BIG DATA ANALYTICS

UNIT - I

INTRODUCTION TO BIG DATA: Big Data Analytics, Characteristics of Big Data – The Four Vs, importance of Big Data, Different Use cases, Data-Structured, Semi-Structured, Un-Structured, Introduction to Hadoop and its use in Solving big Data Problems. Comparison of Hadoop with RDBMS, Brief History of Hadoop, Apache Hadoop Ecosystem, Components of Hadoop, The Hadoop Distributed File System(HDFS), Architecture and Design of HDFS in detail, Working with HDFS(Commands)

INTRODUCTION:

With the fruition of the online services through the extensive use of the Internet, have fundamentally transformed various aspects of society, including businesses, economies, governments, and individual lifestyles. With the increase in all of these the transformation has indeed led to an explosion in data generation and collection. Thus creating a need for effective mechanisms to handle and derive value from this vast amount of information. This is where the concept of **Big Data** comes into play.

Data:

- Data is nothing but facts and statistics stored or free flowing over a network, generally it's raw and unprocessed.
- When data are processed, organized, structured or presented in a given context so as to make them useful, they are called **Information.**

What is Big Data? –

- The term **Big Data** refers to a huge volume of data that cannot be stored processed by any traditional data storage or processing units.
- In short such data is so large and complex that none of the **traditional data management tools** are able to store it or process it efficiently.
- The data of size MB(WordDoc ,Excel) or maximum GB(Movies, Codes), is termed as normal data range, but data in **Peta bytes** i.e. **10^15 byte size & beyond** is termed as **Big Data.**
- Today, there are millions of data sources that generate data at a very rapid rate. These data sources are present across the world. Some of the largest sources of data are social media platforms and networks. For example Facebook—it generates more than 500 terabytes of data every day. This data includes pictures, videos, messages, and more .
- Data also exists in different formats, like structured data, semi-structured data, and unstructured data. For example, in a regular Excel sheet, data is classified as structured data—with a definite format. In contrast, emails fall under semi-structured, and your pictures and videos fall under unstructured data. All this data combined makes up Big Data.

Other Definition: **Big data** is a term that is used to describe data that is high volume, high velocity, or high variety, which requires new technologies and techniques to capture, store, and analyze it.

Example of Big Data:

Following are some of the Big Data examples-

- **Stock Exchange Data**: The 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, posts submitted by millions of people worldwide 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.**
- **Search Engine Data:** Retrieve a wide variety of unprocessed information that is stored in SE databases.

DATA RANGE:

Normal Range		
Kilo Byte	(KB)	10^3
Mega Byte	(MB)	10^6
Giga Byte	(GB)	10^9
Tera Byte	(TB)	10^{12}

Big Data Range		
Peta Byte	(PB)	10^{15}
Exa Byte	(EB)	10^{18}
Zetta Byte	(ZB)	10^{21}
Yottabyte	(YB)	10^{24}

Data Generation:

- Data is growing at a tremendous pace that **is 3.5** quintillion bytes is how much data is created every day.
- A Single person generates **1.7 MB** of data **every second**.
- On average a person generates **around 1.7 GB** of **data per day**.
- Around 5.35 billion internet users worldwide, each person can generate approximately 15.87 TB of data daily.
- On average, Google Processes **99,000** searches each second.

- **Facebook** Generates **4PB** (Peta Bytes) of data daily, which is equivalent to million gigabytes.
- On average data consumption per user per month in **2023** in India was **24.1GB.**
- **Spotify** uses 2MB+ per 3-minute song which is 40MB every hour or 960 MB per day

Type of Media	Amount per Minute	Amount per Day
Emails sent	231.4 million	333.22 billion
Snaps shared on Snapchat	2.43 million	3.5 billion
Tweets shared on Twitter	347,200	499.97 million

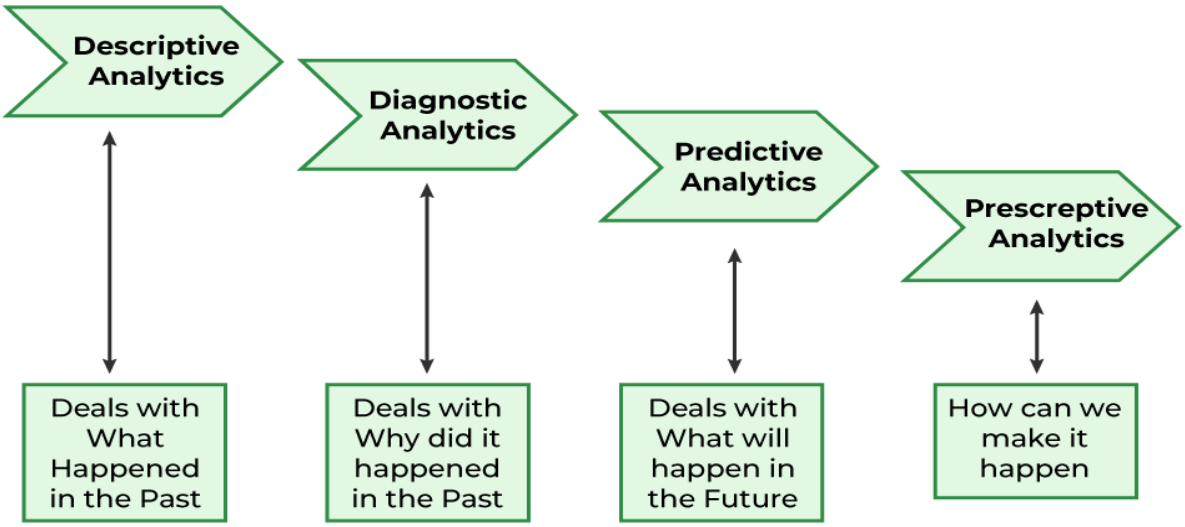
Big Data Challenges: The major challenges associated with big data are as follows −

- Capturing data
- Storage
- Security
- Analysis
- To fulfill the above challenges, organizations normally take the help of enterprise servers.

BIG DATA ANALYTICS

- Big Data analytics is the process of examining and deriving meaningful insights from large and complex datasets.

- It involves the use of various techniques, technologies, and tools to analyze data with the goal of uncovering patterns, trends, correlations, and valuable information that can lead to informed decision-making and strategic planning.
- Data analytics can shape business processes, improve decision-making, and business growth.
- There are **mainly 4 types of analytics**:



- **Descriptive Analytics:** Descriptive Analytics is focused solely on historical data, describing without judgment. It tells what happen in the past and uses some statistics functions and returns answers**("What happened?").** This helps in creating reports, like a company's revenue, profit, sales, and so on.
  - Ex: Google Analytics and Netflix
- **Diagnostic Analytics:** It finds out the root cause for happened thing in medical diagnosis**. (why Did this happen.)**
- This is done to understand what caused a problem in the first place. Techniques like data mining, and data recovery are all examples.

Organizations use diagnostic analytics because they provide an in-depth insight into a particular problem

- Use Case: An **e-commerce company's** report shows that their sales have gone down, although customers are adding products to their carts. This can be due to **various reasons** like the **form didn't load correctly, the shipping fee is too high,** or there are not enough payment options available. This is where you can use diagnostic analytics to find the reason.

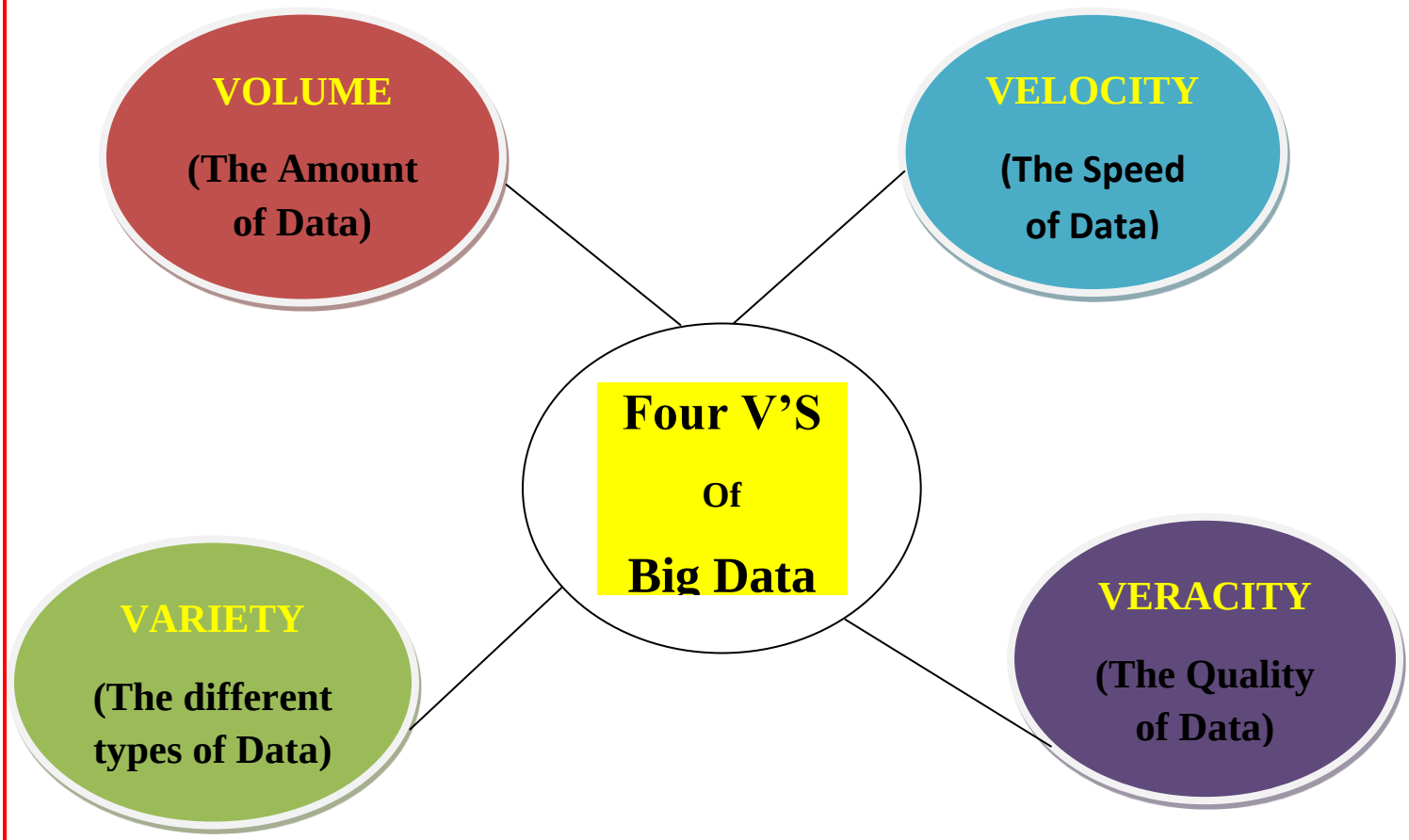
- **Predictive Analytics** : This type of analytics looks into the historical and present data to make predictions of the future. States that some specified event will happen in future i.e. what might happen in future, to reduce loses for the users. **("What might happen in the future?").** Predictive analytics uses data mining, AI, and machine learning to analyze current data and make predictions about the future. It works on predicting customer trends, market trends, and so on.
  - **Example:** Netflix uses AI-powered algorithms to make predictions based on the user's watch history, search history, demographics, ratings, and preferences

- **Prescriptive Analytics:** It deals with Imposition of a rule/method. What we should do like what kind of action should be performed. This type of analytics prescribes the solution to a particular problem. Perspective analytics works with both descriptive and predictive analytics. Most of the time, it relies on AI and machine learning.

Eg: self driving cars which analyzes environment and moves on the road.**("What should we do next?")**

Characteristics of Big Data – The Four V's:

- **Volume**
- **Velocity**
- **Variety**
- **Veracity**



Volume: Volume refers to the **unimaginable amounts** of information generated every second from many sources such as social media, cell phones, credit cards, sensors, images, video, and many more.

Velocity:

- Velocity Refers to the **speed** by which the data is created in **real-time**.
- It deals with the speed at the data flows from sources like application logs, business processes, networks, and social media sites, sensors, mobile devices**,** etc.
- It contains the linking of incoming **data sets speeds, rate of change**, and **activity bursts**.

Variety:

- It Refers to Data **generated** in multiple varieties.
- It refers to nature of data that is **structured, semi-structured** and **unstructured** data that are being collected from different sources.
- Variety is basically the arrival of data from new sources that are both inside and outside of an enterprise. It can be structured, semi-structured and unstructured.
    - **Structured data**: This data is basically an organized data. It is in a tabular form. Structured Data is stored in the relational database management system.
    - **Semi- Structured data**: This data is basically a semi-organised data. It is generally a form of data that do not conform to the formal structure of data. Log files are the

examples of this type of data, **JSON, XML,** and **email etc**.

- **Unstructured data**: This data basically refers to unorganized data.
- It generally refers to data that doesn't fit neatly into the traditional row and column structure of the relational database.
- Texts, pictures, videos etc. are the examples of unstructured data which can't be stored in the form of rows and columns.

Veracity:

- Veracity refers to the Trustworthiness in terms of quality and accuracy.
- Veracity means how much the data is reliable.
- It has many ways to filter or translate the data. Veracity is the process of being able to handle and manage data efficiently.
- Big Data is also essential in business development.

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 of following Reasons:

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 learnings.

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/Control online reputation:

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.

5. Big Data Analytics in Product Development:

Another huge advantage of big data is the ability to help companies innovate and redevelop their products

Different Use Cases /Applications:

Banking, Financial Services, and Insurance (BFSI)

- BFSI is one of the most data-intensive domains in the world economy.
- Financial institutions have huge amounts of customer data, such as information on customer profile data collected for KYC, deposits and withdrawals at ATMs, online payments, and more.
- Big data technologies enable financial institutions to easily access data and eliminate redundancy and overlapping.

Banking and finance institutions leverage big data technologies data sets to maximize customer understanding and gain a competitive advantage.

Big Data use cases in the BFSI industry

- Improved levels of customer insight
- Customer engagement
- Fraud detection and prevention
- Market trading analysis
- Risk management
- New data-driven products and services

Healthcare

- Healthcare institutions gather a large amount of data in the form of patient details, physician's prescriptions, medical imaging, lab reports, insurance, and other administrative data. Using big data, the vast amount of data can be stored systematically and easily accessed when needed.
- Many healthcare institutions are using electronic health records (EHR) to gain a deeper understanding of patient disease patterns.

- Using big data, healthcare practitioners can access a wide range of data and make informed decisions related to the patient's health, hospital performance, and more.

Big Data use cases in the Healthcare industry

- Improved patient predictions
- Real-time alerts
- Electronic Health Records (EHRs)
- Better patient engagement
- Smoother hospital administration

Education

- In the education sector, a lot of data is collected in the form of names of students enrolled in a program/course, enrollment year, course details, student ID, marks obtained in each subject, and more.

- Using big data, educators can store this information efficiently and identify patterns and trends to spot opportunities for positive change in the performance of both the students and the educational institutions.

- Big data analytics help educators reveal trends in students' behavior and their preferences to create customized programs. It also gives a base to evaluate the state of the entire education system.

Big Data use cases in the Education industry

- Create customized programs
- Improve student's results
- Reduce dropouts
- Identify leaner's strengths
- Data-driven decision making

Manufacturing

- In manufacturing, data is gathered from machines, devices, and operators at every stage of production. Big data help manufacturers store this data efficiently.
- The use of big data also allows firms to identify new ways to save costs and improve product quality.
- Using big data analytics, companies can find patterns to solve existing problems and improve the overall process.Monitoring equipment performance and predicting maintenance needs to minimize downtime.

Big Data use cases in the Manufacturing industry

- Customize product design
- Predictive quality
- Anomaly detection
- Better management of supply chain
- Production forecasting
- Yield improvement
- Risk evaluation

Social Media:

- Analyzing user interactions for sentiment analysis, brand perception, and trend forecasting.

IT :

- One of the largest users of Big Data, IT companies around the world are using Big Data to optimize their functioning, enhance employee productivity, and minimize risks in business operations. By combining Big Data technologies with ML and AI, the IT sector is

continually powering innovation to find solutions even for the most complex of problems.

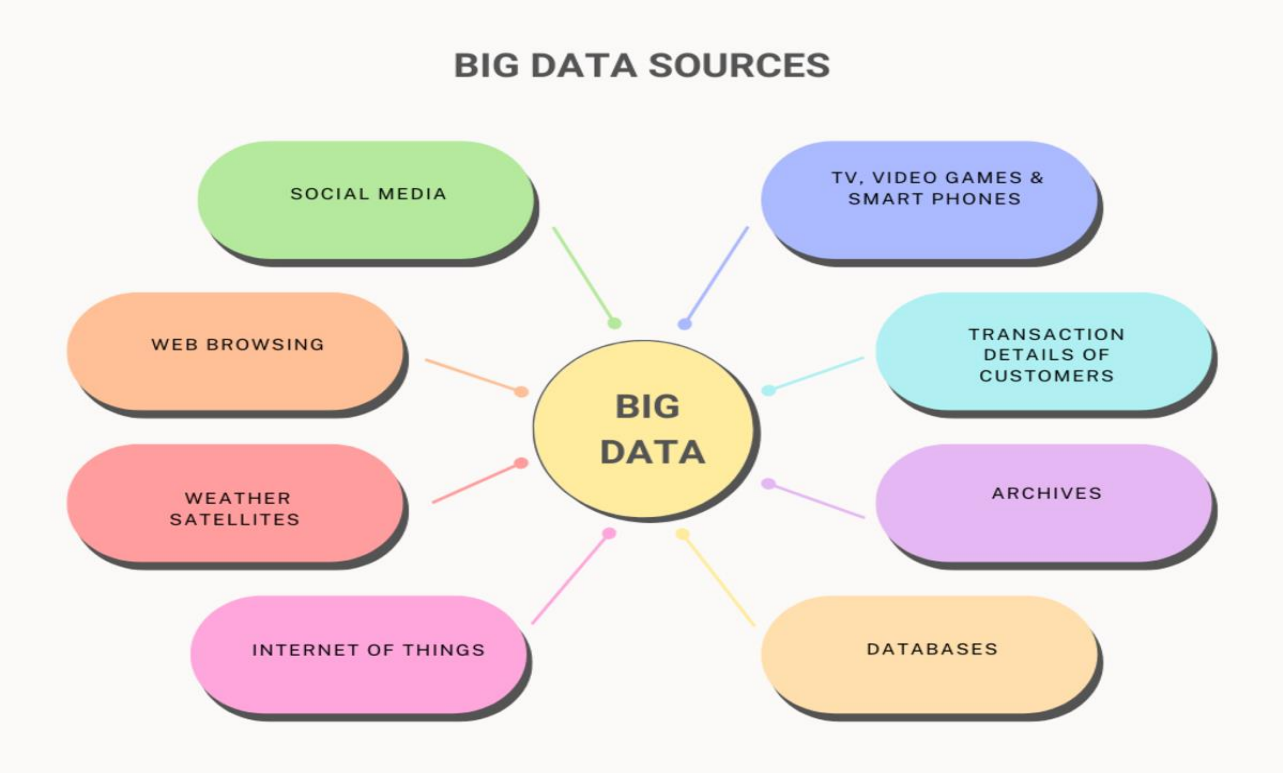
Ecommerce –

- Predicting customer trends and optimizing prices are a few of the ways e-commerce uses Big Data analytics

Marketing – Big Data analytics helps to drive high ROI marketing campaigns, which result in improved sales.

Media and entertainment – Used to understand the demand of shows, movies, songs, and more to deliver a personalized recommendation list to its users

SOURCES OF BIG DATA



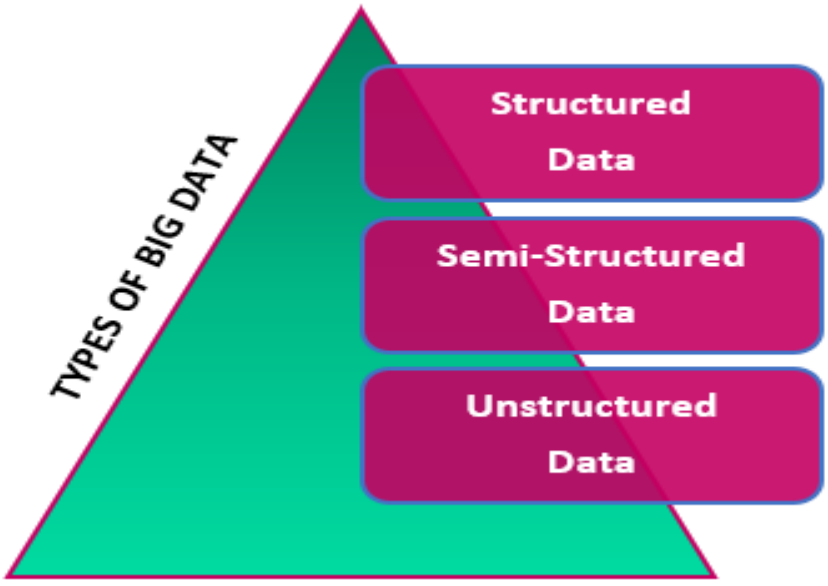
Source of Big Data:

- **Social Media:** Today's world a good percent of the total world population is engaged with social media like Facebook, WhatsApp, Twitter, YouTube, Instagram, etc. Each activity on such media like uploading a photo, or video, sending a message, making comment, putting like, etc create data.
- **A sensor placed in various places:** Sensor placed in various places of the city that gathers data on temperature, humidity, etc. A camera placed beside the road gather information about traffic condition and creates data. Security cameras placed in sensitive areas like airports, railway stations, and shopping malls create a lot of data.
- **Customer Satisfaction Feedback:** Customer feedback on the product or service of the various company on their website creates data. For Example, retail commercial sites like Amazon, Walmart, Flipkart, and Myntra gather customer feedback on the quality of their product and delivery time. Telecom companies, and other service provider organizations seek customer experience with their service. These create a lot of data.
- **E-commerce:** In e-commerce transactions, business transactions, banking, and the stock market, lots of records stored are considered one of the sources of big data. Payments through credit cards, debit cards, or other electronic ways, all are kept recorded as data.
- **Global Positioning System (GPS):** GPS in the vehicle helps in monitoring the movement of the vehicle to shorten the path to a destination to cut fuel, and time consumption. This system creates huge data on vehicle position and movement.
- **Transactional Data:** Transactional data, as the name implies, is information obtained through online and offline transactions at various points of sale. The data contains important information about transactions, such as the date and time of the transaction, the location where it took place, the items bought,

their prices, the methods of payment, the discounts or coupons that were applied, and other pertinent quantitative data. These are some of the sources of transactional data: orders for payment, Invoices, E-receipts and recordkeeping etc.

- **Machine Data:** Automatically generated machine data is produced in reaction to an event or according to a set timetable. This indicates that all of the data was compiled from a variety of sources, including satellites, desktop computers, mobile phones, industrial machines, smart sensors, SIEM logs, medical and wearable devices, road cameras, IoT devices, and more. These sensors are used to capture this kind of information: In a broader sense, machine data includes data that is generated by servers, user applications, websites, cloud programmes, and other sources.

Big Data Formats/Types:

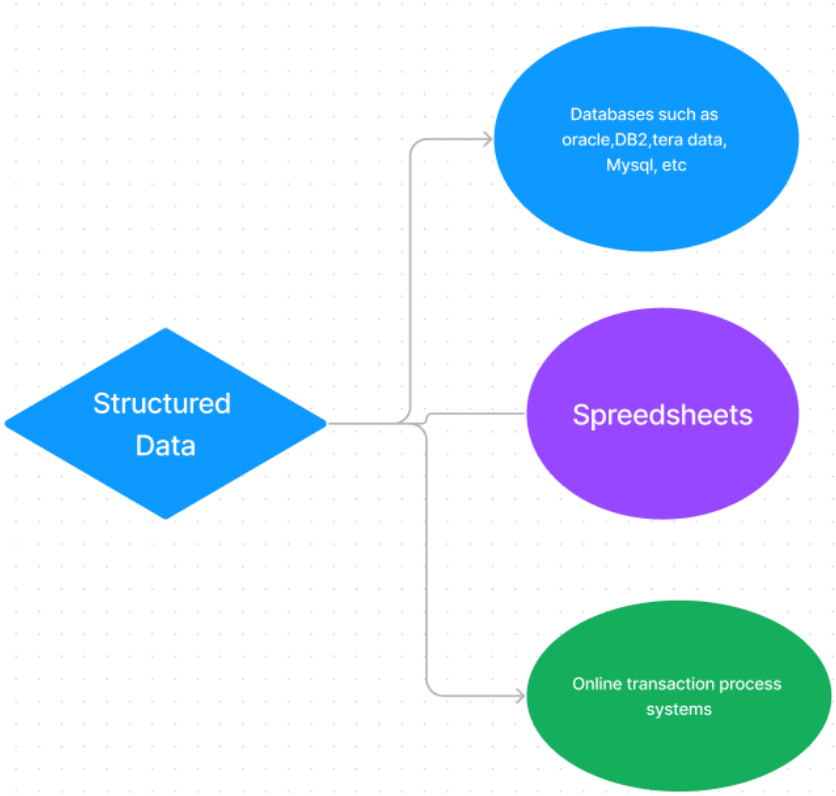


STRUCTURED DATA: The data that can be stored, accessed processed & retrieved in the form of fixed format is termed as a 'structured' data.

- Structured data can be crudely defined as the data that resides in a fixed field within a record.
- It is represented in a Tabular Format.
- **Eg:** An '**Employee'** table in a database is an example of Structured Data, etc.

Employees Table

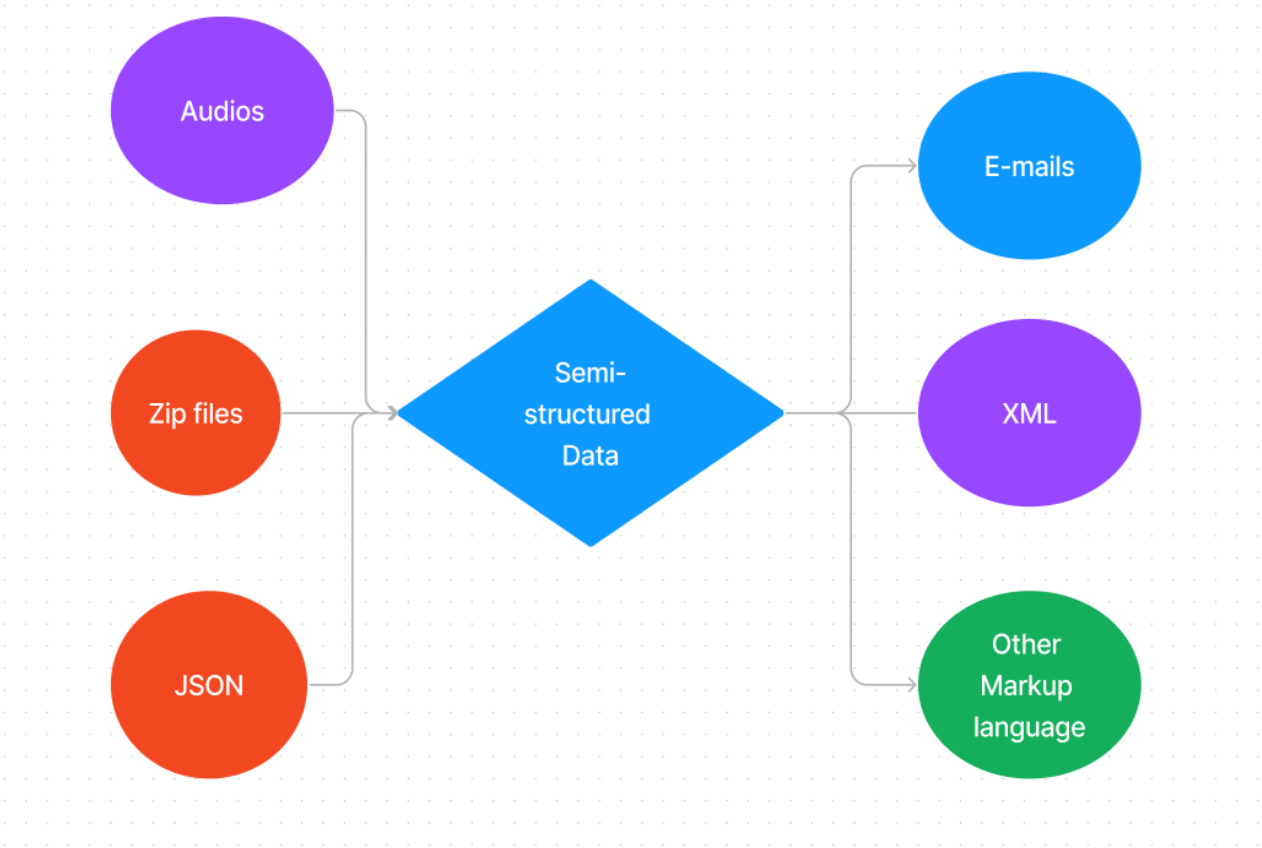
Employee ID	Last	First	Phone Number	Work Location	Project 1	Project 2	Project 3
AA123	Adams	Adam	222-333-4444	Smith Tower 22222			
BB234	Breen	Betty	333-444-5555	Nakatomi Plaza 33333			
CC456	Chen	Chao	444-555-6666	Smith Tower 22222	RDBMS Update May 2017	Network Security Audit May 2018	
DD789	Dickinson	Durah	555-666-7777	Nakatomi Plaza 33333	OS Update for PCs Sept 2017	OS Update for Macs Jan 2018	
EE012	Edinburgh	Elvis	666-777-8888	Tall Tower 22222	Network Security Audit May 2018	OS Update for Macs Jan 2018	
FF345	Fawzi	Farah	888-999-0000	Tall Tower 22222			
GG456	Giovanni	Georgio	000-111-2222	Nakatomi Plaza 33333	RDBMS Update May 2017	Network Security Audit May 2018	



SEMI-STRUCTURED DATA:

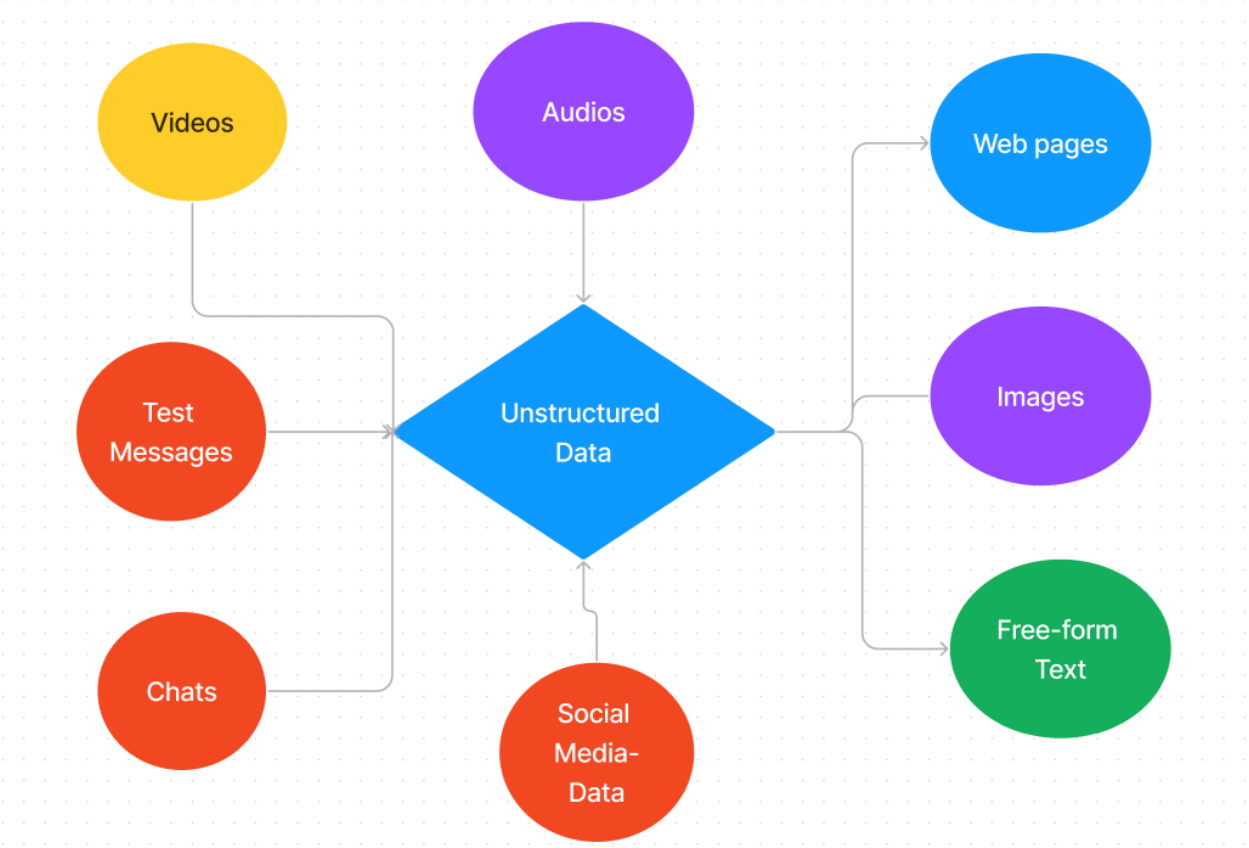
- Semi-structured data refers to data that is not captured or formatted in conventional ways.
- Semi-structured data does not follow the format of a tabular data model or relational databases because it does not have a fixed schema.
- Since semi-structured data doesn't need a structured query language, it is commonly called NoSQL data.
- However, the data is not completely raw or unstructured, and does contain some structural elements such as tags and organizational metadata that make it easier to analyze.
- The advantages of semi-structured data is that it is more flexible and simpler to scale compared to **structured data**.

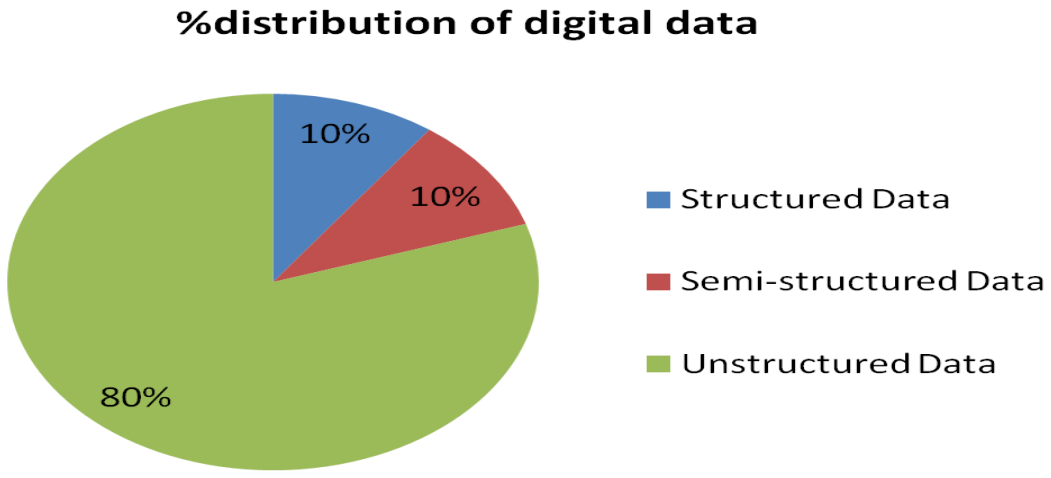
Eg: HTML code, Email's
graphs and tables,
XML ,JSON documents



UN-STRUCTURED DATA:

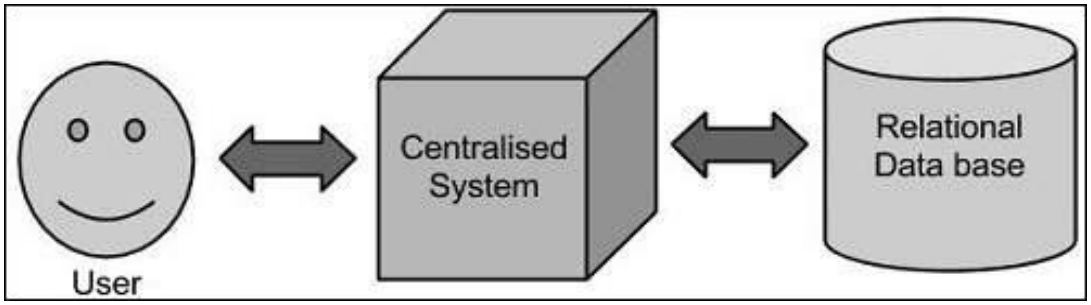
- Unstructured data is the kind of data that doesn't adhere to any definite schema or set of rules.
- Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data.
- A typical example of unstructured data is a heterogeneous data source containing a combination of simple text files, images, videos etc.
- Additionally, Unstructured data is also known as "dark data" because it cannot be analyzed without the proper software tools
- Eg: Audio, simple text files, images, PDF's, videos ,web pages etc.


Videos
Audios
Web pages
Test Messages
Unstructured Data
Images
Free-form Text
Chats
Social Media-Data


%distribution of digital data
10%
10%
80%
Structured Data
Semi-structured Data
Unstructured Data

Introduction to Hadoop and its use in Solving big Data Problems

Traditional Approach of Storing Data:



Limitation

- This approach works fine with those applications that process less voluminous data that can be accommodated by standard database servers, or up to the limit of the processor that is processing the data.
- But when it comes to dealing with huge amounts of scalable data, it is a hectic task to process such data through a single database.

To Overcome the Limitation **, HADOOP** Comes in as a Solution For Storing & Processing Large Voluminous Amounts of Data.

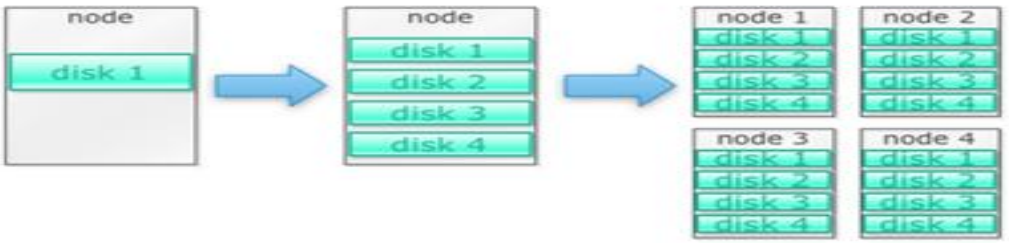
INTRODUCTION TO HADOOP:

- Hadoop Stands for **High Availability Distributed Object Oriented Platform**. Hadoop is an open-source framework Designed for distributed storage and processing large datasets across Clusters of Commodity Hardware. Or other definition is: It is used to store process and analyze data which are very huge in volume.
- Its framework is based on Java programming

- It was developed by Doug Cutting and Mike Cafarella and is now, administered by the Apache Software Foundation.
- The primary goal of Hadoop is to enable the processing of vast amounts of data in a cost-effective and efficient manner
- It is designed to handle big data and is based on the MapReduce programming model, which allows for the parallel processing of large datasets.
- Unlike traditional, structured platforms, Hadoop is able to store any kind of data in its native format and to perform a wide variety of analyses and transformations on that data.
- Hadoop solves the problem of Big data by storing the data in distributed form in different machines. There are plenty of data but that data have to be store in a cost effective way and process it efficiently.
- Hadoop stores terabytes, and even petabytes, of data inexpensively.
- It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more.
- It is robust and reliable and handles hardware and system failures automatically, without losing data or interrupting data analyses.

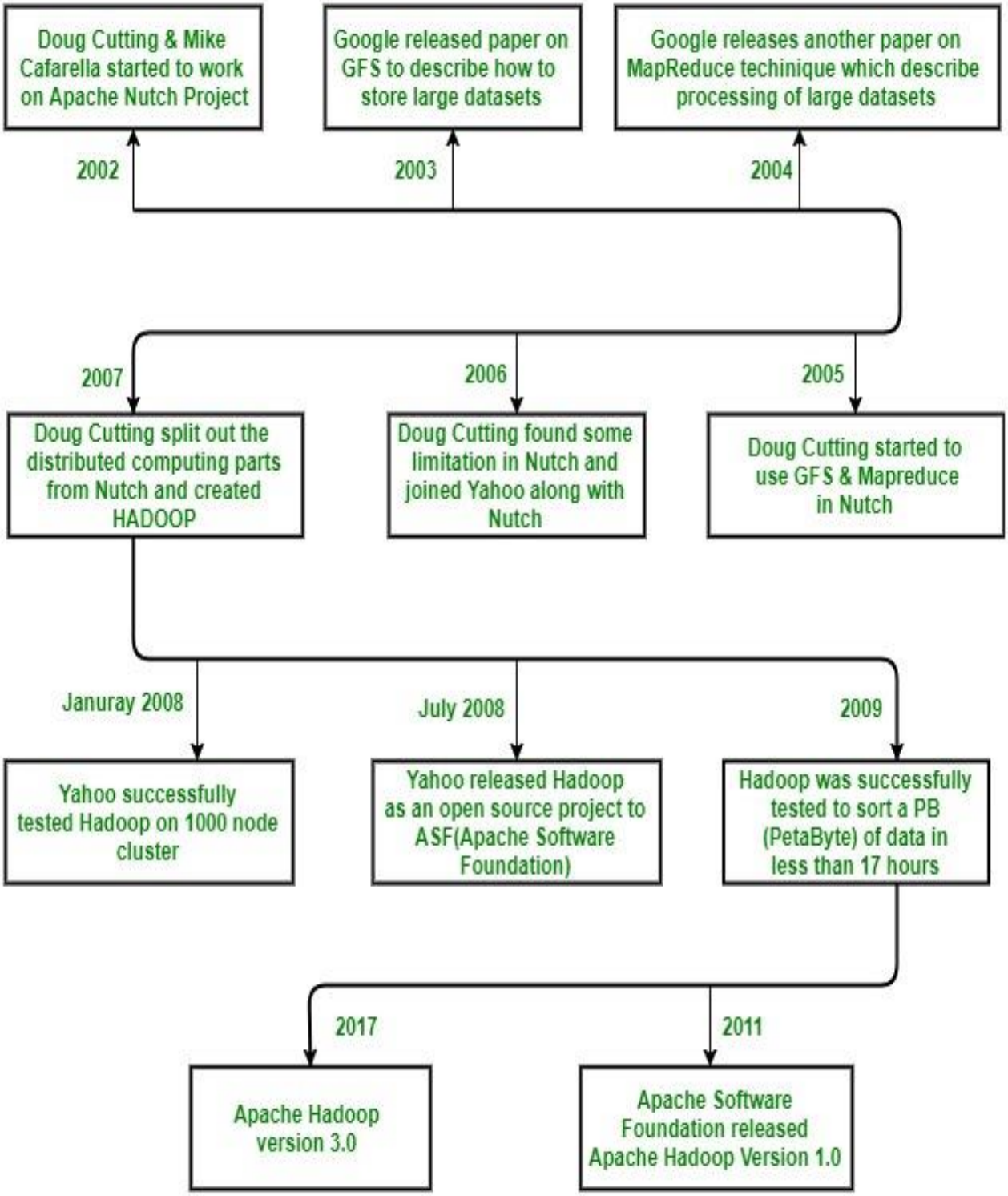
How Hadoop Solves the Big Data Problem:

- **Hadoop is built to run on a cluster of machines:**



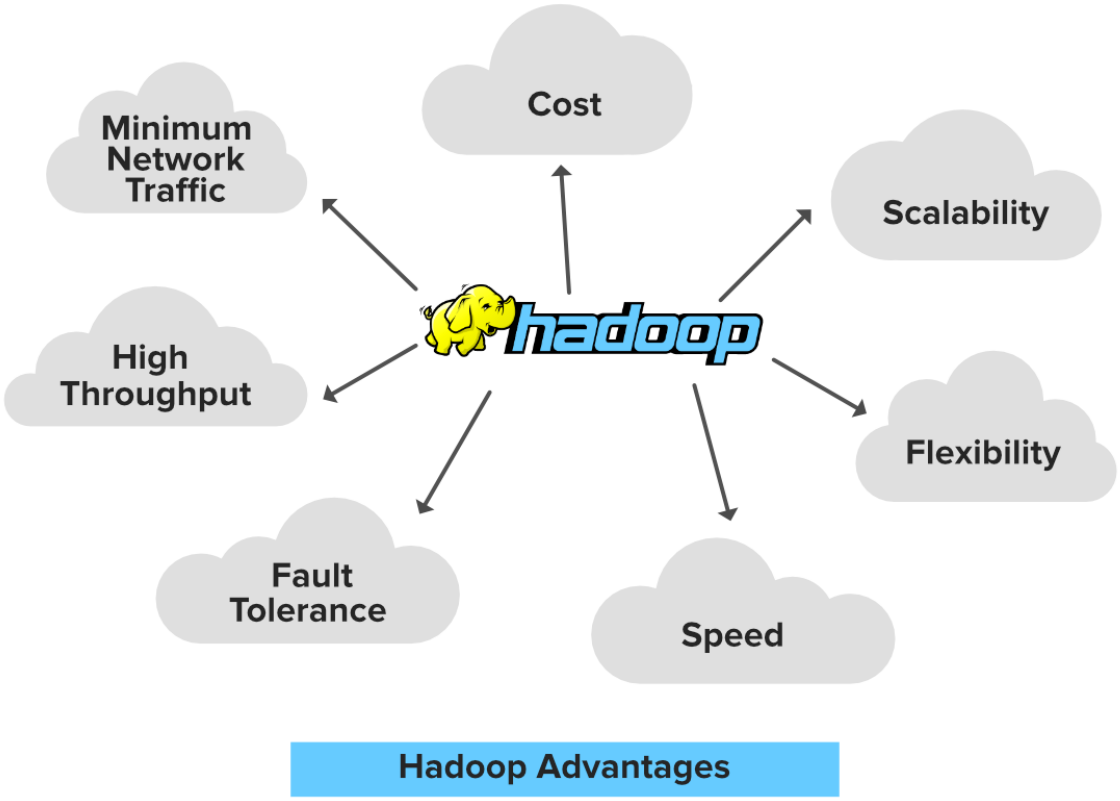
- **Hadoop clusters scale horizontally**: More storage and compute power can be achieved by adding more nodes to a Hadoop cluster. This eliminates the need to buy more and more powerful and expensive hardware.
- **Hadoop can handle unstructured/semi-structured data**: Hadoop doesn't enforce a schema on the data it stores. It can handle arbitrary text and binary data. So Hadoop can digest any unstructured data easily.
- **Hadoop clusters provides storage and computing**: Hadoop clusters, however, provide storage and distributed computing all in one.
- **Hadoop provides storage for big data at reasonable cost**: Storing big data using traditional storage can be expensive. Hadoop is built around commodity hardware, so it can provide fairly large storage for a reasonable cost. Hadoop has been used in the field at petabyte scale.
- **With Hadoop, you can store data longer**: To manage the volume of data stored, companies periodically purge older data. For example, only logs for the last three months could be stored, while older logs were deleted. With Hadoop it is possible to store the historical data longer. This allows new analytics to be done on older historical data.

BRIEF HISTORY OF HADOOP:



1. In **2002, Doug Cutting** and **Mike Cafarella** started to work on a project, **Apache Nutch.** Apache Nutch is a highly extensible and scalable open-source web crawler software project.
   While working on Apache **Nutch**, they were dealing with big data. To store that data they have to spend a lot of costs which becomes the consequence of that project. This problem becomes one of the important reason for the emergence of Hadoop.
2. In **2003,** Google introduced a file system known as **GFS** (**Google file system**). It is a proprietary distributed file system developed to provide efficient access to data.
3. In **2004**, Google released a white paper on **Map Reduce**. This technique simplifies the data processing on large clusters.
4. In **2005,** Doug Cutting and Mike Cafarella introduced a new file system known as **NDFS (Nutch Distributed File System).** This file system also includes Map reduce.
5. In **2006**, Doug Cutting quit Google and joined Yahoo. On the basis of the Nutch project, Dough Cutting introduces a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System). Hadoop first version 0.1.0 released in this year.
6. Doug Cutting gave named his project **Hadoop** after his **son's toy elephant**.
7. In **2007,** Dough cutting split the Distributed computing parts from Nutch ad Created Hadoop. Yahoo runs two clusters of 1000 machines.
8. In **2008,** Yahoo Successfully Tested Hadoop on **1000 Node Cluster**. Yahoo Releasd Hadoop as open source project to **Apache Software Foundation.**
9. In **2009,** Hadoop was successfully tested to sort Peta bytes of data in less than17hrs.
10. In **2011**, Hadoop **1.0** was released.
11. In **2012,** Hadoop **2.0** was released.
12. In **2017,** Hadoop **3.0** was released.

13. In **2020,** Hadoop **3.1.3** was released
14. In **2022,** Hadoop **3.2.4** was released



Hadoop Advantages

1. Cost

Hadoop is open-source and uses cost-effective commodity hardware which provides a cost-efficient model, unlike traditional Relational databases that require expensive hardware and high-end processors to deal with Big Data. The problem with traditional Relational databases is that storing the Massive volume of data is not cost-effective, so the company's started to remove the Raw data. which may not result in the correct scenario of their business.

2. Scalability

Hadoop is a highly scalable model. A large amount of data is divided into multiple inexpensive machines in a cluster which is processed parallelly. the number of these machines or nodes can be increased or decreased as per the enterprise's requirements. In traditional RDBMS(Relational DataBase Management System) the systems can not be scaled to approach large amounts of data.

3. Flexibility

Hadoop is designed in such a way that it can deal with any kind of dataset like structured(MySql Data), Semi-Structured(XML, JSON), Un-structured (Images and Videos) very efficiently. This means it can easily process any kind of data independent of its structure which makes it highly flexible. which is very much useful for enterprises as they can process large datasets easily, so the businesses can use Hadoop to analyze valuable insights of data from sources like social media, email, etc. with this flexibility Hadoop can be used with log processing, Data Warehousing, Fraud detection, etc.

4. Speed

Hadoop uses a distributed file system to manage its storage i.e. HDFS(Hadoop Distributed File System). In DFS(Distributed File System) a large size file is broken into small size file blocks then distributed among the Nodes available in a Hadoop cluster, as this massive number of file blocks are processed parallelly which makes Hadoop faster, because of which it provides a High-level performance as compared to the traditional DataBase Management Systems.

5. Fault Tolerance

Hadoop uses commodity hardware(inexpensive systems) which can be crashed at any moment. In Hadoop data is replicated on various

DataNodes in a Hadoop cluster which ensures the availability of data if somehow any of your systems got crashed.

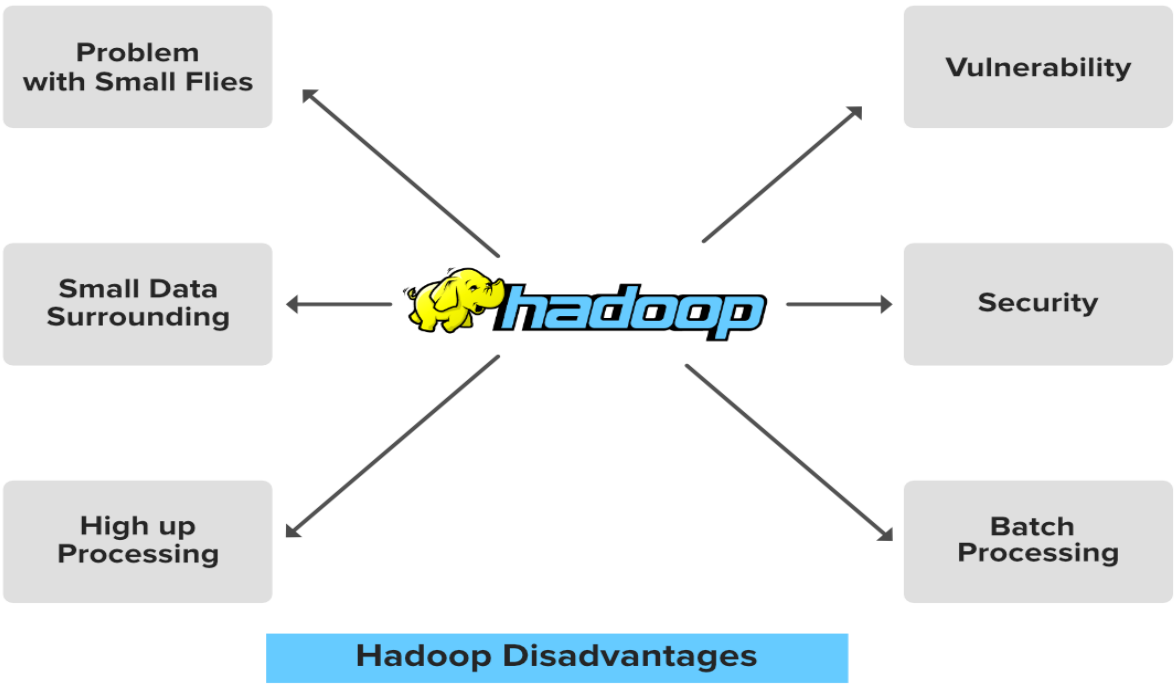
6. High Throughput

Hadoop works on Distributed file System where various jobs are assigned to various Data node in a cluster, the bar of this data is processed parallelly in the Hadoop cluster which produces high throughput. Throughput is nothing but the task or job done per unit time.

7. Minimum Network Traffic

In Hadoop, each task is divided into various small sub-task which is then assigned to each data node available in the Hadoop cluster. Each data node process a small amount of data which leads to low traffic in a Hadoop cluster.

Hadoop Disadvantages:



1. Problem with Small files

Hadoop can efficiently perform over a small number of files of large size. Hadoop stores the file in the form of file blocks which are from 128MB in size(by default) to 256MB. Hadoop fails when it needs to access the small size file in a large amount. This so many small files surcharge the Namenode and make it difficult to work.

2. Vulnerability

Hadoop is a framework that is written in java, and java is one of the most commonly used programming languages which makes it more insecure as it can be easily exploited by any of the cyber-criminal.

3. Low Performance In Small Data Surrounding

Hadoop is mainly designed for dealing with large datasets, so it can be efficiently utilized for the organizations that are generating a massive volume of data. It's efficiency decreases while performing in small data surroundings.

4. Lack of Security

Data is everything for an organization, by default the security feature in Hadoop is made un-available. So the Data driver needs to be careful with this security face and should take appropriate action on it. Hadoop uses Kerberos for security feature which is not easy to manage. Storage and network encryption are missing in Kerberos which makes us more concerned about it.

5. High Up Processing

Read/Write operation in Hadoop is immoderate since we are dealing with large size data that is in TB or PB. In Hadoop, the data read or write done

from the disk which makes it difficult to perform in-memory calculation and lead to processing overhead or High up processing.

6. Supports Only Batch Processing

The batch process is nothing but the processes that are running in the background and does not have any kind of interaction with the user. The engines used for these processes inside the Hadoop core is not that much efficient. Producing the output with low latency is not possible with it.

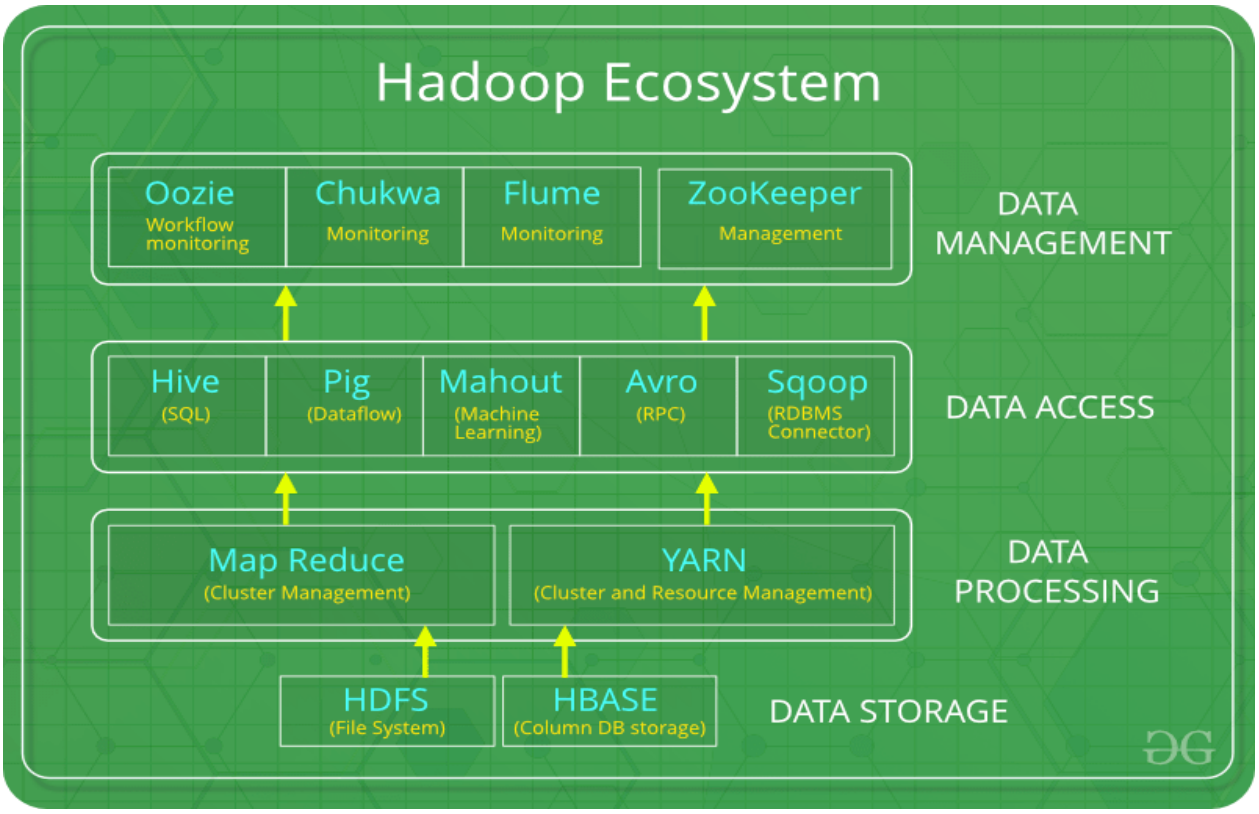
Comparison of Hadoop with RDBMS

	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.(**Online Transaction Processing)**	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	The data schema of RDBMS is static type.	The data schema of Hadoop is dynamic type.
8.	High data integrity available.	Low data integrity available than RDBMS.
9.	Cost is applicable for licensed software.	Free of cost, as it is an open source software.

APACHE HADOOP ECOSYSTEM:

Apache Hadoop ecosystem is a collection of open-source projects and tools that work together to extend and enhance the capabilities of the Hadoop platform. These projects provide various functionalities for data storage, processing, analysis, and management. Here are some key components of the Apache Hadoop ecosystem:



HDFS -> Hadoop Distributed File System

- Hadoop Distributed File System (HDFS), HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- It is a distributed file system able to store large files running over the cluster of commodity hardware.
- HDFS consists of two core components i.e.
- **Name node**
- **Data Node**

HBase : **NoSQL Database**

- HBase is a column-oriented non-relational database management system that runs on top of Hadoop Distributed File System (HDFS). This means that data is stored in individual columns, and indexed by a unique row key.
- Apache HBase is an open-source, NoSQL, distributed big data store. It enables random, strictly consistent, real-time access to petabytes of data.
- HBase is very effective for handling large, sparse datasets.
- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database
- HBase integrates seamlessly with Apache Hadoop and the Hadoop ecosystem and runs on top of the Hadoop Distributed File System (HDFS
- It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data.
- A distributed, scalable NoSQL database that provides random access to large amounts of structured data.
- It is well suited for real-time data processing or random read/write access to large volumes of data.

MAPEDUCE-> **Data processing using programming**

- A programming model and processing framework for distributed computation, widely used for batch processing of large datasets.

- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others.
- In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.
- This is another data processing layer of Hadoop.
- It has the capability to process large structured and unstructured data as well as to manage very large data files in parallel by dividing the job into a set of independent tasks (sub-job).
- This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
- MapReduce makes the use of two functions i.e. **Map() and Reduce()** whose task is:
- Map() performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
- Reduce(), as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

YARN -> Yet Another Resource Negotiator (allocates Resources)

- YARN stands for Yet Another Resource Negotiator. It is one of the core components in open source Apache Hadoop suitable for resource management.
- It is responsible for managing workloads, monitoring, and security controls implementation.
- YARN is the one who helps to manage the resources across the clusters. In short, it performs job scheduling and resource allocation for the Hadoop System.
- It also allocates system resources to the various applications running in a Hadoop cluster while assigning which tasks should be executed by each cluster nodes. YARN has three main components:
  - Resource Manager
  - Node Manager
  - Application Manager

Resource manager has the privilege of allocating resources for the applications in a system .

Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager.

Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

HIVE-> Data Processing Services using Query (SQL)

- A data warehousing and SQL-like query language that provides a high-level abstraction over Hadoop and allows users to query and analyze data using SQL queries.
- It provides SQL type language for querying called HiveQL or HQL(Hive Query Language). **HIVE + SQL = HQL**

- Hive is an ETL and Data warehousing tool used to query or analyze large datasets stored within the Hadoop ecosystem.
- Hive has three main functions: **data summarization, query**, and **analysis** of unstructured and semi-structured data in Hadoop.
- It features a SQL-like interface, HQL language that works similar to SQL and automatically translates queries into MapReduce jobs.
- Hive uses a language called HiveQL(HQL), which is similar to SQL.
- Hive QL works as a translator which translates the SQL queries into MapReduce Jobs, which will be executed on Hadoop.
- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: JDBC Drivers and HIVE Command Line.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

PIG:

- Apache Pig is a high-level language platform for analyzing and querying large data sets that are stored in HDFS.
- Pig works as an alternative language to Java programming for MapReduce and generates MapReduce functions automatically.
- Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.
- Pig can translate the Pig Latin scripts into MapReduce which can run on YARN and process data in HDFS cluster.

- Pig is best suitable for solving complex use cases that require multiple data operations.
- It is more like a processing language than a query language (ex:Java, SQL). Pig is considered as a highly customized one because the users have a choice to write their functions by using their preferred scripting language.
- A platform for analyzing large datasets using a high-level scripting language called Pig Latin, which simplifies data processing tasks.
- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.
- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.
- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

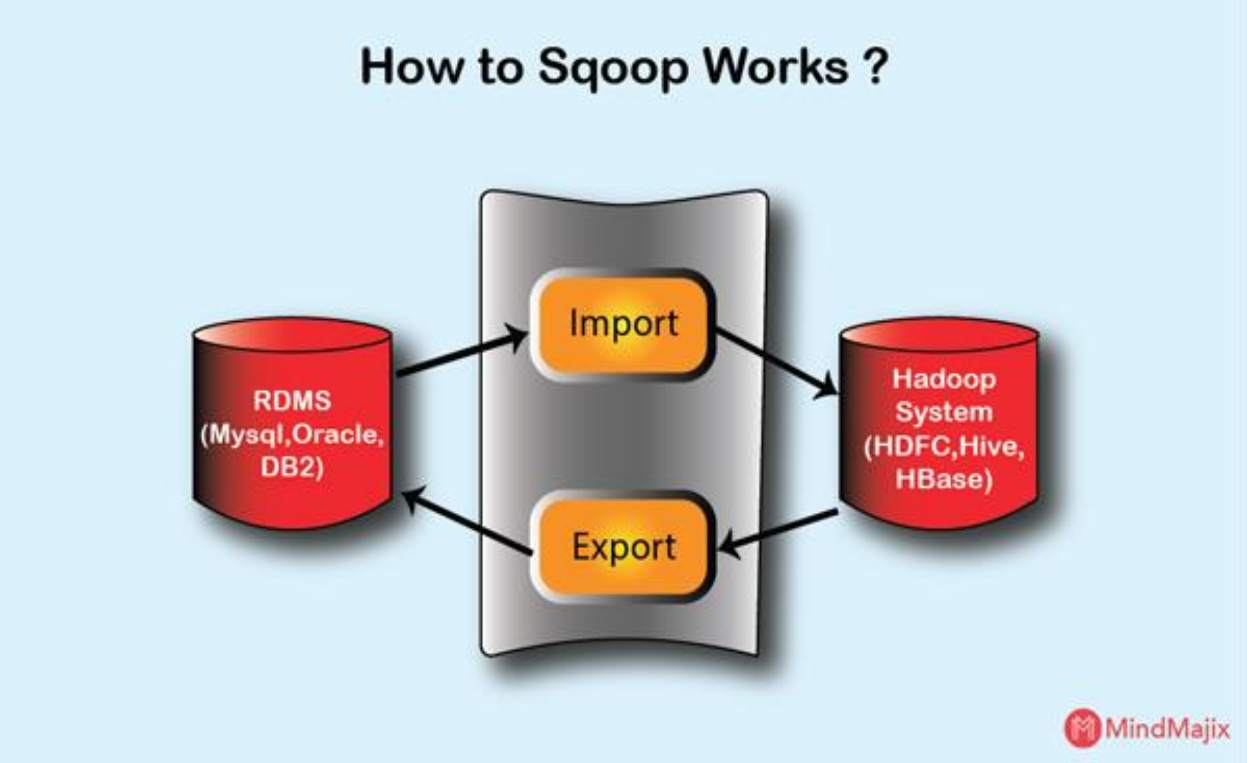
MAHOUT: **MLlib -> Machine Learning**

- A machine learning library for building scalable and effective machine learning algorithms on top of Hadoop.
- Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or on the basis of algorithms.
- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning. It allows invoking algorithms as per our need with the help of its own libraries.
- It implements popular machine learning techniques such as:
    - Filtering

- Classification
- Clustering

SQOOP -> **A tool for transferring data between Hadoop and relational databases, facilitating the import and export of data.**

- Sqoop works as a front-end loader of Big data. Sqoop is a front-end interface that enables in moving bulk data from Hadoop to relational databases and Vice versa.
- Sqoop fulfills the growing need to transfer data from the mainframe to HDFS.



Avro:

- Apache Avro is a part of the Hadoop ecosystem, and it works as a data serialization system.

- It is an open-source project which helps Hadoop in data serialization and data exchange.
- Avro enables big data in exchanging programs written in different languages. It serializes data into files or messages.
- Avro Schema: Schema helps Avro in the serialization and deserialization process without code generation. Avro needs a schema for data to read and write. Whenever we store data in a file it's schema also stored along with it, with this the files may be processed later by any program.

Avro features

- Avro makes Fast, compact, dynamic data formats.
- It has a Container file to store continuous data format.
- It helps in creating efficient data structures.

Oozie (job Scheduling) :

Apache Oozie is an open-source workflow scheduler that helps manage Hadoop jobs in Big Data analytics

- Apache Ooze is a tool in which all sort of programs can be pipelined in a required manner to work in Hadoop's distributed environment.
- Oozie works as a scheduler system to run and manage Hadoop jobs.
- Oozie allows combining multiple complex jobs to be run in a sequential order to achieve the desired output.
- It is strongly integrated with Hadoop stack supporting various jobs like Pig, Hive, Sqoop, and system-specific jobs like Java, and Shell.
- Oozie is an open-source Java web application.

Chukwa:

- Chukwa is a data collection system for monitoring and analyzing large distributed systems.

- It’s designed to collect data from various sources within a Hadoop cluster, such as log files and metrics, and store it in a central repository for further analysis.
- Chukwa is a part of the Hadoop ecosystem and can help manage and understand your Hadoop cluster’s performance

Apache Flume: Data Ingesting Services

- Flume collects, aggregates, and moves large sets of data from its origin and sends it back to HDFS.
- It works as a fault-tolerant mechanism.
- It helps in transmitting data from a source into a Hadoop environment.
- Flume enables its users in getting data from multiple servers immediately into Hadoop.
- A distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of log and event data from various sources like network traffic, social media,email etc to HDFS.

Apache Zookeeper-> Managing Cluster

- A distributed coordination service that helps manage and synchronize distributed applications and services
- Apache Zookeeper is an open-source project designed to coordinate multiple services in the Hadoop ecosystem.
- Organizing and maintaining a service in a distributed environment is a complicated task.
- Zookeeper solves this problem with its simple APIs and Architecture. Zookeeper allows developers to focus on core

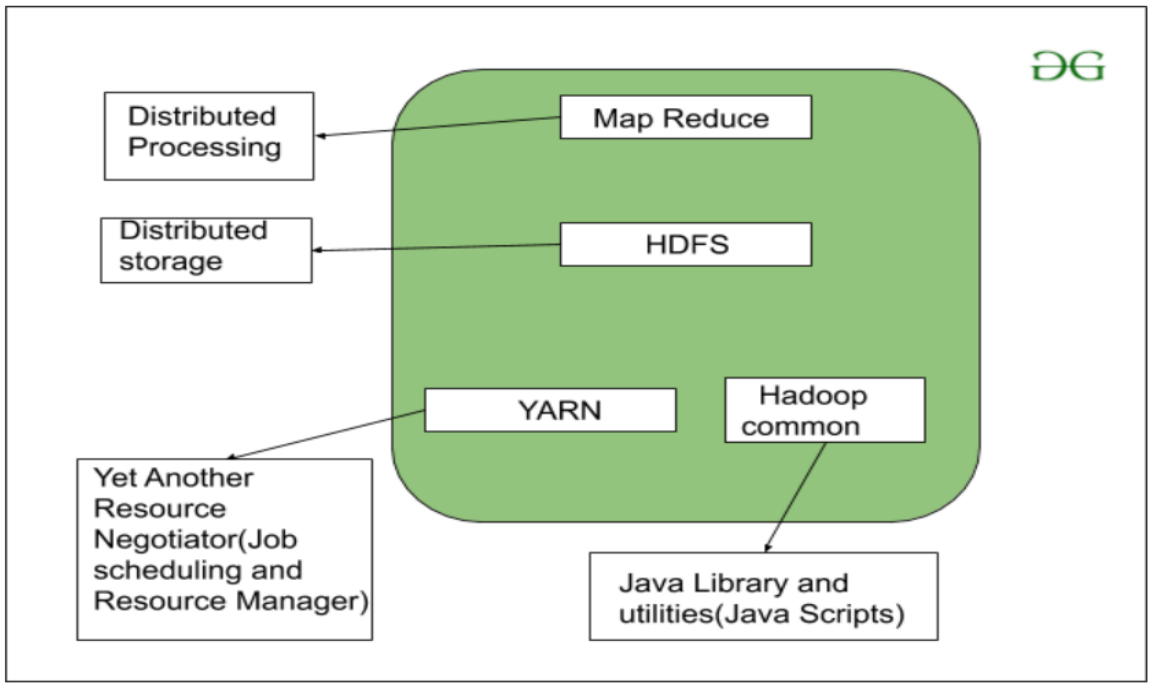
applications instead of concentrating on a distributed environment of the application.

- **Features of Zookeeper**
- Zookeeper acts fast enough with workloads where reads to data are more common than writes.
- Zookeeper acts as a disciplined one because it maintains a record of all transactions.

COMPONENTS OF HADOOP:

Hadoop is a framework that uses distributed storage and parallel processing to store and manage big data. It is the software most used by data analysts to handle big data. Below These are the Core Components of Hadoop:

1. Hadoop **HDFS** - Hadoop Distributed File System (HDFS) is the **storage unit.**
2. Hadoop **MapReduce** - Hadoop MapReduce is the **processing unit.**
3. Hadoop **YARN** - Yet Another Resource Negotiator (YARN) is a **resource management unit.**
4. Hadoop Common – Java Libraries & Utilities



HDFS :

- Hadoop Distributed File System (HDFS): HDFS is a distributed file system designed to store large datasets across multiple machines.
- It divides files into blocks and replicates them across the cluster to ensure fault tolerance and data availability.
- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
  Name node
  Data Node
- **Name Node** is the prime node which contains **metadata** (data about data) requiring comparatively fewer resources than the **data nodes** that stores the **actual data**. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.

- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.
- HDFS supports high throughput data access, making it suitable for applications that require large sequential reads.

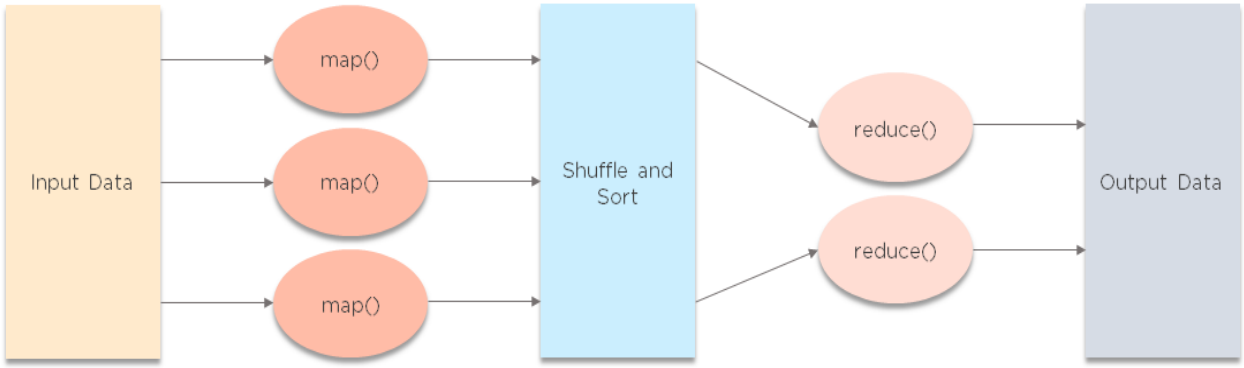
H Base

Hbase is considered as a Hadoop database, because it is scalable, distributed, and because NoSQL database that runs on top of Hadoop. Apache HBase is designed to store the structured data on table format which has millions of columns and billions of rows. HBase gives access to get the real-time data to read or write on HDFS.

- HBase is an open source, NoSQL database.
- It is featured after Google's big table, which is considered as a distributed storage system designed to handle big data sets.
- It has a unique feature to support all types of data. With this feature, it plays a crucial role in handling various types of data in Hadoop.

MAP REDUCE:

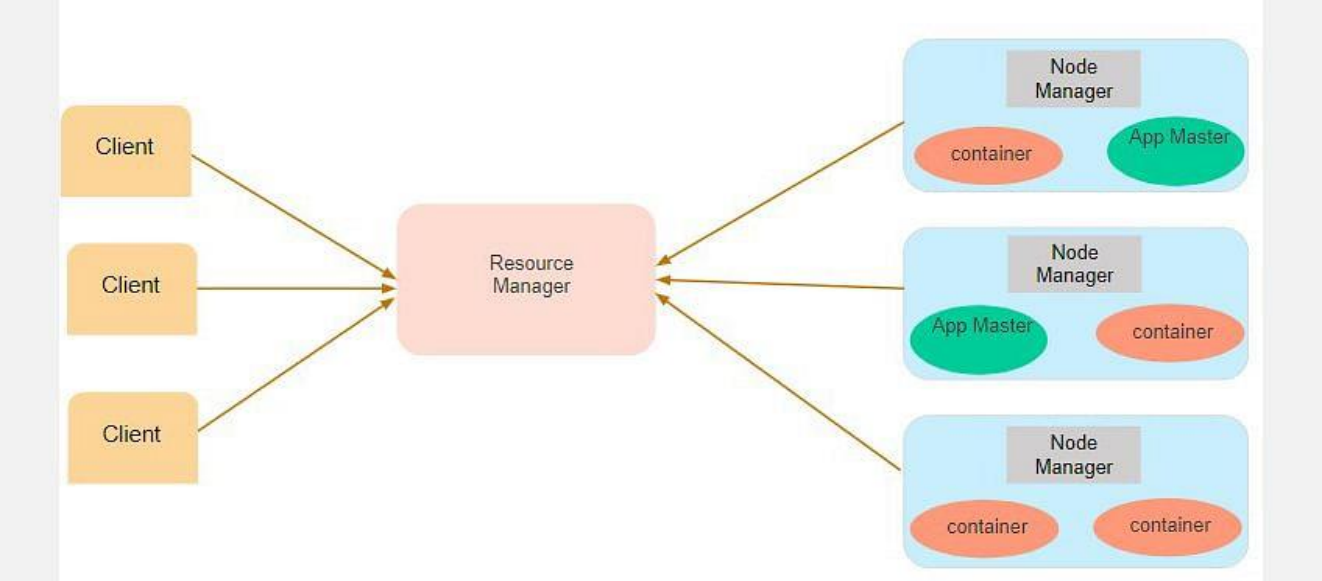
- MapReduce Programming Model: MapReduce is a programming paradigm or framework that is used for processing the large datasets in parallel across a distributed cluster.
- It comprises two main phases: the "**Map" phase** & **"Reduce" phase**,
- ***MapPhase:*** performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
- ***Reduce Phase:*** as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.



YARN (Yet Another Resource Negotiator):

- YARN is a resource management platform that manages and allocates resources across the cluster, enabling different processing frameworks to run concurrently on the same cluster.
- Hadoop YARN acts like an OS to Hadoop. It is a file system that is built on top of HDFS.
- It is responsible for managing cluster resources to make sure you don't overload one machine.
- It performs job scheduling to make sure that the jobs are scheduled in the right place.
- It Consists of three major components i.e.

  Resource Manager
  Nodes Manager
  Application Manager
- **Resource manager** has the privilege of allocating resources for the applications in a system.
- **Node managers** work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager.
- **Application manager** works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.



Example:

- Suppose a client machine wants to do a query or fetch some code for data analysis. This job request goes to the resource manager (Hadoop Yarn), which is responsible for resource allocation and management.
- In the node section, each of the nodes has its node managers.
- These node managers manage the nodes and monitor the resource usage in the node.
- The containers contain a collection of physical resources, which could be RAM, CPU, or hard drives.
- Whenever a job request comes in, the app master requests the container from the node manager. Once the node manager gets the resource, it goes back to the Resource Manager.

Hadoop Distributed File System(HDFS)-(Working)

- HDFS(Hadoop Distributed File System) is the primary or major component of Hadoop ecosystem and is responsible for storing large datasets across multiple machines.

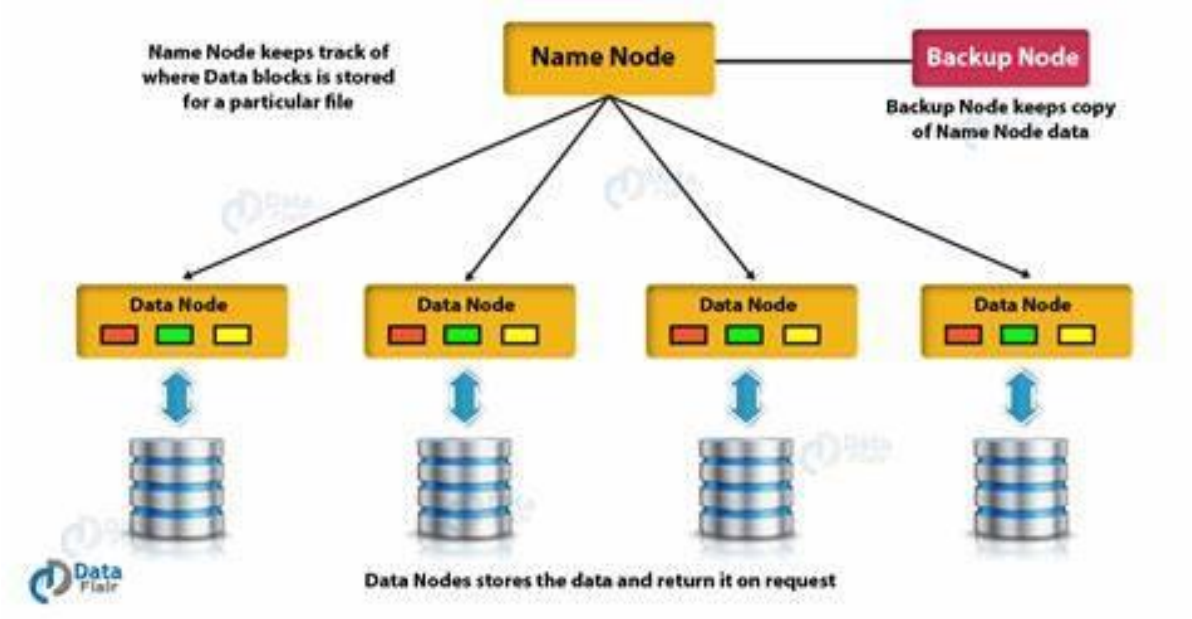
- It divides files into blocks and replicates them across the cluster to ensure fault tolerance and data availability.
- HDFS supports high throughput data access, making it suitable for applications that require large sequential reads.
- HDFS consists of two core components i.e.
    1. **Name node(Master node)**
    2. **Data Node(Slave node)**

Name Node - Name Node is the master Node.

- There is only one active Name Node.
- It manages the Data Nodes and stores all the **metadata.**
- These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

Data Node - Data Node is the **slave** daemon.

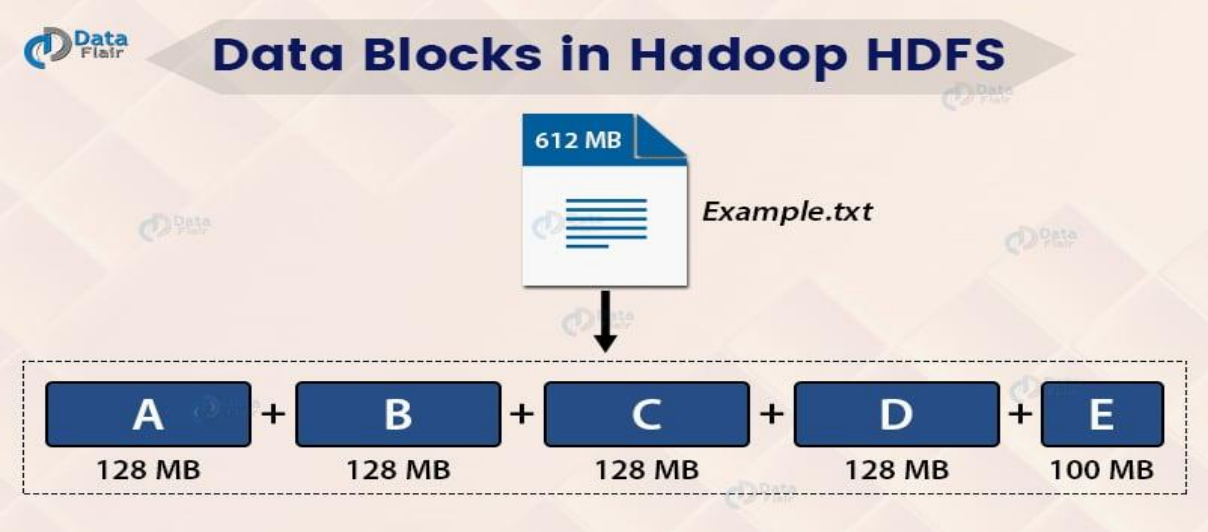
- There can be multiple DataNodes.
- It stores the **actual data**
- The data nodes read, write, process, and replicate the data.
- They also send signals, known as heartbeats, to the name node. These heartbeats show the status of the data node.



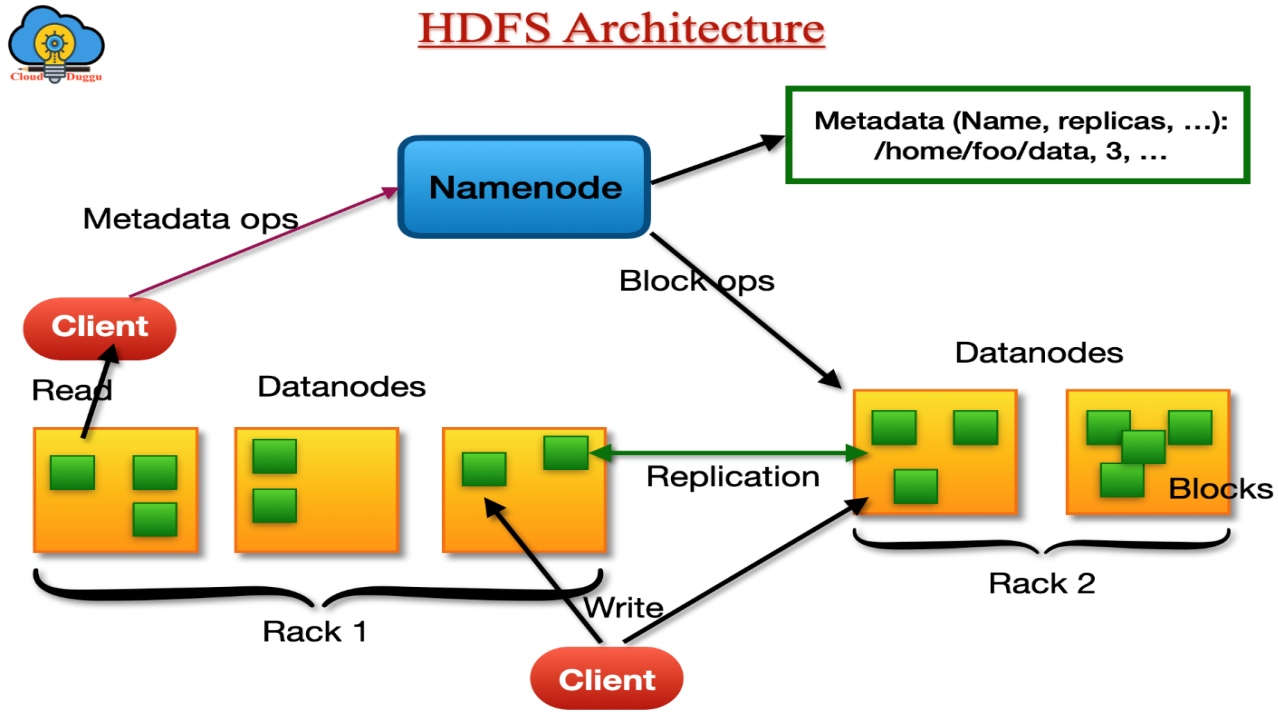
- **HDFS splits** the data into multiple blocks, defaulting to a maximum of 128 MB. The default block size can be changed depending on the processing speed and the data distribution. Let's look at the example below:

- As seen from the above image, we have 300 MB of data. This is broken down into 128 MB, 128 MB, and 44 MB. The final block handles the remaining needed storage space, so it doesn't have to be sized at 128 MB. This is how data gets stored in a distributed manner in HDFS.

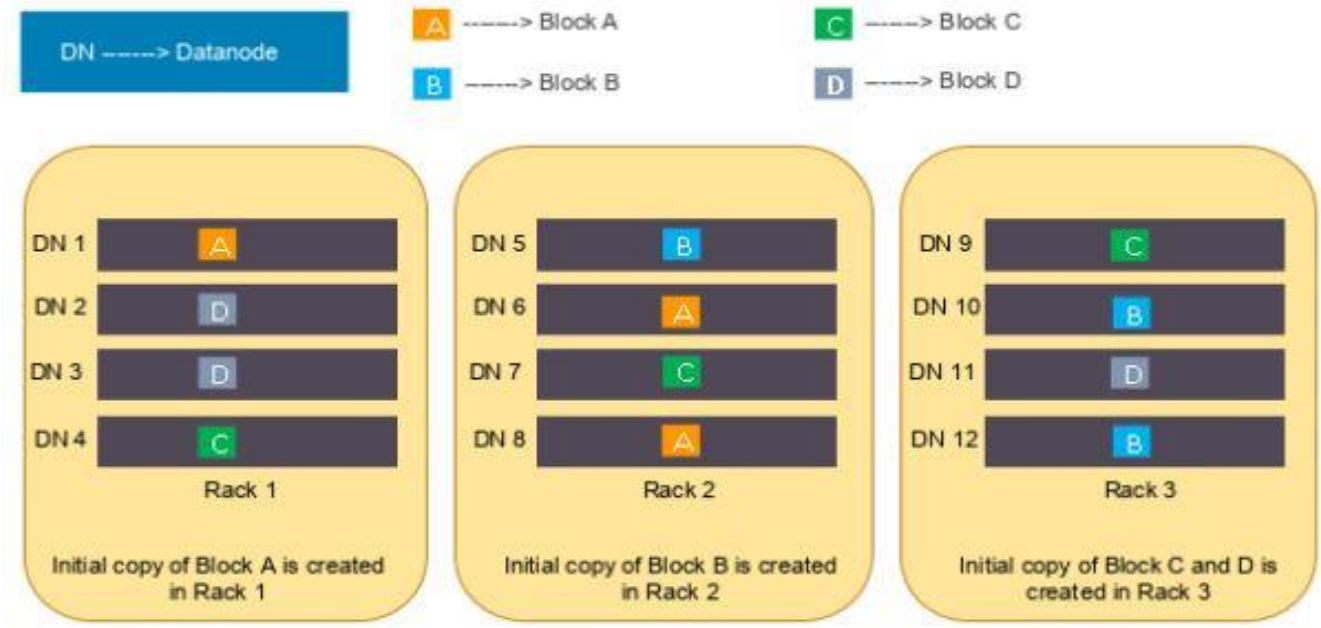
- Example 2:



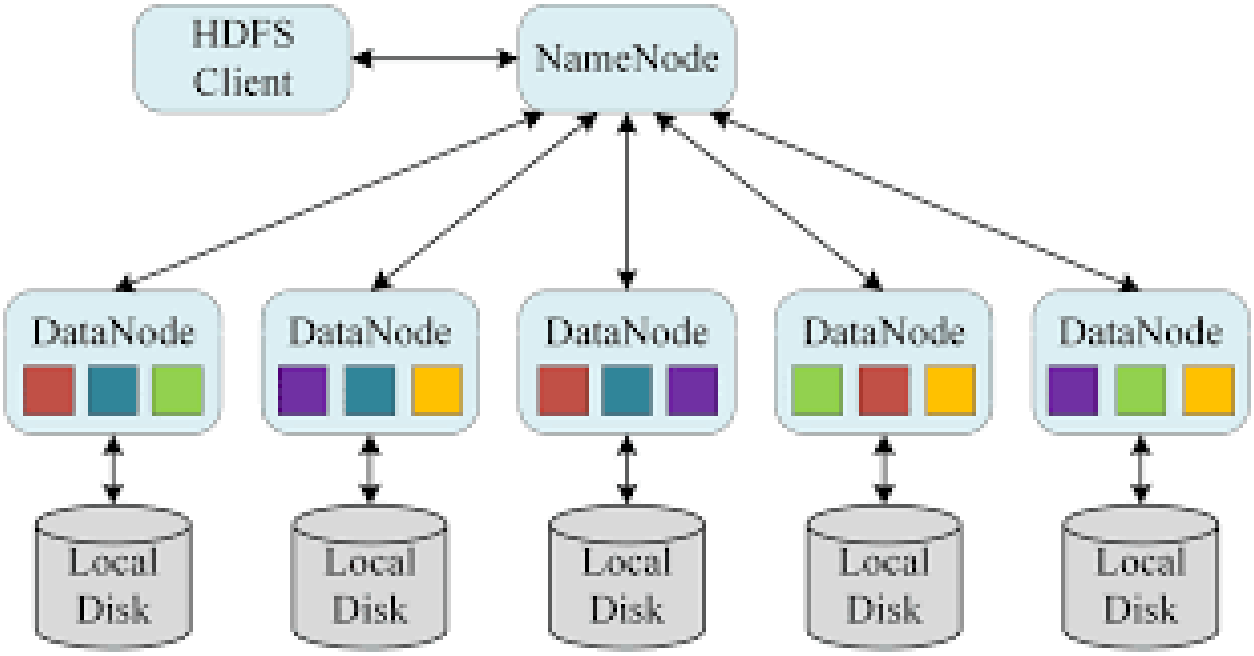
Architecture and Design of HDFS in detail



Example of Data Nodes containing Data blocks:



Splitting of Data Nodes:



The HDFS architecture consists of the following components:

- **Name Node**
- **Data Node**
- **Secondary Name Node**
- **Checkpoint Node**
- **Rack Awareness**
- **Replication**

- The **NameNode** is responsible for the following tasks:

  - The NameNode is the master server in HDFS & Manages the file system namespace
  - It executes the file system namespace operations like opening, re-naming, and closing files and directories.
  - NameNode manages and maintains the DataNodes.
  - It determines the mapping of blocks of a file to DataNodes.
  - NameNode records each change made to the file system namespace.
  - It Stores metadata about files and directories
  - Tracking the location of data blocks
  - Handling client requests for files
  - It executes the file system namespace operations like opening, renaming, and closing files and directories.
  - NameNode manages and maintains the DataNodes.
  - It determines the mapping of blocks of a file to DataNodes.

- NameNode records each change made to the file system namespace.
- Rebalancing data blocks if a DataNode fails
- Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

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

- **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.
- **EditLogs:** The EditLogs file keeps track of what modifications have been made to the files of the filesystem.

DataNodes : Data nodes are known as Slave nodes .

- DataNodes are mainly utilized for storing the data in a Hadoop cluster, the number of DataNodes can be from 1 to 500 or even more than that. The more number of DataNode, the Hadoop cluster will be able to store more data. So it is advised that the DataNode should have High storing capacity to store a large number of file blocks.

- The **DataNodes** are responsible for the following tasks:

  - The DataNodes are the slave servers in HDFS.
  - It Stores the Actual data blocks
  - Serving read and write requests for files
  - DataNode is responsible for serving the client read/write requests.
  - Based on the instruction from the NameNode, DataNodes performs block creation, replication, and deletion.

- DataNodes send a heartbeat to NameNode to report the health of HDFS.
- DataNodes also sends block reports to NameNode to report the list of blocks it contains.
- 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.
- Rebalancing data blocks if a DataNode fails

➢ The **secondary NameNode** is responsible for the following tasks:

- The secondary NameNode is a hot standby for the NameNode
- Periodically synchronizing its metadata with the NameNode
- Serving read and write requests for files if the NameNode fails
- Taking over as the NameNode if the Primary NameNode fails
- Secondary NameNode works as a helper node to primary NameNode but doesn't replace primary NameNode.
- When the NameNode starts, the NameNode merges the Fsimage and edit logs file to restore the current file system namespace. Since the NameNode runs continuously for a long time without any restart, the size of edit logs becomes too large. This will result in a long restart time for NameNode.
- Secondary NameNode solves this issue.
- Secondary NameNode downloads the Fsimage file and edit logs file from NameNode.
- It periodically applies edit logs to Fsimage and refreshes the edit logs. The updated Fsimage is then sent to the NameNode

so that NameNode doesn't have to re-apply the edit log records during its restart. This keeps the edit log size small and reduces the NameNode restart time.

- The **checkpoint node** is responsible for the following tasks:

The Checkpoint node is a node that periodically creates checkpoints of the namespace.Checkpoint Node in Hadoop first downloads Fsimage and edits from the Active Namenode. Then it merges them (Fsimage and edits) locally, and at last, it uploads the new image back to the active NameNode.

- Storing a copy of the NameNode's metadata
- The checkpoint node is not a mandatory component of HDFS, but it is recommended for production deployments
- Restoring the NameNode's metadata if the NameNode fails

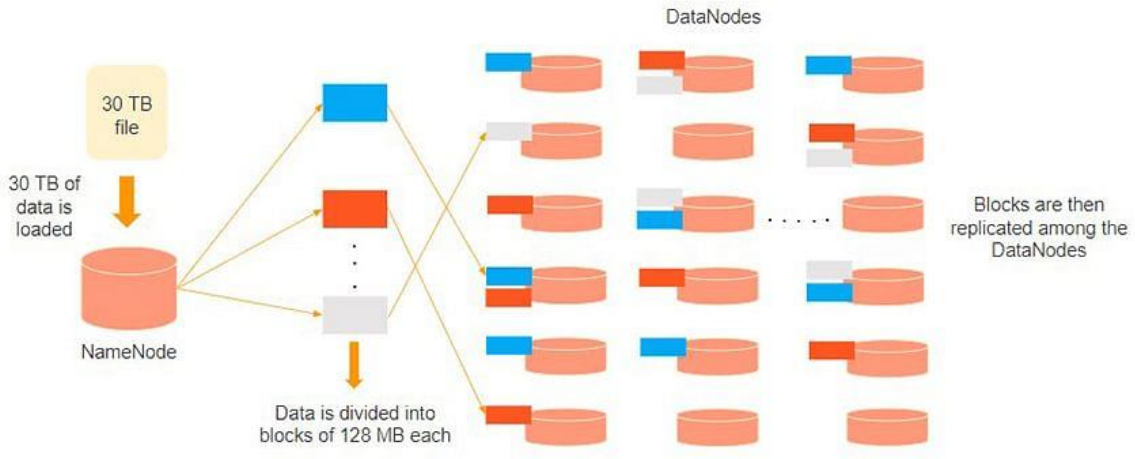
Replication In HDFS :

- Replication ensures the availability of the data. Replication is making a copy of something and the number of times you make a copy of that particular thing can be expressed as it's Replication Factor.

- In Hadoop, HDFS stores replicas of a block on multiple DataNodes based on the replication factor.
- If the replication factor is 3, then three copies of a block get stored on different DataNodes. So if one DataNode containing the data block fails, then the block is accessible from the other DataNode containing a replica of the block.

- If we are storing a file of 128 Mb and the replication factor is 3, then (3*128=384) 384 Mb of disk space is occupied for a file as three copies of a block get stored.

- Replication of the data is performed three times by default. If a commodity machine fails, you can replace it with a new machine that has the same data.



- Consider that 30TB of data is loaded into the name node. The name node distributes it across the data nodes, and this data is replicated among the data notes. You can see in the image above that the blue, grey, and red data are replicated among the three data nodes.

Rack Awareness:

- In a large Hadoop cluster, there are multiple racks.

- Rack is the collection of around 40-50 machines (DataNodes) connected using the same network switch. If the network goes down, the whole rack will be unavailable.
- Rack Awareness is the concept of choosing the closest node based on the rack information.

- To ensure that all the replicas of a block are not stored on the same rack or a single rack, NameNode follows a rack awareness algorithm to store replicas and provide latency and fault tolerance. Suppose if the replication factor is 3, then according to the rack awareness algorithm:

- The first replica will get stored on the local rack.
- The second replica will get stored on the other DataNode in the same rack.
- The third replica will get stored on a different rack.
- Each rack consists of DataNodes. Communication between the DataNodes on the same rack is more efficient as compared to the communication between DataNodes residing on different racks.

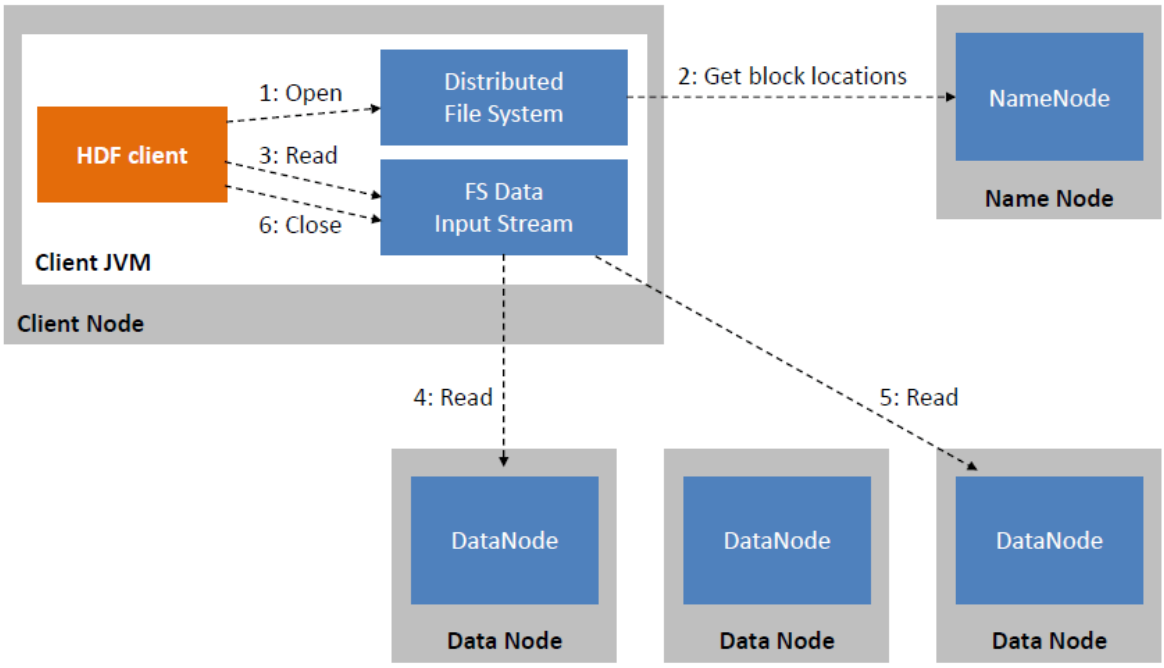
- To reduce the network traffic during file **read/write**, NameNode chooses the closest DataNode for serving the client read/write request.

- NameNode maintains **rack ids** of each DataNode to achieve this rack information.

- This concept of choosing the closest DataNode based on the rack information is known as **Rack Awareness**.

Blocks: A Block is the minimum amount of data that it can read or write. HDFS blocks are 128 MB by default and this is configurable. Files n HDFS are broken into block-sized chunks,which are stored as independent .

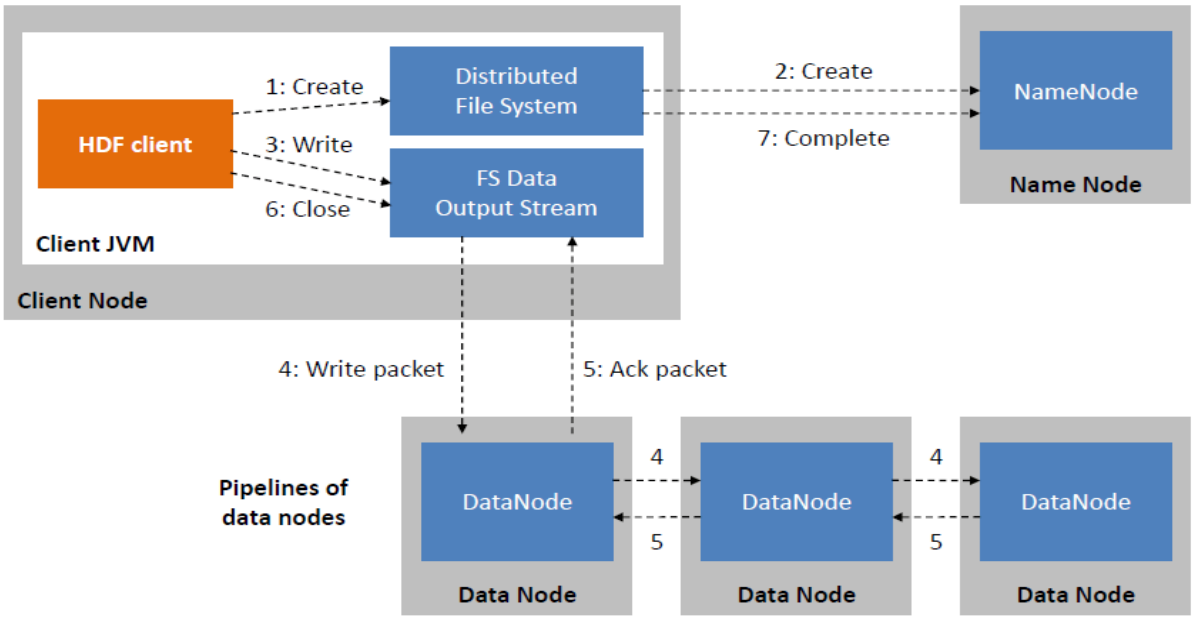
Client : READ Data from HDFS:



- The client opens a connection to the NameNode by using the Hadoop DistributedFileSystem API. The Hadoop DistributedFileSystem API provides methods for interacting with the NameNode and DataNodes.
- A client initiates read request by calling **'open()'** method of FileSystem object; it is an object of type **DistributedFileSystem**.
- The client requests the location of the data blocks by calling the getFileBlockLocations() method on the Hadoop DistributedFileSystem API. The getFileBlockLocations() method returns an array of BlockLocation objects. Each BlockLocation object contains the address of the DataNode that is storing a data block.
- This object connects to namenode using RPC and gets metadata information such as the locations of the blocks of the file. Please note that these addresses are of first few blocks of a file.

- In response to this metadata request, addresses of the DataNodes having a copy of that block is returned back.
- The NameNode returns the locations of the data blocks to the client. The NameNode keeps track of the location of all the data blocks in HDFS. It uses this information to answer requests from clients about the location of data blocks.
- Once addresses of DataNodes are received, an object of type **FSDataInputStream** is returned to the client.
- **FSDataInputStream** contains **DFSInputStream** which takes care of interactions with DataNode and NameNode.
- In step 4 shown in the above diagram, a client invokes **'read()'** method which causes **DFSInputStream** to establish a connection with the first DataNode with the first block of a file.
- Data is read in the form of streams wherein client invokes **'read()'** method repeatedly.
- The client reassembles the data blocks into a file. The client uses the Hadoop InputStream API to reassemble the data blocks into a file.
- The Hadoop client can read and write data from and to HDFS in parallel. This is because the client can connect to multiple DataNodes at the same time. This allows the client to read and write data from and to HDFS very quickly.
- This process of **read()** operation continues till it reaches the end of block.
- Once the end of a block is reached, DFSInputStream closes the connection and moves on to locate the next DataNode for the next block
- Once a client has done with the reading, it calls **a close()** method.

Client: WRITE Data to HDFS:



- The client opens a connection to the NameNode by using the Hadoop DistributedFileSystem API. The Hadoop DistributedFileSystem API provides methods for interacting with the NameNode and DataNodes.
- A client initiates write operation by calling 'create()' method of DistributedFileSystem object which creates a new file – Step no. 1 in the above diagram.

- DistributedFileSystem object connects to the NameNode using RPC call and initiates new file creation. However, this file creates operation does not associate any blocks with the file. It is the responsibility of NameNode to verify that the file (which is being created) does not exist already and a client has correct permissions to create a new file. If a file already exists or client does not have sufficient permission to create a new file, then **IOException** is

thrown to the client. Otherwise, the operation succeeds and a new record for the file is created by the NameNode.

- Once a new record in NameNode is created, an object of type FSDataOutputStream is returned to the client. A client uses it to write data into the HDFS. Data write method is invoked (step 3 in the diagram).
- FSDataOutputStream contains DFSOutputStream object which looks after communication with DataNodes and NameNode. While the client continues writing data, **DFSOutputStream** continues creating packets with this data. These packets are enqueued into a queue which is called as **DataQueue**.
- There is one more component called **DataStreamer** which consumes this **DataQueue**. DataStreamer also asks NameNode for allocation of new blocks thereby picking desirable DataNodes to be used for replication.
- Now, the process of replication starts by creating a pipeline using DataNodes. In our case, we have chosen a replication level of 3 and hence there are 3 DataNodes in the pipeline.
- The DataStreamer pours packets into the first DataNode in the pipeline.
- Every DataNode in a pipeline stores packet received by it and forwards the same to the second DataNode in a pipeline.
- Another queue, ‘Ack Queue’ is maintained by DFSOutputStream to store packets which are waiting for acknowledgment from DataNodes.
- Once acknowledgment for a packet in the queue is received from all DataNodes in the pipeline, it is removed from the ‘Ack Queue’. In the event of any DataNode failure, packets from this queue are used to reinitiate the operation.

- After a client is done with the writing data, it calls a close() method (Step 9 in the diagram) Call to close(), results into flushing remaining data packets to the pipeline followed by waiting for acknowledgment.
- Once a final acknowledgment is received, NameNode is contacted to tell it that the file write operation is complete.

Advantages of HDFS:

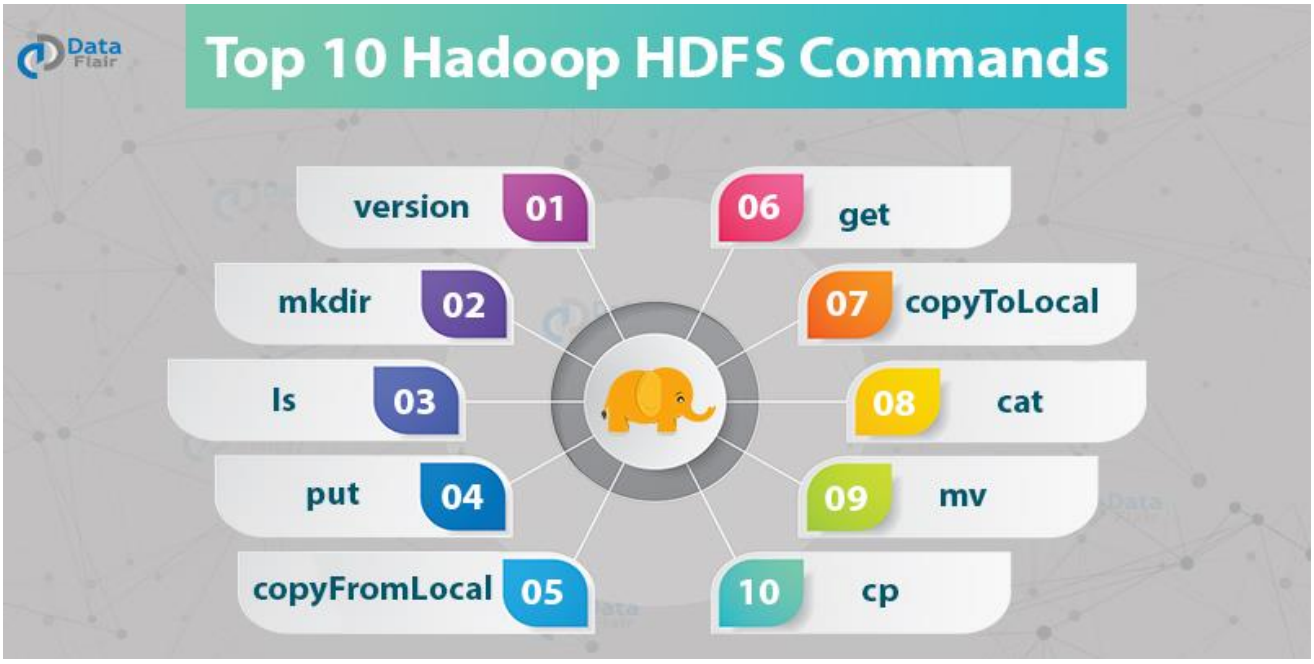
1. **Scalability:** HDFS is designed to store and process large volumes of data, making it highly scalable. It can easily handle petabytes and even exabytes of data.
2. **Fault tolerance:** HDFS is fault-tolerant, meaning that it can continue to function even in the event of hardware or software failures. It does this by replicating data across multiple nodes, ensuring that there is always a copy available.
3. **Cost-effective:** HDFS is an open-source technology that can be run on commodity hardware, making it a cost-effective solution for big data storage and procession

Disadvantages of HDFS:

1. **High latency:** HDFS has high latency due to the overhead of replicating data across multiple nodes. This can lead to slower performance when reading or writing data.
2. **Not suitable for small files:** HDFS is designed for storing and processing large files, and it is not optimized for small files. This means that it may not be the best choice for applications that deal with a large number of small files

Working with HDFS (Commands)

- Working with Hadoop Distributed File System (HDFS) involves using a set of commands to interact with the file system.
- These commands are executed through the Hadoop command-line interface (CLI) or other interfaces provided by Hadoop distributions.
- Before working with HDFS you need to Deploy Hadoop(installation). Below are some commonly used HDFS commands along with explanations of their functionality:
- FS relates to a generic file system
- hadoop fs: It is used when we are dealing with different file systems such as Local FS, HDFS etc.
- hdfs dfs: It is used when we are dealing for operations related to HDFS.



To check the Hadoop services are up and running use the following command:

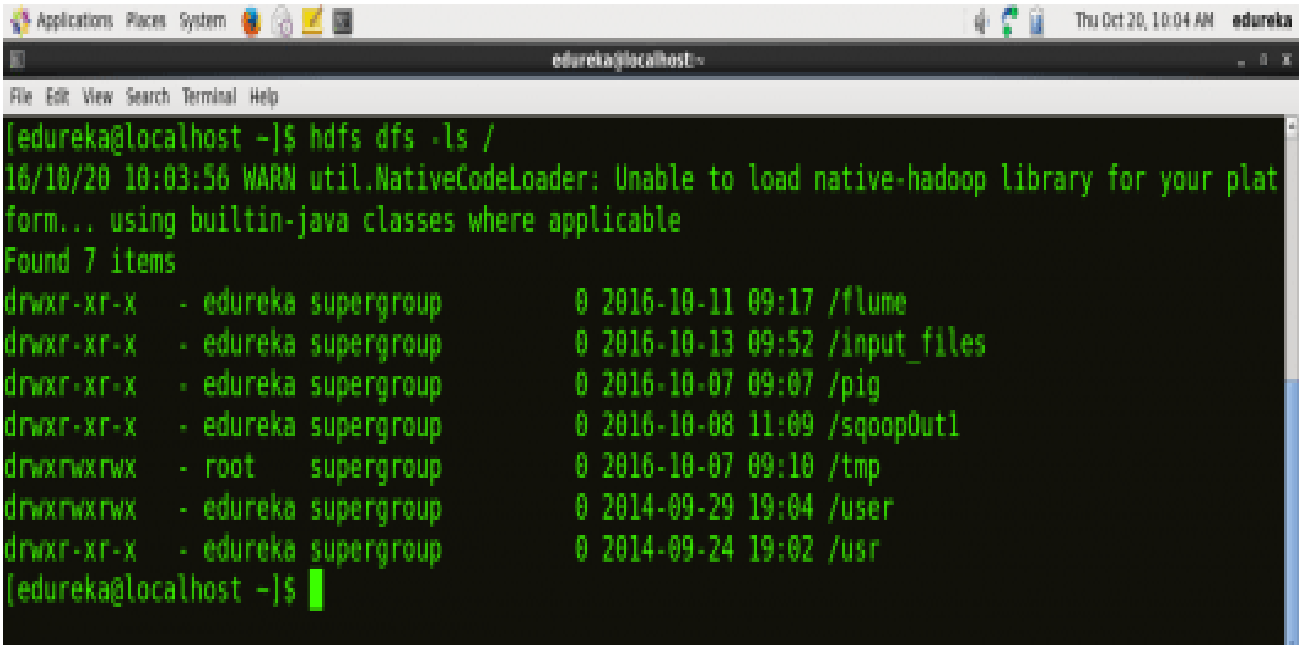
jps

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ jps
2546 SecondaryNameNode
2404 DataNode
2295 NameNode
2760 ResourceManager
2874 NodeManager
4251 Jps
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

Command To Start: $ start-dfs.sh

- **LS: It is a Command to display the list of Files and Directories in HDFS.**

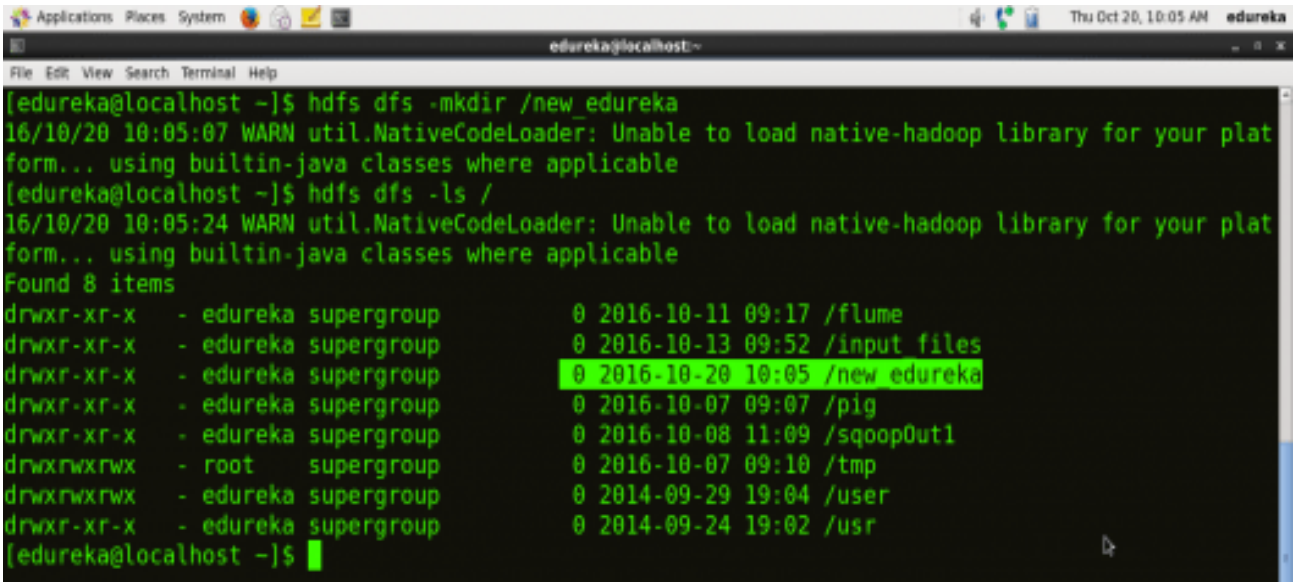
Syntax : **hdfs dfs –ls /**



- **Mkdir:** **HDFS Command to create the directory in HDFS.**

Syntax : **hdfs dfs –mkdir /directory_name**

Example : **hdfs dfs –mkdir /new_edureka**

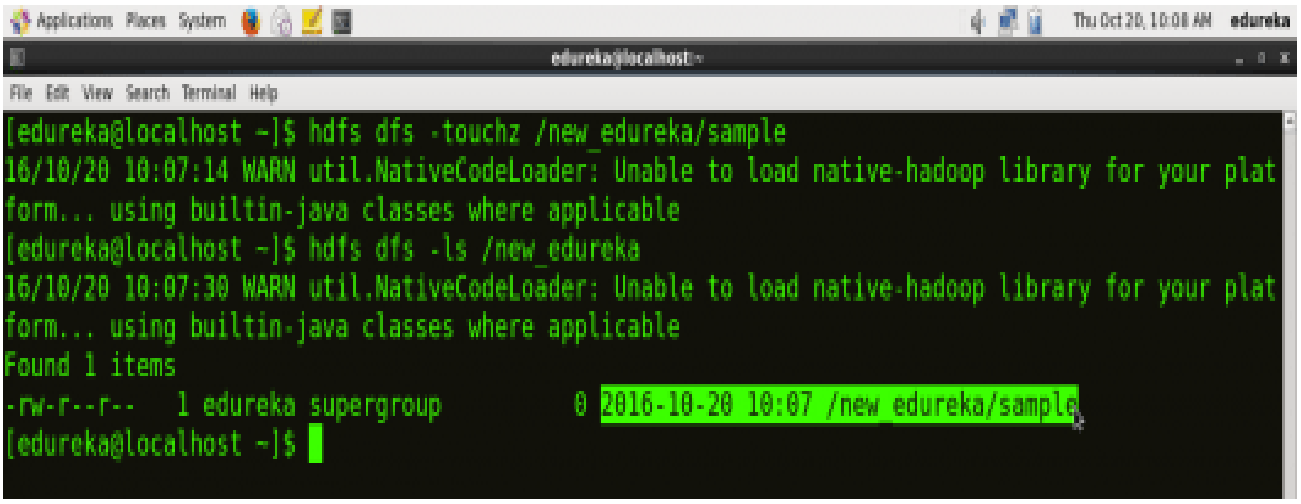


Note: Here we are trying to create a directory named "new_edureka" in HDFS.

- **Touchz:** **HDFS Command to create a Empty file in HDFS with file size 0 bytes.**

Syntax: **: hdfs dfs –touchz /directory/filename**

Example: **hdfs dfs –touchz /new_edureka/sample**

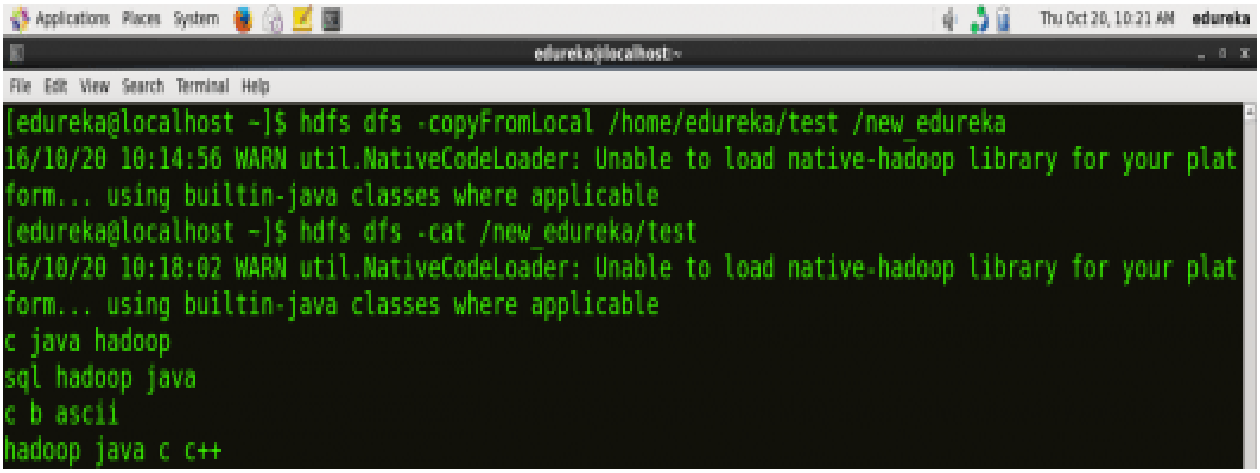


Note: Here we are trying to create a file named "sample" in the directory "new_edureka" of hdfs with file size 0 bytes.

- **copyFromLocal Or put:** HDFS Command to copy the file from a Local file system to HDFS.

Syntax: **hdfs dfs -copyFromLocal <localsrc> <hdfs destination>**

Example: **hdfs dfs copyFromLocal/home/edureka/test/new_edureka**

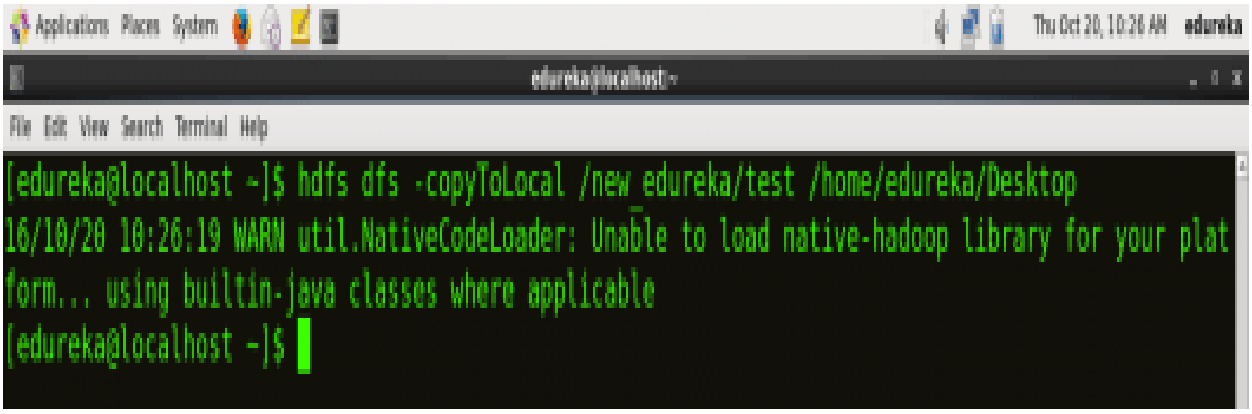


Note: Here the test is the file present in the local directory /home/edureka and after the command gets executed the test file will be copied in /new_edureka directory of HDFS.

- **copyToLocal or get: HDFS Command to copy the file from HDFS to Local File System.**

Syntax: **hdfs dfs -copyToLocal <hdfs source> <localdst>**

Example: **hdfs dfs copyToLocal/new_edureka/test/home/edureka**

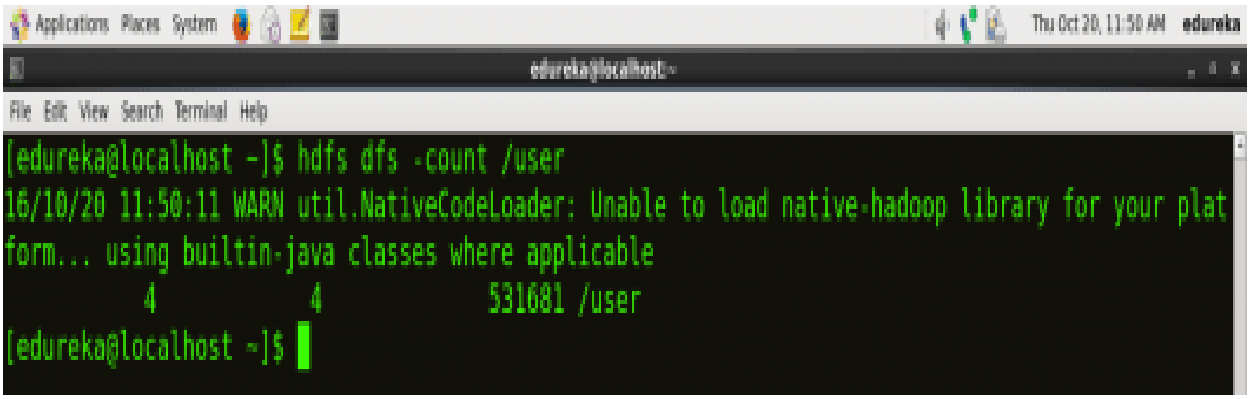


Note: Here test is a file present in the new_edureka directory of HDFS and after the command gets executed the test file will be copied to local directory /home/edureka

- **Count:** HDFS Command to count the number of directories, files, and bytes under the paths that match the specified file pattern.

Syntax: **hdfs dfs -count <path>**

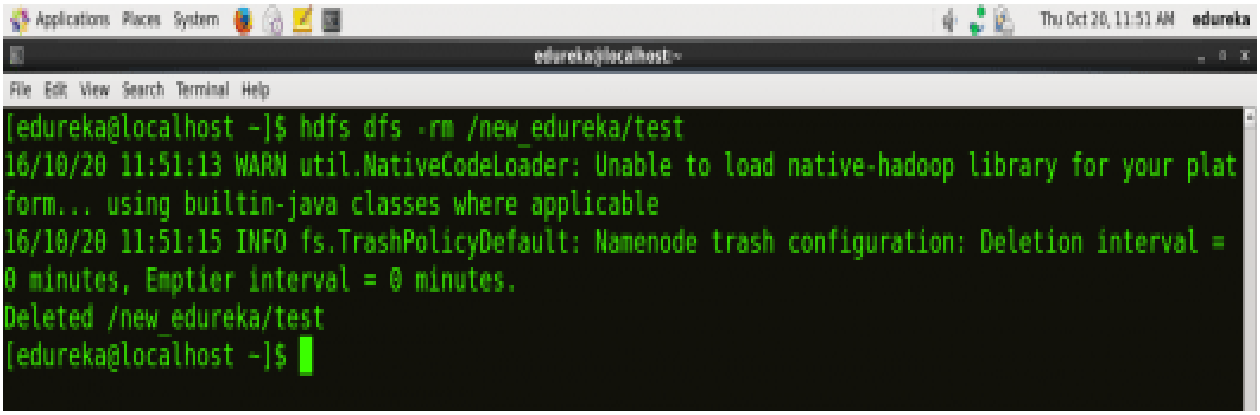
Example: **hdfs dfs –count /user**



- **rm:** HDFS Command to remove the file from HDFS.

Syntax: **hdfs dfs –rm <path>**

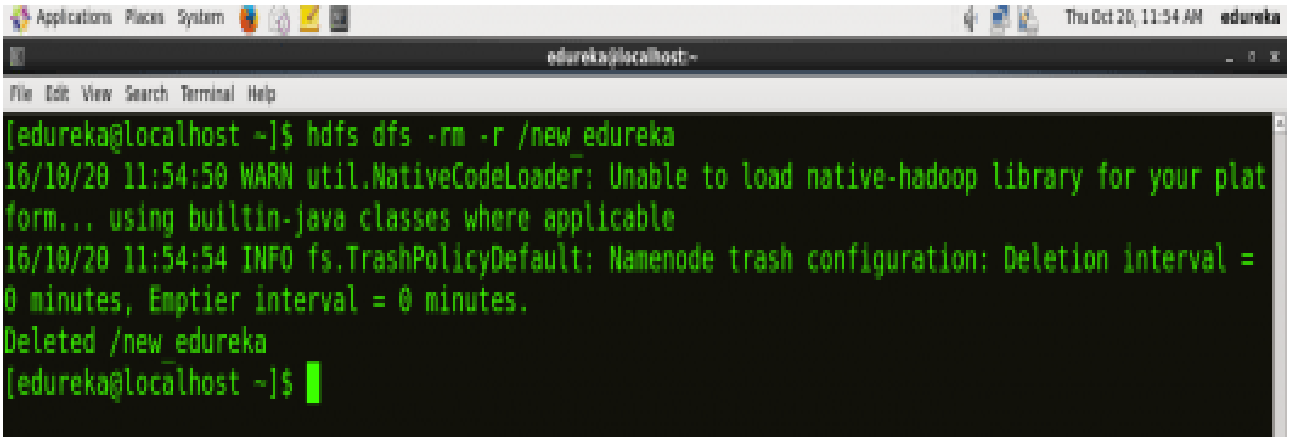
Example: **hdfs dfs –rm /new_edureka/test**



- **rm -r:** HDFS Command to remove the entire directory and all of its content from HDFS.

Syntax: **hdfs dfs -rm -r <path>**

Example: **hdfs dfs -rm -r /new_edureka**



- **cp:** HDFS Command to copy files from source to destination. This command allows multiple sources as well, in which case the destination must be a directory.

  Syntax: **hdfs dfs -cp <src> <dest>**

  Example: **hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2**

- **mv:** HDFS Command to move files from source to destination. This command allows multiple sources as well, in which case the destination needs to be a directory.

  Syntax: **hdfs dfs -mv <src> <dest>**

  Example:: **hdfs dfs -mv /user/hadoop/file1 /user/hadoop/file2**

- **rmdir: HDFS Command to remove the directory.**

  Syntax: **hdfs dfs -rmdir <path>**

  Example: **hdfs dfs –rmdir /user/Hadoop**

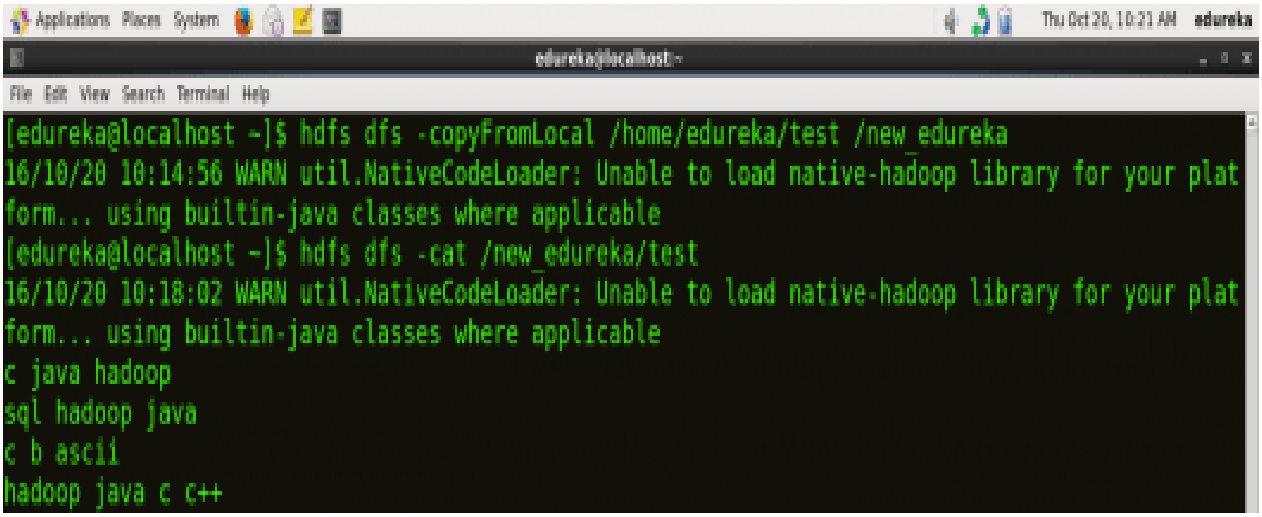
rmr: This command deletes a file from HDFS *recursively*. It is very useful command when you want to delete a *non-empty directory*.

bin/hdfs dfs -rmr <filename/directoryName>

- **cat:** HDFS Command that reads a file on HDFS and prints the content of that file to the standard output.

  Syntax: **hdfs dfs –cat /path/to/file_in_hdfs**

  Example:: **hdfs dfs –cat /new_edureka/test**



These are just a few examples of the many HDFS commands available. Note that the specific commands and syntax might vary depending on the Hadoop distribution and version .