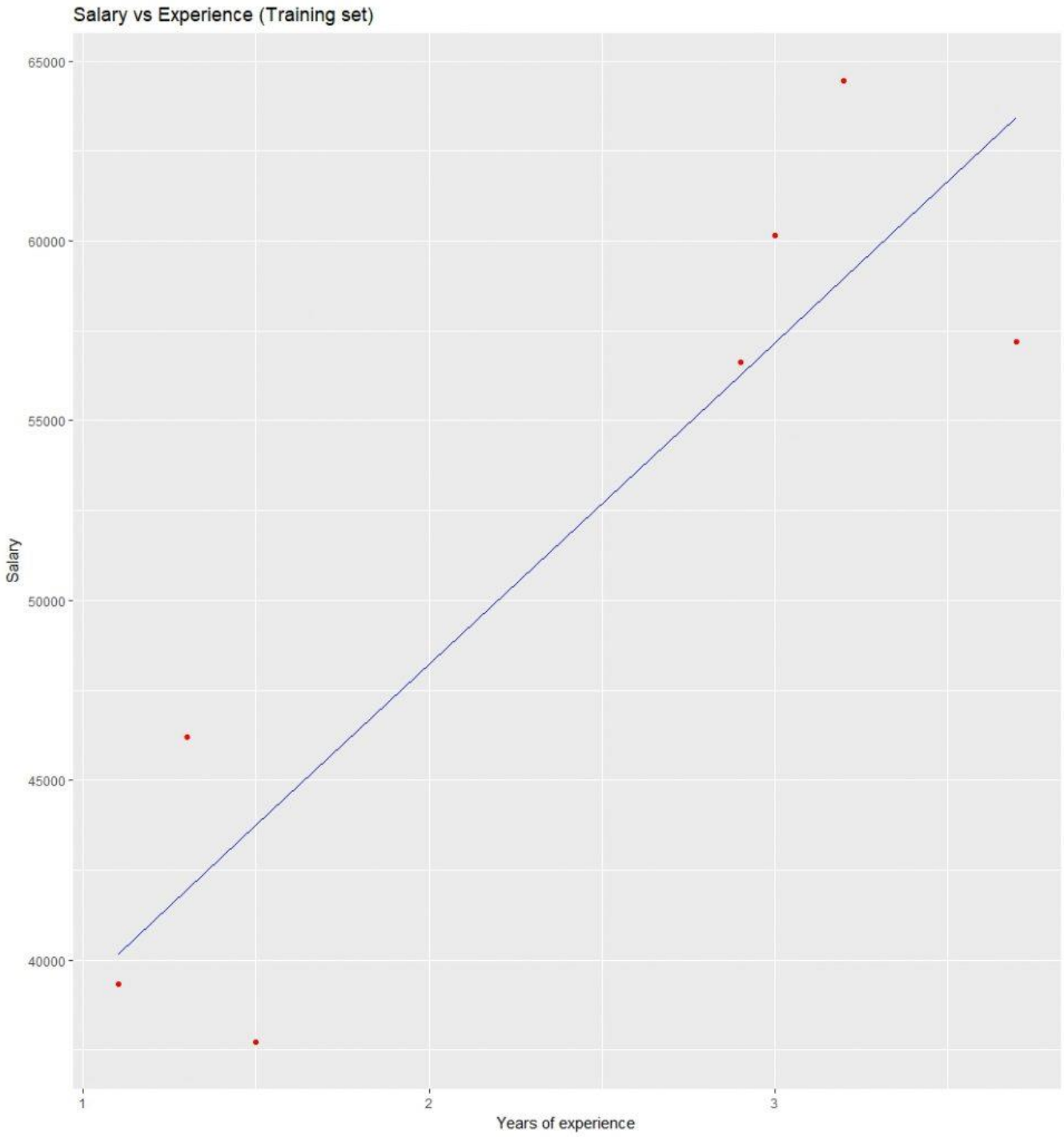
```
# Simple Linear Regression
# Importing the dataset
dataset = read.csv('salary.csv')
# Splitting the dataset into the
# Training set and Test set
install.packages('caTools')
library(caTools)
split = sample.split(dataset$Salary,
                     SplitRatio = 0.7)
trainingset = subset(dataset, split == TRUE)
testset = subset(dataset, split == FALSE)
# Fitting Simple Linear Regression
# to the Training set
lm.r = lm(formula = Salary ~ YearsExperience,
                      data = trainingset)
coef(lm.r)

# Predicting the Test set results
ypred = predict(lm.r, newdata = testset)

install.packages("ggplot2")
library(ggplot2)

# Visualising the Training set results
ggplot() + geom_point(aes(x = trainingset$YearsExperience,
                          y = trainingset$Salary),
                          colour = 'red') +
          geom_line(aes(x = trainingset$YearsExperience,
                        y = predict(lm.r, newdata = trainingset)),
                        colour = 'blue') +

ggtitle('Salary vs Experience (Training set)') +
xlab('Years of experience') +
ylab('Salary')

# Visualising the Test set results
ggplot() + geom_point(aes(x = testset$YearsExperience,
                          y = testset$Salary),
                          colour = 'red') +
          geom_line(aes(x = trainingset$YearsExperience,
                        y = predict(lm.r, newdata = trainingset)),
                        colour = 'blue') +

ggtitle('Salary vs Experience (Test set)') +
xlab('Years of experience') +
ylab('Salary')
```

Output:

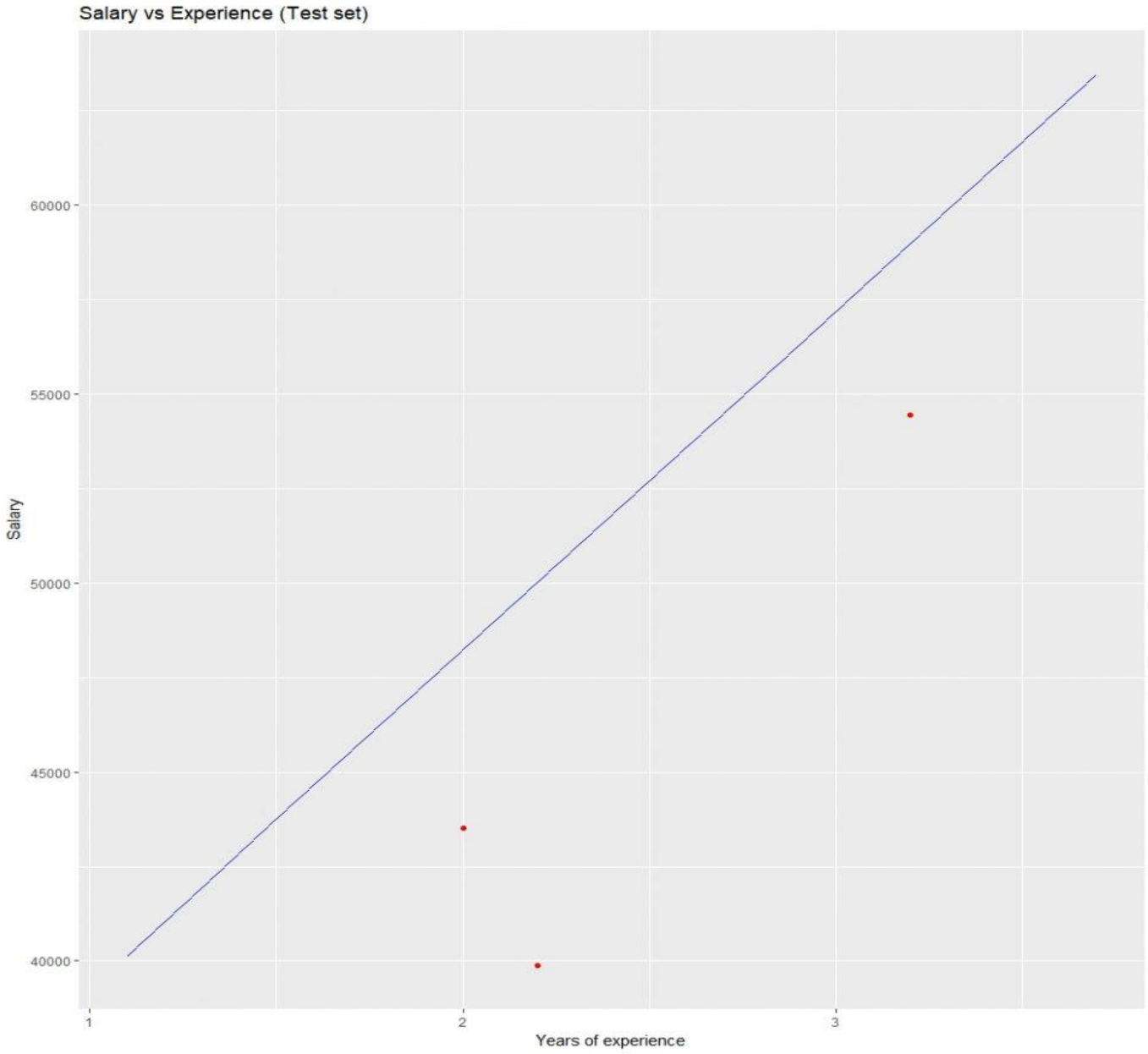
```
Intercept Years Experience
24558.39 10639.23
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Visualizing the Training set results:



Visualizing the Testing set results:



Unsupervised Learning:

R – Unsupervised learning is the training of machines using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to finding the hidden structure in unlabeled data by our-self. Unsupervised learning is classified into two categories of algorithms:

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Types

Clustering:

1. Exclusive (partitioning)
2. Agglomerative
3. Overlapping
4. Probabilistic

Clustering Types:

1. Hierarchical clustering
2. K-means clustering
3. K-NN (k nearest neighbors)
4. Principal Component Analysis
5. Singular Value Decomposition
6. Independent Component Analysis

Implement in k-means clustering in R

Let's implement one of the very popular Unsupervised Learning i.e K-means clustering in R programming. K means clustering in R Programming is an Unsupervised Non-linear algorithm that clusters data based on similarity or similar groups. It seeks to partition the observations into a pre-specified number of clusters. Segmentation of data takes place to assign each training example to a segment called a cluster. In the unsupervised algorithm, high reliance on raw data is given with large expenditure on manual review for review of relevance is given. It is used in a variety of fields like Banking, healthcare, retail, Media, etc.

➔ ***Syntax: kmeans(x, centers = 3, nstart = 10)***

Where:

- *x is numeric data*
- *centers is the pre-defined number of clusters*
- *the k-means algorithm has a random component and can be repeated nstart times to improve the returned model*

Example:

```
# Installing Packages
install.packages("ClusterR")
install.packages("cluster")

# Loading package
library(ClusterR)
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

```
library(cluster)

# Loading data
data(iris)

# Structure
str(iris)

# Removing initial label of
# Species from original dataset
iris_1 <- iris[, -5]

# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.re <- kmeans(iris_1, centers = 3,
                    nstart = 20)
kmeans.re

# Cluster identification for
# each observation
kmeans.re$cluster

# Confusion Matrix
cm <- table(iris$Species, kmeans.re$cluster)
cm

# Model Evaluation and visualization
plot(iris_1[c("Sepal.Length", "Sepal.Width")])
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster)
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster,
     main = "K-means with 3 clusters")

# Plotiing cluster centers
kmeans.re$centers
kmeans.re$centers[, c("Sepal.Length",
                      "Sepal.Width")]

# cex is font size, pch is symbol
points(kmeans.re$centers[, c("Sepal.Length",
                             "Sepal.Width")],
   col = 1:3, pch = 8, cex = 3)

# Visualizing clusters
y_kmeans <- kmeans.re$cluster
clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
         y_kmeans,
         lines = 0,
         shade = TRUE,
```

```
                color = TRUE,
                labels = 2,
                plotchar = FALSE,
                span = TRUE,
                main = paste("Cluster iris"),
                xlab = 'Sepal.Length',
                ylab = 'Sepal.Width')
```

Output:

- **Model kmeans_re:**

```
> kmeans.re
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [44] 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2
 [87] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3
[130] 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

The 3 clusters are made which are of 50, 62, and 38 sizes respectively. Within the cluster, the sum of squares is 88.4%.

- **Cluster identification:**

```
> confusionMatrix(cm)
Confusion Matrix and Statistics


            setosa versicolor virginica
  setosa        20          0         0
  versicolor     0         20         0
  virginica      0          0        20

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9404, 1)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           1.0000
Specificity                 1.0000            1.0000           1.0000
Pos Pred Value              1.0000            1.0000           1.0000
Neg Pred Value              1.0000            1.0000           1.0000
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3333
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            1.0000           1.0000
```

The model achieved an accuracy of 100% with a p-value of less than 1. This indicates the model is good.

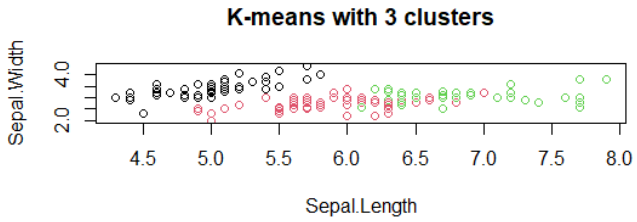
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

- **Confusion Matrix:**

```
> cm

           1  2  3
setosa    50  0  0
versicolor 0 48  2
virginica  0 14 36
```

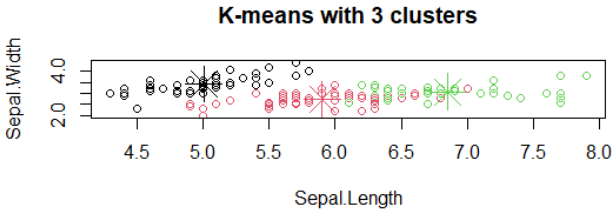
So, 50 Setosa are correctly classified as Setosa. Out of 62 Versicolor, 48 Versicolor are correctly classified as Versicolor, and 14 are classified as virginica. Out of 36 virginica, 19 virginica are correctly classified as virginica and 2 are classified as Versicolor.

- **K-means with 3 clusters plot:**



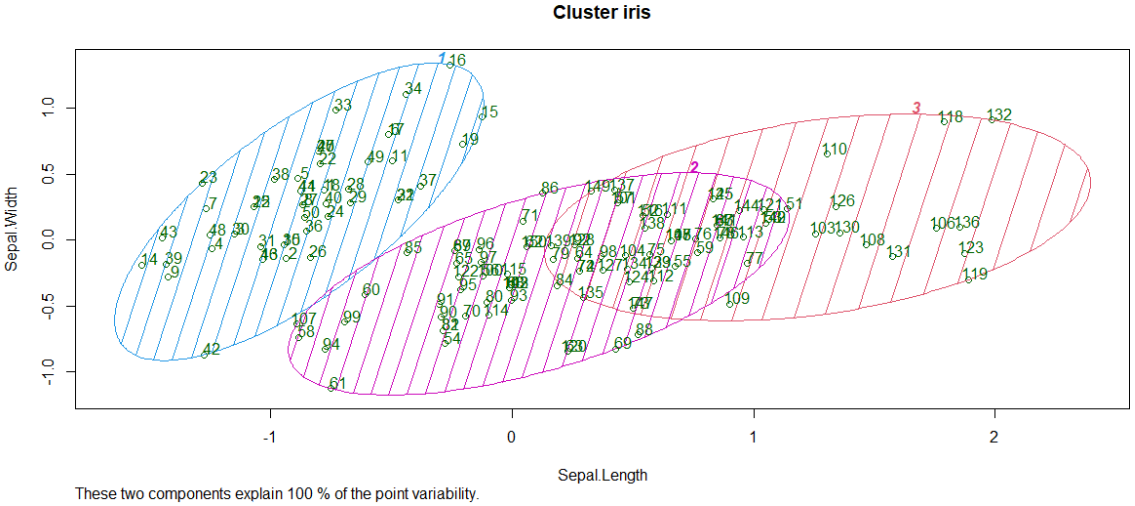
The model showed 3 cluster plots with three different colors and with Sepal.length and with Sepal.width.

- **Plotting cluster centers:**



In the plot, centers of clusters are marked with cross signs with the same color of the cluster.

- **Plot of clusters:**



So, 3 clusters are formed with varying sepal length and sepal width.

3. Collaborative Filtering:

There are a lot of applications where websites collect data from their users and use that data to predict the likes and dislikes of their users. This allows them to recommend the content that they like. Recommender systems are a way of suggesting similar items and ideas to a user's specific way of thinking.

There are basically two types of recommender Systems:

- **Collaborative Filtering:** Collaborative Filtering recommends items based on similarity measures between users and/or items. The basic assumption behind the algorithm is that users with similar interests have common preferences.
- **Content-Based Recommendation:** It is supervised machine learning used to induce a classifier to discriminate between interesting and uninteresting items for the user.

In this article, we will mainly focus on the Collaborative Filtering method.

What is Collaborative Filtering?

In Collaborative Filtering, we tend to find similar users and recommend what similar users like. In this type of recommendation system, we don't use the features of the item to recommend it, rather we classify the users into clusters of similar types and recommend each user according to the preference of its cluster.

There are basically four types of algorithms o say techniques to build Collaborative filtering based recommender systems:

- Memory-Based
- Model-Based
- Hybrid
- Deep Learning

Advantages of Collaborative Filtering-Based Recommender Systems

As we know there are two types of recommender systems the content-based recommender systems have limited use cases and have higher time complexity. Also, this algorithm is based on some limited content but that is not the case in Collaborative Filtering based algorithms. One of the main advantages that these recommender systems have is that they are highly efficient in providing personalized content but also able t adapt to changing user preferences.

Measuring Similarity

A simple example of the movie recommendation system will help us in explaining:

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

In this type of scenario, we can see that User 1 and User 2 give nearly similar ratings to the movie, so we can conclude that Movie 3 is also going to be averagely liked by User 1 but Movie 4 will be a good recommendation to User 2, like this we can also see that there are users who have different choices like User 1 and User 3 are opposite to each other. One can see that User 3 and User 4 have a common interest in the movie, on that basis we can say that Movie 4 is also going to be disliked by User 4. This is Collaborative Filtering, we recommend to users the items which are liked by users of similar interest domains.

Cosine Similarity

We can also use the cosine similarity between the users to find out the users with similar interests, larger cosine implies that there is a smaller angle between two users, hence they have similar interests. We can apply the cosine distance between two users in the utility matrix, and we can also give the zero value to all the unfilled columns to make calculation easy, if we get smaller cosine then there will be a larger distance between the users, and if the cosine is larger than we have a small angle between the users, and we can recommend them similar things.

*** QuickLaTeX cannot compile formula:
\text{similarity} = \frac{A\cdot B}{\left\| A\right\|\times \left\| B\right\|}=\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}}

*** Error message:
File ended while scanning use of \frac .
Emergency stop.

Rounding the Data

In collaborative filtering, we round off the data to compare it more easily like we can assign below 3 ratings as 0 and above of it as 1, this will help us to compare data more easily, for example:

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

We again took the previous example and we apply the rounding-off process, as you can see how much more readable the data has become after performing this process, we can see that User 1 and User 2 are more similar and User 3 and User 4 are more alike.

Normalizing Rating

In the process of normalizing, we take the average rating of a user and subtract all the given ratings from it, so we'll get either positive or negative values as a rating, which can simply classify further into similar groups. By

normalizing the data we can make clusters of the users that give a similar rating to similar items and then we can use these clusters to recommend items to the users.

What are some of the Challenges to be Faced while using Collaborative Filtering?

As we know that every algorithm has its pros and cons and so is the case with Collaborative Filtering Algorithms. Collaborative Filtering algorithms are very dynamic and can change as well as adapt to the changes in user preferences with time. But one of the main issues which are faced by recommender systems is that of **scalability** because as the user base increases then the respective sizes for the computation and the data storage space all increase manifold which leads to slow and inaccurate results. Also, collaborative filtering algorithms fail to recommend a diversity of products as it is based on historical data and hence provide recommendations related to them as well.

4. Social Media Analytics:

Social media analytics in big data involves extracting, analyzing, and deriving insights from large volumes of social media data. Social media platforms generate enormous amounts of data, including posts, comments, likes, shares, and user profiles, which can provide valuable information for various purposes such as market research, sentiment analysis, brand monitoring, and customer behavior analysis. Here's an overview of how social media analytics can be performed in the context of big data:

- **Data collection:** To analyze social media data at a large scale, you need to collect and aggregate the data from multiple sources. Social media platforms often provide APIs (Application Programming Interfaces) that allow developers to access their data. R offers packages like "rtweet" and "twitteR" for collecting data from Twitter, while other packages like "rFacebook" and "Rlinkedin" can be used for Facebook and LinkedIn data respectively. Additionally, there are packages like "streamR" that enable real-time data streaming and capture.
- **Data preprocessing:** Once the data is collected, preprocessing is necessary to clean and prepare it for analysis. This step may involve removing duplicates, handling missing values, normalizing text (e.g., removing stop words, stemming, or lemmatization), and identifying entities such as hashtags or mentions. R provides packages like "tm" and "tidytext" that offer functionalities for text preprocessing.
- **Sentiment analysis:** Sentiment analysis is a common technique in social media analytics that involves determining the sentiment or opinion expressed in social media posts. R offers packages like "tidytext" and "sentimentr" that provide pre-trained sentiment lexicons and sentiment analysis functions to classify text sentiment as positive, negative, or neutral.
- **Network analysis:** Social media platforms often exhibit complex network structures, where users are connected through relationships such as followers, friends, or retweets. Network analysis can uncover influential users, community structures, and information flow patterns. R offers packages like "igraph" and "networkD3" for network analysis and visualization.

- **Topic modeling:** Topic modeling algorithms like Latent Dirichlet Allocation (LDA) can be used to discover latent topics within a collection of social media posts. R provides packages like "topicmodels" and "ldatuning" that implement these algorithms and enable topic modeling.
- **Visualization:** Visualizations are crucial for conveying insights from social media data effectively. R offers packages like "ggplot2," "plotly," and "wordcloud" for creating visualizations such as bar charts, line plots, word clouds, and interactive dashboards.
- **Real-time analytics:** Social media data is often generated in real-time, requiring real-time analytics to monitor trends, detect anomalies, or respond to events. R packages like "shiny" and "flexdashboard" enable building interactive dashboards that can be updated in real-time.

It's important to note that social media analytics in big data often involves handling large volumes of data, which may require distributed computing frameworks like Apache Spark or cloud-based solutions for efficient processing and storage. Additionally, the choice of specific techniques and tools may vary depending on the social media platform(s) you are analyzing and the specific objectives of your analysis. R provides a wide range of packages and tools to perform social media analytics in the context of big data, allowing you to extract valuable insights from social media data and gain a deeper understanding of user behavior, sentiment, and trends.

5. Mobile Analytics:

Mobile app analytics is a way to gather data from app users to show user behavior and track app performance. If you have questions about your users, you need to dive into analytics. Mobile app analytics platforms are tools used to track, observe, and support the evaluation of user behavior and experience.

Here are some common situations for product managers where mobile app analytics can help uncover solutions:

- Which screens are most engaged with and where do users spend most of their time?
- Are my users flowing through my app as intended, or are there areas of friction?
- Which screens in my app have the highest crash rate?
- What are the most common interactions (gestures) across my app?
- Which features seem confusing to users?

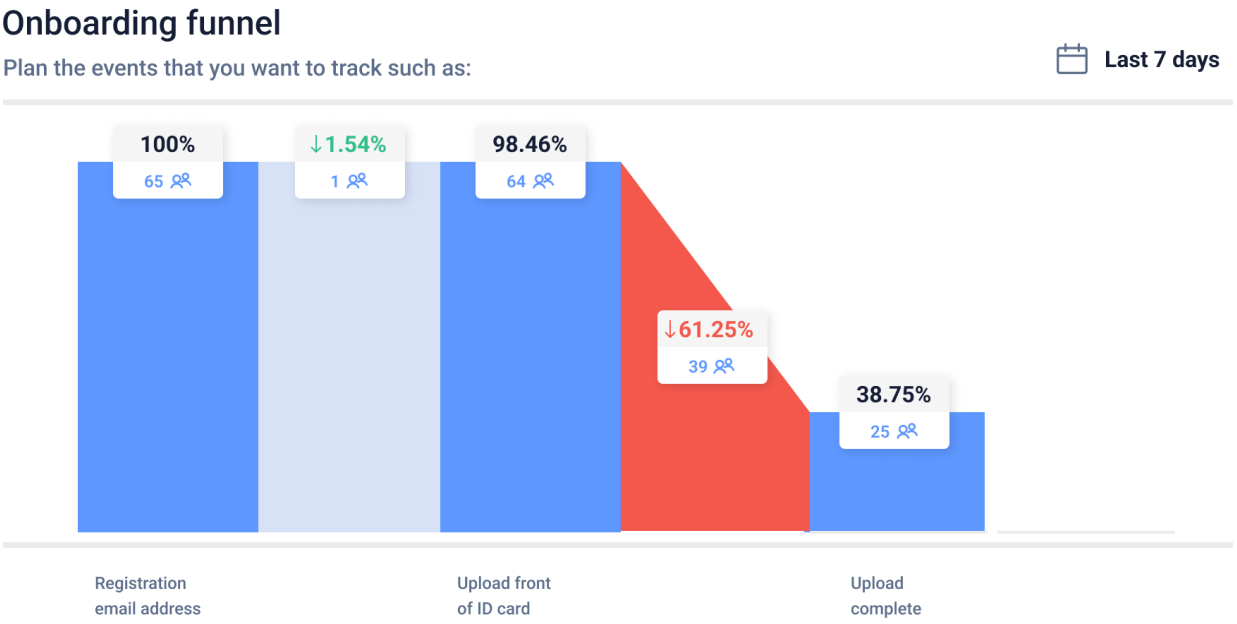
These are just a few examples of the sort of data mobile teams gain access to with mobile app analytics. With this much detail on how users interact with your app, you can consistently improve the user experience and improve retention, engagement, and conversion rates.

What are the benefits of using mobile app analytics?

This all might sound familiar to you if you have a user researcher on your team. Mobile app analytics platforms are not here to replace your in-depth user interviews or observations, but rather to enhance them.

All product teams share similar struggles of collecting, prioritizing, and validating product feedback, and doing it all on a budget. User research is time-consuming, expensive, and sometimes untrustworthy. This is because users will often

subconsciously change behaviors while being observed, and give unreliable information as they can't remember exactly the steps they took. Using tools like session replay eliminates this gap as it offers firsthand user experience data. Using a mobile app analytics platform will scale your user research efforts, not hinder them.



Having as many data points as possible also helps when trying to justify a hypothesis. You will find it much easier to convince a stakeholder that your potential solution to a problem is a bet worth making when you have evidence to back it up. It's also easier to make changes like design decisions when you can clearly see issues with the app that are driving users away.

Why is mobile analytics important?

On average, after downloading your app, only 13% of users will continue to engage with your app regularly or at all. Tracking app analytics is critical to figure out how users are interacting with your app, where they're likely to drop off, and areas that are frustrating or confusing to users to improve your user engagement rates.

Making informed, data-driven decisions will lead you toward a more user-centered approach to product development. Let's look at an example.

Costa Coffee had a goal of optimizing their app registration process to increase its user base and revenue through the app. Using UXCam, their analytics manager tracked custom events related to the registration metrics, then built a funnel. With this visualization, the team could see the way users were typically moving through the sign-up process.

This combination of custom events and funnels uncovered their biggest bottleneck: 15% of users dropped off after entering an invalid password. After looking into the specific session recordings for these users and noticing patterns, the team redesigned the sign-up flow, fixing the issue with incorrect password attempts.

If the mobile team at Costa Coffee didn't have those granular insights it would have been a lot more time-consuming to figure out the issue and potential solutions than by using quantitative methods.

Some more advantages of using an app analytics platform:

- **Gain data-driven insights.** Apps with high retention rates usually experience better user engagement, which means more customers and revenue. Using an analytics tool to get this data frees up time for mobile teams to focus on delivering user-focused product updates.
- **Collect accurate data.** Without performing or checking user feedback or interviews, app analytics platforms uncover the why behind user behavior and experience with detailed, reliable data.
- **Helps customize apps.** Having this data means being able to easier identify issues more through features like session replay and heatmaps to fix obstacles in the user journey. This way product managers have the ability to customize the app and become closer to the goal of providing a seamless experience to end-users.
- **Aligns cross-functional teams.** Being able to share session recordings, customizable dashboards, and crash reports means separate teams can quickly gain clarity on issues. For example, with UXCam customer support teams can speak to a customer about an issue, review their session and then share that session with the PM or engineering team to solve.
- **Measure and track KPIs.** Dashboards make it possible to digest insights and confidently act on them while keeping track of overall app performance. A mobile app KPI dashboard is a convenient way to get an overview of key mobile analytics at a glance. For specific examples of buildable dashboards to track for product managers, engineers, and designers read our in-depth KPI article.

What is the difference between web and mobile analytics?

There are many different ways to track product analytics and it can be overwhelming to decide which approach would best suit your product and your team.

- **Web analytics** is used to focus on metrics like page views, behavior-related metrics, patterns generated by user cohorts, interaction frequency, and user interactions with ads.
- **Mobile app analytics** generate a higher amount of data and supports product teams to understand user behavior. UXCam auto-captures user data, device data, and behavioral data, as well as issue and performance data.
- **Hybrid analytics** allows product teams to, in theory, track data for both web products and mobile applications. The downside to this is that the data coming from hybrid platforms is not as granular as using a dedicated mobile app analytics platform.

It's especially important to track as much data as possible for mobile apps as they're a little more complex than web pages. Mobile apps have multiple versions live across different devices for different app versions, whereas websites have one version. Because of this, PMs are often left with having to understand users across multiple platforms, versions, and device types. This means that specific, in-depth tooling solutions are required.

The question is: how do you quantify and understand user engagement? The days of simple demographic segmentation are over. Users in completely different age groups can have very similar behavior when it comes to mobile apps. For this, you'll need to invest in mobile app analytics that dives into the KPIs and metrics that matter most for your business.

Mobile app analytics techniques

Mobile app analytics, or experience analytics, goes beyond the traditional analytics techniques combining both quantitative and qualitative data methods to provide a holistic picture of user behavior.

While both of those techniques provide powerful insights on their own, they're best utilized in combination with each other.

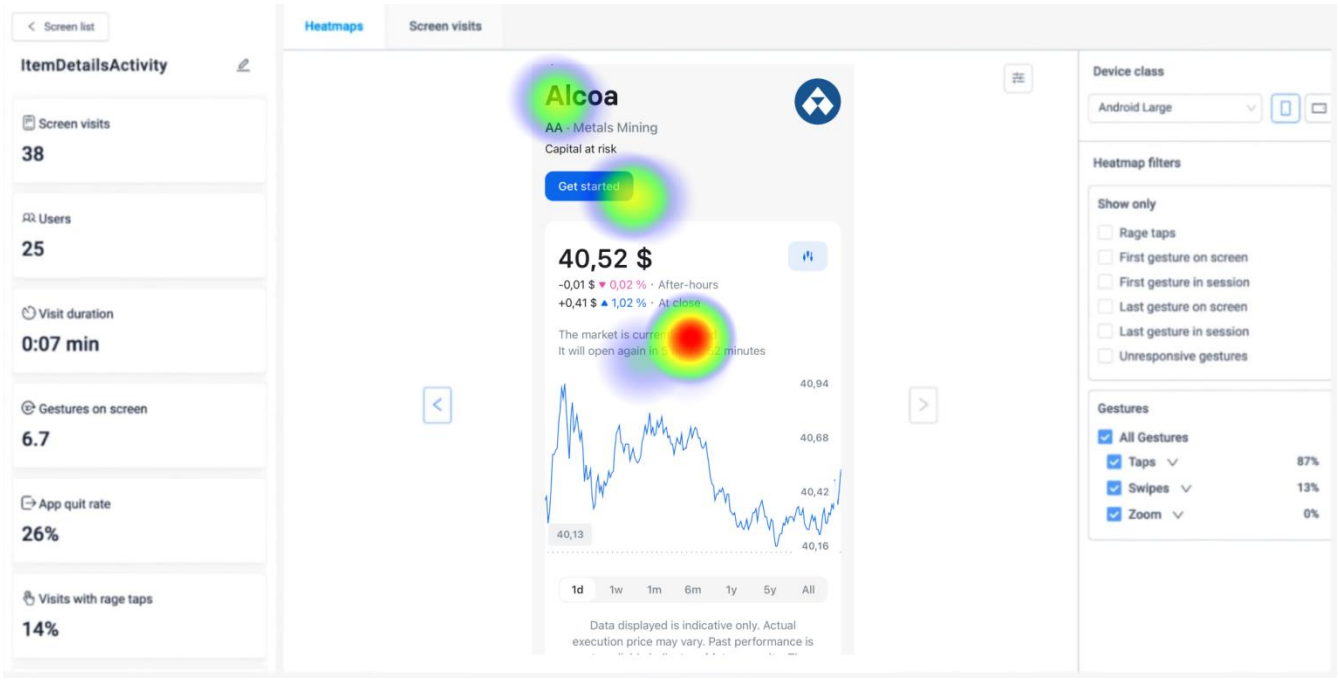
Qualitative methods

Qualitative analysis focuses on subjective information like user behavior, emotions, and experiences. It focuses on the 'why' behind user behavior. Usability testing and user feedback are some of the better-known qualitative analysis methods. While these techniques can be helpful, they don't tell the whole story of what users do on an app — but rather what they *think* they do.

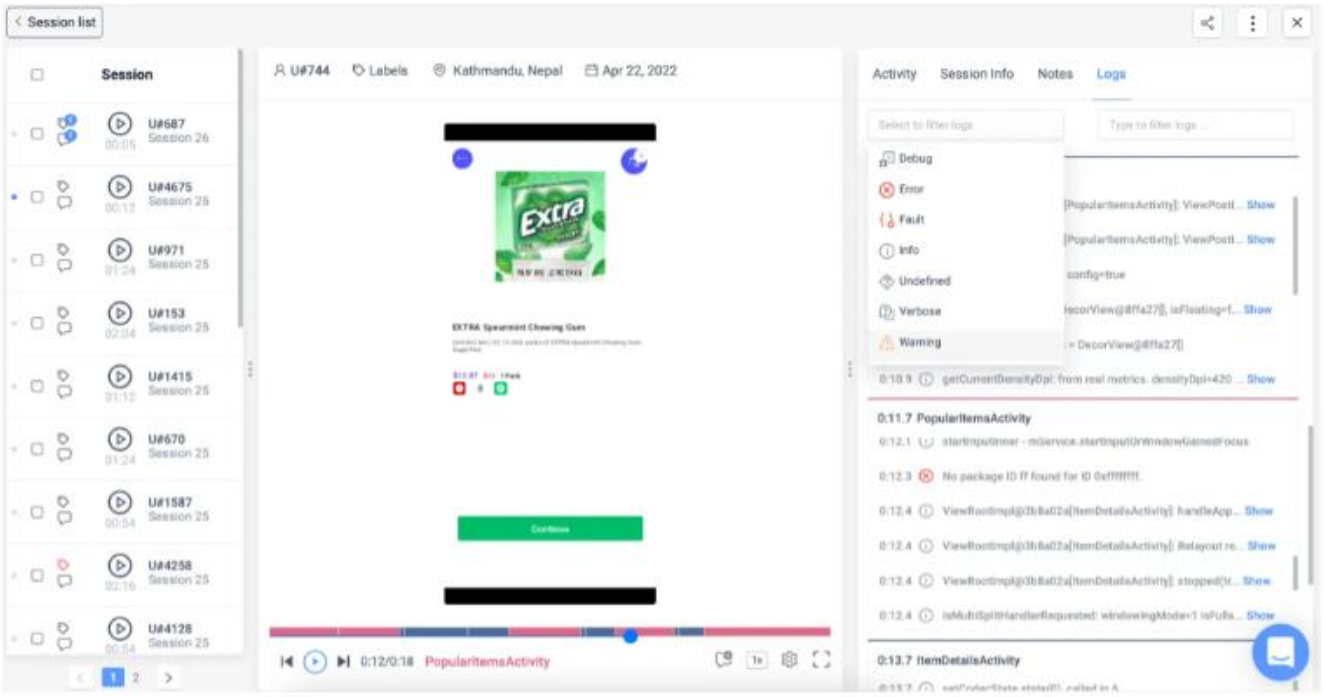
Opinions are powerful and this feedback is invaluable, but observing how real users truly behave is a different story. When paired with data visualization, qualitative analysis techniques can be easily interpreted.

These are the main methods used in qualitative analysis:

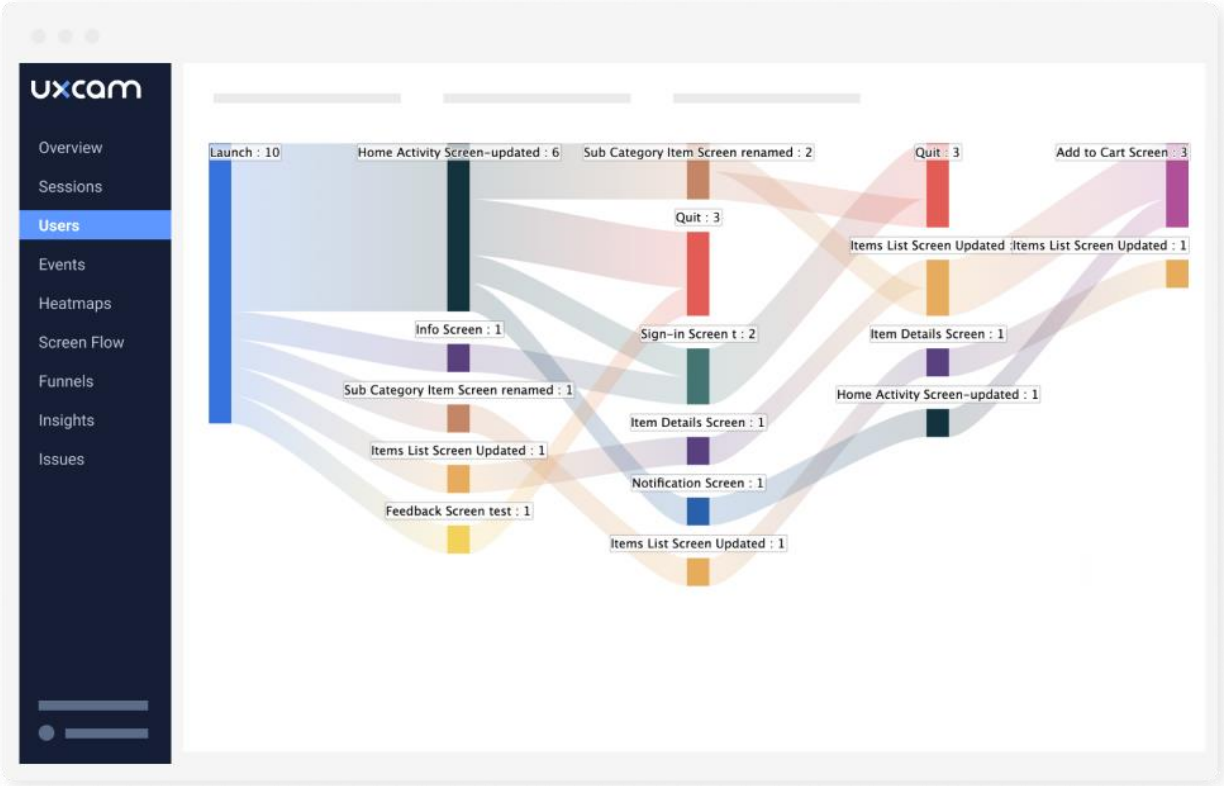
- **Gesture-based heatmaps.** Heatmaps visualize how users engage with your app. Tracking gestures (like double taps, long presses, and frustration signals like rage taps, etc.) is invaluable to understanding customer needs and avoiding churn.



- **Session recordings.** Also known as session replay, this method allows you to view the user's experience and journey from the moment they open the app to when they send it into the background or close it.



- **Behavioral screen flow.** Screen flows validate funnels, prove your hypotheses, or show what unexpected paths users take in your app.



Quantitative methods

Quantitative analysis describes objective information that can be expressed with numbers, graphs, charts, etc. It focuses on the 'what'.

There are three main quantitative analytics methods:

- **User analytics.** This combines lots of technical information like user device data with demographic data (like age or gender).
- **Funnel analysis**. Funnels break up user journeys into key milestones. Most funnels will only work in tandem with custom event analytics. UXCam allows users to view sessions based on the steps in the funnel.
- **Session exploration.** This technique combines many quantitative data points to gain a complete picture of the sessions taking place on your mobile application. By analyzing these metrics over time, you can see trends and draw conclusions about specific user sets.

Technical analysis

Technical analytics comprise crash analytics, exceptions, and UI freeze analytics. Here's a short overview:

- **Crash analytics.** This method helps you stay on top of technical problems that can damage the customer experience. It can also help decipher drops in a defined funnel. With UXCam engineers can see crash rates, number of users affected, reasons the crash occured, and comparisons across different versions, devices, and platforms.

- **UI freeze analytics.** Also known as issue analytics, digs into the issues that cause the crashes in the first place. With UXCam, UI freeze analytics measures the same things that crash analytics does, plus screens the freeze happened on, and how long it lasted.

Privacy and mobile app analytics

For many product and mobile teams it's valuable and insightful to gain all of this data about user behavior and app performance. But, is it safe?

Many apps need to collect personal information about their users to allow them access to the app. Think about financial apps that need users to go through a KYC process and enter data points like date of birth, addresses, and copies of passports or ID cards. When looking into how your user behaves on your app you'll need to take a look into session recordings, even for the screens collecting the most sensitive information.

When sending sessions to a mobile app analytics platform, companies need to know that their users' data is safe. With UXCam you'll be able to mask texts, elements or screens from being recorded, as well as being able to hide all text fields to block out the input, and occlude specific screens.

If you're using customizable dashboards, you can choose to blur all screens from a specific set straight from your dashboard. You can also set rules for which screens you'd like blurred or occluded, blur only specific elements of a screen, or blur entire screens.

Measuring and tracking mobile app analytics is valuable for many different teams. Having all your key app metrics in one place will make it easier to check the health of your app and business

6. Big Data Analytics with BigR:

BigR is an R package that facilitates big data analytics by providing an interface to distributed computing frameworks like Apache Hadoop and Apache Spark. It allows you to work with large datasets that cannot fit into the memory of a single machine by leveraging the power of distributed computing. The general concept behind **R** is to serve as an interface to other software developed in compiled languages such as C, C++, and Fortran and to give the user an interactive tool to analyze data.

Navigate to the folder of the book zip file **bda/part2/R_introduction** and open the **R_introduction.Rproj** file. This will open an RStudio session. Then open the 01_vectors.R file. Run the script line by line and follow the comments in the code. Another useful option in order to learn is to just type the code, this will help you get used to R syntax. In R comments are written with the # symbol. In order to display the results of running R code in the book, after code is evaluated, the results R returns are commented. This way, you can copy paste the code in the book and try directly sections of it in R.

```
# Create a vector of numbers
numbers = c(1, 2, 3, 4, 5)
print(numbers)
# [1] 1 2 3 4 5
# Create a vector of letters
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

```
ltrs = c('a', 'b', 'c', 'd', 'e')
# [1] "a" "b" "c" "d" "e"
# Concatenate both
mixed_vec = c(numbers, ltrs)
print(mixed_vec)
# [1] "1" "2" "3" "4" "5" "a" "b" "c" "d" "e"
```

Let's analyze what happened in the previous code. We can see it is possible to create vectors with numbers and with letters. We did not need to tell R what type of data type we wanted beforehand. Finally, we were able to create a vector with both numbers and letters. The vector mixed_vec has coerced the numbers to character, we can see this by visualizing how the values are printed inside quotes.

The following code shows the data type of different vectors as returned by the function class. It is common to use the class function to "interrogate" an object, asking him what his class is.

```
### Evaluate the data types using class
### One dimensional objects
# Integer vector
num = 1:10
class(num)
# [1] "integer"

# Numeric vector, it has a float, 10.5
num = c(1:10, 10.5)
class(num)
# [1] "numeric"

# Character vector
ltrs = letters[1:10]
class(ltrs)
# [1] "character"

# Factor vector
fac = as.factor(ltrs)
class(fac)
# [1] "factor"
```

R supports two-dimensional objects also. In the following code, there are examples of the two most popular data structures used in R: the matrix and data.frame.

```
# Matrix
M = matrix(1:12, ncol = 4)
#      [,1] [,2] [,3] [,4]
# [1,]    1    4    7   10
# [2,]    2    5    8   11
# [3,]    3    6    9   12
lM = matrix(letters[1:12], ncol = 4)
#     [,1] [,2] [,3] [,4]
# [1,] "a"  "d"  "g"  "j"
# [2,] "b"  "e"  "h"  "k"
# [3,] "c"  "f"  "i"  "l"
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

```
# Coerces the numbers to character
# cbind concatenates two matrices (or vectors) in one matrix
cbind(M, lM)
#     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
# [1,] "1"  "4"  "7"  "10" "a"  "d"  "g"  "j"
# [2,] "2"  "5"  "8"  "11" "b"  "e"  "h"  "k"
# [3,] "3"  "6"  "9"  "12" "c"  "f"  "i"  "l"

class(M)
# [1] "matrix"
class(lM)
# [1] "matrix"

# data.frame
# One of the main objects of R, handles different data types in the same object.
# It is possible to have numeric, character and factor vectors in the same
data.frame

df = data.frame(n = 1:5, l = letters[1:5])
df
#   n l
# 1 1 a
# 2 2 b
# 3 3 c
# 4 4 d
# 5 5 e
```

As demonstrated in the previous example, it is possible to use different data types in the same object. In general, this is how data is presented in databases, APIs part of the data is text or character vectors and other numeric. In is the analyst job to determine which statistical data type to assign and then use the correct R data type for it. In statistics we normally consider variables are of the following types −

- Numeric
- Nominal or categorical
- Ordinal

In R, a vector can be of the following classes −

- Numeric - Integer
- Factor
- Ordered Factor

R provides a data type for each statistical type of variable. The ordered factor is however rarely used, but can be created by the function factor, or ordered.

The following section treats the concept of indexing. This is a quite common operation, and deals with the problem of selecting sections of an object and making transformations to them.

```
# Let's create a data.frame
df = data.frame(numbers = 1:26, letters)
head(df)
```

```
#     numbers  letters
# 1       1       a
# 2       2       b
# 3       3       c
# 4       4       d
# 5       5       e
# 6       6       f

# str gives the structure of a data.frame, it's a good summary to inspect an object
str(df)
#   'data.frame': 26 obs. of  2 variables:
#   $ numbers: int  1 2 3 4 5 6 7 8 9 10 ...
#   $ letters: Factor w/ 26 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9 10 ...

# The latter shows the letters character vector was coerced as a factor.
# This can be explained by the stringsAsFactors = TRUE argumnet in data.frame
# read ?data.frame for more information

class(df)
# [1] "data.frame"

### Indexing
# Get the first row
df[1, ]
#     numbers  letters
# 1       1       a

# Used for programming normally - returns the output as a list
df[1, , drop = TRUE]
# $numbers
# [1] 1
#
# $letters
# [1] a
# Levels: a b c d e f g h i j k l m n o p q r s t u v w x y z

# Get several rows of the data.frame
df[5:7, ]
#     numbers  letters
# 5       5       e
# 6       6       f
# 7       7       g

### Add one column that mixes the numeric column with the factor column
df$mixed = paste(df$numbers, df$letters, sep = '')

str(df)
# 'data.frame': 26 obs. of  3 variables:
# $ numbers: int  1 2 3 4 5 6 7 8 9 10 ...
# $ letters: Factor w/ 26 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9 10 ...
# $ mixed  : chr  "1a" "2b" "3c" "4d" ...
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

```
### Get columns
# Get the first column
df[, 1]
# It returns a one dimensional vector with that column

# Get two columns
df2 = df[, 1:2]
head(df2)

#     numbers  letters
# 1       1       a
# 2       2       b
# 3       3       c
# 4       4       d
# 5       5       e
# 6       6       f

# Get the first and third columns
df3 = df[, c(1, 3)]
df3[1:3, ]

#     numbers  mixed
# 1       1     1a
# 2       2     2b
# 3       3     3c

### Index columns from their names
names(df)
# [1] "numbers" "letters" "mixed"
# This is the best practice in programming, as many times indeces change, but variable names don't
# We create a variable with the names we want to subset
keep_vars = c("numbers", "mixed")
df4 = df[, keep_vars]

head(df4)
#     numbers  mixed
# 1       1     1a
# 2       2     2b
# 3       3     3c
# 4       4     4d
# 5       5     5e
# 6       6     6f

### subset rows and columns
# Keep the first five rows
df5 = df[1:5, keep_vars]
df5

#     numbers  mixed
# 1       1     1a
```

```
# 2      2    2b
# 3      3    3c
# 4      4    4d
# 5      5    5e

# subset rows using a logical condition
df6 = df[df$numbers < 10, keep_vars]
df6

#      numbers  mixed
# 1      1    1a
# 2      2    2b
# 3      3    3c
# 4      4    4d
# 5      5    5e
# 6      6    6f
# 7      7    7g
# 8      8    8h
# 9      9    9i
```

--ooOoo--