KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

KG REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

B. Tech III Year II Semester **Academic Year: 2023-2024**

SUBJECT: BIG DATA ANALYTICTS **SUBJECT CODE: KG21CD605** **REGULATION: KGR21**

UNIT-1 NOTES

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.

Introduction to Big Data:

What is 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.

What is an Example of Big Data?

Following are some of the Big Data examples-

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, 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 data:

The digital data is divided into three types

1.**unstructured data:** This is the data which does not conform to a data model or is not in a form which can be used easily by a computer program.

Forexample:

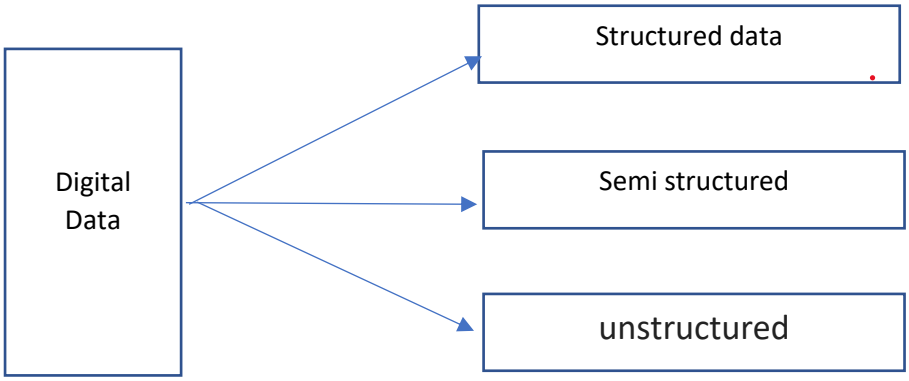
chatrooms,memos,power point presentation,images,videos,letters,researches,white papers,body of an email.

Semi-structured data:

This is the data which does not conform to a data model but has some structure.how ever it is not in the form which can be used easily by a computer program.

Examples:E-mails,XML,markup languages,HTML

Structured data: This is the data which is in the organized form. i.e in the form of rows and columns and can be easily used by a computer programs. Relationships exist between the entities of data, such as classes and their objects.Data stored in data bases is an example of structured data.



Structured data:

The data when it conforms to the schema/structure we say it is structured data.

- **Structured data** is generally tabular data that is represented by columns and rows in a database.
- Databases that hold tables in this form are called ***relational databases***.
- The mathematical term *"relation"* specify to a formed set of data held as a table.
- In structured data, all row in a table has the same set of columns.

- SQL (Structured Query Language) programming language used for structured data.

Table 1.1 A relation/table with rows and columns

	Column 1	Column 2	Column 3	Column 4
Row 1				

Table 1.2 Schema of an "Employee" table in a RDBMS such as Oracle

Column Name	Data Type	Constraints
EmpNo	Varchar(10)	PRIMARY KEY
EmpName	Varchar(50)	
Designation	Varchar(25)	NOT NULL
DeptNo	Varchar(5)	
ContactNo	Varchar(10)	NOT NULL

Table 1.3 Sample records in the "Employee" table

EmpNo	EmpName	Designation	DeptNo	ContactNo
E101	Allen	Software Engineer	D1	0999999999
E102	Simon	Consultant	D1	0777777777

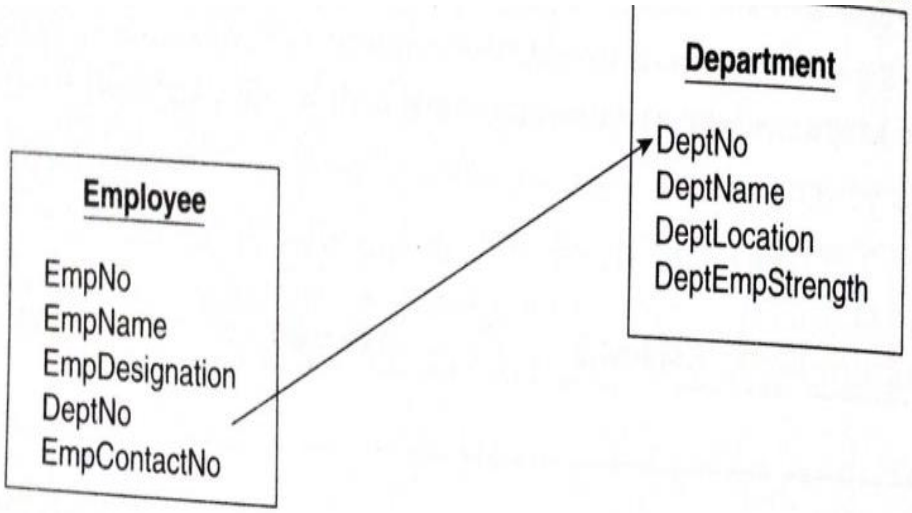


Figure 1.3 Relationship between "Employee" and "Department" tables.

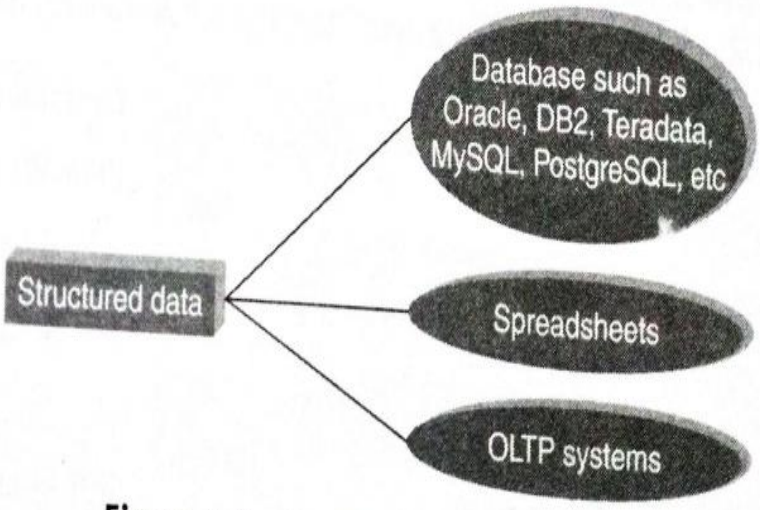


Figure 1.4 Sources of structured data.

1.1.1.2 Ease of Working with Structured Data

Structured data provides the ease of working with it. Refer Figure 1.5. The ease is with respect to the following:

1. **Insert/update/delete:** The Data Manipulation Language (DML) operations provide the required ease with data input, storage, access, process, analysis, etc.
2. **Security:** How does one ensure the security of information? There are available staunch encryption and tokenization solutions to warrant the security of information throughout its lifecycle. Organizations are able to retain control and maintain compliance adherence by ensuring that only authorized individuals are able to decrypt and view sensitive information.
3. **Indexing:** An index is a data structure that speeds up the data retrieval operations (primarily the SELECT DML statement) at the cost of additional writes and storage space, but the benefits that ensue in search operation are worth the additional writes and storage space.
4. **Scalability:** The storage and processing capabilities of the traditional RDBMS can be easily scaled up by increasing the horsepower of the database server (increasing the primary and secondary or peripheral storage capacity, processing capacity of the processor, etc.).
5. **Transaction processing:** RDBMS has support for Atomicity, Consistency, Isolation, and Durability (ACID) properties of transaction. Given next is a quick explanation of the ACID properties:
   - ***Atomicity:*** A transaction is atomic, means that either it happens in its entirety or none of it at all.
   - ***Consistency:*** The database moves from one consistent state to another consistent state. In other words, if the same piece of information is stored at two or more places, they are in complete agreement.
   - ***Isolation:*** The resource allocation to the transaction happens such that the transaction gets the impression that it is the only transaction happening in isolation.
   - ***Durability:*** All changes made to the database during a transaction are permanent and that accounts for the durability of the transaction.

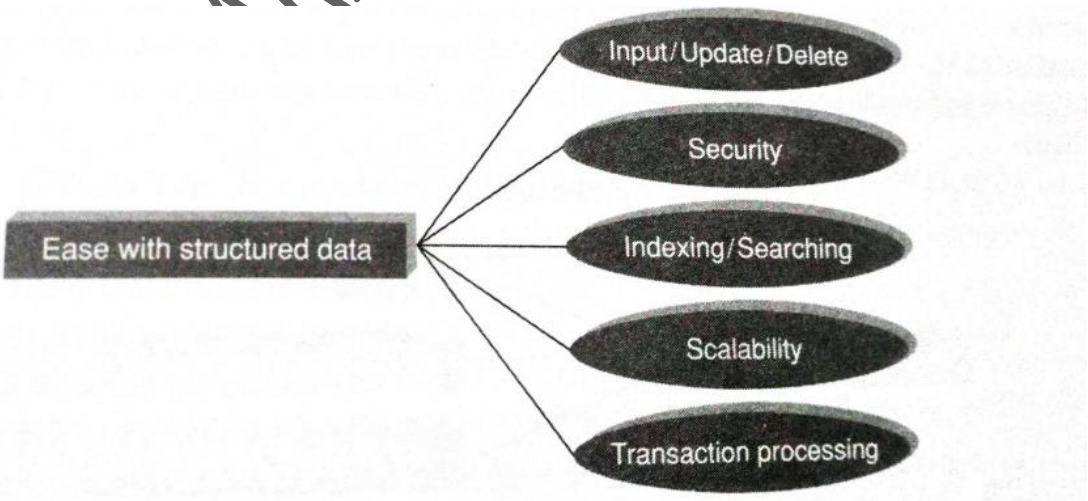


Figure 1.5 Ease of working with structured data.

Sources of structured data:

The sources from where the data is generated is RDBMS,oracle,DB2,Microsoft Sql server,Teradata,Mysql and OLTP systems.

Semi structured data:

The semi structured data is also referred to as self describing tags.

It uses tags to segregate the semantic elements.

Sources of semi structured data:

The sources of semi structured are

XML-Extensible mark up language

JSON-Java script object Notation.

An example of HTML as follows

<html>

<head>

<title>place your title here

</head>

<body bgcolor=”FFFFFF”

</html>

Sample JSON document

{

_id:9

Booktitle:Fundamentals of Business Analyticts’

Author Name:”Seema acharya”

Publisher:”Wiley India”

Year of Publication:”2011”

}

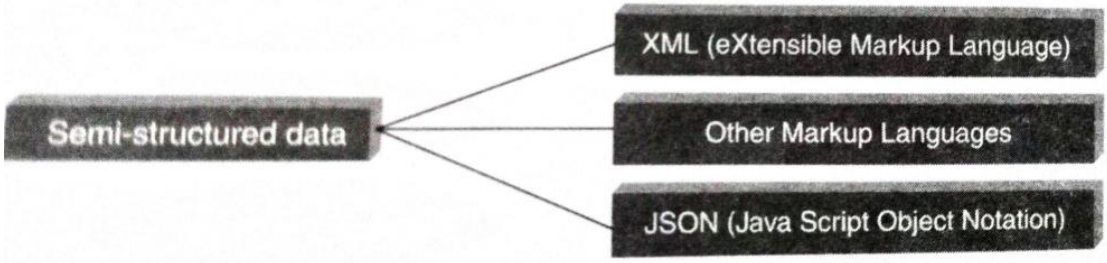


Figure 1.7 Sources of semi-structured data.

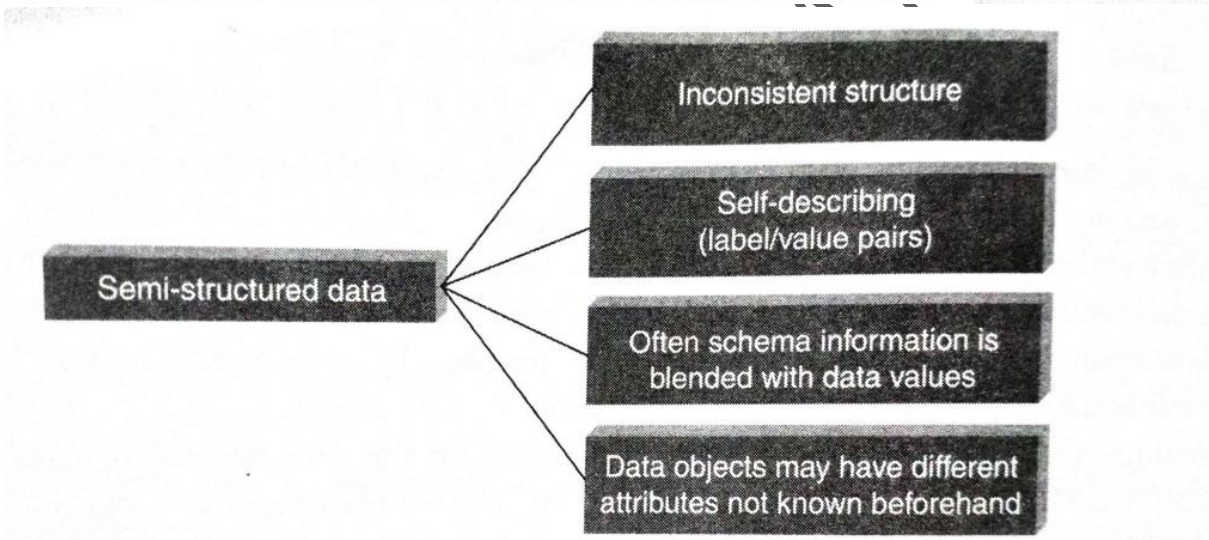


Figure 1.6 Characteristics of semi-structured data.

Unstructured data:

Unstructured data does not conform to any predefined model.

Sources of Unstructured data:

Table 1.4 Few examples of disparate unstructured data

Twitter message	Feeling miffed ☹. Victim of twishing.
Facebook post	LOL. C ya. BFN
Log files	127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I; Nav)"
Email	Hey Joan, possible to send across the first cut on the Hadoop chapter by Friday EOD or maybe we can meet up over a cup of coffee. Best regards, Tom

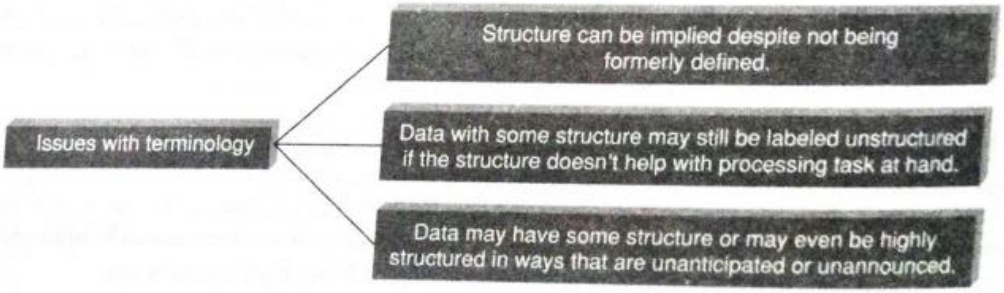


Figure 1.9 Issues with terminology of unstructured data.

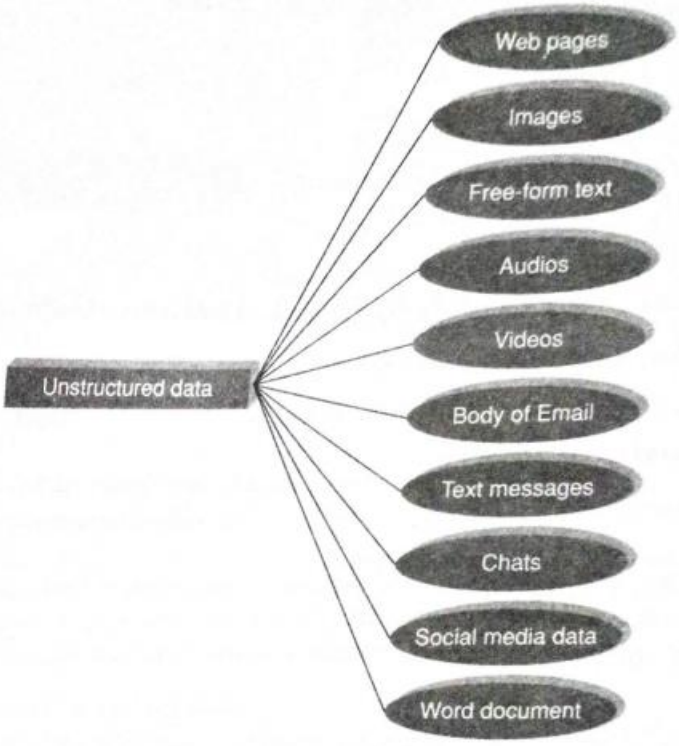


Figure 1.8 Sources of unstructured data.

How to deal with unstructured data:

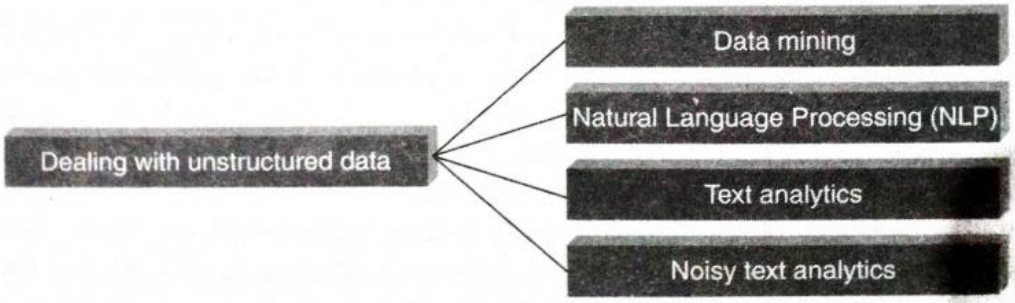


Figure 1.11 Dealing with unstructured data.

The following techniques are used to find patterns in or interpret unstructured data:

1. **Data mining:** First, we deal with large data sets. Second, we use methods at the intersection of artificial intelligence, machine learning, statistics, and database systems to unearth consistent patterns in large data sets and/or systematic relationships between variables. It is the analysis step of the "knowledge discovery in databases" process.

   Few popular data mining algorithms are as follows:

   - ***Association rule mining:*** It is also called "market basket analysis" or "affinity analysis". It is used to determine "What goes with what?" It is about when you buy a product, what is the other product that you are likely to purchase with it. For example, if you pick up bread from the grocery, are you likely to pick eggs or cheese to go with it.
   - ***Regression analysis:*** It helps to predict the relationship between two variables. The variable whose value needs to be predicted is called the dependent variable and the variables which are used to predict the value are referred to as the independent variables.

- ***Collaborative filtering:*** It is about predicting a user's preference or preferences based on the preferences of a group of users. For example, take a look at Table 1.5. We are looking at predicting whether User 4 will prefer to learn using videos or is a textual learner depending on one or a couple of his or her known preferences. We analyze the preferences of similar user profiles and on the basis of it, predict that User 4 will also like to learn using videos and is not a textual learner.

2. **Text analytics or text mining:** Compared to the structured data stored in relational databases, text is largely unstructured, amorphous, and difficult to deal with algorithmically. Text mining is the process of gleaning high quality and meaningful information (through devising of patterns and trends by means of statistical pattern learning) from text. It includes tasks such as text categorization, text clustering, sentiment analysis, concept/entity extraction, etc.
3. **Natural language processing (NLP):** It is related to the area of human computer interaction. It is about enabling computers to understand human or natural language input.
4. **Noisy text analytics:** It is the process of extracting structured or semi-structured information from noisy unstructured data such as chats, blogs, wikis, emails, message-boards, text messages, etc. The noisy unstructured data usually comprises one or more of the following: Spelling mistakes, abbreviations, acronyms, non-standard words, missing punctuation, missing letter case, filler words such as "uh", "um", etc.
5. **Manual tagging with metadata:** This is about tagging manually with adequate metadata to provide the requisite semantics to understand unstructured data.
6. **Part-of-speech tagging:** It is also called POS or POST or grammatical tagging. It is the process of reading text and tagging each word in the sentence as belonging to a particular part of speech such as "noun", "verb", "adjective", etc.
7. **Unstructured Information Management Architecture (UIMA):** It is an open source platform from IBM. It is used for real-time content analytics. It is about processing text and other unstructured data to find latent meaning and relevant relationship buried therein. Read up more on UIMA at the link: http://www.ibm.com/developerworks/data/downloads/uima/

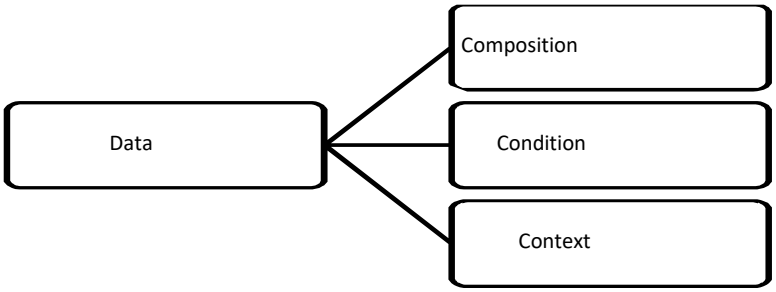
Characteristics of Data:

1.Composition: The composition of data deals with the structure of data, that is, the sources of data, the granularity, the types, and the nature of data as to whether it is static or real-time streaming.

2.Condition: The condition of data deals with the state of data, that is, "Can one use this data as is for analysis?" or "Does it require cleansing for further enhancement and enrichment?"

3.Context: The context of data deals with "Where has this data been generated?" "Whywas this data generated?" How sensitive is this data?" "What are the events associated with this data?" and so on.

Small data (data as it existed prior to the big data revolution) is about certainty. It is about known data sources; it is about no major changes to the composition or context of data.



Evolution of Big Data:

1970s and before was the era of mainframes. The data was essentially primitive and structured. Relational databases evolved in 1980s and 1990s. The era was of data intensive applications. The World Wide Web (WWW) and the Internet of Things (IOT) have led to anonslaught of structured, unstructured, and multimedia data. Refer Table 1.1.

	DATA GENERATIONAND STORAGE	DATA UTILZATION	DATA DRIVEN
COMPLEX AND UNSTRUCTURED			Structured data, Unstructured data, Multimedia data
COMPLEX AND RELATIONAL		Relational databases: Data-intensive applications	
PRIMITIVE AND STRUCTURED	Mainframes: Basic data storage		
	1970 and before	**Relational (1980 and 1990s)**	**2000 and beyond**

Big Data and its Importance

The importance of big data does not revolve around how much data a company has but how a company utilizes the collected data. Every company uses data in its own way; the more efficiently a company uses its data, the more potential it has to grow. The company can take data from any source and analyze it to find answers which will enable:

Big Data importance doesn't revolve around the amount of data a company has but 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.

By analysing the big data pools effectively the companies can get answers to :

Cost Savings :

o Some tools of Big Data like Hadoop can bring cost advantages to business when large amounts of data are to be stored.

o These tools help in identifying more efficient ways of doing business.

Time Reductions :

o The high speed of tools like Hadoop and in-memory analytics can easily identify new sources of data which helps businesses analyzing data immediately.

o This helps us to make quick decisions based on the learnings.

Understand the market conditions :

o By analyzing big data we can get a better understanding of current market conditions.

o For example: By analyzing customers' purchasing behaviours, a company can find out the products that are sold the most and produce products according to this trend. By this, it can get ahead of its competitors.

Control online reputation :

o Big data tools can do sentiment analysis.

o Therefore, you can get feedback about who is saying what about your company.

o If you want to monitor and improve the online presence of your business, then big data tools can help in all this.

Using Big Data Analytics to Boost Customer Acquisition(purchase) and Retention :

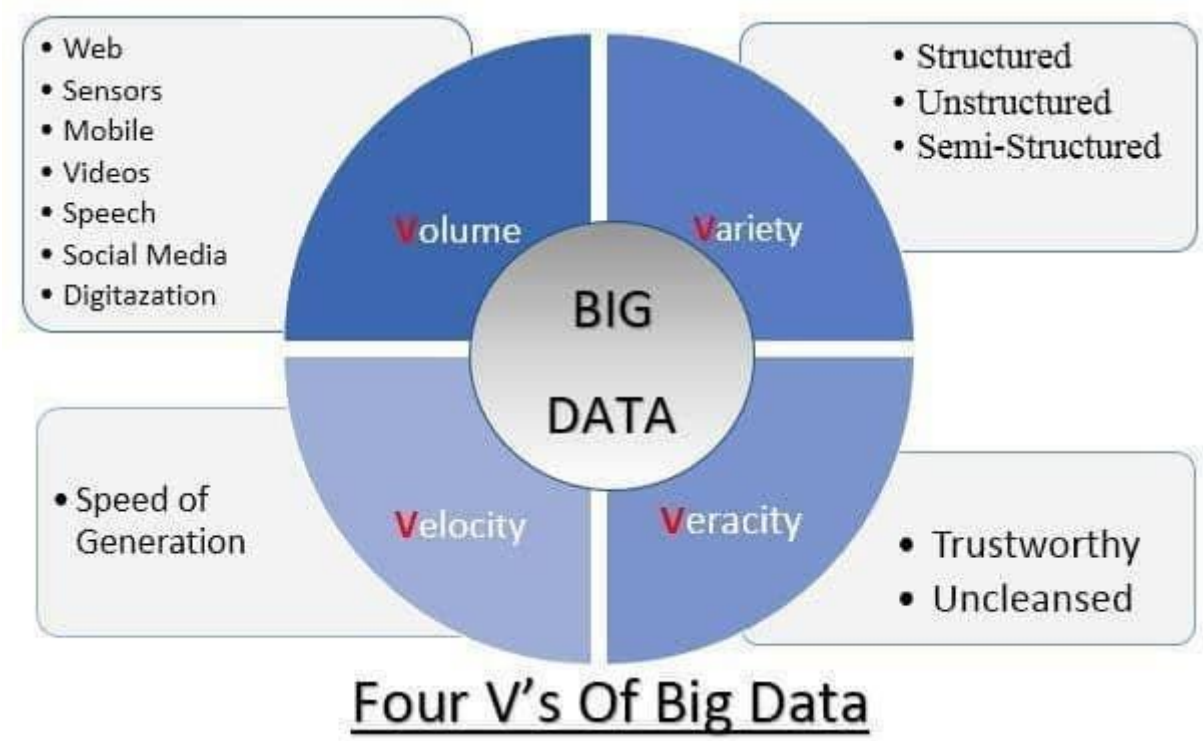
o The customer is the most important asset any business depends on.
o No single business can claim success without first having to establish a solid customer base.
o If a business is slow to learn what customers are looking for, then it is very likely to deliver poor quality products.
o The use of big data allows businesses to observe various customer-related patterns and trends.

Using Big Data Analytics to Solve Advertisers Problem and Offer Marketing Insights :

o Big data analytics can help change all business operations.
o Like the ability to match customer expectations, changing company's product line, etc.
o And ensuring that the marketing campaigns are powerful

Four v's Of Big Data

In recent years, Big Data was defined by the "*3Vs*" but now there is "6*Vs*" of Big Data which are also termed as the characteristics of Big Data as follows:



Four V's Of Big Data

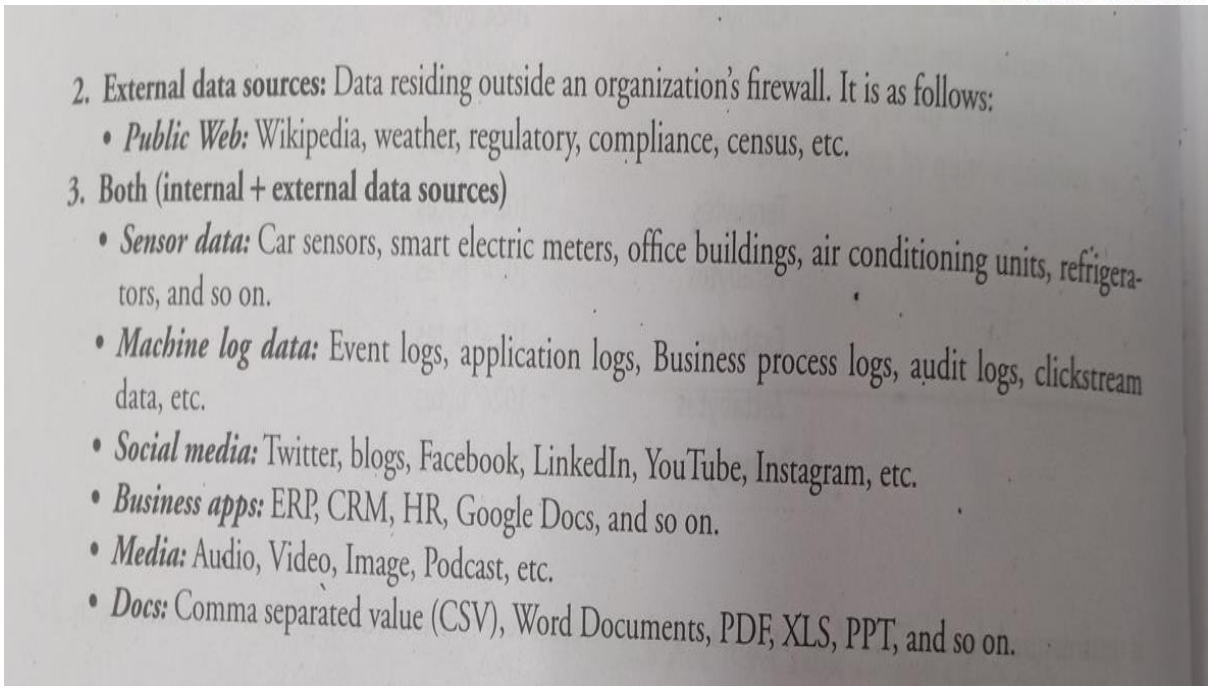
1. Volume:

Table 2.2 Growth of data

Bits	0 or 1
Bytes	8 bits
Kilobytes	1024 bytes
Megabytes	1024^2 bytes
Gigabytes	1024^3 bytes
Terabytes	1024^4 bytes
Petabytes	1024^5 bytes
Exabytes	1024^6 bytes
Zettabytes	1024^7 bytes
Yottabytes	1024^8 bytes

is unstructured data; a CCTV coverage, a weather forecast report is unstructured data too. Refer Figure 2.7 for the sources of big data.

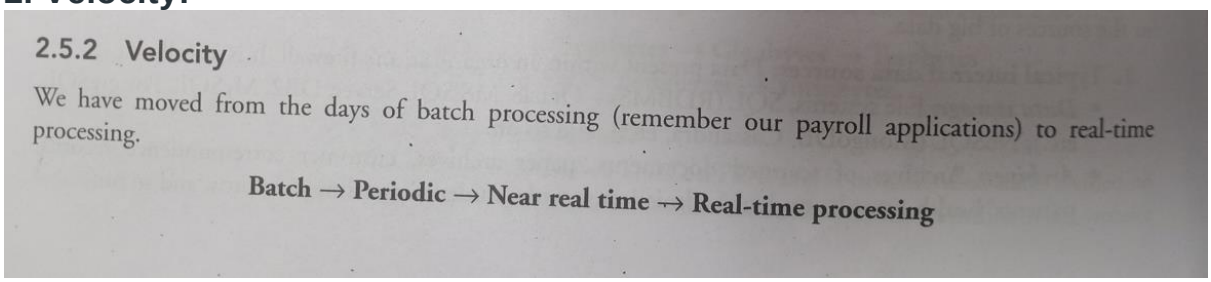
1. **Typical internal data sources:** Data present within an organization's firewall. It is as follows:
   - ***Data storage:*** File systems, SQL (RDBMSs – Oracle, MS SQL Server, DB2, MySQL, PostgreSQL, etc.), NoSQL (MongoDB, Cassandra, etc.), and so on.
   - ***Archives:*** Archives of scanned documents, paper archives, customer correspondence records, patients' health records, students' admission records, students' assessment records, and so on.



2. **External data sources:** Data residing outside an organization's firewall. It is as follows:
   - ***Public Web:*** Wikipedia, weather, regulatory, compliance, census, etc.
3. **Both (internal + external data sources)**
   - ***Sensor data:*** Car sensors, smart electric meters, office buildings, air conditioning units, refrigerators, and so on.
   - ***Machine log data:*** Event logs, application logs, Business process logs, audit logs, clickstream data, etc.
   - ***Social media:*** Twitter, blogs, Facebook, LinkedIn, YouTube, Instagram, etc.
   - ***Business apps:*** ERP, CRM, HR, Google Docs, and so on.
   - ***Media:*** Audio, Video, Image, Podcast, etc.
   - ***Docs:*** Comma separated value (CSV), Word Documents, PDF, XLS, PPT, and so on.

- The name 'Big Data' itself is related to a size which is enormous.
- Volume is a huge amount of data.
- To determine the value of data, size of data plays a very crucial role. If the volume of data is very large then it is actually considered as a 'Big Data'. This means whether a particular data can actually be considered as a Big Data or not, is dependent upon the volume of data.
- Hence while dealing with Big Data it is necessary to consider a characteristic 'Volume'.
- *Example:* In the year 2016, the estimated global mobile traffic was 6.2 Exabytes(6.2 billion GB) per month. Also, by the year 2020 we will have almost 40000 Exa Bytes of data.

2. Velocity:



2.5.2 Velocity

We have moved from the days of batch processing (remember our payroll applications) to real-time processing.

Batch → Periodic → Near real time → Real-time processing

- Velocity refers to the high speed of accumulation of data.
- In Big Data velocity data flows in from sources like machines, networks, social media, mobile phones etc.
- There is a massive and continuous flow of data. This determines the potential of data that how fast the data is generated and processed to meet the demands.

- Sampling data can help in dealing with the issue like 'velocity'.
- *Example:* There are more than 3.5 billion searches per day are made on Google. Also, FaceBook users are increasing by 22%(Approx.) year by year.

3. Variety:

- It refers to nature of data that is structured, semi-structured and unstructured data.
- It also refers to heterogeneous 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 generally refers to data that has defined the length and format of data.
    - **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.
    - **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.

4. Veracity:

- It refers to inconsistencies and uncertainty in data, that is data which is available can sometimes get messy and quality and accuracy are difficult to control.
- Big Data is also variable because of the multitude of data dimensions resulting from multiple disparate data types and sources.
- *Example:* Data in bulk could create confusion whereas less amount of data could convey half or Incomplete Information.

5. Value:

- After having the 4 V's into account there comes one more V which stands for Value!. The bulk of Data having no Value is of no good to the company, unless you turn it into something useful.
- Data in itself is of no use or importance but it needs to be converted into something valuable to extract Information. Hence, you can state that Value! is the most important V of all the 6V's.

6. Variability:

- How fast or, available data that extent is the structure of your data is changing ?.
- How often does the meaning or shape of your data!to ?change?.
- example : if you are eating same ice-cream daily and the taste just keep changing.

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 :

1. The digitization of society
2. The drop in technology costs
3. Connectivity through cloud computing
4. Increased knowledge about data science
5. Social media applications
6. 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.

1. The digitization of society

Big Data is largely consumer driven and consumer oriented. Most of the data in the world is generated by consumers, who are nowadays 'always-on'.

Most people now spend 4-6 hours per day consuming and generating data through a variety of devices and (social) applications.

With every click, swipe or message, new data is created in a database somewhere around the world.

Because everyone now has a smartphone in their pocket, the data creation sums to incomprehensible amounts.

Some studies estimate that 60% of data was generated within the last two years, which is a good indication of the rate with which society has digitized.

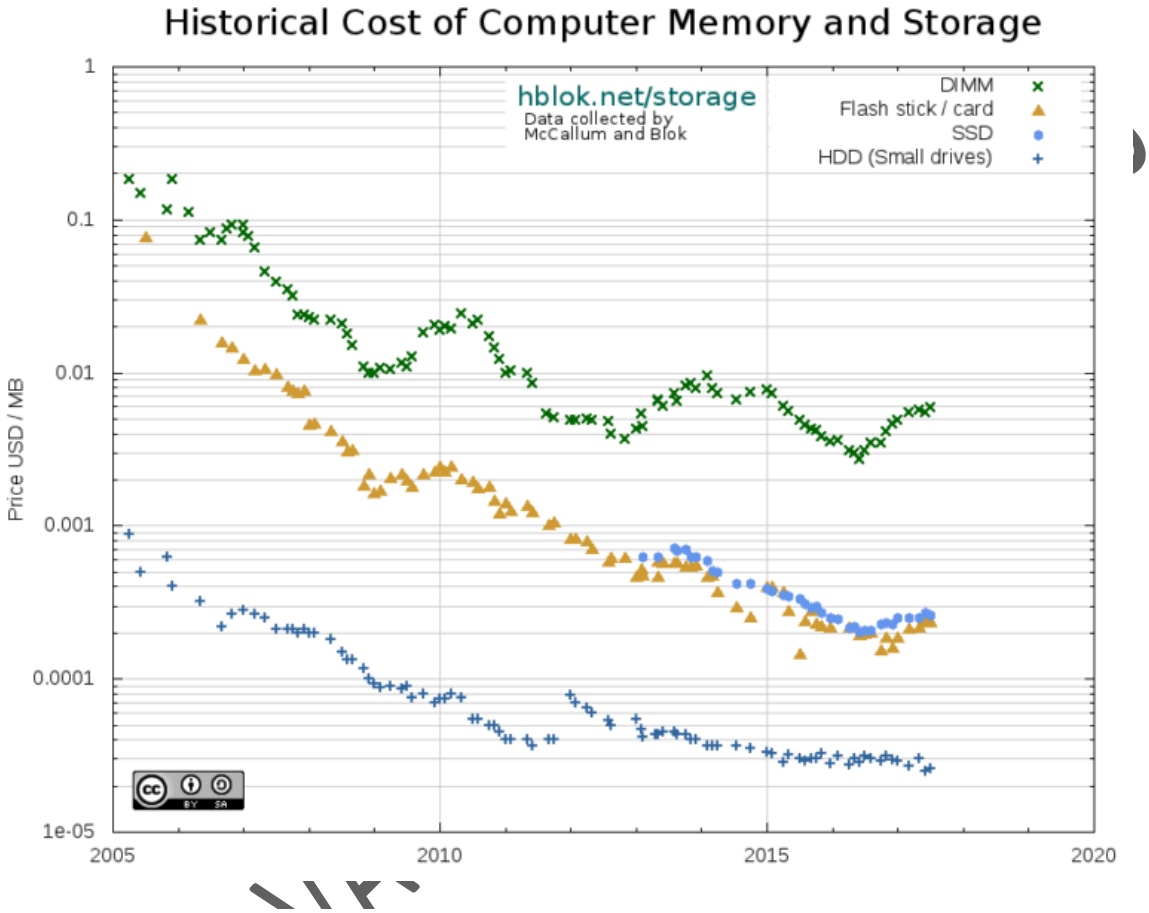
2. The drop in technology costs

Technology related to collecting and processing massive quantities of diverse (high variety) data has become increasingly more affordable.

The costs of data storage and processors keep declining, making it possible for small businesses and individuals to become involved with Big Data.

For storage capacity, the often-cited Moore's Law still holds that the storage density (and therefore capacity) still doubles every two years.

The plummeting of technology costs has been depicted in the figure below.



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

The most popular software framework (nowadays considered the standard for Big Data) is Apache Hadoop for distributed storage and processing.

Due to the high availability of these software frameworks in open sources, it has become increasingly inexpensive to start Big Data projects in organizations.

3. Connectivity through cloud computing

Cloud computing environments (where data is remotely stored in distributed storage systems) have made it possible to quickly scale up or scale down IT infrastructure and facilitate a pay-as-you-go model.

This means that organizations that want to process massive quantities of data (and thus have large storage and processing requirements) do not have to invest in large quantities of IT infrastructure.

Instead, they can license the storage and processing capacity they need and only pay for the amounts they actually used. As a result, most of Big Data solutions leverage the possibilities of cloud computing to deliver their solutions to enterprises.

4. Increased knowledge about data science

In the last decade, the term data science and data scientist have become tremendously popular. In October 2012, Harvard Business Review called the data scientist "sexiest job of the 21st century" and many other publications have featured this new job role in recent years. The demand for data scientist (and similar job titles) has increased tremendously and many people have actively become engaged in the domain of data science.

As a result, the knowledge and education about data science has greatly professionalized and more information becomes available every day. While statistics and data analysis mostly remained an academic field previously, it is quickly becoming a popular subject among students and the working population.

5. Social media applications

Everyone understands the impact that social media has on daily life. However, in the study of Big Data, social media plays a role of paramount importance. Not only because of the sheer volume of data that is produced everyday through platforms such as Twitter, Facebook, LinkedIn and Instagram, but also because social media provides nearly real-time data about human behavior.

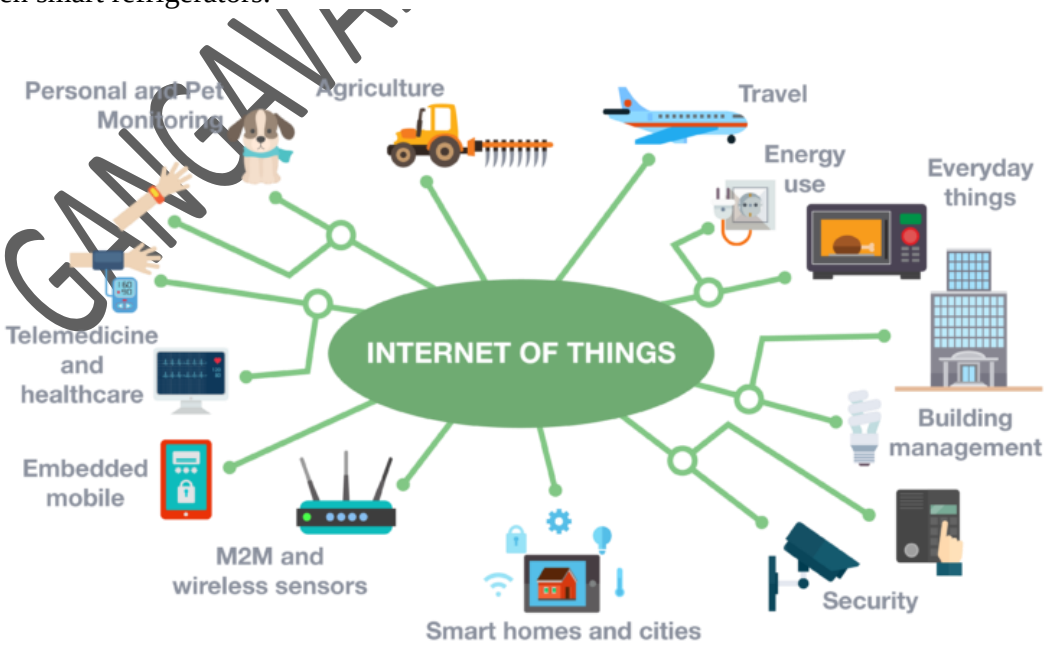
Social media data provides insights into the behaviors, preferences and opinions of 'the public' on a scale that has never been known before. Due to this, it is immensely valuable to anyone who is able to derive meaning from these large quantities of data. Social media data can be used to identify customer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

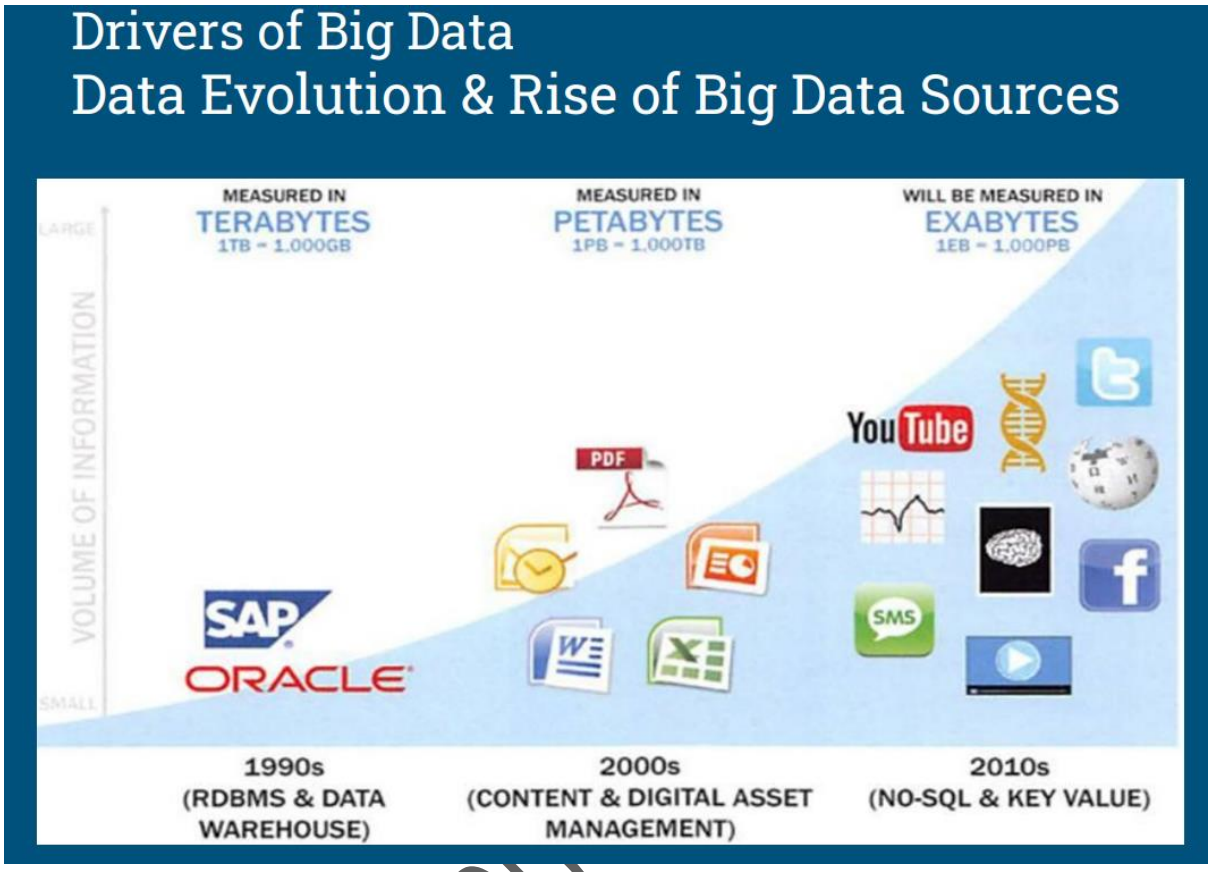
6. The upcoming internet of things (IoT)

The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and network connectivity which enables these objects to connect and exchange data.

It is increasingly gaining popularity as consumer goods providers start including 'smart' sensors in household appliances. Whereas the average household in 2010 had around 10 devices that connected to the internet, this number is expected to rise to 50 per household by 2020.

Examples of these devices include thermostats, smoke detectors, televisions, audio systems and even smart refrigerators.





● Medical information, such as diagnostic imaging

● Photos and video footage uploaded to the World Wide Web

● Video surveillance, such as the thousands of video cameras across a city

● Mobile devices, which provide geospatial location data of the users

● Metadata about text messages, phone calls, and application usage on smart phones

● Smart devices, which provide sensor-based collection of information from smart

● Non traditional IT devices, including the use of RFID readers,

GPS navigation systems, and seismic processing.

These are the multiple sources where the data can be generated from multiple sources.

Challenges of Big Data

The major challenges associated with big data are as follows –

- Capturing data
- Curation
- Storage
- Searching
- Sharing
- Transfer
- Analysis
- Presentation

1.Data is growing in an exponential rate.Most of the data have been generated in the last 2-3 years.

Introduction to Big Data Analyticts

Organizations and data collectors are realizing that the data they can gather from individuals contains intrinsic value and, as a result, a new economy is emerging.

As this new digital economy continues to evolve, the market sees the introduction of data vendors and data cleaners that use crowdsourcing to test the outcomes of machine learning techniques.

Other vendors offer added value by repackaging open source tools in a simpler way and bringing the tools to market. Vendors such as Cloudera, Hortonworks, and Pivotal have provided this value-add for the open source framework Hadoop.

Data devices and the "Sensornet" gather data from multiple locations and continuously generate new data about this data. For each gigabyte of new data created, an additional petabyte of data is created about that data.

For example, consider someone playing an online video game through a PC, game console, or smartphone. In this case, the video game provider captures data about the skill and levels attained by the player. Intelligent systems monitor and log how and when the user plays the game. As a consequence, the game provider can fine-tune the difficulty of the game, suggest other related games that would most likely interest the user, and offer additional equipment and enhancements for the character based on the user's age, gender, and interests. This information may get stored locally or uploaded to the game provider's cloud to analyze the gaming habits and opportunities for upsell and cross-sell, and identify archetypical profiles of specific kinds of users.

Smartphones provide another rich source of data. In addition to messaging and basic phone usage, they store and transmit data about Internet usage, SMS usage, and real-time location. This metadata can be used for analyzing traffic patterns by scanning the density of smartphones in locations to track the speed of cars or the relative traffic congestion on busy roads. In this way, GPS devices in cars can give drivers real-time updates and offer alternative routes to avoid traffic delays.

Retail shopping loyalty cards record not just the amount an individual spends, but the locations of stores that person visits, the kinds of products purchased, the stores where goods are purchased most often, and the combinations of products purchased together. Collecting this data provides insights into shopping and travel habits and the likelihood of successful advertisement targeting for certain types of retail promotions.

Data collectors include sample entities that collect data from the device and users.

Data results from a cable TV provider tracking the shows a person watches, which TV channels someone will and will not pay for to watch on demand, and the prices someone is willing to pay for premium TV content

Retail stores tracking the path a customer takes through their store while pushing a shopping cart with an RFID chip so they can gauge which products get the most foot traffic using geospatial data collected from the RFID chips

Data aggregators make sense of the data collected from the various entities from the "SensorNet" or the "Internet of

Things." These organizations compile data from the devices and usage patterns collected by government agencies, retail stores, and websites. In turn, they can choose to transform and package the data as products to sell to list brokers, who may want to generate marketing lists of people who may be good targets for specific ad campaigns.

Data users and buyers

These groups directly benefit from the data collected and aggregated by others within the data value chain.

Retail banks, acting as a data buyer, may want to know which customers have the highest likelihood to apply for a second mortgage or a home equity line of credit. To provide input for this analysis, retail banks may purchase data from a data aggregator. This kind of data may include demographic information about people living in specific locations; people who appear to have a specific level of debt, yet still have solid credit scores (or other characteristics such as paying bills on

time and having savings accounts) that can be used to infer credit worthiness; and those who are searching the web for information about paying off debts or doing home remodeling projects.

Obtaining data from these various sources and aggregators will enable a more targeted marketing campaign, which would have been more challenging before Big Data due to the lack of information or high-performing technologies.

Using technologies such as Hadoop to perform natural language processing on unstructured, textual data from social media websites, users can gauge the reaction to events such as presidential campaigns. People may, for example, want to determine public sentiments toward a candidate by analyzing related blogs and online comments. Similarly, data users may want to track and prepare for natural disasters by identifying which areas a hurricane affects first and how it moves, based on which geographic areas are tweeting about it or discussing it via social media.

A field to analyze and to extract information about the big data involved in the business or the data world so that proper conclusions can be made is called big data Analytics.

These conclusions can be used to predict the future or to forecast the business.

Also, this helps in creating a trend about the past.

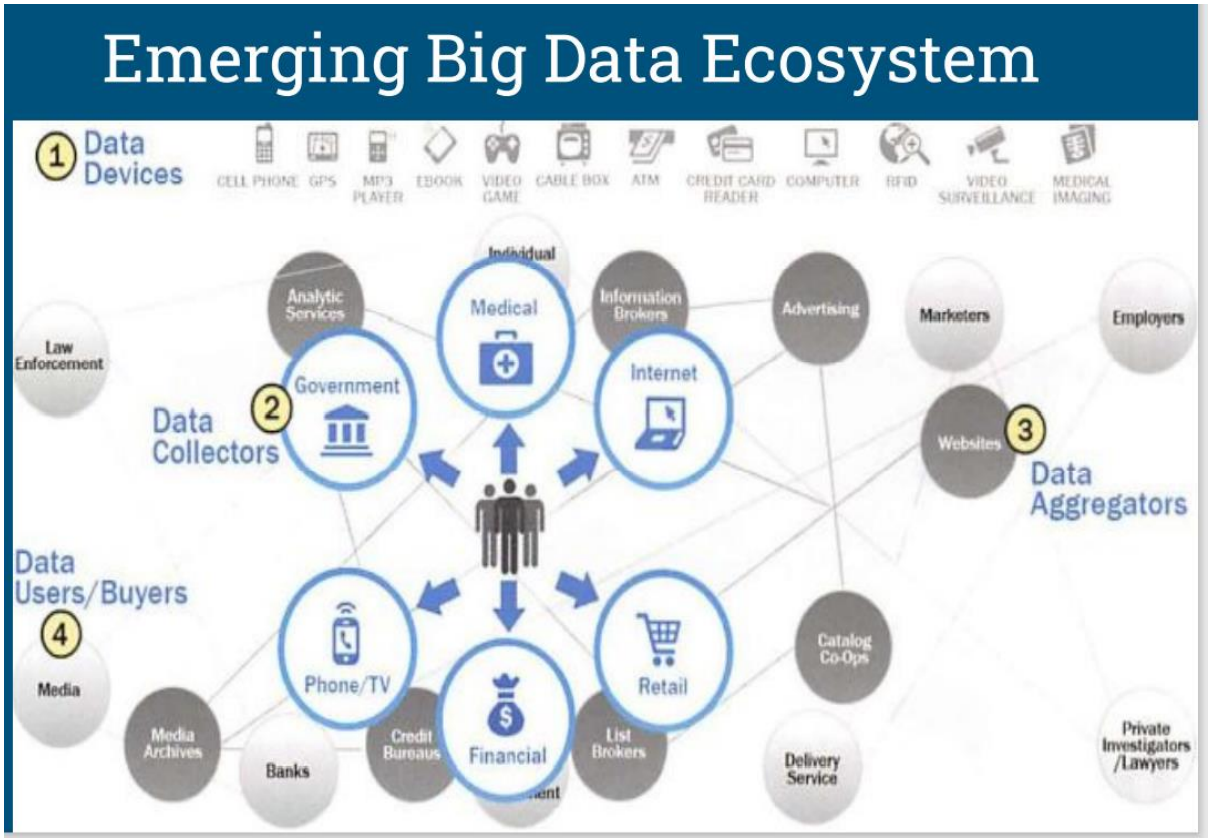
Skilled professionals in statistics and engineering with domain knowledge are needed in the analysis of big data as the data is huge, and analysis needs proper determination and skillset.

This data is more complex that it cannot be dealt with with traditional methods of analysis.

We produce a massive amount of data each day, whether we know about it or not. Every click on the internet, every bank transaction, every video we watch on

YouTube, every email we send, every like on our Instagram post makes up data for tech companies.

With such a massive amount of data being collected, it only makes sense for companies to use this data to understand their customers and their behavior better. This is the reason why the popularity of Data Science has grown manifold over the last few years.



A big data platform is a type of IT solution that combines the features and capabilities of several big data applications and utilities within a single solution, this is then used further for managing as well as analyzing Big Data.
It focuses on providing its users with efficient analytics tools for massive datasets.
The users of such platforms can custom build applications according to their use case like to calculate customer loyalty (E-Commerce user case), and so on.

Goal: The main goal of a Big Data Platform is to achieve: Scalability, Availability, Performance, and Security.

Example: Some of the most commonly used Big Data Platforms are :

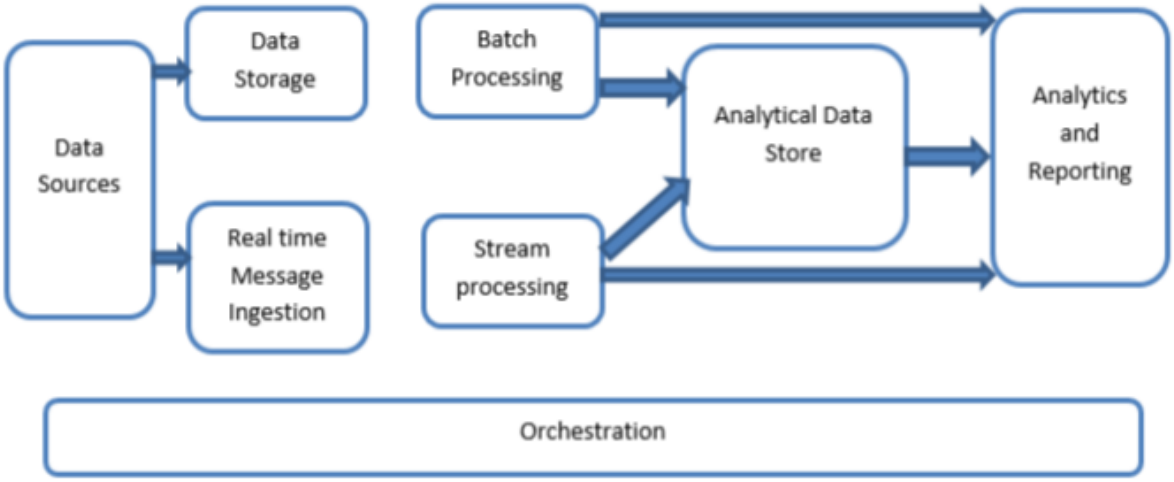
- Hadoop Delta Lake Migration Platform
- Data Catalog Platform
- Data Ingestion Platform
- IoT Analytics Platform
- Drivers for Big Data
- Big Data has quickly risen to become one of the most desired topics in the industry.
- The main business drivers for such rising demand for Big Data Analytics are :

1. The digitization of society
2. The drop in technology costs
3. Connectivity through cloud computing
4. Increased knowledge about data science
5. Social media applications
6. 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.

- **Big Data Architecture :**
- Big data architecture is designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems.



- The big data architectures include the following components:
- **Data sources**: All big data solutions start with one or more data sources.
- Example,
- Application data stores, such as relational databases.
- Static files produced by applications, such as web server log files.
- Real-time data sources, such as IoT devices.
- **Data storage**: Data for batch processing operations is stored in a distributed file store that can hold high volumes of large files in various formats (also called data lake).
- Example,
- Azure Data Lake Store or blob containers in Azure Storage.
- **Batch processing:** Since the data sets are so large, therefore a big data solution must process data files using long-running batch jobs to filter, aggregate, and prepare the data for analysis.
- **Real-time message ingestion:** If a solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing.
- **Stream processing:** After capturing real-time messages, the solution must process them by filtering, aggregating, and preparing the data for analysis. The processed stream data is then written to an output sink. We can use open-source Apache streaming technologies like Storm and Spark Streaming for this.

- **Analytical data store:** Many big data solutions prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools. Example: Azure Synapse Analytics provides a managed service for large-scale, cloud-based data warehousing.
- **Analysis and reporting:** The goal of most big data solutions is to provide insights into the data through analysis and reporting. To empower users to analyze the data, the architecture may include a data modelling layer. Analysis and reporting can also take the form of interactive data exploration by data scientists or data analysts.
- **Orchestration:** Most big data solutions consist of repeated data processing operations, that transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report. To automate these workflows, we can use an orchestration technology such as Azure Data Factory.

Applications of Big Data

In today's world big data have several applications, some of them are listed below :

Tracking Customer Spending Habit, Shopping Behavior :

In big retails stores, the management team has to keep data of customer's spending habits, shopping behaviour, most liked product, which product is being searched/sold most, based on that data, the production/collection rate of that product gets fixed.

Recommendation :

By tracking customer spending habits, shopping behaviour, big retail stores provide recommendations to the customers.

Smart Traffic System :

Data about the condition of the traffic of different roads, collected through cameras, GPS devices placed in the vehicle.

All such data are analyzed and jam-free or less jam way, less time taking ways are recommended.

One more profit is fuel consumption can be reduced.

Secure Air Traffic System :

At various places of flight, sensors are present.

These sensors capture data like the speed of flight, moisture, temperature, and other environmental conditions.

Based on such data analysis, an environmental parameter within flight is set up and varied.

By analyzing flight's machine-generated data, it can be estimated how long the machine can operate flawlessly and when it can be replaced/repaired.

Auto Driving Car :

In the various spots of the car camera, a sensor is placed that gathers data like the size of the surrounding car, obstacle, distance from those, etc.

These data are being analyzed, then various calculations are carried out.

These calculations help to take action automatically.

Virtual Personal Assistant Tool :

Big data analysis helps virtual personal assistant tools like Siri, Cortana and Google Assistant to provide the answer to the various questions asked by users.

This tool tracks the location of the user, their local time, season, other data related to questions asked, etc.

Analyzing all such data provides an answer.

Example: Suppose one user asks "Do I need to take Umbrella?"The tool collects data like location of the user, season and weather condition at that location, then analyzes these data to conclude if there is a chance of raining, then provides the answer.

IoT :

Manufacturing companies install IOT sensors into machines to collect operational data.

Analyzing such data, it can be predicted how long a machine will work without any problem when it requires repair.

Thus, the cost to replace the whole machine can be saved.

Education Sector Energy Sector :

Online educational courses conducting organization utilize big data to search candidates interested in that course.

If someone searches for a YouTube tutorial video on a subject, then an online or offline course provider organization on that subject sends an ad online to that person about their course.

Media and Entertainment Sector :

Media and entertainment service providing company like Netflix, Amazon Prime, Spotify do analysis on data collected from their users. Data like what type of video, music users are watching, listening to most, how long users are spending on site, etc are collected and analyzed to set the next business strategy.

Traditional BI versus Big data

Here are some key differences between traditional BI and big data:

1. **Data sources**: Traditional BI typically relies on structured data from internal systems, while big data can come from a wide range of external and internal sources, including social media, IoT devices, and sensors.
2. **Data volume**: Traditional BI deals with relatively small amounts of data, while big data deals with extremely large and complex sets of data.
3. **Data variety**: Traditional BI typically deals with structured data, while big data can include structured, semi-structured, and unstructured data from various formats such as text, images, videos, and audio.
4. **Data processing**: Traditional BI relies on traditional data processing techniques such as SQL, while big data uses newer technologies such as Hadoop, NoSQL databases, and machine learning to process and analyze the data.
5. **Time frame**: Traditional BI focuses on historical data, while big data can also include real-time data streams.
6. **Scale:** Traditional BI is typically implemented in a small scale, while big data is implemented in large scale and distributed environments.

Traditional Data	Big Data
Traditional data is generated in enterprise level.	Big data is generated outside the enterprise level.
Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
Traditional database system deals with structured data.	Big data system deals with structured, semi-structured,database, and unstructured data.
Traditional data is generated per hour or per day or more.	But big data is generated more frequently mainly per seconds.
Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.
Data integration is very easy.	Data integration is very difficult.
Normal system configuration is capable to process traditional data.	High system configuration is required to process big data.
The size of the data is very small.	The size is more than the traditional data size.
Traditional data base tools are required to perform any data base operation.	Special kind of data base tools are required to perform any databaseschema-based operation.
Normal functions can manipulate data.	Special kind of functions can manipulate data.
Its data model is strict schema based and it is static.	Its data model is a flat schema based and it is dynamic.
Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.

Traditional Data	Big Data
Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.
It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
Its data sources includes ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

A typical data warehouse often includes the following elements:

- A relational database to store and manage data
- An extraction, loading, and transformation (ELT) solution for preparing the data for analysis
- Statistical analysis, reporting, and data mining capabilities
- Client analysis tools for visualizing and presenting data to business users

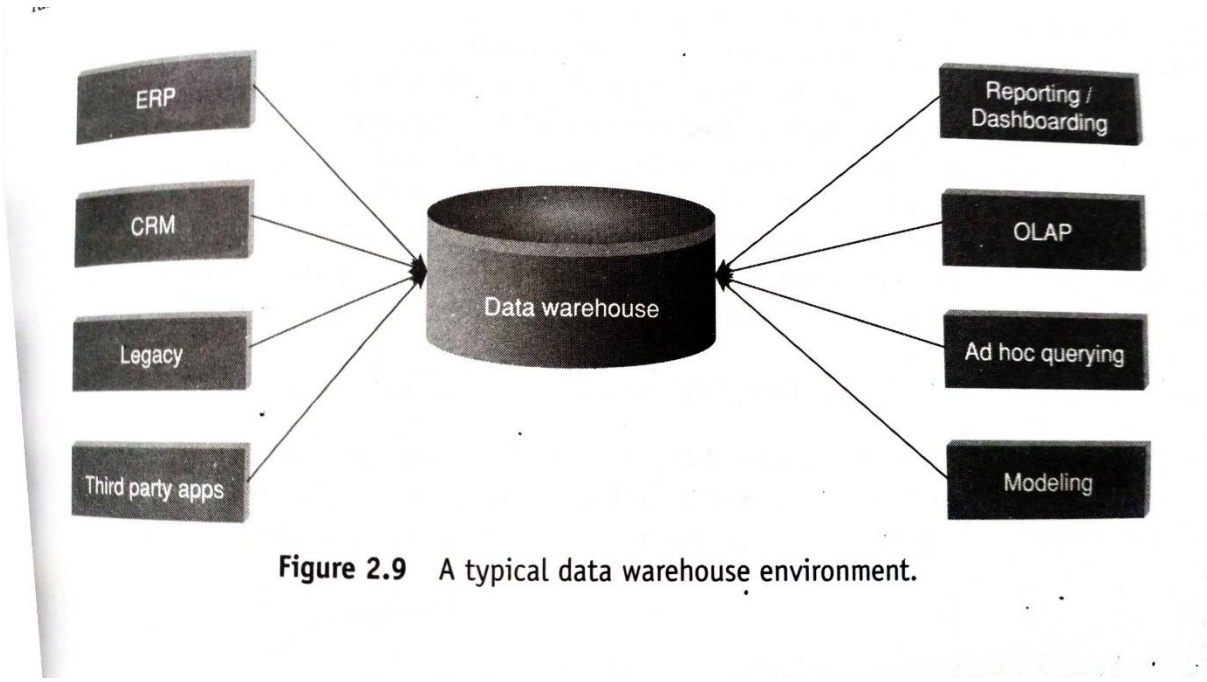


Figure 2.9 A typical data warehouse environment.

A typical Hadoop environment

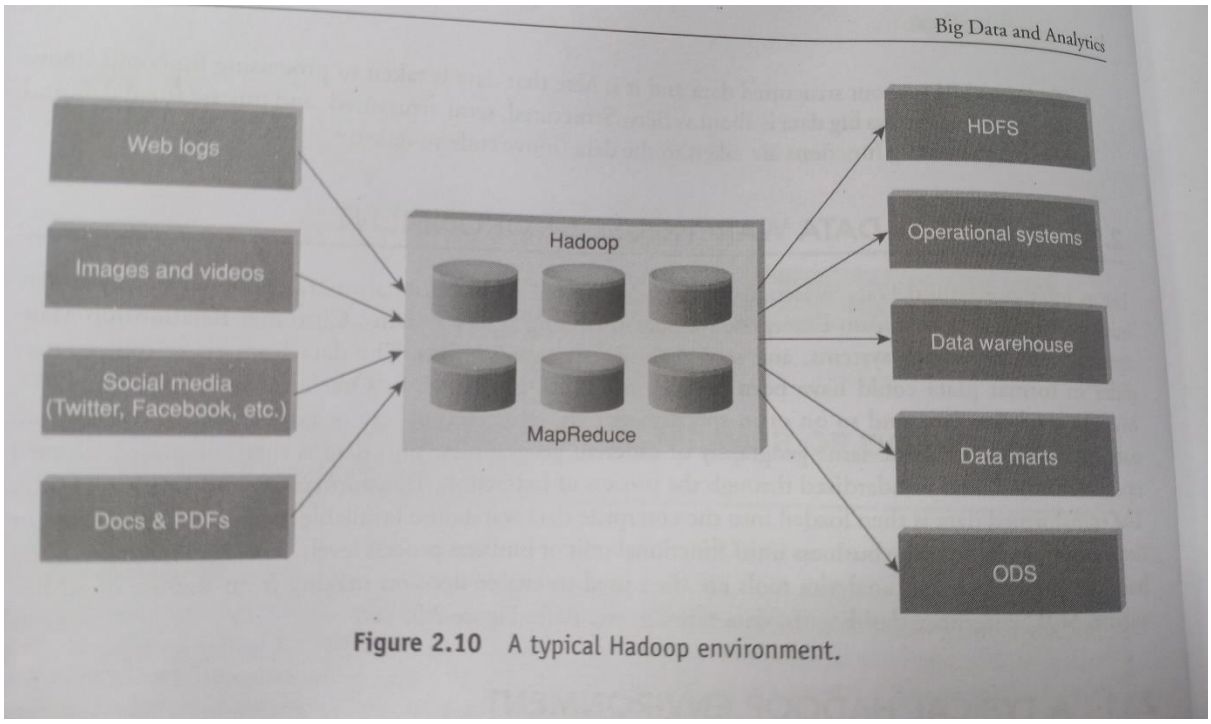


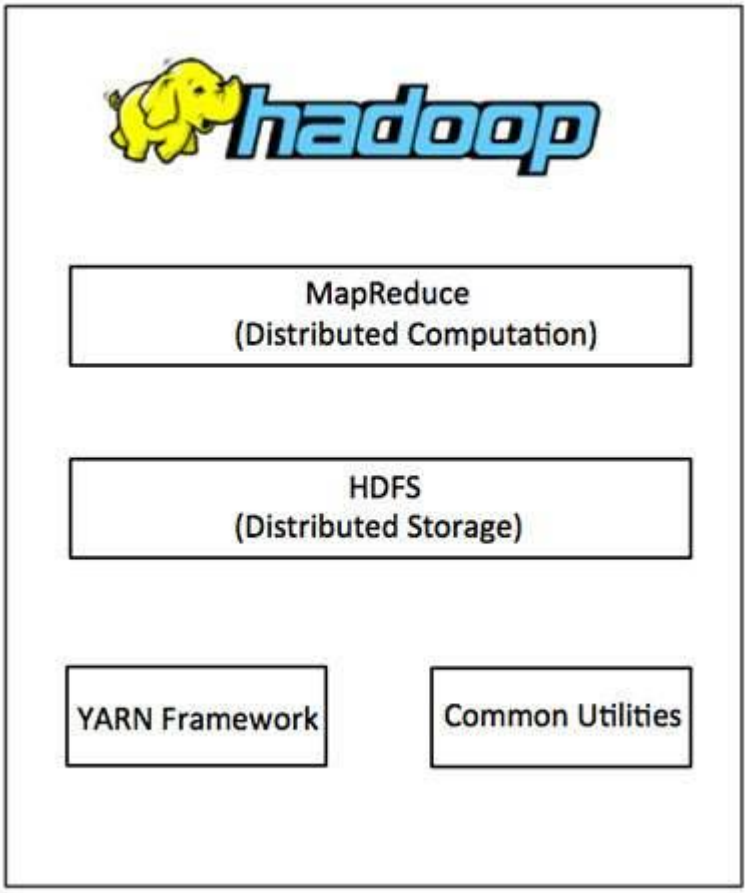
Figure 2.10 A typical Hadoop environment.

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 –

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).



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.

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.

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

- **Hadoop Common** – These are Java libraries and utilities required by other Hadoop modules.
- **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.

A coexistence of data warehouse and Hadoop environment

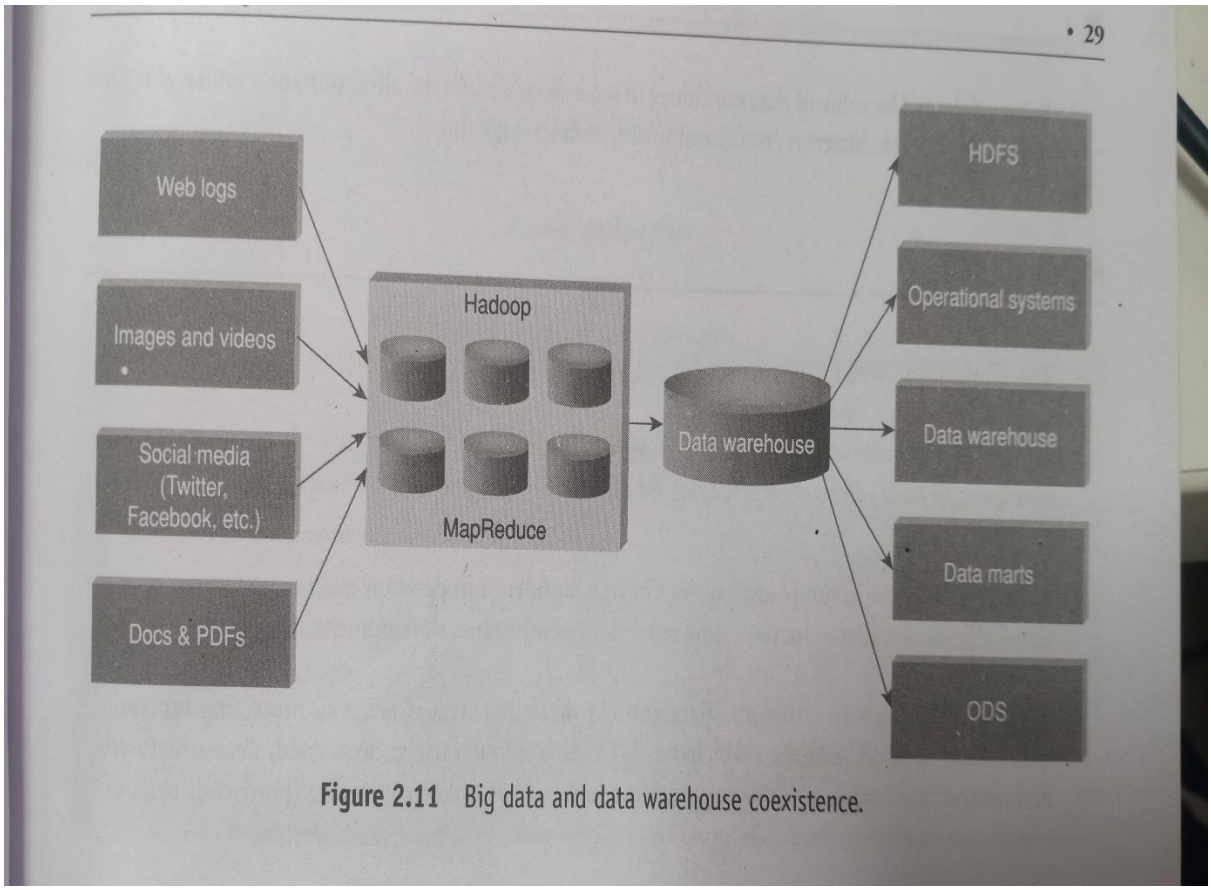


Figure 2.11 Big data and data warehouse coexistence.

KG REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

B. Tech III Year II Semester **Academic Year: 2023-2024**

SUBJECT: BIG DATA ANALYTICTS **SUBJECT CODE: KG21CD605** **REGULATION: KGR21**

UNIT-2 NOTES

UNIT II
Big Data Technologies: Hadoop's Parallel World – Data discovery – Opensource technology for Big Data Analytics – cloud and Big Data – Predictive Analytics – Mobile Business Intelligence and Big Data

Big Data Technologies: Hadoop's Parallel World

Brief History of Hadoop

There are many Big Data technologies that have been making an impact on the new technology stacks for handling Big Data, but Apache Hadoop is one technology that has been the darling of Big Data talk.

→Hadoop is an open-source platform for storage and processing of diverse data types that enables data-driven enterprises to rapidly derive the complete value from all their data.

→The original creators of Hadoop are Doug Cutting (used to be at Yahoo! now at Cloudera) and Mike

→Doug and Mike were building a project called "Nutch" with the goal of creating a large Web index.

→They saw the MapReduce and GFS papers from Google, which were obviously super relevant to the problem Nutch was trying to solve.

→Hadoop gives organizations the flexibility to ask questions across their structured and unstructured data that were previously impossible to ask or solve:

The scale and variety of data have permanently overwhelmed the ability to cost-effectively extract value using traditional platforms.

→The scalability and elasticity of free, open-source Hadoop running on standard hardware allow organizations to hold onto more data than ever before,

at a transformationally lower TCO than proprietary solutions and thereby take advantage of all their data to increase operational efficiency and gain a competitive edge.

→At one-tenth the cost of traditional solutions, Hadoop excels at supporting complex analyses— including detailed, special-purpose computation—across large collections of data.

Hadoop workloads

Hadoop handles a variety of workloads, including search, log processing, recommendation systems, data warehousing, and video/image analysis.

Today 's explosion of data types and volumes means that Big Data equals big opportunities and Apache Hadoop empowers organizations to work on the most modern scale-out architectures using a clean-sheet design data framework, without vendor lock-in.

- Apache Hadoop is an open-source project administered by the Apache Software Foundation.
- The software was originally developed by the world 's largest Internet companies to capture and analyze the data that they generate.
- 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 stores terabytes, and even petabytes, of data inexpensively. It is robust and reliable and handles hardware and system fail.
- Hadoop runs on clusters of commodity servers and each of those servers has local CPUs and disk storage that can be leveraged by the system.

The two critical components of Hadoop are:

1. The Hadoop Distributed File System (HDFS).

HDFS is the storage system for a Hadoop cluster. When data lands in the cluster, HDFS breaks it into pieces and distributes those pieces among the different servers participating in the cluster. Each server stores just a small fragment of the complete data set, and each piece of data is replicated on more than one server.

2. MapReduce.

Because Hadoop stores the entire dataset in small pieces across a collection of servers, analytical jobs can be distributed, in parallel, to each of the

servers storing part of the data. Each server evaluates the question against its local fragment simultaneously and reports its results back for collation into a comprehensive answer. MapReduce is the agent that distributes the work and collects the results.

Features of Hadoop

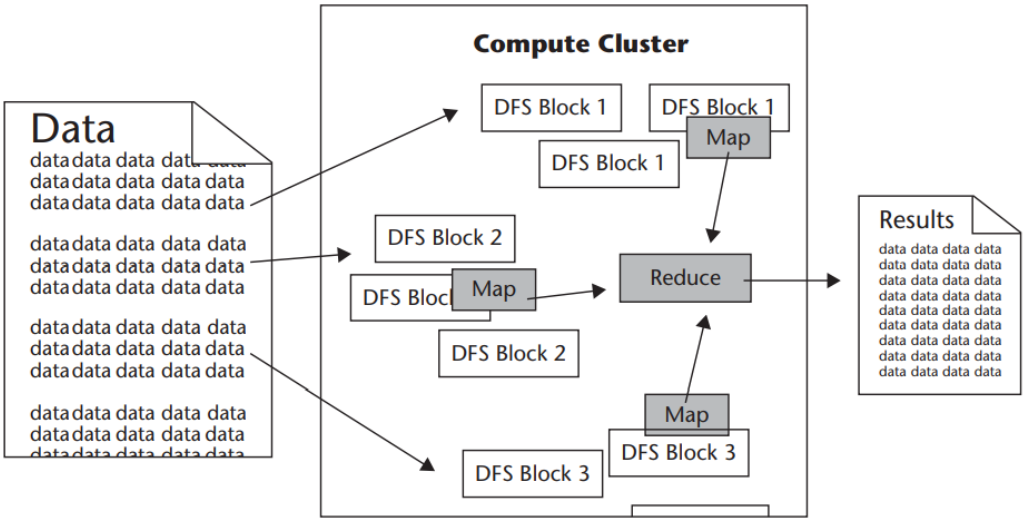
→Both HDFS and MapReduce are designed to continue to work in the face of system failures.

→HDFS continually monitors the data stored on the cluster. If a server becomes unavailable, a disk drive fails, or data is damaged, whether due to hardware or software problems, HDFS automatically restores the data from one of the known good replicas stored elsewhere on the cluster.

→Likewise, when an analysis job is running, MapReduce monitors progress of each of the servers participating in the job.

→If one of them is slow in returning an answer or fails before completing its work, MapReduce automatically starts another instance of that task on another server that has a copy of the data.

→Because of the way that HDFS and MapReduce work, Hadoop provides scalable, reliable, and fault-tolerant services for data storage and analysis at very low cost.



Source: Apache Software Foundation.

Old vs. New Approaches

- The old way is a data and analytics technology stack with different layers "cross-communicating data" and working on "scale-up" expensive hardware.
- The new way is a data and analytics platform that does all the data processing and analytics in one "layer," without moving data back and forth on cheap but scalable ("scale out") commodity hardware.
- The new approach is based on two foundational concepts. Number one, data needs to be stored in a system in which the hardware is infinitely scalable.
- In other words, you cannot allow hardware (storage and network) to become the bottleneck. Number two, data must be processed, and converted into usable business intelligence where it sits.

The three main important points are

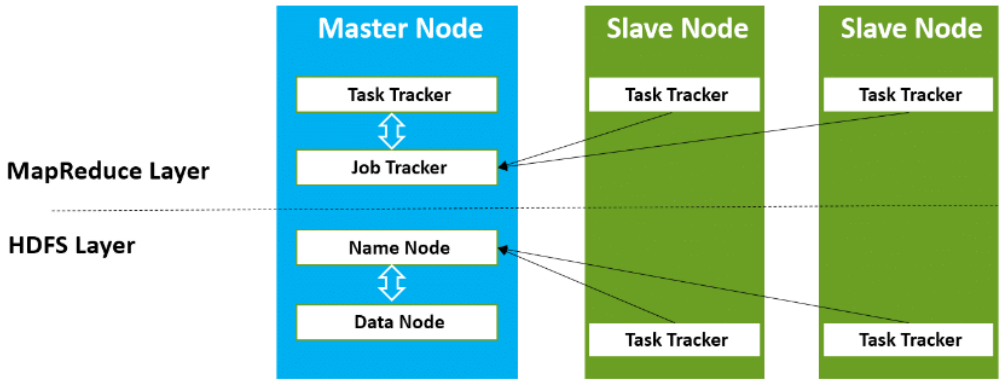
1.The technology stack has changed. New proprietary technologies and open-source inventions enable different approaches that make it easier and more affordable to store, manage, and analyze data.

2. Hardware and storage is affordable and continuing to get cheaper to enable massive parallel processing.

3. The variety of data is on the rise and the ability to handle unstructured data is on the rise.

The Hadoop High-level Architecture

Hadoop Architecture based on the two main components namely MapReduce and HDFS



The Apache Hadoop Module

Hadoop Common: Includes the common utilities which supports the other Hadoop modules

HDFS: Hadoop Distributed File System provides unrestricted, high-speed access to the data application.

Hadoop YARN: This technology is basically used for scheduling of job and efficient management of the cluster resource.

MapReduce: This is a highly efficient methodology for parallel processing of huge volumes of data.

Apache Ambari : It is a tool for managing, monitoring and provisioning of the Hadoop clusters. Apache Ambari supports the HDFS and MapReduce programs. Major highlights of Ambari are:

- Managing of the Hadoop framework is highly efficient, secure and consistent.
- Management of cluster operations with an intuitive web UI and a robust API
- The installation and configuration of Hadoop cluster are simplified effectively.
- It is used to support automation, smart configuration and recommendations
- Advanced cluster security set-up comes additional with this tool kit.
- The entire cluster can be controlled using the metrics, heat maps, analysis and troubleshooting
- Increased levels of customization and extension make this more valuable.

Cassandra: It is a distributed system to handle extremely huge amount of data which is stored across several commodity servers. The database management system (DBMS)is highly available with no single point of failure.

HBase: it is a non-relational, distributed database management system that works efficiently on sparse data sets and it is highly scalable.

Apache Spark: This is highly agile, scalable and secure the Big Data compute engine, versatiles the sufficient work on a wide variety of applications like real-time processing, machine learning, ETL and so on.

Hive: It is a data warehouse tool basically used for analyzing, querying and summarizing of analyzed data concepts on top of the Hadoop framework.

Pig: Pig is a high-level framework which ensures us to work in coordination either with Apache Spark or MapReduce to analyze the data. The language used to code for the frameworks are known as Pig Latin.

Sqoop: This framework is used for transferring the data to Hadoop from relational databases. This application is based on a command-line interface.

Oozie: This is a scheduling system for workflow management, executing workflow routes for successful completion of the task in a Hadoop.

Zookeeper: Open source centralized service which is used to provide coordination between distributed applications of Hadoop. It offers the registry and synchronization service on a high level.

- **Hadoop Mapreduce (Processing/Computation layer) –**MapReduce is a parallel programming model mainly used for writing large amount of data distribution applications devised from Google for efficient processing of large amounts of datasets, on large group of clusters.
- **Hadoop HDFS (Storage layer) –**Hadoop Distributed File System or HDFS is based on the Google File System (GFS) which provides a distributed file system that is especially designed to run on commodity hardware. It reduces the faults or errors and helps incorporate low-cost hardware. It gives high level processing throughput access to application data and is suitable for applications with large datasets.

- **Hadoop YARN** –Hadoop YARN is a framework used for job scheduling and cluster resource management.
- **Hadoop Common** –This includes Java libraries and utilities which provide those java files which are essential to start Hadoop.
- **Task Tracker** –It is a node which is used to accept the tasks such as shuffle and Mapreduce form job tracker.
- **Job Tracker** –It is a service provider which runs Mapreduce jobs on cluster.
- **Name Node** –It is a node where Hadoop stores all file location information(data stored location) in Hadoop distributed file system.
- **Data Node** – The data is stored in the Hadoop distributed file system.

Data Discovery: Work the Way People 's Minds Work

There is a lot of buzz in the industry about data discovery.

The term used to describe the new wave of business intelligence that enables users to explore data, make discoveries, and uncover insights in a dynamic and intuitive way versus predefined queries and preconfigured drill-down dashboards.

Tableau Software and QlikTech are the two Business intelligence tools used for reporting .

- Analytics and reporting are produced by the people using the results. IT provides the infrastructure, but business people create their own reports and dashboards.
- There is a simple example of powerful visualization that the Tableau team is referring to.
- A company uses an interactive dashboard to track the critical metrics driving their business.

Example of Tableau Software

- A company uses an interactive dashboard to track the critical metrics driving their business.
- Every day, the CEO and other executives are plugged in real-time to see how their markets are performing in terms of sales and profit, what the service quality scores look like against advertising investments, and how products are performing in terms of revenue and profit.

- Interactivity is key: a click on any filter lets the executive look into specific markets or products.
- She can click on any data point in any one view to show the related data in the other views.
- Hovering over a data point can get any unusual pattern or outlier by showing details on demand.

Or she can click through the underlying information in a split-second

"Business intelligence needs to work the way people 's minds work.

Users need to navigate and interact with data any way they want to—asking and answering questions on their own and in big groups or teams.

One capability that we have all become accustomed to is search, what many people refer to as "Googling."

This is a prime example of the way people 's minds work. Qliktech has designed a way for users to leverage direct— and indirect—search.

- With QlikView search, users type relevant words or phrases in any order and get instant, associative results.
- With a global search bar, users can search across the entire data set. With search boxes on individual list boxes, users can confine the search to just that field.
- Users can conduct both direct and indirect searches. For example, if a user wanted to identify a sales rep but couldn't 't remember the sales rep 's name—just details about the person, such as that he sells fish to customers in the Nordic region.

Open-Source Technology for Big Data Analytics

- 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 refers to the use of open-source software and tools for analysing 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 services encompass:

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

Proprietary Software:

- Proprietary software is computer software where the source codes are publicly not available only the company that has created them can modify it.
- Here the software is developed and tested by the individual or organization by which it is owned not by the public.
- This software is managed by a closed team of individuals or groups that developed it.
- We have to pay to get this software and its commercial support is available for maintenance.
- The company gives a valid and authenticated license to the users to use this software. But this license puts some restrictions on users also like.
- **Some examples of Proprietary software include Windows, macOS, Internet Explorer, Google Earth, Microsoft Office, etc.**

1. Hadoop

Even if you are a beginner in this field, we are sure that this is not the first time you've read about Hadoop. It is recognized as one of the most popular big data tools to analyze large data sets, as the platform can send data to different servers. Another benefit of using Hadoop is that it can also run on a cloudinfrastructure.

This open-source software framework is used when the data volume exceeds the available memory. This big data tool is also ideal for data exploration, filtration, sampling, and summarization. It consists of four parts:

- **Hadoop Distributed File System:** This file system, commonly known as HDFS, is a distributed file system compatible with very high-scale bandwidth.
- **MapReduce:** It refers to a programming model for processing big data.
- **YARN:** All Hadoop's resources in its infrastructure are managed and scheduled using this platform.
- **Libraries:** They allow other modules to work efficiently with Hadoop.

2. Apache Spark

The next hype in the industry among big data tools is Apache Spark. the reason behind this is that this open-source big data tool fills the gaps of Hadoop when it comes to data processing. This big data tool is the most preferred tool for data analysis over other types of programs due to its ability to store large computations in memory**.** It can run complicated algorithms, which is a prerequisite for dealing with large data sets.

Proficient in handling batch and real-time data, Apache Spark is flexible to work with HDFS and OpenStack Swift or Apache Cassandra. Often used as an alternative to MapReduce, Spark can run tasks 100x faster than Hadoop's MapReduce.

3. Cassandra

Apache Cassandra is one of the best big data tools to process structured data sets. Created in **2008 by Apache Software Foundation,** it is recognized as the best open-source big data tool for scalability. This big data tool has a proven fault-tolerance on cloud infrastructure and commodity hardware, making it more critical for big data uses.

It also offers features that no other relational and NoSQL databases can provide. This includes simple operations, cloud availability points, performance, and continuous availability as a data source, to name a few. Apache Cassandra is used by giants like **Twitter, Cisco, and Netflix.**

4. MongoDB

MongoDB is an ideal alternative to modern databases. A document-oriented database is an ideal choice for businesses that need fast and real-time data for instant decisions. One thing that sets it apart from other traditional databases is that it makes use of documents and collections instead of rows and columns.

Thanks to its power to store data in documents, it is very flexible and can be easily adapted by companies. It can store any data type, be it integer, strings, Booleans, arrays, or objects**.** MongoDB is easy to learn and provides support for multiple technologies and platforms.

5.Apache Hive

Hive is an open source big data software tool. It allows programmers analyze large data sets on Hadoop. It helps with querying and managing large datasets real fast.

Features:

- It Supports SQL like query language for interaction and Data modeling
- It compiles language with two main tasks map, and reducer
- It allows defining these tasks using Java or Python

- Hive designed for managing and querying only structured data
- Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming
- It offers Java Database Connectivity (JDBC) interface.

6.kaggle

- Kaggle is the world's largest big data community. It helps organizations and researchers to post their data & statistics. It is the best place to analyze data seamlessly.

Features:

- The best place to discover and seamlessly analyze open data
- Search box to find open datasets
- Contribute to the open data movement and connect with other data enthusiasts

7.Apache Hbase

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.

HBase integrates seamlessly with Apache Hadoop and the Hadoop ecosystem and runs on top of the Hadoop Distributed File System (HDFS) or Amazon S3 using Amazon Elastic MapReduce (EMR) file system, or EMRFS. HBase serves as a direct input and output to the Apache MapReduce framework for Hadoop, and works with Apache Phoenix to enable SQL-like queries over HBase tables.

8.Apache Storm

It is a free big data open-source computation system. It is one of the best big data tools that offers a distributed, real-time, fault-tolerant processing system.

Having been benchmarked as processing one million 100-byte messages per second per node, it has big data technologies and tools that use parallel calculations that can run across a cluster of machines.

Being open source, robust and flexible, it is preferred by medium and large-scale organizations. It guarantees data processing even if the messages are lost, or nodes of the cluster die.

9.Apache Pig

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

10.Apache Flink

Apache Flink is an open-source platform that provides a scalable, distributed, fault-tolerant, and stateful stream processing capabilities. Flink is one of the most recent and pioneering Big Data processing frameworks.

Apache Flink allows to ingest massive streaming data (up to several terabytes) from different sources and process it in a distributed fashion way across multiple nodes, before pushing the derived streams to other services or applications such as Apache Kafka, DBs, and Elastic search. Simply, the basics building blocks of a Flink pipeline: input, processing, and output. Its runtime supports low-latency processing at extremely high throughputs in a fault-tolerant manner. Flink capabilities enable real-time insights from streaming data and event-based

capabilities. Flink enables real-time data analytics on streaming data and fits well for continuous Extract-transform-load (ETL) pipelines on streaming data and for event-driven applications as well.

The open-source projects are managed and supported by commercial companies, such as Cloudera, that provide extra capabilities, training, and professional services that support open-source projects such as Hadoop.

This is similar to what Red Hat has done for the open-source project Linux.

The advantage of the open source stack—flexibility, extensibility, and lower cost.

"One of the great benefits of open source lies in the flexibility of the adoption model: you download and deploy it when you need it," .With open source, you can try it and adopt it at your own pace.

The several assumptions of big data are

1.The amounts of data generated would be manageable

2. Programming resources would remain scarce

3. Faster data processing would require bigger, more expensive hardware.

- The old model was top-down, slow, inflexible and expensive.
- The new software development model is bottom-up, fast, flexible, and considerably less costly.
- A traditional proprietary stack is defined and controlled by a single vendor, or by a small group of vendors.
- It reflects the old command-and-control mentality of the traditional corporate world and the old economic order.

David then makes the case for an open-source analytics stack.

An open-source stack is defined by its community of users and contributors.

No one "controls" an open-source stack, and no one can predict exactly how it will evolve.

The open-source stack reflects the new realities of the networked global economy, which is increasingly dependent on big data.

It have been designing their solutions to plug and play with technology such as Hadoop.

For example, Teradata Aster designed SQL-H, which is a seamless way to execute SQL and SQL-MapReduce on Apache Hadoop data.

Cloud and Big Data

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). It's a virtualization framework.

It is like a resource on demand whether it be storage, computing etc. Cloud follows pay per usage model. You need to pay the amount of resource you use.

This computing service by cloud charges you based only on the amount of computing resources we use. So for example, if you want to give demo to a client on a cluster of more than 100 machines and you do not have so many machines currently available with you, then in such case cloud computing plays a very important role.

Cloud plays an important role within the Big Data world, by providing horizontally expandable and optimized infrastructure that supports practical implementation of Big Data.

In cloud computing, all data is gathered in data centers and then distributed to the end-users. Further, automatic backups and recovery of data is also ensured for business continuity, all such resources are available in the cloud.

We do not know exact physical location of these resources provided to us. You just need dummy terminals like desktops, laptops, phones etc. and a net connection.

There are multiple ways to access the cloud:

1. Applications or software as a service (SAAS) ex. Salesforce.com, dropbox, google drive etc.
2. Platform as a service (PAAS)
3. Infrastructure as a service (IAAS)

Features of Cloud Computing

Let us see few features of cloud computing:

a. Scalability

Scalability is provided by using distributed computing

b. Elasticity

Customers are allowed to use and pay for only that much resource which it is using.

In cloud computing, elasticity is defined as the degree to which a system is able to adapt to workload changes in an autonomic manner, so that at any time the available resources match the current demand as closely as possible.

c. Resource Pooling

Same resources are allowed to be used by multiple organizations. The computing resources are pooled for serving various consumers via multi-tenant model, with different resources dynamically assigned and reassigned according to consumer demand.

d. Self service

Customers are provided easy to use interface through which they can choose services they want. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed without requiring human interaction.

e. Low Costs

It charges you based only on the amount of computing resources we use and you need not buy expensive infrastructure. Pricing on a utility computing basis is usage-based and fewer IT skills are required for implementation.

f. Fault Tolerance

Allows recovery in case of a part in cloud system fails to respond.

Cloud Deployment Models

There are mainly 2 types of cloud deployments models:

- Public cloud – A cloud is called a "public cloud" when the services are open over a network for public use.
- Private Cloud – Private cloud is operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally.

Cloud Delivery Models

Cloud services are categorized as below:

1. **Infrastructure as a service (IAAS):** It means complete infrastructure will be provided to you. Maintenance related tasks will be done by cloud provider and you can use it as per your requirement. It can be used as public and private both.

Examples of IaaS are virtual machines, load balancers, and network attached storage.

2. **Platform as a service (PAAS):** Here we have object storage, queuing, databases, runtime etc. All these we can get directly from the cloud provider. It's

our responsibility to configure and use that. Providers will give us the resources but connectivity to our database and other similar activities are our responsibility.

Examples of PaaS are Windows Azure and Google App Engine (GAE).

3. **Applications or software as a service** (SAAS) ex. Salesforce.com, dropbox, google drive etc. Here we do not have any responsibility. We are using the application that is running on the cloud. All infrastructure setup is the responsibility of the service provider.

For SaaS to work, the infrastructure (IaaS) and the platform (PaaS) must be in place.

Cloud for Big Data

Below are some examples of how cloud applications are used for Big Data:

IAAS in a public cloud: Using a cloud provider's infrastructure for Big Data services, gives access to almost limitless storage and compute power.
IaaS can be utilized by enterprise customers to create cost-effective and easily scalable IT solutions where cloud providers bear the complexities and expenses of managing the underlying hardware.

If the scale of a business customer's operations fluctuates, or they are looking to expand, they can tap into the cloud resource as and when they need it rather than purchase, install and integrate hardware themselves.

PAAS in a private cloud: PaaS vendors are beginning to incorporate Big Data technologies such as Hadoop and MapReduce into their PaaS offerings, which eliminate the dealing with the complexities of managing individual software and hardware elements.
For example, web developers can use individual PaaS environments at every stage of development, testing and ultimately hosting their websites. However, businesses that are developing their own internal software can also utilize Platform as a Service, particularly to create distinct ring-fenced development and testing environments.

SAAS in a hybrid cloud: Many organizations feel the need to analyze the customer's voice, especially on social media. SaaS vendors provide the platform for the analysis as well as the social media data.

Office software is the best example of businesses utilizing SaaS. Tasks related to accounting, sales, invoicing, and planning can all be performed through SAAS.

Businesses may wish to use one piece of software that performs all of these tasks or several that each performs different tasks.

The software can be subscribed through the internet and then accessed online via any computer in the office using a username and password. If needed, they can switch to software that fulfills their requirements in better manner.

Everyone who needs access to a particular piece of software can be set up as a user, whether it is one or two people or every employee in a corporation that employs hundreds.

Providers in the Big Data Cloud Market

In addition there are many startups that have interesting products in cloud space. Here we have a list of major vendors of cloud computing. Few of the cloud providers are google, citrix, netmagic, redhat, rackspace etc. Amazon (aws) is the leading cloud provider amongst all. Microsoft is also providing cloud services and it is called as azure.

Infrastructure as a Service cloud computing companies:

- Amazon's offerings include S3 (Data storage/file system), SimpleDB (non-relational database) and EC2 (computing servers).
- Rackspace's offerings include Cloud Drive (Data storage/file system), Cloud Sites (web site hosting on cloud) and Cloud Servers(computing servers).
- IBM's offerings include Smart Business Storage Cloud and Computing on Demand (CoD).
- AT&T's provides Synaptic Storage and Synaptic Compute as a service.

Platform as a Service cloud computing companies

- Googles AppEngine is a development platform that is built upon Python and Java.
- com's provides a development platform that is based upon Apex.
- Microsoft Azure provides a development platform based upon .Net.

Software as a Service companies

- In SaaS, Google provides space that includes Google Docs, Gmail, Google Calendar and Picasa.
- IBM provides LotusLive iNotes, a web-based email service for messaging and calendaring capabilities to business users.
- Zoho provides online products similar to Microsoft office suite.

Predictive Analytics Moves into the Limelight

What is predictive analyticts?

Predictive analytics is a branch of advanced analytics that makes predictions about future outcomes using historical data combined with statistical modeling, data mining techniques and machine learning. Companies employ predictive analytics to find patterns in this data to identify risks and opportunities. Predictive analytics is often associated with big data and data science.

predictive analytics models are designed to assess historical data, discover patterns, observe trends, and use that information to predict future trends.

Predictive analytics can be deployed in across various industries for different business problems. Below are a few industry use cases to illustrate how predictive analytics can inform decision-making within real-world situations.

- **Banking:** Financial services use machine learning and quantitative tools to predict credit risk and detect fraud. Predictive analytics allows them to support dynamic market changes in real-time in addition to static market constraints. This use of technology allows it to both customize personal services for clients and to minimize risk.
- **Healthcare:** Predictive analytics in health care is used to detect and manage the care of chronically ill patients, as well as to track specific infections such as sepsis. Geisinger Health used predictive analytics to mine health records to learn more about how sepsis is diagnosed and treated. Geisinger created a predictive model based on health records for more than 10,000 patients who had been diagnosed with sepsis in the past. The model yielded impressive results, correctly predicting patients with a high rate of survival.
- **Human resources (HR):** HR teams use predictive analytics and employee survey metrics to match prospective job applicants, reduce employee turnover and increase employee engagement. This combination of quantitative and qualitative data allows businesses to reduce their recruiting costs and increase employee satisfaction, which is particularly useful when labor markets are volatile.
- **Marketing and sales:** While marketing and sales teams are very familiar with business intelligence reports to understand historical sales performance, predictive analytics enables companies to be more proactive in the way that they engage with their clients across the customer lifecycle. For example, churn predictions can enable sales teams to identify dissatisfied clients sooner, enabling them to initiate conversations to promote retention. Marketing teams can leverage predictive data analysis for cross-sell strategies, and this commonly manifests itself through a recommendation engine on a brand's website.
- **Supply chain:** Businesses commonly use predictive analytics to manage product inventory and set pricing strategies. This type of predictive analysis helps companies meet customer demand without overstocking warehouses. It also enables companies to assess the cost and return on their products over time. If one part of a given product becomes more expensive to import, companies can project the long-term impact on revenue if they do or do not pass on additional costs to their customer base. For a deeper look at a case study, you can read more about how FleetPride used this type of data analytics to inform their decision making on their inventory of parts for excavators and tractor trailers. Past shipping orders enabled them to plan more precisely to set appropriate supply thresholds based on demand.

Benefits of predictive modelling

- **Security:** Every modern organization must be concerned with keeping data secure. A combination of automation and predictive analytics improves security. Specific patterns associated with suspicious and unusual end user behavior can trigger specific security procedures.
- **Risk reduction:** In addition to keeping data secure, most businesses are working to reduce their risk profiles. For example, a company that extends credit can use data analytics to better understand if a customer poses a higher-than-average risk of defaulting. Other companies may use predictive analytics to better understand whether their insurance coverage is adequate.
- **Operational efficiency**: More efficient workflows translate to improved profit margins. For example, understanding when a vehicle in a fleet used for delivery is going to need maintenance before it's broken down on the side of the road means deliveries are made on time, without the additional costs of having the vehicle towed and bringing in another employee to complete the delivery.
- **Improved decision making:** Running any business involves making calculated decisions. Any expansion or addition to a product line or other form of growth requires balancing the inherent risk with the potential outcome. Predictive analytics can provide insight to inform the decision-making process and offer a competitive advantage.

Applications of predictive analyticts:

Fraud Detection

Financial services can use predictive analytics to examine transactions, trends, and patterns. If any of this activity appears irregular, an institution can investigate it for fraudulent activity. This may be done by analyzing activity between bank accounts or analyzing when certain transactions occur.

Predictive Analytics vs. Machine Learning

A common misconception is that predictive analytics and machine learning are the same things. Predictive analytics help us understand possible future occurrences by analyzing the past. At its core, predictive analytics includes a series of statistical techniques (including machine learning, predictive modeling, and data mining) and uses statistics (both historical and current) to estimate, or predict, future outcomes.

Credit

Credit scoring makes extensive use of predictive analytics. When a consumer or business applies for credit, data on the applicant's credit history and the credit record of borrowers with similar characteristics are used to predict the risk that the applicant might fail to perform on any credit extended.

Underwriting

Data and predictive analytics play an important role in underwriting. Insurance companies examine policy applicants to determine the likelihood of having to pay out for a future claim based on the current risk pool of similar policyholders, as well as past events that have resulted in payouts. Predictive models that consider characteristics in comparison to data about past policyholders and claims are routinely used by actuaries.

Marketing

Individuals who work in this field look at how consumers have reacted to the overall economy when planning on a new campaign. They can use these shifts in demographics to determine if the current mix of products will entice consumers to make a purchase.

Active traders, meanwhile, look at a variety of metrics based on past events when deciding whether to buy or sell a security. Moving averages, bands, and breakpoints are based on historical data and are used to forecast future price movements.

Supply Chain

Supply chain analytics is used to predict and manage inventory levels and pricing strategies. Supply chain predictive analytics use historical data and statistical models to forecast future supply chain performance, demand, and potential disruptions.

This helps businesses proactively identify and address risks, optimize resources and processes, and improve decision-making. These steps allow companies to forecast what materials will be on hand at any given moment and whether there will be any shortages.

Human Resources

Human resources uses predictive analytics to improve various processes, such as forecasting future workforce needs and skills requirements or analyzing employee data to identify factors that contribute to high turnover rates.

Predictive analytics can also analyze an employee's performance, skills, and preferences to predict their career progression and help with career

development planning in addition to forecasting diversity or inclusion initiatives.

Types of Predictive Analytical Models

There are three common techniques used in predictive analytics: Decision trees, neural networks, and regression. Read more about each of these below.

Decision Trees

If you want to understand what leads to someone's decisions, then you may find decision trees useful. This type of model places data into different sections based on certain variables, such as price or market capitalization. Just as the name implies, it looks like a tree with individual branches and leaves. Branches indicate the choices available while individual leaves represent a particular decision.

Decision trees are the simplest models because they're easy to understand and dissect. They're also very useful when you need to make a decision in a short period of time.

Regression

This is the model that is used the most in statistical analysis. Use it when you want to determine patterns in large sets of data and when there's a linear relationship between the inputs. This method works by figuring out a formula, which represents the relationship between all the inputs found in the dataset. For example, you can use regression to figure out how price and other key factors can shape the performance of a security.

Neural Networks

Neural networks were developed as a form of predictive analytics by imitating the way the human brain works. This model can deal with complex data relationships using artificial intelligence and pattern recognition. Use it if you have several hurdles that you need to overcome like when you have too much data on hand, when you don't have the formula you need to help you find a relationship between the inputs and outputs in your dataset, or when you need to make predictions rather than come up with explanations.

Cluster Models

Clustering describes the method of aggregating data that share similar attributes. Consider a large online retailer like Amazon.

Amazon can cluster sales based on the quantity purchased or it can cluster sales based on the average account age of its consumer. By separating data into similar groups based on shared features, analysts may be able to identify other characteristics that define future activity.

Time Series Modeling

Sometimes, data relates to time, and specific predictive analytics rely on the relationship between what happens when. These types of models assess inputs at specific frequencies such as daily, weekly, or monthly iterations. Then, analytical models seek seasonality, trends, or behavioral patterns based on timing. This type of predictive model can be useful to predict when peak customer service periods are needed or when specific sales will be made.

What Is the Best Model for Predictive Analytics?

The best model for predictive analytics depends on several factors, such as the type of data, the objective of the analysis, the complexity of the problem, and the desired accuracy of the results. The best model to choose from may range from linear regression, neural networks, clustering, or decision trees.

- Predictive analytics uses statistics and modeling techniques to determine future performance.
- Industries and disciplines, such as insurance and marketing, use predictive techniques to make important decisions.
- Predictive models help make weather forecasts, develop video games, translate voice-to-text messages, customer service decisions, and develop investment portfolios.
- People often confuse predictive analytics with machine learning even though the two are different disciplines.
- Types of predictive models include decision trees, regression, and neural networks.

Popular predictive analytics models include classification, clustering, and time series models.

- Recommendation engines similar to those used in Netflix and Amazon that use past purchases and buying behavior to recommend new purchases.
- Risk engines for a wide variety of business areas, including market and credit risk, catastrophic risk, and portfolio risk.
- Innovation engines for new product innovation, drug discovery, and consumer and fashion trends to predict potential new product formulations and discoveries.
- Customer insight engines that integrate a wide variety of customer related info, including sentiment, behavior, and even emotions.
- Customer insight engines will be the backbone in online and set-top box advertisement targeting, customer loyalty programs to maximize customer lifetime value, optimizing marketing campaigns for revenue lift, and targeting individuals or companies at the right time to maximize their spend.
- Optimization engines that optimize complex interrelated operations and decisions that are too overwhelming for people to systematically handle at scales, such as when, where, and how to seek natural resources to maximize output while reducing operational costs— or what potential competitive strategies should be used in a global business that takes into account the various political, economic, and competitive pressures along with both internal and external operational capabilities.

Software as a Service BI

- Software-as-a-Service Business Intelligence (SaaS BI) is a business intelligence (BI) delivery model in which applications are implemented outside of a company and usually employed at a hosted location accessed by an end user via protected Internet access.
- SaaS BI generally implies a pay-as-you-go or subscription model, versus the conventional software licensing model with annual maintenance or license fees.
- SaaS BI is also known as cloud BI or on-demand BI.
- SaaS BI allows organizations to use BI tools without on-site installation or maintenance, allowing customers to concentrate on generating analytic queries and BI reports, rather than unnecessary tasks. The SaaS BI approach also allows organizations to broaden their BI systems as usage

is increased. Heavy equipment purchases are not required because there are no on-premise deployments.

- SaaS BI can be a fit if there is no available budget to purchase BI software or related hardware. Because there is no upfront purchase expense or extra staffing demands for handling the BI system, the total cost of ownership (TCO) may be much lower than procedures involving traditional on-premise software.

Our first question for James was why his business was so successful: In addition to the Omniture people, several other reasons stand out to me. They include:

- **Scaling the SaaS delivery model.** We built Omniture from the ground up to be SaaS and we understood the math better than the competition. We invented a concept called the Magic Number.The Magic Number helps you look at your SaaS business and helps you understand the value you are creating when standard GAAP accounting numbers would lead you to believe the opposite.

■ **Killer sales organization.** Once we had a few well-known customers like HP, eBay, and Gannett, we stepped on the pedal from a competitive standpoint and really went to battle against the other sales organizations and we won. We focused the whole company on sales.

A focus on customer success. We had 98 percent retention rate. Customer happiness and success were always fi rst because in a SaaS business, unlike traditional enterprise software, it ’s too easy for customers to leave if they are not happy. James explained the three market reasons why he started Domo, knowing we had to fix three problems in traditional BI. Here is a summary in his own words:

1**. Relieving the IT choke point.** Removing the friction for BI to become useful and enabling IT to be more strategic by enabling self-service BI.

2**. Transforming BI from cost center to a revenue generator.** Addresses a very common frustration that I ’ve experienced as a CEO and that other CEOs have shared with me . . . now that we ’ve invested in capturing all this data—how do we benefi t from it?

3. **The user experience.** Is where we are putting all our marbles. Today ’s BI is not designed for the end user. It ’s not intuitive, it ’s not accessible, it ’s not real time, and it doesn ’t meet the expectations of today ’s consumers of

technology, who expect a much more connected experience than enterprise software delivers.

We ’ll deliver an experience with BI that redefines BI and is unlike anything seen to date.

Mobile Business Intelligence Is Going Mainstream

- 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.

Ease of Mobile Application Deployment

Three elements that have impacted the viability of mobile BI:

1. Location—the GPS component and location . . . know where you are in time as well as the movement.
2. It’s not just about pushing data; you can transact with your smart phone based on information you get.
3. Multimedia functionality allows the visualization pieces to really come into play.

Three challenges with mobile BI include:

1. Managing standards for rolling out these devices.
2. Managing security (always a big challenge).
3. Managing “bring your own device,” where you have devices both owned by the company and devices owned by the individual, both contributing to productivity.

Crowdsourcing Analytics

What is crowdsourcing in data analytics?

What Is Crowdsourcing? Crowdsourcing involves **obtaining work, information, or opinions from a large group of people who submit their data via the Internet, social media, and smartphone apps**. People involved in crowdsourcing sometimes work as paid freelancers, while others perform small tasks voluntarily.

In October 2006, Netflix, an online DVD rental business, announced a contest to create a new predictive model for recommending movies based on past user ratings.

- Netflix already had an algorithm to solve the problem but thought there was an opportunity to realize additional model "lift," which would translate to huge top-line revenue.

- Netflix was an innovator in a space now being termed crowdsourcing. Crowdsourcing is a recognition that you can 't possibly always have the best and brightest internal people to solve all your big problems.

- By creating an open, competitive environment with clear rules and goals, Netflix realized their objective and, yes, they did create a lot of buzz about their organization in the process.

- Crowdsourcing is a great way to capitalize on the resources that can build algorithms and predictive models.

- Kaggle describes itself as "an innovative solution for statistical/analytics outsourcing." That 's a very formal way of saying that Kaggle manages competitions among the world 's best data scientists.

How it works?

- Corporations, governments, and research laboratories are confronted with complex statistical challenges.
- They describe the problems to Kaggle and provide data sets.
- Kaggle converts the problems and the data into contests that are posted on its web site.
- The contests feature cash prizes ranging in value from $100 to $3 million.

- Kaggle ’s clients range in size from tiny start-ups to multinational corporations such as Ford Motor Company and government agencies such as NASA.

Inter- and Trans-Firewall Analytics

In the health care industry, rich consumer insights can be generated by collaborating on data and insights from the health insurance provider, pharmacy delivering the drugs, and the drug manufacturer.

In-fact, this is not necessarily limited to companies within the traditional demand-supply value chain.

For example, there are instances where a retailer and a social media company can come together to share insights on consumer behavior that will benefit both players.

Some of the more progressive companies are taking this a step further and working on leveraging the large volumes of data outside the fi rewall such as social data, location data, and so forth.

In other words, it will be not very long before internal data and insights from within the firewall is no longer a differentiator. We see this trend as the move from intra- to inter- and trans-firewall analytics.

Today they are doing intra-firewall analytics with data within the firewall. Tomorrow they will be collaborating on insights with other companies to do inter-firewall analytics as well as leveraging the public domain spaces to do trans-firewall analytics.

Organizations will need to complement just intra-firewall insights with inter- and trans-firewall analytics

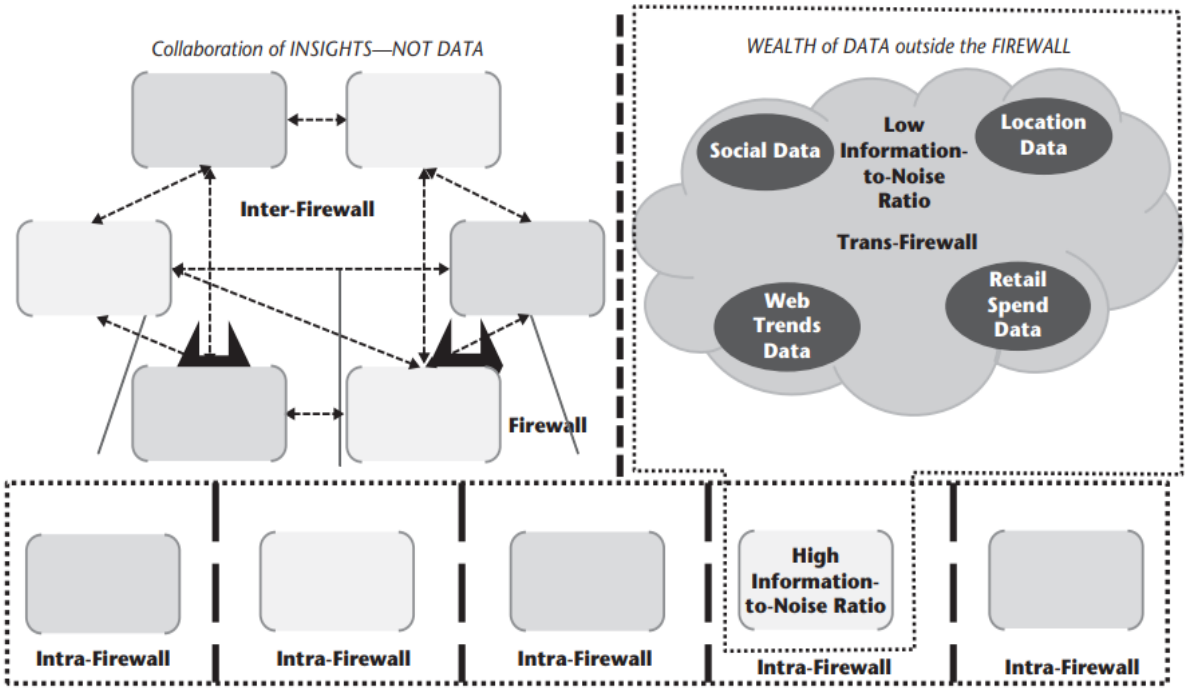


Figure 3.1 Inter- and Trans-Firewall Analytics
Source: Mu Sigma and author Ambiga is a cofounder.

Disruptive value and efficiencies can be extracted by cooperating and exploring outside the boundaries of the firewall

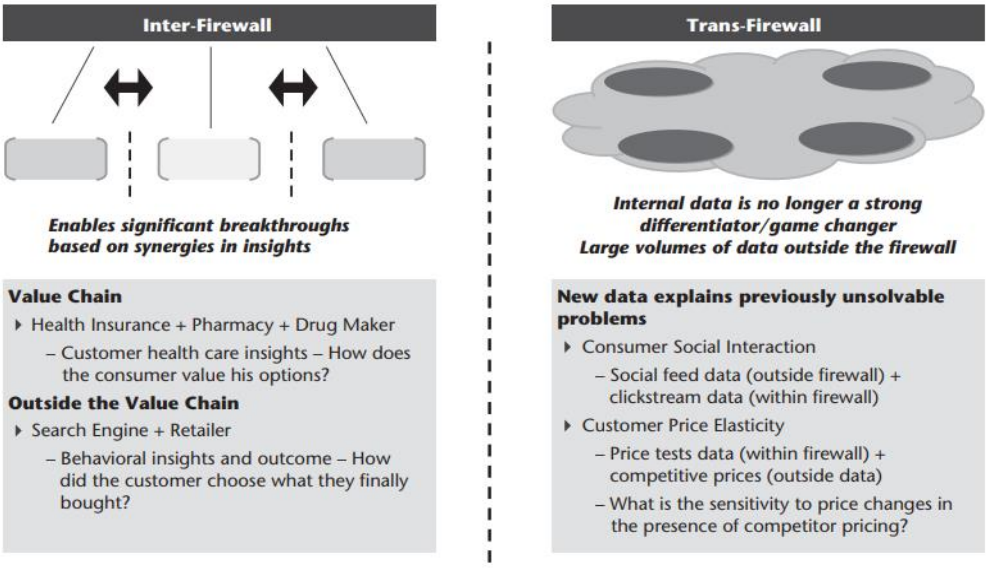


Figure 3.2 Value Chain for Inter-Firewall and Trans-Firewall Analytics
Source: Mu Sigma.

Challenges

- As one moves outside the firewall, the INR increases
- This gives rise to additional requirements on analytical methods and technology
- Fear of collaboration due to competitive fear, data privacy concerns, and proprietary orientations that limit opportunities for cross-organizational learning and innovation
- Though the transition to an inter- and trans-firewall analytics is not easy, it would soon become a key weapon available for the decision scientists

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

KG REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

B. Tech III Year II Semester **Academic Year: 2023-2024**

SUBJECT: BIG DATA ANALYTICTS **SUBJECT CODE: KG21CD605** **REGULATION: KGR21**

UNIT-3 NOTES

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

Apache Hadoop and Hadoop Echo system

What is Hadoop?

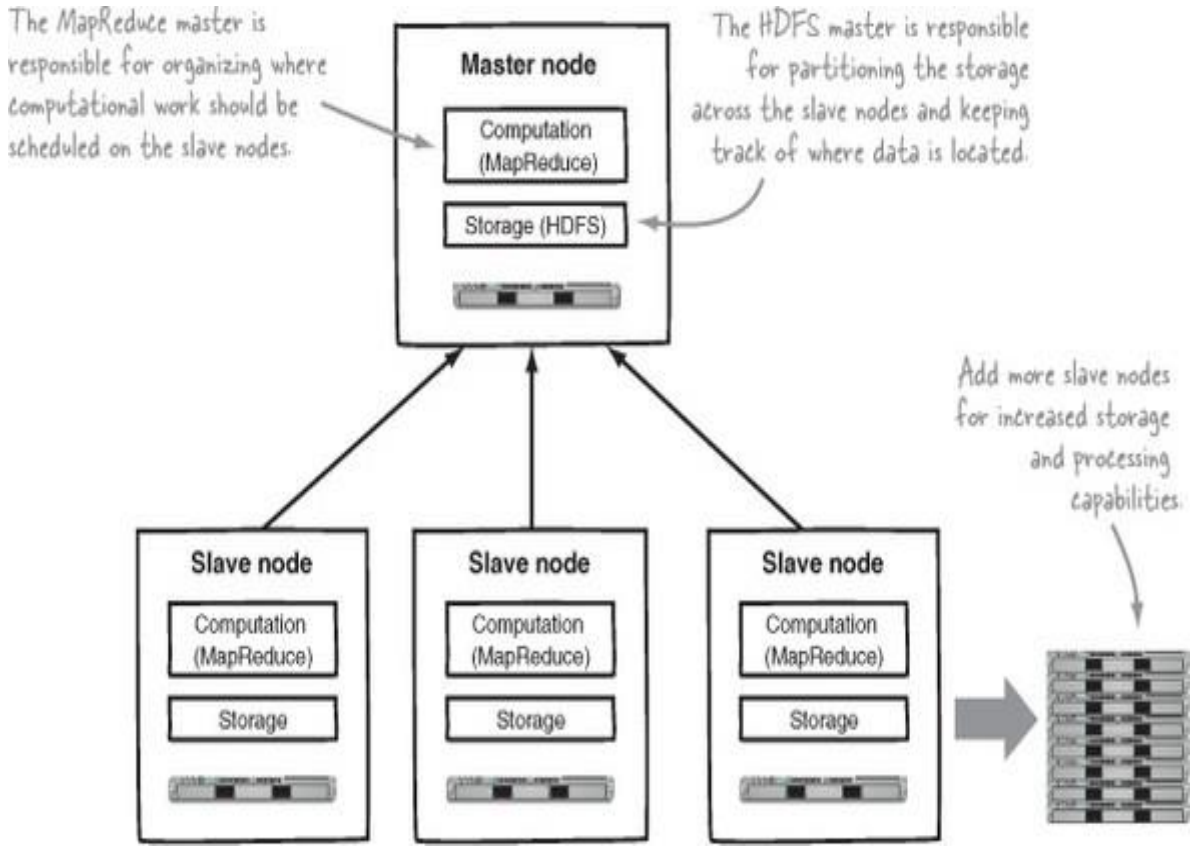
Hadoop is a platform that provides both distributed storage and computational capabilities. Hadoop was first conceived to fix a scalability issue that existed in Nutch, an open source crawler and search engine.

At the time Google had published papers that described its novel distributed filesystem, the Google File System (GFS), and Map-Reduce, a computational framework for parallel processing.

The successful implementation of these papers' concepts in Nutch resulted in its split into two separate projects, the second of which became Hadoop, a first-class Apache project.

The Nutch project, and by extension Hadoop, was led by Doug Cutting and Mike Cafarella.

Figure High-level Hadoop architecture



Core Hadoop components

To understand Hadoop's architecture we'll start by looking at the basics of HDFS.

HDFS

HDFS is the storage component of Hadoop. It's a distributed filesystem that's modeled after the Google File System (GFS) paper

HDFS replicates files for a configured number of times, is tolerant of both software and hardware failure, and automatically re-replicates data blocks on nodes that have failed.

shows a logical representation of the components in HDFS: the Name-Node and the DataNode.

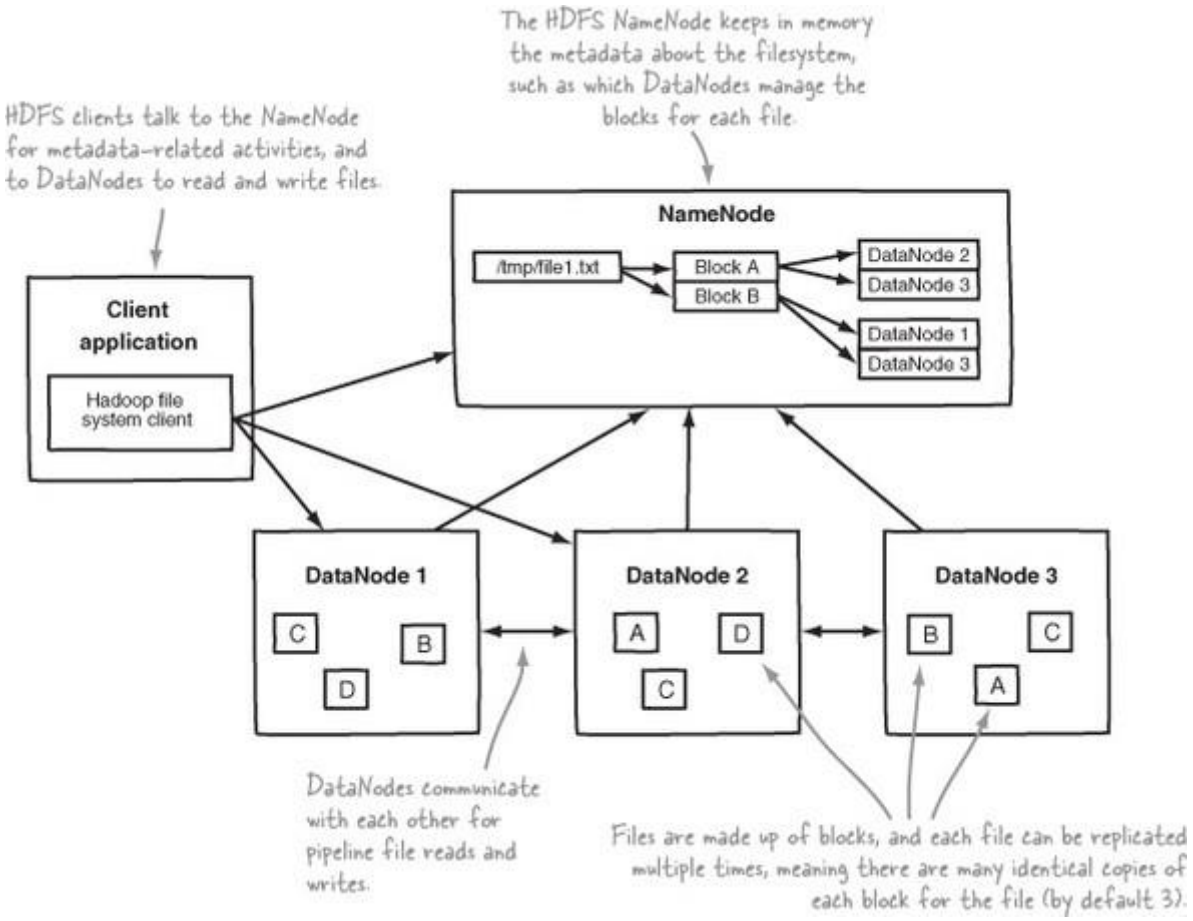
It also shows an application that's using the Hadoop filesystem library to access HDFS.

Mapreduce

MapReduce is a batch-based, distributed computing framework modeled after Google's paper on MapReduce.

It allows you to parallelize work over a large amount of raw data, such as combining web logs with relational data from an OLTP database to model how users interact with your website.

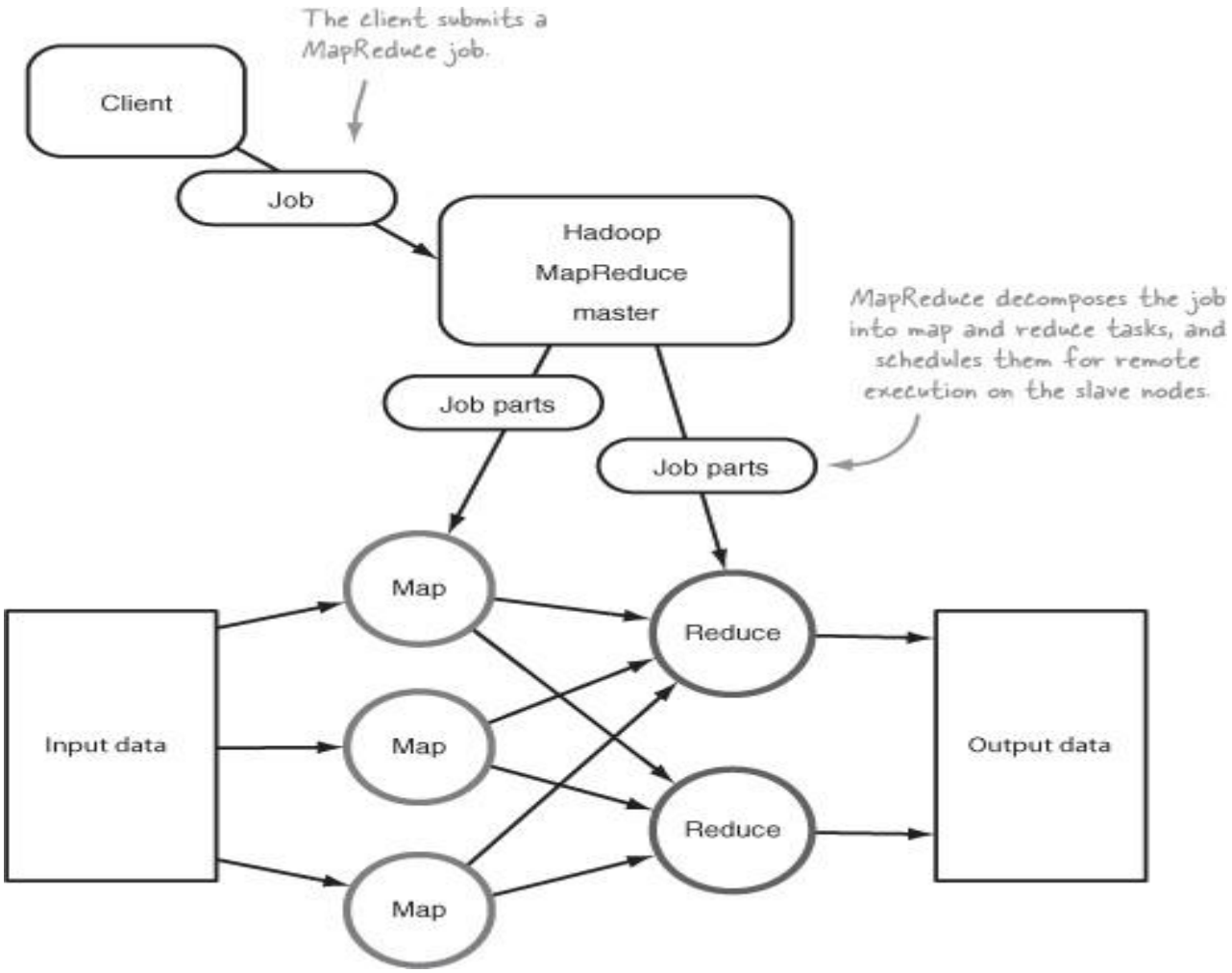
This type of work, which could take days or longer using conventional serial programming techniques, can be reduced down to minutes using MapReduce on a Hadoop cluster.



A client application submits a job to master node i.e name node.The text file is deployed on the 2 blocks Block A and Block B.These blocks of data are placed on Data Nodes. On these data nodes the 3 copies of data are maintained. If there is hardware failure or data corruption can retrieve the data from the other nodes.

Figure 1.3. HDFS architecture shows an HDFS client communicating with the master Name Node and slave Data Nodes.

A client submitting a job to MapReduce



The role of the programmer is to define map and reduce functions, where the map function outputs key/value tuples, which are processed by reduce functions to produce the final output.

MapReduce's shuffle and sort

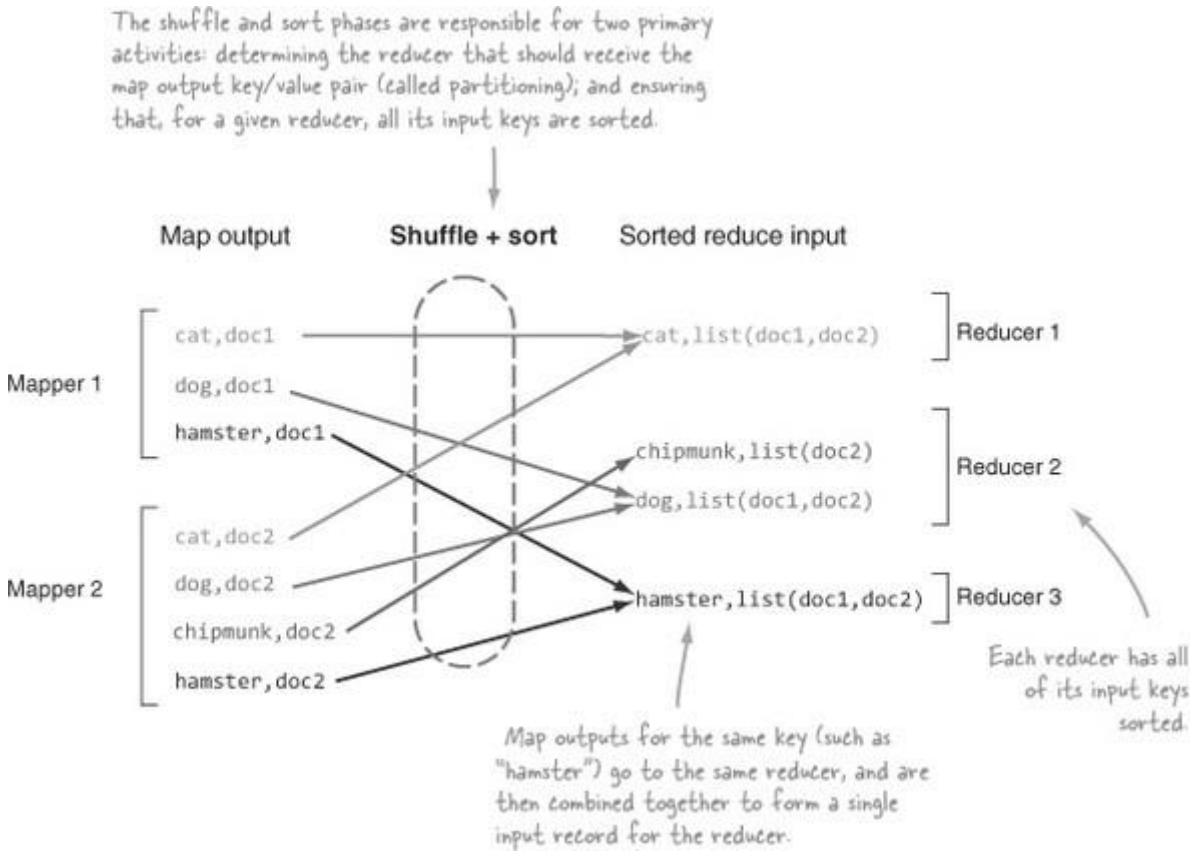


Figure A logical view of the reduce function

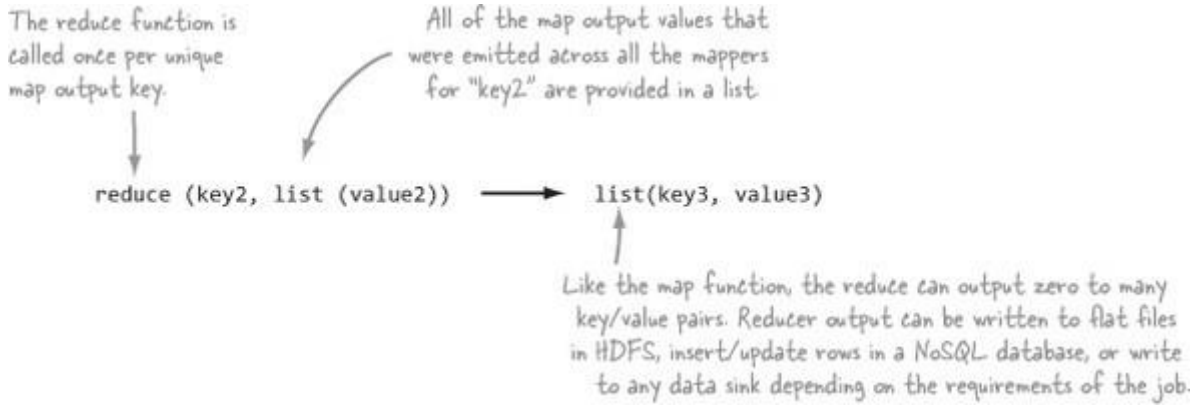
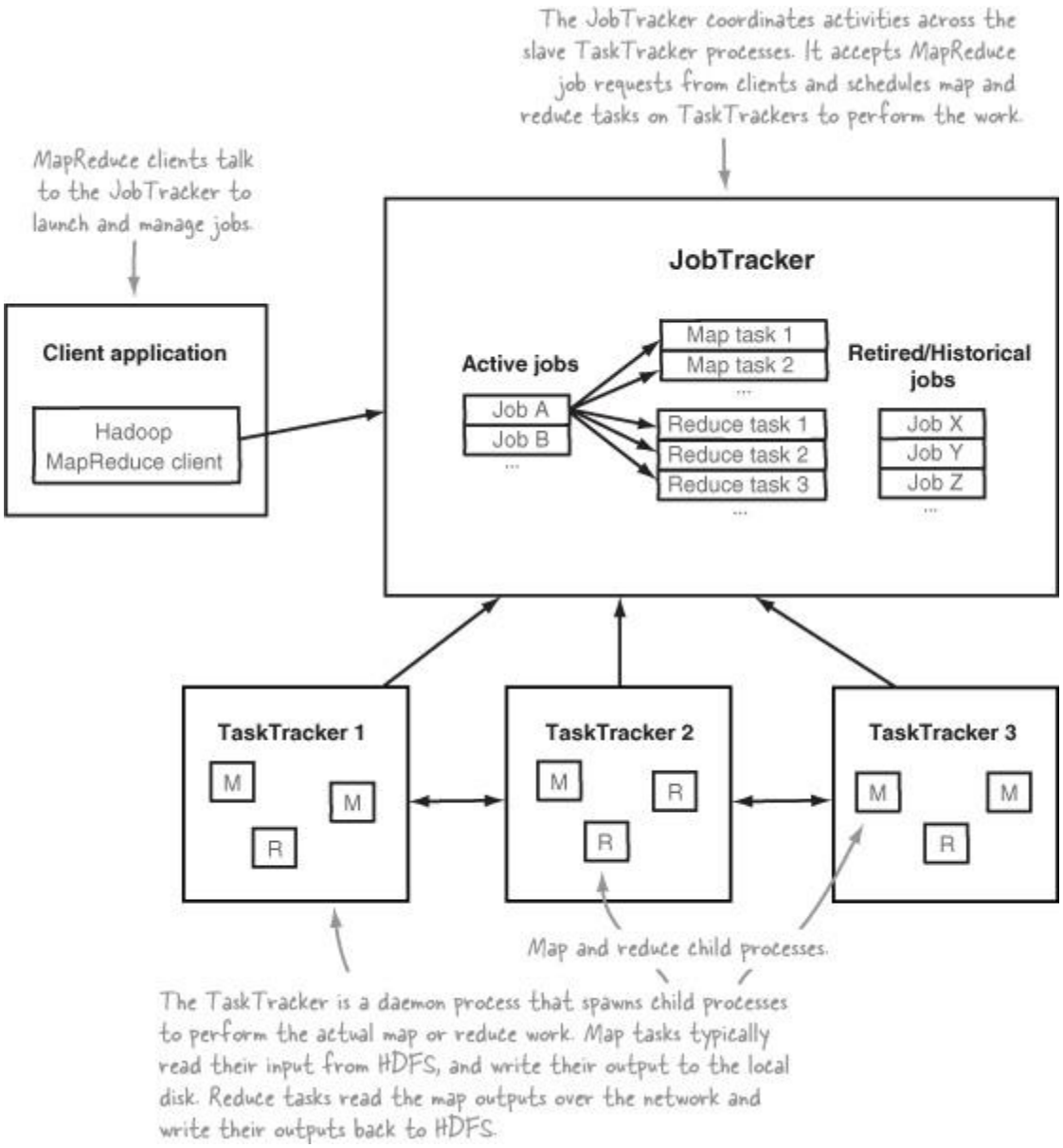


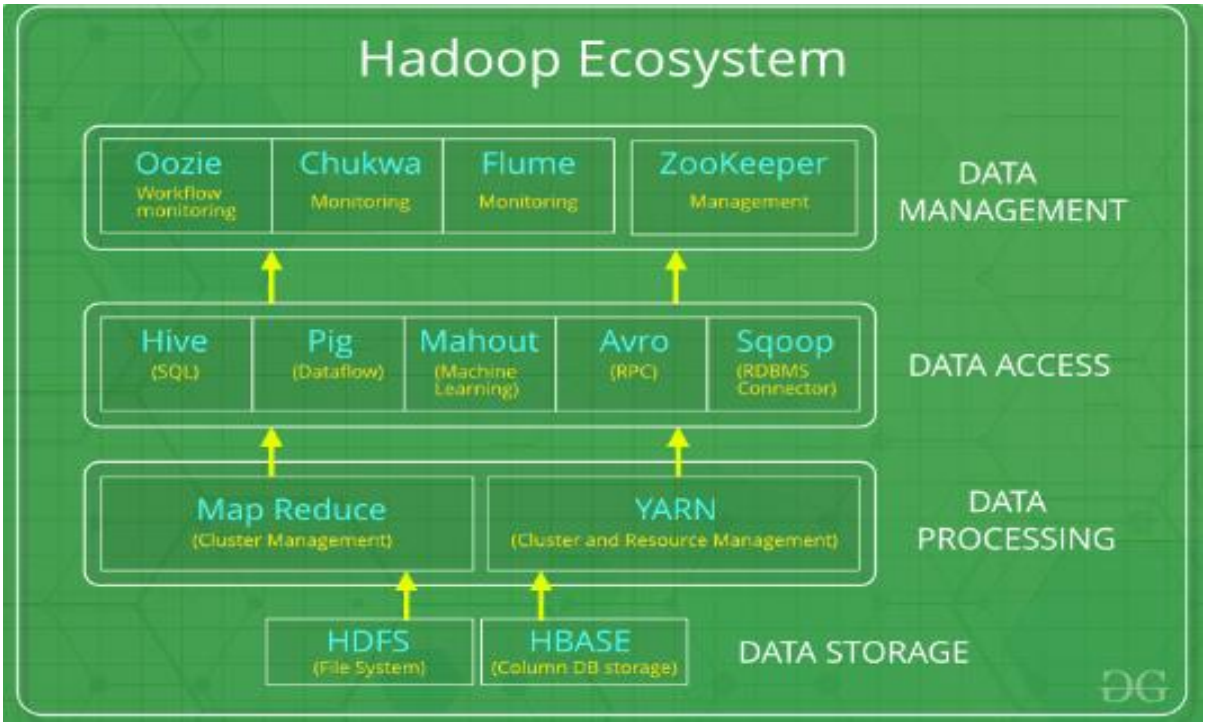
Figure MapReduce logical architecture



The Hadoop ecosystem

The Hadoop ecosystem is diverse and grows by the day. It's impossible to keep track of all of the various projects that interact with Hadoop in some form

Figure Hadoop and related technologies



Hadoop Ecosystem is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions.

There are *four major elements of Hadoop* i.e. **HDFS, MapReduce, YARN, and Hadoop Common**. Most of the tools or solutions are used to supplement or support these major elements.

All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

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.
- HDFS consists of two core components i.e.
    1. Name node
    2. 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.

YARN:

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
    1. Resource Manager
    2. Nodes Manager
    3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas 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.

MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
    1. *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.
    2. *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.

PIG:

Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.

- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- 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.
- 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.

HIVE:

- 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.

Mahout:

- 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.

ApacheSpark:

- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.

- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

ApacheHBase:

- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. 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

Other Components: Apart from all of these, there are some other components too that carry out a huge task in order to make Hadoop capable of processing large datasets. They are as follows:

- **Solr, Lucene:** These are the two services that perform the task of searching and indexing with the help of some java libraries, especially Lucene is based on Java which allows spell check mechanism, as well. However, Lucene is driven by Solr.
- **Zookeeper:** There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie:** Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

Apache Avro

It is a row-oriented remote procedure call and
data serialization framework developed within Apache's Hadoop project. It
uses JSON for defining data types and protocols, and serializes data in a compact
binary format.

Its primary use is in Apache Hadoop, where it can provide both a serialization format for persistent data, and a wire format for communication between Hadoop nodes, and from client programs to the Hadoop services.

Avro uses a schema to structure the data that is being encoded. It has two different types of schema languages; one for human editing (Avro IDL) and another which is more machine-readable based on JSON.

Apache Sqoop

Apache Sqoop is part of the Hadoop ecosystem. Since a lot of the data had to be transferred from relational database systems onto Hadoop, there was a need for a dedicated tool to do this task fast. This is where Apache Sqoop came into the picture which is now extensively used for transferring data from **RDBMS files to the** Hadoop ecosystem for MapReduce processing and so on.

When it comes to transferring data, there is a certain set of requirements to be taken care of. It includes the following: Data has to have consistency; it should be prepared for provisioning the downstream pipeline, and the users should ensure the consumption of production system resources; among other things. The MapReduce application is not able to directly access the data that is residing in external relational databases. This method can expose the system to the risk of too much load generation from the cluster nodes.

Apache Chukwa

Apache Chukwa is an open source data collection system for monitoring large distributed systems.

Apache Chukwa is built on top of the Hadoop Distributed File System (HDFS) and Map/Reduce framework and inherits Hadoop's scalability and robustness.

Apache Chukwa also includes a flexible and powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data.

Apache Flume

Apache Flume is an open-source tool for collecting, aggregating, and moving huge amounts of streaming data from the external web servers to the central

store, say HDFS, HBase, etc. It is a highly available and reliable service which has tunable recovery mechanisms.

The main purpose of designing Apache Flume is to move streaming data generated by various applications to Hadoop Distributed FileSystem.

A company has millions of services that are running on multiple servers. Thus, produce lots of logs. In order to gain insights and understand customer behavior, they need to analyze these logs altogether.

In order to process logs, a company requires an extensible, scalable, and reliable distributed data collection service.

That service must be capable of performing the flow of unstructured data such as logs from source to the system where they will be processed (such as in Hadoop Distributed FileSystem). Flume is an open-source distributed data collection service used for transferring the data from source to destination.

It is a reliable, and highly available service for collecting, aggregating, and transferring huge amounts of logs into HDFS. It has a simple and flexible architecture.

Apache Flume is highly robust and fault-tolerant and has tunable reliability mechanisms for fail-over and recovery. It allows the collection of data collection in batch as well as in streaming mode.

Moving data in and out of Hadoop

Moving data in and out of Hadoop is as data ingress and egress, is the process by which data is transported from an external system into an internal system, and vice versa. Hadoop supports ingress and egress at a low level in HDFS and MapReduce.

Files can be moved in and out of HDFS, and data can be pulled from external data sources and pushed to external data sinks using MapReduce.

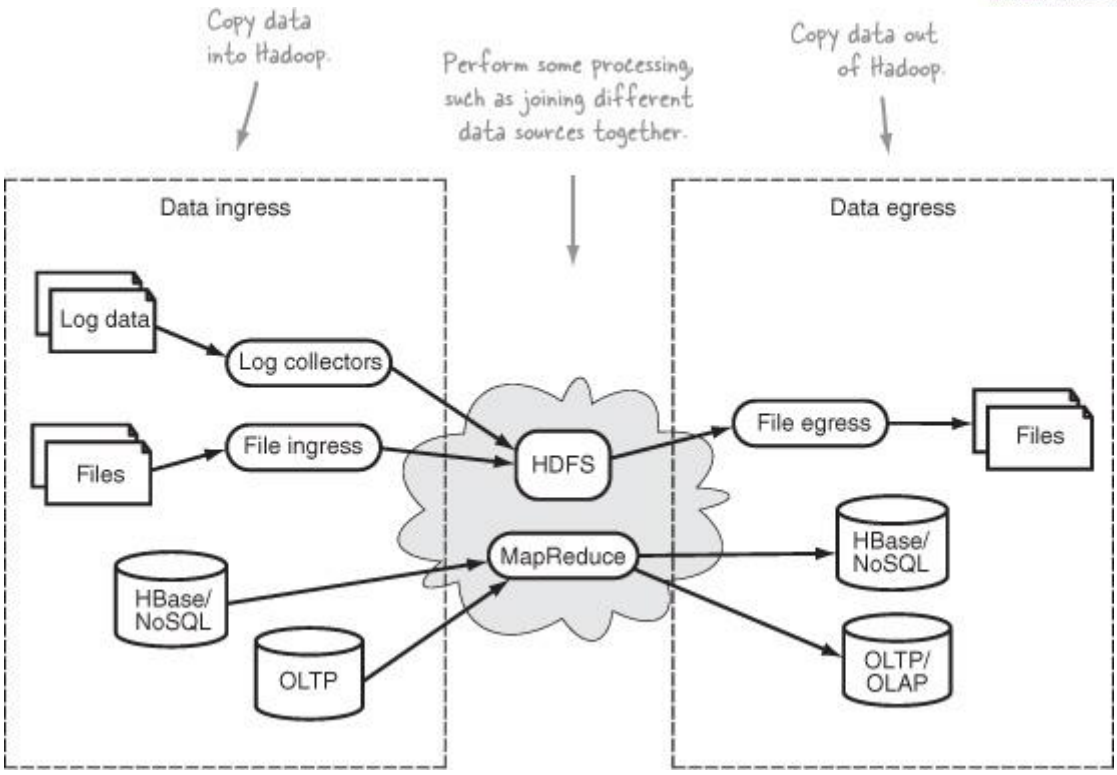


Figure :Hadoop data ingress and egress transports data to and from an external system to an internal one.

The fact that your data exists in various forms and locations throughout your environments complicates the process of ingress and egress.

How do you bring in data that's sitting in an OLTP (online transaction processing) database?

Or ingress log data that's being produced by tens of thousands of production servers? Or work with binary data sitting behind a firewall?

Further, how do you automate your data ingress and egress process so that your data is moved at regular intervals?

Automation is a critical part of the process, along with monitoring and data integrity responsibilities to ensure correct and safe transportation of data.

Key elements of ingress and egress

Idempotence

An idempotent operation produces the same result no matter how many times it's executed. In a relational database the inserts typically aren't idempotent,

because executing them multiple times doesn't produce the same resulting database state.

Alternatively, updates often are idempotent, because they'll produce the same end result.

Any time data is being written idempotence should be a consideration, and data ingress and egress in Hadoop is no different.

How well do distributed log collection frameworks deal with data retransmissions?

How do you ensure idempotent behavior in a MapReduce job where multiple tasks are inserting into a database in parallel.

Aggregation

- The data aggregation process combines multiple data elements.
- In the context of data ingress this can be useful because moving large quantities of small files into HDFS potentially translates into NameNode memory woes, as well as slow MapReduce execution times.
- Having the ability to aggregate files or data together mitigates this problem, and is a feature to consider.

Data Format Transformation

- The data format transformation process converts one data format into another.
- Often your source data isn't in a format that's ideal for processing in tools such as Map-Reduce.
- If your source data is multiline XML or JSON form, for example, you may want to consider a preprocessing step.
- This would convert the data into a form that can be split, such as a JSON or an XML element per line, or convert it into a format such as Avro.

Recoverability

- Recoverability allows an ingress or egress tool to retry in the event of a failed operation.

- Because it's unlikely that any data source, sink, or Hadoop itself can be 100 percent available, it's important that an ingress or egress action be retried in the event of failure.

Correctness

- In the context of data transportation, checking for correctness is how you verify that no data corruption occurred as the data was in transit.

- When you work with heterogeneous systems such as Hadoop data ingress and egress tools, the fact that data is being transported across different hosts, networks, and protocols only increases the potential for problems during data transfer.

- Common methods for checking correctness of raw data such as storage devices include Cyclic Redundancy Checks (CRC), which are what HDFS uses internally to maintain block-level integrity.

Resource Consumption and Performance

Resource consumption and performance are measures of system resource utilization and system efficiency, respectively. Ingress and egress tools don't typically incur significant load (resource consumption) on a system, unless you have appreciable data volumes.

For performance, the questions to ask include whether the tool performs ingress and egress activities in parallel, and if so, what mechanisms it provides to tune the amount of parallelism.

For example, if your data source is a production database, don't use a large number of concurrent map tasks to import data.

Monitoring

Monitoring ensures that functions are performing as expected in automated systems.

For data ingress and egress, monitoring breaks down into two elements: ensuring that the process(es) involved in ingress and egress are alive, and validating that source and destination data are being produced as expected.

Moving data into Hadoop

There are two primary methods that can be used for moving data into Hadoop: writing external data at the HDFS level (a data push), or reading external data at the MapReduce level (more like a pull).

Reading data in MapReduce has advantages in the ease with which the operation can be parallelized and fault tolerant.

Low-Level Hadoop Ingress Mechanisms

These mechanisms include Hadoop's Java HDFS API, WebHDFS, the new Hadoop 0.23 REST API, and MapReduce.

Pushing log files into Hadoop

Log data has long been prevalent across all applications, but with Hadoop came the ability to process the large volumes of log data produced by production systems.

Various systems produce log data, from network devices and operating systems to web servers and applications.

These log files all offer the potential for valuable insights into how systems and applications operate as well as how they're used. What unifies log files is that they tend to be in text form and line-oriented, making them easy to process.

Comparing Flume, Chukwa, and Scribe

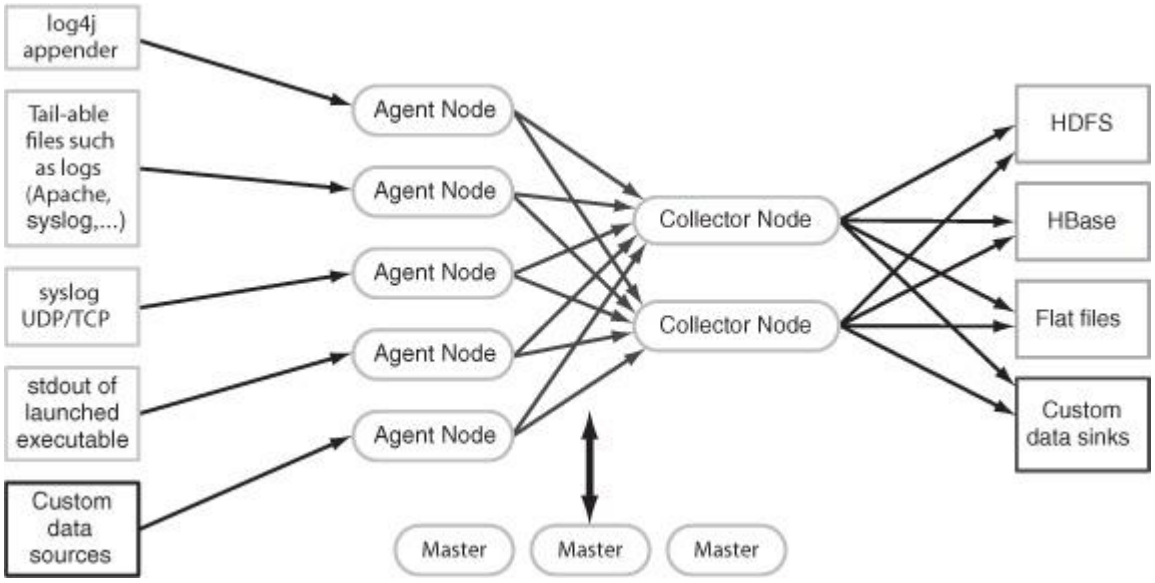
Flume, Chukwa, and Scribe are log collecting and distribution frameworks that have the capability to use HDFS as a data sink for that log data. It can be challenging to differentiate between them because they share the same features.

Flume

Apache Flume is a distributed system for collecting streaming data. It's an Apache project in incubator status, originally developed by Cloudera.

It offers various levels of reliability and transport delivery guarantees that can be tuned to your needs. It's highly customizable and supports a plugin architecture where you can add custom data sources and data sinks.

Flume architecture for collecting streaming data



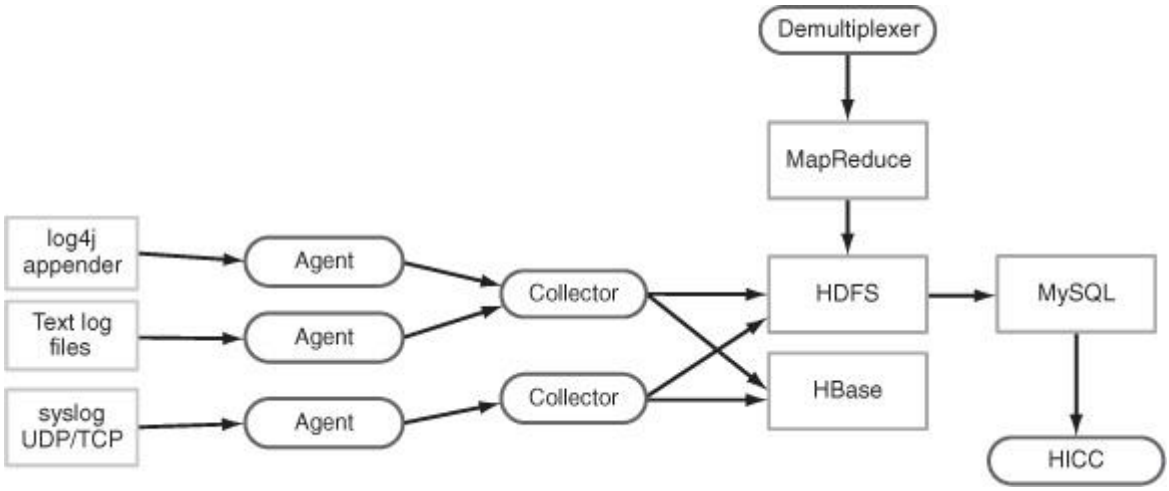
Chukwa

- Chukwa is an Apache subproject of Hadoop that also offers a large-scale mechanism to collect and store data in HDFS.

- And it's also in incubator status. Chukwa's reliability model supports two levels: end-to-end reliability, and fast-path delivery, which minimizes latencies.

- After writing data into HDFS Chukwa runs a MapReduce job to demultiplex the data into separate streams. Chukwa also offers a tool called Hadoop Infrastructure Care Center (HICC), which is a web interface for visualizing system performance.

Chukwa architecture for collecting and storing data in HDFS



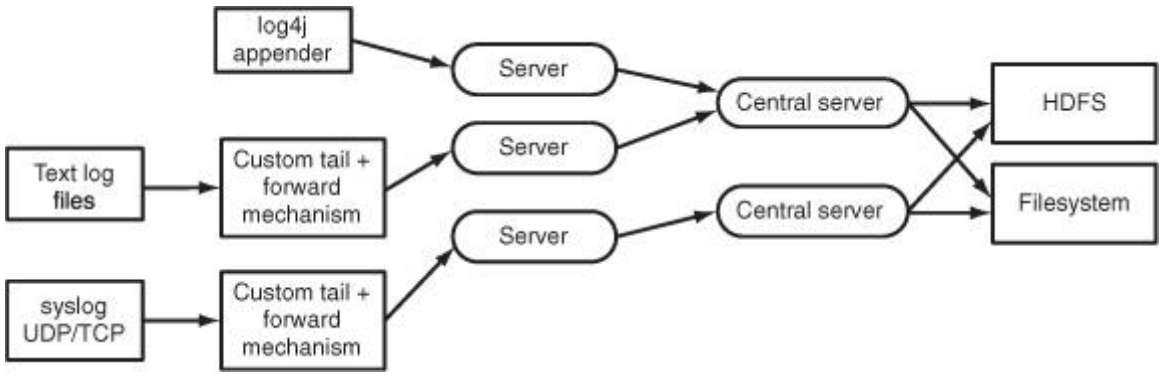
Scribe

Scribe is a rudimentary streaming log distribution service, developed and used heavily by Facebook.

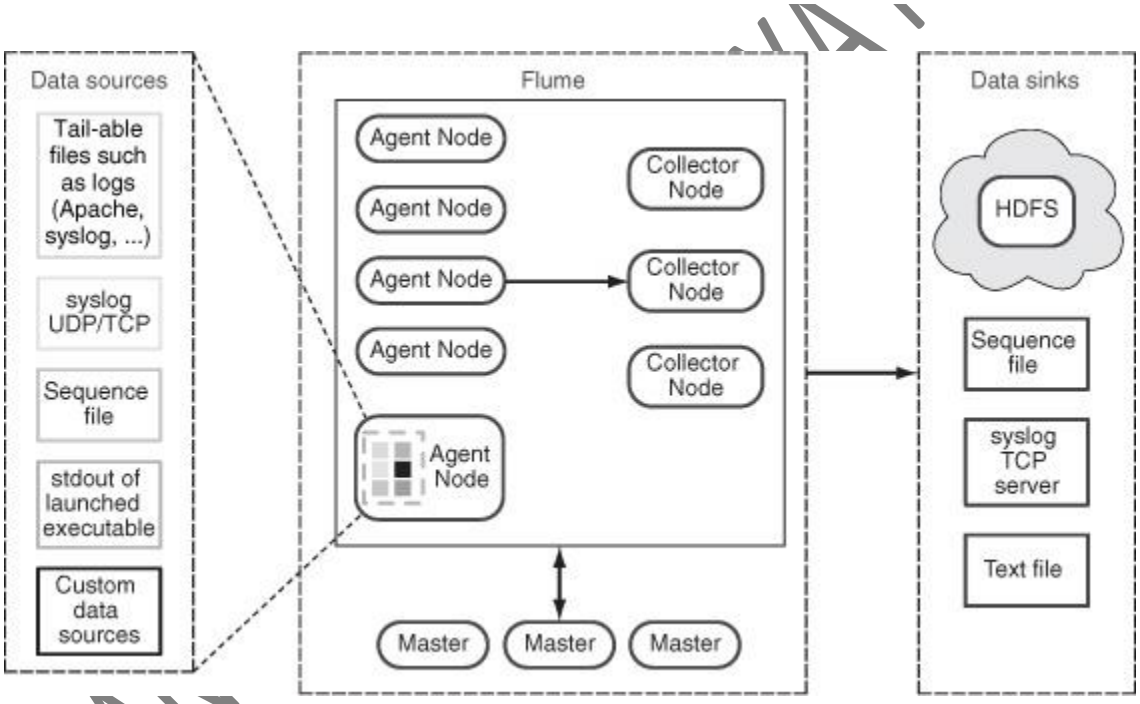
A scribe server that collects logs runs on every node and forwards them to a central Scribe server. Scribe supports multiple data sinks, including HDFS, regular filesystems, and NFS.

Scribe's reliability comes from a file-based mechanism where the server persists to a local disk in the event it can't reach the downstream server.

Scribe architecture also pushes log data into HDFS.



Pushing system log messages into HDFS with Flume



Example of Flume deployment for collecting streaming data

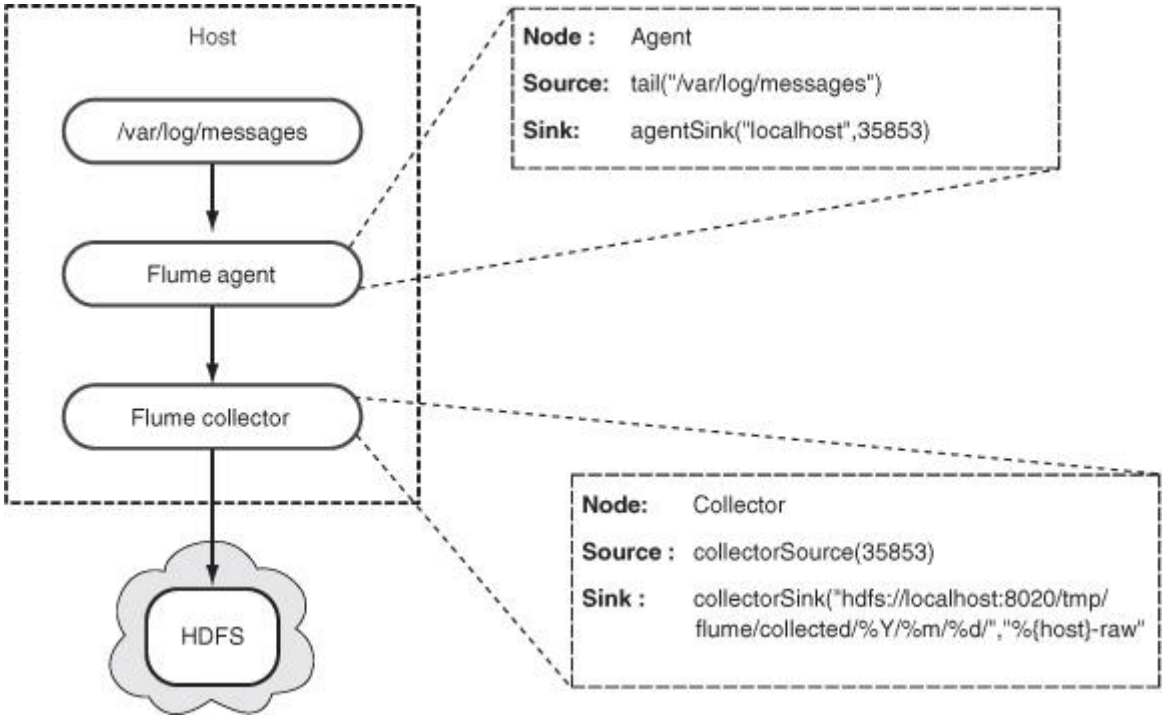
The primary components of the Flume are

- *Nodes*—Flume data paths that ferry data from a data source to a data sink. Agents and Collectors are simply Flume Nodes that are deployed in a way to efficiently and reliably work with a large number of data sources.

- *Agents*—Collect streaming data from the local host and forward it to the Collectors.
- *Collectors*—Aggregate data sent from the Agents and write that data into HDFS.
- *Masters*—Perform configuration management tasks and also help with reliable data flow.
-

Examples include application logs and Linux system logs, as well as nontext data that can be supported with custom data sources.

Data sinks are the destination of that data, which can be HDFS, flat files, and any data target that can be supported with custom data sinks.

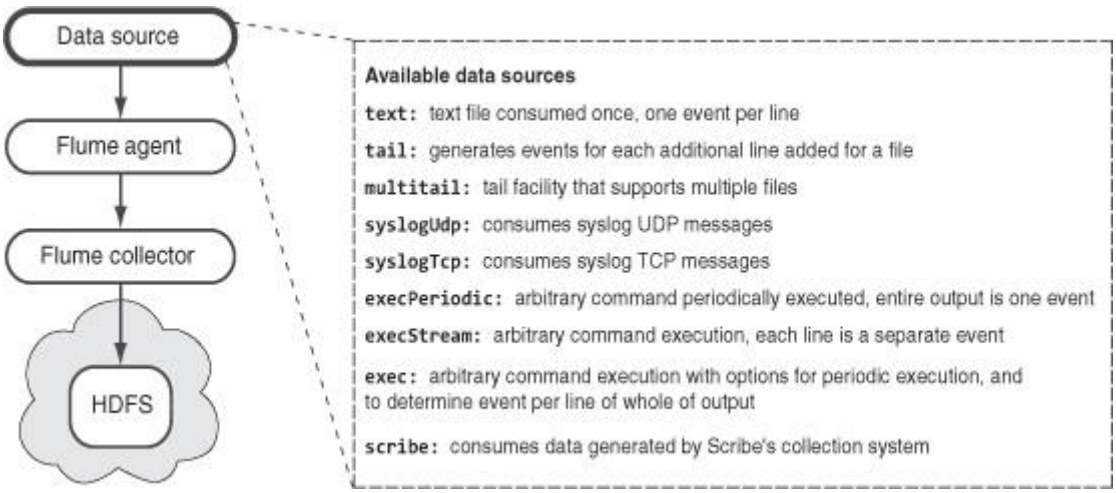


Data flow from /var/log/messages into HDFS

Flume Data Sources

A data source is required for both the Agent and Collector Nodes; it determines where they collect their data. The Agent Node's data source is your application or system data that you want to transfer to HDFS, and the Collector Node's data source is the Agent's data sink.

Figure Flume Agent Node data sources supported by the Agent Node



An automated mechanism to copy files into HDFS

Existing file transportation mechanisms such as Flume, Scribe, and Chukwa are geared towards supporting log files. What if you have different file formats for your files, such as semistructured or binary?

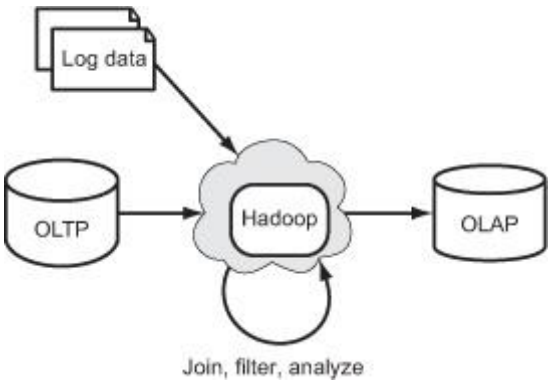
If the files were siloed in a way that the Hadoop slave nodes couldn't directly access, then you couldn't use Oozie to help with file ingress either.

Pulling data from databases

Most organizations' crucial data exists across a number of OLTP databases.

The data stored in these databases contains information about users, products, and a host of other useful items. If you want to analyze this data the traditional mechanism for doing so would be to periodically copy that data into a OLAP data warehouse.

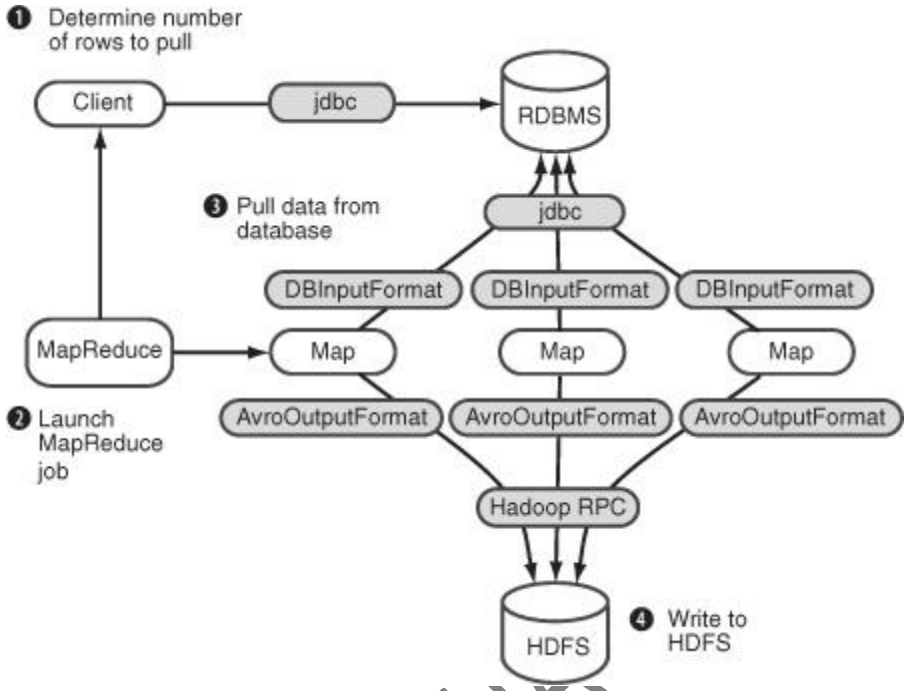
Using Hadoop for data ingress, joining, and egress to OLAP



Facebook is an example of an organization that has successfully utilized Hadoop and Hive as an OLAP platform to work with petabytes of data.

shows an architecture similar to that of Facebook's. This architecture also includes a feedback loop into the OLTP system, which can be used to push discoveries made in Hadoop, such as recommendations for users.

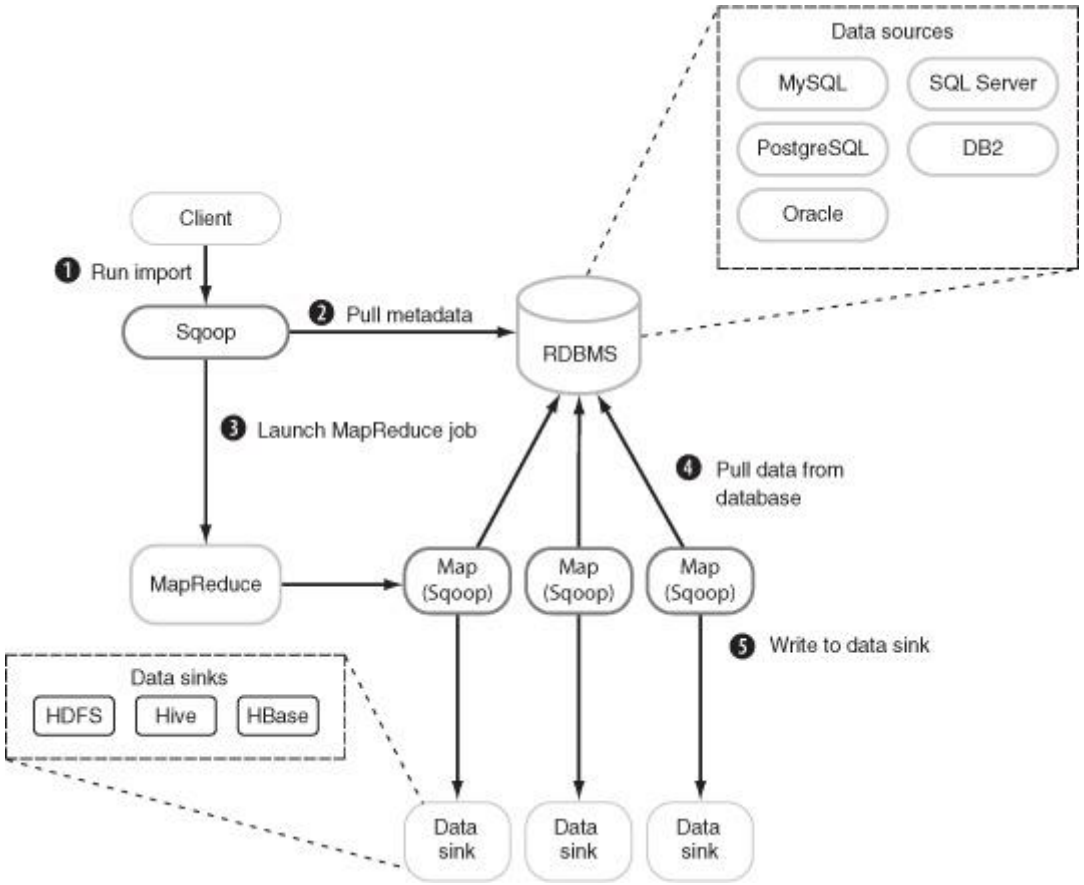
Example of MapReduce in four stages where `DBInputFormat` is used to pull data from a database



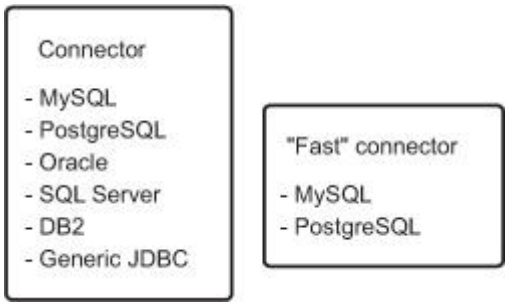
Using Sqoop to import data from MySQL

To use Sqoop as a simple mechanism to bring relational data into Hadoop clusters. We'll walk through the process of importing data from MySQL into Sqoop.

Figure Five-stage Sqoop import overview: connecting to the data source and using MapReduce to write to a data sink



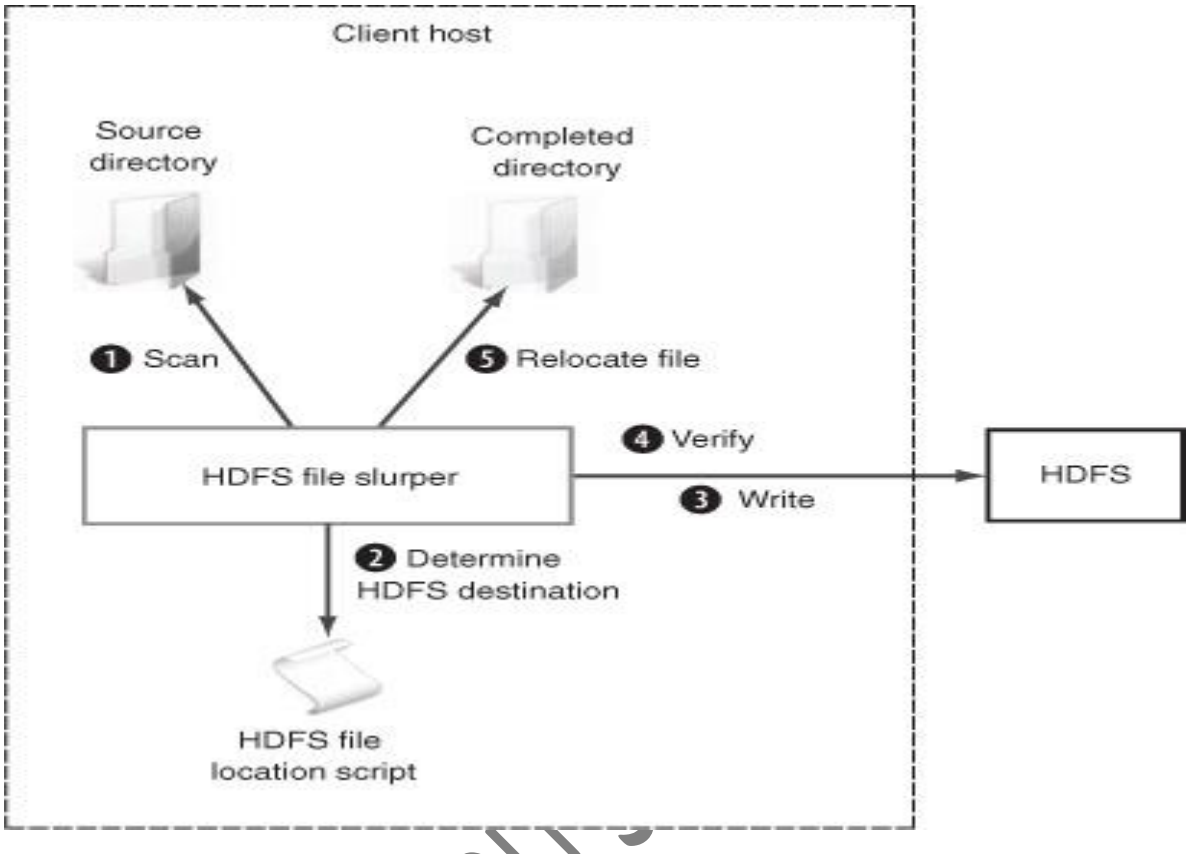
Sqoop connectors used to read and write to external systems



An automated mechanism to copy files into HDFS

Existing file transportation mechanisms such as Flume, Scribe, and Chukwa are geared towards supporting log files. What if you have different file formats for your files, such as semistructured or binary? If the files were siloed in a way that the Hadoop slave

nodes couldn't directly access, then you couldn't use Oozie to help with file ingress either.



- How do you effectively partition your writes to HDFS so that you don't lump everything into a single directory?
- How do you determine that your data is ready in HDFS for processing in order to avoid reading files that are mid-copy?
- How do you automate regular execution of your utility?

- The first step is to download and build the code. The following assumes that you have git, Java, and version 3.0 or newer of Maven installed locally:

```
$ git clone git://github.com/alexholmes/hdfs-file-slurper.git
$ cd hdfs-file-slurper/
$ mvn package
```

- Next you'll need to untar the tarball that the build created under /usr/local:

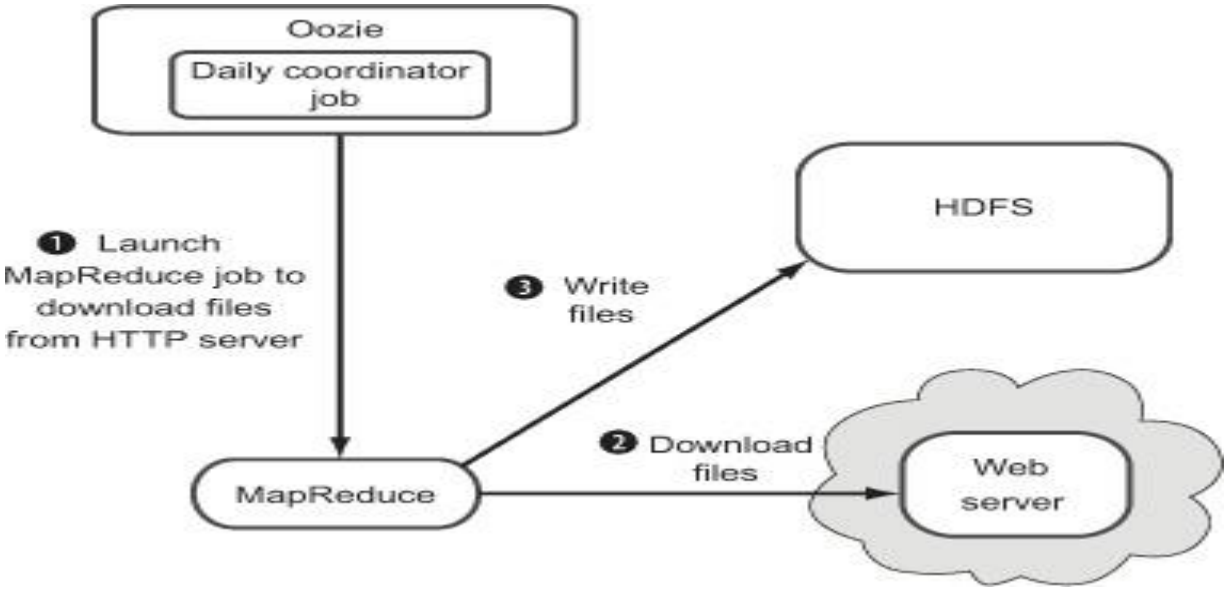
```
$ sudo tar -xzf target/hdfs-slurper-<version>-package.tar.gz \
  -C /usr/local/

$ sudo ln -s /usr/local/hdfs-slurper-<version>
/usr/local/hdfs-slurper
```

Scheduling regular ingress activities with Oozie

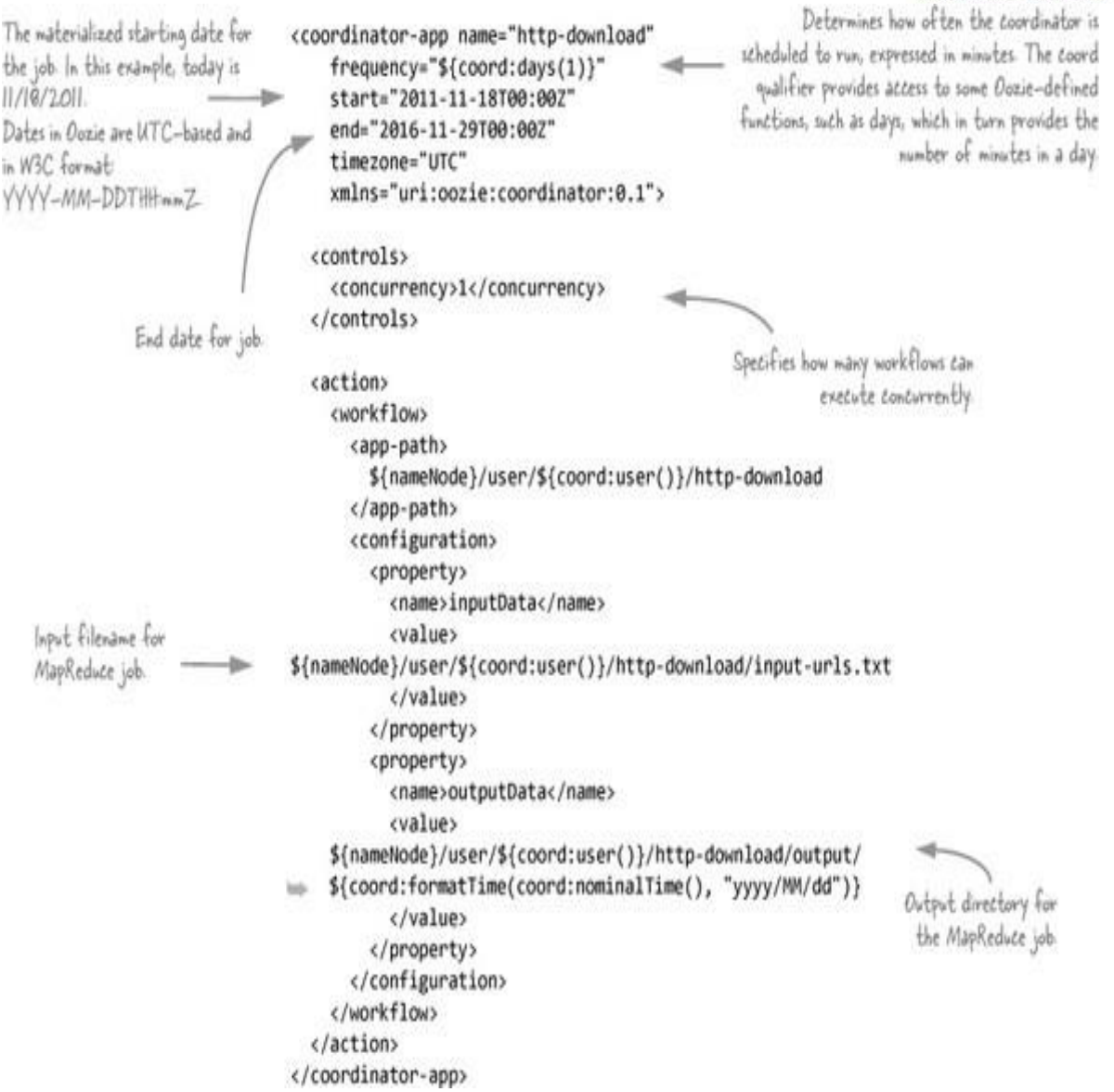
If your data is sitting on a filesystem, web server, or any other system accessible from your Hadoop cluster, you'll need a way to periodically pull that data into Hadoop. While tools exist to help with pushing log files and pulling from databases (which we'll cover in this chapter), if you need to interface with some other system, it's likely you'll need to handle the data ingress process yourself.

There are two parts to this data ingress process: the first is how you import data from another system into Hadoop, and the second is how you regularly schedule the data transfer.

Oozie can be used to ingress data into HDFS, and can also be used to execute post-ingress activities such as launching a MapReduce job to process the ingressed data. An Apache project, Oozie started life inside Yahoo. It's a Hadoop workflow engine that manages data processing activities. For our scenario Oozie has a coordinator engine that can start workflows based on data and time triggers.

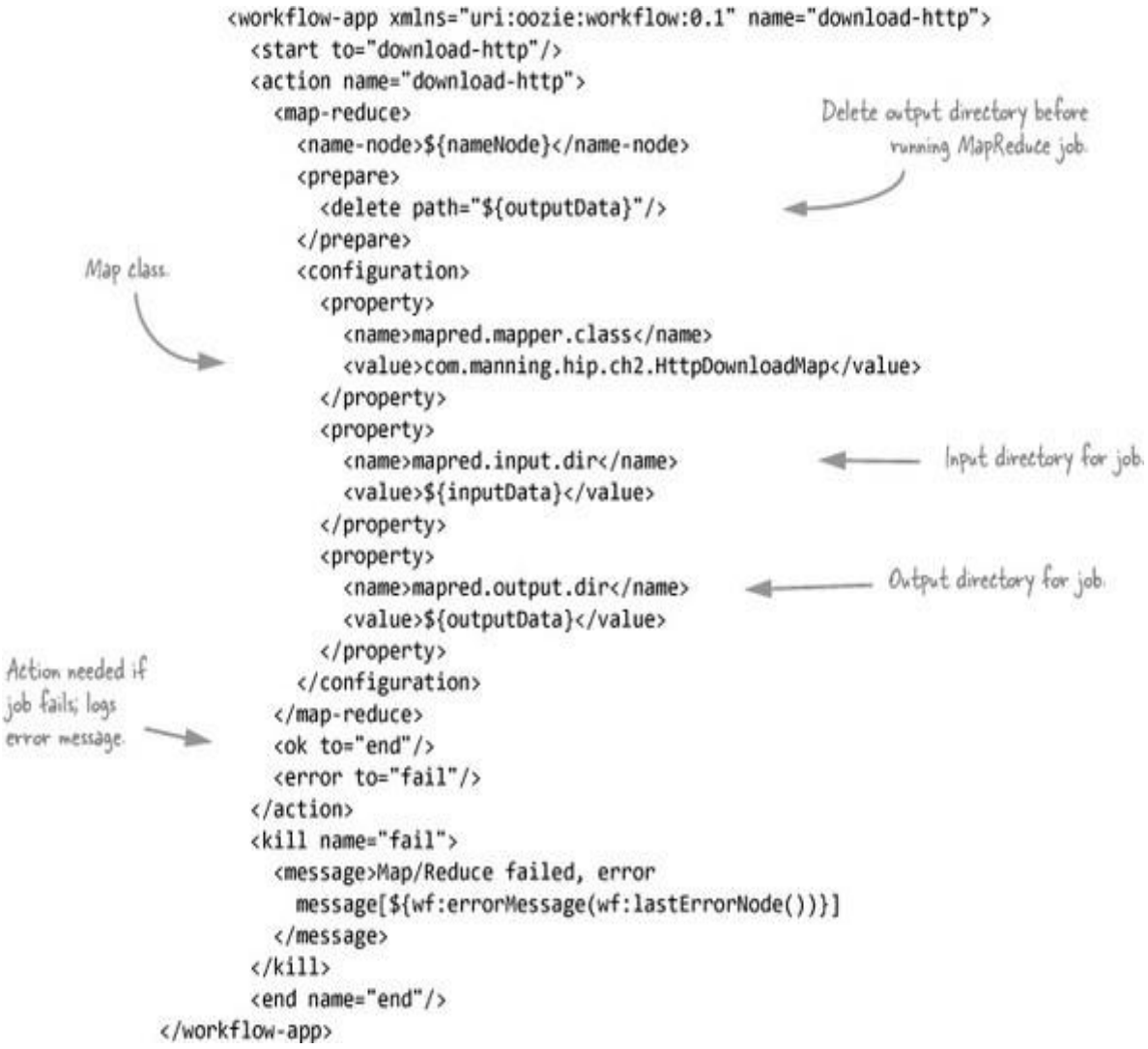


You'll use Oozie's data triggering capabilities to kick off a MapReduce job every 24 hours. Oozie has the notion of a coordinator job, which can launch a workflow at fixed intervals. The first step is to look at the coordinator XML configuration file. This file is used by Oozie's Coordination Engine to determine when it should kick off a workflow. Oozie uses the JSP expression language to perform parameterization, as you'll see in the following code. Create a file called coordinator.xml with the content shown in the next listing.




```
<coordinator-app name="http-download"
    frequency="${coord:days(1)}"
    start="2011-11-18T00:00Z"
    end="2016-11-29T00:00Z"
    timezone="UTC"
    xmlns="uri:oozie:coordinator:0.1">

  <controls>
    <concurrency>1</concurrency>
  </controls>

  <action>
    <workflow>
      <app-path>
        ${nameNode}/user/${coord:user()}/http-download
      </app-path>
      <configuration>
        <property>
          <name>inputData</name>
          <value>
${nameNode}/user/${coord:user()}/http-download/input-urls.txt
          </value>
        </property>
        <property>
          <name>outputData</name>
          <value>
    ${nameNode}/user/${coord:user()}/http-download/output/
    ${coord:formatTime(coord:nominalTime(), "yyyy/MM/dd")}
          </value>
        </property>
      </configuration>
    </workflow>
  </action>
</coordinator-app>
```

Defining the past workflow using Oozie's coordinator




```
<workflow-app xmlns="uri:oozie:workflow:0.1" name="download-http">
  <start to="download-http"/>
  <action name="download-http">
    <map-reduce>
      <name-node>${nameNode}</name-node>
      <prepare>
        <delete path="${outputData}"/>
      </prepare>
      <configuration>
        <property>
          <name>mapred.mapper.class</name>
          <value>com.manning.hip.ch2.HttpDownloadMap</value>
        </property>
        <property>
          <name>mapred.input.dir</name>
          <value>${inputData}</value>
        </property>
        <property>
          <name>mapred.output.dir</name>
          <value>${outputData}</value>
        </property>
      </configuration>
    </map-reduce>
    <ok to="end"/>
    <error to="fail"/>
  </action>
  <kill name="fail">
    <message>Map/Reduce failed, error
      message[${wf:errorMessage(wf:lastErrorNode())}]
    </message>
  </kill>
  <end name="end"/>
</workflow-app>
```

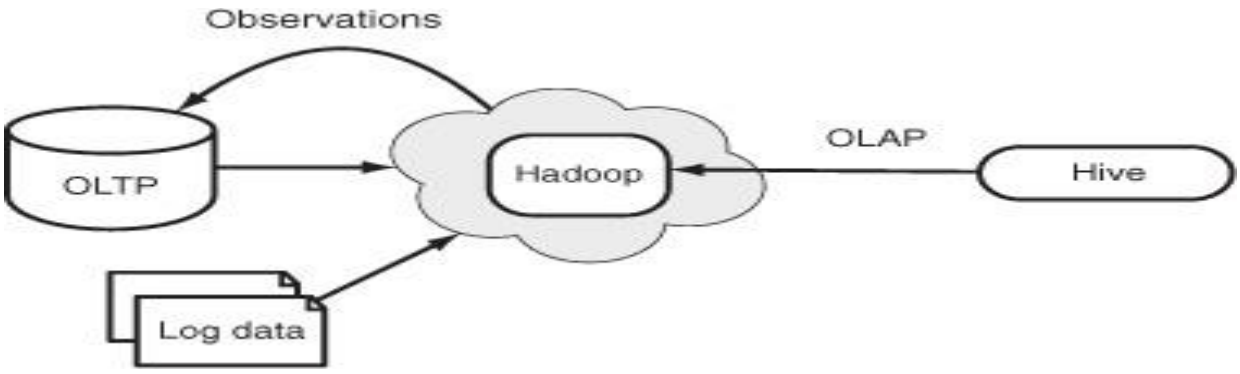
Pulling data from databases

Most organizations' crucial data exists across a number of OLTP databases. The data stored in these databases contains information about users, products, and a host of other useful items. If you want to analyze this data the traditional mechanism for doing so would be to periodically copy that data into a OLAP data warehouse.

Using Hadoop for data ingress, joining, and egress to OLAP

Facebook is an example of an organization that has successfully utilized Hadoop and Hive as an OLAP platform to work with petabytes of data.

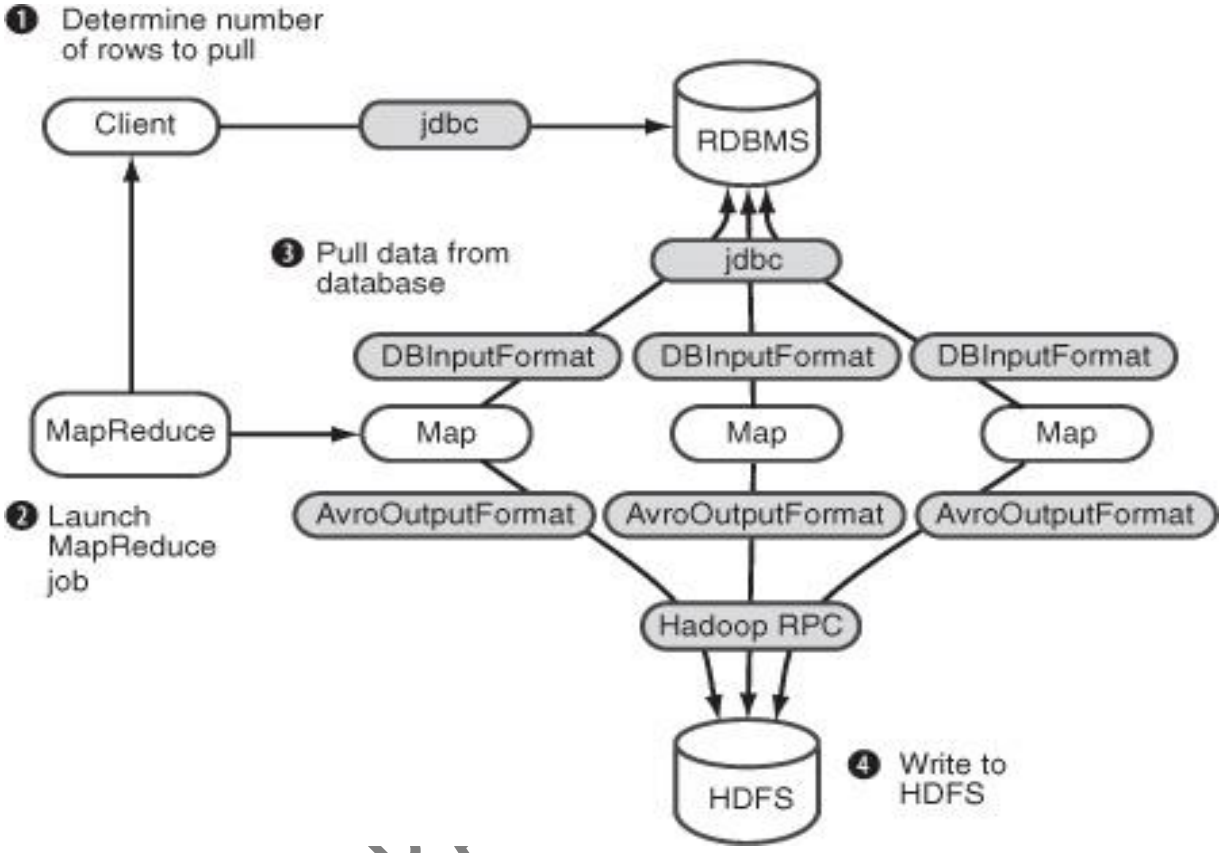
Using Hadoop for OLAP and feedback to OLTP systems



In either of the usage models shown in the previous figures, you need a way to bring relational data into Hadoop, and to also export it into relational databases. In the next techniques we'll cover two mechanisms you can use for database ingress. The first uses some built-in MapReduce classes, and the second provides an easy-to-use tool that removes the need for you to write your own code.

Database ingress with MapReduce

Example of MapReduce in four stages where `DBInputFormat` is used to pull data from a database



The details on the `Writable` interface and an implementation of the `Writable` called `StockPriceWritable`, which represents the stock data. The `DBInputFormat` class requires a bean representation of the table being imported, which implements both the `Writable` and `DBWritable` interfaces. Because you'll also need to implement the `DBWritable` interface, you'll do so by extending the `StockPriceWritable` class, as shown in the following code.

HBase

Our final foray into the area of moving data into Hadoop is a look at HBase. HBase is a real-time, column-oriented database, and is often either co-located on the same hardware that serves as your Hadoop cluster, or is in close proximity to a Hadoop cluster. Being able to work with HBase data directly in MapReduce, or push it into HDFS, is one of the huge advantages when picking HBase as a solution.

Two techniques in this section, the first focusing on how to import HBase data into HDFS, and the second on how to use HBase as a data source for a MapReduce job.

HBase ingress into HDFS

What if you had some customer data sitting in HBase that you wanted to leverage in MapReduce in conjunction with data in HDFS? You could write a MapReduce job which takes as input the HDFS dataset and pulls data directly from HBase in your map or reduce code. But in some cases it may be more useful to take a dump of the data in HBase into HDFS directly, especially if you plan to utilize that data in multiple MapReduce jobs and the HBase data is immutable, or changes infrequently.

How do you get HBase data into HDFS?

HBase includes an `Export` class that can be used to import HBase data into HDFS in SequenceFile format. This technique also walks through code that can be used to read the imported HBase data.

```
$ hbase shell



hbase(main):012:0> list

stocks_example

1 row(s) in 0.0100 seconds



hbase(main):007:0> scan 'stocks_example'

ROW              COLUMN+CELL

AAPL2000-01-03   column=details:stockAvro,
timestamp=1322315975123,...

AAPL2001-01-02   column=details:stockAvro, timestamp=1322315975123,
```

Moving data out of Hadoop

After data has been brought into Hadoop it will likely be joined with other datasets to produce some results. At this point either that result data will stay in HDFS for future access, or it will be pushed out of Hadoop. An example of this scenario would be one where you pulled some data from an OLTP database, performed some machine learning activities on that data, and then copied the results back into the OLTP database for use by your production systems.

In this section we'll cover how to automate moving regular files from HDFS to a local filesystem. We'll also look at data egress to relational databases and HBase. To start off we'll look at how to copy data out of Hadoop using the HDFS Slurper.

Egress to a local filesystem

we looked at two mechanisms to move semistructured and binary data into HDFS, the HDFS File Slurper open source project, and Oozie to trigger a data ingress workflow. The challenge to using a local filesystem for egress (and ingress for that matter) is that map and reduce tasks running on clusters won't have access to the filesystem on a specific server. You need to leverage one of the following three broad options for moving data from HDFS to a filesystem:

1. Host a proxy tier on a server, such as a web server, which you would then write to using MapReduce.
2. Write to the local filesystem in MapReduce and then as a postprocessing step trigger a script on the remote server to move that data.
3. Run a process on the remote server to pull data from HDFS directly.

The third option is the preferred approach because it's the simplest and most efficient, and as such is the focus of this section. We'll look at how you can use the HDFS Slurper to automatically move files from HDFS out to a local filesystem.

Databases

Databases are usually the target of Hadoop data egress in one of two circumstances: either when you move data back into production databases to

be used by production systems, or when you move data into OLAP databases to perform business intelligence and analytics functions.

we'll use Apache Sqoop to export data from Hadoop to a MySQL database. Sqoop is a tool that simplifies database imports and exports. Sqoop is covered in detail in technique 5.

We'll walk through the process of exporting data from HDFS to Sqoop. We'll also cover methods using the regular connector, as well as how to do bulk imports using the fast connector.

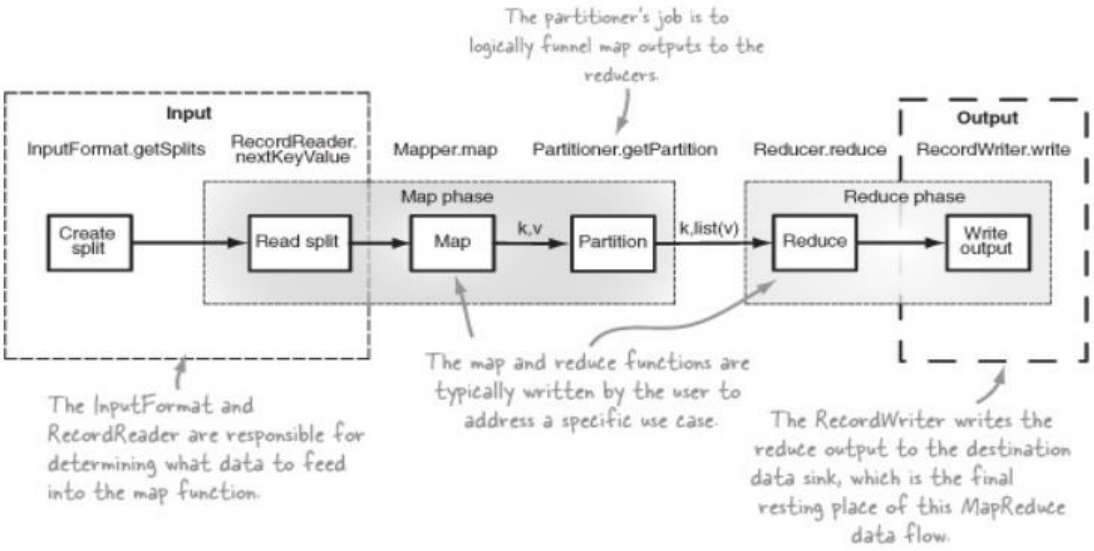
Using Sqoop to export data to MySQL

Hadoop excels at performing operations at scales which defeat most relational databases, so it's common to extract OLTP data into HDFS, perform some analysis, and then export it back out to a database.

3.Understanding inputs and outputs of Map reduce

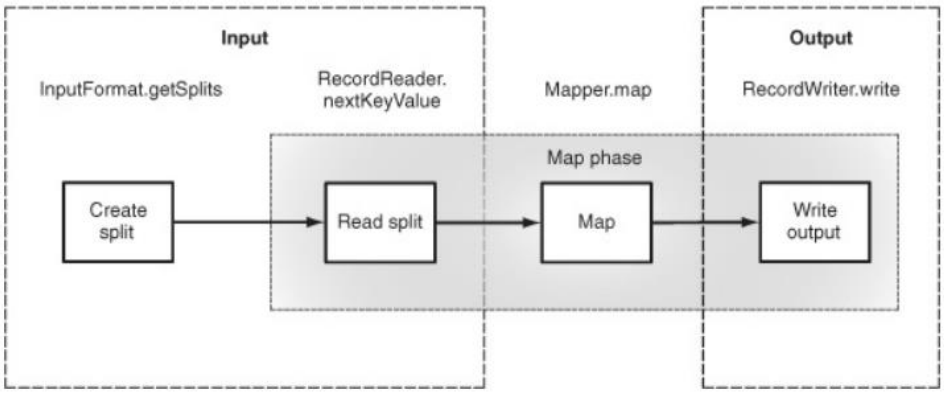
Your data might be XML files sitting behind a number of FTP servers, text log files sitting on a central web server, or Lucene indexes[1] in HDFS. How does MapReduce support reading and writing to these different serialization structures across the various storage mechanisms? You'll need to know the answer in order to support a specific serialization format.

Figure 3.1. High-level input and output actors in MapReduce



This diagram shows the high-level data flows through MapReduce and identifies the actors responsible for various parts of the flow. On the input side you see that some work (*Create split*) is performed outside of the map phase, and other work is performed as part of the map phase (*Read split*). All of the output work is performed in the reduce phase (*Write output*).

Figure 3.2. Input and output actors in MapReduce with no reducers



The above diagram shows the same flow with a map-only job. In a map-only job the MapReduce framework still uses the `OutputFormat` and `RecordWriter` classes to write the outputs directly to the data sink.

Let's walk through the data flow and describe the responsibilities of the various actors. As we do this, we'll also look at the relevant code from the built-in `TextInput-Format` and `TextOutputFormat` classes to better understand the concepts. The `Text-InputFormat` and `TextOutputFormat` classes read and write line-oriented text files.

Data input

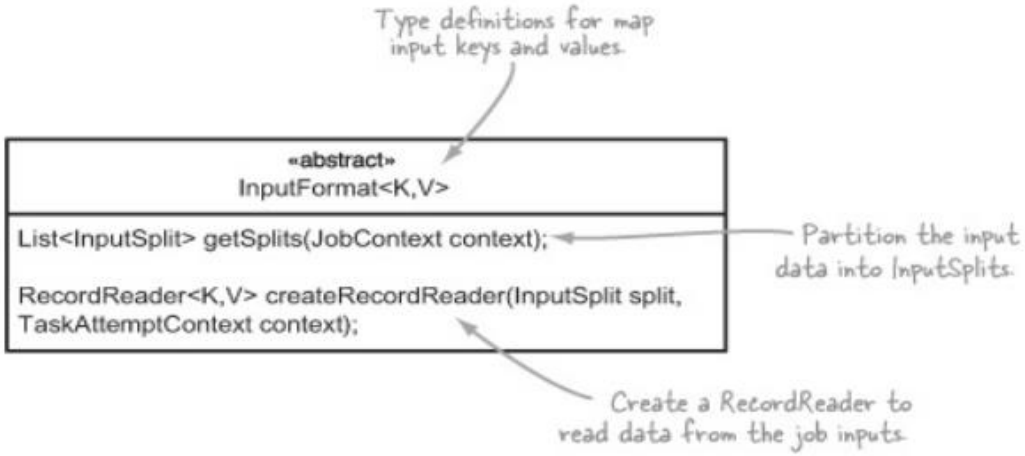
The two classes that support data input in MapReduce are `InputFormat` and `Record-Reader`. The `InputFormat` class is consulted to determine how the input data should be partitioned for the map tasks, and the `RecordReader` performs the reading of data from the inputs.

Inputformat

Every job in MapReduce must define its inputs according to contracts specified in the `InputFormat` abstract class. `InputFormat` implementers must fulfill three contracts: first, they describe type information for map input keys and values;

next, they specify how the input data should be partitioned; and finally, they indicate the `RecordReader` instance that should read the data from source.

The below diagram shows the `InputFormat` class and how these three contracts are defined.



Arguably the most crucial contract is that of determining how to divide the input data. In MapReduce nomenclature these divisions are referred to as *input splits*. The input splits directly impact the map parallelism because each split is processed by a single map task.

Working with an `InputFormat` that is unable to create multiple input splits over a single data source (such as a file) will result in a slow map phase because the file will be processed sequentially.

The `TextInputFormat` class provides an implementation of the `InputFormat` class's `createRecordReader` method but delegates the calculation of input splits to its parent class, `FileInputFormat`.

The following code shows the relevant parts of the `TextInputFormat` class:

```
public class TextInputFormat
            extends FileInputFormat<LongWritable, Text> {

  @Override
  public RecordReader<LongWritable, Text>
    createRecordReader(InputSplit split,
                       TaskAttemptContext context) {
    String delimiter = context.getConfiguration().get(
        "textinputformat.record.delimiter");
    byte[] recordDelimiterBytes = null;
    if (null != delimiter)
      recordDelimiterBytes = delimiter.getBytes();
    return new LineRecordReader(recordDelimiterBytes);
  }
...
```

The parent class, FileInputFormat, provides all of the input split functionality.

The default record delimiter is newline, but it can be overridden with textinputformat.record.delimiter.

Construct the RecordReader to read the data from the data source.

The code in `FileInputFormat` to determine the input splits is a little more complicated. A simplified form of the code is shown in the following example to portray the main elements of this method:

```
public List<InputSplit> getSplits(JobContext job
                                  ) throws IOException {
  List<InputSplit> splits = new ArrayList<InputSplit>();
  List<FileStatus>files = listStatus(job);
  for (FileStatus file: files) {
    Path path = file.getPath();

    BlockLocation[] blkLocations =
      FileSystem.getFileBlockLocations(file, 0, length);

    long splitSize = file.getBlockSize();

    while (splitsRemaining()) {
      splits.add(new FileSplit(path, ...));
    }
  }
  return splits;
}
```

The listStatus method determines all the input files for the job.

Retrieve all of the file blocks.

The size of the splits is the same as the block size for the file. Each file can have a different block size.

Create a split for each file block and add it to the result.

The following code is an example of how you specify the `InputFormat` to use for a MapReduce job:

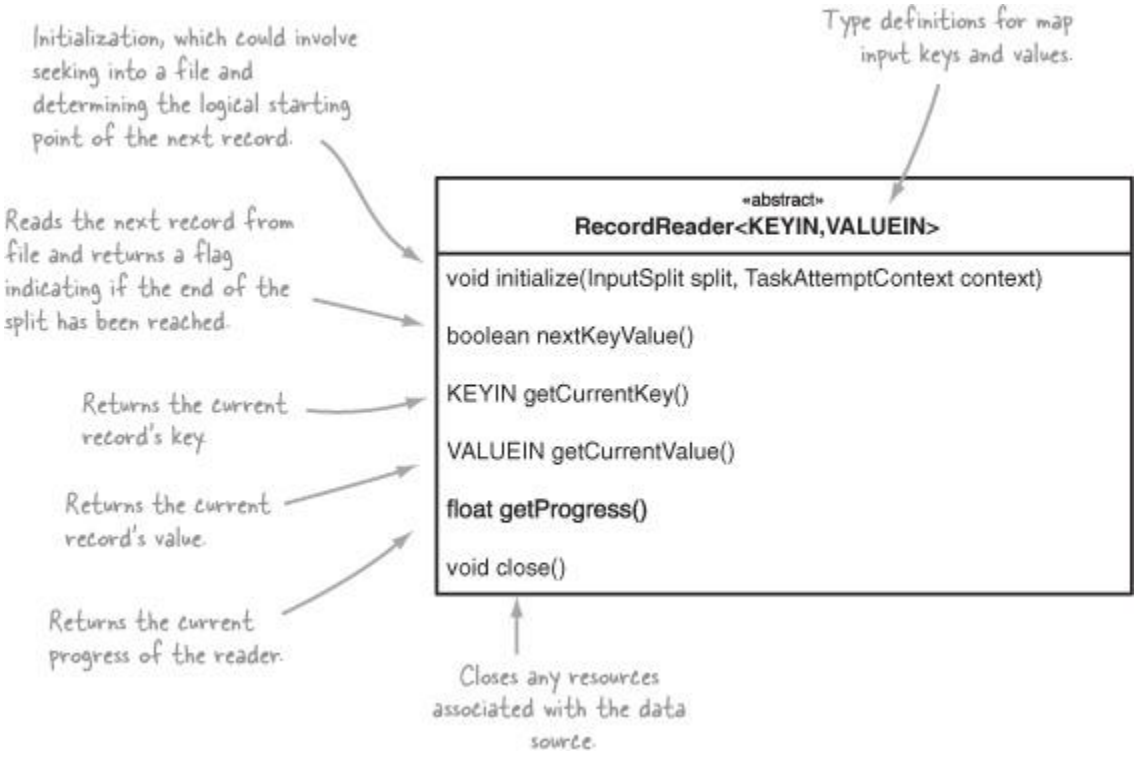
```
job.setInputFormatClass(TextInputFormat.class);
```

Recordreader

The `RecordReader` class is used by MapReduce in the map tasks to read data from an input split and provide each record in the form of a key/value pair for use by mappers.

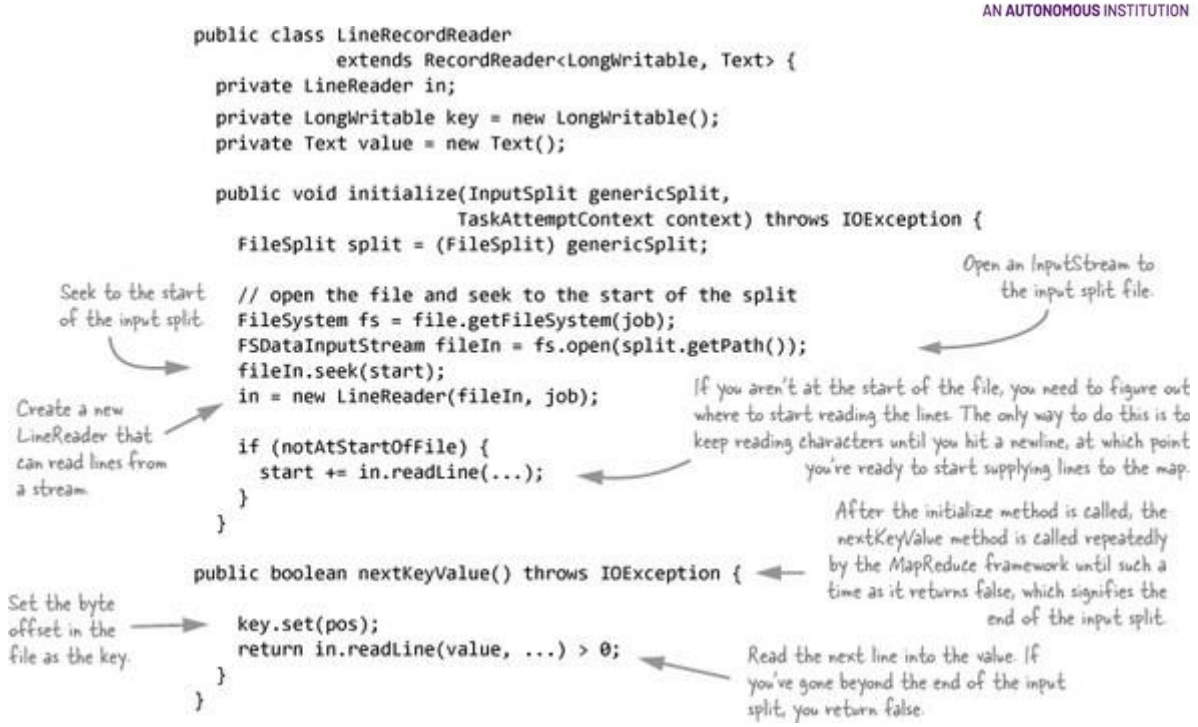
A task is commonly created for each input split, and each task has a single `RecordReader` that's responsible for reading the data for that input split. Figure 3.4 shows the abstract methods you must implement.

Figure 3.4. The annotated RecordReader class and its abstract methods



As shown in a previous section, the `TextInputFormat` class created a `LineRecordReader` to read records from the input splits.
The `LineRecordReader` directly extends the `RecordReader` class and leverages the `LineReader` class to read lines from the input split.
The `LineRecordReader` uses the byte offset in the file for the map key and the contents of the line for the map value.

I have included a simplified version of the `LineRecordReader` in the following example



Because the `LineReader` class is easy, we'll skip that code. The next step will be a look at how MapReduce supports data outputs.

Data output

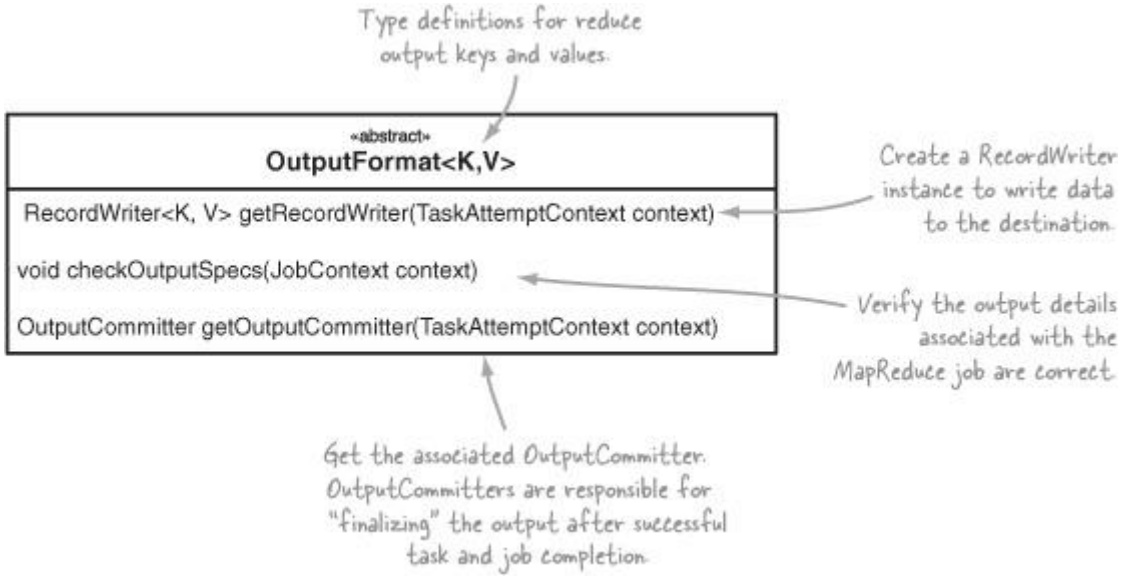
MapReduce uses a similar process for supporting output data as it does for input data.

Two classes must exist, an `OutputFormat` and a `RecordWriter`. The `OutputFormat` performs some basic validation of the data sink properties, and the `RecordWriter` writes each reducer output to the data sink.

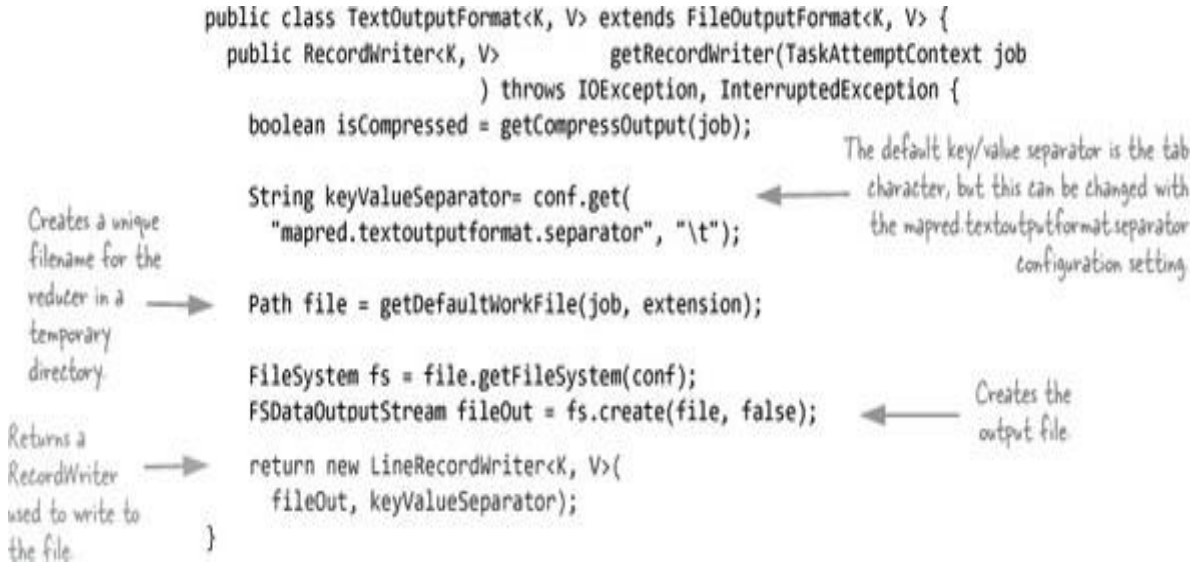
Outputformat

Much like the `InputFormat` class, the `OutputFormat` class, defines the contracts that implementers must fulfill, including checking the information related to the job output, providing a `RecordWriter`, and specifying an output committer, which allows writes to be staged and then made "permanent" upon task and/or job success.

Figure 3.5. The annotated OutputFormat class



Just like the `TextInputFormat`, the `TextOutputFormat` also extends a base class, `File-OutputFormat`, which takes care of some complicated logistics such as output committing, For now let's take a look at the work the `TextOutputFormat` is performing.



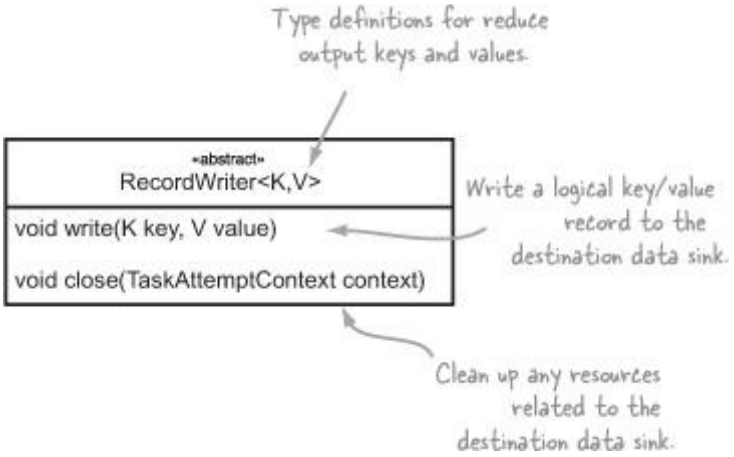
The following code is an example of how you specify the `OutputFormat` that should be used for a MapReduce job:

```
job.setOutputFormatClass(TextOutputFormat.class);
```

Recordwriter

You'll use the `RecordWriter` to write the reducer outputs to the destination data sink. It's a simple class, as figure 3.6 illustrates.

Figure The annotated RecordWriter class overview



The `TextOutputFormat` returned a `LineRecordWriter` object (`LineRecordWriter` is an inner class of `TextOutputFormat`) to perform the writing to file. A simplified version of that class (source at http://goo.gl/8ab7Z) is shown in the following example:

```
protected static class LineRecordWriter<K, V>  extends RecordWriter<K, V> {

  protected DataOutputStream out;

  public synchronized void write(K key, V value)
    throws IOException {

    writeObject(key);
    out.write(keyValueSeparator);
    writeObject(value);
    out.write(newline);
  }

  private void writeObject(Object o) throws IOException {
    out.write(o);
  }
```

Write out the key, separator, value, and newline.

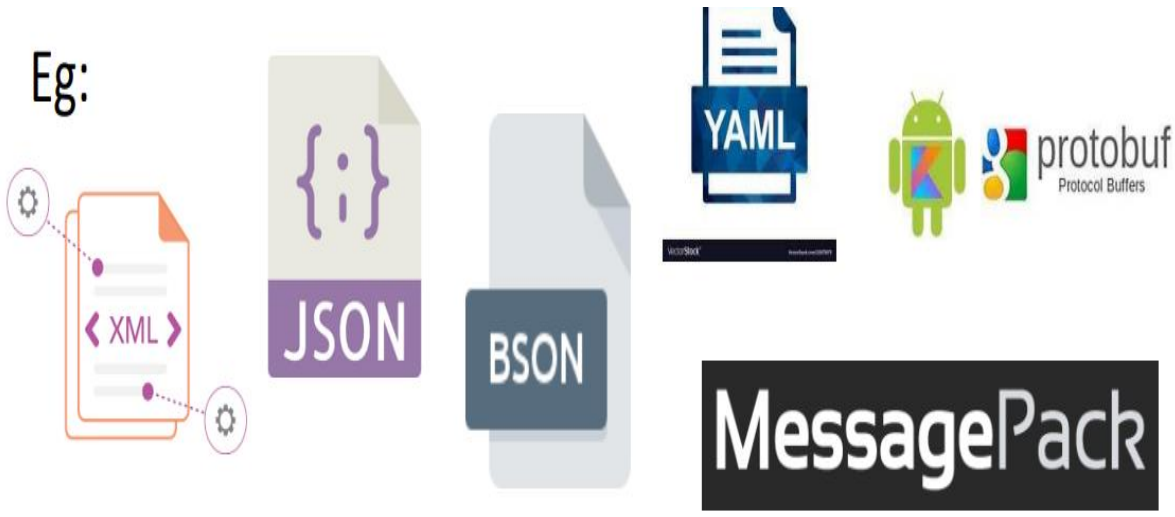
Write out the Object to the output stream.

While on the map side it's the `InputFormat` that determines how many map tasks are executed, on the reducer side the number of tasks is solely based on the value for `mapred.reduce.tasks` set by the client (or if it isn't set, the value is picked up from mapred-site.xml, or mapred-default.xml if it doesn't exist in the site file).

Now that you know what's involved in working with input and output data in MapReduce, it's time to apply that knowledge to solving some common data serialization problems. Your first step in this data serialization journey is to learn how to work with file formats such as XML.

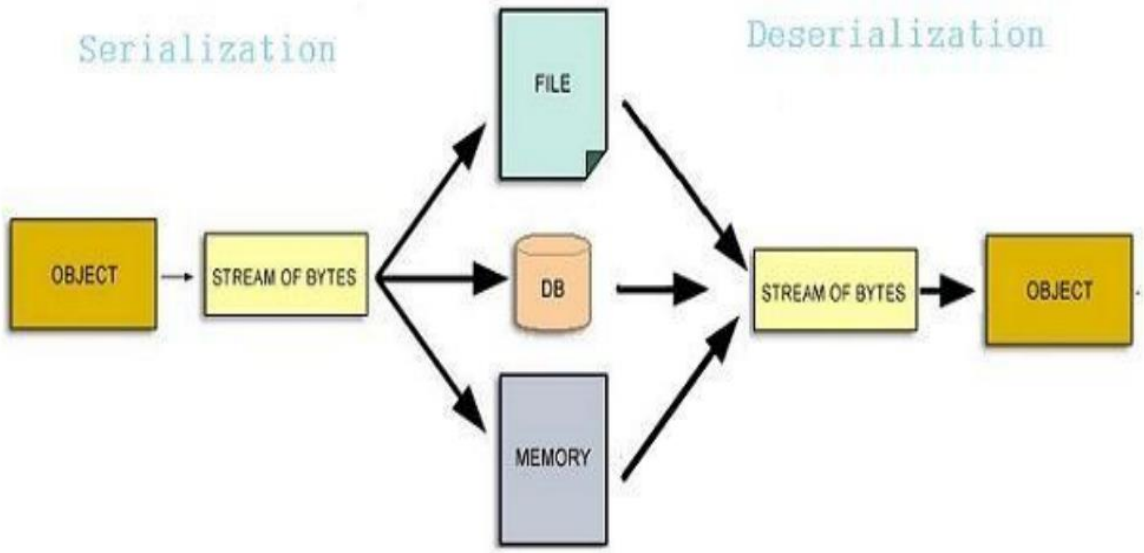
Data Serialization

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. Once the serialized data is transmitted the reverse process of creating objects from the byte sequence called deserialization.



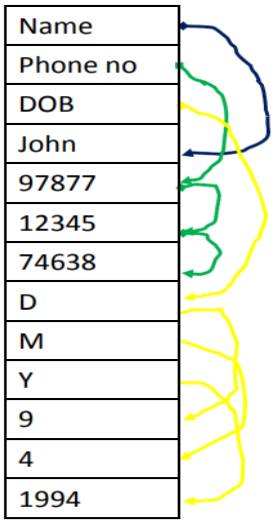
Computer data is generally organized in data structures such as arrays, tables, trees, classes. When data structures need to be stored or transmitted to another location, such as across a network, they are serialized.

Serialization becomes complex for nested data structures and object references.



COMPLICATED STRUCTURE

Name: John
Phone no: 97877
12345
74638
DOB: D: 9
M: 4
Y: 1994



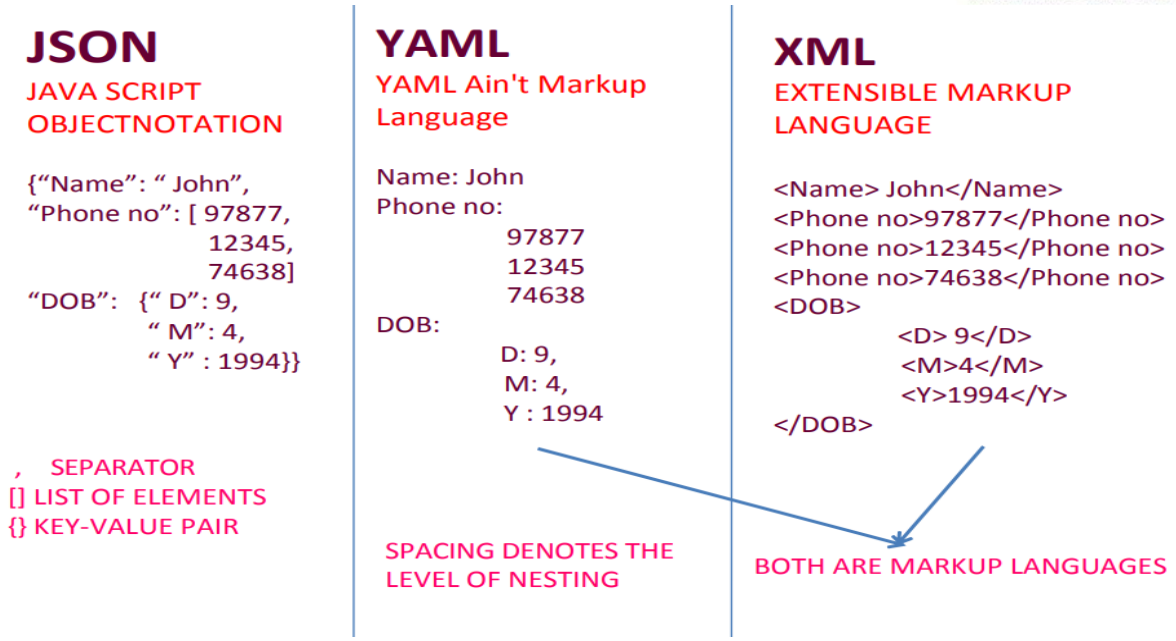
SERIALIZED FORMAT

{“Name”: “ John”,
“Phone no”: [97877,
12345,
74638]
“DOB”: {“ D”: 9,
“ M”: 4,
“ Y” : 1994}}

JSON

IN ONE LINE

{“Name”: “John”,“Phone no”: [97877,12345,74638”],“DOB”: {“ D”: 9,“ M”: 4,“ Y” : 1994}}



APPLICATIONS OF DATA SERIALIZATION

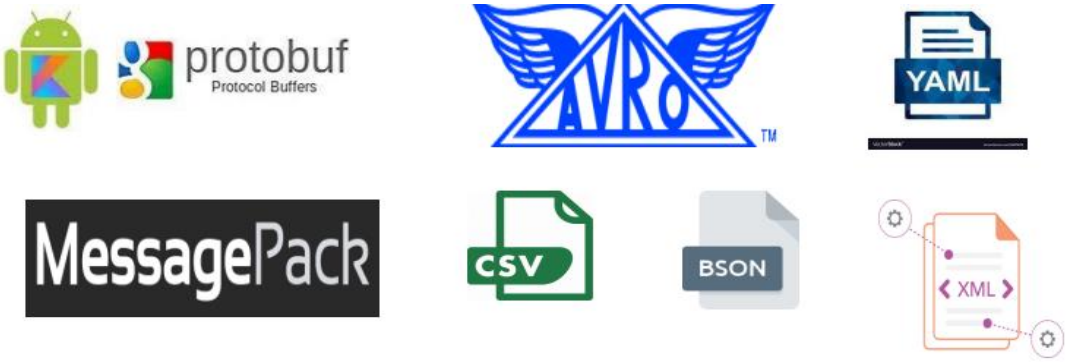
- Serialization allows a program to save the state of an object and recreate it when needed.
- Persisting data onto files – happens mostly in language-neutral formats such as CSV or XML. However, most languages allow objects to be serialized directly into binary using APIs
- Storing data into Databases – 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 clientto server and vice versa.
- Sharing data in a Distributed Object Model – When programs written in different languages need to share object data over a distributed network .However, SOAP, REST and other web services have replaced these applications now.

POTENTIAL RISK DUE TO SERIALIZATION

- It may allow a malicious party with access to the serialization byte stream to read private data, create objects with illegal or dangerous state, or obtain references to the private

fields of deserialized objects. Workarounds are tedious, not guaranteed.

- Open formats too have their security issues.
- XML might be tampered using external entities like macrosor unverified schema files.
- JSON data is vulnerable to attack when directly passed to a JavaScript engine due to features like JSONP requests.



DATA SERIALIZATION FORMATS

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

Serialization Formats in Hadoop

- XML
- CSV
- YAML
- JSON
- BSON
- MessagePack
- Thrift
- Protocol buffers
- Avro

TEXT-BASED DATA SERIALIZATION FORMATS

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.
- CSV file is the most commonly used data file format.
- Easy to read, Easy to parse, Easy to export data from an RDBMS table.

It has three major drawbacks when used for HDFS.

1. All lines in a CSV file is a record, therefore, we should not include any headers or footers. In other word, CSV file cannot be stored in HDFS with any meta data.
2. CSV file has very limited support for schema evolution. Because the fields for each record are ordered, we are not able to change the orders. We can only append new fields to the end of each line.
3. It does not support block compression which many other file formats support. The whole file has to be compressed and decompressed for reading, adding a significant read performance cost to the files.

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. No direct support for DATE data type. All data is dynamically processed
- It is in text format that stores meta data with the data, so it fully supports schema evolution and also spiltable.
- We can easily add or remove attributes for each datum. However, because it's text file, it doesn't support block compression.

YAML Ain't Markup Language :

- It is a data serialization language which is designed to be human -friendly and works well with other programming languages for everyday tasks.
- Superset of JSON
- Supports complex data types. Maps easily to native data structures.

Binary JSON:

- It is a binary-encoded serialization of JSON-like documents.
- MongoDB uses BSON ,when storing documents in collections
- It deals with attribute-value pairs like JSON. Includes datetime, bytearray and other data types not present in JSON.

•It is Created by Google
•It is Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data
•Protocol buffers currently support generated code in Java, Python, Objective-C, and C++.

- It is designed for data to be transparently converted from/to JSON.
- Support rich set of data structures
- It create schema based annotation
- Primary use is network communication

AVRO

- Apache **Avro** is a language-neutral data
- serialization system, developed by **Doug Cutting.** the father of Hadoop.
- It also called a schema-based serialization technique.

FEATURES

- Avro uses JSON format to declare the data structures. Presently, it supports languages such as Java, C, C++, C#, Python, and Ruby.
- Avro creates binary structured format that is both **compressible** and **splitable**. Hence it can be efficiently used as the input to Hadoop MapReduce jobs.
- Avro provides **rich data structures**.

- Avro **schemas** defined in **JSON**, facilitate implementation in the languages that already have JSON libraries.
- Avro creates a self-describing file named *Avro Data File,* in which it stores data along with its schema in the metadata section.
- Avro is also used in Remote Procedure Calls (RPCs).

Thrift and **Protocol Buffers** are the most competent libraries with Avro. Avro differs from these frameworks in the following ways

- Avro supports both dynamic and static types as per the requirement. Protocol Buffers and Thrift use Interface Definition Languages (IDLs) to specify schemas and their types. These IDLs are used to generate code for serialization and deserialization.
- Avro is built in the Hadoop ecosystem. Thrift and Protocol Buffers are not built in Hadoop ecosystem.

PERFORMANCE CHARACTERISTICS

- Speed – Binary formats are faster than textual formats. A late entrant, **protobuf reports the best times**. JSON is preferable due to readability and being schema-less.
- Data size – This refers to the physical space in bytes post serialization. For small data, compressed JSON data occupies more space compared to binary formats like protobuf. Generally, **binary formats always occupy less space**.
- Usability – Human readable formats like JSON are naturally preferred over binary formats. For editing data, YAML is good. Schema definition is easy in protobuf, with in-built tools.
- Compatibility-Extensibility – JSON is a closed format. XML is average with schema versioning. Backward compatibility (extending schemas) is best handled by protobuf.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

KG REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

B. Tech III Year II Semester **Academic Year: 2023-2024**

SUBJECT: BIG DATA ANALYTICTS **SUBJECT CODE: KG21CD605** **REGULATION: KGR21**

UNIT-4 NOTES

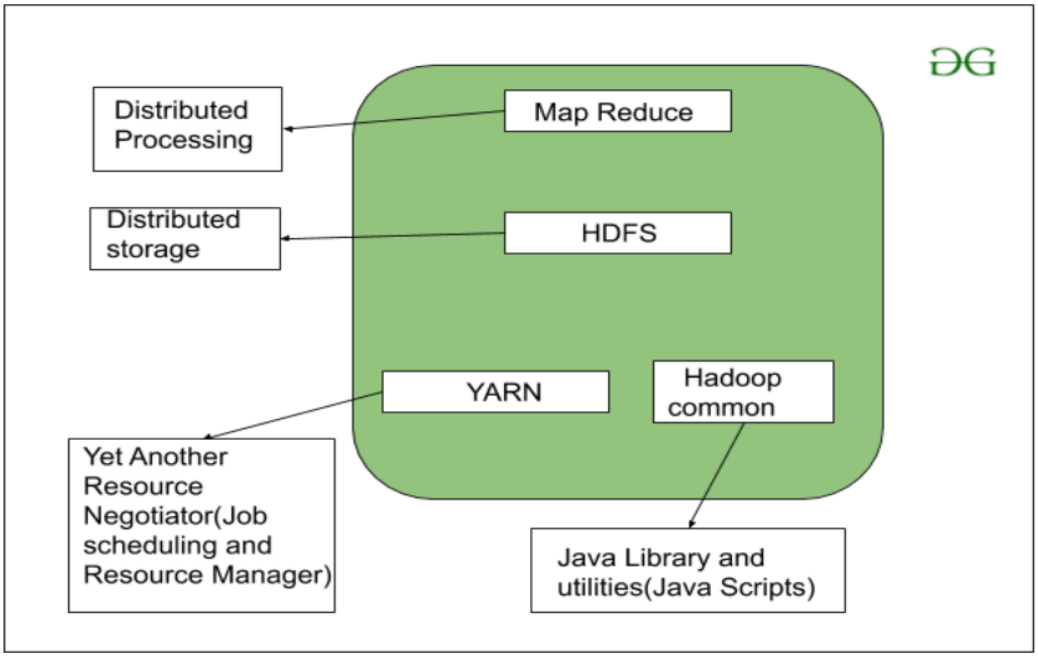
UNIT IV
Hadoop Architecture: Hadoop: RDBMS Vs Hadoop, Hadoop Overview, Hadoop distributors, HDFS, HDFS Daemons, Anatomy of File Write and Read., Name Node, Secondary Name Node, and Data Node, HDFS Architecture, Hadoop Configuration, Map Reduce Framework, Role of HBase in Big Data processing, HIVE, PIG.

Hadoop Architecture: Hadoop: RDBMS Vs Hadoop

Hadoop Architecture:

As we all know Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google. Today lots of Big Brand Companies are using Hadoop in their Organization to deal with big data, eg. Facebook, Yahoo, Netflix, eBay, etc. The Hadoop Architecture Mainly consists of 4 components.

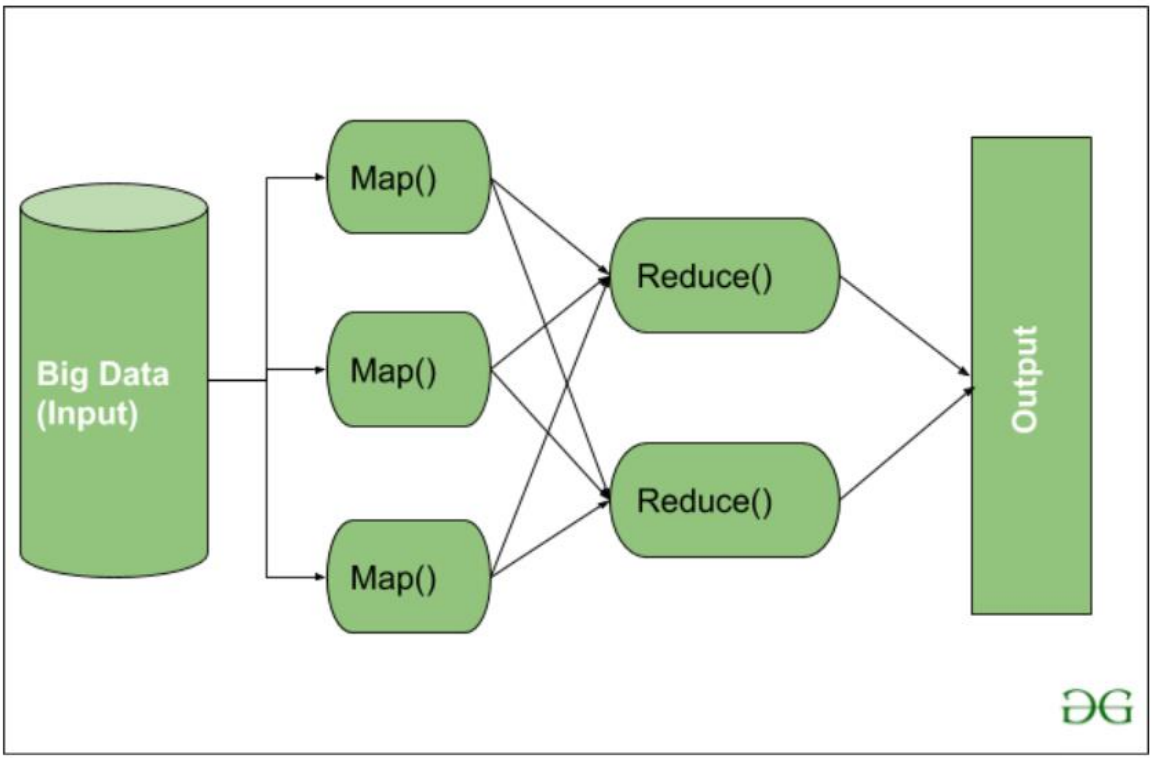
- MapReduce
- HDFS(Hadoop Distributed File System)
- YARN(Yet Another Resource Negotiator)
- Common Utilities or Hadoop Common



Let's understand the role of each one of this component in detail.

1. MapReduce

MapReduce nothing but just like an Algorithm or a data structure that is based on the YARN framework. The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use. MapReduce has mainly 2 tasks which are divided phase-wise:
In first phase, **Map** is utilized and in next phase **Reduce** is utilized.



Here, we can see that the *Input* is provided to the Map() function then it's *output* is used as an input to the Reduce function and after that, we receive our final output. Let's understand What this Map() and Reduce() does.

As we can see that an Input is provided to the Map(), now as we are using Big Data. The Input is a set of Data. The Map() function here breaks this DataBlocks into **Tuples** that are nothing but a key-value pair. These key-value pairs are now sent as input to the Reduce(). The Reduce() function then combines this broken Tuples or key-value pair based on its Key value and form set of Tuples, and perform some operation like sorting, summation type job, etc. which is then sent to the final Output Node. Finally, the Output is Obtained.

The data processing is always done in Reducer depending upon the business requirement of that industry. This is How First Map() and then Reduce is utilized one by one.

Let's understand the *Map Task* and *Reduce Task* in detail.

Map Task:

- **RecordReader** The purpose of *recordreader* is to break the records. It is responsible for providing key-value pairs in a Map()

function. The key is actually is its locational information and value is the data associated with it.

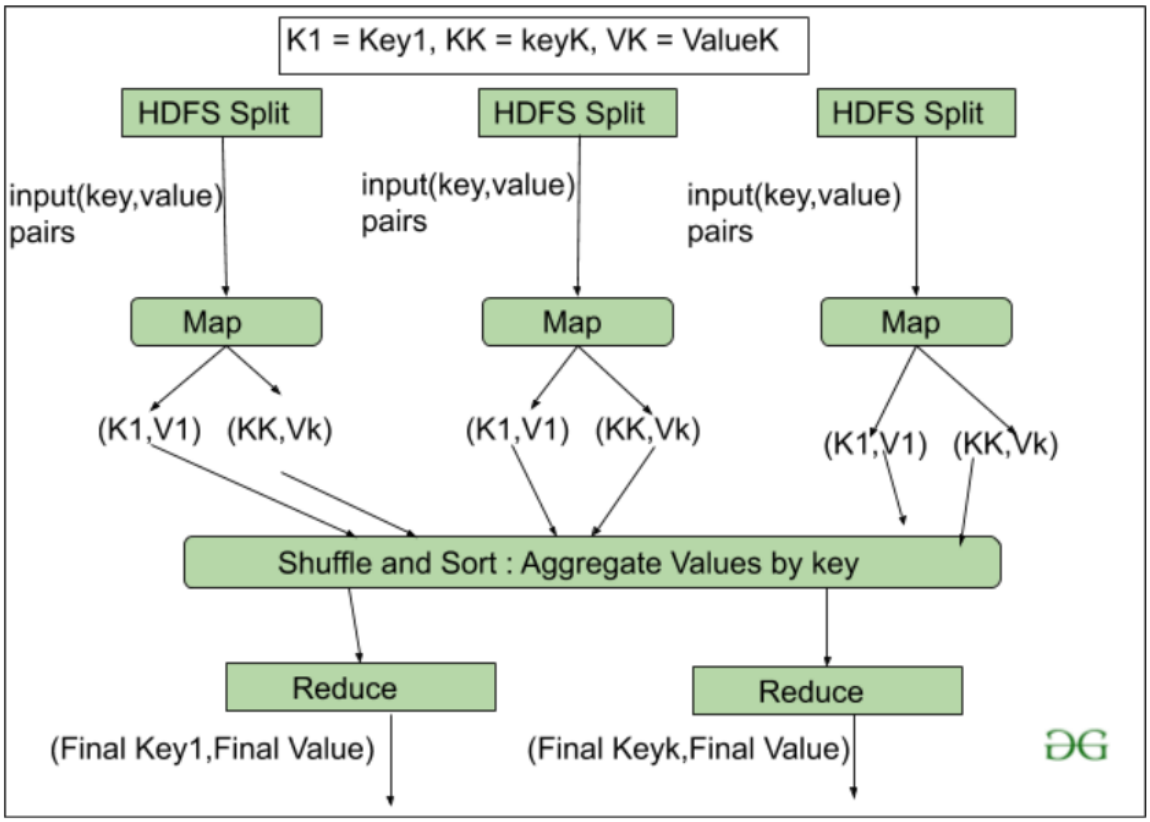
- **Map:** A map is nothing but a user-defined function whose work is to process the Tuples obtained from record reader. The Map() function either does not generate any key-value pair or generate multiple pairs of these tuples.
- **Combiner:** Combiner is used for grouping the data in the Map workflow. It is similar to a Local reducer. The intermediate key-value that are generated in the Map is combined with the help of this combiner. Using a combiner is not necessary as it is optional.
- **Partitionar:** Partitional is responsible for fetching key-value pairs generated in the Mapper Phases. The partitioner generates the shards corresponding to each reducer. Hashcode of each key is also fetched by this partition. Then partitioner performs it's(Hashcode) modulus with the number of reducers(*key.hashcode()%(number of reducers)*).

Reduce Task

- **Shuffle and Sort:** The Task of Reducer starts with this step, the process in which the Mapper generates the intermediate key-value and transfers them to the Reducer task is known as *Shuffling*. Using the Shuffling process the system can sort the data using its key value.

  Once some of the Mapping tasks are done Shuffling begins that is why it is a faster process and does not wait for the completion of the task performed by Mapper.

- **Reduce:** The main function or task of the Reduce is to gather the Tuple generated from Map and then perform some sorting and aggregation sort of process on those key-value depending on its key element.
- **OutputFormat:** Once all the operations are performed, the key-value pairs are written into the file with the help of record writer, each record in a new line, and the key and value in a space-separated manner.



2. HDFS

HDFS(Hadoop Distributed File System) is utilized for storage permission. It is mainly designed for working on commodity Hardware devices(inexpensive devices), working on a distributed file system design. HDFS is designed in such a way that it believes more in storing the data in a large chunk of blocks rather than storing small data blocks.

HDFS in Hadoop provides Fault-tolerance and High availability to the storage layer and the other devices present in that Hadoop cluster. Data storage Nodes in HDFS.

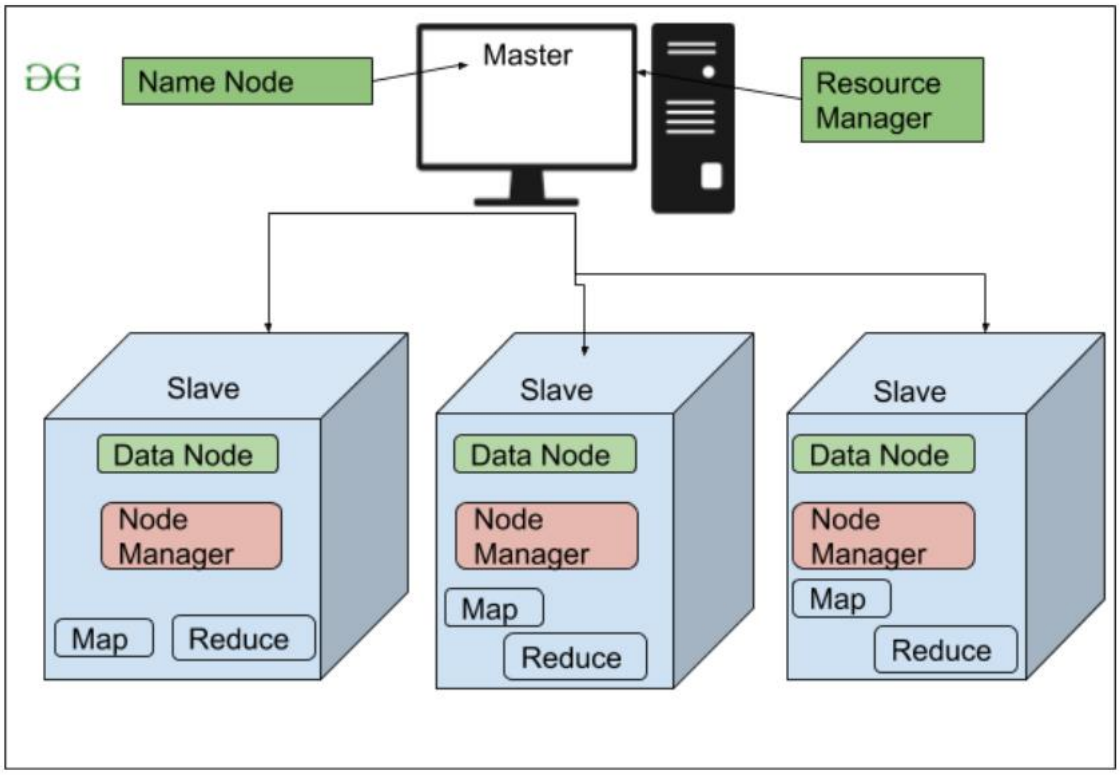
- NameNode(Master)
- DataNode(Slave)

NameNode: NameNode works as a Master in a Hadoop cluster that guides the Datanode(Slaves). Namenode is mainly used for storing the Metadata

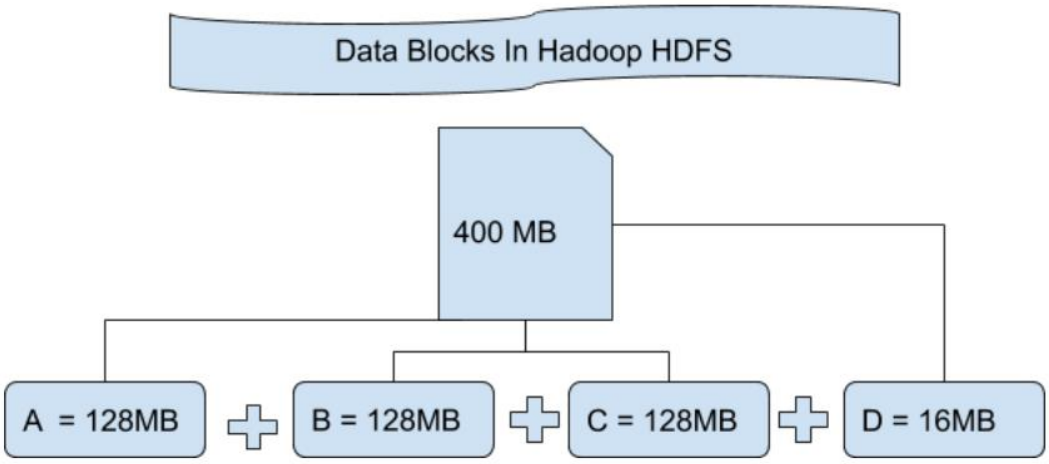
i.e. the data about the data. Meta Data can be the transaction logs that keep track of the user's activity in a Hadoop cluster.
Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication. Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

DataNode: DataNodes works as a Slave 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.

High Level Architecture Of Hadoop



File Block In HDFS: Data in HDFS is always stored in terms of blocks. So the single block of data is divided into multiple blocks of size 128MB which is default and you can also change it manually.



Let's understand this concept of breaking down of file in blocks with an example. Suppose you have uploaded a file of 400MB to your HDFS then what happens is this file got divided into blocks of 128MB+128MB+128MB+16MB = 400MB size. Means 4 blocks are created each of 128MB except the last one. Hadoop doesn't know or it doesn't care about what data is stored in these blocks so it considers the final file blocks as a partial record as it does not have any idea regarding it. In the Linux file system, the size of a file block is about 4KB which is very much less than the default size of file blocks in the Hadoop file system. As we all know Hadoop is mainly configured for storing the large size data which is in petabyte, this is what makes Hadoop file system different from other file systems as it can be scaled, nowadays file blocks of 128MB to 256MB are considered in Hadoop.

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.

As we have seen in File blocks that the HDFS stores the data in the form of various blocks at the same time Hadoop is also configured to make a copy of those file blocks.

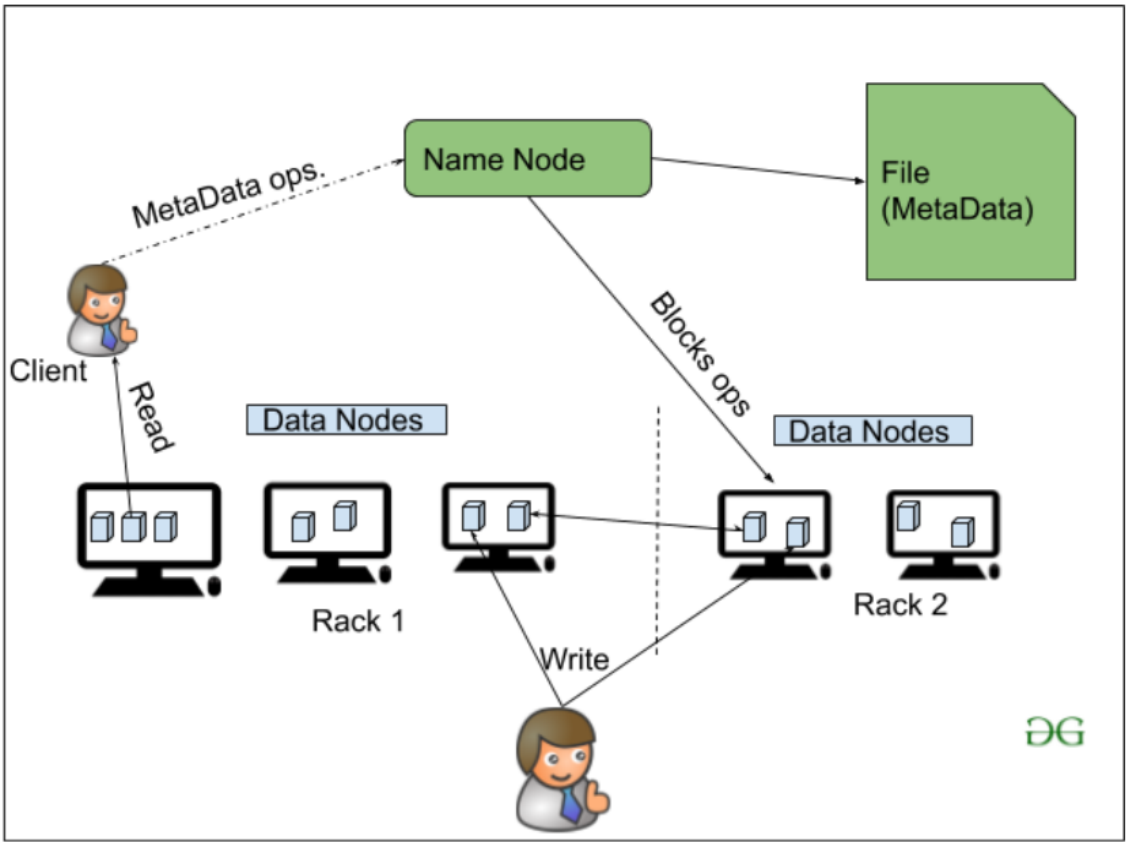
By default, the Replication Factor for Hadoop is set to 3 which can be configured means you can change it manually as per your requirement like in above example we have made 4 file blocks which means that 3 Replica or copy of each file block is made means total of 4×3 = 12 blocks are made for the backup purpose.

This is because for running Hadoop we are using commodity hardware (inexpensive system hardware) which can be crashed at any time. We are not using the supercomputer for our Hadoop setup. That is why we need such a feature in HDFS which can make copies of that file blocks for backup purposes, this is known as fault tolerance.

Now one thing we also need to notice that after making so many replica's of our file blocks we are wasting so much of our storage but for the big brand organization the data is very much important than the storage so nobody cares for this extra storage. You can configure the Replication factor in your *hdfs-site.xml* file.

Rack Awareness The rack is nothing but just the physical collection of nodes in our Hadoop cluster (maybe 30 to 40). A large Hadoop cluster is consists of so many Racks . with the help of this Racks information Namenode chooses the closest Datanode to achieve the maximum performance while performing the read/write information which reduces the Network Traffic.

HDFS Architecture



3. YARN(Yet Another Resource Negotiator)

YARN is a Framework on which MapReduce works. YARN performs 2 operations that are Job scheduling and Resource Management. The Purpose of Job schedular is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and Processing can be Maximized. Job Scheduler also keeps track of which job is important, which job has more priority, dependencies between the jobs and all the other information like job timing, etc. And the use of Resource Manager is to manage all the resources that are made available for running a Hadoop cluster.

Features of YARN

- Multi-Tenancy
- Scalability
- Cluster-Utilization
- Compatibility

4. Hadoop common or Common Utilities

Hadoop common or Common utilities are nothing but our java library and java files or we can say the java scripts that we need for all the other components present in a Hadoop cluster. these utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common verify that Hardware failure in a Hadoop cluster is common so it needs to be solved automatically in software by Hadoop Framework.

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.

S.No.	RDBMS	Hadoop
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.

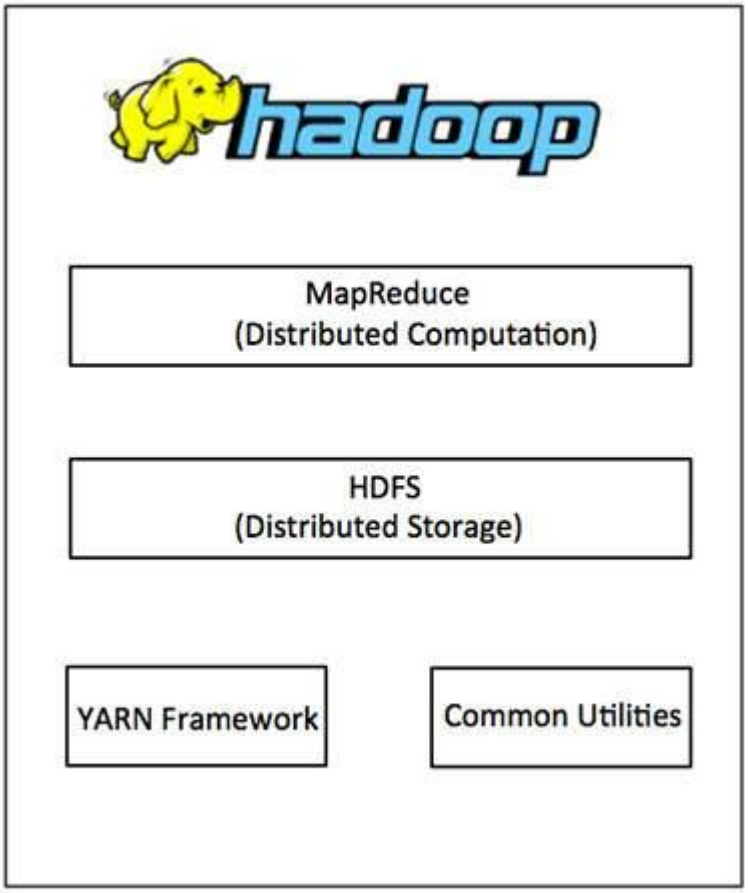
Hadoop Overview

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 –

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).



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.

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.

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

- **Hadoop Common** – These are Java libraries and utilities required by other Hadoop modules.
- **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.

Hadoop distributors

Top Commercial Hadoop Vendors

Here is a list of top Hadoop Vendors who will play a key role in big data market growth for the coming years-

1) Amazon Elastic MapReduce

2) Cloudera CDH Hadoop Distribution

3) Hortonworks Data Platform (HDP)

4) MapR Hadoop Distribution

5) IBM Open Platform

6) Microsoft Azure's HDInsight -Cloud based Hadoop Distrbution

7) Pivotal Big Data Suite

8) Datameer Professional

9) Datastax Enterprise Analytics

10) Dell- Cloudera Apache Hadoop Solution.

1) Amazon Web Services Elastic MapReduce Hadoop Distribution

Amazon EMR (previously known as Amazon Elastic MapReduce) is an Amazon Web Services (AWS) tool for big data processing and analysis. Amazon markets EMR as an expandable, low-configuration service that provides an alternative to running on-premises cluster computing.

Amazon EMR is based on Apache Hadoop, a Java-based programming framework that supports the processing of large data sets in a distributed computing environment. Using MapReduce, a core component of the Hadoop software framework, developers can write programs that process massive amounts of unstructured data across a distributed cluster of processors or standalone computers. It was developed by Google for

indexing webpages and replaced its original indexing algorithms and heuristics in 2004.

Amazon EMR processes big data across a Hadoop cluster of virtual servers on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). The *Elastic* in EMR's name refers to its dynamic resizing ability, which enables administrators to increase or reduce resources, depending on their current needs.

Amazon EMR is used for data analysis in log analysis, web indexing, data warehousing, machine learning (ML), financial analysis, scientific simulation and bioinformatics. It also supports workloads based on Apache Spark, Apache Hive, Presto and Apache HBase -- the latter of which integrates with Hive and Pig, which are open source data warehouse tools for Hadoop. Hive uses queries and analyzes data, and Pig offers a high-level mechanism for programming MapReduce jobs to be executed in Hadoop.

Amazon EMR use cases

There are several ways enterprises can use Amazon EMR, including:

- **Machine learning.** EMR's built-in ML tools use the Hadoop framework to create a variety of algorithms to support decision-making, including decision trees, random forests, support-vector machines and logistic regression.
- **Extract, transform and load.** ETL is the process of moving data from one or more data stores to another. Data transformations -- such as sorting, aggregating and joining -- can be done using EMR.
- **Clickstream analysis.** Clickstream data from Amazon S3 can be analysed with Apache Spark and Apache Hive. Apache Spark is an open source data processing tool that can help make data easy to manage and analyze. Spark uses a framework that enables jobs

to run across large clusters of computers and can process data in parallel. Apache Hive is a data warehouse infrastructure built on top of Hadoop that provides tools for working with data that Spark can analyze. Clickstream analysis can help organizations understand customer behaviors, find ways to improve a website layout, discover which keywords people are using in search engines and see which word combinations lead to sales.

- **Real-time streaming.** Users can analyze events using streaming data sources in real time with Apache Spark Streaming and Apache Flink. This enables streaming data pipelines to be created on EMR.
- **Interactive analytics.** EMR Notebooks are a managed service that provide a secure, scalable and reliable environment for data analytics. Using Jupyter Notebook -- an open source web application data scientists can use to create and share live code and equations -- data can be prepared and visualized to perform interactive analytics.
- **Genomics.** Organizations can use EMR to process genomic data to make data processing and analysis scalable for industries including medicine and telecommunications.

2) Hortonworks Hadoop Distribution

Hortonworks is a pure play Hadoop company that drives open source Hadoop distributions in the IT market.

The main goal of Hortonworks is to drive all its innovations through the Hadoop open data platform and build an ecosystem of partners that speeds up the process of Hadoop adoption amongst enterprises.

Apache Ambari is an example of Hadoop cluster management console developed by Hortonworks Hadoop vendor for provision, managing and monitoring Hadoop clusters.

The Hortonworks Hadoop vendor is reported to attract 60 new customers every quarter with some giant accounts like Samsung, Spotify, Bloomberg and eBay.

Hortonworks has grown its revenue at a rapid pace.

However, the professional services revenue generated by Hortonworks Hadoop vendor increases at a faster pace when compared to support and subscription services revenue.

3) Cloudera Hadoop Distribution

Hadoop is an Apache open-source framework that store and process Big Data in a distributed environment across the cluster using simple programming models. Hadoop provides parallel computation on top of distributed storage.

Since Apache Hadoop is open source, many companies have developed distributions that go beyond the original open source code. This is very akin to Linux distributions such as RedHat, Fedora, and Ubuntu. Each of the Linux distributions supports its own functionalities and features like user-friendly GUI in Ubuntu. Similarly, **Red Hat** is popular within enterprises because it offers support and also provides ideology to make changes to any part of the system at will. Red Hat relieves you from software compatibility problems. This is usually a big issue for users who are transitioning from Windows.

Likewise, there are 3 main types of Hadoop distributions which have its own set of functionalities and features and are built under the base HDFS.

Cloudera is the market trend in Hadoop space and is the first one to release commercial Hadoop distribution. It offers consulting services to bridge the gap between – "what does Apache Hadoop provides" and "what organizations need".

Cloudera Distribution is:

- **Fast for business**: From analytics to data science and everything in between, Cloudera delivers the performance you need to unlock the potential of unlimited data.
- **Makes Hadoop easy to manage**: With Cloudera Manager, automated wizards let you quickly deploy your cluster, irrespective of the scale or deployment environment.

- **Secure without compromise:** Meets stringent data security and compliance needs without sacrificing business agility. Cloudera provides an integrated approach to data security and governance.

4) MapR Hadoop Distribution

The MapR Converged Data Platform supports big data storage and processing through the Apache collection of Hadoop products, as well as its added-value components. These components from MapR Technologies provide several enterprise-grade proprietary tools to better manage and ensure the resiliency and reliability of data in the Hadoop cluster.

These platform components include MapR File System (MapR-FS); MapReduce; and MapR Control System, the product's user interface. The MapR Hadoop distribution includes a complete implementation of the Hadoop APIs, enabling the product to be fully compatible with the Hadoop ecosystem.

Unlike HDFS, which follows the write-once-read-many paradigm, MapR-FS is a fully read/write Portable Operating System Interface-compliant file system.

By supporting industry-standard NFS, users can easily mount a MapR cluster and execute any file-based application directly on the data residing in the cluster. This enables data from nearly any source to be processed and allows for standard tools to be used to directly access data in the cluster without any modifications.

Additionally, unlike other Hadoop distributions, MapR can process distributed files, database tables and event streams all in the same cluster of nodes. This lets organizations run operational tools such as Apache HBase and analytic tools such as Hive or Impala on one cluster, reducing hardware and operational costs.

	Hortonworks	Cloudera	MapR
Performance and Scalability			
Data Ingest	Batch	Batch	Batch and streaming writes
Metadata Architecture	Centralized	Centralized	Distributed
HBase Performance	Latency spikes	Latency spikes	Consistent low latency
NoSQL Applications	Mainly batch applications	Mainly batch applications	Batch and online/real-time applications
Dependability			
High Availability	Single failure recovery	Single failure recovery	Self healing across multiple failures
MapReduce HA	Restart jobs	Restart jobs	Continuous without restart
Upgrading	Planned downtime	Rolling upgrades	Rolling upgrades
Replication	Data	Data	Data + metadata
Snapshots	Consistent only for closed files	Consistent only for closed files	Point-in-time consistency for all files and tables
Disaster Recovery	No	File copy scheduling (BDR)	Mirroring
Manageability			
Management Tools	Ambari	Cloudera Manager	MapR Control System
Volume Support	No	No	Yes
Heat map, Alarms, Alerts	Yes	Yes	Yes
Integration with REST API	Yes	Yes	Yes
Data and Job Placement Control	No	No	Yes
Data Access			
File System Access	HDFS, read-only NFS	HDFS, read-only NFS	HDFS, read/write NFS (POSIX)
File I/O	Append only	Append only	Read/write
Security: ACLs	Yes	Yes	Yes
Wire-level Authentication	Kerberos	Kerberos	Kerberos, Native

5) IBM Infosphere BigInsights Hadoop Distribution

IBM is an analytics platform that enables analysis of a wide variety of unconventional information types and formats at large-scale volumes, without up-front preprocessing. IBM BigInsights V2.0:

- Delivers enterprise Hadoop capabilities with easy-to-use administration and management capabilities, rich developer tools, powerful analytic functions, and the latest versions of Apache Hadoop and associated projects.
- Provides extensive new capabilities by offering enhanced big data tools for visualization, monitoring, development, enhanced functionality for better enterprise integration with other IBM products, and additional platform support for Cloudera.

- Introduces two new Application Accelerators that may help you to a faster ROI: The Social Data Analytics Accelerator and the Machine Data Analytics Accelerator.
- Provides analytics and enterprise functionality on top of Apache Hadoop technology to meet big data enterprise requirements.
- Lets you run with the IBM-provided Apache Hadoop distribution or deploy to a Cloudera cluster.
- Includes data discovery to enable exploratory analysis and modeling of unconventional data types.

New capabilities in IBM InfoSphere BigInsights Enterprise Edition V2.0:

- **Enhanced big data tools: Visualization, monitoring, development**
  - With a consistent, unified, and extensible user interface, the big data tools bring big data collaboration to the enterprise and can help users unlock the value within data by enabling various roles of an organization to collaboratively leverage, discover, and explore large amounts of data. In this release, BigInsights adds the following new features for four major roles:
  - **Business analysts and business users**
    - A centralized dashboard to visualize analytic results, including new charts, BigSheets workbooks, data operations, and to visualize metrics from the monitoring service.
    - The ability to view BigSheets data flows between and across data sets to quickly navigate and relate analysis and charts.
    - BigSheets usability enhancements, including inner outer joins, enhanced filters for BigSheets columns, column data-type mapping for collections, application of analytics to BigSheets columns, data preparation enabling users to define and update schemas for data sources to improve error checking and analysis, and additional charting capabilities.
    - Workflow composition that enables users to compose new applications from existing applications and BigSheets, and to invoke analytics applications from the web console, including integration within BigSheets.
    - New Apps providing enhanced data import capability: a new REST data source App that enables users to load data from any data source supporting REST APIs into BigInsights , including popular social media services; a new Sampling App that enables users to sample data for analysis; and a new Subsetting App that enables users to subset data for data analysis.
  - **Data scientists**
    - A unified tooling environment that supports the data analytics lifecycle by enabling users to sample data and define, test, and deploy analytics applications from the BigInsights Eclipse tools, and to administer, execute, and monitor the deployed applications from the BigInsights Web Console.
    - Integration with R with an App that allows users to execute R scripts directly from the BigInsights Web Console.

        - Extended text analytics capability that performs global analysis to address key use cases such as customer profiling and lead generation.
    - **Administrators**
        - New monitoring capabilities that provide a centralized dashboard view to visualize key performance indicators including CPU, disk, and memory and network usage for the cluster, data services such as HDFS, HBase, Zookeeper, and Flume, and application services including MapReduce, Hive, and Oozie.
        - Enhanced status information and control over the major cluster capabilities, building on the existing server management capabilities.
        - Usability improvements for application permissions and deployment.
        - New capability to view and control all applications from a single page.
    - **Developers**
        - A workflow editor that greatly simplified the creation of complex Oozie workflows with a consumable interface.
        - A Pig editor with content assist and syntax highlighting that enables users to create and execute new applications using Pig in local or cluster mode from the Eclipse IDE.
        - Usability improvements to the Jaql Editor, such as extended support for Jaql syntax, extended content assist, and improved execution feedback.
        - Enablement of BigSheets macro and BigSheets reader development.
        - Enhanced Text Analytics development, including the support for modular rule sets, populating external artifacts, and providing workflow UI extensions for domain and language extensions.
        - Enhanced scope of the development artifacts during the deployment phase, including artifacts for Text Analytics, scripts for Jaql, Hive SQL, Pig, and Derby, and BigSheets macros and readers.
- **Enhanced enterprise integration**
    - **InfoSphere Data Explorer:** InfoSphere BigInsights includes a limited-use license to the included IBM InfoSphere Data Explorer program, which helps organizations discover, navigate, and visualize vast amounts of structured and unstructured information across many enterprise systems and data repositories. It also provides a cost-effective and efficient entry point for exploring the value of big data technologies through a powerful framework for developing applications that leverage existing enterprise data.
    - **InfoSphere Streams:** InfoSphere BigInsights includes a limited-use license to the included InfoSphere Streams program, which enables real-time, continuous analysis of data on the fly. Streams is an enterprise-class stream processing system that can be used to extract

actionable insights from data as it arrives in the enterprise while transforming data and ingesting it into BigInsights at high speeds.

- **Netezza Data Warehouse Appliances:** New UDFs to connect SQL workloads from Netezza with the power of InfoSphere BigInsights . Now you can control the data read and written from or to HDFS via UDFs.
- **Rational® and Data Studio:** By installing the InfoSphere BigInsights Eclipse development tools into your existing Rational Data Studio (RAD), Rational Team Concert™ , or Data Studio environment, Infosphere BigInsights provides the functionality to allow for integration with IBM Rational Application Developer V8.5, Rational Team Concert and Data Studio, enabling more collaborative enterprise big data application development.
- **InfoSphere Guardium® :** InfoSphere BigInsights can be used with InfoSphere Guardium to audit the Web Console and HBase and various other open source logs for added protection and audit purposes.
- **IBM Cognos® Business Intelligence:** InfoSphere BigInsights includes a limited-use license to the included IBM Cognos Business Intelligence program, which enables business users to access and analyze the information they need to improve decision making, gain better insight, and manage performance. IBM Cognos Business Intelligence includes software for query, reporting, analysis, and dashboards, as well as software to gather and organize information from multiple sources.

6) Microsoft Azure's HDInsight Cloud based Hadoop Distribution

Forrester rates Microsoft Hadoop Distribution as 4/5- based on the Big Data Vendor's current Hadoop Distributions, market presence and strategy - with Cloudera and Hortonworks scoring 5/5

Microsoft is an IT organization not known for embracing open source software solutions, but it has made efforts to run this open data platform software on Windows. Hadoop as a service offering by Microsoft's big data solution is best leveraged through its public cloud product -Windows Azure's HDInsight particularly developed to run on Azure. There is another production ready feature of Microsoft named Polybase that lets the users search for information available on SQL Server during the execution of Hadoop queries. Microsoft has great significance in delivering a growing Hadoop Stack to its customers.Microsoft Azure's HDInsight is public-cloud only based product and customers can not run on their own hardware with this.

According to analyst Mike Gualtieri at Forrester: "Hadoop's momentum is unstoppable as its open source roots grow wildly into enterprises. Its refreshingly unique approach to data management is transforming how companies store, process, analyze, and share big data".

Commercial Hadoop Vendors continue to mature overtime with increased worldwide adoption of Big Data technologies and growing vendor revenue. There are several top Hadoop vendors namely Hortonworks, Cloudera, Microsoft and IBM. These Hadoop vendors are facing a tough competition in the open data platform. With the war heating up amongst big data vendors, nobody is sure as to who will top the list of commercial Hadoop vendors. With Hadoop buying cycle on the upswing, Hadoop vendors must capture the market share at a rapid pace to make the venture investors happy.

7) Pivotal Big Data Suite

Pivotal's Hadoop distribution, Pivotal HD is 100 percent Apache compliant, uses other Apache components, and is based on the Open Data Platform. Pivotal GemFire is a distributed data management platform designed for diverse data management situations, but is optimized for high volume, latency-sensitive, mission-critical, transactional systems.

The Pivotal GreenPlum Database is a shared-nothing, massively parallel processing (MPP) database used for business intelligence processing as well as for advanced analytics. Pivotal's HAWQ is an ANSI compliant SQL dialect that supports application portability and the use of data visualization tools such as SAS and Tableau.

8) Datameer Professional

Datameer Professional allows you to ingest, analyze, and visualize terabytes of structured and unstructured data from more than 60 different sources including social media, mobile data, web, machine data, marketing information, CRM data, demographics, and databases to name a few. Datameer also offers you 270 pre-built analytic functions to combine and analyze your unstructured and structured data after ingest.

Datameer focuses on big data analytics in a single application built on top of Hadoop. Datameer features a wizard-based data integration tool, iterative point-and-click analytics, drag-and-drop visualizations, and scales from a single workstation up to thousands of nodes. Datameer is available for all major Hadoop distributions.

9) Datastax Enterprise Analytics

DataStax delivers powerful integrated analytics to 20 of the Fortune 100 companies and well-known companies such as eBay and Netflix. DataStax is

built on open source software technology for its primary services: Apache Hadoop (analytics0, Apache Cassandra (NoSQL distributed database), and Apache Solr (enterprise search).

DataStax made the choice to use Apache Cassandra, which provides an "always-on" capability for DataStax Enterprise (DSE) Analytics. DataStax OpsCenter also offers a web-based visual management system for DSE that allows cluster management, point-and-click provisioning and administration, secured administration, smart data protection, and visual monitoring and tuning.

10) Dell- Cloudera Apache Hadoop Solution.

Dell's Statistica Big Data Analytics is an integrated, configurable, cloud-enabled software platform that you can easily deploy in minutes. You can harvest sentiments from social media and the web and combine that data to better understand market traction and trends. Dell leverages Hadoop, Lucene/Solr search, and Mahout machine learning to bring you a highly scalable analytic solution running on Dell PowerEdge servers.

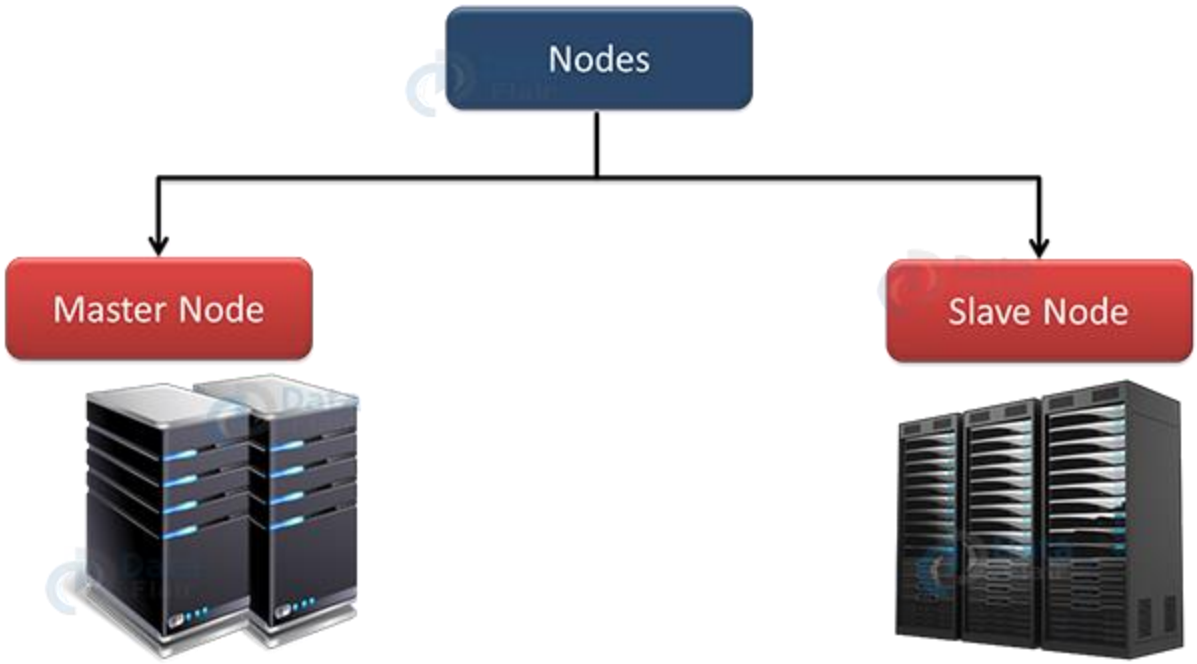
Dell summarizes its hardware software requirements for your Hadoop cluster simply as, 2 – 100 Linux servers for Hadoop Cluster, 6GB RAM, 2+ Core, 1TB HDD per server. The point is that entry into a Hadoop solution is simple and inexpensive. And as Dell puts it, "Gain robust big data analytics on an open and easily deployed platform."

HDFS

Hadoop Distributed File system – HDFS is the world's most reliable storage system. HDFS is a Filesystem of Hadoop designed for storing very large files running on a cluster of commodity hardware. It is designed on the principle of storage of less number of large files rather than the huge number of small files.
Hadoop HDFS provides a fault-tolerant storage layer for Hadoop and its other components. HDFS Replication of data helps us to attain this feature. It stores data reliably, even in the case of hardware failure. It provides high throughput access to application data by providing the data access in parallel.

HDFS Nodes

As we know, Hadoop works in **master-slave** fashion, HDFS also has two types of nodes that work in the same manner. These are the **NameNode(s)** and the **DataNodes**.



1. HDFS Master (Namenode)

NameNode regulates file access to the clients. It maintains and manages the slave nodes and assigns tasks to them. NameNode executes file system namespace operations like opening, closing, and renaming files and directories.

NameNode runs on the high configuration hardware.

2. HDFS Slave (Datanode)

There are n number of slaves (where n can be up to 1000) or DataNodes in the Hadoop Distributed File System that manages storage of data. These slave nodes are the actual worker nodes that do the tasks and serve read and write requests from the file system's clients.

They perform block creation, deletion, and replication upon instruction from the NameNode. Once a block is written on a DataNode, it replicates it to other DataNode, and the process continues until creating the required number of replicas.

DataNodes runs on commodity hardware having an average configuration.

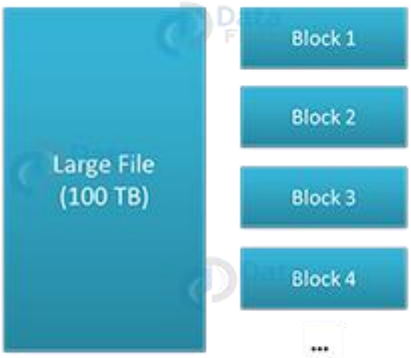
4. Hadoop HDFS Daemons

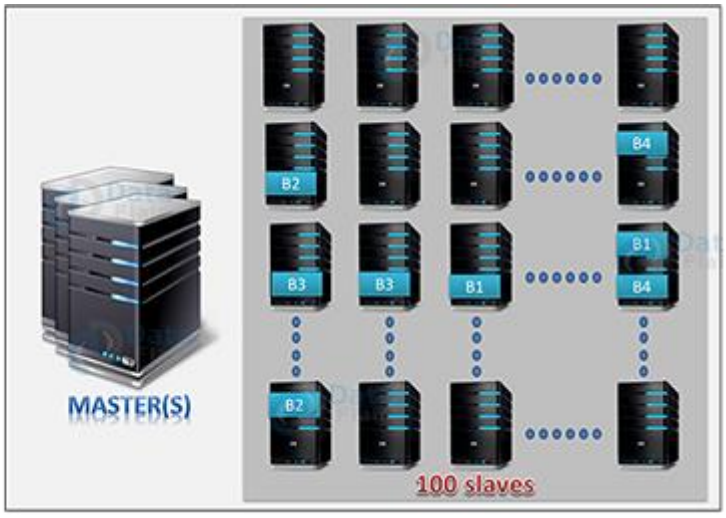
There are two daemons which run on HDFS for data storage:

- **Namenode:** This is the daemon that runs on all the masters. NameNode stores metadata like filename, the number of blocks, number of replicas, a location of blocks, block IDs, etc. This metadata is available in memory in the master for faster retrieval of data. In the local disk, a copy of the metadata is available for persistence. So NameNode memory should be high as per the requirement.
- **Datanode:** This is the daemon that runs on the slave. These are actual worker nodes that store the data.

Data storage in HDFS

Hadoop HDFS broke the files into small pieces of data known as blocks. The default block size in HDFS is 128 MB. We can configure the size of the block as per the requirements. These blocks are stored in the cluster in a distributed manner on different nodes. This provides a mechanism for MapReduce to process the data in parallel in the cluster.





HDFS stores multiple copies of each block across the cluster on different nodes. This is a replication of data. By default, the HDFS replication factor is
3. Hadoop HDFS provides high availability, fault tolerance, and reliability.

HDFS splits a large file into n number of small blocks and stores them on different DataNodes in the cluster in a distributed manner. It replicates each block and stored them across different DataNodes in the cluster.

Rack Awareness in Hadoop HDFS

Hadoop runs on a cluster of computers spread commonly across many racks.

Name Node places replicas of a block on multiple racks for improved fault tolerance.

Name Node tries to place at least one replica of a block in a different rack so that if a complete rack goes down, then also the system will be highly available.

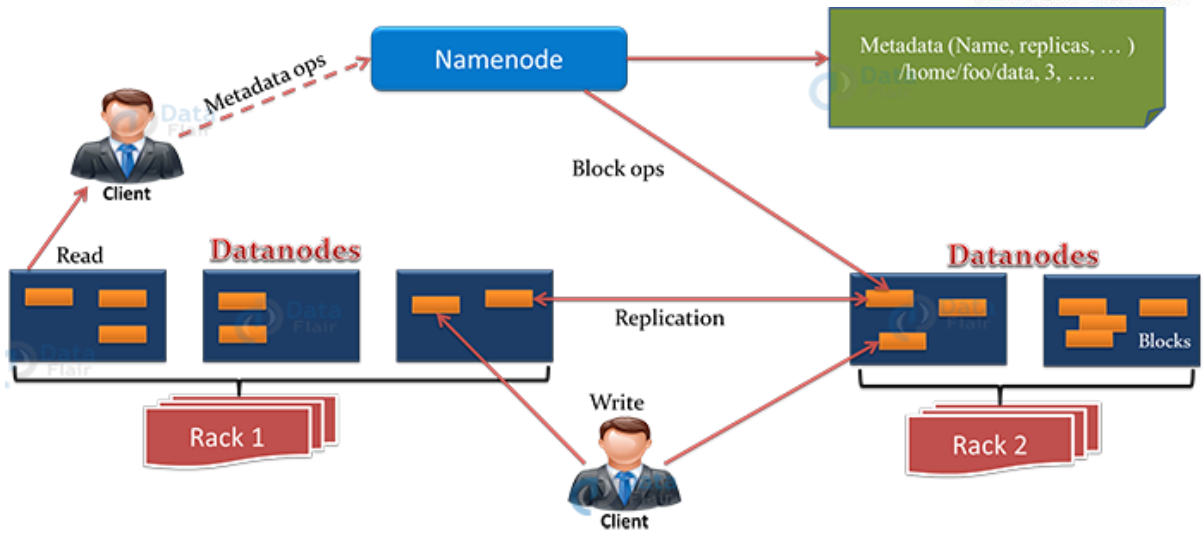
Optimize replica placement distinguishes HDFS from other distributed file systems. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization.

For more understanding, learn **Rack Awareness** in detail.

HDFS Architecture

This architecture gives you a complete picture of the Hadoop Distributed File System. There is a single Name Node that stores metadata, and there are multiple Data Nodes that do actual storage work. Nodes are arranged in racks, and replicas of data blocks are stored on different racks in the cluster to provide fault tolerance.

In the remaining section of this tutorial, we will see how read and write operations are performed in HDFS? To read or write a file in HDFS, the client needs to interact with Name Node. HDFS applications need a write-once-read-many access model for files. A file, once created and written, cannot be edited.



Name Node stores metadata, and Data Node stores actual data. The client interacts with Name Node for performing any tasks, as Name Node is the center piece in the cluster.

There are several Data Nodes in the cluster which store HDFS data in the local disk. Data Node sends a heartbeat message to Name Node periodically to indicate that it is alive. Also, it replicates data to other Data Node as per the replication factor.

Hadoop HDFS Features

a. Distributed Storage

HDFS stores data in a distributed manner. It divides the data into small pieces and stores it on different DataNodes in the cluster. In this manner, the Hadoop Distributed File System provides a way to MapReduce to process a subset of large data sets broken into blocks, parallelly on several nodes. MapReduce is the heart of Hadoop, but HDFS is the one who provides it all these capabilities.

b. Blocks

HDFS splits huge files into small chunks known as blocks. Block is the smallest unit of data in a filesystem. We (client and admin) do not have any control on the block like block location. NameNode decides all such things.

HDFS default block size is 128 MB. We can increase or decrease the block size as per our need. This is unlike the OS filesystem, where the block size is 4 KB.

If the data size is less than the block size of HDFS, then block size will be equal to the data size.

For example, if the file size is 129 MB, then 2 blocks will be created for it. One block will be of default size 128 MB, and the other will be 1 MB only and not 128 MB as it will waste the space (here block size is equal to data size). Hadoop is intelligent enough not to waste the rest of 127 MB. So it is allocating 1 MB block only for 1 MB data.

The major advantage of storing data in such block size is that it saves disk seek time and another advantage is in the case of processing as mapper processes 1 block at a time. So 1 mapper processes large data at a time.

c. Replication

Hadoop HDFS creates duplicate copies of each block. This is known as replication. All blocks are replicated and stored on different DataNodes across the cluster. It tries to put at least 1 replica in a different rack.

HDFS Daemons:

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:

- NameNode
- DataNode
- Secondary Name Node
- Resource Manager
- 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
```

The **namenode** daemon is a master daemon and is responsible for storing all the location information of the files present in HDFS. The actual data is never stored on a namenode. In other words, it holds the metadata of the files in HDFS.

The name node maintains the entire metadata in RAM, which helps clients receive quick responses to read requests. Therefore, it is important to run name node from a machine that has lots of RAM at its disposal. The higher the number of files in HDFS, the higher the consumption of RAM. The name node daemon also maintains a persistent checkpoint of the metadata in a file stored on the disk called the `fsimage` file.

Whenever a file is placed/deleted/updated in the cluster, an entry of this action is updated in a file called the `edits` logfile. After updating the `edits` log, the metadata present in-memory is also updated accordingly. It is important to note that the `fsimage` file is not updated for every write operation.

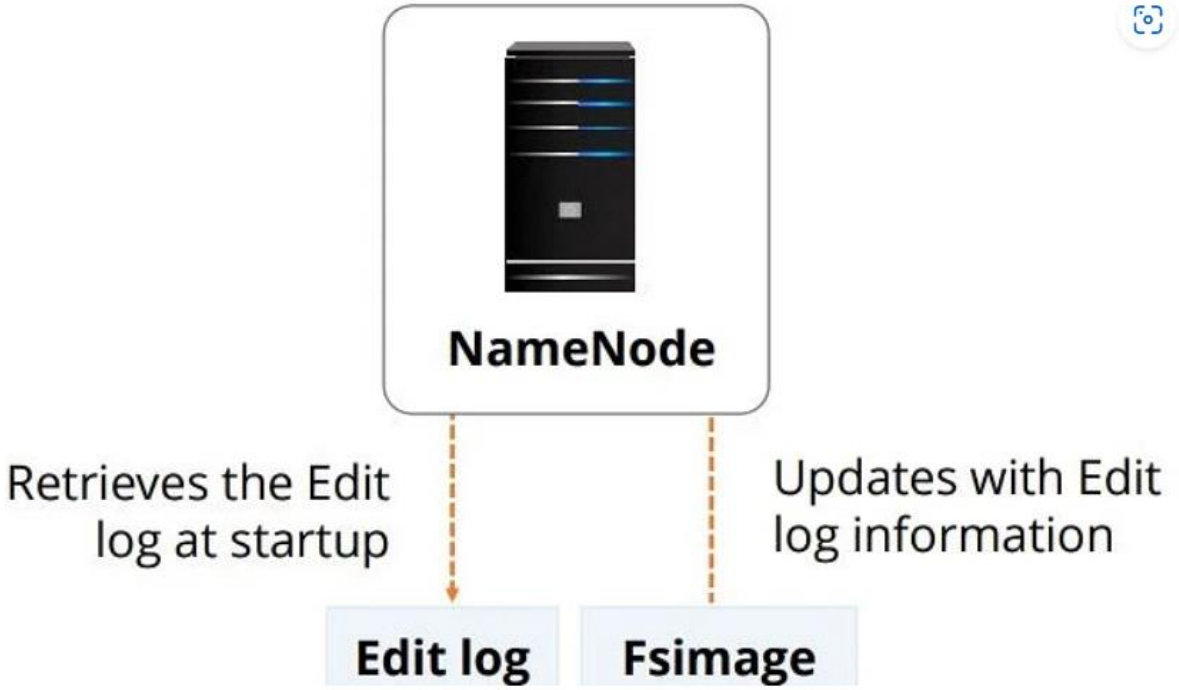
In case the name node daemon is restarted, the following sequence of events occur at name node boot up:

1. Read the `fsimage` file from the disk and load it into memory (RAM).
2. Read the actions that are present in the `edits` log and apply each action to the in-memory representation of the `fsimage` file.
3. Write the modified in-memory representation to the `fsimage` file on the disk.

The preceding steps make sure that the in-memory representation is up to date.

The namenode daemon is a single point of failure in Hadoop 1.x, which means that if the node hosting the namenode daemon fails, the filesystem becomes unusable. To handle this, the administrator has to configure the namenode to write the fsimage file to the local disk as well as a remote disk on the network. This backup on the remote disk can be used to restore the namenode on a freshly installed server. Newer versions of Apache Hadoop (2.x) now support **High Availability** (**HA**), which deploys two namenodes in an active/passive configuration, wherein if the active namenode fails, the control falls onto the passive namenode, making it active. This configuration reduces the downtime in case of a namenode failure.

Since the fsimage file is not updated for every operation, it is possible the edits logfile would grow to a very large file. The restart of namenode service would become very slow because all the actions in the large edits logfile will have to be applied on the `fsimage` file. The slow boot up time could be avoided using the secondary namenode daemon.



The namespace image and the edit log stores information of the data and the metadata. NameNode also determines the linking of blocks to DataNodes. Furthermore, the NameNode is a single point of failure. The DataNode is a multiple instance server. There can be several numbers of DataNode servers. The number depends on the type of network and the storage system.

The DataNode servers, stores, and maintains the data blocks. The NameNode Server provisions the data blocks on the basis of the type of job submitted by the client.

DataNode also stores and retrieves the blocks when asked by clients or the NameNode. Furthermore, it reads/writes requests and performs block creation, deletion, and replication of instruction from the NameNode. There can be only one Secondary NameNode server in a cluster. Note that you cannot treat the Secondary NameNode server as a disaster recovery server. However, it partially restores the NameNode server in case of a failure.

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.

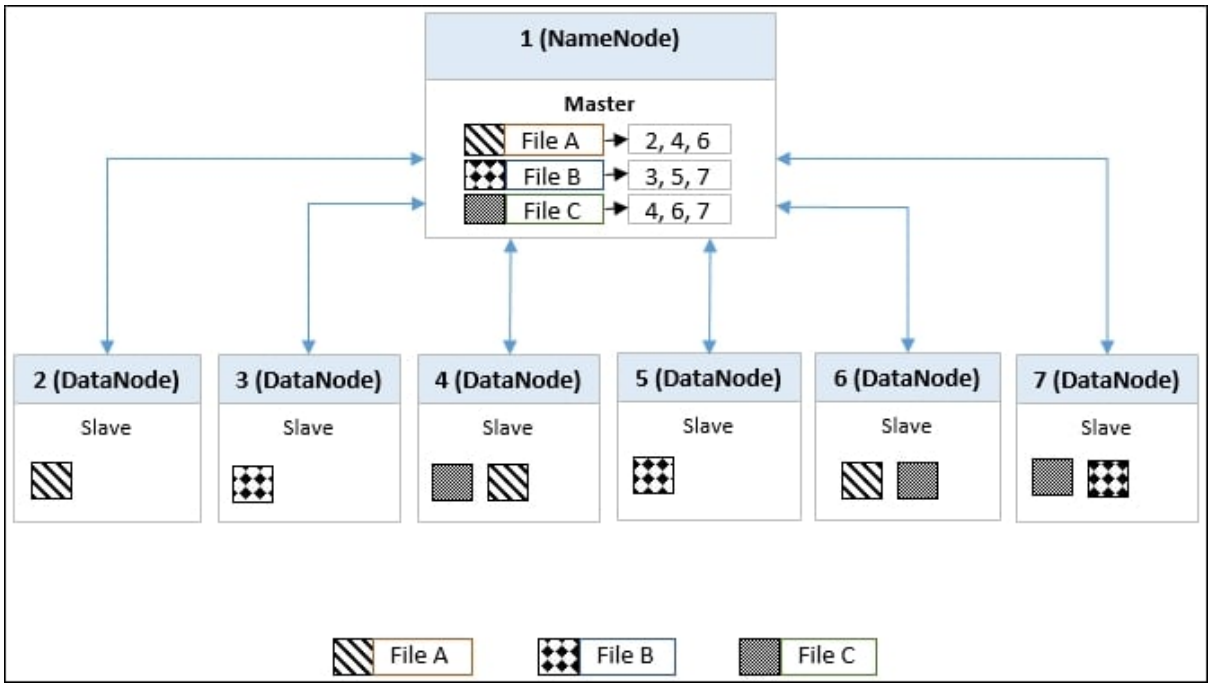
How to start Data Node?

```
hadoop-daemon.sh start datanode
```

How to stop Data Node?

```
hadoop-daemon.sh stop datanode
```

The **datanode** daemon acts as a slave node and is responsible for storing the actual files in HDFS. The files are split as data blocks across the cluster. The blocks are typically 64 MB to 128 MB size blocks. The block size is a configurable parameter. The file blocks in a Hadoop cluster also replicate themselves to other datanodes for redundancy so that no data is lost in case a datanode daemon fails. The datanode daemon sends information to the namenode daemon about the files and blocks stored in that node and responds to the namenode daemon for all filesystem operations. The following diagram shows how files are stored in the cluster:



File blocks of files A, B, and C are replicated across multiple nodes of the cluster for redundancy. This ensures availability of data even if one of the nodes fail. You can also see that blocks of file A are present on nodes 2, 4, and 6; blocks of

file B are present on nodes 3, 5, and 7; and blocks of file C are present on 4, 6, and 7. The replication factor configured for this cluster is 3, which signifies that each file block is replicated three times across the cluster. It is the responsibility of the namenode daemon to maintain a list of the files and their corresponding locations on the cluster. Whenever a client needs to access a file, the namenode daemon provides the location of the file to client and the client, then accesses the file directly from the data node daemon.

3. **secondary Name Node** 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.

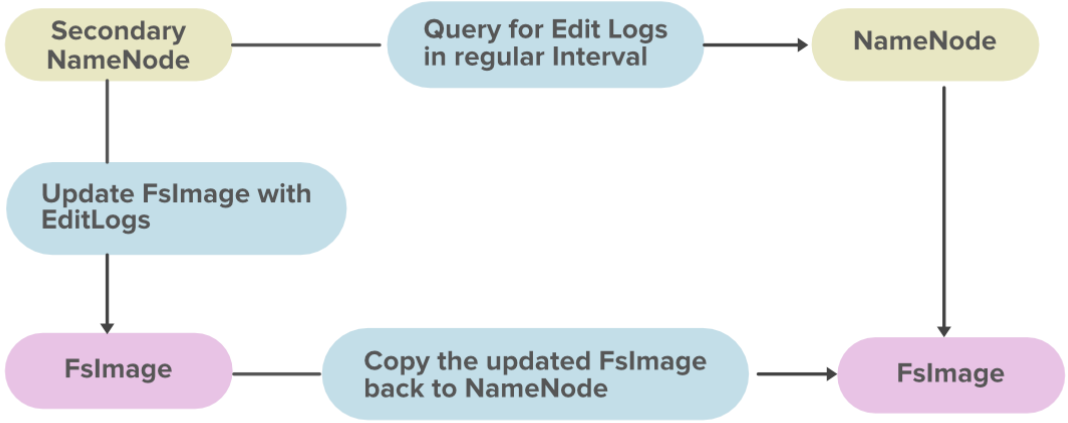
A new Meta Data is assigned to that new system and a new Master is created with this Meta Data, and the cluster is made to run again correctly. This is the benefit of Secondary Name Node.

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.

1. Applications Manager
2. 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
```

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?

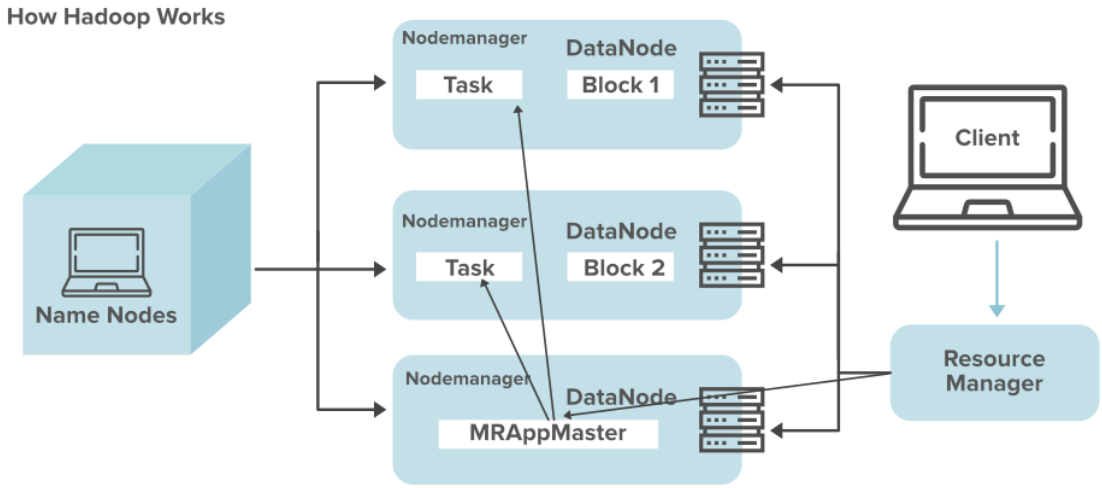
```
yarn-daemon.sh start node manager
```

How to stop Node Manager?

```
yarn-daemon.sh stop nodemanager
```

The Hadoop Daemon's	Port
ResourceManager	8088
NodeManager	8042

The below diagram shows how Hadoop works.



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:

- HDFS (Hadoop Distributed File System)
- YARN (Yet Another Resource Negotiator)

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.

Some of the characteristics of HDFS are:

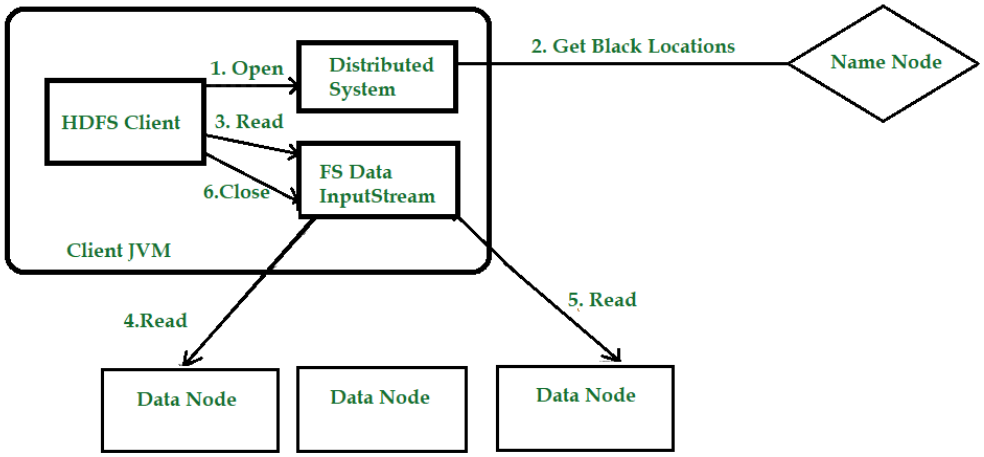
Fault-Tolerance
Scalability
Distributed Storage
Reliability
High availability
Cost-effective
High throughput

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:



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

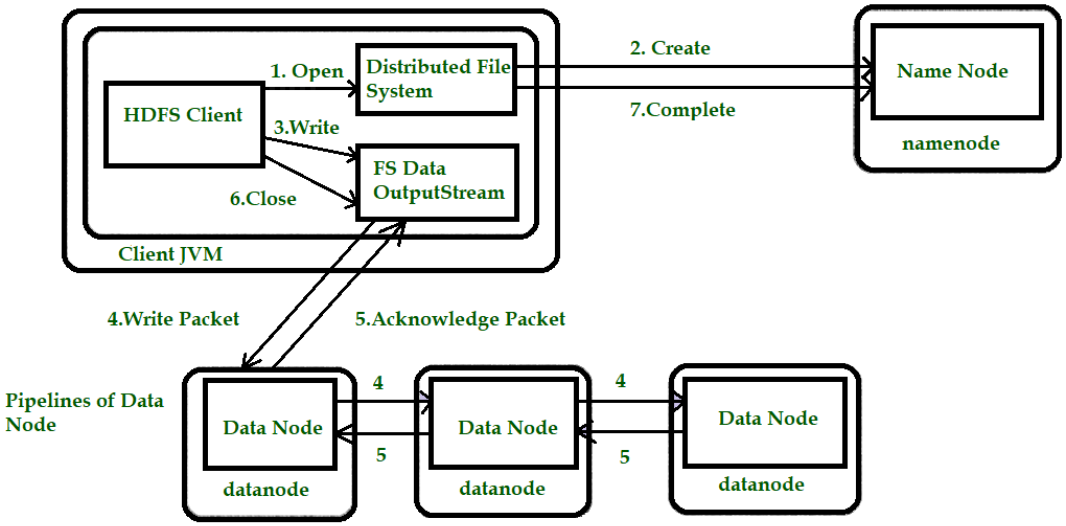
reads through the stream. It will also call the name node to retrieve the data node locations for the next batch of blocks as needed.

Step 6: When the client has finished reading the file, a function is called, close() on the FSDataInputStream.

Anatomy of File Write in HDFS

Next, we'll check out how files are written to HDFS. Consider figure 1.2 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.



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.

Name Node

NameNode works as Master in Hadoop cluster. Below listed are the main function performed by NameNode:

1. Stores metadata of actual data. E.g. Filename, Path, No. of Data Blocks, Block IDs, Block Location, No. of Replicas, Slave related configuration
2. Manages File system namespace.
3. Regulates client access request for actual file data file.
4. Assign work to Slaves(DataNode).
5. Executes file system name space operation like opening/closing files, renaming files and directories.
6. As Name node keep metadata in memory for fast retrieval, the huge amount of memory is required for its operation. This should be hosted on reliable hardware.

1. NameNode is the centerpiece of HDFS.
2. NameNode is also known as the Master

NameNode only stores the metadata of HDFS – the directory tree of all files in the file system, and tracks the files across the cluster.

3. NameNode does not store the actual data or the dataset. The data itself is actually stored in the DataNodes.
4. NameNode knows the list of the Blocks and its location for any given file in HDFS. With this information NameNode knows how to construct the file from blocks.
5. NameNode is so critical to HDFS and when the NameNode is down, HDFS/Hadoop cluster is inaccessible and considered down.
6. NameNode is a single point of failure in Hadoop cluster.
7. NameNode is usually configured with a lot of memory (RAM). Because the block locations are held in main memory

HDFS NameNode
1. NameNode is the main central component of HDFS architecture framework.
2. NameNode is also known as Master node.
3. HDFS Namenode stores meta-data i.e. number of data blocks, file name, path, Block IDs, Block location, no. of replicas, and also Slave related configuration. This meta-data is available in memory in the master for faster retrieval of data.
4. NameNode keeps metadata related to the file system namespace in memory, for quicker response time. Hence, more memory is needed. So NameNode configuration should be deployed on reliable configuration.
5. NameNode maintains and manages the slave nodes, and assigns tasks to them.
6. NameNode has knowledge of all the DataNodes containing data blocks for a given file.
7. NameNode coordinates with hundreds or thousands of data nodes and serves the requests coming from client applications.

Two files 'FSImage' and the 'EditLog' are used to store metadata information.

FsImage: It is the snapshot the file system when Name Node is started. It is an "Image file". FsImage contains the entire filesystem namespace and stored as a file in the NameNode's local file system. It also contains a serialized form of all the directories and file inodes in the filesystem. Each inode is an internal representation of file or directory's metadata.

EditLogs: It contains all the recent modifications made to the file system on the most recent FsImage. NameNode receives a create/update/delete request from the client. After that this request is first recorded to edits file.

Functions of NameNode in HDFS

1. It is the master daemon that maintains and manages the DataNodes (slave nodes).
2. It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc.
3. It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
4. It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
5. It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
6. The NameNode is also responsible to take care of the replication factor of all the blocks.
7. In case of the DataNode failure, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

Secondary Name Node

What is Secondary Name Node?

Role of Secondary Namenode in Managing the Filesystem Metadata.

Each and every transaction that occurs on the file system is recorded within the edit log file. At some point of time this file becomes very large.

Namenode holds the metadata for HDFS like Block information, size etc. This Information is stored in main memory as well as disk for persistence storage . The information is stored in 2 different files .They are

- **Editlogs-** It keeps track of each and every changes to HDFS.
- **Fsimage-** It stores the snapshot of the file system.

Any changes done to HDFS gets noted in the edit logos the file size grows where as the size of fsimage remains same.

This not have any impact until we restart the server. When we restart the server the edit file logs are written into fsimage file and loaded into main memory which takes some time.

If we restart the cluster after a long time there will be a vast down time since the edit log file would have grown. Secondary name node would come into picture in rescue of this problem.

Secondary Name node simply gets edit logs from name node periodically and copies to fsimage. This new fsimage is copied back to namenode. Namenode now, this uses this new fsimage for next restart which reduces the startup time.

It is a helper node to Name node and to precise Secondary Name node whole purpose is to have checkpoint in HDFS, which helps name node to function effectively. Hence, It is also called as **Checkpoint node**.

Now there are two important files which reside in the namenode' s current directory,

1. FsImage file :-This file is the snapshot of the HDFS metadata at a certain point of time .
2. Edits Log file :-This file stores the records for changes that have been made in the HDFS namespace .

The main function of the **Secondary namenode** is to store the latest copy of the FsImage and the Edits Log files.

How does it help?

When the namenode is restarted , the latest copies of the Edits Log files are applied to the FsImage file in order to keep the HDFS metadata latest. So it becomes very important to store a copy of these two files , which is done by secondary namenode.

Now to keep latest versions of these two files, the secondary name node takes the checkpoints at hourly basis which is the default time gap .

Checkpoint:-

A checkpoint is nothing but the updation of the latest FsImage file by applying the latest Edits Log files to it .If the time gap of a checkpoint is large the there

will be too many Edits Log files generated and it will be very cumbersome and time consuming to apply them all at once on the latest FsImage file . And this may lead to acute start time for the primary namenode after a reboot .

However, the secondary namenode is just a helper to the primary namenode in a HDFS cluster as it cannot perform all the functions of the primary namenode .

Note:-
There are two options to which can be used along with secondary namenode command

1. -geteditsize:- this option helps to find the current size of the edit_ingress file present in namenode' s current directory.Here edit_ingress file is the ongoing in progress Edits Log file .
2. -checkpoint [force]:- this option forcefully checkpoints the secondary namenode to the latest state of the primary namenode , whatever may the size of the Edits Log file may be. But ideally the size of the Edits Log file should be greater than or equal to the checkpoint file size .

Namenode holds the meta data for the HDFS like Namespace information, block information etc. When in use, all this information is stored in main memory. But these information also stored in disk for persistence storage.

fsimage – Its the snapshot of the filesystem when namenode started
Edit logs – Its the sequence of changes made to the filesystem after namenode started.

Only in the restart of namenode , edit logs are applied to fsimage to get the latest snapshot of the file system. But namenode restart are rare in production clusters which means edit logs can grow very large for the clusters where namenode runs for a long period of time. The following issues we will encounter in this situation.

1.Editlog become very large , which will be challenging to manage it
2.Namenode restart takes long time because lot of changes has to be merged
3.In the case of crash, we will lost huge amount of metadata since fsimage is very old

Secondary Namenode helps to overcome the above issues by taking over responsibility of merging editlogs with fsimage from the namenode.
1.It gets the edit logs from the namenode in regular intervals and applies to fsimage.

2.Once it has new fsimage, it copies back to namenode
3.Namenode will use this fsimage for the next restart,which will reduce the startup time.

Things have been changed over the years especially with Hadoop 2.x. Now Namenode is highly available with fail over feature. Secondary Namenode is optional now & Standby Namenode has been to used for failover process. **Standby NameNode** will stay up-to-date with all the file system changes the Active NameNode makes .

The **Secondary namenode** is a helper node in hadoop, To understand the functionality of the secondary namenodelet's understand how the namenode works.

Name node stores metadata like file system namespace information, blockinformation etc in the memory.It also stores the persistent copy of the same on the disk. Name node stores information in two files.

fsimage: It's a snapshot of the file system, stores information like modification time access time, permission, replication.

Edit logs: It stores details of all the activities/transactions being performed on the HDFS..

When the namenode is in the active state the edit logs size grows continuously as the edit logs can only be applied to the fsimage at the time of name node restart, to get the latest state of the HDFS. If edit logs grows significantly and name node tries to apply it on fsimage at the time of name node restart, the process can take very long, here secondary node come into the play.

Secondary namenode keeps the checkpoint on the name node, It reads the edit logs from the namenode continuously after a specific interval and applies it to the fsimage copy of secondary name node. In this way the fsimage file will have the most recent state of HDFS.
The secondary namenode copies new fsimage to primary, so fsimage is updated.

Since fsimage is updated, there will be no overhead of copying of edit logs at the moment of restarting the cluster.
Secondary namenode is a helper node and can't replace the name node.

Data Node

HDFS Data Node
1. Data Node is also known as Slave node.
2. In Hadoop HDFS Architecture, Data Node stores actual data in HDFS.
3. Data Nodes responsible for serving, read and write requests for the clients.
4. Data Nodes can deploy on commodity hardware.
5. Data Nodes sends information to the Name Node about the files and blocks stored in that node and responds to the Name Node for all filesystem operations.
6. When a Data Node starts up it announce itself to the Name Node along with the list of blocks it is responsible for.
7. Data Node is usually configured with a lot of hard disk space. Because the actual data is stored in the Data Node.

Functions of Data Node in HDFS
1. These are slave daemons or process which runs on each slave machine.
2. The actual data is stored on Data Nodes.
3. The Data Nodes perform the low-level read and write requests from the file system's clients.
4. Every Data Node sends a heartbeat message to the Name Node every 3 seconds and conveys that it is alive. In the scenario when Name Node does not receive a heartbeat from a Data Node for 10 minutes, the Name Node considers that particular Data Node as dead and starts the process of Block replication on some other Data Node..
5. All Data Nodes are synchronized in the Hadoop cluster in a way that they can communicate with one another and make sure of
i. Balancing the data in the system
ii. Move data for keeping high replication
iii. Copy Data when required

Basic Operations of Datanode:

- Datanodes is responsible of storing actual data.

- Upon instruction from Namenode, it performs operations like creation/replication/deletion of data blocks.
- When one of Datanode gets down then it will not make any effect on Hadoop cluster due to **replication**.
- All Datanodes are synchronized in the Hadoop cluster in a way that they can communicate with each other for various operations.

What happens if one of the Datanodes gets failed in HDFS?

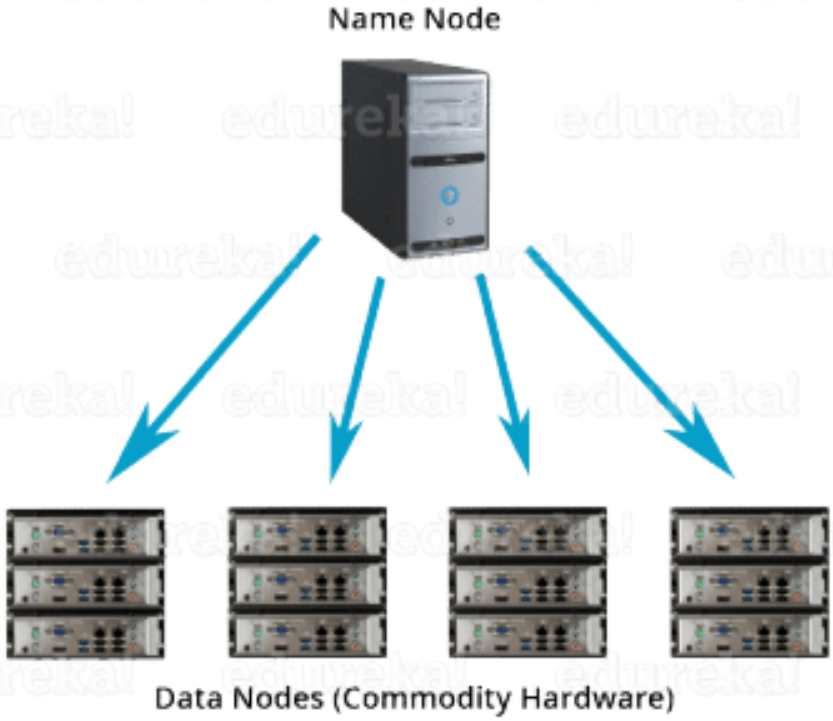
Namenode periodically receives a heartbeat and a Block report from each Datanode in the cluster. Every Datanode sends heartbeat message after every **3 seconds** to Namenode.

The health report is just information about a particular Datanode that is working properly or not. In the other words we can say that particular Datanode is alive or not.

A block report of a particular Data node contains information about all the blocks on that resides on the corresponding Data node. When Name node doesn't receive any heartbeat message for **10 minutes**(ByDefault) from a particular Data node then corresponding Data node is considered Dead or failed by Name node.

Since blocks will be under replicated, the system starts the replication process from one Data node to another by taking all block information from the Block report of corresponding Datanode. The Data for replication transfers directly from one Data node to another without data passing through Name node.

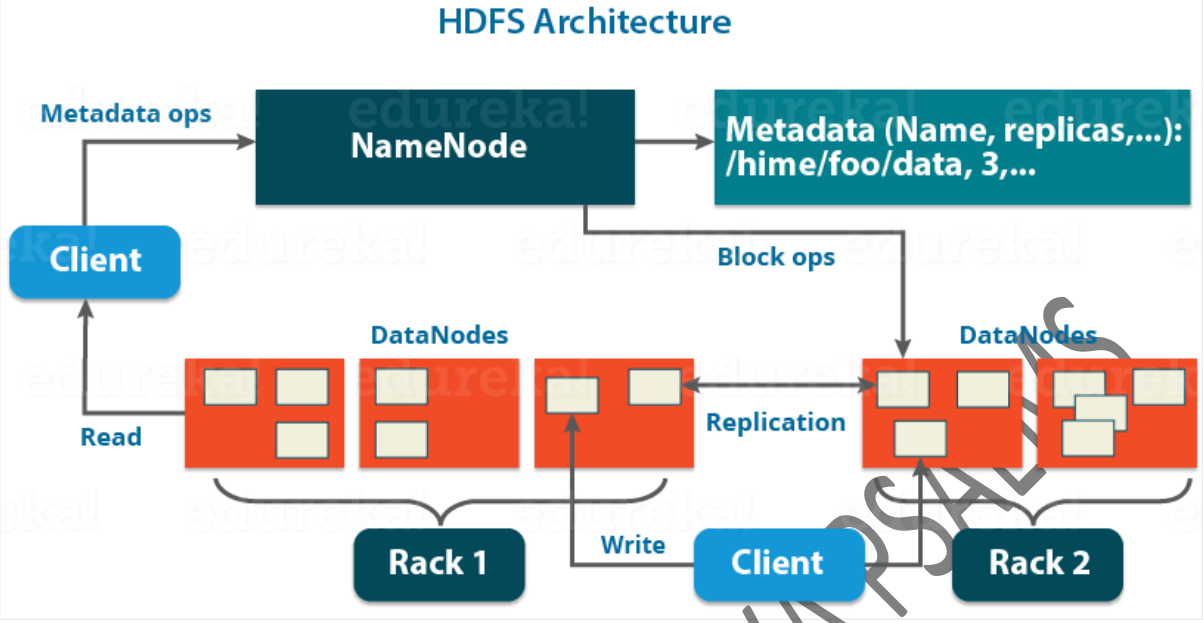
HDFS Architecture



Apache HDFS or **Hadoop Distributed File System** is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines.

Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

NameNode:



NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. I will be discussing this High Availability feature of Apache Hadoop HDFS in my next blog. The HDFS architecture is built in such a way that the user data never resides on the NameNode. The data resides on DataNodes only.

Functions of NameNode:

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata:
  - **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
  - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.
- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.

- The NameNode is also responsible to take care of the **replication factor** of all the blocks which we will discuss in detail later in this HDFS tutorial blog.
- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

Understand the various properties of Namenode, Datanode and Secondary Namenode from the Hadoop Administration Course.

DataNode:

DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

Functions of DataNode:

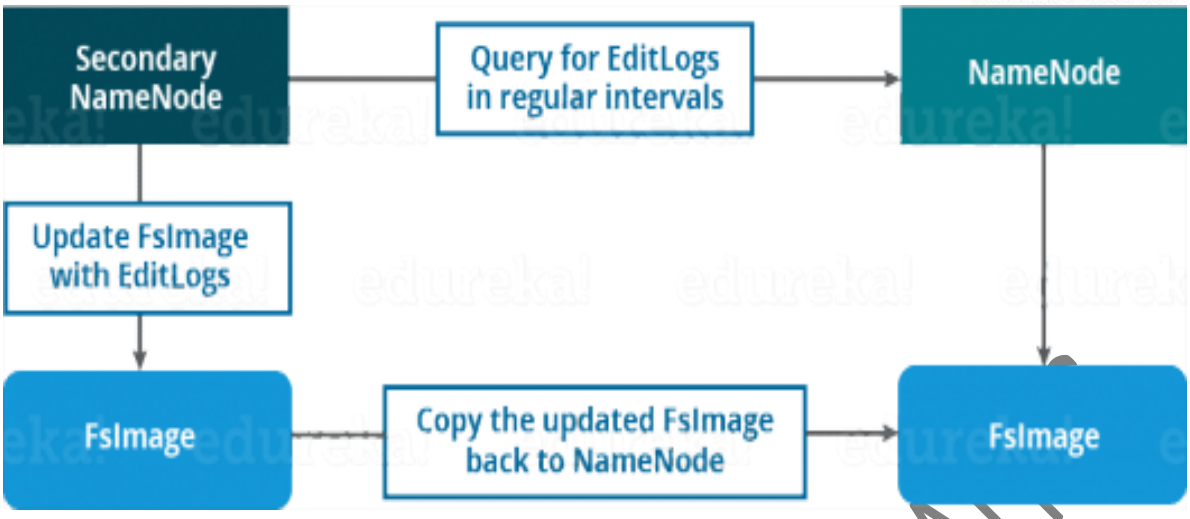
- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Till now, you must have realized that the NameNode is pretty much important to us. If it fails, we are doomed. But don't worry, we will be talking about how Hadoop solved this single point of failure problem in the next Apache Hadoop HDFS Architecture blog. So, just relax for now and let's take one step at a time.

Learn more about Big Data and its applications from the Data Engineer certification.

Secondary NameNode:

Apart from these two daemons, there is a third daemon or a process called Secondary NameNode. The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon.** And don't be confused about the Secondary NameNode being a **backup NameNode because it is not.**



Functions of Secondary NameNode:

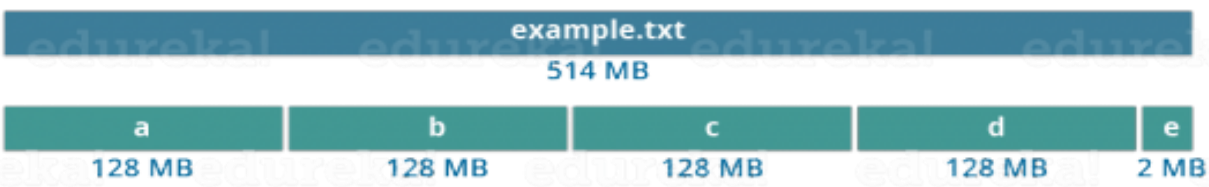
- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

Blocks:

Now, as we know that the data in HDFS is scattered across the DataNodes as blocks. **Let's have a look at what is a block and how is it formed?**

Blocks are the nothing but the smallest continuous location on your hard drive where data is stored. In general, in any of the File System, you store the data as a collection of blocks. Similarly, HDFS stores each file as blocks which are scattered throughout the Apache Hadoop cluster. The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x) which you can configure as per your requirement.



It is not necessary that in HDFS, each file is stored in exact multiple of the configured block size (128 MB, 256 MB etc.). Let's take an example where I have a file "example.txt" of size 514 MB as shown in above figure. Suppose that we are using the default configuration of block size, which is 128 MB. Then, how many blocks will be created? 5, Right. The first four blocks will be of 128 MB. But, the last block will be of 2 MB size only.

Now, you must be thinking why we need to have such a huge blocks size i.e. 128 MB?

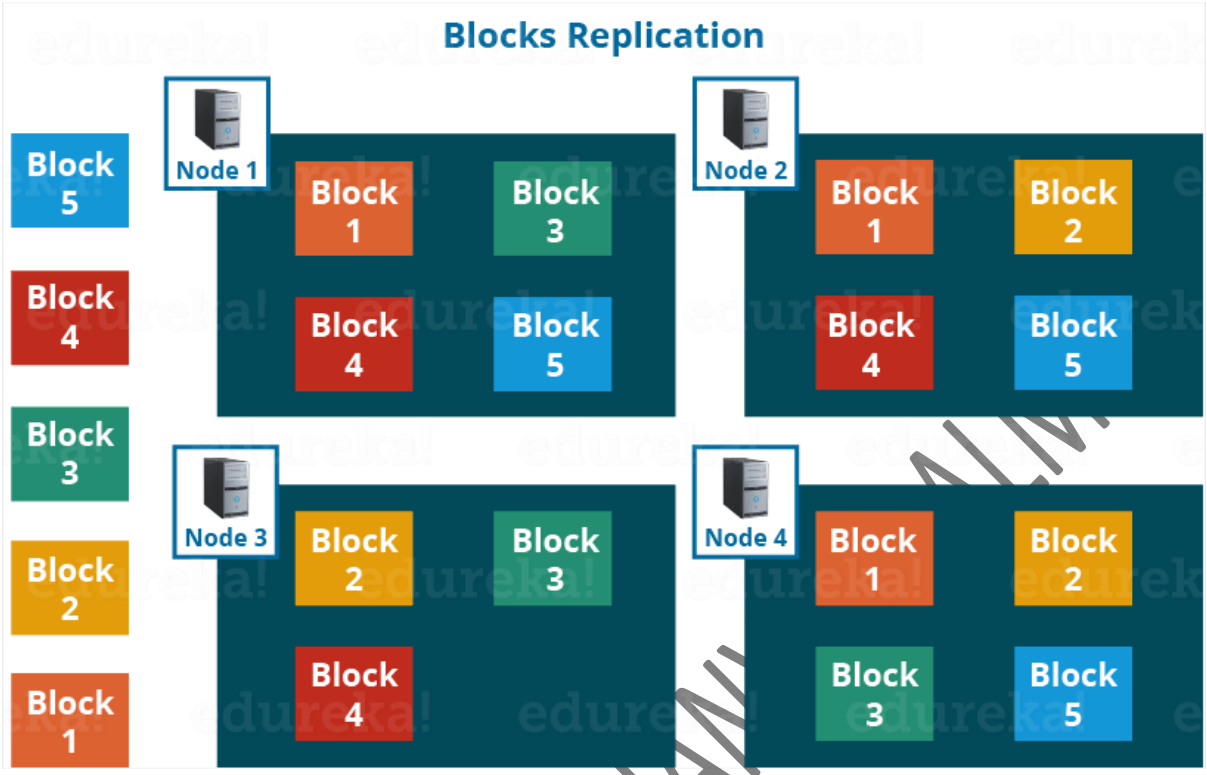
Well, whenever we talk about HDFS, we talk about huge data sets, i.e. Terabytes and Petabytes of data. So, if we had a block size of let's say of 4 KB, as in Linux file system, we would be having too many blocks and therefore too much of the metadata. So, managing these no. of blocks and metadata will create huge overhead, which is something, we don't want.

*As you understood **what a block is**, let us understand how the replication of these blocks takes place in the next section of this HDFS Architecture. Meanwhile, you may check out this video tutorial on HDFS Architecture where all the HDFS Architecture concepts has been discussed in detail:*

Replication Management:

HDFS provides a reliable way to store huge data in a distributed environment as data blocks. The blocks are also replicated to provide fault tolerance. The default replication factor is 3 which is again configurable. So, as you can see in the figure below where each block is replicated three times and stored on different DataNodes (considering the default replication

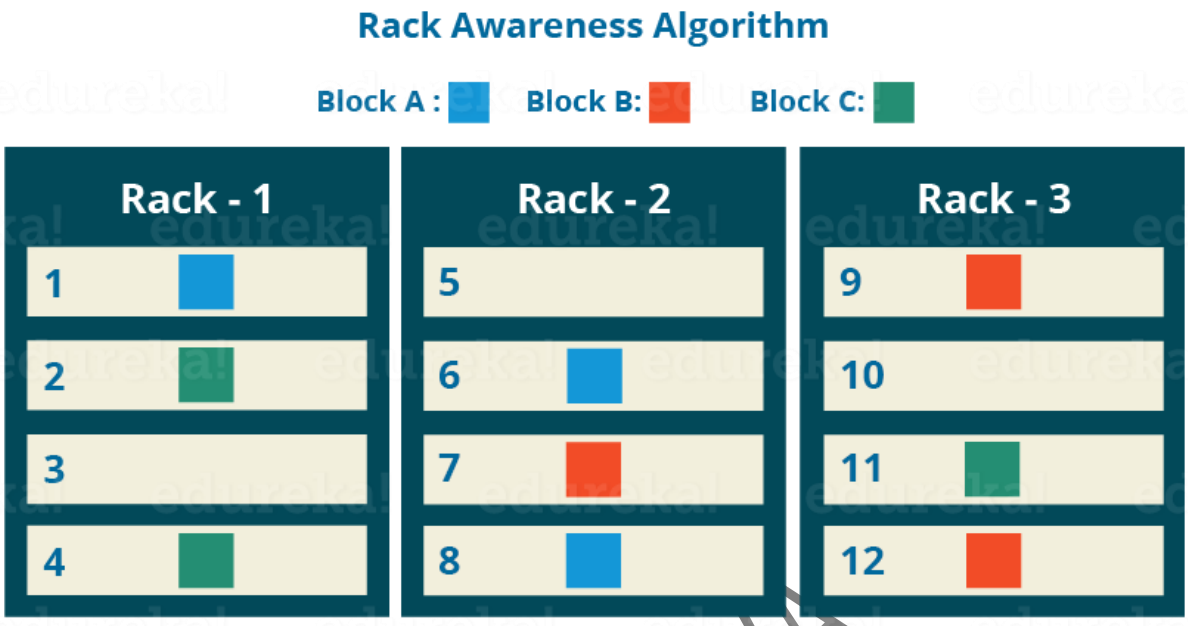
factor):



Therefore, if you are storing a file of 128 MB in HDFS using the default configuration, you will end up occupying a space of 384 MB (3*128 MB) as the blocks will be replicated three times and each replica will be residing on a different DataNode.

Note: The NameNode collects block report from DataNode periodically to maintain the replication factor. Therefore, whenever a block is over-replicated or under-replicated the NameNode deletes or add replicas as needed.

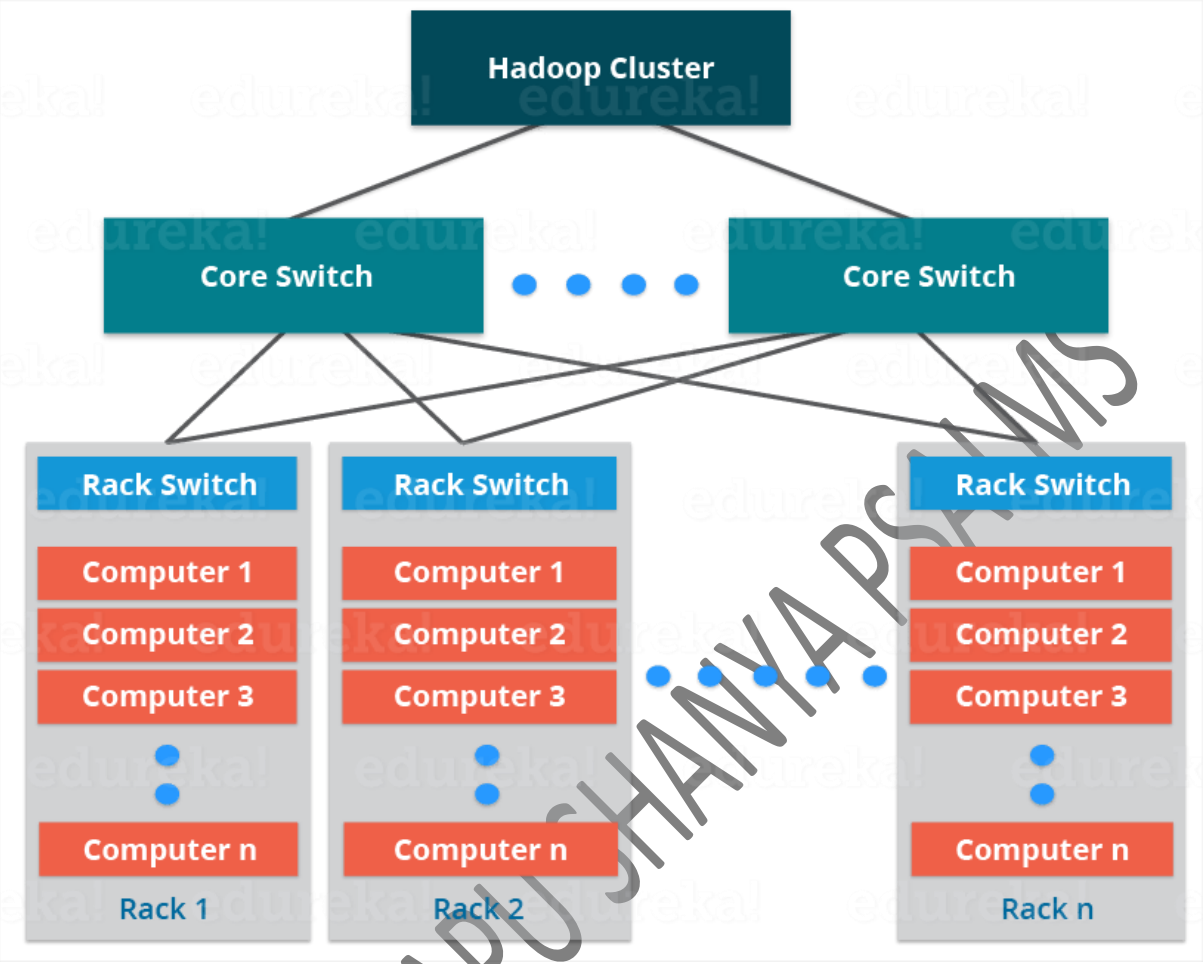
Rack Awareness:



Anyways, moving ahead, let's talk more about how HDFS places replica and what is rack awareness? Again, the NameNode also ensures that all the replicas are not stored on the same rack or a single rack. It follows an in-built Rack Awareness Algorithm to reduce latency as well as provide fault tolerance.

Considering the replication factor is 3, the Rack Awareness Algorithm says that the first replica of a block will be stored on a local rack and the next two replicas will be stored on a different (remote) rack but, on a different DataNode within that (remote) rack as shown in the figure above.

This is how an actual Hadoop production cluster looks like. Here, you have multiple racks populated with DataNodes:



Advantages of Rack Awareness:

So, now you will be thinking why do we need a Rack Awareness algorithm? The reasons are:

- **To improve the network performance:** The communication between nodes residing on different racks is directed via switch. In general, you will find *greater network bandwidth* between machines in the same rack than the machines residing in different rack. So, the Rack Awareness helps you to have reduce write traffic in between different racks and thus providing a better write performance. Also, you will be gaining increased read performance because you are using the bandwidth of multiple racks.

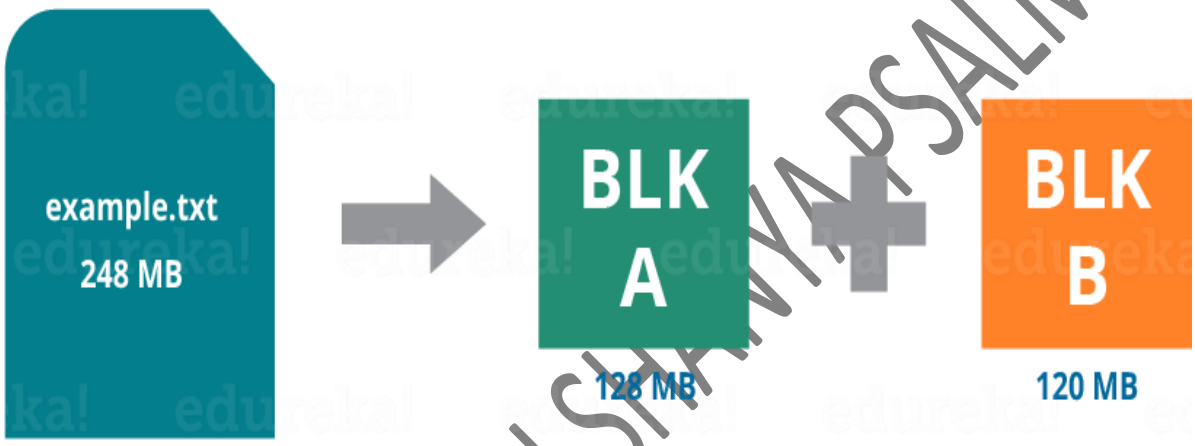
- **To prevent loss of data:** We don't have to worry about the data even if an entire rack fails because of the switch failure or power failure. And if you think about it, it will make sense, as it is said that *never put all your eggs in the same basket.*

HDFS Read/ Write Architecture:

Now let's talk about how the data read/write operations are performed on HDFS. HDFS follows Write Once – Read Many Philosophy. So, you can't edit files already stored in HDFS. But, you can append new data by re-opening the file. Get a better understanding of the Hadoop Clusters, nodes, and architecture from the Hadoop Admin Training in Chennai.

HDFS Write Architecture:

Suppose a situation where an HDFS client, wants to write a file named "example.txt" of size 248 MB.

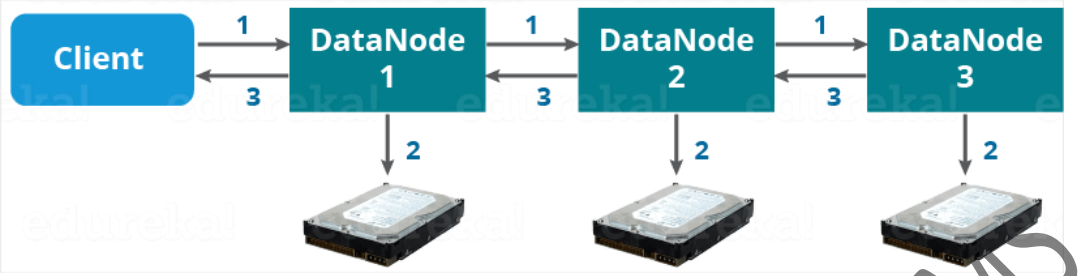


Assume that the system block size is configured for 128 MB (default). So, the client will be dividing the file "example.txt" into 2 blocks – one of 128 MB (Block A) and the other of 120 MB (block B).

Now, the following protocol will be followed whenever the data is written into HDFS:

- At first, the HDFS client will reach out to the NameNode for a Write Request against the two blocks, say, Block A & Block B.
- The NameNode will then grant the client the write permission and will provide the IP addresses of the DataNodes where the file blocks will be copied eventually.
- The selection of IP addresses of DataNodes is purely randomized based on availability, replication factor and rack awareness that we have discussed earlier.
- Let's say the replication factor is set to default i.e. 3. Therefore, for each block the NameNode will be providing the client a list of (3) IP addresses of DataNodes. The list will be unique for each block.
- Suppose, the NameNode provided following lists of IP addresses to the client:
    - For Block A, list A = {IP of DataNode 1, IP of DataNode 4, IP of DataNode 6}
    - For Block B, set B = {IP of DataNode 3, IP of DataNode 7, IP of DataNode 9}

- Each block will be copied in three different DataNodes to maintain the replication factor consistent throughout the cluster.
- Now the whole data copy process will happen in three stages:

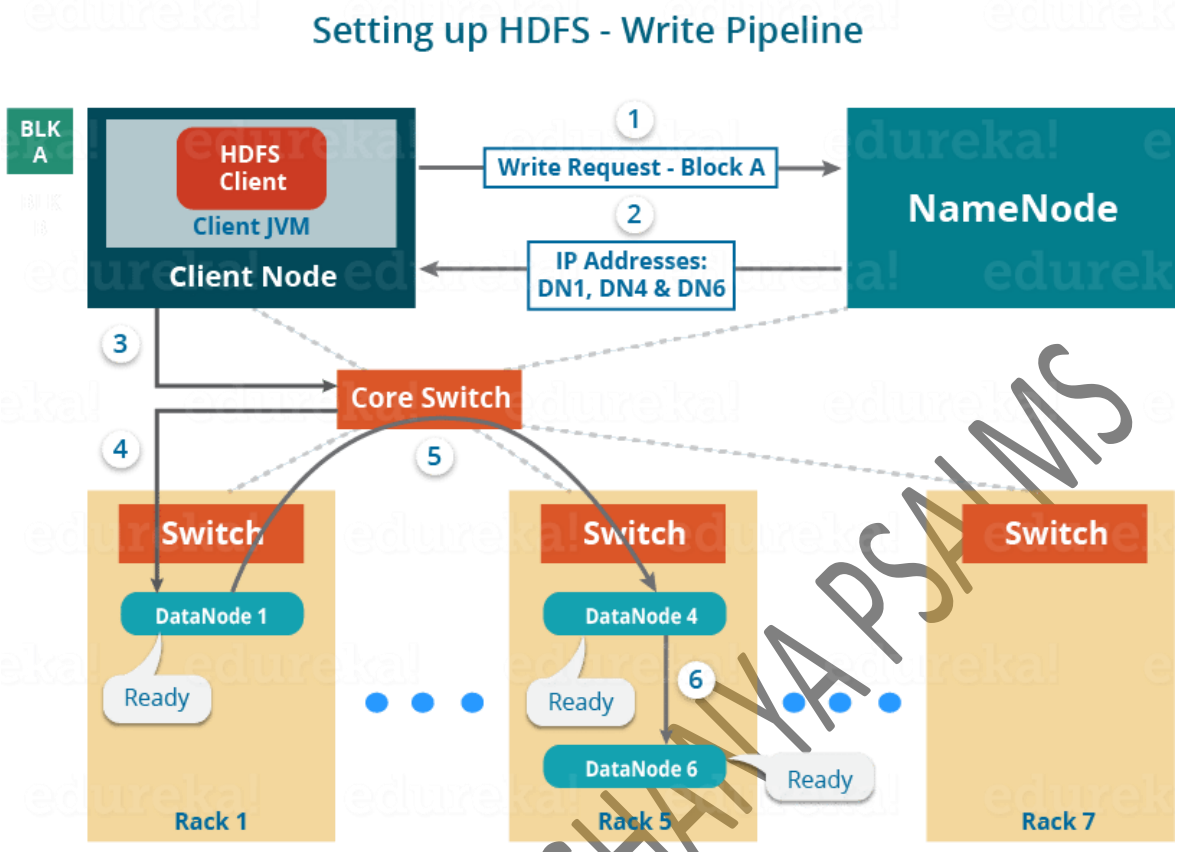


1. Set up of Pipeline
2. Data streaming and replication
3. Shutdown of Pipeline (Acknowledgement stage)

1. Set up of Pipeline:

Before writing the blocks, the client confirms whether the DataNodes, present in each of the list of IPs, are ready to receive the data or not. In doing so, the client creates a pipeline for each of the blocks by connecting the individual DataNodes in the respective list for that block. Let us consider Block A. The list of DataNodes provided by the NameNode is:

For Block A, list A = {IP of DataNode 1, IP of DataNode 4, IP of DataNode 6}.



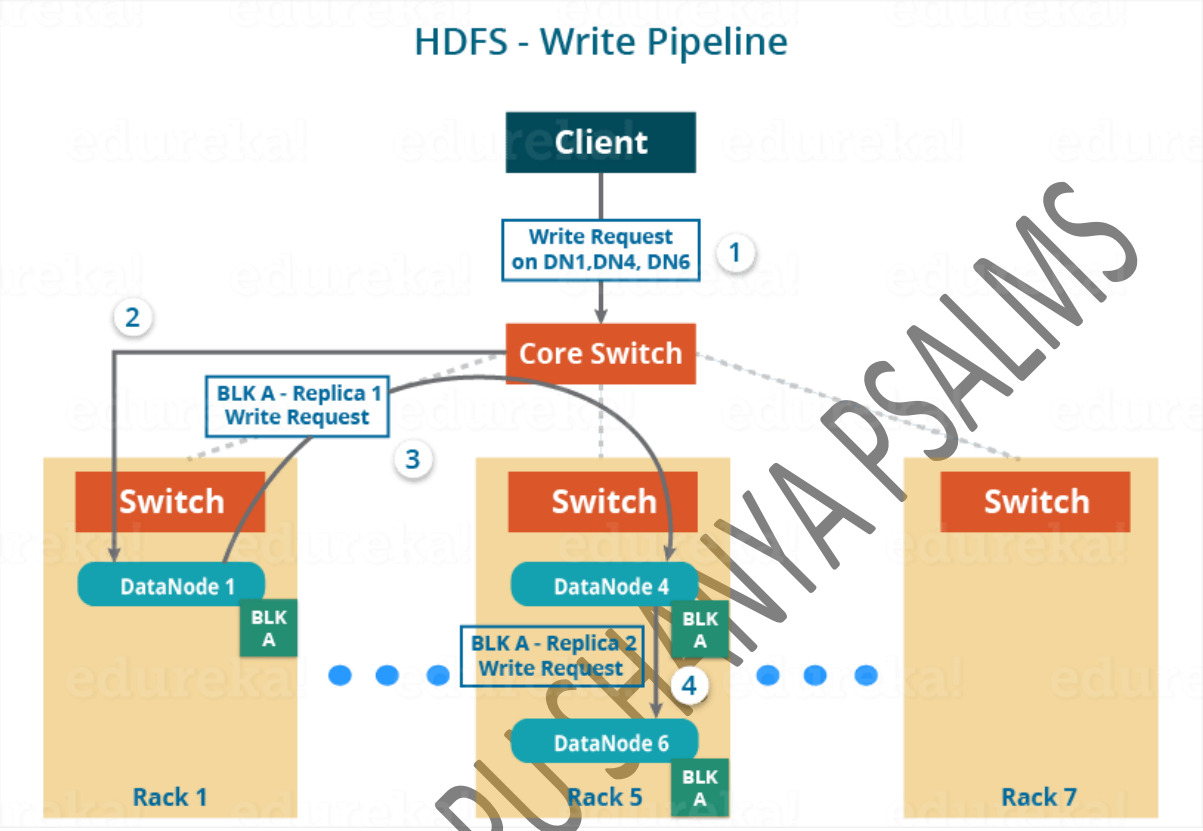
So, for block A, the client will be performing the following steps to create a pipeline:

- The client will choose the first Data Node in the list (Data Node IPs for Block A) which is Data Node 1 and will establish a TCP/IP connection.
- The client will inform Data Node 1 to be ready to receive the block. It will also provide the IPs of next two Data Nodes (4 and 6) to the Data Node 1 where the block is supposed to be replicated.
- The Data Node 1 will connect to Data Node 4. The DataNode 1 will inform Data Node 4 to be ready to receive the block and will give it the IP of DataNode 6. Then, Data Node 4 will tell Data Node 6 to be ready for receiving the data.
- Next, the acknowledgement of readiness will follow the reverse sequence, i.e. From the DataNode 6 to 4 and then to 1.
- At last DataNode 1 will inform the client that all the DataNodes are ready and a pipeline will be formed between the client, DataNode 1, 4 and 6.
- Now pipeline set up is complete and the client will finally begin the data copy or streaming process.

2. Data Streaming:

As the pipeline has been created, the client will push the data into the pipeline. Now, don't forget that in HDFS, data is replicated based on replication factor. So,

here Block A will be stored to three DataNodes as the assumed replication factor is 3. Moving ahead, the client will copy the block (A) to DataNode 1 only. The replication is always done by DataNodes sequentially.



So, the following steps will take place during replication:

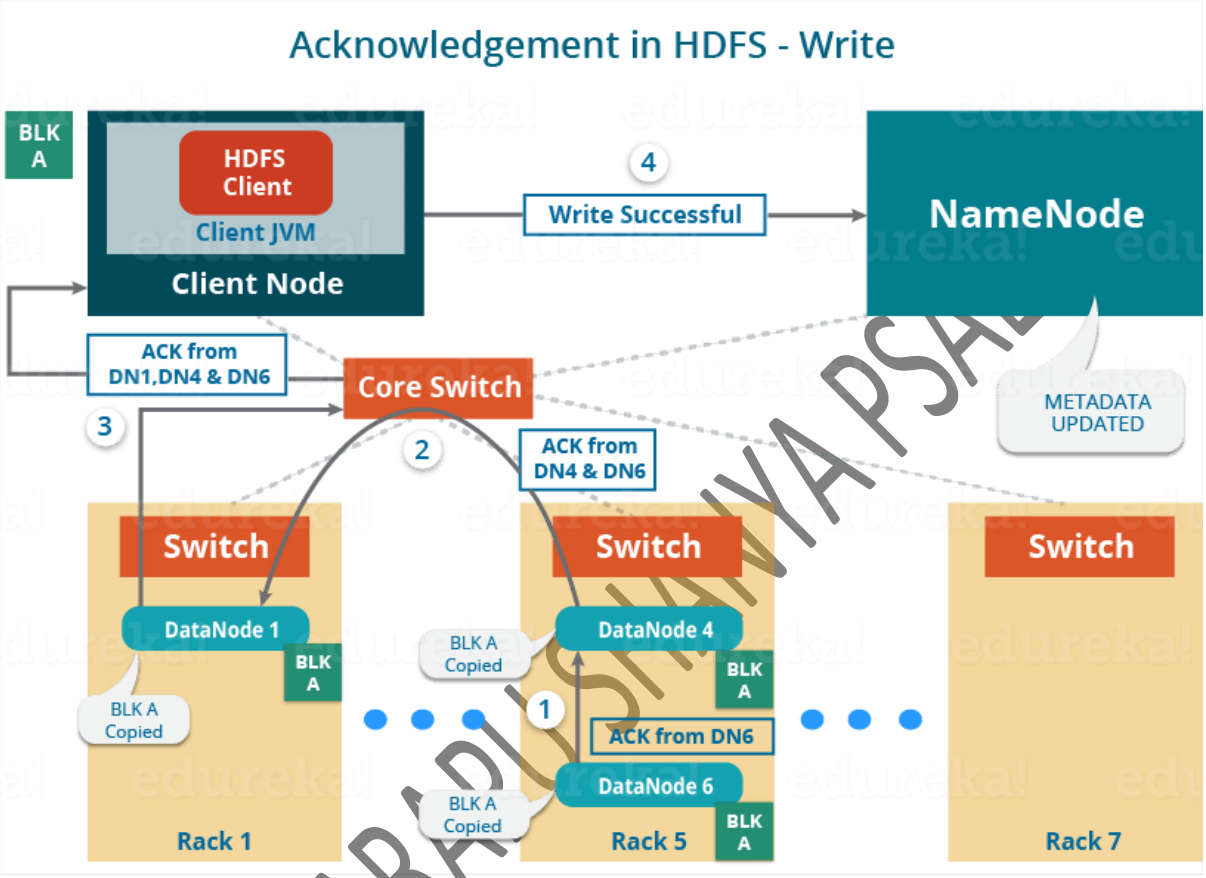
- Once the block has been written to DataNode 1 by the client, DataNode 1 will connect to DataNode 4.
- Then, DataNode 1 will push the block in the pipeline and data will be copied to DataNode 4.
- Again, DataNode 4 will connect to DataNode 6 and will copy the last replica of the block.

3. Shutdown of Pipeline or Acknowledgement stage:

Once the block has been copied into all the three DataNodes, a series of acknowledgements will take place to ensure the client and NameNode that the data has been written successfully. Then, the client will finally close the pipeline to end the TCP session.

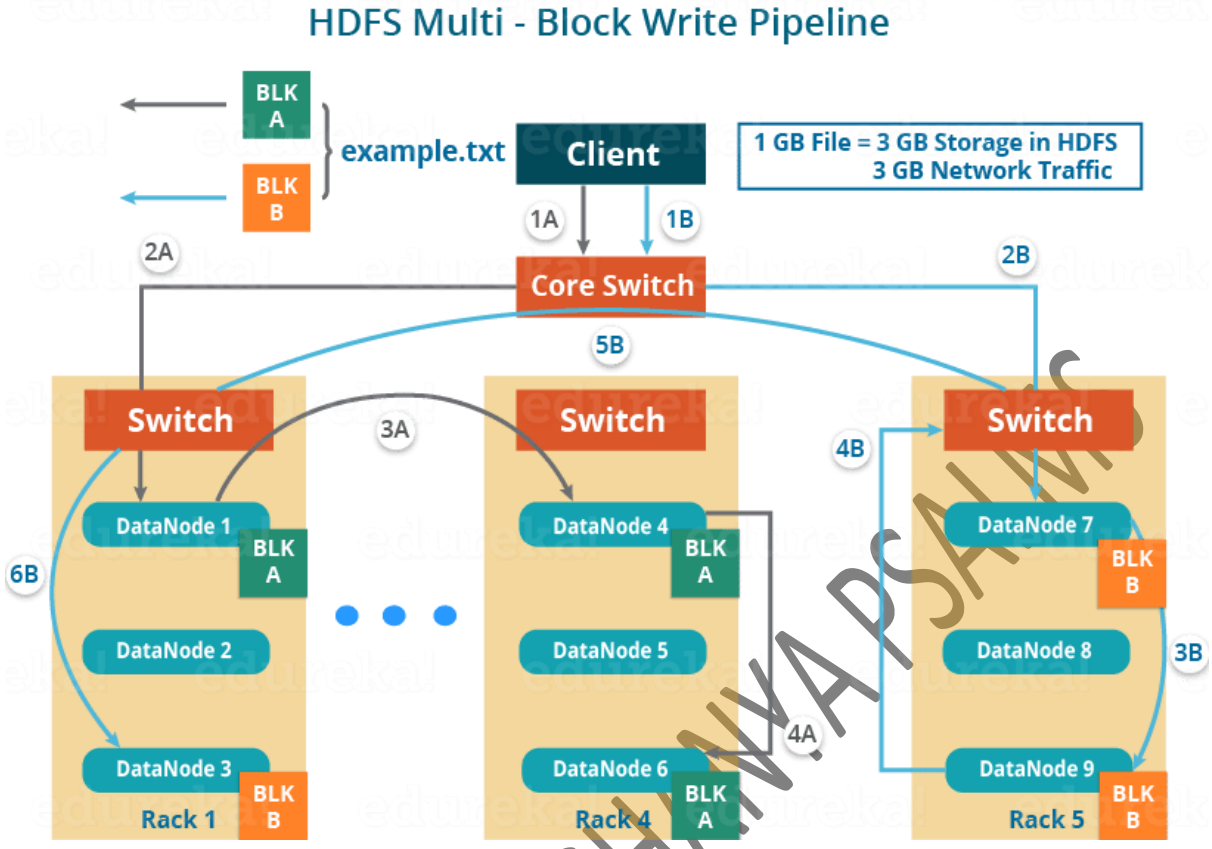
As shown in the figure below, the acknowledgement happens in the reverse sequence i.e. from DataNode 6 to 4 and then to 1. Finally, the DataNode 1 will

push three acknowledgements (including its own) into the pipeline and send it to the client. The client will inform NameNode that data has been written successfully. The NameNode will update its metadata and the client will shut down the pipeline.



Similarly, Block B will also be copied into the DataNodes in parallel with Block A. So, the following things are to be noticed here:

- The client will copy Block A and Block B to the first DataNode **simultaneously**.
- Therefore, in our case, two pipelines will be formed for each of the block and all the process discussed above will happen in parallel in these two pipelines.
- The client writes the block into the first DataNode and then the DataNodes will be replicating the block sequentially.

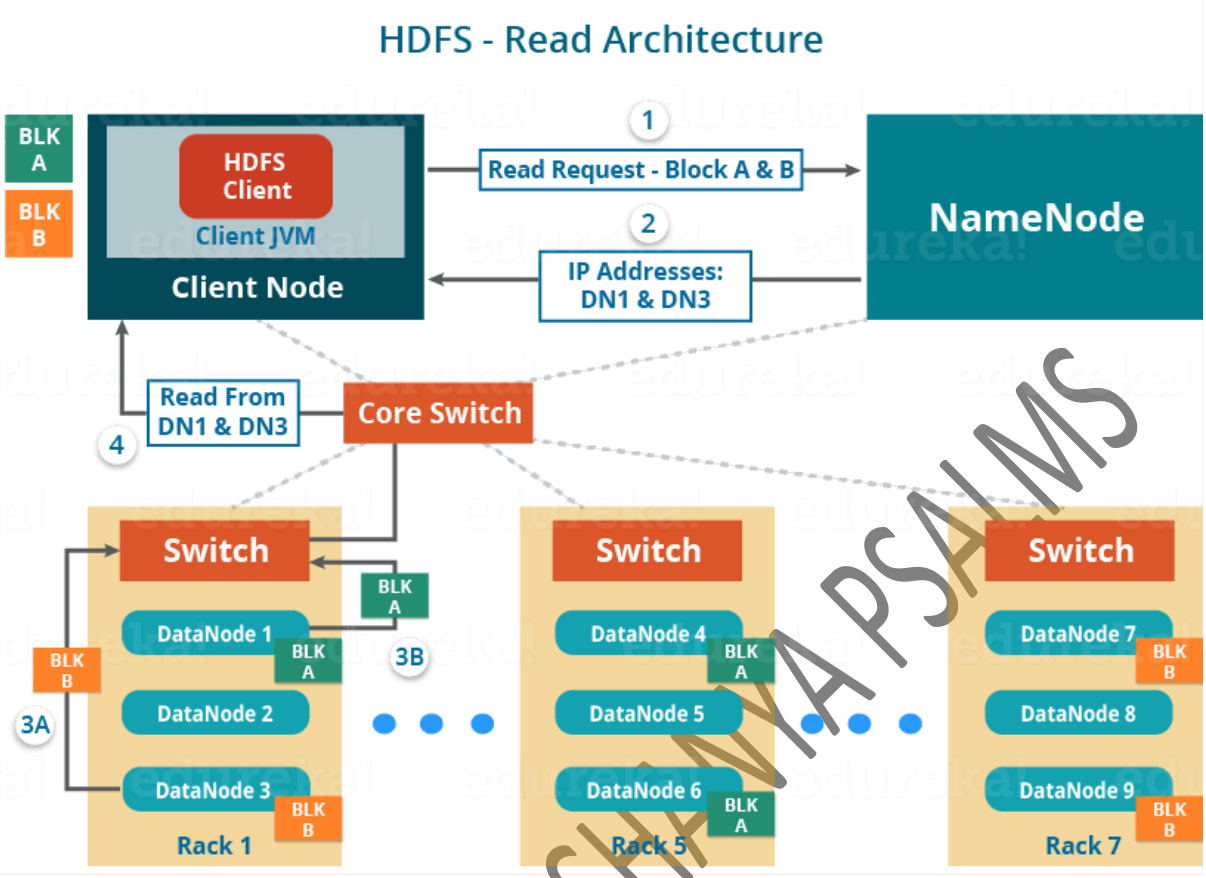


As you can see in the above image, there are two pipelines formed for each block (A and B). Following is the flow of operations that is taking place for each block in their respective pipelines:

- For Block A: 1A -> 2A -> 3A -> 4A
- For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

HDFS Read Architecture:

HDFS Read architecture is comparatively easy to understand. Let's take the above example again where the HDFS client wants to read the file "example.txt" now.



Now, following steps will be taking place while reading the file:

- The client will reach out to NameNode asking for the block metadata for the file "example.txt".
- The NameNode will return the list of DataNodes where each block (Block A and B) are stored.
- After that client, will connect to the DataNodes where the blocks are stored.
- The client starts reading data parallel from the DataNodes (Block A from DataNode 1 and Block B from DataNode 3).
- Once the client gets all the required file blocks, it will combine these blocks to form a file.

While serving read request of the client, HDFS selects the replica which is closest to the client. This reduces the read latency and the bandwidth consumption. Therefore, that replica is selected which resides on the same rack as the reader node, if possible.

Now, you should have a pretty good idea about Apache Hadoop HDFS Architecture. I understand that there is a lot of information here and it may not be easy to get it in one go. I would suggest you to go through it again and I am

sure you will find it easier this time. Now, in my next blog, I will be talking about Apache Hadoop HDFS Federation and High Availability Architecture.

Hadoop Configuration

It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce. The hdfs-site. xml file contains the configuration settings for HDFS daemons; the Name Node, the Secondary Name Node, and the Data Nodes. Here, we can configure hdfs-site.

Slaves & Masters:
Slaves contain a list of hosts, one per line, that are needed to host DataNode and TaskTracker servers. The Masters contain a list of hosts, one per line, that are required to host secondary NameNode servers. The Masters file informs about the Secondary NameNode location to Hadoop daemon. The 'Masters' file at Master server contains a hostname, Secondary Name Node servers.

The Hadoop-env.sh, core-ite.xml, hdfs-site.xml, mapred-site.xml, Masters and Slaves are all available under 'conf" directory of Hadoop installation directory.

Core-site.xml and hdfs-site.xml:
The core-site.xml file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.

The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes. Here, we can configure hdfs-site.xml to specify default block replication and permission checking on HDFS. The actual number of replications can also be specified when the file is created. The default is used if replication is not specified in create time.

Defining HDFS Details in hdfs-site.xml:

Property	Value	Description
dfs.data.dir	<value> /disk1/hdfs/data, /disk2/hdfs/data </value>	A list of directories where the datanode stores blocks. Each block is stored in only one of these directories. ${hadoop.tmp.dir}/dfs/data
fs.checkpoint.dir	<value> /disk1/hdfs/namesecondary, /disk2/hdfs/namesecondary </value>	A list of directories where the secondary namenode stores checkpoints. It stores a copy of the checkpoint in each directory in the list ${hadoop.tmp.dir}/dfs/namesecondary

Mapred-site.xml:

mapred-site.xml

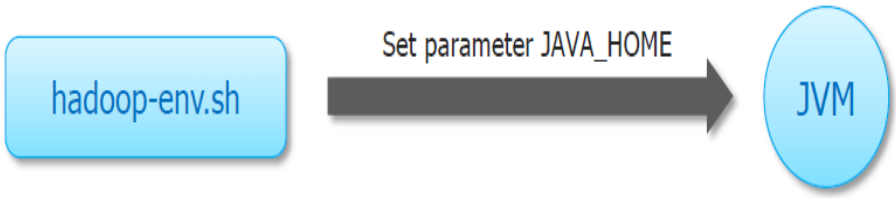
```
<?xml version="1.0"?>
<configuration>
<property>
        <name>mapred.job.tracker</name>
        <value>localhost:8021</value>
<property>
</configuration>
```

The mapred-site.xml file contains the configuration settings for MapReduce daemons; the job tracker and the task-trackers.

Defining mapred-site.xml:

Property	Value	Description
mapred.job.tracker	<value> localhost:8021 </value>	The hostname and the port that the jobtracker RPC server runs on. If set to the default value of local, then the jobtracker runs in-process on demand when you run a MapReduce job.
mapred.local.dir	${hadoop.tmp.dir}/mapred/local	A list of directories where MapReduce stores intermediate data for jobs. The data is cleared out when the job ends.
mapred.system.dir	${hadoop.tmp.dir}/mapred/system	The directory relative to fs.default.name where shared files are stored, during a job run.
mapred.tasktracker.map.tasks.maximum	2	The number of map tasks that may be run on a tasktracker at any one time
mapred.tasktracker.reduce.tasks.maximum	2	The number of reduce tasks tat may be run on a tasktracker at any one time.

Per-Proccess Run Time Environment:



This file offers a way to provide customer parameters for each of the servers. Hadoop-env.sh is sourced by the entire Hadoop core scripts provided in the 'conf/' directory of the installation.

Here are some examples of environment variables than can be specified:

exportHADOOP_DATANODE_HEAPSIZE="128"

exportHADOOP_TASKTRACKER_HEAPSIZE="512"

The 'hadoop-metrics.properties' file controls the reporting and the default condition is set as not to report.

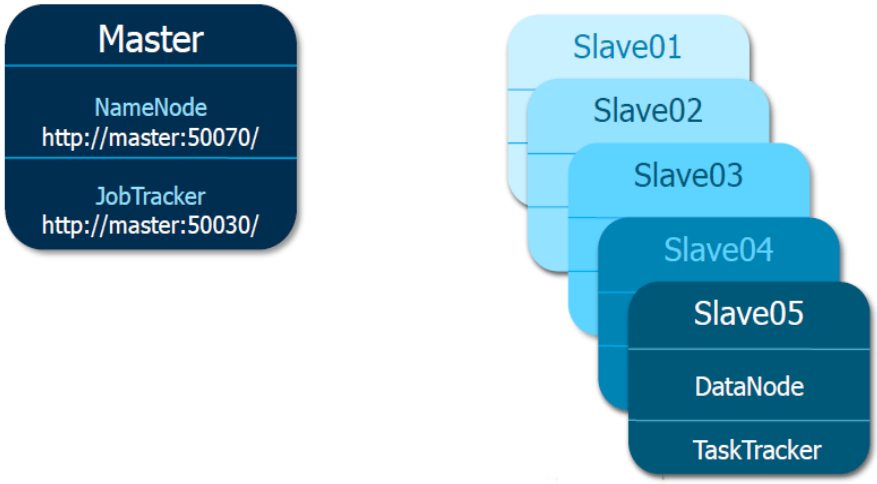
Hadoop Cluster in Facebook:

Facebook uses Hadoop to store copies of internal log and dimension data sources and use it as a source for reporting, analytics and machine learning. Currently, Facebook has two major clusters: A 1100-machine cluster with 800

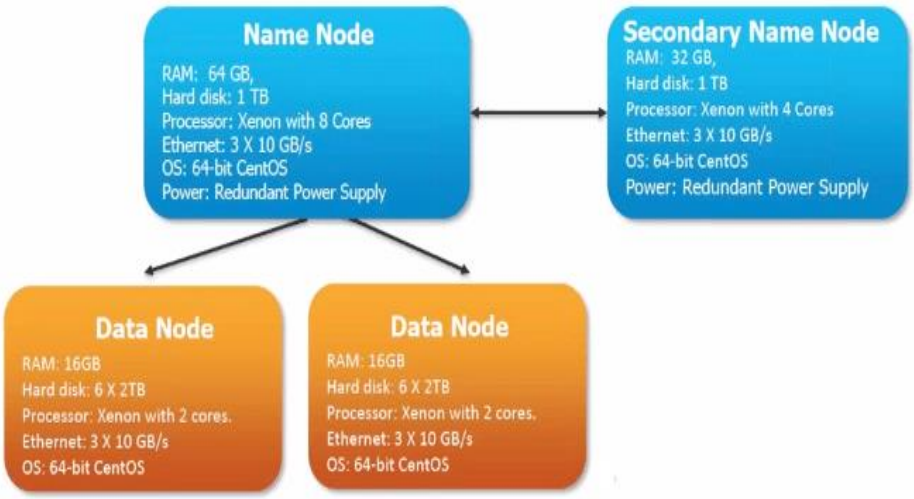
cores and about 12 PB raw storage. Another one is a 300 machine cluster with 2,400 cores and about 3 PB raw storage. Each of the commodity node has 8 cores and 12 TB storage.

Facebook uses streaming and Java API a lot and have used Hive to build a higher-level data warehousing framework. They have also developed a FUSE application over HDFS.

Sample Cluster Configuration:



Hadoop Cluster – A Typical Use Case:



The above image clearly explains the configuration of each nodes. NameNode has high memory requirement and will have a lot of RAM and does

not require a lot of memory on hard disk. The memory requirement for a secondary NameNode is not as high as the primary NameNode. Each DataNode requires 16 GB of memory and are high on hard disk as they are supposed to store data. They have multiple drives as well. Learn more from this Big Data Course about Hadoop Clusters, HDFS, and other important topics to become a Hadoop professional.

Map Reduce Framework:

MapReduce is a software framework and programming model used for processing huge amounts of data. **MapReduce** program work in two phases, namely, Map and Reduce. Map tasks deal with splitting and mapping of data while Reduce tasks shuffle and reduce the data.
Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. The programs of Map Reduce in cloud computing are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster.

The input to each phase is **key-value** pairs. In addition, every programmer needs to specify two functions: **map function** and **reduce function**.

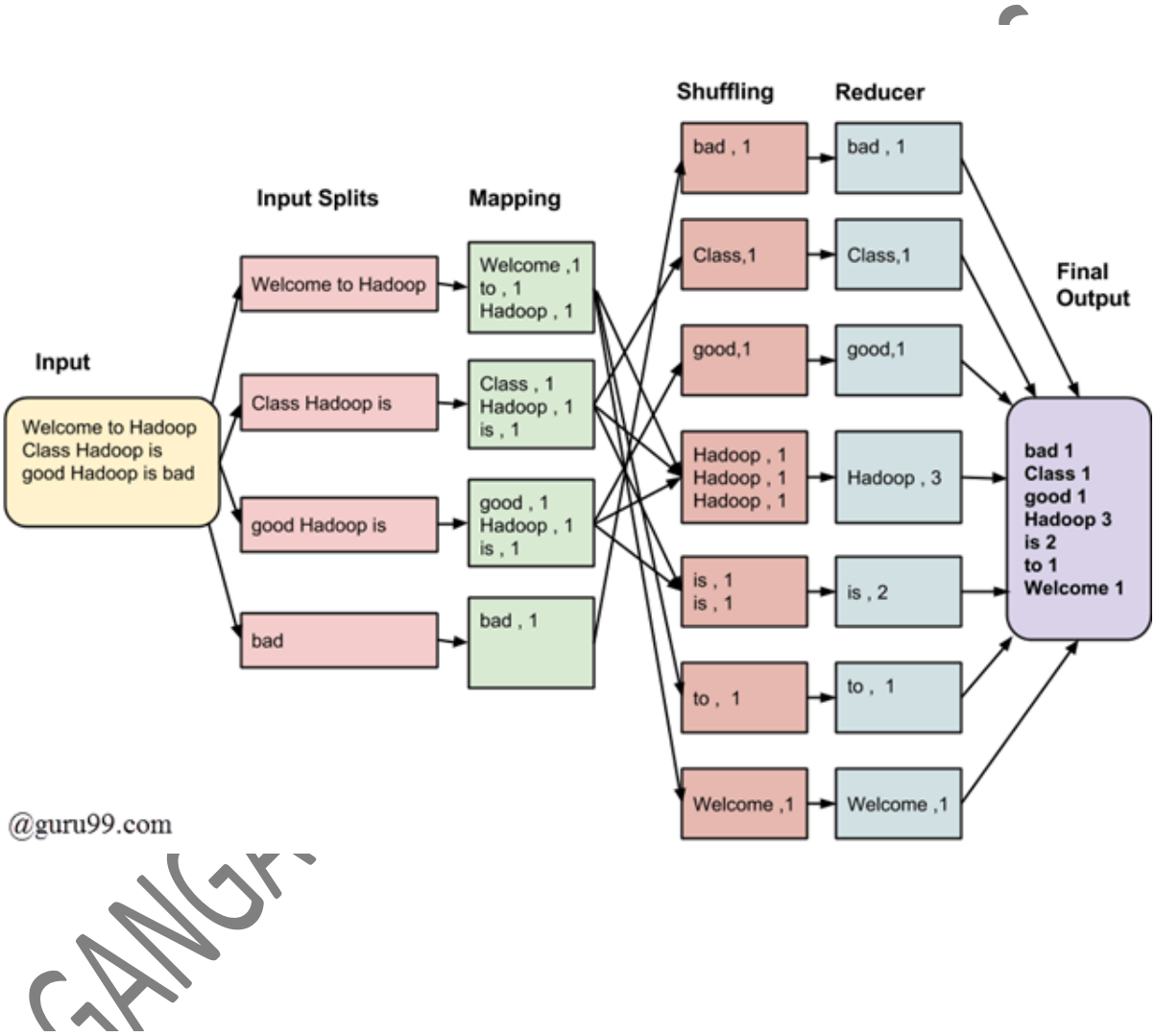
MapReduce Architecture in Big Data explained with Example

The whole process goes through four phases of execution namely, splitting, mapping, shuffling, and reducing.

Now in this MapReduce tutorial, let's understand with a MapReduce example–

Consider you have following input data for your MapReduce in Big data Program

```
Welcome to Hadoop Class
Hadoop is good
Hadoop is bad
```



The final output of the MapReduce task is

```
bad
Class
good
```

Hadoop

is

to

Welcome

The data goes through the following phases of MapReduce in Big Data

Input Splits:

An input to a MapReduce in Big Data job is divided into fixed-size pieces called **input splits** Input split is a chunk of the input that is consumed by a single map

Mapping

This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits (more details about input-split is given below) and prepare a list in the form of <word, frequency>

Shuffling

This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubed together along with their respective frequency.

Reducing

In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.

In our example, this phase aggregates the values from Shuffling phase i.e., calculates total occurrences of each word.

MapReduce Architecture

- One map task is created for each split which then executes map function for each record in the split.
- It is always beneficial to have multiple splits because the time taken to process a split is small as compared to the time taken for processing of the whole input. When the splits are smaller, the processing is better to load balanced since we are processing the splits in parallel.
- However, it is also not desirable to have splits too small in size. When splits are too small, the overload of managing the splits and map task creation begins to dominate the total job execution time.
- For most jobs, it is better to make a split size equal to the size of an HDFS block (which is 64 MB, by default).
- Execution of map tasks results into writing output to a local disk on the respective node and not to HDFS.
- Reason for choosing local disk over HDFS is, to avoid replication which takes place in case of HDFS store operation.
- Map output is intermediate output which is processed by reduce tasks to produce the final output.
- Once the job is complete, the map output can be thrown away. So, storing it in HDFS with replication becomes overkill.
- In the event of node failure, before the map output is consumed by the reduce task, Hadoop reruns the map task on another node and re-creates the map output.
- Reduce task doesn't work on the concept of data locality. An output of every map task is fed to the reduce task. Map output is transferred to the machine where reduce task is running.
- On this machine, the output is merged and then passed to the user-defined reduce function.

- Unlike the map output, reduce output is stored in HDFS (the first replica is stored on the local node and other replicas are stored on off-rack nodes). So, writing the reduce output

How MapReduce Organizes Work?

Now in this MapReduce tutorial, we will learn how MapReduce works

Hadoop divides the job into tasks. There are two types of tasks:

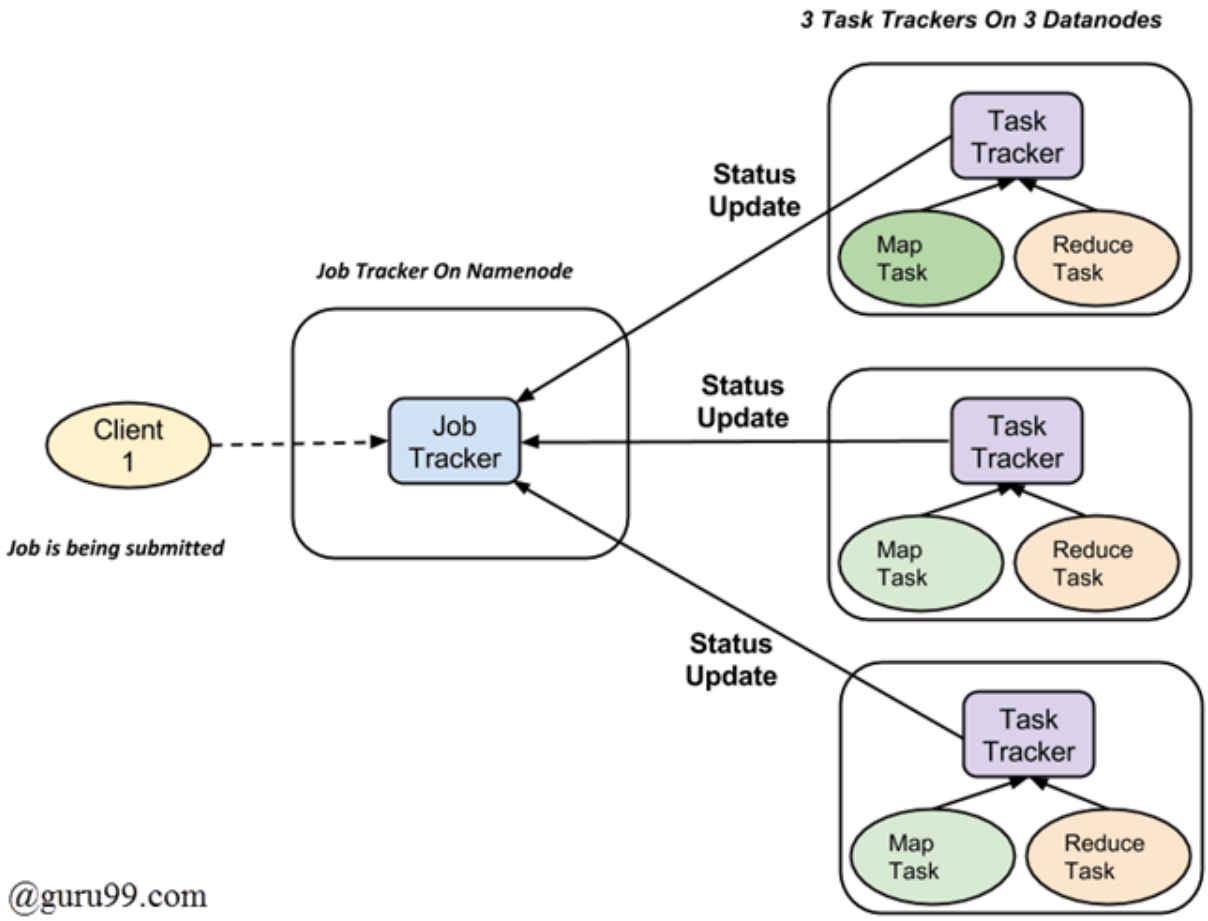
1. **Map tasks** (Splits & Mapping)
2. **Reduce tasks** (Shuffling, Reducing)

as mentioned above.

The complete execution process (execution of Map and Reduce tasks, both) is controlled by two types of entities called a

1. **Jobtracker**: Acts like a **master** (responsible for complete execution of submitted job)
2. **Multiple Task Trackers**: Acts like **slaves,** each of them performing the job

For every job submitted for execution in the system, there is one **Jobtracker** that resides on **Namenode** and there are **multiple tasktrackers** which reside on **Datanode**.



How Hadoop MapReduce Works

- A job is divided into multiple tasks which are then run onto multiple data nodes in a cluster.
- It is the responsibility of job tracker to coordinate the activity by scheduling tasks to run on different data nodes.
- Execution of individual task is then to look after by task tracker, which resides on every data node executing part of the job.
- Task tracker's responsibility is to send the progress report to the job tracker.
- In addition, task tracker periodically sends **'heartbeat'** signal to the Jobtracker so as to notify him of the current state of the system.
- Thus job tracker keeps track of the overall progress of each job. In the event of task failure, the job tracker can reschedule it on a different task tracker.

Role of Hbase in Big Data Processing:

HBase provides low latency random read and write access to petabytes of data by distributing requests from applications across a cluster of hosts. Each host has access to data in HDFS and S3, and serves read and write requests in milliseconds.

Since 1970, RDBMS is the solution for data storage and maintenance related problems. After the advent of big data, companies realized the benefit of processing big data and started opting for solutions like Hadoop.

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.

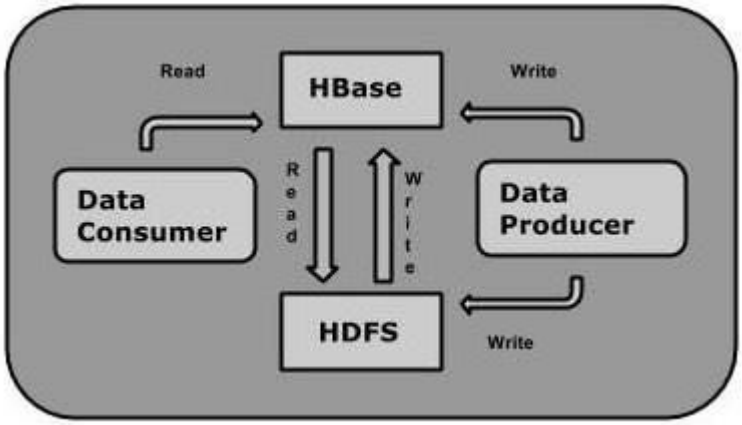
What is HBase?

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.

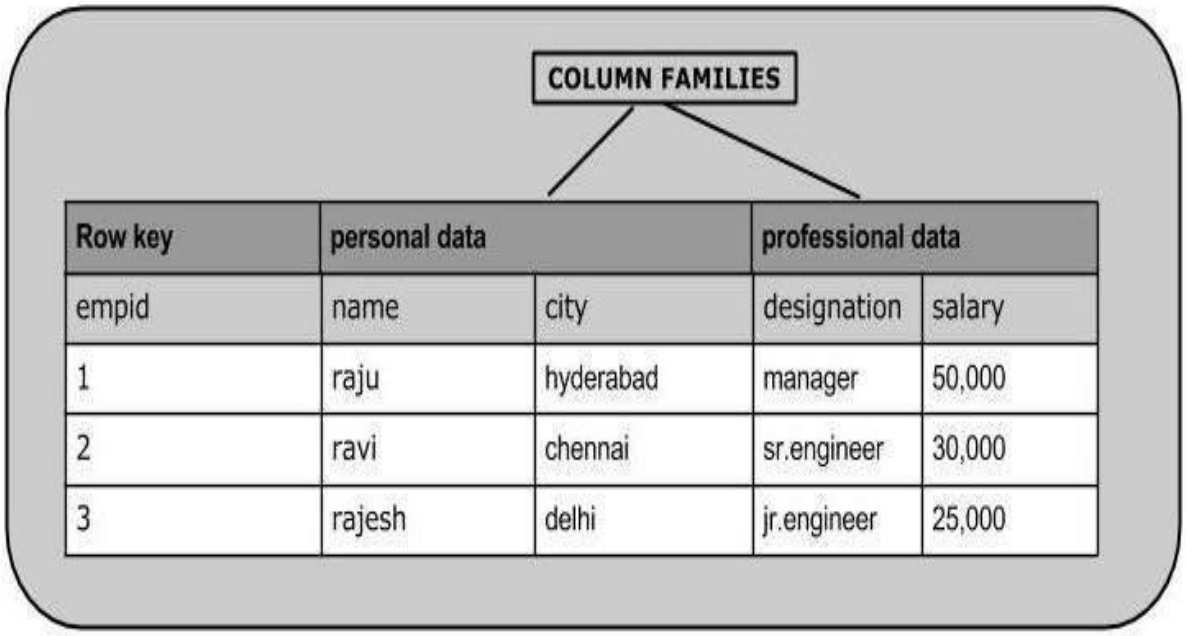
Rowid	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3	col1	col2	col3
1												
2												
3												

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:



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

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.

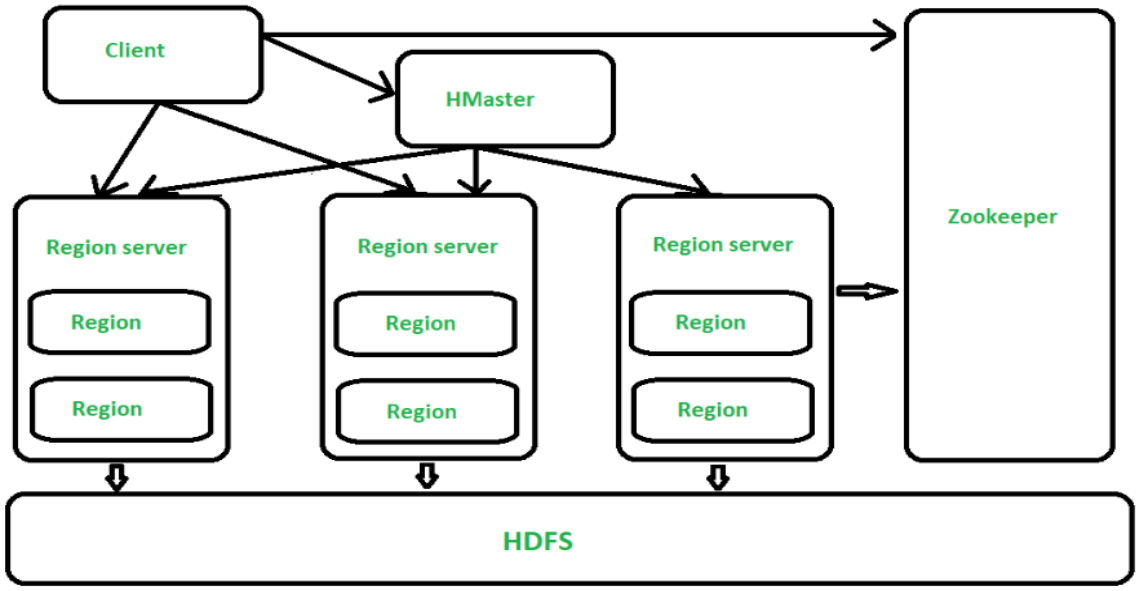
Applications of HBase

- 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 - Architecture

In HBase, tables are split into regions and are served by the region servers. Regions are vertically divided by column families into “Stores”. Stores are saved as files in HDFS. Shown below is the architecture of HBase.

Note: The term ‘store’ is used for regions to explain the storage structure.



HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement.

MasterServer

The master server -

- Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is responsible for schema changes and other metadata operations such as creation of tables and column families.

Regions

Regions are nothing but tables that are split up and spread across the region servers.

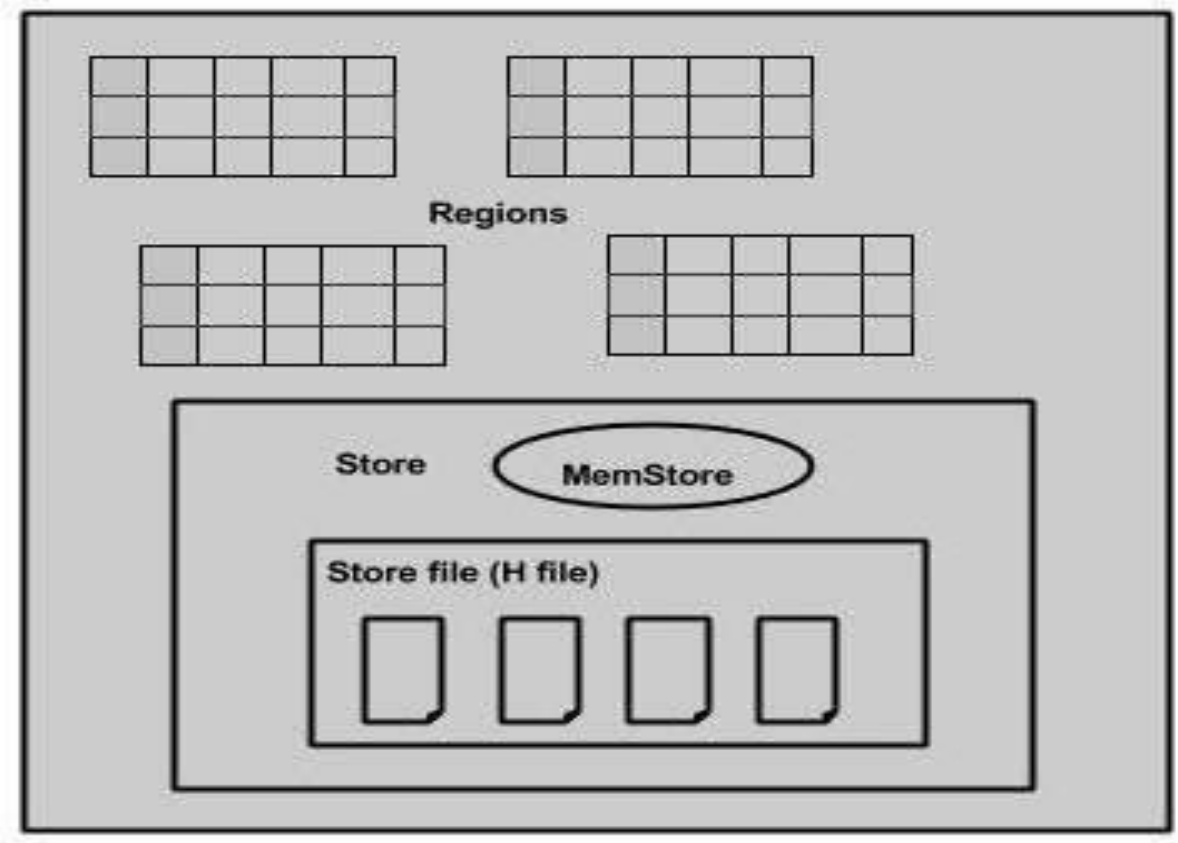
Region server

The region servers have regions that -

- Communicate with the client and handle data-related operations.

- Handle read and write requests for all the regions under it.
- Decide the size of the region by following the region size thresholds.

When we take a deeper look into the region server, it contain regions and stores as shown below:



The store contains memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.

Zookeeper

- Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
- Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.
- In addition to availability, the nodes are also used to track server failures or network partitions.
- Clients communicate with region servers via zookeeper.

- In pseudo and standalone modes, HBase itself will take care of zookeeper.

Hive

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

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

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

Features of Hive

These are the following features of Hive:

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

Limitations of Hive

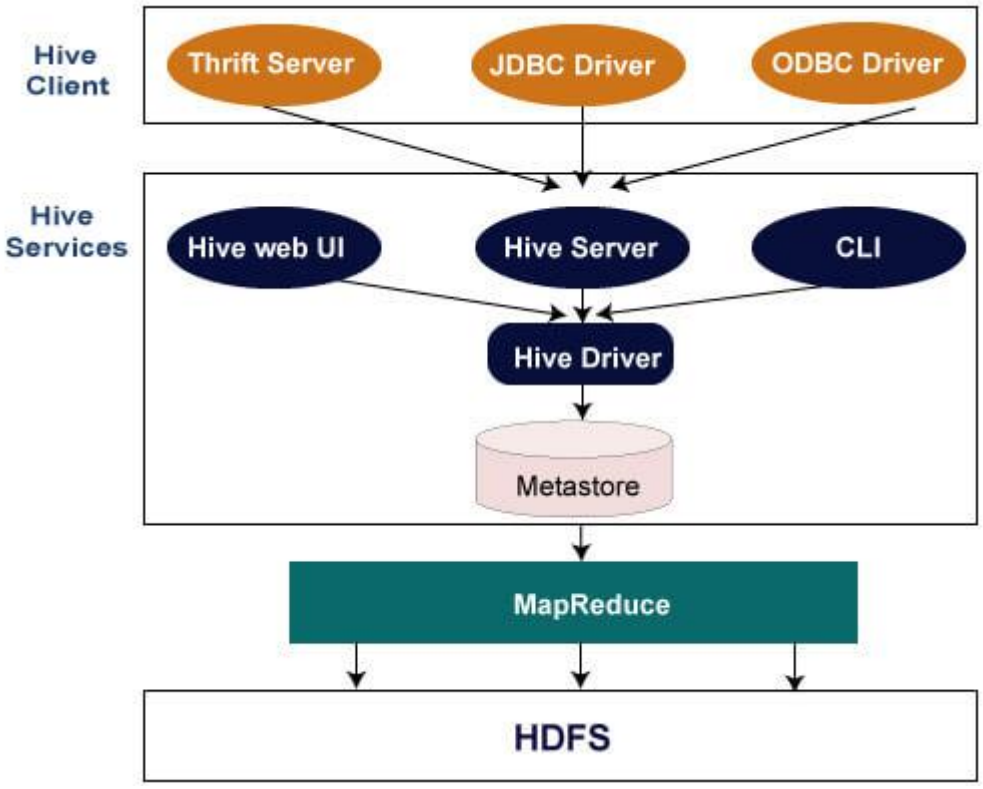
- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Differences between Hive and Pig

Hive	Pig
Hive is commonly used by Data Analysts.	Pig is commonly used by programmers.
It follows SQL-like queries.	It follows the data-flow language.
It can handle structured data.	It can handle semi-structured data.
It works on server-side of HDFS cluster.	It works on client-side of HDFS cluster.
Hive is slower than Pig.	Pig is comparatively faster than Hive.

Hive Architecture

The following architecture explains the flow of submission of query into Hive.



Hive Client

Hive allows writing applications in various languages, including Java, Python, and C++. It supports different types of clients such as:-

- Thrift Server - It is a cross-language service provider platform that serves the request from all those programming languages that supports Thrift.
- JDBC Driver - It is used to establish a connection between hive and Java applications. The JDBC Driver is present in the class org.apache.hadoop.hive.jdbc.HiveDriver.
- ODBC Driver - It allows the applications that support the ODBC protocol to connect to Hive.

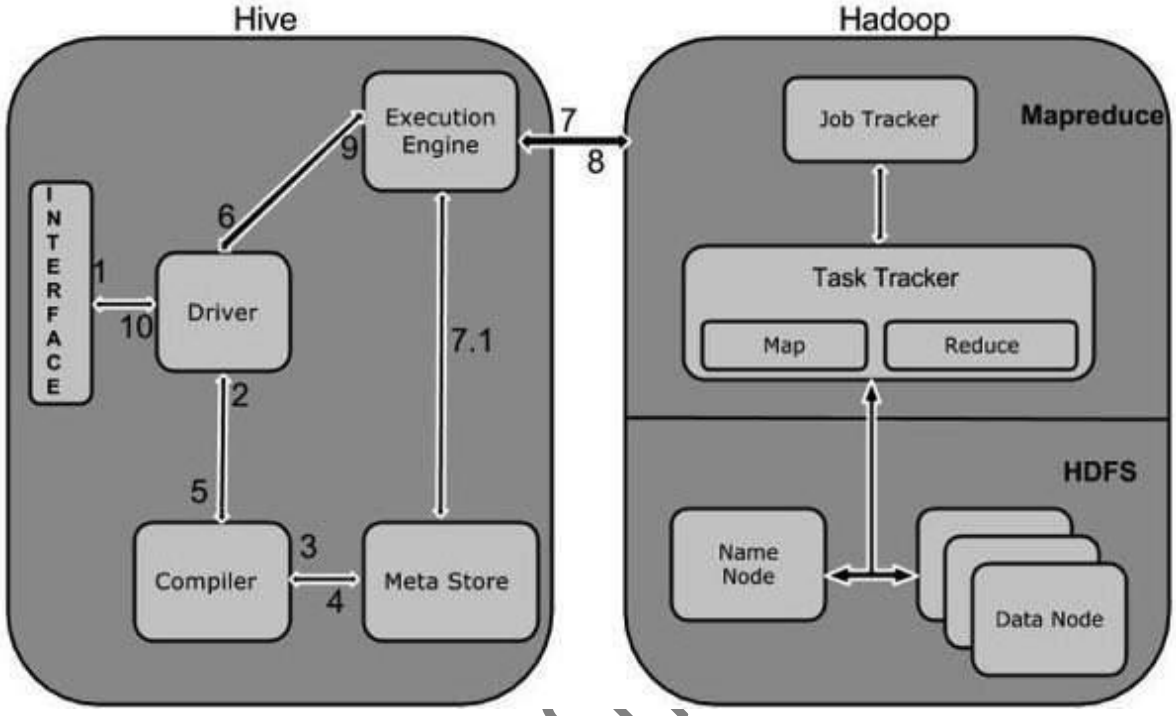
Hive Services

The following are the services provided by Hive:-

- Hive CLI - The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands.
- Hive Web User Interface - The Hive Web UI is just an alternative of Hive CLI. It provides a web-based GUI for executing Hive queries and commands.
- Hive MetaStore - It is a central repository that stores all the structure information of various tables and partitions in the warehouse. It also includes metadata of column and its type information, the serializers and deserializers which is used to read and write data and the corresponding HDFS files where the data is stored.
- Hive Server - It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver.
- Hive Driver - It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver. It transfers the queries to the compiler.
- Hive Compiler - The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions. It converts HiveQL statements into MapReduce jobs.
- Hive Execution Engine - Optimizer generates the logical plan in the form of DAG of map-reduce tasks and HDFS tasks. In the end, the execution engine executes the incoming tasks in the order of their dependencies.

Working of Hive

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.

5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job** Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result** The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

PIG

What is Apache Pig?

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used

with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Apache Pig.

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

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

Why Do We Need Apache Pig?

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

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

Features of Pig

Apache Pig comes with the following features –

- **Rich set of operators** – It provides many operators to perform operations like join, sort, filer, etc.
- **Ease of programming** – Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.
- **Optimization opportunities** – The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.

- **Extensibility** – Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF's** – Pig provides the facility to create **User-defined Functions** in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data** – Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Apache Pig Vs MapReduce

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

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

Apache Pig Vs SQL

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

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

In addition to above differences, Apache Pig Latin –

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

Apache Pig Vs Hive

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

Apache Pig	Hive
Apache Pig uses a language called **Pig Latin**. It was originally created at **Yahoo**.	Hive uses a language called **HiveQL**. It was originally created at **Facebook**.
Pig Latin is a data flow language.	HiveQL is a query processing language.
Pig Latin is a procedural language and it fits in pipeline paradigm.	HiveQL is a declarative language.

Apache Pig can handle structured, unstructured, and semi-structured data.	Hive is mostly for structured data.

Applications of Apache Pig

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

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

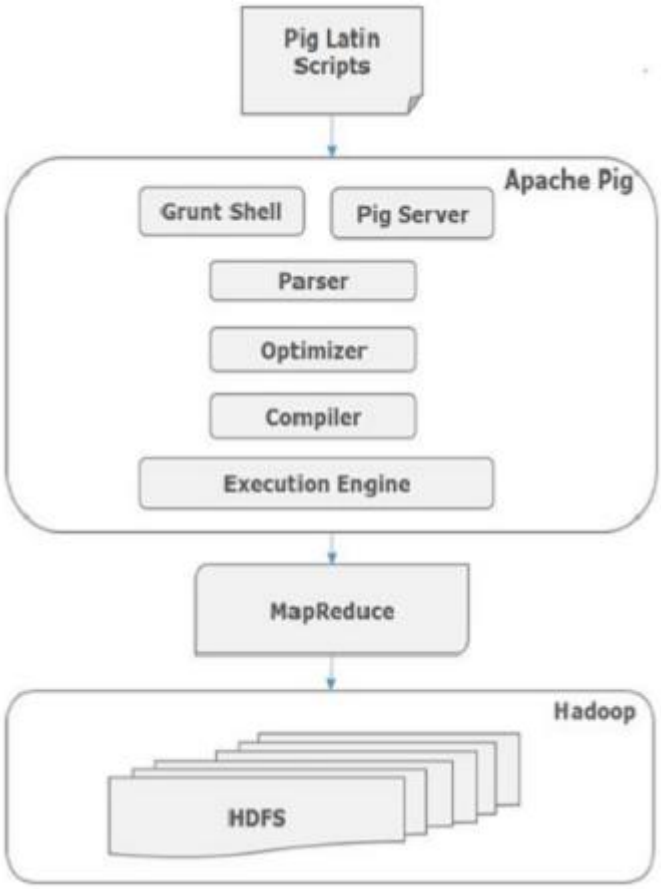
Apache Pig – History

In **2006**, Apache Pig was developed as a research project at Yahoo, especially to create and execute MapReduce jobs on every dataset. In **2007**, Apache Pig was open sourced via Apache incubator. In **2008**, the first release of Apache Pig came out. In **2010**, Apache Pig graduated as an Apache top-level project.

The language used to analyze data in Hadoop using Pig is known as **Pig Latin**. It is a highlevel data processing language which provides a rich set of data types and operators to perform various operations on the data.

To perform a particular task Programmers using Pig, programmers need to write a Pig script using the Pig Latin language, and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded). After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output.

Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy. The architecture of Apache Pig is shown below.



Apache Pig Components

As shown in the figure, there are various components in the Apache Pig framework. Let us take a look at the major components.

Parser

Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.

In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.

Optimizer

The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.

Compiler

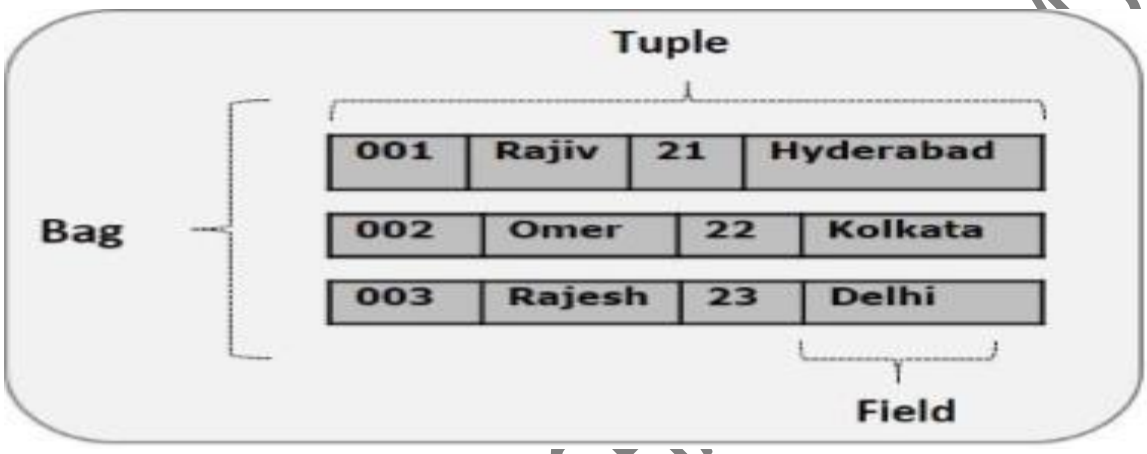
The compiler compiles the optimized logical plan into a series of MapReduce jobs.

Execution engine

Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

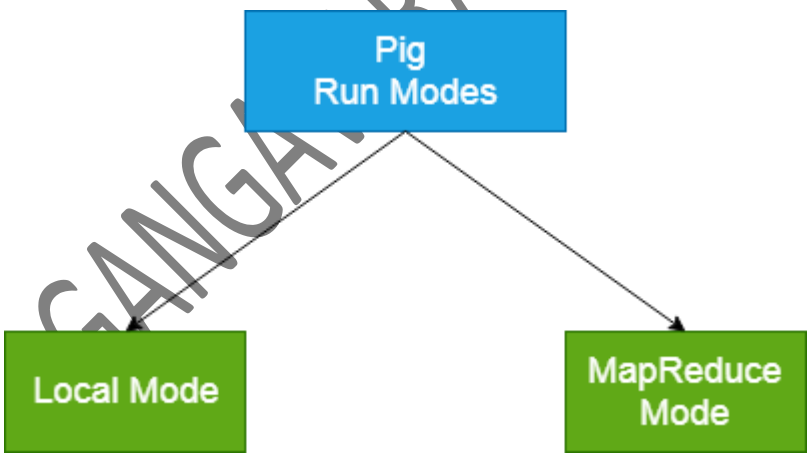
Pig Latin Data Model

The data model of Pig Latin is fully nested and it allows complex non-atomic datatypes such as **map** and **tuple**. Given below is the diagrammatical representation of Pig Latin's data model.



Apache Pig Run Modes

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



Local Mode

- It executes in a single JVM and is used for development experimenting and prototyping.
- Here, files are installed and run using localhost.

- The local mode works on a local file system. The input and output data stored in the local file system.

The command for local mode grunt shell:

1. $ pig-x local

MapReduce Mode

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

The command for Map reduce mode:

1. $ pig

Ways to execute Pig Program

These are the following ways of executing a Pig program on local and MapReduce mode: -

- **Interactive Mode** - In this mode, the Pig is executed in the Grunt shell. To invoke Grunt shell, run the pig command. Once the Grunt mode executes, we can provide Pig Latin statements and command interactively at the command line.
- **Batch Mode** - In this mode, we can run a script file having a .pig extension. These files contain Pig Latin commands.
- **Embedded Mode** - In this mode, we can define our own functions. These functions can be called as UDF (User Defined Functions). Here, we use programming languages like Java and Python.

Pig Latin

The Pig Latin is a data flow language used by Apache Pig to analyze the data in Hadoop. It is a textual language that abstracts the programming from the Java MapReduce idiom into a notation.

Pig Latin Statements

The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.

- It can span multiple lines.
- Each statement must end with a semi-colon.
- It may include expression and schemas.
- By default, these statements are processed using multi-query execution

Pig Latin Conventions

Convention	Description
( )	The parenthesis can enclose one or more items. It can also be used to indicate the tuple data type. Example - (10, xyz, (3,6,9))
[ ]	The straight brackets can enclose one or more items. It can also be used to indicate the map data type. Example - [INNER | OUTER]
{ }	The curly brackets enclose two or more items. It can also be used to indicate the bag data type Example - { block | nested_block }
...	The horizontal ellipsis points indicate that you can repeat a portion of the code. Example - cat path [path ...]

Pig Example

Use case: Using Pig find the most occurred start letter.

Solution:

Case 1: Load the data into bag named "lines". The entire line is stuck to element line of type character array.

```
grunt> lines  = LOAD "/user/Desktop/data.txt" AS (line: chararray);
```

Case 2: The text in the bag lines needs to be tokenized this produces one word per row.

```
grunt>tokens = FOREACH lines GENERATE flatten(TOKENIZE(line))  As token:
 chararray;
```

Case 3: To retain the first letter of each word type the below command .This commands uses substring method to take the first character.

```
grunt>letters = FOREACH tokens  GENERATE SUBSTRING(0,1)   as letter : cha
rarray;
```

Case 4: Create a bag for unique character where the grouped bag will contain the same character for each occurrence of that character.

```
grunt>lettergrp = GROUP letters by letter;
```

Case 5: The number of occurrence is counted in each group.

```
grunt>countletter  = FOREACH  lettergrp  GENERATE group  , COUNT(letters)
;
```

Case 6: Arrange the output according to count in descending order using the commands below.

```
grunt> OrderCnt = ORDER countletter  BY  $1  DESC;
```

Case 7: Limit to One to give the result.

```
grunt> result  =LIMIT    OrderCnt    1;
```

Case 8: Store the result in HDFS . The result is saved in output directory under sonoo folder.

```
grunt> STORE   result   into 'home/sonoo/output';
```

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

KG REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

B. Tech III Year II Semester **Academic Year: 2023-2024**

SUBJECT: BIG DATA ANALYTICTS **SUBJECT CODE: KG21CD605** **REGULATION: KGR21**

UNIT-5 NOTES

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

Data Analytics with R Machine Learning:

What is machine learning

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for **building mathematical models and making predictions using historical data or information**. Currently, it is being used for various tasks such as **image recognition**, **speech recognition**, **email filtering**, **Facebook auto-tagging**, **recommender system**, and many more.

The machine learning techniques such as **Supervised**, **Unsupervised**, and **Reinforcement** learning. Regression and classification models, clustering methods, hidden Markov models, and various sequential models.

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

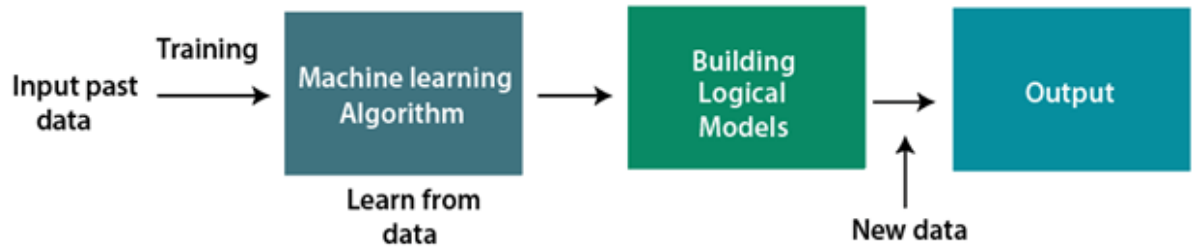
With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

A machine has the ability to learn if it can improve its performance by gaining more data.

How does Machine Learning work

A Machine Learning system **learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it**. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Need for Machine Learning

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.

We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases, Currently, machine learning is used in **self-driving cars**, **cyber fraud detection**, **face recognition**, and **friend suggestion by Facebook**, etc. Various top companies such as Netflix and Amazon have build machine learning models that are using a vast amount of data to analyze the user interest and recommend product accordingly.

Following are some key points which show the importance of Machine Learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

Data Analytics with R Machine Learning:

What is R Analytics?

R analytics is data analytics using R programming language, an open-source language used for statistical computing or graphics. This programming language is often used in statistical analysis and data mining. It can be used for analytics to identify patterns and build practical models. R not only can help analyze organizations' data, but also be used to help in the creation and development of software applications that perform statistical analysis.

With a graphical user interface for developing programs, R supports a variety of analytical modeling techniques such as classical statistical tests, clustering, time-series analysis, linear and nonlinear modeling, and more. The interface has four windows: the script window, console window, workspace and history window, and tabs of interest (help, packages, plots, and files). R allows for publication-ready plots and graphics and for storage of reusable analytics for future data.

R has become increasingly popular over many years and remains a top analytics language for many universities and colleges. It is well established today within academia as well as among corporations around the world for delivering robust, reliable, and accurate analytics. While R programming was originally seen as difficult for non-statisticians to learn, the user interface has become more user-friendly in recent years. It also now allows for extensions and other plugins like R Studio and R Excel, making the learning process easier and faster for new business analysts and other users. It has become the industry standard for

statistical analysis and data mining projects and is due to grow in use as more graduates enter the workforce as R-trained analysts.

What are the Benefits of R Analytics?

Business analytics in R allows users to analyze business data more efficiently. The following are some of the main benefits realized by companies employing R in their analytics programs:

Democratizing Analytics Across the Organization: R can help democratize analytics by enabling business users with interactive data visualization and reporting tools. R can be used for data science by non data scientists so that business users and citizen data scientists can make better business decisions. R analytics can also reduce time spent on data preparation and data wrangling, allowing data scientists to focus on more complex data science initiatives.

Providing Deeper, More Accurate Insights: Today, most successful companies are data driven and therefore data analytics affects almost every area of business. And while there are a whole host of powerful data analytics tools, R can help create powerful models to analyze large amounts of data. With more precise data collection and storage through R analytics, companies can deliver more valuable insights to users. Analytics and statistical engines using R provide deeper, more accurate insights for the business. R can be used to develop very specific, in-depth analyses.

Leveraging Big Data: R can help with querying big data and is used by many industry leaders to leverage big data across the business. With R analytics, organizations can surface new insights in their large data sets and make sense of their data. R can handle these big datasets and is arguably as easy if not easier for most analysts to use as any of the other analytics tools available today.

Creating Interactive Data Visualizations: R is also helpful for data visualization and >data exploration because it supports the creation of graphs and diagrams. It includes the ability to create interactive visualizations and 3D charts and graphs that are helpful for communicating with business users.

How Can R analytics Be Implemented?

While R programming was originally designed for statisticians, it can be implemented for a variety of uses including predictive analytics, data modeling, and data mining. Businesses can implement R to create custom models for data collection, clustering, and analytics. R analytics can provide a valuable way to quickly develop models targeted at understanding specific areas of the business and delivering tailored insights on day-to-day needs.

R analytics can be used for the following purposes:

- Statistical testing
- Prescriptive analytics
- Predictive analytics
- Time-series analysis
- What-if analysis
- Regression models
- Data exploration
- Forecasting
- Text mining
- Data mining
- Visual analytics
- Web analytics
- Social media analytics
- Sentiment analysis

R can be used to solve real-world business problems by turbocharging an organization's analytics program. It can be integrated into a business's analytics platform to help users get the most out of their data. With an extensive library of R functions and advanced statistical techniques, R can be used to apply statistical models to your analysis and better understand trends in the data. It can help predict potential business outcomes, identify opportunities and risks and create interactive dashboards to gain a comprehensive view of the data. This can lead to better business decisions and increased revenue.

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.

Popular R Language Packages Used to Implement Machine Learning

- **lattice:** The lattice package supports the creation of the graphs displaying the variable or relation between multiple variables with conditions.
- **DataExplorer:** This R package focus to automate the data visualization and data handling so that the user can pay attention to data insights of the project.
- **Dalex(Descriptive Machine Learning Explanations):** This package helps to provide various explanations for the relation between the input variable and its output. It helps to understand the complex models of machine learning
- **dplyr:** This R package is used to summarize the tabular data of machine learning with rows and columns. It applies the "split-apply-combine" approach.
- **Esquisse:** This R package is used to explore the data quickly to get the information it holds. It also allows to plot bar graph, histograms, curves, and scatter plots.
- **caret:** This R package attempts to streamline the process for creating predictive models.
- **janitor:** This R package has functions for examining and cleaning dirty data. It is basically built for the purpose of user-friendliness for beginners and intermediate users.
- **rpart:** This R package helps to create the classification and regression models using two-stage procedures. The resulting models are represented as binary trees.

Application Of R in Machine Learning

There are many top companies like Google, Facebook, Uber, etc using the R language for application of Machine Learning. The application are:

- Social Network Analytics
- To analyze trends and patterns
- Getting insights for behaviour of users
- To find the relationships between the users
- Developing analytical solutions
- Accessing charting components
- Embedding interactive visual graphics

Example of Machine Learning Problems

- **Web search like Siri, Alexa, Google, Cortona:** Recognize the user's voice and fulfill the request made
- **Social Media Service:** Help people to connect all over the world and also show the recommendations of the people we may know
- **Online Customer Support:** Provide high convenience of customer and efficiency of support agent
- **Intelligent Gaming:** Use high level responsive and adaptive non player characters similar to human like intelligence
- **Product Recommendation:** A software tool used to recommend the product that you might like to purchase or engage with
- **Virtual Personal Assistance:** It is the software which can perform the task according to the instructions provided
- **Traffic Alerts:** Help to switch the traffic alerts according to the situation provided
- **Online Fraud Detection:** Check the unusual functions performed by the user and detect the frauds
- **Healthcare:** Machine Learning can manage a large amount of data beyond the imagination of normal human being and help to identify the illness of the patient according to symptoms
- **Real world example:** When you search for some kind of cooking recipe on youTube, you will see the recommendations below with the title "You May Also Like This". This is a common use of Machine Learning.

Types of Machine Learning Problems

- **Regression:** The regression technique helps the machine learning approach to predict continuous values. For example, the price of a house.
- **Classification:** The input is divided into one or more classes or categories for the learner to produce a model to assign unseen modules. For example, in the case of email fraud, we can divide the emails into two classes i.e "spam" and "not spam".
- **Clustering:** This technique follows the summarization, finding a group of similar entities. For example, we can gather and take readings of the patients in the hospital.
- **Association:** This technique finds co-occurring events or items. For example, market-basket.
- **Anomaly Detection:** This technique works by discovering abnormal cases or behavior. For example, credit card fraud detection.
- **Sequence Mining:** This technique predicts the next stream event. For example, click-stream event.
- **Recommendation:** This technique recommends the item. For example, songs or movies according to the celebrity in it.

Packages

We will be using, directly or indirectly, the following packages through the chapters:

- caret
- ggplot2
- mlbench
- class
- caTools
- randomForest
- impute
- ranger
- kernlab
- class
- glmnet
- naivebayes
- rpart
- rpart.plot

Supervised Learning:

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

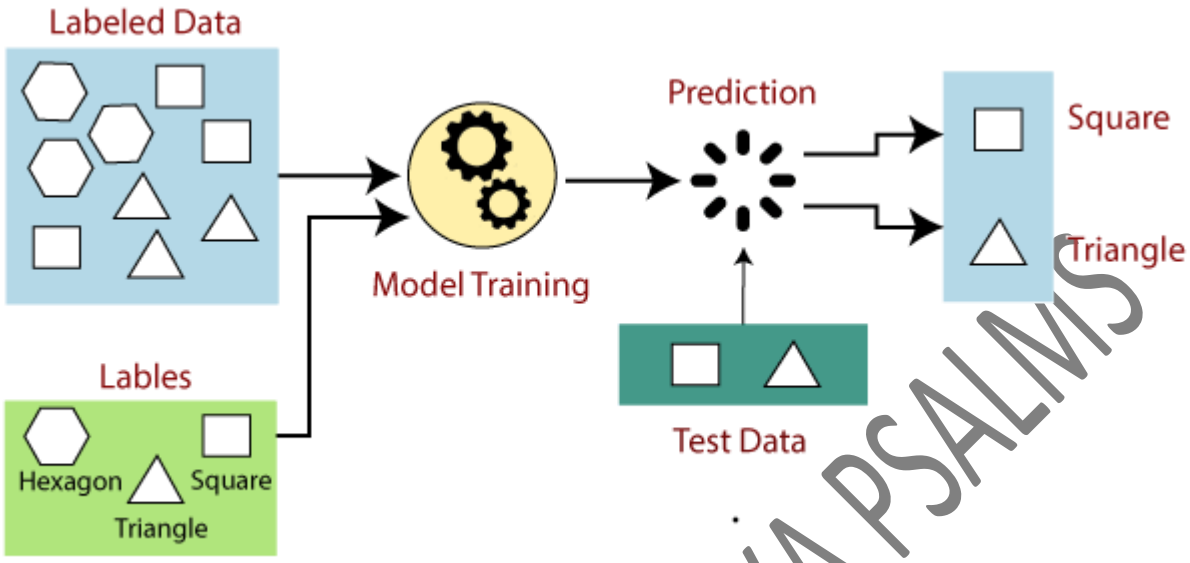
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.

- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

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” , “disease” or “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
- Naive Bayes Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

Regression:

- A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables.
- A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.
- In regression, we normally have one dependent variable and one or more independent variables.

- Here we try to "regress" the value of the dependent variable "Y" with the help of the independent variables.
- In other words, we are trying to understand, how the value of 'Y' changes w.r.t change in 'X'.

For the regression analysis is be a successful method, we understand the following terms:

- **Dependent Variable:** This is the variable that we are trying to understand or forecast.
- **Independent Variable:** These are factors that influence the analysis or target variable and provide us with information regarding the relationship of the variables with the target variable.

Regression analysis is used for prediction and forecasting. This statistical method is used across different industries such as,

- Financial Industry- Understand the trend in the stock prices, forecast the prices, and evaluate risks in the insurance domain
- Marketing- Understand the effectiveness of market campaigns, and forecast pricing and sales of the product.
- Manufacturing- Evaluate the relationship of variables that determine to define a better engine to provide better performance
- Medicine- Forecast the different combinations of medicines to prepare generic medicines for diseases.

Logistic Regression

- Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of data belonging to a given class or not.
- It is a kind of statistical algorithm, which analyze the relationship between a set of independent variables and the dependent binary variables.
- It is a powerful tool for decision-making. For example email spam or not.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact

value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Classification

In Regression algorithms, we have predicted the output for continuous values, but to predict the categorical values, we need Classification algorithms.

What is the Classification Algorithm?

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a

Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

Naïve Bayes classifer

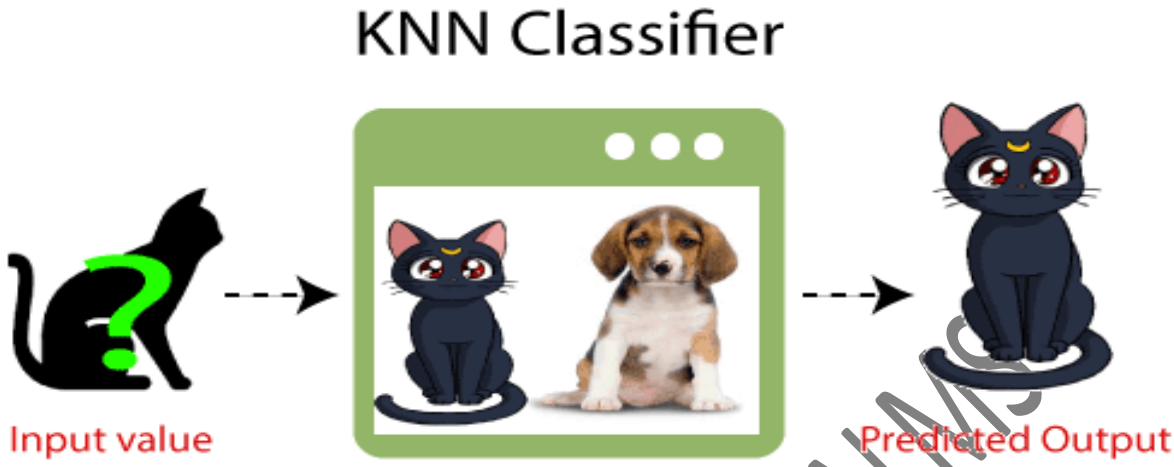
- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

K-NN (k nearest neighbors)

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity.

- This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

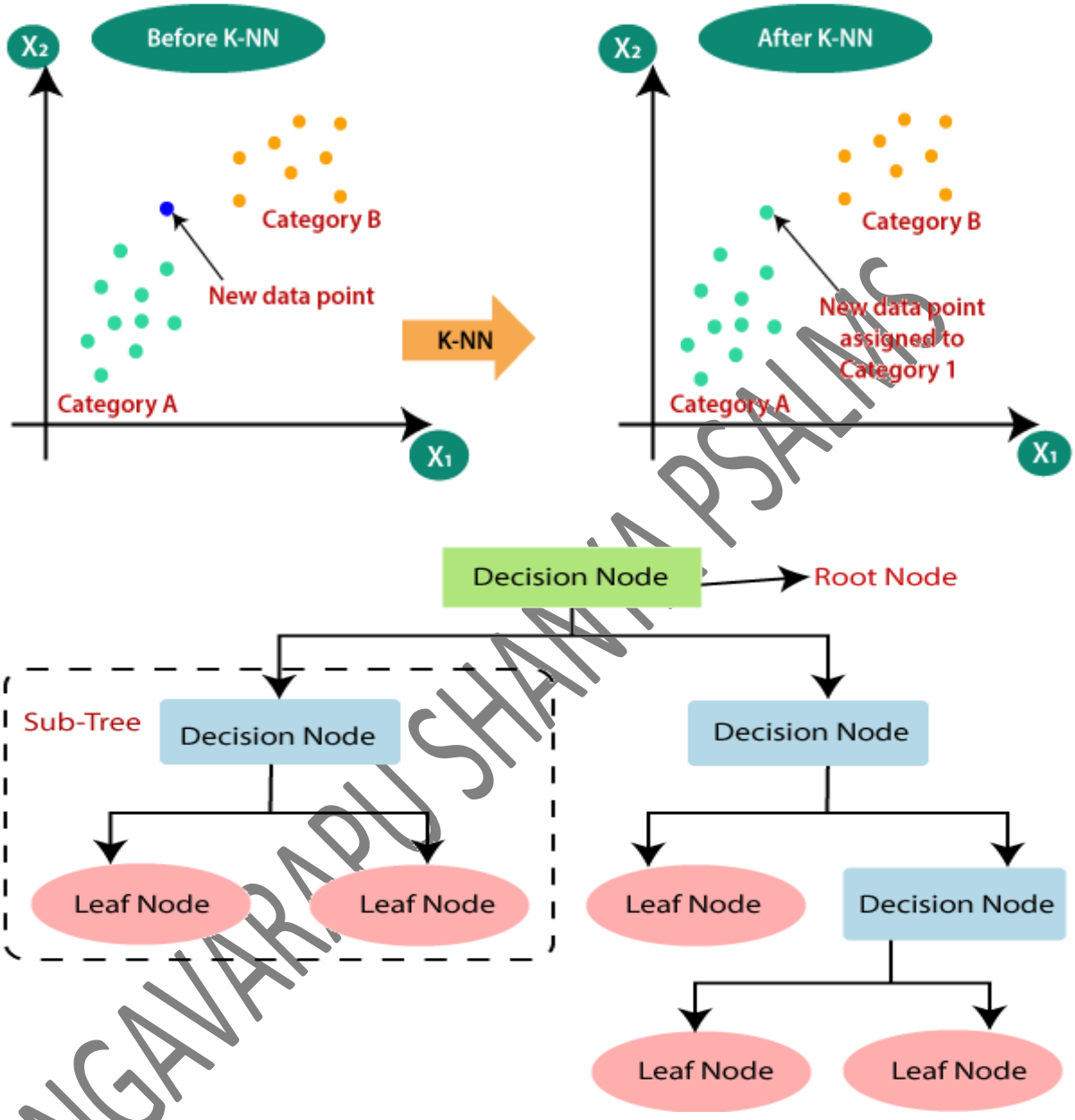
  Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



How does K-NN work?

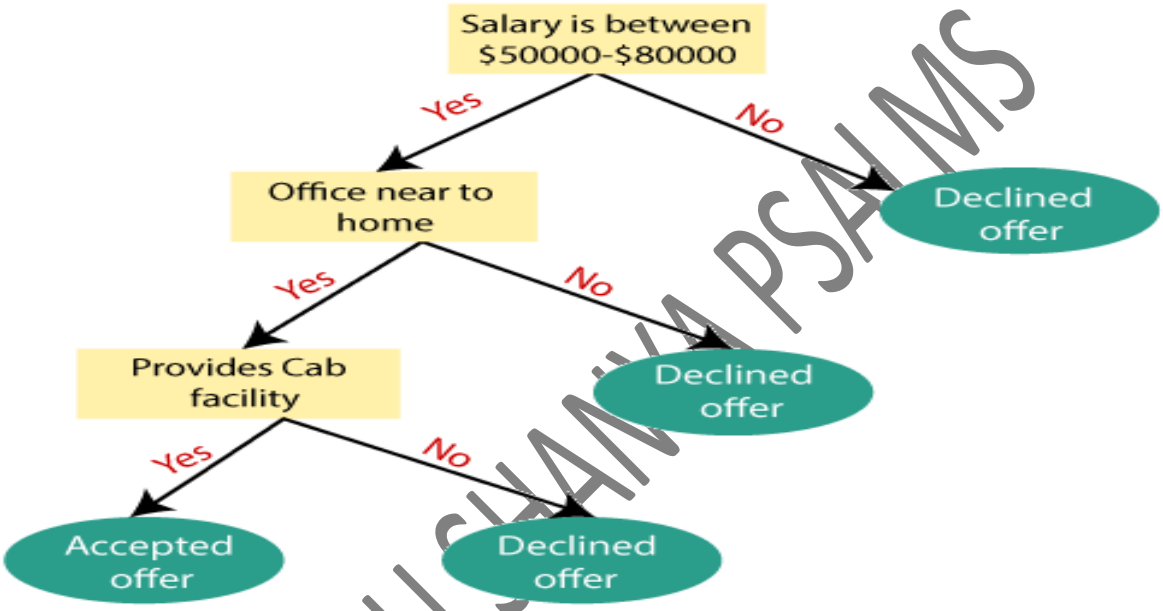
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.



- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

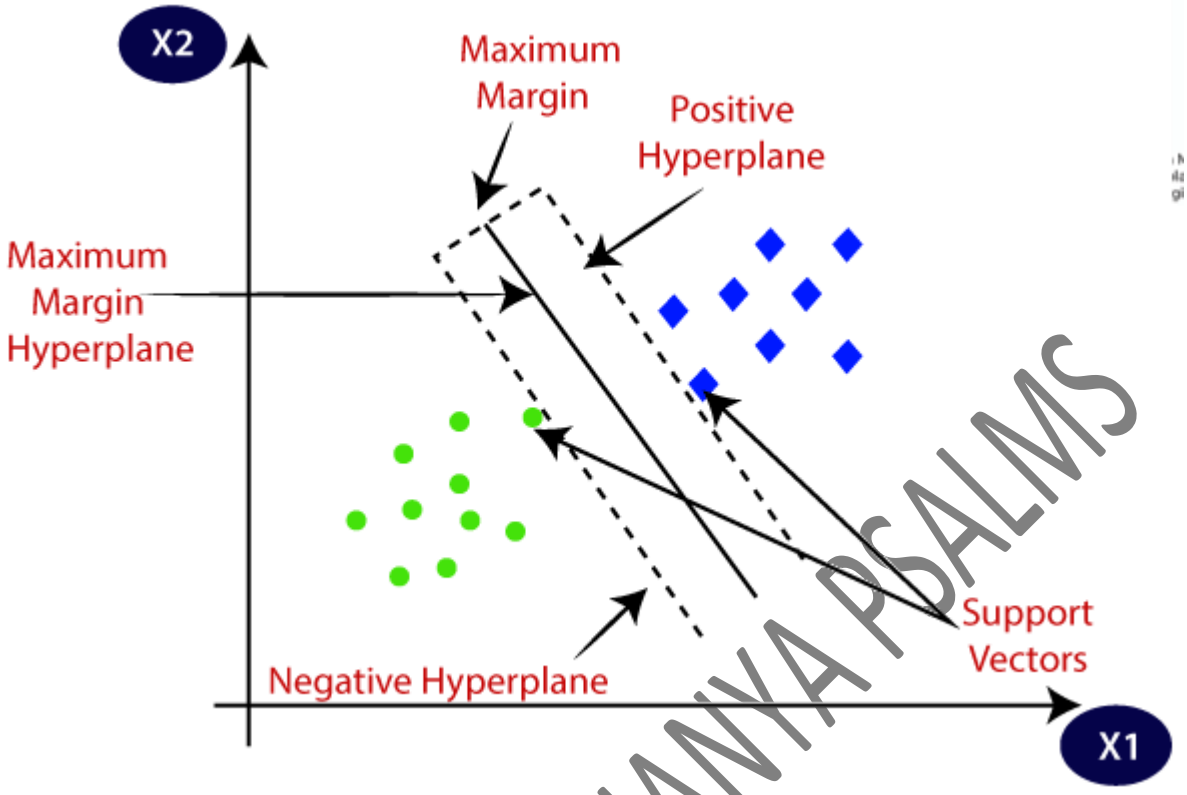


SUPPORT VECTOR MACHINES

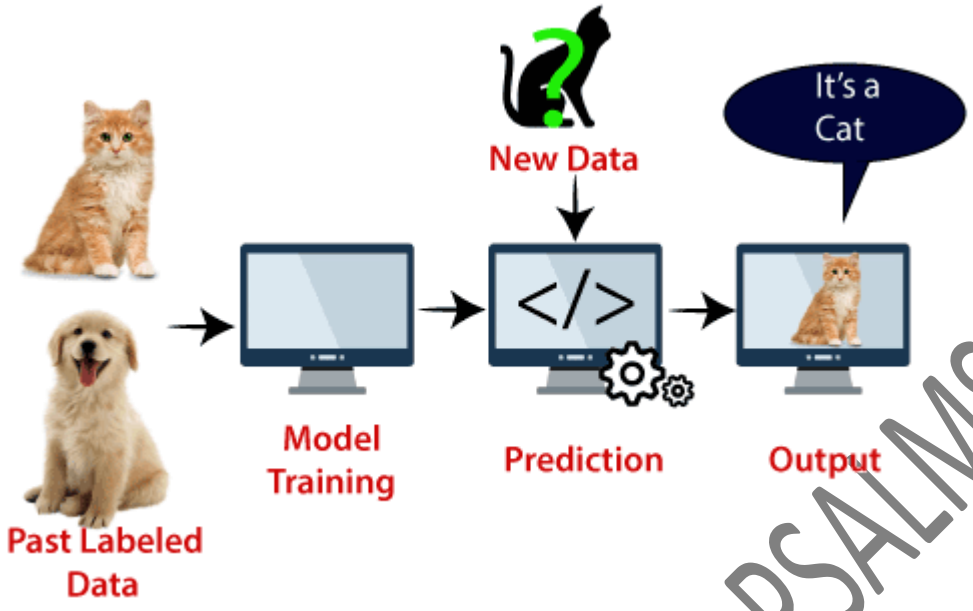
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

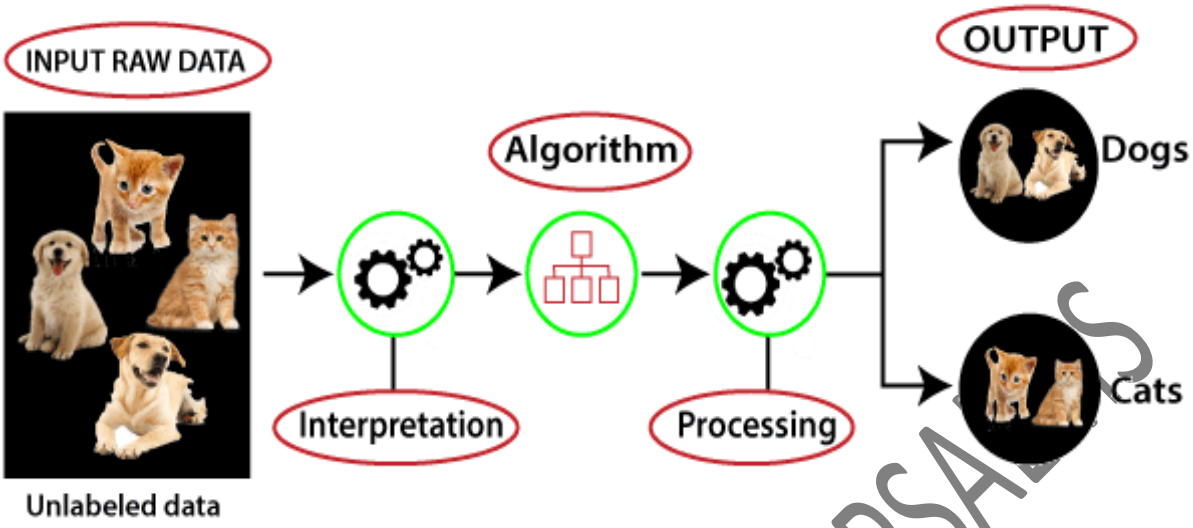
Unsupervised learning

It uses machine learning algorithms to analyze and cluster unlabeled datasets.

These algorithms discover hidden patterns or data groupings without the need for human intervention.

- Unsupervised machine learning finds all kind of unknown patterns in data.
- Unsupervised methods help you to find features which can be useful for categorization.
- It is taken place in real time, so all the input data to be analyzed and labeled in the presence of learners.
- It is easier to get unlabeled data from a computer than labeled data, which needs manual intervention.

Working of Unsupervised Learning



Clustering Types of Unsupervised Learning Algorithms

Below are the clustering types of Unsupervised Machine Learning algorithms:

Unsupervised learning problems further grouped into clustering and association problems.

Clustering

Clustering is an important concept when it comes to unsupervised learning.

It mainly deals with finding a structure or pattern in a collection of uncategorized data. Unsupervised Learning Clustering algorithms will process your data and find natural clusters(groups) if they exist in the data.

There are different types of clustering you can utilize:

Exclusive (partitioning)

In this clustering method, Data are grouped in such a way that one data can belong to one cluster only.

Example: K-means

Agglomerative

In this clustering technique, every data is a cluster. The iterative unions between the two nearest clusters reduce the number of clusters.

Example: Hierarchical clustering

Overlapping

In this technique, fuzzy sets is used to cluster data. Each point may belong to two or more clusters with separate degrees of membership.

Here, data will be associated with an appropriate membership value. Example: Fuzzy C-Means

Probabilistic

This technique uses probability distribution to create the clusters.

Clustering Types

Following are the clustering types of Machine Learning:

- Hierarchical clustering
- K-means clustering
- K-NN (k nearest neighbors)
- Principal Component Analysis
- Singular Value Decomposition
- Independent Component Analysis

Hierarchical Clustering

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

Agglomerative Hierarchical clustering

The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this

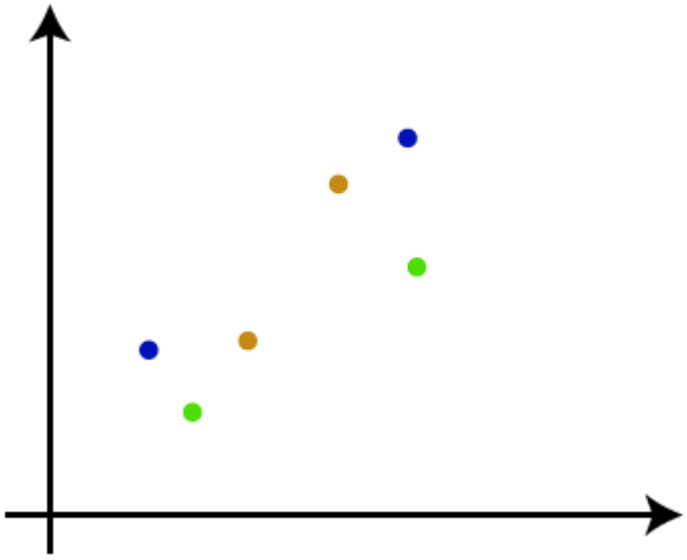
algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

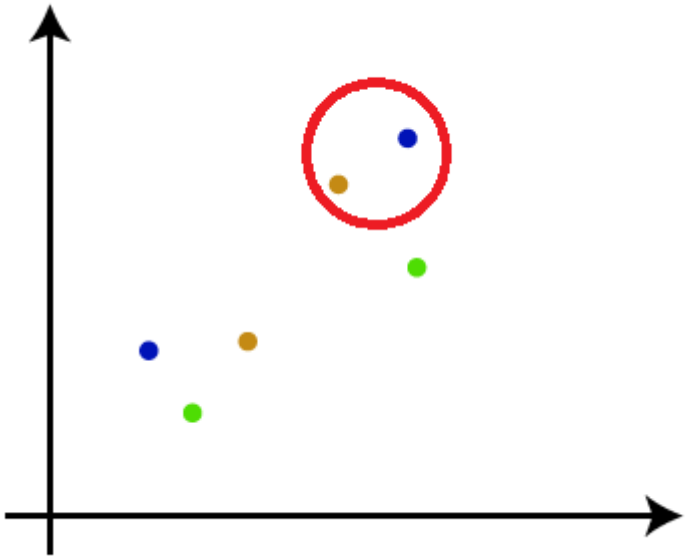
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

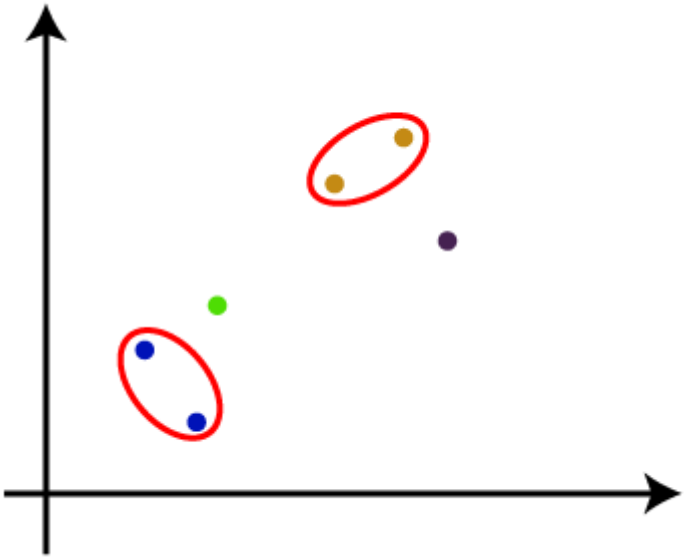
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



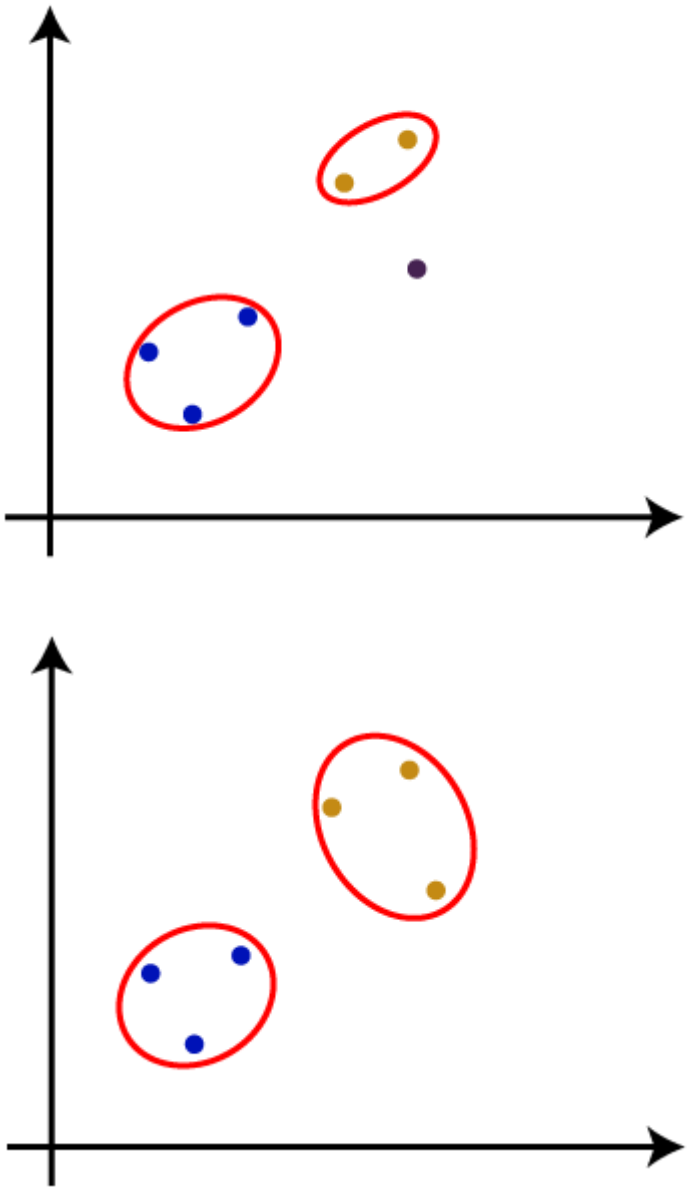
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

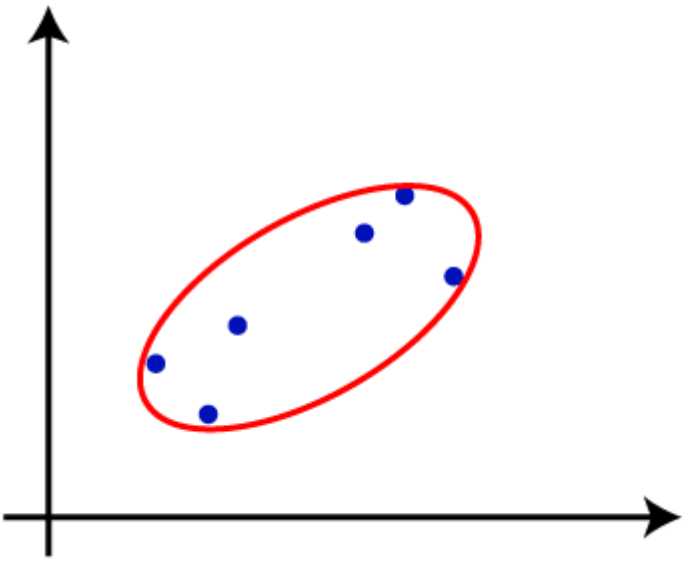


- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:



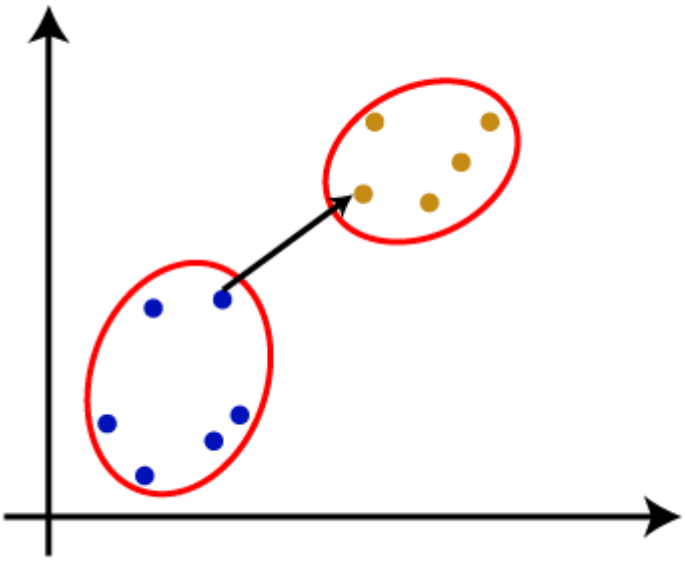


- **Step-5:** Once all the clusters are combined into onebig cluster, develop the dendrogram to divide the clusters as per the problem.

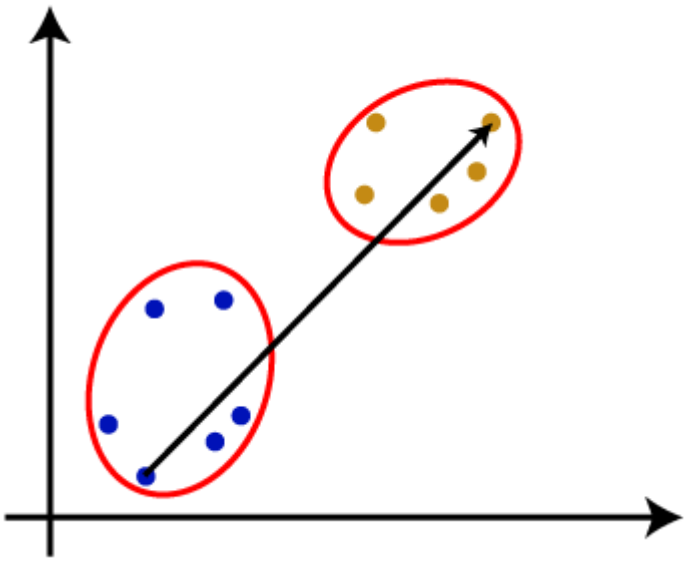
Measure for the distance between two clusters

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

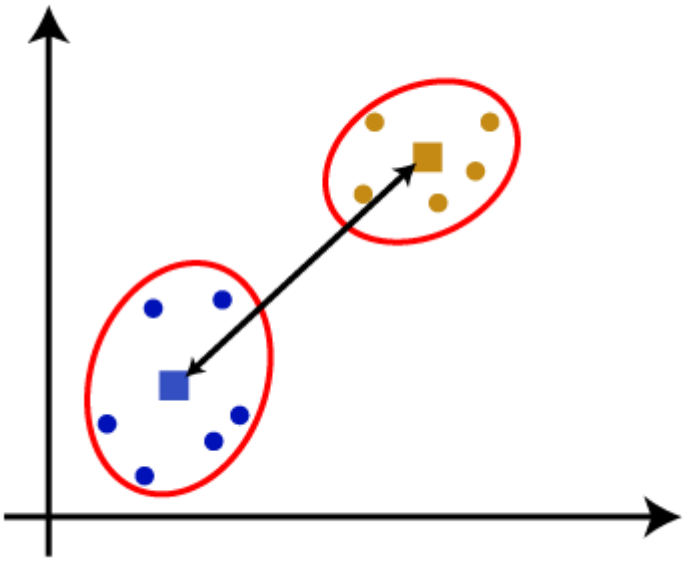
1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:



2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.
4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

K-means Clustering

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

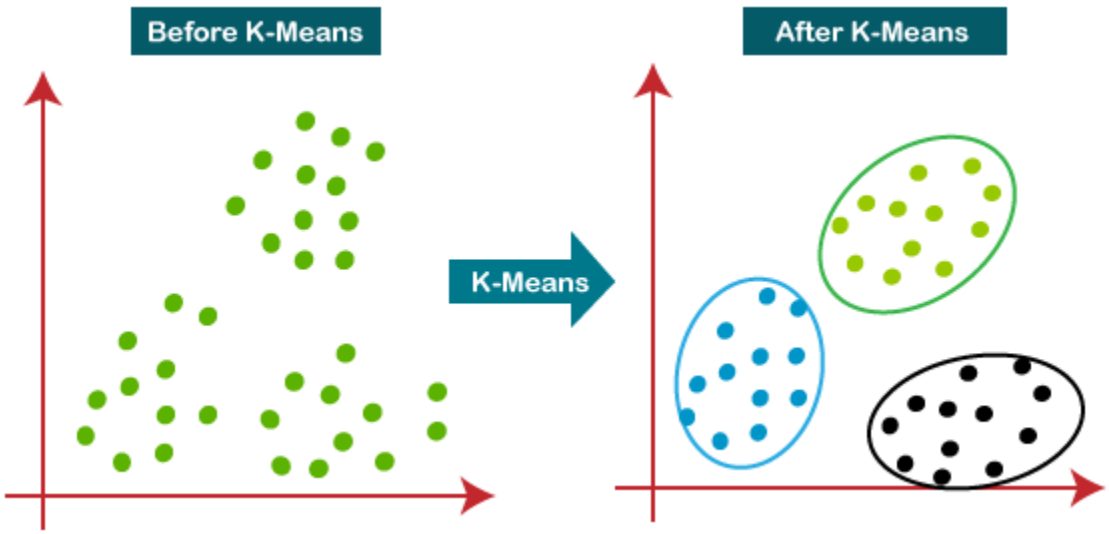
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

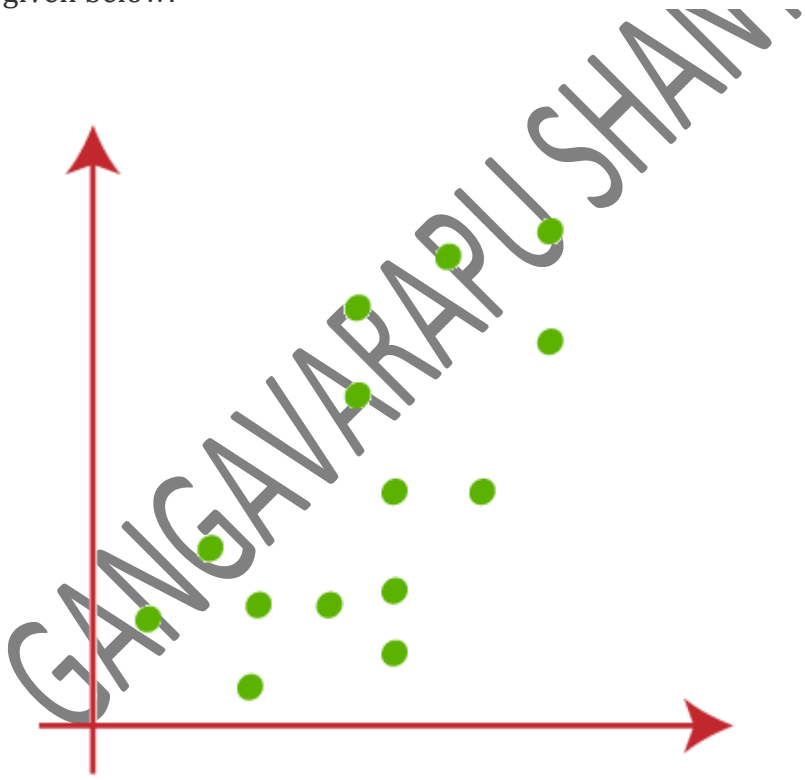
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

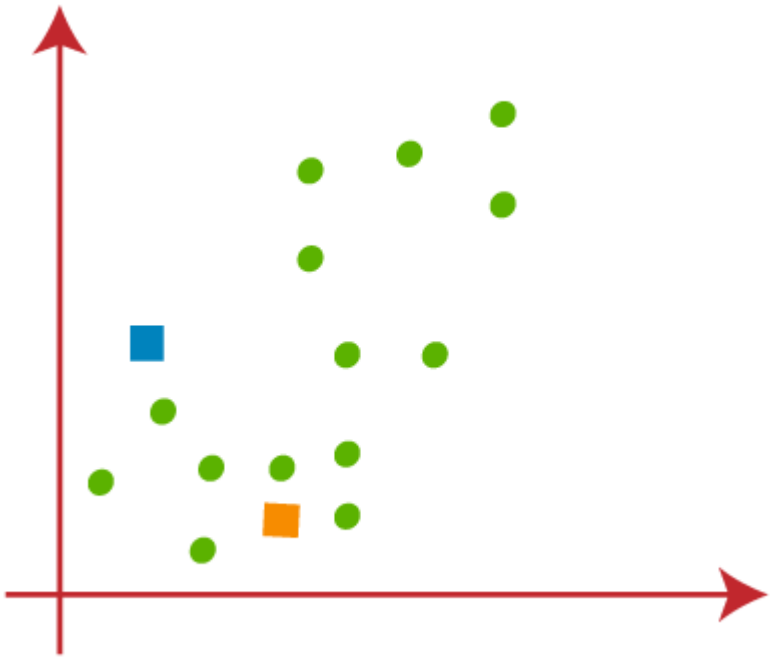
Let's understand the above steps by considering the visual plots:

Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



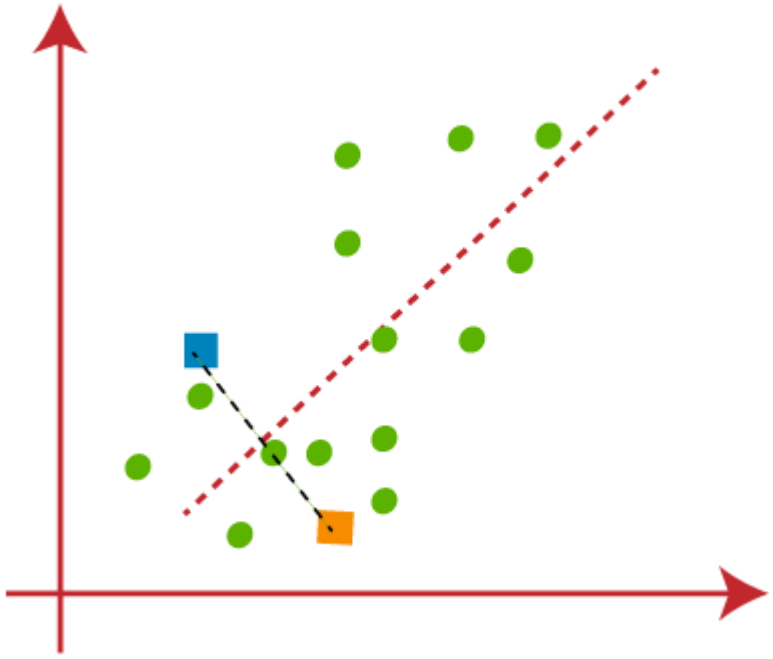
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.

- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:

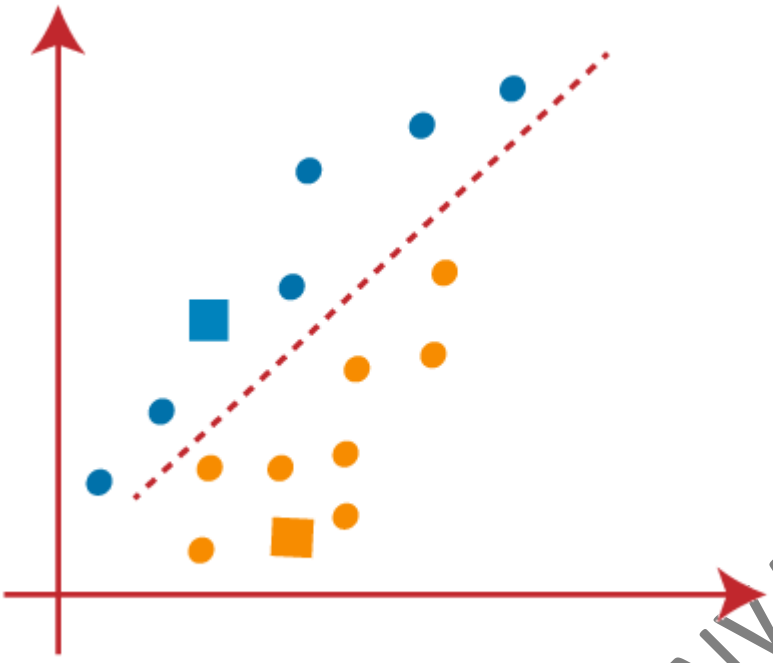


- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between

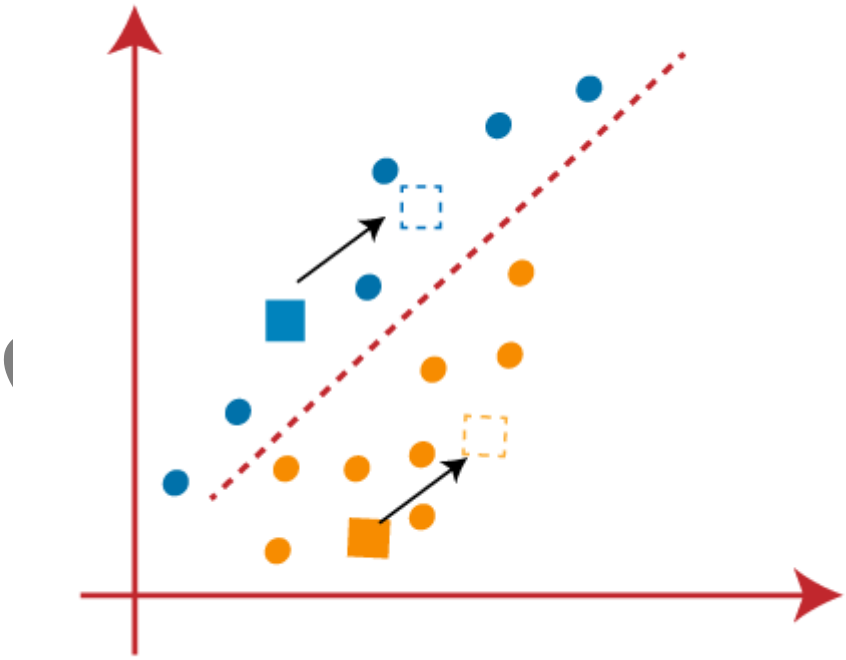
both the centroids. Consider the below image:



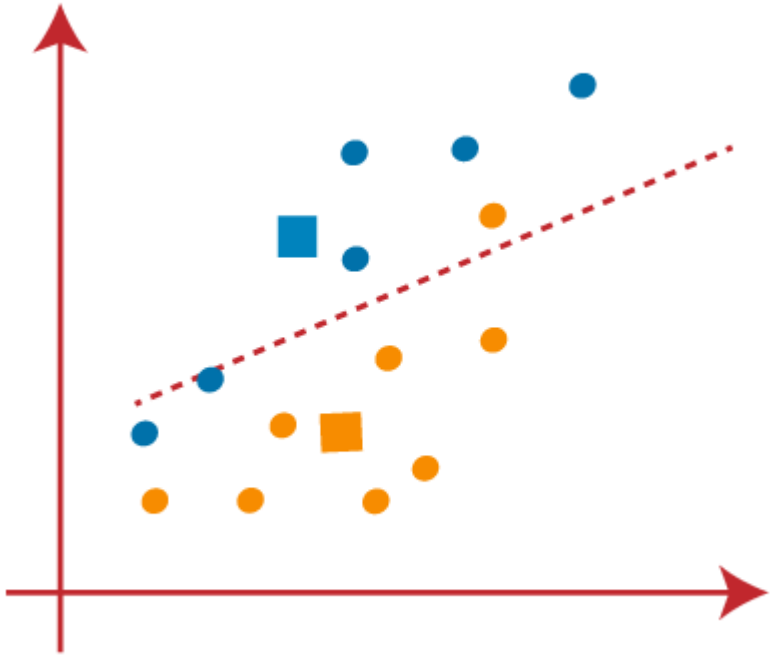
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



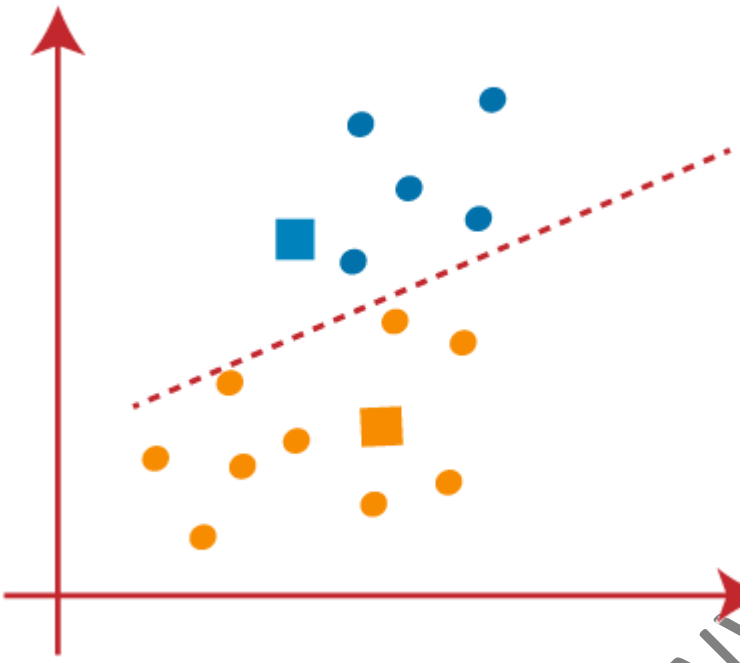
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:



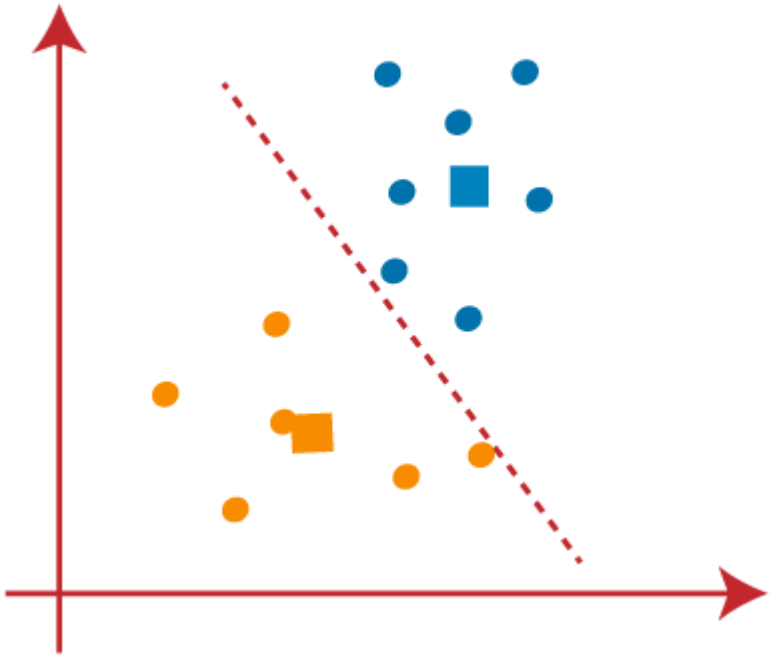
From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.



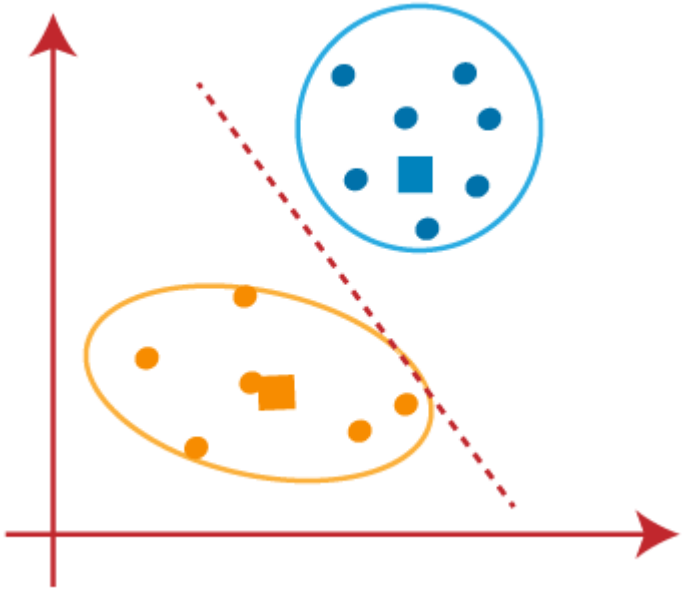
As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:

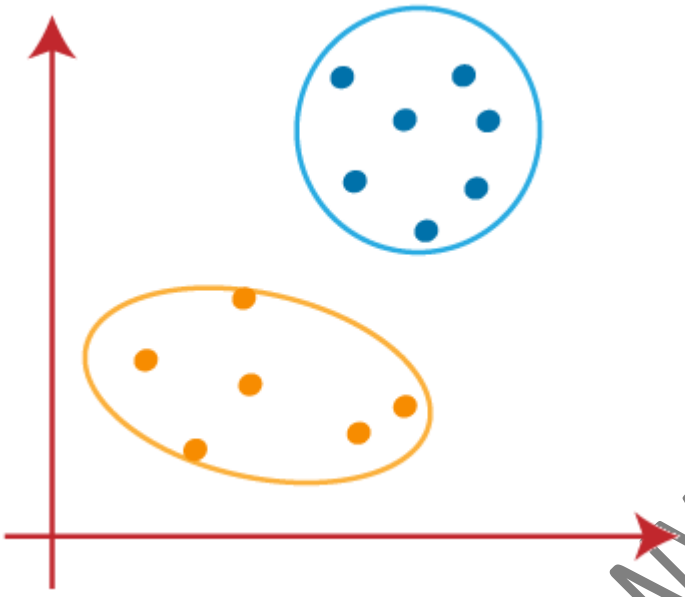
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:

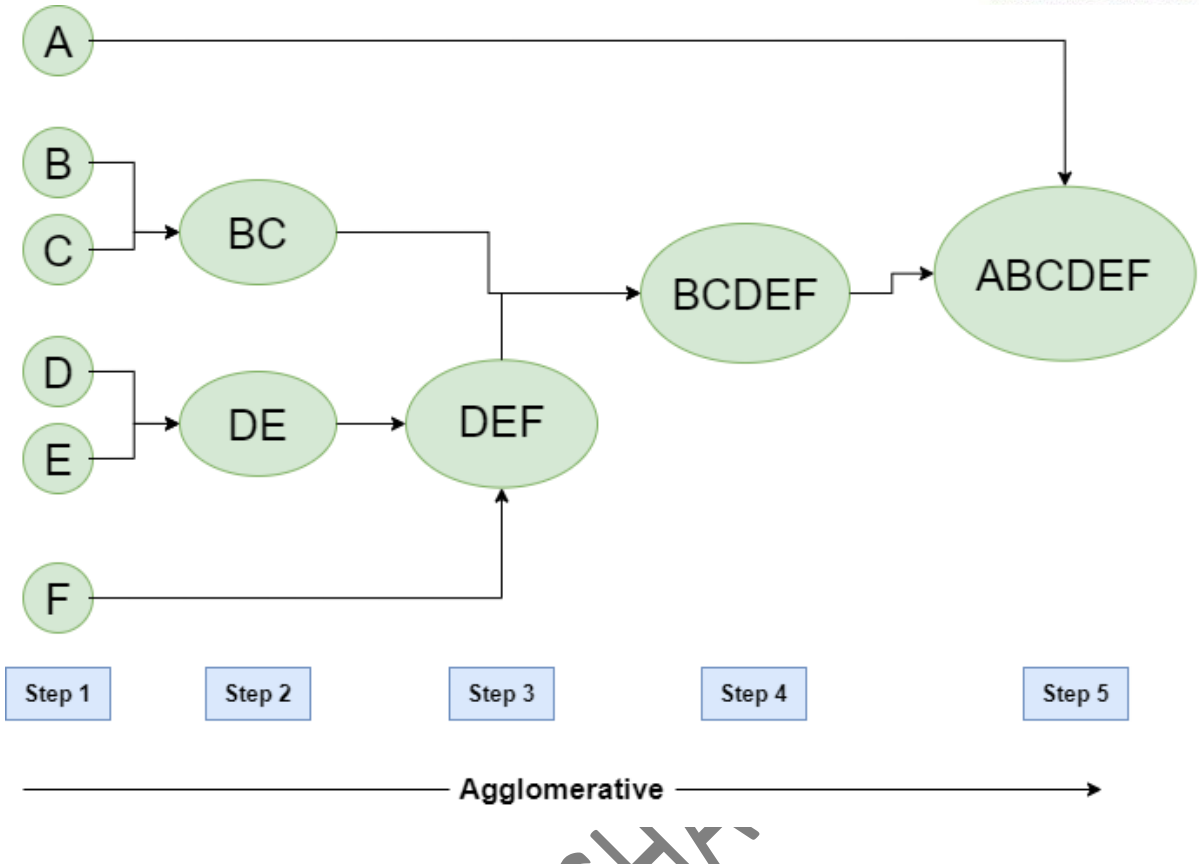


As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



Agglomerative clustering

It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

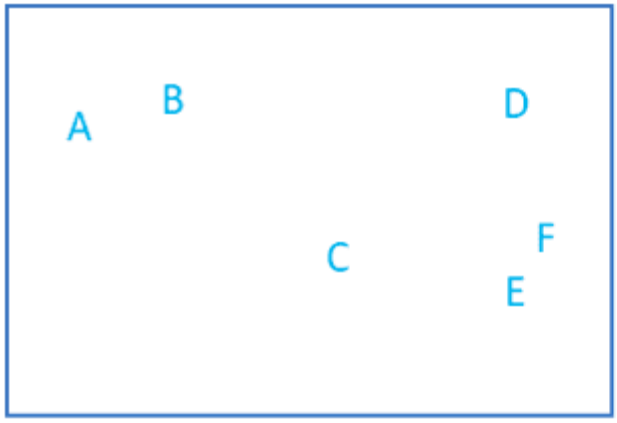


Steps:

- Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.
- In the second step, comparable clusters are merged together to form a single cluster. Let's say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly to cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]
- We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]
- Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We're now left with clusters [(A), (BCDEF)].
- At last, the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

Dendrogram

A *dendrogram* is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from *hierarchical clustering*. The main use of a dendrogram is to work out the best way to allocate objects to clusters. The dendrogram below shows the hierarchical clustering of six *observations* shown on the *scatterplot* to the left. (Dendrogram is often miswritten as dendogram.

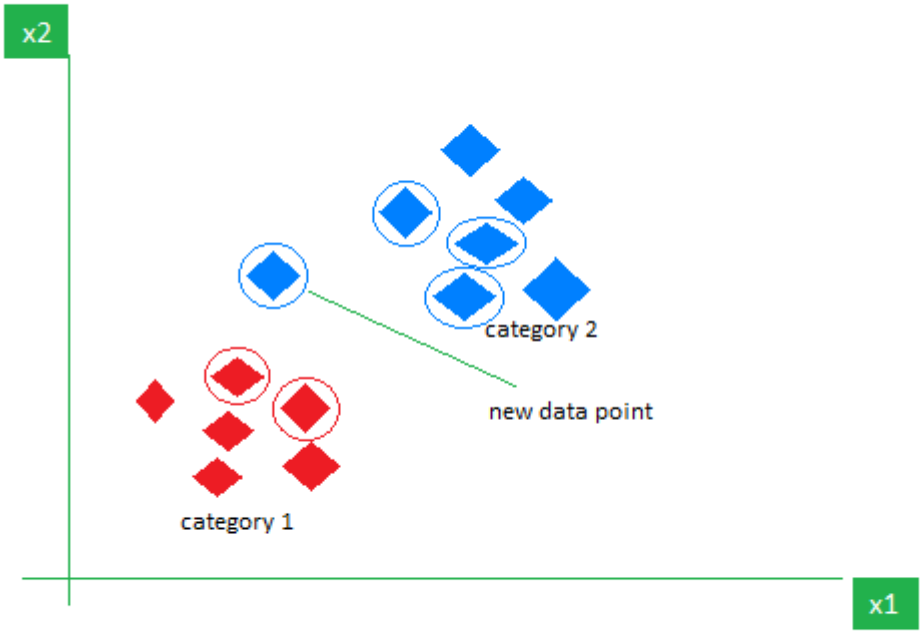


K- Nearest neighbors

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

- Euclidean Distance
- Manhattan Distance
- Minkowski Distance

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

Manhattan Distance

This distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be boolean as well as string values.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

Applications of the KNN Algorithm

- **Data Preprocessing** – While dealing with any Machine Learning problem we first perform the EDA part in which if we find that the data contains missing values then there are multiple imputation methods are available as well. One of such method is KNN

Imputer which is quite effective ad generally used for sophisticated imputation methodologies.

- **Pattern Recognition** – KNN algorithms work very well if you have trained a KNN algorithm using the MNIST dataset and then performed the evaluation process then you must have come across the fact that the accuracy is too high.
- **Recommendation Engines** – The main task which is performed by a KNN algorithm is to assign a new query point to a pre-existed group that has been created using a huge corpus of datasets. This is exactly what is required in the recommender systems to assign each user to a particular group and then provide them recommendations based on that group's preferences.

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

- **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
- **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
- **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as

well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

Principal Components Analysis

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.
   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components**
   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.

8. **Remove less or unimportant features from the new dataset.** The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

Association

Association rules allow you to establish associations amongst data objects inside large databases. This unsupervised technique is about discovering interesting relationships between variables in large databases.

For example, people that buy a new home most likely to buy new furniture.

Other Examples:

- A subgroup of cancer patients grouped by their gene expression measurements.
- Groups of shopper based on their browsing and purchasing histories.
- Movie group by the rating given by movies viewers.

Supervised vs. Unsupervised Machine Learning

Parameters	Supervised machine learning technique	Unsupervised machine learning technique
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data which is not labelled
Computational Complexity	Supervised learning is a simpler method.	Unsupervised learning is computationally complex
Accuracy	Highly accurate and trustworthy method.	Less accurate and trustworthy method.

Collaborative filtering

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.

It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user.

What is a Recommendation system?

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.

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

- **Memory-based**

Memory-based methods use user rating historical data to compute the similarity between users or items. The idea behind these methods is to define a similarity measure between users or items, and find the most similar to recommend unseen items.

- **Model-based**

Model-based CF uses machine learning algorithms to predict users' rating of unrated items.

There are many model-based CF algorithms, the most commonly used are matrix factorization models such as to applying a SVD to reconstruct the rating matrix, latent Dirichlet allocation or Markov decision process based models.

- Hybrid

These aim to combine the memory-based and the model-based approaches. One of the main drawbacks of the above methods, is that you'll find yourself having to choose between historical user rating data and user or item attributes.

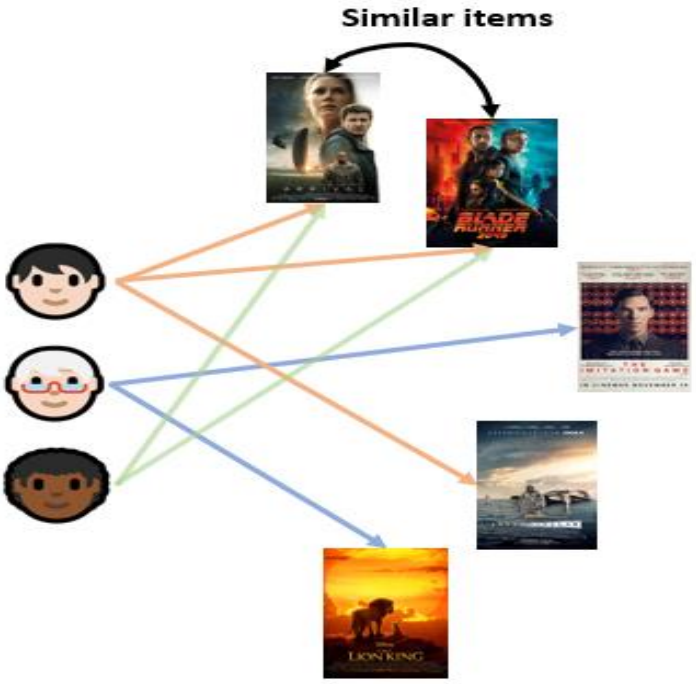
Hybrid methods enable us to leverage both, and hence tend to perform better in most cases. The most widely used methods nowadays are factorization machines.

Memory-based CF

There are 2 main types of memory-based collaborative filtering algorithms: User-Based and Item-Based. While their difference is subtle, in practice they lead to very different approaches, so it is crucial to know which is the most convenient for each case. Let's go through a quick overview of these methods:

- **Item-Based**

The idea is similar, but instead, starting from a given movie (or set of movies) we find similar movies based on other users' preferences.



Also, since a *single item is enough* to recommend other similar items, this method will not suffer from the cold-start problem.

The algorithm for **used-based CF** can be summarised as:

1. *Compute the similarity between the new user with all other users (if not already done)*
2. *Compute the mean rating of all movies of the* ***k*** *most similar users*
3. *Recommend the top* ***n*** *rated movies by other users unseen by the user*

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 to adapt to changing user preferences.

Measuring Similarity

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		

Social Media Analytics

Social media analytics is the ability to gather and find meaning in data gathered from social channels to support business decisions — and measure the performance of actions based on those decisions through social media.

Social media analytics is broader than metrics such as likes, follows, retweets, previews, clicks, and impressions gathered from individual channels. It also differs from reporting offered by services that support marketing campaigns such as LinkedIn or Google Analytics.

- Social media analytics uses specifically designed software platforms that work similarly to web search tools.
- Data about keywords or topics is retrieved through search queries or web ‘crawlers’ that span channels. Fragments of text are returned, loaded into a database, categorized and analyzed to derive meaningful insights.

Why is social media analytics important?

Social media analytics helps companies address these experiences and use them to:

- Spot trends related to offerings and brands
- Understand conversations — what is being said and how it is being received
- Derive customer sentiment towards products and services
- Gauge response to social media and other communications
- Identify high-value features for a product or service
- Uncover what competitors are saying and its effectiveness
- Map how third-party partners and channels may affect performance

Key capabilities of effective social media analytics

From there, topics or keywords can be selected and parameters such as date range can be set.

Sources also need to be specified — responses to YouTube videos, Facebook conversations, Twitter arguments, Amazon product reviews, comments from news sites.

- **Natural language processing and machine learning** technologies identify entities and relationships in unstructured data — information not pre-formatted to work with data analytics. Virtually all social media content is unstructured. These technologies are critical to deriving meaningful insights.
- **Segmentation** is a fundamental need in social media analytics. It categorizes social media participants by geography, age, gender, marital status, parental status and other demographics. It can help identify influencers in those categories. Messages, initiatives and responses can be better tuned and targeted by understanding who is interacting on key topics.
- **Behavior analysis** is used to understand the concerns of social media participants by assigning behavioral types such as user, recommender, prospective user and detractor. Understanding these roles helps develop targeted messages and responses to meet, change or deflect their perceptions.

- **Sentiment analysis** measures the tone and intent of social media comments. It typically involves natural language processing technologies to help understand entities and relationships to reveal positive, negative, neutral or ambivalent attributes.
- **Share of voice** analyzes prevalence and intensity in conversations regarding brand, products, services, reputation and more. It helps determine key issues and important topics. It also helps classify discussions as positive, negative, neutral or ambivalent.
- **Clustering analysis** can uncover hidden conversations and unexpected insights. It makes associations between keywords or phrases that appear together frequently and derives new topics, issues and opportunities. The people that make baking soda, for example, discovered new uses and opportunities using clustering analysis.
- **Dashboards and visualization** charts, graphs, tables and other presentation tools summarize and share social media analytics findings — a critical capability for communicating and acting on what has been learned. They also enable users to grasp meaning and insights more quickly and look deeper into specific findings without advanced technical skills.

Mobile Analytics

Mobile analytics involves measuring and analysing data generated by mobile platforms and properties, such as mobile sites and mobile applications. AT Internet's analytics solution lets you track, measure and understand how your mobile users are interacting with your mobile sites and mobile apps.

Why do companies use mobile analytics?

Mobile analytics gives companies unparalleled insights into the otherwise hidden lives of app users. Analytics usually comes in the form of software that integrates into companies' existing websites and apps to capture, store, and analyze the data.

This data is vitally important to marketing, sales, and product management teams who use it to make more informed decisions.

Without a mobile analytics solution, companies are left flying blind. They're unable to tell what users engage with, who those users are, what brings them to the site or app, and why they leave.

Why are mobile analytics important?

- Mobile usage surpassed that of desktop in 2015 and smartphones are fast becoming consumers' preferred portal to the internet. Consumers spend 70 percent of their media consumption and screen

time on mobile devices, and most of that time in mobile apps.

- This is a tremendous opportunity for companies to reach their consumers, but it's also a highly saturated market. There are more than 6.5 million apps in the major mobile app stores, millions of web apps, and more than a billion websites in existence.
- Companies use mobile analytics platforms to gain a competitive edge in building mobile experiences that stand out. Mobile analytics tools also give teams a much-needed edge in advertising.
- As more businesses compete for customers on mobile, teams need to understand how their ads perform in detail, and whether app users who interact with ads end up purchasing.

How do mobile analytics work?

Mobile analytics typically track:

- Page views
- Visits
- Visitors
- Source data
- Strings of actions
- Location
- Device information
- Login / logout
- Custom event data

Companies use this data to figure out what users want in order to deliver a more satisfying user experience.

For example, they're able to see:

- What draws visitors to the mobile site or app
- How long visitors typically stay
- What features visitors interact with
- Where visitors encounter problems
- What factors are correlated with outcomes like purchases
- What factors lead to higher usage and long-term retention

How different teams use mobile analytics:

- **Marketing:** Tracks campaign ROI, segments users, automates marketing
- **UX/UI:** Tracks behaviors, tests features, measures user experience
- **Product:** Tracks usage, A/B test features, debugs, sets alerts
- **Technical teams**: Track performance metrics such as app crashes

How to implement mobile analytics

Mobile analytics platforms vary widely in features and functionality. Some free applications have technical limitations and struggle with tracking users as they move between mobile websites and apps. **A top tier mobile analytics platform should be able to:**

- **Integrate easily:** With a codeless mobile feature, for instance
- **Offer a unified view of the customer:** Track data across operating systems, devices, and platforms
- **Measure user engagement:** For both standard and custom-defined events
- **Segment users:** Create cohorts based on location, device, demographics, behaviors, and more

- **Offer dashboards:** View data and surface insights with customizable reporting
- **A/B test:** Test features and messaging for performance
- **Send notifications:** Alert administrators and engage users with behavior-based messaging such as push notifications and in-app messages
- **Out-of-the-box metrics**: Insights with minimal client-side coding
- **Real-time analytics**: Proactively identify user issues
- **Reliable infrastructure**: Guaranteed uptime for consistent access to the platform

The actual installation of mobile analytics involves adding tracking code to the sites and SDKs to the mobile applications teams want to track. Most mobile analytics platforms will be set up to automatically track website visits.

Platforms with codeless mobile features will be able to automatically track certain basic features of apps such as crashes, errors, and clicks, but you'll want to expand that by manually tagging additional actions for tracking. With mobile analytics in place, you'll have deeper insights into your mobile web and app users which you can use to create competitive, world-class products and experiences.

The Challenges of Mobile Analytics

Because mobile analytics is a fairly new field of analytics and continues to change with rapidly changing consumer expectations, there are many challenges to be faced in implementing it.

Collecting the data necessary for successful mobile analytics is often the greatest challenge organizations face when attempting to understand consumer behavior on mobile devices. Many devices do not allow for cookies to track actions or do not use Javascript which can also help with website data tracking.

Another challenge to ensuring the usefulness of mobile analytics is correctly segmenting users based on their mobile behavior. Organizations must guarantee quality data and tracking so that all the necessary attributes are correctly captured and never duplicated, including network, device, entry page, time spent on page, etc. It is possible for dirty data from mobile analytics, or even algorithmic bias, to lead to incorrect conclusions. To combat this risk, businesses should put measures in place to eliminate bad data, biases, and opportunities for misrepresentation of the data.

Finally, implementing mobile analytics may require a culture shift if your company is not yet fully data driven. To receive a return on your investment

in mobile analytics, make sure to follow best practices for onboarding new technologies and gaining and maintaining buy-in from key stakeholders.

The Benefits of Mobile Analytics

Despite the challenges of mobile analytics, it's essential for modern businesses to invest in and can lead to many opportunities for the business.

- **Improved UI/UX**: Mobile analytics can help businesses develop content and campaigns that best meet their target audience's needs. With metrics on user engagement, designers can ensure a smooth, consistent user experience that is responsive to user behavior.
- **Optimized Websites and Apps**: Mobile analytics is also useful for product managers and app developers. It can be used to optimize the performance of mobile websites and apps and therefore enhance consumers' interactions with your business on mobile devices.
- **More Customer Loyalty**: Every company is always on the lookout for ways to improve customer retention. Mobile analytics can help organizations identify those areas for improvement. User retention on a business' mobile app shows strong brand loyalty and can increase the likelihood of repeat customers once downloaded. It's also a great way to analyze customer feedback and implement changes rapidly for more agile app development.
- **Analytics Insights**: Mobile analytics looks at easy-to-understand metrics and often provides user-friendly dashboards that non-technical users like marketers and product managers can quickly assess and use to improve performance. This can lead to more targeting marketing and accurate segmentation for overall improved business results.

Big Data Practice – Current Limitations

- There are hundreds of Vendors in the Big Data Space with each having its own limitations/strengths. So it becomes very hard to learn multiple software for each of the tasks.
- Also connecting these individual systems using customized connectors becomes a big challenge.
- The main deterrent is the steep learning curve behind these technologies and hence no human resources can be found for implementation projects.

Points in Favor of R

- R is open source and has a large community behind it working on and coming up with lot of innovation.
- It has close to 5000 packages and the count is increasing exponentially.
- There is a specific research community working on using R for Big Data Analytics.
- Companies like Revolutionary Analytics are coming up with innovative approaches in customizing and using R for handling massive datasets.

Points Against R

- R is a single threaded programming language with limited memory management capabilities as it uses the system memory.
- In order to use R for handling massive datasets it has to scale up both in processing and memory management capabilities.
- We will look into these aspects and some specific strategies/approaches to circumvent these issues.

Verdict – Provide a chance to R.

- We will suggest some new ways/techniques in R to handle the Big data though lot of these are in research stages and not in production yet.
- However companies and research communities are in high hopes that the current limitations will be addressed and R can be used as a single go to software offering integrated End to End capabilities in this space.

Approaches to address R current limitations

- We can spread the work to be handled in parallel by running in multiple CPU's or running in clusters.
- There are some latest packages in R which would be of interest to explore in this direction.
- We have described some of these in the next slides.

R Packages for Big Data

- **Snow**

  The package snow (an acronym for Simple Network of Workstations) provides a high-level interface for using a workstation cluster for parallel computations in R.

- **Multicore**

  This package provides functions for parallel execution of R code on machines with multiple cores or CPUs. Unlike other parallel processing methods all jobs share the full state of R when spawned, so no data or code needs to be initialized. The actual spawning is very fast as well since no new R instance needs to be started.

- **Parallel**

  Includes a parallel random number generator (RNG); important for simulations. Particularly suitable for 'single program, multiple data' (SPMD) problems.

R Packages for Big Data

- **scaleR Algorithms (Revolutionary Analytics)**
  ScaleR algorithms enable R developers to run R scripts on massive data sets at high speeds. ScaleR enables R developers to easily maximize compute capability without writing any distributed applications themselves.

- **Rhipe**
  Rhipe is a software package that allows the R user to create MapReduce jobs that work entirely within the R environment using R expressions.

- **Seague**
  We can peform simple parallel processing with a fast & easy setup on AWS Elastic Map Reduce. So its like performing the parallel computing directly on top of cloud. Segue has a simple goal: Parallel functionality in R; two lines of code; in under 15 minutes.

Sample Applications

- Social media mining with R (Ex: Mining Twitter streams)

- WebCrawling using R with the RCurl package. Also Apache Nutch will be another option.

- Social Network Analysis (SNA) with graph data.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION