UNIT 5 BIG ARCHITECTURE

Structure

5.1 INTRODUCTION

In this modern era of Information and knowledge, terabytes of data are produced by a wide variety of sources every day. These sources include social media, the data of companies including production data, customer data, financials etc.; the interactions of users, the data created by sensors and data produced by electronic devices such as mobile phones and automobiles, amongst others. This voluminous increase in data relates to the field of Big Data. This concept of "Big Data" applies to the rapidly expanding quantity of data that is being collected, and it is described primarily by having the following characteristics: volume, velocity, veracity, and variety.

The process of deriving useful information and insights from massive amounts of data is referred to as "big data analytics". It requires the application of technologically advanced tools and procedures. "Big data architecture" refers to the pattern or design that describes how this data is received from many sources, processed for further ingestion, assessed, and eventually made available to the end-user.

The building blocks of big data analytics are found in the big data architecture. In most cases, the architecture components of big data analytics consist of four logical levels or layers, as discussed below:

- Big Data Source Layer: A big data environment may manage both batch processing and real-time processing of big data sources, like data warehouses, relational database management systems, SaaS applications, and Internet of Things devices. This layer is referred to as the big data sources layer.

- The Management and Storage Layer: it is responsible for receiving data from the source, converting that data into a format that the data analytics tool can understand, and storing the data in accordance with the format in which it was received.
- Analysis Layer: This layer is where the business intelligence that was acquired from the big data storage layer is analysed.
- The Consumption Layer: it is responsible for collecting the findings from the Big Data Analysis Layer and delivering them to the Appropriate Output Layer, which is also referred to as the Business Intelligence Layer.

The architecture components of big data analytics typically include four logical layers or levels (discussed above), which perform fundamental processes as given below :

- Connecting to Data Sources: Connectors and adapters can connect to a wide range of storage systems, protocols, and networks. They can also connect to any type of data format.
- Data Governance: includes rules for privacy and security that work from the time data is taken in until it is processed, analysed, stored, and deleted.
- Systems Management: Modern big data architectures are usually built on large-scale, highly scalable clusters that are spread out over a wide area. These architectures must be constantly monitored using central management consoles.
- Protecting Quality of Service: The Quality of Service framework helps define data quality, compliance policies, and how often and how much data should be taken in.

In this Unit, we will discuss Big Data its Characteristics and Applications. We will also discuss Big Data architecture and related technologies such as Map Reduce, HDFS, Apache Hadoop and Apache YARN.

5.2 OBJECTIVES

After going through this unit, you will be able to:

- explain Big Data and its Characteristics
- compare the characteristics of Structured Semi-Structured and Unstructured data
- differentiate Big Data and Data warehouse
- describe the Distributed file system
- explain HDFS and Map Reduce
- differentiate between Apache Hadoop 1 and 2 (YARN).

5.3 BIG DATA AND CHARACTERISTICS

To define the concept of Big Data, we first revise the fundamentals of data and thereafter we will extend our discussion to the advanced concepts of Big Data. What exactly does "Data" mean? Data can be numbers or characters or symbols or any other form such as an image or signal, which can be processed by a computer or may be stored in the storage media or sent in the form of electrical impulses.

Now, we define - What is Big Data? The term "big data" refers to a collection of information that is not only extremely large in quantity but also growing at an exponential rate over the course of time. Because of the vast amount and the high level of complexity, none of the normal methods that are used for managing data can effectively store or deal with such data. Simply speaking, "big data" refers to information that is kept in exceedingly vast amounts.

It is vital to have a thorough understanding of the characteristics of big data in order to get a handle on how the concept of big data works and how it can be used. The **characteristics of big data** are:

1. Volume: The amount of information that you currently have access to is referred to as its volume. We measure the quantity of data that we have in our possession using the units of Gigabytes (GB), Terabytes (TB), Petabytes (PB), Exabytes (EB), Zettabytes (ZB), and Yottabytes (YB). According to the patterns that have been seen in the industry, it is anticipated that the quantity of data has increased exponentially.

2. Velocity: The speed at which data is generated and therefore should be processed, is referred to as the "velocity" of that process. When it comes to the efficiency of any operation involving vast amounts of data, a high velocity is a fundamental requirement. This characteristic can be broken down into its component parts, which include the rate of change, activity bursts, and the linking of incoming data sets.

3. Variety: The term "variety" relates to the numerous forms that big data might take. When we talk about diversity, we mean the various kinds of big data that are available. This is a major problem for the big data sector since it has an impact on the efficiency of analysis. It is critical that you organize your data in order to effectively manage its diversity. You get a wide range of data from a variety of sources.

4. Veracity: The veracity of data relates to its accuracy. Lack of veracity can cause significant harm to the precision of your findings; it is one of the Big Data characteristics that is considered to be of the utmost importance.

5. Value: The benefits that your organization gains from utilizing the data are referred to as "value." Are the results of big data analysis in line with your company's goals? Is it helping your organization expand in any way? In big data, this is a vital component.

6. Validity: This characteristic of Big Data relates to how valid and pertinent are the facts that are going to be used for the purpose that they were designed for?

7. Volatility: Volatility relates to the life span of big data. Some data items may be valid for a very short duration like the sentiments of people.

8. Variability: The field of big data is continually changing. In some cases, the information you obtained from one source may not be the same as what you found today. This phenomenon is known as data variability, and it has an impact on the homogeneity of your data.
9. Visualization: You may be able to express the insights that were generated by big data by using visual representations such as charts and graphs. The insights that big data specialists have to provide are increasingly being presented to audiences who are not technically oriented.

Some examples of big data are given below:

- Big Data can be illustrated by looking at the Stock Exchange, which creates approximately one terabyte of new trade data every single day.
- Social Media: According to various reports, more than 500 Gigabytes of fresh data are added to the databases of the social media website each and every day. The uploading of photos and videos, sending and receiving of messages, posting of comments, and other activities are the primary sources of this data.
- In just thirty minutes of flying time, a single jet engine is capable of producing more than ten Gigabytes of data. Because there are many thousands of flights each day, the amount of data generated can reach many Peta bytes.

A big data system is primarily an application of big data in an organization for decision support. The following four components are required for a Big Data system to function properly:

1. Ingestion (collecting and preparing the data)
2. Storage (storing the data)
3. Analysis (analyzing the data)
4. Consumption (presenting and sharing the insights)

You need a component that is responsible for collecting the data and another that is responsible for storing it. In order for your big data ecosystem to be complete, you require a system that analyses and reports the results of big data analysis.

Big data may be the future of worldwide governance and businesses. It presents businesses with a great number of opportunities for improvement. The following are some of the more important ones:

1. Improved Decision Making
2. Data-driven Customer Service
3. Efficiency Optimization
4. Real-time Decision Making

Based on these opportunities, this field of Big Data has enormous applications in a variety of fields, and they are discussed in the subsequent section i.e. section 5.4.

☞ **Check Your Progress 1**

1. What does "Big Data" mean?

……………………………………………………………………………………………………
……………………………………………………………………………………………………

2. List the components required by a Big Data system to function properly

……………………………………………………………………………………………………
……………………………………………………………………………………………………

3. Discuss the essential characteristics of Big Data.

……………………………………………………………………………………………………
……………………………………………………………………………………………………

5.4 BIG DATA APPLICATIONS

It is generally stated that Big Data is one of the most valuable and effective fuel that can power the vast IT companies of the 21st century. Big data has applications in virtually every industry. Big data enables businesses to make better use of the vast quantities of data they generate and collect from a variety of sources. There are many different applications for big data, which is why it is currently one of the skills that is most in demand. The following are some examples of important applications of big data:

- One of the industries that makes the most use of big data technology is the travel and tourism sector. It has made it possible for us to anticipate the need for travel facilities in a variety of locations, thereby improving business through dynamic pricing and a number of other factors.

- The social media sites produce a significant amount of data. Big data allows marketers to make greater use of the information provided by social media platforms, which results in improved promotional activities. It enables them to construct accurate client profiles, locate their target audience, and comprehend the criteria that are important to them.

- Big data technology is utilized widely within the financial and banking sectors. The use of big data analytics can assist financial institutions in better comprehending the behaviours of their clients on the basis of the inputs obtained from their investment behaviours, shopping habits, reasons for investing, and their personal or financial histories.

- The field of healthcare has already witnessed significant changes brought on by Big Data application implementation. Individual patients can now receive individualised medical care that is tailored to their specific needs. This is possible due to the application of predictive analytics by medical professionals and personnel in the healthcare industry.

- The use of big data in recommendation systems is another popular application of big data. Big data allows businesses to recognize patterns of client behavior in order to provide better and more individualized services to those customers.

- One of the most important industries that produce and use big data is the telecommunications and multimedia business. Every day, Zettabytes of new data are created and managed.

- In addition, the Government and the military make heavy use of big data technology. You may think of the quantity of data that the Government generates on its records; and in the military, a standard fighter jet plane needs to handle Petabytes of data while it is in the air.

- Companies are able to do predictive analysis because of the capabilities provided by big data technologies. It lets businesses to make more accurate predictions of the outcomes of processes and events, which in turn helps them reduce risk.

- Companies are able to develop insights that are more accurate because of big data. The big data is collected by them from a variety of sources, which allows them the capacity to use relevant data to generate insights that can be put into action. If a corporation has more accurate information, it will be able to make decisions that are more profitable and reduce risks.

5.5 STRUCTURED, SEMI-STRUCTURED AND UNSTRUCTURED DATA

The terms "structured," "unstructured," and "semi-structured" data are frequently brought up whenever we are having a discussion about data or analytics. These are the three varieties of data that are becoming increasingly important for many kinds of commercial applications i.e. Structured, Semi-Structured and Unstructured data. Structured data has been around for quite some time, and even today, conventional systems and reporting continue to rely on this type of data. In spite of this, there has been a rapid increase in the production of unstructured and semi-structured data sources over the course of the last several years. As a consequence of this, an increasing number of companies are aiming to include all three kinds of data in their business intelligence and analytics systems in order to take these systems to the next level.

In this section of unit, we are going to discuss data classification used for understanding and implementing the Big Data i.e. Structured, Semi Structured and Unstructured data.

Structured Data: Structured data is a term that refers to information that has been reformatted and reorganised in accordance with a data model that has been decided in advance. After mapping the raw data into the predesigned fields, the data can then be extracted and read using SQL in a straightforward manner. Relational databases, which are characterised by their organisation of data into tables comprising of rows and columns, and the query language supported by them (SQL) provide the clearest example possible of structured data.

This relational model of the data format decreases the quantity of information that is duplicated. On the other side, organized data is more dependent on one another and is less flexible than unstructured data. Both humans and machines are responsible for the generation of this kind of data.

There are several examples of structured data that are generated by machines, such as data from point-of-sale terminals (POS), such as quantity, barcodes, and statistics from blogs. In a similar vein, anybody who works with data has probably used spreadsheets at least once in their life. Spreadsheets are an example of a classic form of structured data that is generated by humans. Because of the way it is organized, structured data is simpler to examine compared to semi-structured data as well as unstructured data. Structured data can be analysed easily because it corresponds to data models that have already been defined. For example, you can organise structured data such as names of customers in alphabetical order, organise telephone numbers in the appropriate format, organise social security numbers in the correct format etc.

Unstructured data: Information that is displayed in its most unprocessed form is referred to as "unstructured data". It is really difficult to work with this data because it has a minimal structure, and the formatting is also very confusing. Unstructured data management can collect data from a variety of sources, such as postings on social media platforms, conversations, satellite imagery, data from the Internet of Things (IoT) sensor devices, emails, and presentations, and organise it in a storage system in a logical and specified manner.

Semi-Structured Data: There is a third kind of data that falls between structured and unstructured data called semi-structured data or partially structured data. Your data sets might not always be structured or unstructured. One variety of such data is known as semi-structured data, and it differs from unstructured data in that it possesses both consistent and definite qualities. It does not restrict itself to a fixed structure like those that are required for relational databases. Although organizational qualities such as metadata or semantics tags are utilised with semi-structured data in order to make it more manageable, there is still a certain amount of unpredictability and inconsistency present in the data.

Use of delimited files is one illustration of a data format that is only semi-structured. It has elements that are capable of separating the hierarchies of the data into their own distinct structures. In a similar manner, digital images have certain structural qualities that make them semi-structured, but the image itself does not have a pre-defined structure. If a picture is shot with a Smartphone, for instance, it will include certain structured properties like a device ID, a geotag, and a date and time stamp. After they have been saved, pictures can be organized further by affixing tags to them, such as "pet" or "dog," for example.

Due to the presence of one or more characteristics that allow for classification, unstructured data may on occasion be categorized as semi-structured data instead. To summarise, organisations need to analyse all three kinds of data in order to stay ahead of the competition and make the most of the knowledge they have.

☞ **Check Your Progress 2**

1. Give Five applications of Big Data.

………
………

2. Compare Structured, Semi-Structured and Unstructured Data.

………
………

3. Discuss the role of Structured, Semi-Structured and Unstructured Data in decision-making.

………
………

5.6 BIG DATA VS DATA WAREHOUSE

What should you buy for implementing an analytics system for your organization - a big data solution or a data warehouse? A data warehouse and a big data solution are quite similar in many respects. Both have the capacity to store a significant amount of information. When reporting, either option is acceptable to utilize. But the question that needs to be answered is whether they can truly be utilised as a substitute for each other. In order to understand this, you need to have a conceptual understanding of both i.e., Big data and Data Warehouse.

The form of big data that can be found in Hadoop, Cloudera, and other similar big data platforms is the type that is understood by most people. The following are the working definitions of big data solutions (You may find an accurate functioning definition in the websites of Cloudera or HortonWorks.):

- A type of technology that is able to store extremely vast volumes of data.
- A technology that is capable of storing the data in devices that are not pricey.
- A type of data storage technology in which the data is kept in an unstructured manner.

Data warehouse has had widespread usage in the past two decades, whereas big data applications have seen significant rise over the course of the past decade. Big data are expected to eventually replace traditional data warehousing, which is the first thing that comes to the mind of someone who is not technically very deep into these technologies. The fact that they share similarities is another argument in favour of this simplistic way of thinking. Following are some of the similarities between the two:

- Both of these have a significant amount of information.
- Both may be utilised in the reporting process.
- Both are controlled by devices that save information electronically.

But despite this, "Big Data" and "Data Warehouse" are not the same thing at all. Why? In order to understand this, we need to recapitulate the basic concepts of Data Warehouse

A data warehouse is a subject-oriented, non-volatile, integrated, and time-variant collection of data that is generated for the aim of the management making decisions with such data. A data warehouse provides a centralized, integrated, granular, and historical point of reference for all of the company's data in one convenient location.

So why do individuals demand a solution that involves big data? People are looking for a solution for big data because there is a significant amount of data in many organizations. And in such companies, that data – if it is accessed properly – can include a great deal of important information that can lead to better decisions, which in turn can lead to more revenue, more profitability, and more customers. And the vast majority of businesses want to do this.

What are the reasons that people require a data warehouse? In order for people to make decisions based on accurate information, a data warehouse is required. You need data that is trustworthy, verifiable, and easily accessible to everyone if you want to have a thorough understanding of what is occurring within your company.

So, in short we can understand the data Warehouse involves a process of collecting data from a number of sources, processing it, and storing it in a repository where it can be analyzed and used for reporting reasons. The procedure described above results in the establishment of a data repository that is known as the data warehouse.

Comparison: Big Data Vs Data Warehouse
Therefore, what do we find when we investigate a data warehouse in conjunction with a big data solution? A big data solution is an example of a form of technology, whereas data warehousing is an example of a type of architectural framework. These two parts of the same thing could not be more different from one another. To put it another way, technology can be loosely defined as any mechanism that is capable of storing and organising huge amounts of data. A facility that stores and organises data with the aim of guaranteeing a company's credibility and integrity is known as a data warehouse. Data warehouses are becoming increasingly common. When data is taken from a data warehouse, the person taking the data is aware that other individuals are using the same data for various purposes. This is because different people

are taking the data. The availability of a data warehouse ensures the availability of a foundation that can enable the reconcilability of data.

S.No.	Big Data	Data Warehouse
1.	Big data refers to data that is stored in an extremely large format that can be utilised by many technologies.	The term "data warehouse" refers to the accumulation of historical data from a variety of business processes within an organisation.
2.	The term "big data" refers to a type of technology that can store and manage extensive amounts of data.	A data warehouse is a structure that is utilised in the process of data organisation.
3.	As input, it will accept data that is either structured, non-structured, or semi-structured.	The only type of data that can be used as input is structured data.
4.	The distributed file system is used for processing big data.	Processing operations in the data warehouse are not carried out using a distributed file system.
5.	When it comes to retrieving data from databases, big data may not use SQL queries.	To get data from relational databases, we use structured query language, or SQL, in the data warehouse.
6.	Apache Hadoop is capable of managing massive amounts of data when handled appropriately.	It is difficult for a data warehouse to manage an extremely large volume of unstructured data.
7.	The modifications to the data that occur as a result of adding new data are saved in the form of a file. which is represented by a table.	Changes in the data that occur as a result of adding new data do not have an immediate and direct effect on the data warehouse.
8.	When opposed to data warehouses, big data does not require the same level of effective management strategies.	As a result of the fact that the data is compiled from a variety of business divisions, the data warehouse calls for management strategies that are more effective.

Big Data uses Distributed File System. So, we extend our discussion on Distributed file Systems in section 5.7 of this unit.

☞ Check Your Progress 3

1. Give similarities between Big Data and Data warehouse.

……………………………………………………………………………………………………
……………………………………………………………………………………………………

2. Compare Big Data and Data warehouse.

……………………………………………………………………………………………………
……………………………………………………………………………………………………

5.7 DISTRIBUTED FILE SYSTEM

A Distributed File System (DFS) is a file system that is distributed across a large number of file servers may be at several locations (It's possible that these servers are located in various parts of the world). It allows programmers to access or store isolated files the same way they do local files, allowing programmers to access files from any network or computer.

The primary objective of the Distributed File System (DFS), is to facilitate the sharing of users' data and resources between physically separate systems through the utilization of a Common File System. A setup on Distributed File System is defined as a group of workstations and mainframes that are linked together over a Local Area Network (LAN). Within the framework of the operating system, a DFS is carried out. A namespace is generated by DFS, and the clients are not exposed to the inner workings of this generation process.

The DFS is comprised of two distinct components, given below:

- **Location transparency**: The namespace component is a useful tool for achieving Location transparency.

- **Redundancy:** It is established by the utilization of a file replication component.

Together, these components make it more likely that data will be accessible in the event that there is a breakdown or a severe load. They accomplish this by making it possible for data that is stored in multiple locations to be logically grouped together and shared under a single folder that is referred to as the "DFS root."

It is possible to use the namespace component of Distributed File System (DFS) without using the file replication component, and it is perfectly possible to use the file replication component of Distributed File System (DFS) between servers without using the namespace component. It is not essential to make use of both of the DFS components at the same time.

In order to gain more clarity on the Distributed File System (DFS), we need to study the Features of DFS:

1. **Transparency:**
   When discussing distributed systems, the term "transparency" refers to the process of hiding information about the separation of components from both the user and the application programmer. This is done in order to protect the integrity of the system. Because of this, it seems as though the whole system is a single thing, rather than a collection of distinct components working together. It is classified into the following types:

a) **Structure Transparency –**
   There is no reason for the client to be aware of the number of file servers and storage devices and their geographical locations. But it is always recommended to provide a number of different file servers to improve performance, adaptability, and dependability.
b) **Access Transparency –**
   The access process for locally stored files and remotely stored files should be identical. It should be possible for the file system to automatically locate the file that is being accessed and send that information to the client's side.
c) **Naming transparency –**
   There should be no indication in the name of the file as to where the file is located in any way, shape, or form. Once a name has been assigned to the file, it should not have any changes made to it while it is being moved from one node to another.
d) **Replication transparency –**
   If a file is copied on multiple nodes, the locations of the copies of the file as well as the copies themselves should be hidden when moving from one node to another.

2. **User Mobility**: It will automatically transfer the user's home directory to the node where the user logs in.

3. **Performance:** The average amount of time it takes to get what the client wants is used to measure performance. This time includes CPU time, the time it takes to get to secondary storage, and the time it takes to get to the network. It would be best for the Distributed File System to work like both DFS and as well as a centralized file system (CFS), based on requirements.

4. **Simplicity and ease of use:** A file system should have a simple user interface and a small number of commands in the file.

5. **High availability:** If a link fails, a node fails, or a storage drive crashes, a Distributed File System should still be able to keep working. A distributed file system (DFS) that is both reliable and flexible should have different file servers that control different storage devices that are also separate.

6. **Scalability:** Adding new machines to the network or joining two networks together is a common way to make the network bigger, so the distributed system will always grow over time. So, a good distributed file system (DFS) should be built so that it can grow quickly as the number of nodes and users grows. As the number of nodes and users grows, the service should not deteriorate too much.

7. **High reliability:** A good distributed file system (DFS) should make it as unlikely as possible that data will be lost. That is, users should not feel like they have to make backup copies of their files because the system is not reliable. Instead, a file system should make copies of important files in case the originals get lost. Stable storage is used by many file systems to make them very reliable.

8. **Data integrity:** A file system is often used by more than one person at a time. The file system must make sure that the data saved in a shared file stays the same. That is, a concurrency control method must be used to keep track of all the different users' requests to access the same file at the same time. Atomic transactions are a type of high-level concurrency management that a file system often offers to users to keep their data safe.

9. **Security:** A distributed file system should be safe so that its users can trust that their data will be kept private. Security measures must be put in place to protect the information in the file system from unwanted and unauthorized access.

10. **Heterogeneity:** Because distributed systems are so big, there is no way to avoid heterogeneity. Users of heterogeneous distributed systems can choose to use different kinds of computers for different tasks.

The working of DFS can be put into practice in one of two different ways:

- **Standalone DFS namespace** is a namespace that only permits DFS roots to be used in it if they are located on the local computer and do not make use of Active Directory. Only the computers on which a Standalone DFS was initially created can be used to acquire the file system. There is no fault liberation offered by it, and it cannot be connected to any other DFS. Standalone DFS roots are not very common because of their limited advantage.

- **Domain-based DFS namespace** — This creates the DFS namespace root, which can be accessed at domainname>dfsroot> and stores the configuration of DFS in Active Directory.

Advantages of Distributed File System (DFS) :

a) DFS permits many active users to access or store data, simultaneously.

b) DFS allows the remote sharing of data

c) DFS facilitates file searching, faster to get to the data, and makes the network to run better.

d) DFS improves the ability to change the size of the data and the ability to exchange data.

e) DFS maintains data transparency and replicability, even if a server or disc fails, Distributed File System makes data available.

Disadvantages of Distributed File System (DFS) :

a) Because all the nodes and connections in a Distributed File System need to be secured, we can safely claim that security is at risk.

b) During the process of moving from one node to another in the network, there is a chance that some of the messages and data will be lost.

c) While using a distributed file system, Difficulty is encountered when attempting to connect to a database

d) When compared to a system with a single user, a Distributed File System makes database management more difficult.
e) If every node in the network attempts to transfer data at the same time, there is a possibility that the network will become overloaded.

After understanding the Distributed File System (DFS), now it is time to understand Hadoop and HDFS (Hadoop Distributed File System). The discussion for HDFS (Hadoop Distributed File System) and MapReduce is given in the subsequent section, and the details of MapReduce are explicitly available in Unit-6.

☞ **Check Your Progress 4**

1. Describe the term "Distributed File System" in the context of Big Data.

……………………………………………………………………………………………………
……………………………………………………………………………………………………

2. What is the primary objective of Distributed File System?

……………………………………………………………………………………………………
……………………………………………………………………………………………………

3. Explain the components of Distributed File System.

……………………………………………………………………………………………………
……………………………………………………………………………………………………

4. Discuss the features of Distributed File System

……………………………………………………………………………………………………
……………………………………………………………………………………………………

5. Discuss the number of ways through which the working of Distributed File System can be put into practice

……………………………………………………………………………………………………
……………………………………………………………………………………………………

6. Give advantages and disadvantages of Distributed File Systems.

……………………………………………………………………………………………………
……………………………………………………………………………………………………

5.8 HDFS AND MAPREDUCE

Data is being produced at a lightning speed in the modern period by a wide variety of sources, such as corporations, scientific data, e-mails, blogs, and other online platforms, amongst other places. It is necessary to implement data-intensive applications and storage clusters in order to analyse and manage the massive amount of data that is being generated and to gain information that is of use to the users. This is a prerequisite for being able to analyse and manage the data. This category of application is required to fulfil a number of characteristics, such as being able to tolerate errors, performing parallel processing, distributing data, maintaining load balance, being scalable, and having highly available operation. The MapReduce programming approach was developed by Google specifically for the purpose of resolving problems of this kind. Hadoop is another name for the open-source software project known as Apache Hadoop, which implements MapReduce technology.

Hadoop is a collection of several software services that are all freely accessible to the public and can be used in conjunction with one another. It offers a software framework for storing a huge amount of data in a variety of locations and for working with that data by utilising the MapReduce programming style. Both of these capabilities are accomplished through the use of software. Hadoop's essential components are the Hadoop Distributed File System (HDFS) and the MapReduce programming paradigm. Hadoop is operated using a programming style called MapReduce, and the Hadoop Distributed File System (HDFS) is the file system that is used to store data. The combination of HDFS and MapReduce creates an architecture that, in addition to being scalable and fault-tolerant, conceals all of the complexity associated with the analysis of big data.

Hadoop Distributed File System, abbreviated as HDFS, is a self-healing, distributed file system that offers dependable, scalable, and fault-tolerant data storage on commodity hardware. HDFS was developed by Hadoop. It collaborates closely with MapReduce by distributing storage and processing across huge clusters. This is accomplished by combining storage resources that, depending on the requests and queries being processed, are able to scale up or down. HDFS is not architecture-specific; it may read data in any format, including text, photos, videos, and so on, and it will automatically optimise it for high bandwidth streaming. The ability to tolerate errors is the most significant benefit of HDFS. The possibility of a catastrophic failure can be reduced by ensuring that there is a quick data movement between the nodes and that Hadoop can continue to offer service even in the event that individual nodes fail.

The information presented in this section can be used for understanding the development of large-scale distributed applications that can make use of the computing capacity of several nodes in order to finish tasks that are data and computation intensive.

Let us discuss the Hadoop Architecture which includes the following components:

- the MapReduce programming framework and
- the Hadoop distributed file system (HDFS)

An understanding of the Hadoop architecture requires the introduction of various daemons involved in its working. So, we need to understand the various daemons involved in Apache Hadoop, which includes the five daemons, three of which i.e. NameNode, the DataNode, and the Secondary NameNode relates to **HDFS** for the purpose of efficiently managing distributed storage, and The JobTracker and TaskTracker components that are utilized by the **MapReduce** engine are responsible for both job tracking and job execution respectively, and each of the mentioned daemons runs on their respective JVM

Firstly, we will discuss the **HDFS**, a distributed file system that is comprised of three nodes i.e. the NameNode, the DataNode, and the Secondary NameNode respectively.

1) NameNode : A single NameNode daemon operates on the master node. NameNode is responsible for storing and managing the metadata that is connected with the file system. This metadata is stored in a file that is known as fsimage. When a client makes a request to read from or write to a file, the metadata is held in a cache that is located within the main memory so that the client may access it more rapidly. The I/O tasks are completed by the slave DataNode daemons, which are directed in their actions by the NameNode.

   The NameNode is responsible for managing and directing how files are divided up into blocks, selecting which slave node should store these blocks, and monitoring the overall health and fitness of the distributed file system. In addition, it decides which slave node should store these blocks. Memory and input/output (I/O) are both put to intensive use in the operations that are carried out by the NameNode in the network.

2) DataNode : A DataNode daemon is present on each slave node, which is a component of the Hadoop cluster. DataNodes are the primary storage parts of HDFS. They are responsible for storing data blocks and catering to requests to read or write files that are stored on HDFS. These are under the authority of NameNode. Blocks that are kept in DataNodes are replicated in accordance with the configuration in order to guarantee both high availability and reliability. These duplicated blocks are dispersed around the cluster so that computation may take place more quickly.

3) Secondary NameNode : A backup for the NameNode is not provided by the Secondary NameNode. The job of the Secondary NameNode is to read the file system at regular intervals, log any changes that have occurred, and then apply those changes to the fsimage file. This assists in updating NameNode so that it can start up more quickly the next time, as shown in Figure 3.

The HDFS layer is responsible for daemons that store data and information, such as NameNode and DataNode. The MapReduce layer is in charge of JobTracker and TaskTracker, which are seen in Figure 1 and are responsible for keeping track of and actually executing jobs.

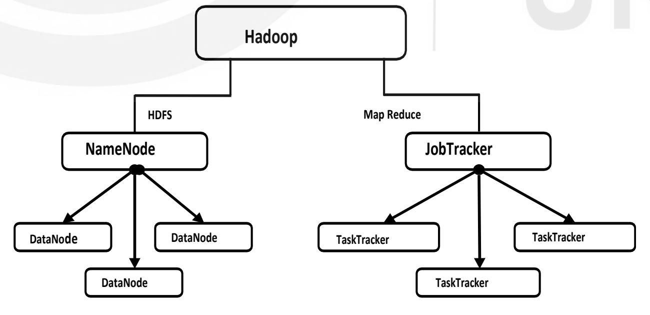


Figure-1-Hadoop Daemons

The master/slave architecture is utilised by HDFS, with the NameNode daemon and secondary NameNode both operating on the master node and the DataNode daemon running on each and every slave node, as depicted in Figure 2. The HDFS storage layer consists of three different daemons.

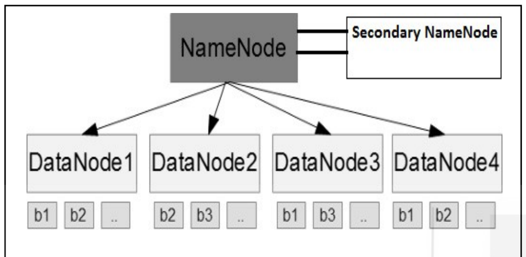


Figure 2 –HDFS Architecture

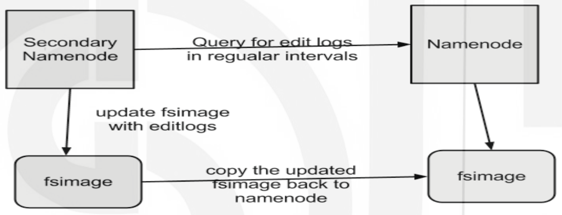

Secondary Namenode
Query for edit logs in regualar intervals
Namenode
update fsimage with editlogs
fsimage
copy the updated fsimage back to namenode
fsimage

Figure – 3 : Task of Secondary name Node

A Hadoop cluster consists of several slave nodes, in addition to a single master node. NameNode is the master daemon for the HDFS storage layer, while JobTracker is the master daemon for the MapReduce processing layer. Both of these daemons are executed by the master node. The remaining machines will be responsible for running the "slave" daemons, which include DataNode for the HDFS layer and TaskTracker for the MapReduce layer.

It is possible for a master node to take on the role of a slave at times, as it possesses the potential to flip roles. As a result, the master node is capable of operating both the master daemons and the slave daemons in addition to running the master daemons. The daemons that are operating on the master node are basically responsible for coordinating and administering the daemons that are running as slaves on all of the other nodes. This responsibility lies with the master node. These slave daemons are responsible for carrying out the tasks essential for the processing and storing of data.

Now we will discuss the MapReduce concept and the daemons involved with MapReduce i.e. the JobTracker and TaskTracker components that are utilized by the **MapReduce** engine.

MapReduce can refer to either a programming methodology or a software framework. Both are utilised in Apache Hadoop. Hadoop MapReduce is a programming framework that is made available for creating applications that can process and analyse massive data sets in parallel on large multi-node clusters of commodity hardware in a manner that is scalable, reliable, and fault tolerant. The processing and analysis of data consist of two distinct stages known as the Map phase and the Reduce phase.

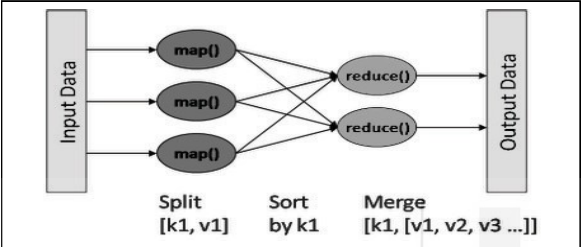


Figure 4 : Map and Reduce Phase

During a MapReduce task, the input data is often segmented and partitioned into pieces before being broken up and processed in parallel by the "Map phase" and then by the "Reduce phase." The data that is generated by the Map phase is arranged and organised by the Hadoop architecture.

The result of the Map phase is sorted by the Hadoop framework, and this information is then sent as input to the Reduce phase in order to begin parallel reduce jobs (see Figure 4). These input and output files are saved in the system's file directory. The HDFS file system is the source of input datasets for the MapReduce framework by default. It is not required that tasks involving Map and Reduce proceed in a sequential way. This means that reduce jobs can begin as soon as any of the Map tasks finishes the work that has been given to them. It is also not required that all Map jobs be finished before any reduction tasks begin their work. MapReduce operates on key-value pairs as its data structure. In theory, a MapReduce job will accept a data set as an input in the form of a key-value pair, and it will only produce output in the form of a key-value pair after processing the data set through MapReduce stages. As can be seen in Figure 5, the output of the Map phase, which is referred to as the intermediate results, is sent on to the Reduce phase as an input.

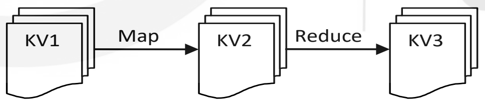


Fig 5: MapReduce key-value pairs

On the same lines as HDFS, MapReduce also makes use of a master/slave architecture. As illustrated in Figure 6, the JobTracker daemon resides on the master node, while the TaskTracker daemon resides on each of the slave nodes.

The MapReduce processing layer consists of two different daemons i.e. JobTracker and TaskTracker, their role are discussed below:

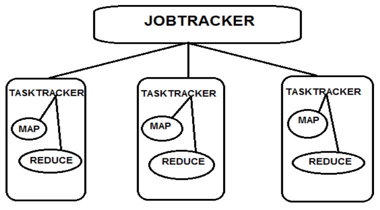


Fig 6: JobTracker and TaskTracker

4) JobTracker: The JobTracker service is responsible for monitoring MapReduce tasks that are carried out on slave nodes and is hosted on the master node. The job is sent to the JobTracker by the user through their interaction with the Master node. The next thing that happens is that JobTracker queries NameNode to find out the precise location of the data in HDFS that needs to be processed. JobTracker searches for TaskTracker on slave nodes and then sends the jobs to be processed to TaskTracker on those nodes. The TaskTracker will occasionally send a heartbeat message back to the JobTracker to verify that the TaskTracker on a specific slave node is still functioning and working on the task that has been assigned to it. If the heartbeat message is not received within the allotted amount of time, the TaskTracker running on that particular slave node is deemed to be inoperable, and the work that was assigned to it is moved to another TaskTracker to be scheduled. The combination of JobTracker and TaskTracker is referred to as the MapReduce engine. If there is a problem with the JobTracker component of the Hadoop MapReduce service, all active jobs will be halted until the problem is resolved.

5)TaskTracker: On each slave node that makes up a cluster, a TaskTracker daemon is executed. It works to complete MapReduce tasks after accepting jobs from the JobTracker. The capabilities of a node determine the total amount of "task slots" that are available in each TaskTracker. Through the use of the heartbeat protocol, JobTracker is able to determine the number of "task slots" that are accessible in TaskTracker on a slave node. It is the responsibility of JobTracker to assign suitable work to the relevant TaskTrackers, and the number of open task slots will determine how many jobs can be assigned. On every slave node, TaskTracker is the master controller of how each MapReduce action is carried out. Even though there is only one TaskTracker for each slave node, each TaskTracker has the ability to start multiple JVMs so that MapReduce tasks can be completed simultaneously. JobTracker, the master node, receives a "heartbeat" message regularly from each slave node's TaskTracker. This message confirms to JobTracker that TaskTracker is still functioning.

In short, it can be said that, when it comes to processing massive and unstructured data volumes, the Hadoop MapReduce computing paradigm and HDFS are becoming increasingly popular choices. While masking the complexity of deploying, configuring, and executing the software components in the public or private cloud, Hadoop makes it possible to interface with the MapReduce programming model. Users are able to establish clusters of commodity servers with the help of Hadoop. MapReduce has been modelled as an independent platform-as-a-service layer that cloud providers can utilise to meet a variety of different requirements. Users are also given the ability to comprehend the data processing and analysis processes.

5.9 APACHE HADOOP 1 AND 2 (YARN)

Apache Hadoop 1.x suffered from a number of architectural flaws, the most notable of which was a decline in the overall performance of the system. Actually, the cause of the problem was the excessive strain that was placed on the MapReduce component of Hadoop-1. MapReduce was responsible for both application management and resource management in Hadoop 1; however, with Hadoop 2, application management is now handled by a new component known as YARN, which takes over responsibility for resource management (yet another resource negotiator). As a result, MapReduce is in charge of managing application management in Hadoop 2, whereas YARN is in charge of managing the resources. With the release of Hadoop 2, YARN has added two additional daemons. These are-

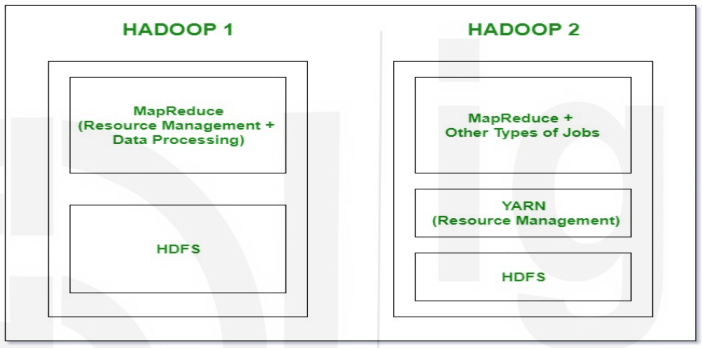
- Resource Manager
- Node Manager

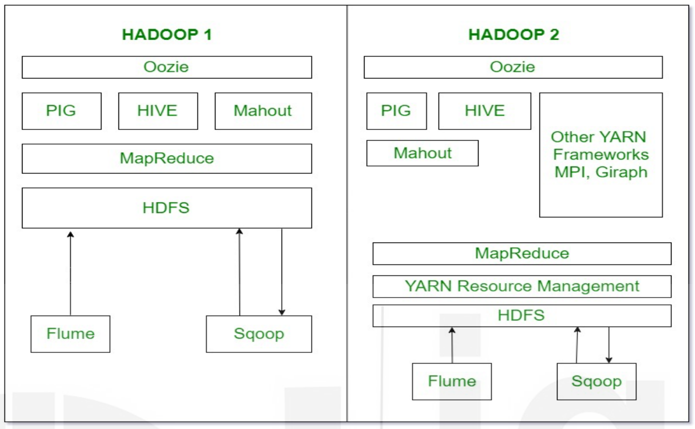
These two new Hadoop 2 daemons have replaced the JobTracker and TaskTracker in Hadoop 1.

In addition, there is only one NameNode in the Hadoop 1.x cluster, which implies that it serves as the single point of failure for the entire system. Hadoop 2. x's architecture, on the other hand, incorporates both Active and Passive NameNodes in several places. In the event that the active NameNode is unable to complete the tasks assigned to it, the passive NameNode will step in and assume responsibility. This element is directly responsible for Hadoop 2. x's high availability, which is one of the most important characteristics of the new version.
The processing of data is problematic in Hadoop 1.x; however, Hadoop 2.x's YARN provides a centralised resource management that allows sharing of sharable resources. This makes it possible for multiple applications included within Hadoop to execute concurrently while sharing a single resource.

The comparison between Apache Hadoop-1 and Hadoop-2 is consolidated below:

COMPARISON PARAMETER	**Apache Hadoop 1**	**Apache Hadoop 2**
Components	Has MapReduce	Has YARN(Yet Another Resource Negotiator) and MapReduce version 2.
Daemons	1) NameNode, 2) DataNode, 3) Secondary NameNode, 4) JobTracker, 5) TaskTracker	1) NameNode, 2) DataNode, 3) Secondary NameNode, 4) Resource Manager, 5) Node Manager

Working	Both HDFS and MapReduce are the components of Hadoop 1 HDFS is responsible for data storage, and MapReduce, which sits above HDFS, manages resources and also perform data processing, as a result of this the system performance is heavily impacted.	The Hadoop Distributed File System, or HDFS, is utilised once again for storage in Hadoop 2, while YARN, a Resource Management system, is layered over HDFS to perform its duties. In essence, it distributes the available resources and ensures that everything continues to function normally. YARN actually shares the load of MapReduce. Hence the system performance improves
	 HADOOP 1 MapReduce (Resource Management + Data Processing) HDFS HADOOP 2 MapReduce + Other Types of Jobs YARN (Resource Management) HDFS	
Limitation	The architecture of Hadoop 1 is known as a Master-Slave architecture. One master rules over a large number of slaves in this arrangement. In the event that the master node experienced a catastrophic failure, the cluster would be wiped out regardless of the quality of the slave nodes. Again, in order to create that cluster, you have to copy system files, picture files, and so on onto another machine, which takes an excessive amount of time and is something that enterprises just cannot accept.	Hadoop 2 utilises the same Master-Slave architecture as its predecessor. However, this is made up of a number of different masters (also known as active namenodes and standby namenodes) and a number of different slaves. In the event that this master node suffers a crash, the standby master node will take control of the network. You are able to create a wide variety of different active-standby node combinations. As a result, the issue of having a single point of failure will be resolved with Hadoop 2.

EcoSystem	 HADOOP 1 Oozie PIG HIVE Mahout MapReduce HDFS Flume Sqoop HADOOP 2 Oozie PIG HIVE Mahout Other YARN Frameworks MPI, Giraph MapReduce YARN Resource Management HDFS Flume Sqoop Oozie is a simplified version of a workflow scheduler. It is responsible for determining the specific times at which jobs will run based on the dependencies between them. Pig, Hive, and Mahout are examples of data processing tools that sit above Hadoop and execute their job there. Sqoop is an application that can import and export structured data. Using a SQL database, it is possible to immediately import and export data into HDFS. Flume is a tool that is used to import and export streaming data as well as unstructured data.	
Support	No Support of Microsoft Windows available for Hadoop 1.x	Support of Microsoft Windows available for Hadoop 2.x

☞ **Check Your Progress 5**

1. What is Hadoop? Discuss the components of Hadoop.

……………………………………………………………………………………………………

……………………………………………………………………………………………………

2. Discuss the role of HDFS and MapReduce in Hadoop Architecture

……………………………………………………………………………………………………

……………………………………………………………………………………………………

3. Discuss the relevance of various nodes in HDFS architecture

……………………………………………………………………………………………………

……………………………………………………………………………………………………

4. Explain, how master/slave process works in HDFS architecture

……………………………………………………………………………………………………

……………………………………………………………………………………………………

5. What do you understand by the Map phase and reduce phase in MapReduce architecture?

…………………………………………………………………………………………………
…………………………………………………………………………………………………

6. List and explain the various daemons involved in the functioning of MapReduce.

…………………………………………………………………………………………………
…………………………………………………………………………………………………

7. Differentiate between Apache Hadoop-1 and Hadoop-2.

…………………………………………………………………………………………………
…………………………………………………………………………………………………

5.10 SUMMARY

The unit covers the concepts necessary for the understanding of Big Data and its respective Characteristics, none the less the understanding of the concept of Big data was also covered from the applications point of view. The unit also covers further details like the comparison between Structured, Semi-structured and Unstructured data; also, a comparison of Big Data and Data warehouse is presented in this unit. This Unit also covers the important concept and understanding of Distributed file systems used in Big Data. Finally, the unit concludes with the comparative presentation of Map Reduce and HDFS, along with Apache Hadoop 1 and 2 (YARN) .

5.11 SOLUTIONS/ANSWERS

☞ **Check Your Progress 1**

1. What exactly does "Big Data" mean?
   Refer to section 5.3
2. List the components required by Big Data system to function properly
   Refer to section 5.3
3. Discuss the essential characteristics of Big Data
   Refer to section 5.3

☞ **Check Your Progress 2**

1. Give Five applications of Big Data
   Refer to section 5.4
2. Compare Structured, Semi-Structured and Unstructured Data
   Refer to section 5.5
3. Discuss the role of Structured, Semi-Structured and Unstructured Data in decision making
   Refer to section 5.5

Commented [AK1]: Please given short answers

☞ **Check Your Progress 3**

1. Give similarities between Big Data and Data warehouse
   Refer to section 5.6

2. Compare Big Data and Data warehouse
   Refer to section 5.6

☞ **Check Your Progress 4**

1. Describe the term "Distributed File System" in context of Big Data.
   Refer to section 5.7
2. What is the primary objective of Distributed File System
   Refer to section 5.7
3. Explain the components of Distributed File System
   Refer to section 5.7
4. Discuss the features of Distributed File System
   Refer to section 5.7
5. Discuss the number of ways through which the working of Distributed File System can be put into practice
   Refer to section 5.7
6. Give advantages and disadvantages of Distributed File System
   Refer to section 5.7

☞ **Check Your Progress 5**

1. What is Hadoop? Discuss the components of Hadoop.
   Refer to section 5.8
2. Discuss the role of HDFS and MapReduce in Hadoop Architecture
   Refer to section 5.8
3. Discuss the relevance of various nodes in HDFS architecture
   Refer to section 5.8
4. Explain, how master/slave process works in HDFS architecture
   Refer to section 5.8
5. What do you understand by the Map phase and reduce phase in MapReduce architecture
   Refer to section 5.8
6. List and explain the various daemons involved in the functioning of MapReduce
   Refer to section 5.8
7. Differentiate between Apache Hadoop-1 and Hadoop-2
   Refer to section 5.9

5.12 FURTHER READINGS

- C. Lam, "Introducing Hadoop", in Hadoop in Action, MANNING, 2011.
- D. Borthakur, "The hadoop distributed file system: architecture and design," Hadoop Project Website [online]. Available: http://hadoop.apache.org/ core/docs/current/hdfs design.pdf