1 UNIT

# Introduction to Big Data

## CONTENTS

## PART-1

*Types of Digital Data, History of Big Data Innovation.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 1.1. Write short note on digital data.**

**Answer**

1. Digital data is data that represents other forms of data using specific machine language systems that can be interpreted by various technologies.
2. One of the biggest strengths of digital data is that all sorts of complex analog input can be represented with the binary system.
3. For example, digital data is used is in cellphones or in MP3 players, digital thermometers and blood pressure meters as well as digital bathroom scales which give discrete but fast readings.
4. Numbers, text and other characters and symbols are naturally in a digital form.
5. Music, movies, and games can also be stored as what are ultimately sequences of 0's and 1's being interpreted by a computer.

**Que 1.2. Explain different types of digital data.**

**Answer**

Following are the different types of digital data :

1. **Structured digital data :**
   i. Structured digital data is data whose elements are addressable for effective analysis. It has been organized into a formatted repository that is typically a database.
   ii. It concerns all data which can be stored in database SQL in a table with rows and columns.



   iii. They have relational keys and can easily be mapped into pre-designed fields.
   iv. For example, Relational data.
2. **Semi-structured digital data :**
   i. Semi-structured digital data is information that does not reside in a relational database but that have some organizational properties that make it easier to analyze.
   ii. With some process, we can store them in the relation database, but semi-structured exist to ease space.
   iii. For example, XML data.
3. **Unstructured digital data :**
   i. Unstructured digital data is a data which is not organized in a predefined manner or does not have a predefined data model, thus it is not a good fit for a mainstream relational database.
   ii. So, for unstructured data, there are alternative platforms for storing and managing, it is increasingly prevalent in IT systems and is used by organizations in a variety of business intelligence and analytics applications.
   iii. For example, Word, PDF, Text, Media logs.

**Que 1.3. Differentiate between structured, unstructured and semi-structured data.**

| No. | Properties | Structured digital data | Unstructured digital data | Semi-structured digital data |
|---|---|---|---|---|
| 1. | Technology | It is based on Relational database table. | It is based on XML/RDF (Resource Description Framework). | It is based on character and binary data. |
| 2. | Transaction management | Matured transaction and various concurrency techniques. | No transaction management and no concurrency. | Transaction is adapted from DBMS not matured. |
| 3. | Version management | Versioning over tuples, row, tables. | Versioning over tuples or graph is possible. | Versioned as a whole. |
| 4. | Flexibility | It is schema dependent and less flexible. | It is more flexible and there is absence of schema. | It is more flexible than structured data but less than unstructured data. |
| 5. | Scalability | It is very difficult to scale DB schema. | It's scaling is simpler than structured data. | It is more scalable. |
| 6. | Robustness | Very robust. | New technology, not very spread. | Less robust. |
| 7. | Query performance | Structured query allow complex joining. | Queries over anonymous nodes are possible. | Only textual queries are possible. |



**Que 1.4.** **Describe the history of Big data innovation.**

**Answer**

**History of Big data is explained in three phases :**

1. **Big data phase 1 :** During 1970 to 2000, it is a DBMS-based, structured content that include :
   i. RDBMS and data warehousing.
   ii. Extract transfer load.
   iii. Online analytical processing.
   iv. Dashboards and scorecards.
   v. Data mining and statistical analysis.
2. **Big data phase 2 :** During 2000 to 2010, it is a web-based unstructured content that include :
   i. Information retrieval and extraction.
   ii. Opinion mining.
   iii. Web analytics and web intelligence.
   iv. Social media analytics.
   v. Social network analysis.
   vi. Spatial-temporal analysis.
3. **Big data phase 3 :** During 2010 till today it is mobile and sensor-based content that include :
   i. Location-aware analysis.
   ii. Person-centered analysis.
   iii. Context-relevant analysis.
   iv. Mobile visualization.
   v. Human-computer interaction.

## PART-2

*Introduction to Big Data Platform, Drivers for Big Data.*

### Questions-Answers

#### Long Answer Type and Medium Answer Type Questions

**Que 1.5.** **Write a short note on : Big data platform.**

**Answer**

1. Big data platform is a type of IT solution that combines the features and capabilities of several big data application and utilities within a single solution.
2. It is an enterprise class IT platform that enables organization in developing, deploying, operating and managing a big data infrastructure.
3. Big data platform consists of big data storage, servers, database, big data management, business intelligence and other big data management utilities.
4. It also supports custom development, querying and integration with other systems.
5. The primary benefit behind a big data platform is to reduce the complexity of multiple vendors/ solutions into a one cohesive solution.
6. Big data platform are also delivered through cloud where the provider provides an all inclusive big data solutions and services.

**Que 1.6. Describe the drivers of Big data.**

**Answer**

Following are the drivers of Big data :

1. **The digitization of society :**
   i. Big data is largely consumer driven and consumer oriented. Most of the data in the world is generated by consumers.
   ii. Most people consume and generate data through a variety of devices and (social) applications.
   iii. With every click, swipe or message, new data is created in a database somewhere around the world.
   iv. Because everyone now has a smartphone in their pocket, the data creation sums to incomprehensible amounts.
2. **The plummeting of technology costs :**
   i. Technology related to collecting and processing massive quantities of diverse (high variety) data has become increasingly more affordable.
   ii. The costs of data storage and processors keep declining, making it possible for small businesses and individuals to become involved with Big data.
3. **Connectivity through cloud computing :**
   i. Cloud computing environments have made it possible to quickly scale up or scale down IT infrastructure and facilitate a pay-as-you-go model.



   ii. This means that organizations that want to process massive quantities of data do not have to invest in large quantities of IT infrastructure.
4. **Increased knowledge about data science :**
   i. The knowledge and education about data science has greatly professionalized and more information becomes available every day.
   ii. 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 :**
   i. Social media data provides insights into the behaviours, preferences and opinions of the public on a scale that have never been known before.
   ii. Due to this, it is immensely valuable to anyone who is able to derive meaning from these large quantities of data.
   iii. 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.
6. **The upcoming Internet of Things (IoT) :**
   i. 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 enable these objects to connect and exchange data.
   ii. It is increasingly gaining popularity as consumer goods providers start including 'smart' sensors in household appliances.
   iii. Whereas the average household in 2015 had around 10 devices that connected to the internet, this number is expected to rise to 50 per household by 2025.

## PART-3

*Big Data Architecture and Characteristics, 5Vs of Big Data.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.7. Describe the architecture of Big data.**

**Answer**

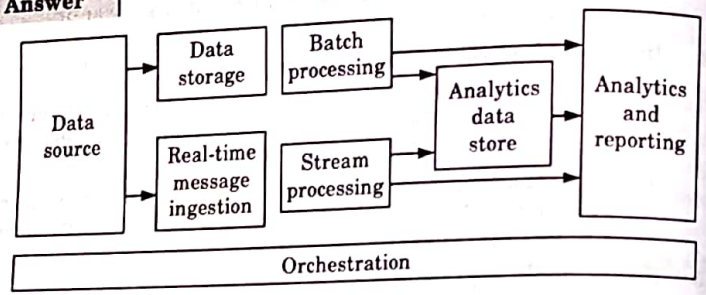



**Fig. 1.7.1.**

Most big data architectures include some or all of the following components :

1. **Data sources :**
   - i. All big data solutions start with one or more data sources. Such as :
     - a. Application data stores, relational databases.
     - b. Static files produced by applications, such as web server log files.
     - c. Real-time data sources, such as IoT devices.
2. **Data storage :**
   - i. Data for batch processing operations is typically stored in a distributed file store that can hold high volumes of large files in various formats.
   - ii. This kind of store is often called a data lake.
3. **Batch processing :**
   - i. Because the data sets are so large, often a big data solution must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis.
   - ii. Usually these jobs involve reading source files, processing them, and writing the output to new files.
4. **Real-time message ingestion :**
   - i. If the solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing.
   - ii. This might be a simple data store, where incoming messages are dropped into a folder for processing.
   - iii. However, many solutions need a message ingestion store to act as a buffer for messages, and to support scale-out processing, reliable delivery, and other message queuing semantics.



5. **Stream processing :**
   - i. After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis.
   - ii. The processed stream data is then written to an output sink.
6. **Analytical data store :**
   - i. 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.
   - ii. Data could be presented through a low-latency NoSQL technology such as HBase, or an interactive Hive database that provides a metadata abstraction over data files in the distributed data store.
7. **Analysis and reporting :**
   - i. The goal of most big data solutions is to provide insights into the data through analysis and reporting.
   - ii. To empower users to analyze the data, the architecture may include a data modeling layer, such as a multi-dimensional OLAP cube or tabular data model.
   - iii. Analysis and reporting can also take the form of interactive data exploration by data scientists or data analysts.
8. **Orchestration :**
   - i. Most big data solutions consist of repeated data processing operations, encapsulated in workflows 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 or dashboard.
   - ii. To automate these workflows, we can use an orchestration technology.

**Que 1.8. What are the characteristics of Big data ?**

**OR**

**Explain 5Vs of Big data.**

**OR**

**Discuss about the three dimensions of Big data.**

**Answer**

Following are the characteristics/5Vs of Big data :

1. **Volume :**
   - i. The name Big data itself is related to a size which is enormous. Size of data plays a very crucial role in determining value out of data.
   - ii. Also, whether a particular data can actually be considered as a Big data or not, is dependent upon the volume of data.

iii. Hence, 'Volume' is one characteristic which needs to be considered while dealing with Big data.

2. **Variety :**
   i. Variety refers to heterogeneous sources and the nature of data, both structured and unstructured.
   ii. Data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc., are also being considered in the analysis applications.
   iii. This variety of unstructured data poses certain issues for storage, mining and analyzing data.
3. **Velocity :**
   i. The term 'velocity' refers to the speed of generation of data.
   ii. Big data velocity deals with the speed at which data flows in from sources like business processes, application logs, networks, and social media sites, sensors, mobile devices, etc.
   iii. The flow of data is massive and continuous.
4. **Value :**
   i. Value is the major issue that we need to concentrate on.
   ii. It is not just the amount of data that we store or process.
   iii. It is actually the amount of valuable, reliable and trustworthy data that needs to be stored, processed, analyzed to find insights.
5. **Veracity :**
   i. It refers to inconsistencies and uncertainty in data, that is, data which is available can sometimes get messy; quality and accuracy are difficult to control.
   ii. Big data is also variable because of the multitude of data dimensions resulting from multiple disparate data types and sources.
   iii. For example, Data in bulk could create confusion whereas less amount of data could convey half or incomplete information.

## PART-4

*Big Data Technology Component.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.9. What are the Big data technology components ?**



**Answer**

Following are the components of Big data technology :

1. **Machine learning (ML) :**
   i. It is the science of making computers learns things by themselves.
   ii. In machine learning, a computer is expected to use algorithms and statistical models to perform specific tasks without any explicit instructions.
   iii. Machine learning applications provide results based on past experience.
2. **Natural Language Processing (NLP) :**
   i. It is the ability of a computer to understand human language as spoken.
   ii. For example, Google Home and Amazon Alexa, both use NLP and other technologies to give us a virtual assistant experience.
3. **Business Intelligence (BI) :**
   i. Business intelligence (BI) is a technology used for analysing data and delivering actionable information that helps executives, managers and workers make informed business decisions.
   ii. The ultimate goal of BI initiatives is to drive better business decisions that enable organizations to increase revenue and gain competitive advantages over business rivals.
4. **Cloud computing :**
   i. Cloud computing is the delivery of computing services including servers, storage, database, networking over the internet.
   ii. For example, Dropbox allow users to access files and store up to one terabyte of data.

## PART-5

*Big Data Importance and Application.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.10. Why Big data is important ?**

**Answer**

Big data is important due to following reasons :

1. **Cost savings :**
   i. Tools of Big data like Hadoop and Cloud-Based Analytics can bring cost advantages to business when large amounts of data are to be stored.
   ii. These tools help in identifying more efficient ways of doing business.
2. **Time reductions :**
   i. The high speed of tools like Hadoop and in-memory analytics can easily identify new sources of data which helps businesses analyzing data immediately and make quick decisions based on the learning.
3. **Understand the market conditions :**
   i. By analyzing Big data we can get a better understanding of current market conditions.
4. **Control online reputation :**
   i. Big data tools can do sentiment analysis.
   ii. Therefore, we can get feedback about who is saying what about our company.
5. **Using Big data analytics to boost customer acquisition and retention :**
   i. The customer is the most important asset any business depends on.
   ii. There is no single business that can claim success without first having to establish a solid customer base.
   iii. Big data can be used for customer acquisition and retention.

**Que 1.11. What are the applications of Big data ?**

**Answer**

Following are the application of Big data :

1. **Health Care :** We have these days wearable devices and sensors that provide real-time updates to the health statement of a patient.
2. **Education :** A student's progress can be tracked and improved by proper analysis through big data analytics.
3. **Weather :**
   i. Weather sensors and satellites, which have been deployed around the globe collect data huge amounts and use that data to monitor the weather and environmental conditions.
   ii. Big data also helps to predict or forecast the weather conditions for the upcoming few days.



4. **Communication media and entertainment :**
   i. Big data helps in collecting, analyzing and utilizing consumer insight.
   ii. It also helps in understanding patterns of real-time, media content usage.
5. **Insurance :**
   i. Big data is used in industry to provide customer insights for transparent products, by analyzing and predicting customer behaviour through social media.
   ii. Big data also helps in better customer retention for insurance companies.

## PART-6

*Big Data Features, Security, Compliance, Auditing and Protection.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.12. What is Big data security ? What are the steps for securing Big data ?**

**Answer**

1. The sheer size of a Big data repository brings with it a major security challenge.
2. Proper security entails more than just keeping the hackers out; it also means backing up data and protecting data from corruption.
3. Big data security is an umbrella term that includes all security measures and tools applied to analytics and data processes.
4. Attacks on big data systems can originate either from offline or online spheres and can crash a system.

**Steps to securing Big data :**

**A. Get rid of unwanted data :**

1. Securing the massive amounts of data can be addressed in several ways.
2. A starting point is to get rid of data that are no longer needed.
3. If you do not need certain information, it should be destroyed, because it represents a risk to the organization.

4. There are situations in which information cannot legally be destroyed; in that case, the information should be securely archived by an offline method.

**B. Classifying data :**

1. Protecting data becomes much easier if the data are classified.
2. The data should be divided into appropriate groupings for management purposes.
3. A classification system does not have to be very sophisticated or complicated to enable the security process.
4. Classification can become a powerful tool for determining the sensitivity of data.
5. Once organizations better understand their data, they can take important steps to segregate the information.

**Que 1.13. Write a short note on : Big data and Compliance.**

**Answer**

1. Compliance issues have a major effect on how big data is protected, stored, accessed, and archived.
2. Big data is not easily handled by the relational databases.
3. Big data is transforming the storage and access paradigms to an emerging world of horizontally scaling and unstructured databases.
4. This new world of file types and data is prompting analysis professionals to think of new problems to solve.
5. It is clear that a rebalancing of the database landscape is about to commence.
6. This has everything to do with compliance.
7. New data types and methodologies are still expected to meet the legislative requirements placed on businesses by compliance laws.
8. There will be no excuses accepted if a new data methodology breaks the law.
9. Preventing compliance from becoming the next big data nightmare is going to be the job of security professionals.

**Que 1.14. Write a short note on : Protecting big data analytics.**

**Answer**

1. Protecting data is an often forgotten inclination in the big data initiatives.
2. Big data contains all of the things you don't want to see when you are trying to protect data.



3. Big data can contain very unique sample sets that are accumulated frequently and in real time.
4. All of the data are unique to the moment, and if they are lost, they are impossible to recreate.
5. That uniqueness also means we cannot leverage time-saving backup preparation and security technologies.
6. This greatly increases the capacity requirements for backup subsystems, slows down security scanning, makes it harder to detect data corruption, and complicates archiving.
7. There is also the issue of the large size and number of files often found in Big data analytic environments.
8. Analytic information is often processed into an Oracle, NoSQL, or Hadoop environment, so real-time protection of that environment may be required.

## PART-7

*Big Data Privacy and Ethics.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.15. What is big data privacy ? Mention big data privacy concerns.**

**Answer**

1. Big data privacy involves properly managing big data to minimize risk and protect sensitive data.
2. Because big data comprises large and complex data sets, many traditional privacy processes cannot handle the scale and velocity required.
3. To safeguard big data we need to create a framework for privacy protection that can handle the volume, velocity, variety, and value of big data.

**Big data privacy concerns :**

1. With more data spread across more locations, the risk of a privacy breach has never been higher.
2. Big data privacy is a matter of customer trust.
3. The more data you collect about users, the easier it gets to understand their current behavior, draw inferences about their future behavior,

and eventually develop deep and detailed profiles of their lives and preferences.

4. The more data you collect, the more important it is to be transparent with your customers.
5. The volume and velocity of data from existing sources is expanding fast.
6. To keep pace, your big data privacy strategy needs to expand, too.
7. That requires you to consider following issues :
   i. What do you intend to do with customer and user data ?
   ii. How accurate is the data, and what are the potential consequences of inaccuracies ?
   iii. How will your data security scale to keep up with threats of data breaches and insider threats ?
   iv. Where is your balancing point between the need to keep data locked down in-place and the need to expose it safely so you can extract value from it ?
   v. How do you maintain compliance with data privacy regulations that vary across the countries and regions where you do business ?
   vi. How do you maintain transparency about what you do with the big data you collect ?

**Que 1.16. Explain principles of Big data ethics.**

**Answer**

Following are the principles of Big data ethics :

1. **Private customer data and identity should remain private :** Privacy does not mean secrecy, as private data might need to be audited based on legal requirements, but that private data obtained from a person with their consent should not be exposed for use by other businesses or individuals with any traces to their identity.
2. **Shared private information should be treated confidentially :**
   i. Third party companies share sensitive data like medical, financial or locational and need to have restrictions on whether and how that information can be shared further.
   ii. Customers should have a transparent view of how their data is being used or sold, and the ability to manage the flow of their private information across third-party analytical systems.
3. **Big data should not interfere with human will :**
   i. Big data analytics can moderate and even determine who we are before we make up our own minds.
   ii. Companies need to begin to think about the kind of predictions and inferences that should be allowed and the ones that should not be.



4. Big data should not institutionalize unfair biases like racism or sexism. Machine learning algorithms can absorb unconscious biases in a population and amplify them via training samples.

## PART-8

*Big Data Analytics.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.17. Describe briefly Big data analytics.**

**Answer**

1. The volume of data that one has to deal has exploded to unimaginable levels in the past decade, and at the same time, the price of data storage has systematically reduced.
2. Private companies and research institutions capture terabytes of data about their user's interactions, business, social media, and also sensors from devices such as mobile phones and automobiles.
3. Big data analytics involves collecting data from different sources, manage it in a way that it becomes available to be consumed by analysts and finally deliver data products useful to the organization business.
4. The process of converting large amounts of unstructured raw data, retrieved from different sources to a data product useful for organizations forms the core of Big data analytics.

**Que 1.18. What are the advantages and disadvantages of Big data analytics ?**

**Answer**

**Advantages of Big data analytics :**

1. It detects and corrects the errors from data sets using data cleansing.
2. Improves quality of data and give benefit to both customers and institution.
3. It removes duplicate information from data sets *i.e.*, save memory space.
4. It helps in displaying relevant advertisements on the online shopping websites based on historic data.

5. It helps in increasing revenue and productivity of the companies.
6. It reduces banking risks by identifying probable fraudulent customers based on historic data analysis.
7. It is used by security agencies for surveillance and monitoring purpose based on information collected by huge number of sensors.

**Disadvantages of Big data analytics :**

1. This may breach privacy of the customers as their information such as purchases, online transactions, subscriptions are visible to their parent companies.
2. The cost of data analytics tools vary based on applications and features supported.
3. The information obtained using data analytics can also be misused against group of people of certain country or community or caste.
4. It is very difficult to select the right data analytics tools.

## PART-9

*Challenges of Conventional System, Intelligent Data Analysis, Nature of Data.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 1.19. What is conventional system ? List some of the challenges of conventional systems.**

**Answer**

1. The system consists of one or more zones each having either manually operated call points or automatic detection devices, or a combination of both.
2. Big data is huge amount of data which is beyond the processing capacity of conventional data base systems to manage and analyze the data in a specific time interval.

**Challenges of conventional systems :** Following are the challenges of conventional system :

1. **The uncertainty of data management landscape :**
   i. Because Big data is continuously expanding, there are new companies and technologies that are being developed every day.



   ii. A big challenge for companies is to find out which technology works bests for them without the introduction of new risks and problems.
2. **Talent gap in Big data :**
   i. While Big data is a growing field, there are very few experts available in this field.
   ii. This is because Big data is a complex field and people who understand the complexity and intricate nature of this field are few.
3. **Getting data into Big data structure :**
   i. Data is increasing every single day. This means that companies have to tackle limitless amount of data on a regular basis.
   ii. The scale and variety of data that is available today can overwhelm any data practitioner and that is why it is important to make data accessibility simple and convenient.
4. **Need for synchronization across data sources :**
   i. As data sets become more diverse, there is a need to incorporate them into an analytical platform.
   ii. If this is ignored, it can create gaps and lead to wrong insights and messages.
5. **Getting important insights through the use of Big data analytics :**
   i. It is important that companies gain proper insights from big data analytics and it is important that the correct department has access to this information.
   ii. A major challenge in the Big data analytics is bridging this gap in an effective fashion.

**Que 1.20. Explain intelligent data analysis.**

**Answer**

1. Intelligent Data Analysis (IDA) reveals implicit, previously unknown and potentially valuable information or knowledge from large amounts of data.
2. Intelligent data analysis is also a kind of decision support process.
3. Based on artificial intelligence, machine learning, pattern recognition, statistics, database and visualization technology mainly, IDA automatically extracts useful information, necessary knowledge and interesting models from a lot of online data in order to help decision makers make the right choices.

4. The process of IDA generally consists of the following three stages :
   i. **Data preparation :** Data preparation involves selecting the required data from the relevant data source and integrating this into a data set to be used for data mining.
   ii. **Rule finding :** Rule finding is working out rules contained in the data set by means of certain methods or algorithms.
   iii. **Result validation :** Result validation requires examining these rules, and result explanation is giving intuitive, reasonable and understandable descriptions using logical reasoning.

**Que 1.21. What is data ? List the properties of data. Describe the types of data.**

**Answer**

1. Data is a set of values of qualitative or quantitative variables; restated, pieces of data are individual pieces of information.
2. Data is measured, collected and reported, and analyzed, whereupon it can be visualized using graphs or images.

**Properties of data :** Following are the properties of data :

1. **Amenability of use :** From the dictionary meaning of data it is learnt that data are facts used in deciding something. In short, data are meant to be used as a base for arriving at definitive conclusions.
2. **Clarity :** Data are a crystallized presentation. Without clarity, the meaning desired to be communicated will remain hidden.
3. **Accuracy :** Data should be real, complete and accurate. Accuracy is thus, an essential property of data.
4. **Essence :** A large quantities of data are collected and they have to be compressed and refined. Data so refined can present the essence or derived qualitative value, of the matter.
5. **Aggregation :** Aggregation is cumulating or adding up.
6. **Compression :** Large amounts of data are always compressed to make them more meaningful. Compress data to a manageable size. Graphs and charts are some examples of compressed data.
7. **Refinement :** Data require processing or refinement. When refined, they are capable of leading to conclusions or even generalizations. Conclusions can be drawn only when data are processed or refined.



**Types of data :** Following are the types of data :

1. **Categorical data :**
   i. These are values or observations that can be sorted into groups or categories.
   ii. There are two types of categorical values, nominal and ordinal.
   iii. A nominal variable has no intrinsic ordering to its categories.
   iv. For example, housing is a categorical variable having two categories (own and rent).
   v. An ordinal variable has an established ordering.
2. **Numerical data :**
   i. These are values or observations that can be measured.
   ii. There are two kinds of numerical values, discrete and continuous.
   iii. Discrete data are values or observations that can be counted and are distinct and separate.
   iv. For example, number of lines in a code.
   v. Continuous data are values or observations that may take on any value within a finite or infinite interval.

## PART-10

*Analytics Process and Tools.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.22. Explain the steps involved in analytic process.**

**Answer**

Following are the steps of analytic process :

**Step 1 : Deployment :**

1. In this phase, we need to plan the deployment, monitoring and maintenance.
2. We produce a final report and review the project.
3. In this phase, we deploy the results of the analysis. This is also known as reviewing the project.

**Step 2 : Business Understanding :**

1. Business objectives are defined in this phase.
2. Whenever any requirement occurs, we need to assess the situation, determine data mining goals and then produce the project plan as per the requirement.

**Step 3 : Data Exploration :**

1. The step consists of data understanding.
2. This is necessary to verify the quality of data collected.
3. In this phase, we gather initial data, describe and explore the data and verify data quality to ensure it contains the data we require.
4. Data collected from the various sources is described in terms of its application and the need for the project in this phase. This is also known as data exploration.

**Step 4 : Data Preparation :**

1. From the data collected in the last step, we need to select data as per the need, clean it, construct it to get useful information and then integrate it all.
2. Finally, we need to format the data to get the appropriate data.
3. Data is selected, cleaned, and integrated into the format finalized for the analysis in this phase.

**Step 5 : Data Modeling :**

1. In this phase, we select a modeling technique, generate test design build a model and assess the model built.
2. The data model is build to analyze relationships between various selected objects in the data.
3. Test cases are built for assessing the model and model is tested and implemented on the data in this phase.

**Que 1.23. What are the tools used for analytic processes ?**

**Answer**

**A. Big data tools for high performance computing (HPC) and supercomputing :**

1. Message Passing Interface (MPI)

**B. Big data tools on clouds :**

1. MapReduce model



2. Iterative MapReduce model
3. DAG model
4. Graph model
5. Collective model

**C. Other BDA tools :**

1. SaS
2. R
3. Hadoop

## PART-11

*Analytics Vs Reporting, Modern Data Analytics Tools.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 1.24. What is analysis ? What is reporting ? Differentiate between analysis and reporting.**

**Answer**

**Analysis :**

1. Analysis is the process of exploring data and reports in order to extract meaningful insights, which can be used to better understand and improve business performance.
2. The goal of analysis is to answer questions by interpreting the data at a deeper level and providing actionable recommendations.
3. A firm may be focused on the general area of analytics (strategy, implementation, reporting, etc.), but not necessarily on the specific aspect of analysis.
4. Analysis transforms data and information into insights.

**Reporting :**

1. Reporting is the process of organizing data into informational summaries in order to monitor how different areas of a business are performing.
2. Measuring core metrics and presenting them falls under this category.

3. Reporting helps companies to monitor their online business and be alerted to when data falls outside of expected ranges.
4. Good reporting should raise questions about the business from its end users. Reporting translates raw data into information.

**Difference between analysis and reporting :** The basic differences between analysis and reporting are as follows :

| S.No. | Analysis | Reporting |
|---|---|---|
| 1. | Provides what is needed. | Provides what is asked for. |
| 2. | Is typically customized. | Is typically standardized. |
| 3. | Involves a person. | Does not involve a person. |
| 4. | Is extremely flexible. | Is fairly inflexible. |

**Que 1.25. Describe modern data analytics tools.**

**Answer**

Current modern analytic tools concentrate on following three classes :

**A. Batch processing tools :**

1. Batch processing system involves collecting a series of processing jobs and carrying them out periodically as a group (or batch) of jobs.
2. It allows a large volume of jobs to be processed at the same time.
3. One of the most famous and powerful batch process-based big data tool is Apache Hadoop.
4. It provides infrastructures and platforms for other specific big data applications.
5. Following are some batch processing tools :
   - i. **Apache Hadoop :** It is used to provide infrastructures and platforms for big data applications. It possesses high scalability, reliability and completeness.
   - ii. **Apache Mahout :** It is used to provide machine learning algorithms to businesses.
   - iii. **Talend Open Studio :** It is used to provide data management and application integration.

**B. Stream processing tools :**

1. Stream processing helps in predicting the life in data as and when it transpires.



2. The key strength of stream processing is that it can provide insights faster.
3. It helps understanding the hidden patterns in millions of data records in real time.
4. It processes the data from single or multiple sources in real or near-real time applying the desired business logic and emitting the processed information.
5. Following are some of the real time data streaming tools :
   - i. **Apache Storm :** It is a distributed real-time computation system. Its applications are designed as directed acyclic graphs. Storm is a stream processing engine without batch support.
   - ii. **Apache Flink :** It is a streaming data flow engine that provides communication fault tolerance and data distribution computation over data stream. It can execute both stream processing and batch processing easily. It is designed as an alternative to Map Reduce.
   - iii. **Amazon Kinesis :** It is an out of the box streaming data tool. It comprises of shards which Kafka calls partitions. It solves a variety of streaming data problems.

**C. Interactive analysis tools :**

1. The interactive analysis presents the data in an interactive environment, allowing users to undertake their own analysis of information.
2. Users are directly connected to the computer and hence can interact with it in real time.
3. The data can be reviewed, compared and analyzed in tabular or graphic format or both at the same time.
4. Following are some interactive analysis tools :
   - i. **Google's Dremel :**
     - a. In 2010, Google proposed an interactive analysis system, named Dremel, which is scalable for processing nested data.
     - b. Dremel provides a very fast SQL like interface to the data by using a different technique than Map Reduce.
     - c. Dremel has a very different architecture compared with well-known Apache Hadoop.
     - d. Dremel has capability to run aggregation queries over trillion-row tables in seconds by means of combining multi-level execution trees and columnar data layout.

**ii. Apache Drill :**

a. It is an Apache open-source SQL query engine for big data exploration.

b. Drill is designed to support high-performance analysis on the semi-structured and rapidly evolving data coming from modern big data applications.

c. Drill provides plug-and-play integration with existing Apache Hive and Apache HBase deployments.

☺☺☺

# 2 UNIT

# Hadoop

## CONTENTS

## PART-1

*History of Hadoop, Apache Hadoop.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 2.1. Describe the history of Hadoop.**

**Answer**

1. In 2002, Doug Cutting and Mike Cafarella started to work on a project, Apache Nutch. It is an open source web crawler software project.
2. While working on Apache Nutch, they were dealing with big data. To store that data they have to spend a lot of costs which becomes the consequence of that project. This problem becomes one of the important reasons for the emergence of Hadoop.
3. In 2003, Google introduced a file system known as GFS (Google file system). It is a proprietary distributed file system developed to provide efficient access to data.

   In 2004, Google released a white paper on Map Reduce. This technique simplifies the data processing on large clusters.

   In 2005, Doug Cutting and Mike Cafarella introduced a new file system known as NDFS (Nutch Distributed File System). This file system also includes Map Reduce.

   In 2006, Doug Cutting quit Google and joined Yahoo. On the basis of the Nutch project, Dough Cutting introduces a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System). Hadoop first version 0.1.0 released in this year.
4. In 2007, Yahoo runs two clusters of 1000 machines.
5. In 2008, Hadoop became the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds.
9. In 2013, Hadoop 2.2 was released.
10. In 2017, Hadoop 3.0 was released.

**Que 2.2. Write short note on Apache Hadoop.**

**Answer**

1. Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data.



2. Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.
3. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
4. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.
5. The Apache Hadoop framework is composed of the following modules :
   i. **Hadoop Common :** It contains libraries and utilities needed by other Hadoop modules.
   ii. **Hadoop Distributed File System (HDFS) :** A distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster.
   iii. **Hadoop YARN :** A resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users applications.
   iv. **Hadoop Map Reduce :** A programming model for large scale data processing.

## PART-2

*The Hadoop Distributed File System, Components of Hadoop, Data Format.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 2.3. What is Hadoop Distributed File System (HDFS) ? How does HDFS work ? Also explain the features of HDFS.**

**Answer**

1. 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.
2. The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications.
3. HDFS employs a NameNode and DataNode architecture to implement a distributed file system.
4. It is highly fault-tolerant and is designed to be deployed on low-cost hardware.

5. It provides high throughput access to application data and is suitable for applications having large datasets.

**Working of HDFS :**

1. The way HDFS works is by having a main NameNode and multiple DataNode on a commodity hardware cluster.
2. All the nodes are usually organized within the same physical rack in the data center.
3. Data is then broken down into separate blocks that are distributed among the various DataNodes for storage.
4. NameNode is the master daemon in HDFS. It runs on the master nodes.
5. It maintains the filesystem namespace.
6. NameNode does not store the actual data.
7. It stores the metadata, such as information about blocks of files, files permission, blocks locations, etc.
8. NameNode manages the DataNode and provides instructions to them.
9. DataNode is the slave daemon in HDFS.
10. DataNodes are the slave nodes that store the actual business data.
11. They are responsible for serving the client's read/write requests based on the instructions from NameNode.

**Features of HDFS :** Following are the features of HDFS :

1. **Data replication :** This is used to ensure that the data is always available and prevents data loss.
2. **Fault tolerance and reliability :** HDFS ability to replicate file blocks and store them across nodes in a large cluster ensures fault tolerance and reliability.
3. **High availability :** Due to replication across nodes the data is available even if the NameNode or a DataNode fails.
4. **Scalability :** Because HDFS stores data on various nodes in the cluster, as requirements increase, a cluster can scale to hundreds of nodes.
5. **High throughput :** Since HDFS stores data in a distributed manner, the data can be processed in parallel on a cluster of nodes, this cuts the processing time and enable high throughput.

**Que 2.4. Describe the goals of HDFS.**

**Answer**

Following are the goals of HDFS :

1. **Fault detection and recovery :**
   i. Since HDFS includes a large number of commodity hardware, failure of components is frequent.



   ii. Therefore, HDFS should have mechanisms for quick and automatic fault detection and recovery.
2. **Huge datasets :**
   i. HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
3. **Hardware at data :**
   i. A requested task can be done efficiently, when the computation takes place near the data.
   ii. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.
4. **Handling the hardware failure :**
   i. The HDFS contains a multiple server machines.
   ii. If any machine fails, the HDFS goal is to recover it quickly.
5. **Streaming data access :**
   i. The HDFS applications usually run on the general purpose file system.
   ii. This application requires streaming access to their sets.
6. **Coherence model :**
   i. The application that runs on HDFS require to follow the write-once-read-many approach.
   ii. So, a file once generated need not to be changed. However, it can be appended and truncate.

**Que 2.5. What are the benefits of using HDFS ?**

**Answer**

Following are the main advantages of using HDFS :

1. **Cost effectiveness :** The DataNodes that store the data rely on inexpensive off-the-shelf hardware, which cuts storage costs. Also, because HDFS is open source, there's no licensing fee.
2. **Large data set storage :** HDFS stores a variety of data of any size – from megabytes to petabytes – and in any format, including structured and unstructured data.
3. **Fast recovery from hardware failure :** HDFS is designed to detect faults and automatically recover on its own.
4. **Portability :** HDFS is portable across all hardware platforms, and it is compatible with several operating systems, including Windows, Linux and Mac OS/X.
5. **Streaming data access :** HDFS is built for high data throughput, which is best for access to streaming data.

**Que 2.6. What are various components of the Hadoop ?**

**Answer**

Different components of the Hadoop are as follows :

1. **HDFS (Hadoop Distributed File System) :**
   i. It is the storage component of Hadoop that stores data in the form of files.
   ii. Each file is divided into blocks of 128MB (configurable) and stores them on different machines in the cluster.
   iii. It has a master-slave architecture with two main components : Name Node and Data Node.
2. **MapReduce :**
   i. To handle Big Data, Hadoop relies on the MapReduce algorithm.
   ii. It essentially divides a single task into multiple tasks and processes them on different machines.
   iii. It has two important phases : Map and Reduce.
   iv. Map phase filters, groups, and sorts the data.
   v. Reduce phase aggregates the data, summarises the result, and stores it on HDFS.
3. **YARN :**
   i. YARN or Yet Another Resource Negotiator manages resources in the cluster and manages the applications over Hadoop.
   ii. It allows data stored in HDFS to be processed and run by various data processing engines.
4. **Hadoop Common :**
   i. It contains libraries and utilities needed by other Hadoop modules.

**Que 2.7. What are the various data formats used in Hadoop ?**

**Answer**

Following are the various data formats used in Hadoop :

1. **Text/CSV :**
   i. A plain text file (CSV) is the most common format both outside and within the Hadoop ecosystem.
   ii. It does not support block compression, so the compression of a CSV file in Hadoop can have a high cost in reading.
2. **SequenceFile :**
   i. The SequenceFile format stores the data in binary format.
   ii. This format accepts compression.



   iii. It does not store metadata and the only option in the evolution of its scheme is to add new fields at the end.
   iv. This is usually used to store intermediate data in the input and output of Map Reduce processes.
3. **Avro :**
   i. Avro is a row-based storage format.
   ii. This format includes in each file, the definition of the scheme of our data in JSON format, improving interoperability and allowing the evolution of the scheme.
   iii. Avro also allows block compression in addition to its divisibility, making it a good choice for most cases when using Hadoop.
4. **Parquet :**
   i. Parquet is a column-based binary storage format that can store nested data structures.
   ii. This format is very efficient in terms of disk input / output operations when the necessary columns to be used are specified.
   iii. This format is very optimized for use with Cloudera Impala.
5. **RCFile (Record Columnar File) :**
   i. RCFile is a columnar format that divides data into groups of rows, and inside it, data is stored in columns.
   ii. This format does not support the evaluation of the scheme and if we want to add a new column it is necessary to rewrite the file, which slows down the process.

## PART-3

*Analyzing Data with Hadoop.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 2.8. Give reasons why Hadoop can be considered a helpful tool to analyze the big data ?**

**Answer**

Following reasons why Hadoop can be considered a helpful tool to analyze the big data :

**A. Storage :**

1. Storing data is one of the biggest challenges for traditional methods of handling large sets of incoming data.
2. HDFS is one of the components of Hadoop that helps maintaining the storage of big data.
3. HDFS consist of single cluster or multiple clusters. Every cluster consists of blocks which is 128 MB by default.
4. When a user defines input, the contents of the input are equally divided into the blocks.
5. And the data are replicated into the data nodes.
6. So, HDFS allows user to store the lots of dataset and when needed, more servers can be added at a very low cost.

**B. Processing :**

1. When the size of the dataset is larger, the time taken to process it is also longer while using the traditional methods.
2. More servers are added to store the large quantity of data, but server does not support the parallel computing.
3. In case of Hadoop the processing of data is done using parallel computing which saves the processing time.

**C. Cost efficiency :**

1. Maintaining the database at a minimum cost is the one of the most important challenges of the big data.
2. Companies using traditional method of handling big data are spending $25,000 to $50,000 per year for 1 terabyte of data.
3. Hadoop software can reduce this cost into few thousand dollars per terabyte per year.

**D. Allows more data to capture :**

1. Due to cost related issues many companies do not capture the large volume of data.
2. But when Hadoop software is used, companies are saving lots of costs of maintaining the data.
3. So, extra data can be stored at the same price if we use Hadoop instead of using traditional method of handling big data.
4. This allows companies to capture more and more data at a low cost.

**E. Provides scalable analytics :**

1. The HDFS and Map Reduce components of Hadoop allow parallel storing and processing of data.
2. With the increase of volume of the data the analytics can be scalable in parallel distributed way.



**F. Provides rich analytics :**

1. Hadoop has a unique quality of handling big data in different programming languages.
2. The project in Hadoop can be done using one of these coding languages Java, Python, SQL, R, and Ruby.
3. They are open source and easy to learn the programming languages.

**Que 2.9. Which tools are used to analyze data using Hadoop ?**

**Answer**

Following tools are used for data analyzing using Hadoop :

**1. Apache Spark :**

i. Apache Spark in an open-source processing engine that is designed for ease of analytics operations.

ii. It is a cluster computing platform that is designed to be fast and made for general purpose uses.

iii. Spark is designed to cover various batch applications, Machine Learning, streaming data processing, and interactive queries.

**2. MapReduce :**

i. MapReduce is just like an algorithm or a data structure that is based on the YARN framework.

ii. The primary feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster, which makes Hadoop working so fast because when we are dealing with Big data, serial processing is no more of any use.

**3. Apache Hive :**

i. Apache Hive is a data warehousing tool that is built on top of the Hadoop.

ii. Hive is one of the best tools used for data analysis on Hadoop.

iii. The query language of used in Hive is known as HQL or HIVEQL.

**4. Apache Impala :**

i. Apache Impala is an open-source SQL engine designed for Hadoop.

ii. Impala overcomes the speed-related issue in Apache Hive with its faster-processing speed.

iii. Apache Impala uses similar kinds of SQL syntax, ODBC driver, and user interface as that of Apache Hive.

iv. Apache Impala can easily be integrated with Hadoop for data analytics purposes.

**5. Apache Mahout :**

i. Apache Mahout runs the algorithm on the top of Hadoop, so it is named Mahout.

ii. Mahout is mainly used for implementing various Machine Learning algorithms on Hadoop like classification, Collaborative filtering, Recommendation.

iii. Apache Mahout can implement the Machine algorithms without integration on Hadoop.

## PART-4

*Scaling Out, Hadoop Streaming, Hadoop Pipes, Hadoop Echo System.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 2.10. Describe the term scaling out.**

OR

**Differentiate "Scale up and scale out". Explain with an example how Hadoop uses scale out feature to improve the performance.**

**Answer**

| S. No. | Scale up | Scale out |
|---|---|---|
| 1. | The term "scaling-up" means to use a powerful single server to process the workload that fits within the server boundaries. | Scale-out utilizes multiple processors as a single entity so we can scale beyond the computer capacity of a single server. |
| 2. | Scale-up system consists of many shelves of drives and a pair of controllers. | Scale-out systems consists of clusters, which are co-equal nodes that work together. |
| 3. | As you need more space, you add more shelves of drives. | As you need more space, nodes can be added or removed. |
| 4. | You need to purchase new hardware every time you want to upgrade your system. | You do not have to purchase new hardware every time you want to upgrade your system. |

**Scale out :**

1. To scale out the data flow for large inputs, we need to store the data in a distributed filesystem, typically HDFS.



2. This allows Hadoop to move the MapReduce computation to each machine hosting a part of the data.
3. Hadoop runs the job by dividing it into tasks.
4. There are two types of task: map tasks and reduce tasks.
5. The nodes that control the job execution process are: jobtracker and tasktracker.
6. The jobtracker coordinates all the jobs run on the system by scheduling tasks to run on tasktrackers.
7. Tasktrackers run tasks and send progress reports to the jobtracker.
8. Hadoop divides the input to a MapReduce job into fixed-size pieces called splits.
9. Hadoop creates one map task for each split.
10. The time taken to process each split is small.
11. If we are processing the splits in parallel, the processing is better load-balanced.
12. For most jobs, a good split size tends to be the size of an HDFS block.
13. Hadoop runs the map task on a node where the input data resides in HDFS.

**Que 2.11. What is Hadoop Streaming ?**

**Answer**

1. Hadoop Streaming is a utility that comes with the Hadoop distribution.
2. This utility allows us to create and run Map Reduce jobs with any executable or script as the mapper and the reducer.
3. It uses Unix streams as the interface between the Hadoop and our Map Reduce program so that we can use any language which can read standard input and write to standard output to write for writing our Map Reduce program.
4. Hadoop Streaming supports the execution of non-Java, programmed Map Reduce jobs execution over the Hadoop cluster.
5. It supports Python, Perl, R, PHP, and C++ programming languages.

**Que 2.12. Write short note on Hadoop Pipes.**

**Answer**

1. Hadoop Pipes is the name of the C++ interface to Hadoop Map Reduce.
2. Unlike streaming, which uses standard input and output to communicate with the map and reduce code, Pipes uses sockets as the channel over which the tasktracker communicates with the process running the C++ map or reduce function.

3. Hadoop pipes allow users to use the C++ language for Map Reduce programming.
4. The main method it takes is to put the C++ code of the application logic in a separate process, and then let the Java code communicate with C++ code through the socket.
5. To a large extent, this approach is similar to Hadoop streaming, where communication differs : one is the standard input output and the other is the socket.

**Que 2.13. Describe briefly Hadoop Ecosystem.**

**Answer**

1. Hadoop Ecosystem is a platform or a suite which provides various services to solve the Big data problems.
2. It includes Apache projects and various commercial tools and solutions.
3. Most of the tools or solutions are used to supplement or support these major elements.
4. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.
5. Different components of the Hadoop are as follows :

**1. HDFS (Hadoop Distributed File System) :**

i. It is the storage component of Hadoop that stores data in the form of files.

ii. Each file is divided into blocks of 128MB (configurable) and stores them on different machines in the cluster.

iii. It has a master-slave architecture with two main components : Name Node and Data Node.

**2. MapReduce :**

i. To handle Big data, Hadoop relies on the MapReduce algorithm.

ii. It essentially divides a single task into multiple tasks and processes them on different machines.

iii. It has two important phases : Map and Reduce.

iv. Map phase filters, groups, and sorts the data.

v. Reduce phase aggregates the data, summarises the result, and stores it on HDFS.

**3. YARN :**

i. YARN or Yet Another Resource Negotiator manages resources in the cluster and manages the applications over Hadoop.

ii. It allows data stored in HDFS to be processed and run by various data processing engines.



**4. HBase :**

i. HBase is a Column-based NoSQL database.

ii. It runs on top of HDFS and can handle any type of data.

iii. It allows for real-time processing and random read/write operations to be performed in the data.

**5. Pig :**

i. Pig was developed for analyzing large datasets and overcomes the difficulty to write map and reduce functions.

ii. It consists of two components : Pig Latin and Pig Engine.

iii. Pig Latin is the Scripting Language.

iv. Pig Engine is the execution engine on which Pig Latin runs.

**6. Hive :**

i. Hive is a distributed data warehouse system.

ii. It allows for easy reading, writing, and managing files on HDFS.

iii. It has its own querying language known as Hive Querying Language (HQL).

**7. Sqoop :**

i. Sqoop plays an important part in bringing data from Relational Databases into HDFS.

ii. The commands written in Sqoop internally converts into MapReduce tasks that are executed over HDFS.

iii. It works with almost all relational databases.

iv. It can also be used to export data from HDFS to RDBMS.

**8. Flume :**

i. Flume is an open-source service used to efficiently collect, aggregate, and move large amounts of data from multiple data sources into HDFS.

ii. It can collect data in real-time as well as in batch mode.

iii. It has a flexible architecture and is fault-tolerant with multiple recovery mechanisms.

**9. Kafka :**

i. Kafka sits between the applications generating data (Producers) and the applications consuming data (Consumers).

ii. Kafka is distributed and has in-built partitioning, replication, and fault-tolerance.

iii. It can handle streaming data and also allows businesses to analyze data in real-time.

9. **Oozie :**
   i. Oozie is a workflow scheduler system that allows users to link jobs written on various platforms like MapReduce, Hive, Pig, etc.
   ii. Using Oozie you can schedule a job in advance and can create a pipeline of individual jobs to be executed sequentially or in parallel to achieve a bigger task.

10. **Zookeeper :**
   i. In a Hadoop cluster, coordinating and synchronizing nodes can be a challenging task.
   ii. Zookeeper is the perfect tool for the problem.
   iii. It is an open-source, distributed, and centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services across the cluster.

## PART-5

*Map Reduce, Map Reduce Framework and Basics, How Map Reduce Works.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 2.14. Write short note on MapReduce.**

**Answer**

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



6. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

**Que 2.15. Explain different phases of MapReduce.**

**OR**

**What is MapReduce ? Explain the stages of MapReduce program execution.**

**Answer**

**MapReduce :** Refer Q. 2.14, Page 2–14Q, Unit-2.

Following are the different phases/stages of MapReduce program execution :

1. **Input splits :**
   i. An input to a MapReduce in Big data job is divided into fixed-size pieces called input splits.
   ii. Input split is a chunk of the input that is consumed by a single map.
2. **Mapping :**
   i. In this phase data in each split is passed to a mapping function to produce output values.
   ii. For example, a job of mapping phase is to count a number of occurrences of each word from input splits and prepare a list in the form of <word, frequency>.
3. **Shuffling :**
   i. This phase consumes the output of Mapping phase.
   ii. Its task is to consolidate the relevant records from Mapping phase output.
   iii. For example, the same words are clubbed together along with their respective frequency.
4. **Reducing :**
   i. In this phase, output values from the shuffling phase are aggregated.
   ii. This phase combines values from shuffling phase and returns a single output value.

**Que 2.16. Describe how MapReduce works.**

**OR**

**Explain in detail about MapReduce workflows.**

**Answer**

**Steps of MapReduce workflows :**

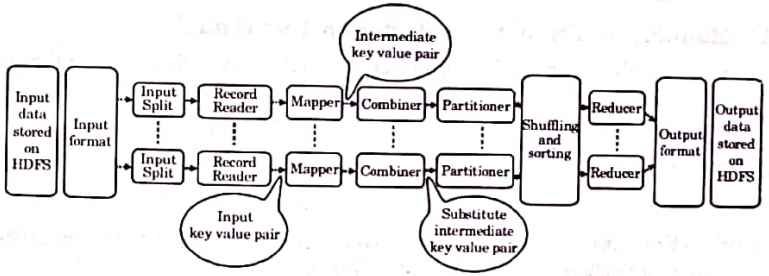
1. **Input Files :**
   i. In input files data for MapReduce job is stored.
   ii. Input files reside in HDFS.

iii. Input files format is arbitrary. Line-based log files and binary format can also be used.

2. **InputFormat :**
   i. InputFormat defines how to split and read these input files.
   ii. It selects the files or other objects for input.
   iii. InputFormat creates InputSplit.
3. **InputSplits :**
   i. It represents the data which will be processed by an individual Mapper.
   ii. For each split, one map task is created. Thus the number of map tasks is equal to the number of InputSplits.
   iii. Framework divide split into records, which mapper process.
4. **RecordReader :**
   i. It communicates with the InputSplit.
   ii. And then converts the data into key-value pairs suitable for reading by the Mapper.
5. **Mapper :**
   i. It processes input record produced by the RecordReader and generates intermediate key-value pairs.
   ii. The intermediate output is completely different from the input pair.
   iii. The output of the mapper is the full collection of key-value pairs.
   iv. The Mapper passes the output to the combiner for further processing.
6. **Combiner :**
   i. Combiner performs local aggregation on the mapper's output.
   ii. It minimizes the data transfer between mapper and reducer.
   iii. When the combiner functionality completes, framework passes the output to the partitioner for further processing.
7. **Partitioner :**
   i. Partitioner comes into the existence if we are working with more than one reducer.
   ii. It takes the output of the combiner and performs partitioning.
   iii. Partitioning in MapReduce execution allows even distribution of the map output over the reducer.
8. **Shuffling and Sorting :**
   i. After partitioning, the output is shuffled to the reduce node.
   ii. The shuffling is the physical movement of the data which is done over the network.



   iii. As all the mappers finish and shuffle the output on the reducer nodes.
   iv. Then framework merges this intermediate output and sort. This is then provided as input to reduce phase.
9. **Reducer :**
   i. Reducer then takes set of intermediate key-value pairs produced by the mappers as the input.
   ii. After that runs a reducer function on each of them to generate the output.
   iii. The output of the reducer is the final output. Then framework stores the output on HDFS.
10. **RecordWriter :**
    i. It writes these output key-value pair from the Reducer phase to the output files.
11. **OutputFormat :**
    i. OutputFormat defines the way how RecordReader writes these output key-value pairs in output files.
    ii. The OutputFormat instances write the final output of reducer on HDFS.




## PART-6

*Developing a Map Reduce Application, Unit Test with MR Unit, Test Data and Local Tests, Anatomy of a Map Reduce Job Run.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 2.17.** **Give the phases of developing a MapReduce application.**

**Answer**

**Phases of developing a MapReduce Application :**

1. **Configuration API :** A Configuration class is used to access the configuration XML and can be combined (if a var is repeated, last is used). Variables can also be expanded using system properties.
2. **Configuring the Development Environment :** All JAR's from top level Hadoop directory must be added to the IDE. Also, you can have local and cluster file configurations.
3. **GenericOptionsParser, Tool and ToolRunner :**
   i. GenericOptionsParser interprets Hadoop command-line options and sets them on a Configuration object.
   ii. Tool is an interface to use the above class.
4. **Writing Unit Tests :**
   i. **Mapper Unit Test :** Because Mapper and Reducers writes to Context files (instead of returning the result) a mock for the Context object is needed. We create the context object passing to the static mock method the class. Then we use it normally.
   ii. **Reducer Unit Test :** Reducer unit test is similar to mapper unit test.
5. **Running locally and in a cluster on Test Data :**
   i. Locally Using the Tool interface you could write a driver to configure the local job.
   ii. Cluster No code changes are needed, just to pack the Jar.
6. **The MapReduce Web UI :** It consists of following information :
   i. Hadoop installation : version, compilation, jobtracker state.
   ii. Summary of the cluster : capacity, utilization, mr running, jobs, tasktrackers, slots, blacklisted Tasktrackers.
   iii. Job Scheduler : Running and failed jobs with id's, owner, name.
   iv. Link to Jobtracker Logs : historic.
7. **Hadoop Logs :** Logfiles can be found on the local fs of each TaskTracker and if JVM reuse is enabled, each log accumulates the entire JVM run. Anything written to standard output or error is directed to the relevant logfile.
8. **Tuning a Job to improve performance :** After the program is working, you may wish to do some tuning, first by running through some standard checks for making MapReduce programs faster and then by doing task profiling.

**Que 2.18.** **How to write a program for MapReduce application ?**



**Answer**

1. Writing a program in MapReduce follows a certain pattern.
2. You start by writing your map and reduce functions, ideally with unit tests to make sure they do what you expect.
3. Then you write a driver program to run a job, which can run from your IDE using a small subset of the data to check that it is working.
4. If it fails, you can use your IDE's debugger to find the source of the problem.
5. With this information, you can expand your unit tests to cover this case and improve your mapper or reducer to handle such input correctly.
6. When the program runs as expected against the small dataset, you are ready to unleash it on a cluster.
7. Running against the full dataset is likely to expose some more issues, which you can fix as before, by expanding your tests and mapper or reducer to handle the new cases.
8. After the program is working, you may wish to do some tuning, first by running through some standard checks for making MapReduce programs faster and then by doing task profiling.
9. Profiling distributed programs is not easy, but Hadoop has hooks to aid the process.

**Que 2.19.** **Write a short note on : Unit tests with MRUnit.**

**Answer**

1. Testing and debugging multi threaded programs is hard.
2. Now take the same programs and massively distribute them across multiple JVMs deployed on a cluster of machines and the complexity goes off the roof.
3. One way to overcome this complexity is to do testing in isolation and catch as many bugs as possible locally.
4. MRUnit is a testing framework that lets you test and debug MapReduce jobs in isolation without spinning up a Hadoop cluster.
5. MRUnit provides a powerful and light-weight approach to do test-driven development.
6. This makes it easy to develop as well as to maintain Hadoop MapReduce code bases.
7. MRUnit supports testing Mappers and Reducers separately as well as testing MapReduce computations as a whole.
8. MRUnit allows you to do TDD (Test Driven Development) and write lightweight unit tests which accommodate Hadoop's specific architecture and constructs.

**Que 2.20.** **Write a short note on : Test data and local tests in MapReduce.**

**Answer**

1. After getting the mapper and reducer working on controlled inputs, the next step is to write a job driver.
2. This job driver is then executed on some test data on a development environment.
3. To run a job, Hadoop comes with a local job runner.
4. It is a cut-down version of the MapReduce execution engine for running MapReduce jobs in a single JVM.
5. It is designed for testing and is very convenient for use in an IDE.
6. However this local job runner is only designed for simple testing of MapReduce programs, so it differs from full MapReduce implementation.
7. The main difference is that it can't run more than one reducer.
8. The local job runner is enabled by a configuration setting.

**Que 2.21.** **How does Hadoop executes a MapReduce program ?**

**Answer**

1. We can run a MapReduce job with a single method call: submit() on a Job object.
2. Now Hadoop executes a MapReduce program depending on following configuration settings :
   - **A. Hadoop up to 0.20 release series :**
     - i. In releases of Hadoop up to 0.20 release series, mapred.job.tracker determines the means of execution.
     - ii. If this configuration property is set to local then the local job runner is used.
     - iii. If this configuration property is set to a colon-separated host and port pair, then the property is interpreted as a jobtracker address.
   - **B. Hadoop 0.23.0 release series :**
     - i. In Hadoop 0.23.0, MapReduce 2 implementation was introduced.
     - ii. It is built on a system called YARN.
     - iii. In this configuration property takes the values local (for the local job runner), classic (for the "classic" MapReduce framework), and yarn (for the new framework).



**Que 2.22.** **Explain anatomy of job run in classic MapReduce (MapReduce 1).**

**Answer**

1. A job run in classic MapReduce is shown in Fig. 2.22.1.
2. On the top level, there are four independent entities : client, jobtracker, tasktrackers, and distributed filesystem.

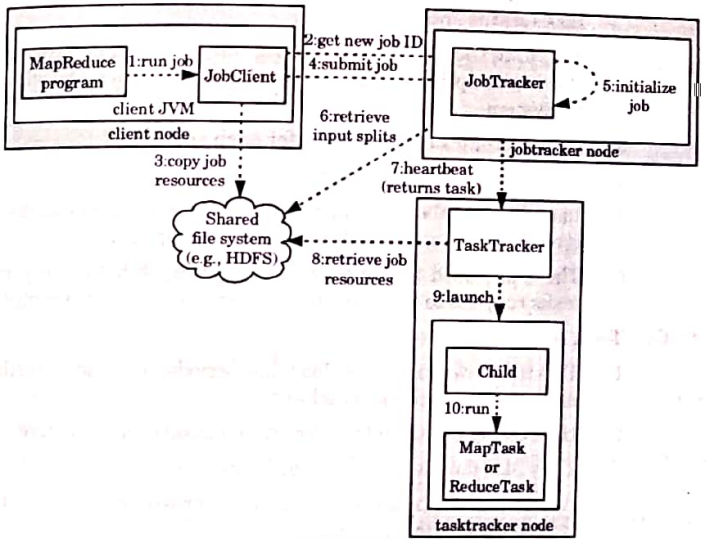



**Fig. 2.22.1.**

3. The job run in classic MapReduce consists of following :
   - **A. Job Submission :**
     1. MapReduce program is submitted to JobClient which will then be submitted to HDFS.
     2. After submitting the job, the job's progress is polled once every second.
     3. This progress is then reported to the console.
     4. When the job is completed successfully the job counters are displayed.
     5. If the job fails, the error that caused the job to fail is logged to the console.
     6. The job submission process does the following :
        - i. Asks the jobtracker for a new job ID.

ii. Checks the output specification of the job.
iii. Computes the input splits for the job.
iv. Copies the resources needed to run the job to the jobtracker's filesystem.
v. Tells the jobtracker that the job is ready for execution.

**B. Job Initialization :**

1. Initialization involves creating an object to represent the job being run and bookkeeping information to keep track of the tasks status and progress.
2. To create the list of tasks to run, the job scheduler retrieves the input splits computed by the client from the shared filesystem.
3. It then creates one map task for each split.
4. Tasks are given IDs at this point.
5. In addition to the map and reduce tasks, two further tasks are created : a job setup task and a job cleanup task.
6. These are used to run code to setup the job before any map tasks run, and to cleanup after all the reduce tasks are complete.

**C. Task Assignment :**

1. Tasktrackers run a simple loop that periodically sends heartbeat method calls to the jobtracker.
2. Heartbeats tell the jobtracker that a tasktracker is alive.
3. They also double as a channel for messages.
4. As a part of the heartbeat, a tasktracker will indicate whether it is ready to run a new task.
5. If it is ready to run a new task, the jobtracker will allocate it a task.
6. This will be communicated to the tasktracker using the heartbeat return value.

**D. Task Execution :**

1. Now that the tasktracker has been assigned a task, the next step is for it to run the task.
2. First, it localizes the job JAR by copying it from the shared filesystem to the tasktracker's filesystem.
3. It also copies any files needed from the distributed cache by the application to the local disk.
4. Second, it creates a local working directory for the task, and un-jars the contents of the JAR into this directory.
5. Third, it creates an instance of TaskRunner to run the task.



6. TaskRunner launches a new Java Virtual Machine to run each task.

**E. Job Completion :**

1. When the jobtracker receives a notification that the last task for a job is complete it changes the status of the job to "successful."
2. When the jobtracker learns that the job has completed successfully it prints a message to tell the user.
3. The jobtracker also sends an HTTP job notification.
4. Last, the jobtracker cleans up its working state for the job and instructs tasktrackers to do the same.

**Que 2.23. How the scalability shortcomings of classic MapReduce is overcome by YARN ?**

**Answer**

1. For very large clusters (4000 nodes and higher), the classic MapReduce system faces issue of scalability bottlenecks.
2. In 2010 work began to design the next generation of MapReduce. This next generation MapReduce was YARN (Yet Another Resource Negotiator).
3. YARN overcame the scalability shortcomings by splitting the responsibilities of jobtracker into separate entities.
4. The jobtracker takes care of both job scheduling and task progress monitoring.
5. YARN separates these two roles into two independent daemons : a resource manager and an application master.
6. The resource manager manages the use of resources across the cluster.
7. The application master manages the lifecycle of applications running on the cluster.

**Que 2.24. Explain anatomy of job run in YARN (MapReduce 2).**

**Answer**

1. A job run in YARN (MapReduce 2) is shown in Fig. 2.24.1.
2. On the top level, there are five independent entities: client, YARN resource manager, YARN node managers, MapReduce application master, and distributed filesystem.

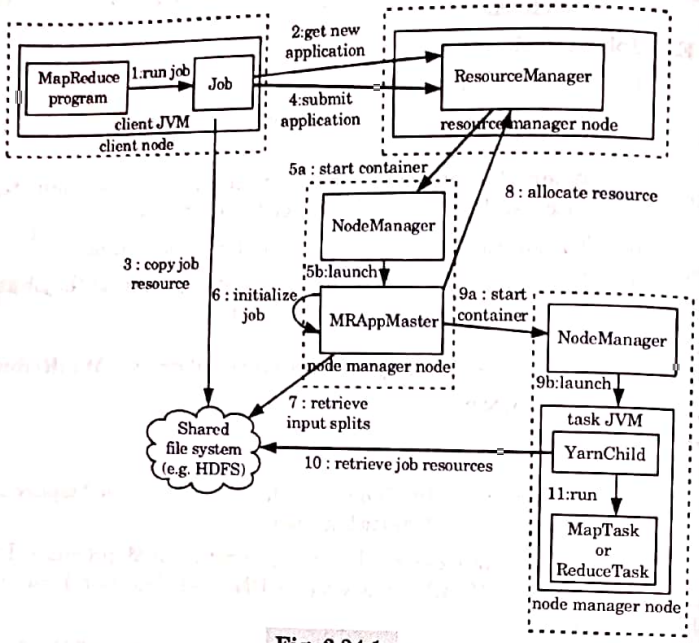



**Fig. 2.24.1.**

3. The job run in YARN (MapReduce 2) consists of following :

**A. Job Submission :**

1. Jobs are submitted in MapReduce 2 using the same user API as classic MapReduce.
2. ClientProtocol in MapReduce 2 is activated when mapreduce.framework.name is set to yarn.
3. The new job ID is retrieved from the resource manager.
4. The job client checks the output specification of the job; computes input splits and copies job resources to HDFS.
5. Finally, the job is submitted on the resource manager.

**B. Job Initialization :**

1. When the resource manager receives a call it hands off the request to the scheduler.
2. The scheduler allocates a container.
3. Under the node manager's management the resource manager then launches the application master's process.



4. The application master initializes the job by creating a number of bookkeeping objects to keep track of the job's progress.
5. Next, it retrieves the input splits computed in the client from the shared filesystem.
6. It then creates a map task object for each split, and a number of reduce task objects.
7. After this the application master decides how to run the tasks that make up the MapReduce job.
8. If the job is small, the application master may choose to run them in the same JVM as itself.
9. Such a job runs as an uber task.

**C. Task Assignment :**

1. If the job is not running as an uber task, then the application master requests containers for all the map and reduce tasks from the resource manager.
2. Each request includes information about each map task's data locality.
3. The scheduler uses this information to make scheduling decisions.
4. In an ideal case it attempts to place tasks on data-local nodes.
5. Requests also specify memory requirements for tasks.

**D. Task Execution :**

1. Once a task has been assigned a container, the application master starts the container by contacting the node manager.
2. The task is executed by a Java application.
3. Before the Java application can run the task it localizes the resources that the task needs.
4. Finally, it runs the map or reduce task.

**E. Job Completion :**

1. Every five seconds the client poll the application master for progress and checks whether the job has completed.
2. This polling interval can be set using configuration property.
3. It also supports notification of job completion via an HTTP callback.
4. On job completion the application master and the task containers clean up their working state.

## PART-7

*Failures, Job Scheduling, Shuffle and Sort, Task Execution, MapReduce Types, Input Formats, Output Formats, Map Reduce Features, Real-World Map Reduce.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 2.25. What are various failures in classic MapReduce (MapReduce 1) ?**

**Answer**

In classic MapReduce (MapReduce 1) following are the three failures modes :

**A. Task Failure :**

1. Tasktracker marks tasks as failed in following situations :
   i. User code in the map or reduce task throws a runtime exception.
   ii. If streaming process exits with a nonzero exit code.
   iii. Sudden exit of child JVM.
   iv. Tasktracker notices that it has not received progress update for a while and proceeds to mark the task as failed.
2. When jobtracker is notified of task attempt that has failed, it will reschedule execution of the task.
3. If the task fails four times or more, it will not be retried further.
4. User may also kill or fail task attempts using the Web UI or the command line.

**B. Tasktracker Failure :**

1. Failure of tasktracker is another failure mode.
2. If tasktracker fails by crashing, or running very slowly, it will stop sending heartbeats to jobtracker.
3. Jobtracker will notice tasktracker that has stopped sending heartbeats and remove it from its pool of tasktrackers to schedule tasks on.



4. Tasktracker can also be blacklisted by jobtracker, even if the tasktracker has not failed.
5. If more than four tasks from the same job fail on particular tasktracker, then the jobtracker records this as a fault.
6. Blacklisted tasktrackers are not assigned tasks, but they continue to communicate with the jobtracker.
7. Faults expire overtime (at rate of one per day), so tasktrackers get chance to run jobs again simply by leaving them running.

**C. Jobtracker Failure :**

1. It is the most serious failure mode.
2. Hadoop has no mechanism for dealing with failure of the jobtracker.
3. It is a single point of failure, so in this case the job fails.
4. However, this failure mode has a low chance of occurring.

**Que 2.26. What are various failures in YARN (MapReduce 2) ?**

**Answer**

In YARN (MapReduce 2) following are the four failures modes :

**A. Task Failure :**

1. Failure of the running task is similar to the classic case.
2. The tasks are marked as failed in following situations :
   i. Runtime exceptions and sudden exits of the JVM are propagated back to the application master.
   ii. If hanging tasks are noticed by the application master by the absence of a ping over the umbilical channel.
   iii. A task is marked as failed after four attempts.

**B. Application Master Failure :**

1. An application master sends periodic heartbeats to the resource manager.
2. In the event of application master failure, the resource manager will detect the failure.
3. The resource manager will start a new instance of the application master running in a new container.
4. Also the client polls the application master for progress reports.
5. If its application master fails the client needs to locate the new instance.

6. The client will go back to the resource manager to ask for the new application master's address.

**C. Node Manager Failure :**

1. If a node manager fails, then it will be removed from the resource manager's pool of available nodes.
2. Node managers may be blacklisted if the number of failures for the application is high.
3. Blacklisting is done by the application master.

**D. Resource Manager Failure :**

1. If the resource manager fails, then neither jobs nor task containers can be launched.
2. The resource manager was designed to be able to recover from crashes, by using a checkpointing mechanism to save its state to persistent storage.
3. The state consists of the node managers in the system and the running applications.

**Que 2.27. What are the types of schedulers in MapReduce ?**

**Answer**

Following are the types of schedulers in MapReduce :

**1. Capacity scheduler :**

i. In capacity scheduler, we have multiple job queues for scheduling our tasks.

ii. The capacity scheduler allows multiple occupants to share a large size Hadoop cluster.

iii. In capacity scheduler corresponding for each job queue, we provide some slots or cluster resources for performing job operation.

iv. Each job queue has its own slots to perform its task.

v. In case we have tasks to perform in only one queue then the tasks of that queue can access the slots of other queues also as they are free to use, and when the new task enters to some other queue then jobs in running in its own slots of the cluster are replaced with its own job.

**2. Fair scheduler :**

i. The Fair scheduler is similar to that of the capacity scheduler. The priority of the job is kept in consideration.



ii. With the help of Fair scheduler, the YARN applications can share the resources in the large Hadoop Cluster and these resources are maintained dynamically so no need for prior capacity.

iii. The resources are distributed in such a manner that all applications within a cluster get an equal amount of time.

iv. Fair scheduler takes scheduling decision on the basis of memory; we can configure it to work with CPU also.

**Que 2.28. What are the advantages and disadvantages of different types of scheduler ?**

**Answer**

Following are the advantages and disadvantages of different types of scheduler :

**Advantage of Capacity scheduler :**

1. Best for working with multiple clients or priority jobs in a Hadoop cluster.
2. Maximizes throughput in the Hadoop cluster.

**Disadvantage of Capacity scheduler :**

1. More complex.
2. Not easy to configure for everyone.

**Advantage of Fair scheduler :**

1. Resources assigned to each application depend upon its priority.
2. It can limit the concurrent running task in a particular queue.

**Disadvantage of Fair scheduler :**

1. The configuration is required.

**Que 2.29. What is shuffle and sort in Hadoop MapReduce ?**

**Answer**

1. Shuffle phase in Hadoop transfers the map output from Mapper to a Reducer in MapReduce.
2. Sort phase in MapReduce covers the merging and sorting of map outputs.
3. Data from the mapper are grouped by the key, split among reducers and sorted by the key.
4. Every reducer obtains all values associated with the same key.
5. Shuffle and sort phase in Hadoop occur simultaneously and are done by the MapReduce framework.

**A. Shuffle in MapReduce :**

1. The process by which the system performs the sort and transfers the map output to the reducer as input is known as shuffle.
2. MapReduce shuffle phase is necessary for the reducers; otherwise, they would not have any input.
3. As shuffle can start even before the map phase has finished so this saves some time and completes the tasks in lesser time.

**B. Sort in MapReduce :**

1. The keys generated by the mapper are automatically sorted by MapReduce Framework.
2. Sorting in Hadoop helps reducer to easily distinguish when a new reduce task should start.
3. This saves time for the reducer.
4. Reducer starts a new reduce task when the next key in the sorted input data is different than the previous.
5. Each reduce task takes key-value pairs as input and generates key-value pair as output.

**Que 2.30. Write a short note on :**

**i. Speculative Execution.**
**ii. Task JVM Reuse.**
**iii. Skipping Bad Records.**

**Answer**

**A. Speculative Execution :**

1. The MapReduce model break jobs into tasks and run these tasks in parallel to make the overall job execution time smaller.
2. This makes job execution time sensitive to slow-running tasks.
3. Tasks may be slow for various reasons, including hardware degradation or software misconfiguration.
4. Hadoop doesn't try to diagnose and fix slow-running tasks.
5. Instead, it tries to detect when a task is running slower and launches another, equivalent, task as a backup.
6. This is termed as speculative execution of tasks.
7. A speculative task is launched only for tasks that have been running for some time (at least a minute) and have failed to make as much progress as the other tasks from the job.
8. Speculative execution is an optimization.
9. It is not a feature to make jobs run more reliably.



**B. Task JVM Reuse :**

1. Starting a new JVM for each task can take around a second.
2. This is insignificant for jobs that run for a minute or so.
3. However, jobs that have a large number of very short-lived tasks can achieve significant performance gains if the JVM is reused for subsequent tasks.
4. When the JVM is reused the JVM runs tasks sequentially.
5. Tasks that are CPU-bound may also benefit from task JVM reuse.
6. These tasks take advantage of runtime optimizations applied by the HotSpot JVM.
7. Another place where a shared JVM is useful is for sharing state between the tasks of a job.

**C. Skipping Bad Records :**

1. Large datasets often have corrupt records.
2. If a small percentage of records are corrupt, then skipping them may not significantly affect the result.
3. We can use Hadoop's optional skipping mode for automatically skipping bad records.
4. When skipping mode is enabled, tasks report the records being processed back to the tasktracker.
5. When the task fails, the tasktracker retries the task, skipping the records that caused the failure.
6. Skipping mode is turned on for a task only after it has failed twice.
7. Skipping mode can detect only one bad record per task attempt, so this mechanism is appropriate only for detecting occasional bad records.

**Que 2.31. What are the different types of input formats in Hadoop ?**

**Answer**

Following the different input formats available :

**A. Input Splits and Records :**

1. An input split is a chunk of the input that is processed by a single map.
2. Each map processes a single split.
3. Each split is divided into records, and the map processes each record in turn.

4. Splits and records are logical.
5. In a database context, a split might correspond to a range of rows from a table and a record to a row in that range.

**B. Text Input :** Following are different InputFormats that Hadoop provides to process text :

1. **TextInputFormat :** TextInputFormat is the default InputFormat. Each record is a line of input. The key is the byte offset within the file of the beginning of the line. The value is the contents of the line and is packaged as a Text object.
2. **KeyValueTextInputFormat :** It is common for each line in a file to be a key-value pair, separated by a delimiter. To interpret such files correctly, KeyValueTextInputFormat is appropriate.
3. **NLineInputFormat :** If you want your mappers to receive a fixed number of lines of input, then NLineInputFormat is used.

**C. Binary Input :** Following are different binary formats Hadoop MapReduce supports :

1. **SequenceFileInputFormat :** To use data from sequence files as the input to MapReduce, you use SequenceFileInputFormat. The keys and values are determined by the sequence file, and you need to make sure that your map input types correspond.
2. **SequenceFileAsTextInputFormat :** It is a variant of SequenceFileInputFormat that converts the sequence file's keys and values to Text objects. This format makes sequence files suitable input for Streaming.
3. **SequenceFileAsBinaryInputFormat :** It is a variant of SequenceFileInputFormat that retrieves the sequence file's keys and values as opaque binary objects.

**D. Multiple Inputs :**

1. Over time the data format evolves. So we have to write our mapper to cope with all of our legacy formats.
2. Or, we have data sources that provide the same type of data but in different formats.
3. These cases are handled easily by using the MultipleInputs class, which allow us to specify the InputFormat and Mapper to use on a per-path basis.

**E. Database Input (and Output) :**

1. DBInputFormat is an input format for reading data from a relational database, using JDBC.



**Que 2.32. What are the different types of output formats in Hadoop ?**

**Answer**

Following the different output formats available :

**A. Text Output :**

1. The default output format, TextOutputFormat, writes records as lines of text.
2. Its keys and values may be of any type.
3. Each key-value pair is separated by a tab character.

**B. Binary Output :** Following are different binary formats Hadoop MapReduce supports :

1. **SequenceFileOutputFormat :** It writes sequence files for its output. This is a good choice of output if it forms the input to a further MapReduce job, since it is compact and is readily compressed.
2. **SequenceFileAsBinaryOutputFormat :** It is the counterpart to SequenceFileAsBinaryInput Format, and it writes keys and values in raw binary format into a SequenceFile container.
3. **MapFileOutputFormat :** It writes MapFiles as output. The keys in a MapFile must be added in order, so you need to ensure that your reducers emit keys in sorted order.

**C. Multiple Outputs :**

1. There is sometimes a need to have more control over the naming of the files or to produce multiple files per reducer.
2. MapReduce comes with the MultipleOutputs class to help you do this.
3. MultipleOutputs allows you to write data to files whose names are derived from the output keys and values.
4. This allows each reducer to create more than a single file.

**D. Lazy Output :**

1. Some applications prefer that empty files not be created, which is where Lazy OutputFormat helps.
2. It is a wrapper output format that ensures that the output file is created only when the first record is emitted for a given partition.

**E. Database Output :**

1. DBOutputFormat is the output format which is useful for dumping job outputs into a database.

**Que 2.33. What are the various advanced features of MapReduce?**

**Answer**

Following are the various advanced features of MapReduce :

**A. Counters :**

1. Counters are a useful channel for gathering statistics about the job.
2. They are also useful for problem diagnosis.
3. Following are various types of counters :

**a. Built-in Counters :**

1. Hadoop maintains some built-in counters for every job, which report various metrics for our job.
2. Several groups for the built-in counters are MapReduce Task Counters, Filesystem Counters, FileInput-Format Counters, FileOutput-Format Counters, Job Counters, etc.

**b. User-Defined Java Counters :**

1. MapReduce allows user code to define a set of counters, which are then incremented as desired in the mapper or reducer.
2. Counters are defined by a Java enum, which serves to group related counters.
3. A job may define an arbitrary number of enums, each with an arbitrary number of fields.

**c. User-Defined Streaming Counters :**

1. A Streaming MapReduce program can increment counters by sending a specially formatted line to the standard error stream.

**B. Sorting :**

1. The ability to sort data is at the core of MapReduce.
2. Even if application isn't concerned with sorting, it may be able to use the sorting stage to organize its data.
3. Following are different ways of sorting datasets that MapReduce provides :

i. Preparation

ii. Partial Sort



iii. Total Sort

iv. Secondary Sort

**C. Joins :**

1. MapReduce can perform joins between large datasets.
2. How we implement the join depends on how large the datasets are and how they are partitioned.
3. If the join is performed by the mapper, it is called a map-side join.
4. If it is performed by the reducer, it is called a reduce-side join.
5. If both datasets are too large, then we can still join them using MapReduce, depending on how the data is structured.

☺☺☺

# 3 UNIT

# HDFS (Hadoop Distributed File System)

## CONTENTS







## PART-1

*HDFS (Hadoop Distributed File System), Design of HDFS, HDFS Concepts.*

### Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 3.1. Discuss the design of Hadoop Distributed File System (HDFS).**

**Answer**

1. The Hadoop Distributed File System (HDFS) was designed for Big data processing.
2. Although it can support many users simultaneously, HDFS is not designed as a true parallel file system.
3. Its design assumes a large file write-once/read-many model.
4. This enables other optimizations and relaxes many of the concurrency and coherence overhead requirements of a true parallel file system.
5. The design of HDFS is based on the design of the Google File System (GFS).
6. HDFS is designed for data streaming where large amounts of data are read from disk in bulk.
7. The HDFS block size is typically 64MB or 128MB.
8. In addition, due to the sequential nature of the data, there is no local caching mechanism.
9. The most interesting aspect of HDFS is its data locality.
10. A principal design aspect of Hadoop MapReduce is the emphasis on moving the computation to the data rather than moving the data to the computation.
11. This distinction is reflected in how Hadoop clusters are implemented.
12. HDFS is designed to work on the same hardware as the compute portion of the cluster.
13. That is, a single server node in the cluster is both a computation engine and a storage engine for the application.
14. HDFS has a redundant design that can tolerate system failure and still provide the data needed by the compute part of the program.

**Que 3.2.** **Discuss HDFS concepts.**

**Answer**

Following are various HDFS concepts :

**A. Blocks :**

1. A Block is the minimum amount of data that can be read or written.
2. HDFS blocks are 128 MB by default and this is configurable.
3. Files in HDFS are broken into block-sized chunks, which are stored as independent units.

**B. Namenode :**

1. NameNode is the centerpiece of HDFS.
2. NameNode is also known as the Master.
3. NameNode only stores the metadata of HDFS – the directory tree of all files in the filesystem, and tracks the files across the cluster.
4. NameNode does not store the actual data or the dataset. The data itself is actually stored in the DataNodes.
5. 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.
6. NameNode is so critical to HDFS and when the NameNode is down, Hadoop cluster is inaccessible and considered down.
7. NameNode is a single point of failure in Hadoop cluster.

**C. DataNode :**

1. DataNode is responsible for storing the actual data in HDFS.
2. DataNode is also known as the Slave.
3. NameNode and DataNode are in constant communication.
4. When a DataNode starts up it announce itself to the NameNode along with the list of blocks it is responsible for.
5. When a DataNode is down, it does not affect the availability of data or the cluster.
6. NameNode will arrange for replication for the blocks managed by the DataNode that is not available.
7. DataNode is usually configured with a lot of hard disk space, because the actual data is stored in the DataNode.

**D. HDFS Federation :**

1. HDFS Federation improves the existing HDFS architecture through a clear separation of NameNode and storage, enabling generic block storage layer.



2. It enables support for multiple NameNodes in the cluster to improve scalability and isolation.
3. Federation also opens up the architecture, expanding the applicability of HDFS cluster to new implementations and use cases.

**E. HDFS High-Availability :**

1. High availability refers to the availability of system or data in the wake of component failure in the system.
2. The high availability feature in Hadoop ensures the availability of the Hadoop cluster without any downtime, even in unfavorable conditions like NameNode failure, DataNode failure, machine crash, etc.
3. It means if the machine crashes, data will be accessible from another path.

## PART-2

*Benefits and Challenges, Files Size, Block Size and Block Abstraction in HDFS.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 3.3.** **Describe the benefits of HDFS.**

**Answer**

Following are the benefits of HDFS :

**1. Scalable :**

i. Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel.

ii. Unlike traditional relational database systems (RDBMS) that cannot scale to process large amounts of data, Hadoop enables businesses to run applications on thousands of nodes involving thousands of terabytes of data.

**2. Cost effective :**

i. Hadoop also offers a cost effective storage solution for businesses' exploding data sets.

ii. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data.

3. **Flexible :**
   i. Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data.
   ii. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations or clickstream data.
4. **Fast :**
   i. Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster.
   ii. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing.
5. **Resilient to failure :**
   i. A key advantage of using Hadoop is its fault tolerance.
   ii. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

**Que 3.4.** **What are the challenges of HDFS ?**

**Answer**

Following are the challenges of HDFS :

1. **Issue with small files :**
   i. Hadoop distributed file system lacks the ability to efficiently support the random reading of small files because of its high capacity design.
   ii. Small files are the major problem in HDFS. A small file is significantly smaller than the HDFS block size (default 128MB).
2. **Slow processing speed :**
   i. In Hadoop, with a parallel and distributed algorithm, the MapReduce process large data sets.
   ii. There are tasks that we need to perform : Map and Reduce and, MapReduce requires a lot of time to perform these tasks thereby increasing latency.
   iii. Data is distributed and processed over the cluster in MapReduce which increases the time and reduces processing speed.
3. **Support for batch processing only :**
   i. Hadoop supports batch processing only, it does not process streamed data, and hence overall performance is slower.
   ii. The MapReduce framework of Hadoop does not leverage the memory of the Hadoop cluster to the maximum.



4. **No real-time data processing :**
   i. Apache Hadoop is for batch processing, which means it takes a huge amount of data in input, process it and produces the result.
   ii. Although batch processing is very efficient for processing a high volume of data, depending on the size of the data that processes and the computational power of the system, an output can delay significantly.
   iii. Therefore, Hadoop is not suitable for Real-time data processing.
5. **No delta iteration :**
   i. Hadoop is not so efficient for iterative processing, as Hadoop does not support cyclic data flow (*i.e.*, a chain of stages in which each output of the previous stage is the input to the next stage).

**Que 3.5.** **Write a short note on : File sizes in HDFS.**

**Answer**

1. HDFS supports large files and large number of files.
2. A typical HDFS cluster has tens of millions of files and a typical file in HDFS is several (hundreds) of MB to GB each, in size.
3. A typical file is 100 MB or larger.
4. The NameNode maintains the namespace metadata of the files such as the filename, directory name, user, group, permissions, etc.
5. Extremely small files are discouraged, as too many small files could degrade performance.
6. A file is considered small if its size is much less than the block size. For example, if the block size is 128MB and the file size is 1 MB to 50 MB, the file is considered a small file.
7. The HDFS is not designed for small files. The HDFS is designed with the goal of providing efficient streaming access to large files.
8. However, extremely large files are also discouraged as too large files could degrade performance.

**Que 3.6.** **Write a short note on : Block sizes in HDFS.**

**Answer**

1. HDFS stores files across the cluster by breaking them down in to fixed-size blocks.
2. These blocks are stored as independent units.
3. The size of these HDFS data blocks is 128 MB by default.
4. The size of these blocks can be configured as per our requirements.

5. Hadoop distributes these blocks on different slave machines, and the master machine stores the metadata about blocks location.
6. Block size of a data product can affect the performance of the filesystem operations.
7. So larger block sizes would be more effective if you are storing and processing very large files.
8. Block size of a data product can also affect the performance of MapReduce computations, as the default behavior of Hadoop is to create one Map task for each data block of the input files.

**Que 3.7. What is block abstraction in HDFS ? What are the benefits of using block abstraction ?**

**Answer**

**A. Block abstraction in HDFS :**

1. HDFS manages the stored data with block units similar to a standard filesystem.
2. Each block has a limited, maximum size configured by HDFS, which defines how flies that would span multiple blocks are divided.
3. The default block size is 128 MB. Each file is separated into 128 MB blocks when written on the HDFS.
4. A file that's smaller than the block size does not occupy the total block.
5. A 100 MB file keeps only 100 MB on one HDFS block. The block is an important abstraction of HDFS.
6. The blocks are distributed across multiple nodes, so that you can create a file larger than the disk size of a single node.
7. Thus, you can create any size of file using the abstraction of the blocks that are used to store the file.

**B. Benefits of using block abstraction :** Following are the benefits of using block abstraction :

1. Files can be bigger than individual disks.
2. Filesystem metadata does not need to be associated with each and every block.
3. Simplifies storage management – Easy to figure out the number of blocks which can be stored on each disk.
4. Fault tolerance and storage replication can be easily done on a per-block basis.



## PART-3

*Data Replication, How does HDFS Store, Read, and Write Files, Java Interface to HDFS.*

### Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 3.8. Write short note on data replication.**

**Answer**

1. Data replication in HDFS increases the availability of data at any point of time.
2. If any node containing a block of data which is used for processing crashes, we can get the same block of data from another node this is because of replication.
3. Replication is one of the major factors in making the HDFS a Fault tolerant system.
4. Replication ensures the availability of the data.
5. Replication is making a copy of something and the number of times we make a copy of that particular thing can be expressed as its Replication Factor.
6. The Replication Factor (RF) is equivalent to the number of nodes where data (rows and partitions) are replicated.
7. Data is replicated to multiple nodes.

**Que 3.9. Explain how HDFS stores data.**

**Answer**

1. HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients.
2. There are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.
3. HDFS exposes a file system namespace and allows user data to be stored in files.
4. A file is split into one or more blocks and these blocks are stored in a set of DataNodes.

5. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories.
6. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients.
7. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

**Que 3.10. Explain how read operation is performed in HDFS.**

**Answer**

1. A client initiates read request by calling 'open()' method of FileSystem object, it is an object of type DistributedFilesystem.
2. This object connects to NameNode using RPC and gets metadata information such as the locations of the blocks of the file.
3. In response to this metadata request, addresses of the DataNodes having a copy of that block is returned back.
4. Once addresses of DataNodes are received, an object of type FSDataInputStream is returned to the client.
5. FSDataInputStream contains DFSInputStream which takes care of interactions with DataNode and NameNode.
6. Data is read in the form of streams wherein client invokes 'read()' method repeatedly. This process of read() operation continues till it reaches the end of block.
7. Once the end of a block is reached, DFSInputStream closes the connection and moves on to locate the next DataNode for the next block.
8. Once a client has done with the reading, it calls a close() method.

**Que 3.11. Explain how write operation is performed in HDFS.**

**Answer**

1. The client calls the create() method on DistributedFileSystem to create a file.
2. DistributedFileSystem interacts with NameNode through the RPC call to create a new file in the filesystem namespace with no blocks associated with it.
3. The NameNode checks for the client privileges and makes sure that the file doesn't already exist.
4. The DistributedFileSystem then returns an FSDataOutputStream for the client where the client starts writing data. FSDataOutputstream, in turn, wraps a DFSOutputStream, which handles communication with the DataNodes and NameNode.



5. As the client starts writing data, the DFSOutputStream splits the client's data into packets and writes it to an internal queue called the data queue.
6. DataStreamer uses this data queue. The DataStreamer streams the packet to the first DataNode in the pipeline, which stores each packet and forwards it to the second node in the pipeline.
7. The DFSOutputStream also maintains another queue of packets, called ack queue, which is waiting for the acknowledgment from DataNodes.
8. The client calls the close() method on the stream when he/she finishes writing data. Thus, before communicating the NameNode to signal about the file complete, the client close() method's action pushes the remaining packets to the DataNode pipeline and waits for the acknowledgment.
9. As the Namenode already knows about the blocks (the file made of), so the NameNode only waits for blocks to be minimally replicated before returning successfully.

**Que 3.12. Write short note on Java interface to HDFS.**

**Answer**

1. Hadoop has an abstract notion of filesystems, of which HDFS is just one implementation.
2. The Java abstract class org.apache.hadoop.fs.FileSystem represents the client interface to a filesystem in Hadoop, and there are several concrete implementations.
3. Hadoop is written in Java, so most Hadoop filesystem interactions are mediated through the Java API.
4. The filesystem shell is a Java application that uses the Java FileSystem class to provide filesystem operations.

**Java Interface :**

**A. Reading Data from a Hadoop URL :**

1. A java.net.URL object is used to open a stream to read the data from Hadoop filesystem.
2. Java recognize Hadoop's hdfs URL by calling the setURLStreamHandlerFactory method on URL with an instance of FsUrlStreamHandlerFactory.
3. This method can only be called once per JVM.

**B. Reading Data Using the FileSystem API :**

1. Sometimes it is not possible to set a URLStreamHandlerFactory for the application.
2. In this case, we use the FileSystem API to open an input stream for a file.

**C. Writing Data :**

1. The FileSystem class has a number of methods for creating a file.

2. The simplest is the method that takes a Path object for the file to b created and returns an output stream.
3. There are also overloaded versions of this method.

**D. Directories :**

1. FileSystem provides following method to create a directory :
   public boolean mkdirs(Path f) throws IOException
2. This method creates all of the necessary parent directories.
3. It returns true if the directory was successfully created.

**E. Deleting Data :**

1. FileSystem provides following method to permanently remove file or directories :
   public boolean delete(Path f, boolean recursive) throws IOExceptio

## PART-4

*Command Line Interface, Hadoop File System Interface, Data Flow.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 3.13. Write short note on Command Line Interface (CLI).**

**Answer**

1. The HDFS can be manipulated through a command-line interface.
2. The command-line is one of the simplest and most familiar interface to HDFS.
3. The Hadoop Command-Line Interface is a user interface providing command-line options for monitoring Hadoop services and cluster management tasks.
4. The filesystem shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS).
5. Using command-line interface we can do all of the usual filesystem operations such as reading files, creating directories, moving files, deleting data, and listing directories.

**Que 3.14. Explain Hadoop Filesystem Interfaces.**

**Answer**

1. Hadoop is written in Java.
2. All Hadoop filesystem interactions are accessed through the Java API.



3. The filesystem shell uses the Java FileSystem class to provide filesystem operations.
4. Following are other filesystems interfaces :

**A. HTTP :**

1. There are two ways of accessing HDFS over HTTP.
2. First directly, where the HDFS daemons serve HTTP requests to clients.
3. Second via a proxy, which accesses HDFS on the client's behalf using DistributedFileSystem API.

**B. C :**

1. Hadoop provides a C library that mirrors the Java FileSystem interface.
2. It works using the Java Native Interface (JNI) to call a Java filesystem client.
3. The C API is very similar to the Java one.
4. But it typically lags the Java one, so newer features may not be supported.

**C. FUSE :**

1. FUSE stands for Filesystem in Userspace.
2. It allows filesystems implemented in user space to be integrated as a Unix filesystem.
3. Hadoop's Fuse-DFS contrib module allows HDFS to be mounted as a standard filesystem.
4. Fuse-DFS is implemented in C (using libhdfs) as the interface to HDFS.

## PART-5

*Data Ingest with Flume and SQOOP, Hadoop Archives.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 3.15. What is data ingestion ? Mention the challenges in data ingestion.**

**Answer**

1. Data ingestion is the process of transporting data from one or more sources to a target site for further processing and analysis.
2. Data ingestion is critical and should be emphasized for any big data project, as the volume of data is usually in terabytes or petabytes.

3. Nowadays, data sources are in different formats and produce data in high velocity.
4. Data ingestion is complex in Hadoop as data sources and processing are now in batch, stream, and real-time. This increases the complexity and management.

**Challenges in data ingestion :** Following are the challenges in data ingestion :

1. Multiple source ingestion
2. Streaming / real-time ingestion
3. Scalability
4. Parallel processing
5. Data quality
6. Machine data can be on a high scale in GB per minute

**Que 3.16. What is Flume ? How is data ingestion done in Flume?**

**Answer**

1. Flume is extremely popular data ingestion system.
2. It can ingest data from different multiple sources and put it in multiple destinations.
3. Flume provides a framework to handle and process data on a larger scale and it is very reliable.

**Data ingestion in Flume :**

1. Flume is usually described as distributed, reliable, scalable, manageable, and customizable to ingest and process data from different multiple data sources to multiple destinations.
2. Since Big data systems has multiple data sources and the data formats changes frequently this makes the design more difficult.
3. Flume is extremely efficient in handling such scenarios and provides a greater control over each data source and the processing layer.
4. Flume is adapted due to its capability to be highly reliable, flexible, customizable, extensible, and can work in a distributed manner in parallel to process big data.
5. Flume can be configured in three modes : single node, pseudo-distributed, and fully-distributed mode.

**Que 3.17. What is Sqoop ? How is data ingestion done in Sqoop?**

**Answer**

1. Sqoop is a tool designed to support bulk export and import of data into HDFS from structured data stores such as relational databases, enterprise data warehouses, and NoSQL systems.



2. Sqoop helps by providing a utility to import and export data in Hadoop from these data sources.
3. Sqoop helps in executing the process in parallel and therefore in much faster speed.
4. Sqoop can process bulk data transfers on HDFS, Hive, or HBase.

**Data ingestion in Sqoop :**

1. Sqoop utilizes connectors and drivers to connect with the underlying database source, and executes the import and export in multiple Mapper process, in order to execute the data in parallel and faster.
2. One of the important steps in configuring Sqoop is to get the driver and configure it with Sqoop.
3. Drivers are not shipped with Sqoop as some drivers are licensed, hence we have to get the JDBC driver of the database and keep it in the Sqoop library.
4. Connectors are required to optimize the data transfer by getting metadata information of the database.
5. Sqoop provides generic connectors that will work with databases such as MySQL, Oracle, PostgreSQL, DB2, and SQL Server, but are not optimal.

**Que 3.18. Differentiate between Sqoop, Flume, HDFS in Hadoop.**

**Answer**

| Sqoop | Flume | HDFS |
|---|---|---|
| Sqoop is used for importing data from structured data sources such as RDBMS. | Flume is used for moving bulk streaming data into HDFS. | HDFS is a distributed file system used by Hadoop ecosystem to store data. |
| Sqoop has a connector based architecture. | Flume has an agent-based architecture. | HDFS has a distributed architecture. |
| HDFS is a destination for data import using Sqoop. | Data flows to HDFS through zero or more channels. | HDFS is an ultimate destination for data storage. |
| Sqoop data load is not event-driven. | Flume data load can be driven by an event. | HDFS just stores data provided to it by whatsoever means. |
| To import data from structured data sources, we use Sqoop commands. | To load streaming data Flume should be used. | HDFS has its own built-in shell commands to store data into it. |

**Que 3.19. Write short note on Hadoop archives. Give limitations of Hadoop archives.**

**Answer**

1. Hadoop Archives (HAR) is used to address the namespace limitations associated with storing many small files.
2. HAR packs a number of small files into large files so that the original files can be accessed transparently (without expanding the files).
3. HAR increases the scalability of the system by reducing the namespace usage and decreasing the operation load in the NameNode.
4. This improvement is orthogonal to memory optimization in the NameNode and distributing namespace management across multiple NameNodes.
5. Hadoop Archive is also compatible with MapReduce, it allows parallel access to the original files by MapReduce jobs.

**Limitations of Hadoop Archives :**

1. Once an archive file is created, you cannot update the file to add or remove files. In other words, har files are immutable.
2. Archive file will have a copy of all the original files so once a .har is created it will take as much space as the original files.
3. When a .har file is given as an input to MapReduce job, the small files inside the .har file will be processed individually by separate mappers which is inefficient.

## PART-6

*Hadoop I/O, Compression, Serialization, Avro and File-Based Data Structure.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 3.20. Describe compression in Hadoop.**

**Answer**

1. File compression brings two major benefits : it reduces the space needed to store files and it speeds up data transfer.
2. When dealing with large volumes of data, both of these savings can be significant.



3. So it is very important to carefully consider how to use compression in Hadoop.
4. Following are the compression formats used in Hadoop :
   1. **GZIP :**
      i. It provides high compression ratio.
      ii. It uses high CPU resources to compress and decompress data.
      iii. Good choice for Cold data which is infrequently accessed.
      iv. Compressed data is not splittable and hence not suitable for MapReduce jobs.
   2. **BZIP2 :**
      i. It provides high compression ratio (even higher than GZIP).
      ii. It takes long time to compress and decompress data.
      iii. Good choice for Cold data which is infrequently accessed.
      iv. Compressed data is splittable.
      v. Even though the compressed data is splittable, it is generally not suited for MR jobs because of high compression/ decompression time.
   3. **LZO :**
      i. It provides low compression ratio.
      ii. Very fast in compressing and decompressing data.
      iii. Compressed data is splittable if an appropriate indexing algorithm is used.
   4. **SNAPPY :**
      i. It provides average compression ratio.
      ii. It aimed at very fast compression and decompression time.
      iii. Compressed data is not splittable if used with normal file like .txt.
      iv. Generally used to compress Container file formats like Avro and SequenceFile because the files inside a Compressed Container file can be split.

**Que 3.21. Write short note on serialization in Hadoop.**

**Answer**

1. Serialization refers to the conversion of structured objects into byte streams for transmission over the network or permanent storage on disk.
2. Serialization is mainly used in two areas of distributed data processing: interprocess communication and permanent storage (*i.e.*, data persistence).

3. Hadoop uses RPC to achieve inter-process communication. The serialization mechanism of RPC has the following characteristics :
   - i. **Compact :** Compact format can make full use of network bandwidth (the scarcest resource in data center) to speed up transmission.
   - ii. **Fast :** It can reduce the cost of serialization and deserialization.
   - iii. **Extensibility :** New parameters for method calls can be added at any time.
   - iv. **Supporting interoperability :** Client and server can be implemented in different languages.
4. Hadoop uses its own serialization format, Writables.
5. It is compact and fast, but not so easy to extend from languages other than Java.
6. Writables are central to Hadoop.
7. Most MapReduce programs use them for their key and value types.

**Que 3.22. Describe Avro in detail.**

**Answer**

1. Avro is a language-neutral data serialization system.
2. Avro deals with data formats that can be processed by multiple languages.
3. Avro is a preferred tool to serialize data in Hadoop.
4. Avro has a schema-based system.
5. A language-independent schema is associated with its read and write operations.
6. Avro serializes the data which has a built-in schema.
7. Avro serializes the data into a compact binary format, which can be deserialized by any application.
8. Avro uses JSON format to declare the data structures.
9. It supports languages such as Java, C, C++, C#, Python, and Ruby.

**Que 3.23. What are the features of Avro ?**

**Answer**

Following are the features of Avro :

1. Avro is a language-neutral data serialization system.
2. It can be processed by many languages (currently C, C++, C#, Java, Python, and Ruby).
3. Avro creates binary structured format that is both compressible and splittable. Hence it can be efficiently used as the input to Hadoop MapReduce jobs.



4. Avro provides rich data structures. For example, we can create a record that contains an array, an enumerated type, and a sub record.
5. Avro schemas defined in JSON facilitate implementation in the languages that already have JSON libraries.
6. Avro creates a self-describing file named Avro Data File, in which it stores data along with its schema in the metadata section.
7. Avro is also used in Remote Procedure Calls (RPCs). During RPC, client and server exchange schemas in the connection handshake.

**Que 3.24. Explain the working of Avro.**

**Answer**

To use Avro, we need to follow the given workflow :

**Step 1 :** Create schemas. Here we need to design Avro schema according to our data.

**Step 2 :** Read the schemas into our program. It is done in two ways :

i. **By Generating a Class Corresponding to Schema :** Compile the schema using Avro. This generates a class file corresponding to the schema

ii. **By Using Parsers Library :** We can directly read the schema using parsers library.

**Step 3 :** Serialize the data using the serialization API provided for Avro, which is found in the package org.apache.avro.specific.

**Step 4 :** Deserialize the data using deserialization API provided for Avro, which is found in the package org.apache.avro.specific.

## PART-7

*Hadoop Environment, Setting up a Hadoop Cluster, Cluster Specification, Cluster Setup and Installation.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 3.25. Write a short note on : Cluster Specification.**

**Answer**

1. Hadoop is designed to run on commodity hardware.
2. That means that you can choose commonly available standardized hardware from any vendor to build your cluster.

3. You are not tied to the proprietary offerings from a single vendor.
4. Also commodity hardware does not mean low-end machines.
5. Low-end machines often have cheap components, which have higher failure rates than more expensive machines.
6. When operating hundreds or thousands of machines, cheap components causes higher failure rate which results in greater maintenance cost.
7. On the other hand, large database class machines are not recommended either, since they don't score well on the price/performance curve.
8. Even though you would need fewer large database class machines build a cluster of comparable performance, if one machine fail it would have a bigger impact on the cluster.
9. Hardware specification for each cluster is different.
10. Hadoop is designed to use multiple cores and disks, so it will be able to take full advantage of more powerful hardware.
11. The bulk of Hadoop is written in Java, and can therefore run on any platform with a JVM.
12. For a small cluster (10 nodes), it is usually acceptable to run the namenode and the jobtracker on a single master machine.
13. As the cluster and the number of files stored in HDFS grow, the namenode needs more memory, so the namenode and jobtracker should be moved onto separate machines.

**Que 3.26.** **Write a short note on : Cluster Setup and Installation.**

**Answer**

1. After the hardware is step up the next step is to install the software needed to run Hadoop.
2. There are various ways to install and configure Hadoop.
3. We will install and configure Hadoop using Apache Hadoop distribution.
4. Following are the customizations needed to run Hadoop :

**A. Installing Java :**

1. Java 6 or later is required to run Hadoop.
2. The latest stable Sun JDK is the preferred option.
3. Although, Java distributions from other vendors may work too.

**B. Creating a Hadoop User :**

1. Its advisable to create a dedicated Hadoop user account.
2. This separates the Hadoop installation from other services running on the same machine.



3. For small clusters, we can make this user's home directory an NFS-mounted drive.
4. The NFS server is typically outside the Hadoop cluster.
5. If you use NFS, it is advisable to use autofs.
6. Autofs allows you to mount the NFS filesystem on demand and provides some protection against the NFS server failing.

**C. Installing Hadoop :**

1. Download the Hadoop Package.
2. Extract the Hadoop tar file.
3. Change the owner of the Hadoop files to be the hadoop user and group.
4. Hadoop is not installed in the hadoop user's home directory, as that may be an NFS-mounted directory.

**D. Testing the Installation :**

1. After creating an installation script, we test it by installing it on the machines in the cluster.
2. It will probably take a few iterations due to kinks in the install.
3. When it's working, you can proceed to configure Hadoop and give it a test run.

## PART-8

*Hadoop Configuration, Security in Hadoop, Administering Hadoop, HDFS Monitoring and Maintenance, Hadoop Benchmark, Hadoop in Cloud.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 3.27.** **Write a short note on : Hadoop Configuration.**

**Answer**

1. In the Hadoop environment, the Hadoop configuration command is very common.
2. It is used very widely.
3. It helps us to list out the number of files on the HDFS level.
4. Hadoop configuration is driven by two types of important configuration files :

i. Read-only default configuration
ii. Site-specific configuration

5. To configure the Hadoop cluster we need to configure the environment in which the Hadoop daemons execute as well as the configuration parameters for the Hadoop daemons.
6. conf/hadoop-env.sh script is used to do site-specific customization of the Hadoop daemons process environment.
7. The core-site.xml file informs Hadoop daemon where NameNode runs in the cluster.
8. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.
9. The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes.
10. We can configure hdfs-site.xml to specify default block replication and permission checking on HDFS.

**Que 3.28. Write a short note on : Security in Hadoop.**

**Answer**

1. In early versions of Hadoop it was assumed that HDFS and MapReduce clusters would be used within a secure environment.
2. So, the measures for restricting access were designed to prevent accidental data loss.
3. They were not designed to prevent unauthorized access to data.
4. Therefore a secure authentication mechanism was missing in Hadoop.
5. HDFS file permissions provide only a mechanism for authorization, which controls what a particular user can do to a particular file.
6. However, authorization is not enough by itself, since the system is still open to abuse via spoofing.
7. To meet regulatory requirements for data protection, secure authentication must be in place for shared clusters.
8. This situation led to the implementation of secure authentication for Hadoop.
9. Hadoop uses Kerberos to authenticate the user.
9. Kerberos doesn't manage permissions.
10. Its Hadoop's job to determine whether the user has permission to perform a given action.

**Que 3.29. How Kerberos is used to authenticate the user in Hadoop ?**



**Answer**

1. When using Kerberos following three steps are taken by a client to access a service :
   i. **Authentication :** The client authenticates itself to the Authentication Server; after this it receives a timestamped Ticket-Granting Ticket (TGT).
   ii. **Authorization :** The client uses this TGT to request a service ticket from the Ticket Granting Server.
   iii. **Service Request :** The client uses the service ticket to authenticate itself to the server that is providing the service.
2. The Authentication Server and the Ticket Granting Server together form the Key Distribution Center (KDC).
3. The authorization and service request steps are performed by the client on user's behalf.
4. The authentication step is carried out explicitly by the user.
5. The user uses the kinit command, which prompts for a password.
6. Since TGTs last for 10 hours by default there is no need to enter password every time we access HDFS.
7. It is also possible to provide single sign-on to Hadoop by automating authentication at operating system login time.

**Que 3.30. Write a short note on : Delegation Tokens.**

**Answer**

1. There are many client-server interactions in HDFS.
2. Each of these interactions must be authenticated.
3. A three-step Kerberos ticket exchange protocol presents a high load on the KDC on a busy cluster.
4. To overcome this Hadoop uses delegation tokens.
5. Delegation tokens allow users authenticated access without having to contact the KDC again.
6. Delegation tokens are created and used transparently by Hadoop on behalf of users.
7. A delegation token is generated by the server (namenode).
8. On the first RPC call to the namenode, the client has no delegation token.
9. So it uses Kerberos to authenticate, and as a part of the response it gets a delegation token from the namenode.
10. In subsequent calls, it presents the delegation token, which the namenode can verify and hence the client is authenticated to the server.

**Que 3.31. Explain HDFS monitoring in Hadoop.**

**Answer**

1. HDFS monitoring in Hadoop is an important part of system administration.
2. The purpose of monitoring is to detect when the cluster is not providing the expected level of service.
3. The master daemons namenodes and jobtracker are the most important to monitor.
4. In larger clusters failure of datanodes and tasktrackers is to be expected
5. So extra capacity is provided to the cluster. This helps cluster tolerate having a small percentage of dead nodes at any time.
6. Also, some administrators run test jobs on a periodic basis as a test of the cluster's health.
7. Following are various monitoring capabilities of Hadoop :
   - **A. Logging :** All Hadoop daemons produce logfiles that can be very useful for finding out what is happening in the system.
   - **B. Metrics :** The HDFS and MapReduce daemons collect information about events and measurements that are collectively known as metrics. Hadoop daemons usually collect metrics under several contexts.
   - **C. Java Management Extensions :** Java Management Extensions (JMX) is a standard Java API for monitoring and managing applications. Hadoop includes several managed beans (MBeans), which expose Hadoop metrics to JMX-aware applications.

**Que 3.32. Write a short note on : Maintenance in Hadoop.**

**Answer**

System maintenance operations such as routine administration procedures, commissioning and decommissioning Nodes, and upgrades are routine operations in any data center.

**A. Routine Administration Procedures :**

**a. Metadata backups :**

1. If the namenode's persistent metadata is lost, the entire filesystem is rendered unusable.
2. Therefore, it is critical to make backups of these files.
3. We should keep multiple copies of different ages (one hour, one day, one week, and one month, say) to protect against corruption.



**b. Data backups :**

1. In HDFS data loss can occur and hence a backup strategy is essential.
2. With the large data volumes that Hadoop can store, deciding what data to back up and where to store it is a challenge.
3. The key is to prioritize the data.
4. The highest priority is the data that cannot be regenerated and is critical to the business.
5. Data that is straightforward to regenerate or disposable is the lowest priority. We may choose not to make backups of this category of data.

**B. Commissioning and Decommissioning Nodes :**

1. In a Hadoop cluster we need to add or remove nodes from time to time.
2. For example, to grow the storage available to a cluster, we commission new nodes.
3. Conversely, sometimes we shrink a cluster. And to do so, we decommission nodes.
4. Also, it can sometimes be necessary to decommission a node if it is misbehaving.
5. Nodes normally run both a datanode and a tasktracker, and both are typically commissioned or decommissioned in tandem.

**C. Upgrades :**

1. Upgrading an HDFS and MapReduce cluster requires careful planning.
2. Part of the planning process should include a trial run on a small test cluster with a copy of data that you can afford to lose.
3. A trial run will allow you to familiarize yourself with the process and iron out any snags before running the upgrade procedure on a production cluster.

**Que 3.33. What do you understand by Hadoop benchmarks ?**

**Answer**

1. Benchmarks are tests to find out whether the cluster is set up correctly.
2. To get the best results, benchmarks are run on a cluster that is not being used by others.
3. Hadoop comes with several benchmarks that can be run very easily with minimal setup cost.
4. Benchmarks are packaged in the test JAR file.

5. Following are widely used Hadoop benchmarks :
   i. **Benchmarking HDFS with TestDFSIO :** TestDFSIO tests the I/O performance of HDFS. It does this by using a MapReduce job as a convenient way to read or write files in parallel.
   ii. **Benchmarking MapReduce with Sort :** Hadoop comes with a MapReduce program that does a partial sort of its input. It is very useful for benchmarking the whole MapReduce system, as the full input dataset is transferred through the shuffle.
   iii. **MRBench :** It runs a small job a number of times. It acts as a good counterpoint to sort, as it checks whether small job runs are responsive.
   iv. **NNBench :** It is useful for load testing namenode hardware.
   v. **Gridmix :** It is a suite of benchmarks designed to model a realistic cluster workload, by mimicking a variety of data-access patterns.

**Que 3.34. What does Hadoop in the cloud mean ? Give reasons to run Hadoop in the cloud.**

**Answer**

1. Hadoop in the cloud means : it is running Hadoop clusters on resources offered by a cloud provider.
2. This practice is normally compared with running Hadoop clusters on your own hardware, called on-premises clusters or "on-prem."
3. Although many organizations choose to run Hadoop in-house, it is also popular to run Hadoop in the cloud.

**Reasons to run Hadoop in the cloud :** Following are various reasons to run Hadoop in the cloud :

1. **Lack of space :** You don't have space to keep racks of physical servers, along with the necessary power and cooling.
2. **Flexibility :** Since everything is controlled through cloud provider APIs and web consoles changes in business needs can be scripted and put into effect based on current conditions.
3. **New usage patterns :** The flexibility of making changes in the cloud leads to new usage patterns that are otherwise impractical.
4. **Speed of change :** It is much faster to launch new cloud instances or allocate new database servers than to purchase, unpack, rack, and configure physical computers.
5. **Lower risk :** In the cloud, you can quickly and easily change how many resources you use, so there is little risk of undercommitment or overcommitment.
6. **Focus :** An organization using a cloud is free to focus on its core competencies to carry out its business.



7. **Worldwide availability :** The largest cloud providers have data centers around the world, ready for you from the start. You can use resources close to where you work, or close to where your customers are, for the best performance.
8. **Data storage requirements :** If you have data that is required by law to be stored within specific geographic areas, you can keep it in clusters that are hosted in data centers in those areas.
9. **Cloud provider features :** Each major cloud provider offers an ecosystem of features to support the core functions of computing, networking, and storage. To use those features most effectively, your clusters should run in the cloud provider as well.

☺☺☺

4 UNIT

# Hadoop Ecosystem and YARN

## CONTENTS







## PART-1

*Hadoop Ecosystem and YARN : Hadoop Ecosystem Components, Scheduler, Fair and Capacity, Hadoop 2.0 New Features, NameNode High Availability, HDFS Federation, MRv2, YARMN, Running MRv1 in YARN.*

### Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 4.1.** **Describe briefly Hadoop ecosystem components.**

**Answer**

Refer Q. 2.13, Page 2–12Q, Unit-2.

**Que 4.2.** **Describe briefly about Fair scheduler and Capacity schedules.**

**Answer**

Refer Q. 2.27, Page 2–28Q, Unit-2.

**Que 4.3.** **Why Hadoop 2.0 has added support for namenode high-availability ?**

**Answer**

1. In Hadoop the namenode is the sole repository of the metadata and file-to-block mapping.
2. If it fails, all clients, including MapReduce jobs, would be unable to read, write, or list files.
3. In such an event the whole Hadoop system would be out of service.
4. To recover from a failed namenode we have to start a new primary namenode with one of the filesystem metadata replicas.
5. We then configure datanodes and clients to use this new namenode.
6. However the new namenode is not able to serve requests immediately.
7. On large clusters the time taken for a namenode to serve requests can be 30 minutes or more.
8. This long recovery time is a problem for routine maintenance.

9. Hadoop 2.0 remedies this situation by adding support for HDFS hig availability.
10. In this implementation there is a pair of namenode in an active-passi configuration.
11. One node is active at a time, and the other node is in standby mode.
12. The active and standby nodes remain synchronized.
13. If the active name node fails, the standby node takes over and promote itself to active state.

**Que 4.4. Write a short note on : Hadoop Federation.**

**Answer**

1. The namenode keeps a reference to every file and block in the filesystem in memory.
2. This results in the limitation on maximum number of files a Hadoop cluster can store.
3. So, on very large cluster memory becomes the limiting factor for scaling.
4. In Hadoop 2.0, HDFS Federation is a way of partitioning the file system namespace over multiple separated namenodes.
5. Each namenode manages only an independent slice of the overall file system namespace.
6. These name nodes are part of a single cluster, but they are actually federated and do not require any coordination with each other.
7. HDFS Federation is useful for small clusters, for file system namespace isolation, as well as large clusters, for horizontal scalability.

**Que 4.5. Write a short note on : MRv2 (MapReduce 2).**

**Answer**

1. For very large clusters (4000 nodes and higher), the classic MapReduce system faces issue of scalability bottlenecks.
2. In 2010 work began to design the next generation of MapReduce. This next generation was MapReduce 2 or YARN (Yet Another Resource Negotiator).
3. YARN overcame the scalability shortcomings by splitting the responsibilities of jobtracker into separate entities.
4. The jobtracker takes care of both job scheduling and task progress monitoring.
5. YARN separates these two roles into two independent daemons : a resource manager and an application master.
6. Resource manager is fixed and static.



7. It performs node management for free and busy nodes for allocating the resources for Map and Reduce phases.
8. For every application, there is a separate application manager dynamically generated.
9. Application manager communicates with the resource manager.
10. Depending on the availability of data nodes it will assign the Map Phase and Reduce Phase to them.

**Que 4.6. Explain how to run MRv1 in yarn.**

**Answer**

1. Most of the MRv1 examples continue to work on YARN.
2. However they are now present in a newly versioned jar file.
3. The syntax to submit applications is similar to the MRv1 framework.
4. A minor difference in MRv2 is the use of the yarn command in the Hadoop-YARN bin folder rather than Hadoop.
5. Although submission of applications is supported using the Hadoop command in MRv2, the yarn command is still preferred.
6. YARN uses the ResourceManager web interface for monitoring applications running on a YARN cluster.
7. The ResourceManager UI shows the basic cluster metrics, list of applications, and nodes associated with the cluster.

## PART-2

*NoSQL Database, Introduction to NoSQL.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 4.7. Write short note on NoSQL database.**

**Answer**

1. A NoSQL refers to non-SQL or non-relational is a database that provides a mechanism for storage and retrieval of data.
2. This data is modeled in means other than the tabular relations used in relational databases.
3. NoSQL databases are used in real-time web applications and big data and their use are increasing over time.

4. NoSQL systems are also called Not only SQL to emphasize the fact that they may support SQL-like query languages.
5. A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines and finer control over availability.
6. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL.
7. The suitability of a given NoSQL database depends on the problem it should solve.
8. Data structures used by NoSQL databases are viewed as more flexible than relational database tables.

**Que 4.8. What are the advantages and disadvantages of NoSQL?**

**Answer**

**Advantages of NoSQL :**

1. **High scalability :**
   i. NoSQL database use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved in sharding.
2. **High availability :**
   i. Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

**Disadvantages of NoSQL :**

1. **Narrow focus :**
   i. NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality.
   ii. Relational databases are a better choice in the field of transaction management than NoSQL.
2. **Open-source :**
   i. NoSQL is open-source database.
   ii. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.
3. **Management challenge :**
   i. The purpose of big data tools is to make management of a large amount of data as simple as possible.
   ii. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.



4. **GUI is not available :** GUI mode tools to access the database is not flexibly available in the market.
5. **Backup :**
   i. Backup is a great weak point for some NoSQL databases like MongoDB.
   ii. MongoDB has no approach for the backup of data in a consistent manner.

**Que 4.9. What are the types of NoSQL ?**

**Answer**

Following are various types of NoSQL :

1. **Key-value pair based :**
   i. Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.
   ii. Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB (Binary Large Objects), string, etc.
2. **Column-based :**
   i. Column-oriented databases work on columns and are based on BigTable paper by Google.
   ii. Every column is treated separately. Values of single column databases are stored contiguously.
3. **Document-oriented :**
   i. Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document.
   ii. The document is stored in JSON or XML formats.
   iii. The value is understood by the DB and can be queried.
4. **Graph-based :**
   i. A graph type database stores entities as well the relations amongst those entities.
   ii. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes.
   iii. Every node and edge has a unique identifier.

## PART-3

*Mongo DB, Introduction, Data Types, Creating, Updating and Deleting Documents, Querying, Introduction to Indexing, Capped Collection.*

## Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 4.10. Write a short note on : MongoDB.**

**Answer**

1. MongoDB is a NoSQL database. It is an open-source, cross-platform, document-oriented database written in C++.
2. MongoDB provides high performance, high availability, and automatic scaling.
3. MongoDB was designed to scale out.
4. The document-oriented data model makes it easier to split data across multiple servers.
5. MongoDB automatically takes care of balancing data and load across a cluster, redistributing documents automatically and routing reads and writes to the correct machines.
6. The topology of a MongoDB cluster is transparent to the application.
7. This allows developers to focus on programming the application rather than scaling it.
8. Also, if the topology of an existing deployment needs to change the application logic can remain the same.

**Que 4.11. Describe different features of MongoDB.**

**Answer**

Following are the features of MongoDB :

**A. Indexing :**

1. MongoDB supports generic secondary indexes.
2. It provides unique, compound, geospatial, and full-text indexing capabilities.
3. Secondary indexes on hierarchical structures such as nested documents and arrays are also supported.
4. It enables developers to model applications in ways that best suit them.

**B. Aggregation :**

1. Based on the concept of data processing pipelines, MongoDB provides an aggregation framework.
2. Aggregation pipelines allow you to build complex analytics engines.



**C. Special collection and index types :**

1. MongoDB supports time-to-live (TTL) collections for data that should expire at a certain time.
2. MongoDB also supports partial indexes to increase efficiency and reduce the amount of storage space required.

**D. File storage :**

1. MongoDB supports an easy-to-use protocol for storing large files and file metadata.

**Que 4.12. Describe the data types used in MongoDB.**

**Answer**

Following are the most common data types used in MongoDB :

1. **Null :** The null type can be used to represent both a null value and a nonexistent field.
2. **Boolean :** The boolean type can be used for the values true and false.
3. **Number :** The shell defaults to using 64-bit floating-point numbers. For integers, we use the NumberInt or NumberLong classes, which represent 4-byte or 8-byte signed integers, respectively.
4. **String :** This is the most commonly used data type to store the data. String in MongoDB must be UTF-8 valid.
5. **Date :** This datatype is used to store the current date or time in UNIX time format. MongoDB stores dates as 64-bit integers representing milliseconds.
6. **Regular expression :** This data type is used to store regular expression.
7. **Array :** This type is used to store arrays or list or multiple values into one key.
8. **Embedded document :** A document can be used as the value for a key. This is called an embedded document. Documents can contain entire documents embedded as values in a parent document.
9. **Object ID :** This data type is used to store the document's ID. An object ID is a 12-byte ID for documents.
10. **Code :** MongoDB also makes it possible to store arbitrary JavaScript in queries and documents.

**Que 4.13. Explain how to update documents in MongoDB.**

**Answer**

1. Once a document is stored in the database, it can be modified using one of several update methods : updateOne, updateMany, and replaceOne.

2. updateOne and updateMany each take a filter document as their first parameter and a modifier document as the second parameter.
3. replaceOne also takes a filter as the first parameter, but as the second parameter replaceOne expects a document with which it will replace the document matching the filter.
4. Updating a document is atomic.
5. If two updates happen at the same time, whichever one reaches the server first will be applied, and then the next one will be applied.
6. Thus, conflicting updates can safely be sent in succession without any documents being corrupted.

**Que 4.14. Explain how to delete documents in MongoDB.**

**Answer**

1. In versions of MongoDB prior to 3.0, remove was the primary method for deleting documents.
2. MongoDB 3.2 drivers introduced the deleteOne and deleteMany methods.
3. The CRUD API in MongoDB provides deleteOne and deleteMany for deleting data from the database.
4. Both of these methods take a filter document as their first parameter.
5. The filter specifies a set of criteria to match against in removing documents.
6. To delete single document that match a filter, we use deleteOne.
7. deleteOne will delete the first document found that matches the filter.
8. However, we can also specify a filter that matches multiple documents in a collection.
9. To delete all the documents that match a filter, we use deleteMany.
10. Removing documents is usually a fairly quick operation using deleteMany.
11. However, if you want to clear an entire collection, it is better to use 'drop'.
12. Once data has been removed, it is gone forever. There is no way to undo a delete or drop operation or recover deleted documents.

**Que 4.15. Write a short note on : Queries in MongoDB.**

**Answer**

1. The find method is used to perform queries in MongoDB.
2. Querying returns a subset of documents in a collection, from no documents at all to the entire collection.



3. Which documents get returned is determined by the first argument to find, which is a document specifying the query criteria.
4. An empty query document (*i.e.*, {}) matches everything in the collection.
5. If find isn't given a query document, it defaults to empty query document.
6. When we start adding key/value pairs to the query document, we begin restricting our search.
7. Querying for a simple type is as easy as specifying the value that you are looking for.
8. Multiple conditions can be strung together by adding more key/value pairs to the query document, which gets interpreted as "condition1 AND condition2 AND ... AND conditionN."
9. If we need specific key/value pairs in a document returned, we can pass a second argument to find (or findOne) specifying the keys we want.
10. This reduces the amount of data and the time and memory used to decode documents.
11. There are some restrictions on queries.
12. The value of a query document must be a constant as far as the database is concerned. That is, it cannot refer to the value of another key in the document.

**Que 4.16. Describe indexing in MongoDB.**

**Answer**

1. A database index is similar to a book's index.
2. Instead of looking through the whole book, the database takes a shortcut and just looks at an ordered list.
3. This allows MongoDB to query orders of magnitude faster.
4. A query that does not use an index is called a collection scan, which means that the server has to look through whole database to find a query's results.
5. We want to avoid making the server do collection scans because the process is very slow for large collections.
6. To create an index, we use createIndex collection method.
7. Creating the index take no longer than few seconds, unless the collection is especially large.
8. An index can make a dramatic difference in query times.
9. The tricky part is to figure out which fields to index.
10. To choose which fields to create indexes for, look through your frequent queries and queries that need to be fast and try to find a common set of keys from those.

**Que 4.17. Describe capped collection.**

**Answer**

1. Capped collections are fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order.
2. Capped collections work in a way similar to circular buffers.
3. Once a collection fills its allocated space, it makes room for new documents by overwriting the oldest documents in the collection.
4. Capped collections guarantee preservation of the insertion order. As a result, queries do not need an index to return documents in insertion order.
5. Without this indexing overhead, capped collections can support higher insertion throughput.
6. To make room for new documents, capped collections automatically remove the oldest documents in the collection without requiring scripts or explicit remove operations.

## PART-4

*Spark : Installing Spark, Spark Application, Jobs, Stages and Tasks, Resilient Distributed Database, Anatomy of a Spark Job Run, Spark on YARN.*

### Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 4.18. What is Spark ? How to install Spark ?**

**Answer**

1. Spark is a fast and general processing engine compatible with Hadoop data.
2. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop InputFormat.
3. It is designed to perform both batch processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning.

**Step 1 : Downloading Apache Spark :**

i. Download the latest version of Spark by visiting the following link Download Spark.



ii. After downloading it, we will find the Spark tar file in the download folder.

**Step 2 : Installing spark :** Follow the steps given below for installing spark :

i. **Extracting Spark tar :** The following command for extracting the spark tar file.

$ tar xvf spark-1.3.1-bin-hadoop2.6.tgz

ii. **Setting up the environment for spark :** It means adding the location, where the spark software file are located to the PATH variable.

export PATH=$PATH:/usr/local/spark/bin

iii. Use the following command for sourcing the ~/.bashrc file.

$ source ~/.bashrc

**Step 3 : Verifying the spark installation :**

i. Write the following command for opening Spark shell.

$spark-shell

**Que 4.19. Explain the features of spark.**

**Answer**

Following are the features of spark :

1. **In-memory processing :**
   i. In-memory processing is faster when compared to Hadoop, as there is no time spent in moving data/processes in and out of the disk.
   ii. Spark is faster than MapReduce because everything is in memory.
2. **Stream processing :**
   i. Apache Spark supports stream processing, which involves continuous input and output of data.
   ii. Stream processing is also called real-time processing.
3. **Less latency :**
   i. Apache spark is faster than Hadoop, since it caches most of the input data in memory by the Resilient Distributed Dataset (RDD).
   ii. RDD manages distributed processing of data and the transformation of that data.
   iii. This is where spark does most of the operations such as transformation and managing the data.
   iv. Each dataset in an RDD is partitioned into logical portions, which can then be computed on different nodes of a cluster.
4. **Lazy evaluation :**
   i. Apache spark starts evaluating only when it is absolutely needed.

ii. This plays an important role in contributing to its speed.

5. **Less lines of code :**

i. Although Spark is written in both Scala and Java, the implementation is in Scala, so the numbers of lines are relatively lesser in Spark when compared to Hadoop.

**Que 4.20. Write a short note on : Spark applications, jobs, stages and tasks.**

**Answer**

1. Like MapReduce, Spark has the concept of a job.
2. A Spark job is more general than a MapReduce job.
3. It is made up of an arbitrary directed acyclic graph (DAG) of stages.
4. Each of arbitrary DAG is roughly equivalent to a map or reduce phase in MapReduce.
5. Stages are split into tasks by the Spark runtime.
6. They are run in parallel on partitions of an RDD spread across the cluster — just like tasks in MapReduce.
7. A job always runs in the context of an application that serves to group RDDs and shared variables.
8. An application can run more than one job, in series or in parallel.
9. It also provides the mechanism for a job to access an RDD that was cached by a previous job in the same application.

**Que 4.21. Write a short note on : Resilient Distributed Datasets (RDD).**

**Answer**

1. Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark.
2. It is an immutable distributed collection of objects.
3. There are three ways of creating RDDs: from an in-memory collection of objects, using a dataset from external storage, or transforming an existing RDD.
4. Spark provides two categories of operations on RDDs: transformations and actions.
5. A transformation generates a new RDD from an existing one.
6. An action triggers a computation on an RDD and does something with the results.
7. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.



8. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.
9. An RDD is a read-only, partitioned collection of records.
10. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

**Que 4.22. Give the anatomy of a Spark job run.**

**Answer**

1. When we run a Spark job at the highest level, there are two independent entities : the driver and the executors.
2. The driver hosts the application and schedules tasks for a job.
3. The executor runs for the duration of the application and execute the application's tasks.
4. Following are various stages of Spark job run :

**A. Job Submission :**

1. A Spark job is submitted automatically when an action is performed on an RDD.
2. This passes the call on to the scheduler that runs as a part of the driver.
3. The scheduler is made up of two parts: a DAG scheduler and a task scheduler.
4. DAG scheduler breaks down the job into a DAG of stages.
5. Task scheduler is responsible for submitting the tasks from each stage to the cluster.

**B. DAG Construction :**

1. There are two types of tasks that can run in a stage : shuffle map tasks and result tasks.
2. Shuffle map tasks are like the map-side part of the shuffle in MapReduce.
3. Each shuffle map task runs a computation on one RDD partition.
4. Shuffle map tasks run in all stages except the final stage.
5. Result tasks run in the final stage that returns the result to the user's program.
6. Each result task runs a computation on its RDD partition, and then sends the result back to the driver.
7. The driver assembles the results from each partition into a final result.

**C. Task Scheduling :**

1. When the task scheduler is sent a set of tasks, it uses its list of executors and constructs a mapping of tasks to executors.
2. Next, the task scheduler assigns tasks to executors that have free cores.
3. As executors finish running tasks, task scheduler continues to assign more tasks, until the task set is complete.
4. Each task is allocated one core by default, although this can be changed.

**D. Task Execution :** An executor runs a task as follows :

1. First, it makes sure that the JAR and file dependencies for the task are up to date.
2. Second, it deserializes the task code from the serialized bytes that were sent as a part of the launch task message.
3. Third, the task code is executed.

**Que 4.23. Write a short note on : Running Spark on YARN.**

**Answer**

1. Running Spark on YARN is the most convenient way to use Spark when you have an existing Hadoop cluster.
2. It also provides the tightest integration with other Hadoop components.
3. Spark offers two deploy modes for running on YARN: YARN client mode and YARN cluster mode.
4. In YARN client mode, the driver runs in the client.
5. YARN client mode is required for programs that have interactive components.
6. Client mode is also useful when building Spark programs, since any debugging output is immediately visible.
7. In YARN cluster mode, the driver runs on the cluster in the YARN application master.
8. YARN cluster mode is appropriate for production jobs.
9. In YARN cluster mode, the entire application runs on the cluster, which makes it much easier to retain logfiles for later inspection.
10. In YARN cluster mode, YARN will also retry the application if the application master fails.





## PART-5

*SCALA, Introduction, Classes and Objects, Basics Types and Operators, Built-in Control Structure, Functions and Closures, Inheritance.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 4.24. Write short note on SCALA.**

**Answer**

1. Scalable Language (SCALA) is a hybrid functional programming language.
2. Scala smoothly integrates the features of object-oriented and functional languages.
3. Scala is compiled to run on the Java Virtual Machine.
4. Many existing companies, who depend on Java for business critical applications, are turning to Scala to boost their development productivity, applications scalability and overall reliability.
5. Scala is also a functional language in the sense that every function is a value and every value is an object so ultimately every function is an object.
6. Scala provides a lightweight syntax for defining anonymous functions, it supports higher-order functions, it allows functions to be nested, and supports currying.
7. Scala allows us to express general programming patterns in an effective way.
8. It reduces the number of lines and helps the programmer to code in a type-safe way.
9. It allows us to write codes in an immutable manner, which makes it easy to apply concurrency and parallelism (Synchronize).

**Que 4.25. Describe classes, fields, and methods in Scala.**

**Answer**

1. A class is a blueprint for objects.
2. Once we define a class, we can create objects from the class blueprint with the keyword new.

3. In Scala, a class declaration contains the class keyword, followed by an identifier(name) of the class.
4. Inside a class definition, we place fields and methods, which are collectively called members.
5. Fields, which we define with either val or var, are variables that refer to objects.
6. The fields hold the state, or data, of an object.
7. Methods, which we define with def, contain executable code.
8. The methods use the data hold by the fields to do the computational work of the object.
9. When we instantiate a class, the runtime sets aside some memory to hold the image of that object's state, i.e., the content of its variables.
10. Fields are also known as instance variables, because every instance gets its own set of the variables.
11. Collectively, an object's instance variables make up the memory image of the object.
12. Robustness of an object is ensured by making sure that the object's state remains valid during its entire lifetime.

**Que 4.26. Describe singleton objects in Scala.**

**Answer**

1. Classes in Scala cannot have static members. Instead, Scala has singleton objects.
2. A singleton object definition looks like a class definition, except instead of the keyword class we use the keyword object.
3. When a singleton object shares the same name with a class, it is called that class's companion object.
4. We must define both the class and its companion object in the same source file.
5. The class is called the companion class of the singleton object.
6. A class and its companion object can access each other's private members.
7. Defining a singleton object doesn't define a type at the Scala level of abstraction.
8. Rather, the type is defined by the singleton object's companion class.
9. Singleton objects cannot take parameters.
10. Each singleton object is implemented as an instance of a synthetic class referenced from a static variable.
11. A singleton object that does not share the same name with a companion class is called a standalone object.



12. We can use standalone objects for many purposes, like collecting related utility methods together, or defining an entry point to a Scala application.

**Que 4.27. Mention the basic types of Scala.**

**Answer**

Several fundamental types of Scala, along with the ranges of values instances are shown in Table 4.27.1.

**Table 4.27.1.** Some basic types

| Value type | Range |
|---|---|
| Byte | 8-bit signed two's complement integer ($-2^7$ to $2^7 - 1$, inclusive) |
| Short | 16-bit signed two's complement integer ($-2^{15}$ to $2^{15} - 1$, inclusive) |
| Int | 32-bit signed two's complement integer ($-2^{31}$ to $2^{31} - 1$, inclusive) |
| Long | 64-bit signed two's complement integer ($-2^{63}$ to $2^{63} - 1$, inclusive) |
| Char | 16-bit unsigned Unicode character (0 to $2^{16} - 1$, inclusive) |
| String | a sequence of Chars |
| Float | 32-bit IEEE 754 single-precision float |
| Double | 64-bit IEEE 754 double-precision float |
| Boolean | true or false |

**Que 4.28. What are the different types of operators used in SCALA ?**

**Answer**

Following are the operators used in SCALA :

1. **Arithmetic operators :** These are used to perform arithmetic/ mathematical operations on operands.
   - i. **Addition(+) operator :** It adds two operands.
   - ii. **Subtraction(-) operator :** It subtracts two operands.
   - iii. **Multiplication(*) operator :** It multiplies two operands.
   - iv. **Division(/) operator :** It divides the first operand by the second.
   - v. **Modulus(%) operator :** It returns the remainder when the first operand is divided by the second.
   - vi. **Exponent(**) operator :** It returns exponential(power) of the operands.
2. **Relational operators :** Relational operators or Comparison operators are used for comparison of two values.

i. **Equal To(==) operator :** It checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false.

ii. **Not Equal To(!=) operator :** It checks whether the two given operands are equal or not. If not, it returns true. Otherwise it returns false.

iii. **Greater Than(>) operator :** It checks whether the first operand is greater than the second operand. If so, it returns true. Otherwise it returns false.

iv. **Less than(<) operator :** It checks whether the first operand is lesser than the second operand. If so, it returns true. Otherwise it returns false.

v. **Greater Than Equal To(>=) operator :** It checks whether the first operand is greater than or equal to the second operand. If so, it returns true. Otherwise it returns false.

vi. **Less Than Equal To(<=) operator :** It checks whether the first operand is lesser than or equal to the second operand. If so, it returns true. Otherwise it returns false.

3. **Logical operators :** They are used to combine two or more conditions/ constraints or to complement the evaluation of the original condition in consideration. They are described as :

i. **Logical AND(&&) operator :** It returns true when both the conditions in consideration are satisfied. Otherwise it returns false.

ii. **Logical OR( || ) operator :** It returns true when one (or both) of the conditions in consideration is satisfied. Otherwise it returns false.

iii. **Logical NOT(!) operator :** It returns true the condition in consideration is not satisfied. Otherwise it returns false.

4. **Assignment operators :** Assignment operators are used to assigning a value to a variable. The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value. Different types of assignment operators are :

i. **Simple Assignment (=) operator :** It is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left.

ii. **Add AND Assignment (+=) operator :** It is used for adding left operand with right operand and then assigning it to variable on the left.

iii. **Subtract AND Assignment (–=) operator :** It is used for subtracting left operand with right operand and then assigning it to variable on the left.

iv. **Multiply AND Assignment (*=) operator :** It is used for multiplying the left operand with right operand and then assigning it to the variable on the left.



v. **Divide AND Assignment (/=) operator :** It is used for dividing left operand with right operand and then assigning it to variable on the left.

vi. **Modulus AND Assignment (%=) operator :** It is used for assigning modulo of left operand with right operand and then assigning it to the variable on the left.

vii. **Exponent AND Assignment (**=) operator :** It is used for raising power of the left operand to the right operand and assigning it to the variable on the left.

viii. **Left shift AND Assignment(<<=)operator :** It is used to perform binary left shift of the left operand with the right operand and assigning it to the variable on the left.

ix. **Right shift AND Assignment(>>=)operator :** It is used to perform binary right shift of the left operand with the right operand and assigning it to the variable on the left.

x. **Bitwise AND Assignment(&=)operator :** It is used to perform Bitwise And of the left operand with the right operand and assigning it to the variable on the left.

xi. **Bitwise exclusive OR and Assignment(^=)operator :** It is used to perform Bitwise exclusive OR of the left operand with the right operand and assigning it to the variable on the left.

xii. **Bitwise inclusive OR and Assignment( | =)operator :** It is used to perform Bitwise inclusive OR of the left operand with the right operand and assigning it to the variable on the left.

5. **Bitwise operator :** In Scala, there are seven bitwise operators which work at bit level or used to perform bit by bit operations. Following are the bitwise operators :

i. **Bitwise AND (&) :** Takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.

ii. **Bitwise OR ( | ) :** Takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 any of the two bits is 1.

iii. **Bitwise XOR (^) :** Takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

iv. **Bitwise left Shift (<<) :** Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.

v. **Bitwise right Shift (>>) :** Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.

vi. **Bitwise ones Complement (~) :** This operator takes a single number and used to perform the complement operation of 8-bit.

vii. **Bitwise shift right zero fill(>>>) :** In shift right zero fill operator, left operand is shifted right by the number of bits specified by the right operand and the shifted values are filled up with zeros.

**Que 4.29. What are various built-in control structures of Scala ?**

**Answer**

1. Scala has only a handful of built-in control structures.
2. The only control structures are if, while, for, try, match, and function calls.
3. The reason Scala has so few control structures is that it has included function literals since its inception.
4. Instead of accumulating one higher-level control structure after another in the base syntax, Scala accumulates them in libraries.
5. Following are few control structures that are built in :

**A. If expressions :**

1. Scala's if works just like in many other languages.
2. It tests a condition and then executes one of two code branches depending on whether the condition holds true.
3. For example,

```
var filename = "default.txt"
if (!args.isEmpty)
    filename = args(0)
```

4. This code declares a variable, filename, and initializes it to a default value.
5. It then uses an if expression to check whether any arguments were supplied to the program.
6. If arguments were supplied, it changes the variable to hold the value specified in the argument list.
7. If no arguments were supplied, it leaves the variable set to the default value.

**B. While loops :**

1. Scala's while loop behaves as in other languages.
2. It has a condition and a body, and the body is executed over and over as long as the condition holds true.
3. For example :

```
while (a != 0) {
val temp = a
a = b % a
b = temp
}
```

4. Scala also has a do-while loop.
5. This works like the while loop except that it tests the condition after the loop body instead of before.

**C. For expressions :**

1. Scala's for expression lets you combine a few simple ingredients in different ways to express a wide variety of iterations.
2. Simple uses enable common tasks such as iterating through a sequence of integers.
3. More advanced expressions can iterate over multiple collections of different kinds, can filter out elements based on arbitrary conditions, and can produce new collections.

**D. Exception handling with try expressions :**

1. Scala's exceptions behave just like in many other languages.
2. Instead of returning a value in the normal way, a method can terminate by throwing an exception.
3. The method's caller can either catch and handle that exception, or it can itself simply terminate, in which case the exception propagates to the caller's caller.
4. The exception propagates in this way, unwinding the call stack, until a method handles it or there are no more methods left.

**E. Match expressions :**

1. Scala's match expression lets you select from a number of alternatives.
2. In general a match expression lets you select using arbitrary patterns.

**Que 4.30. What are various functions used in Scala ?**

**Answer**

1. When programs get larger, you need some way to divide them into smaller, more manageable pieces.
2. For dividing up control flow, Scala divide the code into functions.
3. A function is a group of statements that perform a task.
4. We can divide up our code into separate functions.
5. Following are Scala's ways to express functions :

**A. Methods :**

1. The most common way to define a function is as a member of some object.
2. Such a function is called a method.

**B. Local functions :**

1. Scala offers an approach in which we can define functions inside other functions.
2. Just like local variables, such local functions are visible only in their enclosing block.

**C. First-class functions :**

1. Scala has first-class functions.
2. Not only can you define functions and call them, but you can write down functions as unnamed literals and then pass them around as values.

**D. Partially applied functions :**

1. In Scala, when you invoke a function, passing in any needed arguments, you apply that function to the arguments.
2. A partially applied function is an expression in which you don't supply all of the arguments needed by the function.
3. Instead, you supply some, or none, of the needed arguments.

**Que 4.31. What do you understand by closure ?**

**Answer**

1. The function value (the object) that's created at runtime from a function literal is called a closure.
2. The name arises from the act of "closing" the function literal by "capturing" the bindings of its free variables.
3. A function literal with no free variables is called a closed term.
4. Function value created at runtime from this function literal is not a closure in the strictest sense.
5. Any function literal with free variables is called an open term.
6. Function value created at runtime requires that a binding for its free variable be captured.
7. This resulting function value, which will contain a reference to the captured free variable, is called a closure.
8. Because the function value is the end product of the act of closing the open term.

**Que 4.32. Write a short note on : Inheritance in Scala.**



**Answer**

1. In Scala inheritance is the superclass/subclass relationship.
2. Inheritance is a way to define a new class in terms of another existing class.
3. It is the mechanism in Scala by which one class is allowed to inherit the features (fields and methods) of another class.
4. We, in general, prefer inheritance if we want code reuse.
5. However, inheritance suffers from the fragile base class problem.
6. This can be overcome by changing a superclass to break subclasses.
7. The keyword used for inheritance is extends.
8. For example,

```
class parent_class_name extends child_class_name {
// Methods and fields
}
```

☺☺☺

# 5 UNIT

# Applications on Big Data using Pig, Hive and HBase

## CONTENTS





## PART-1

*PIG, Introduction to PIG, Execution Modes of PIG, Comparison of PIG with Database Grunt, PIG Latin, User Defined Function, Data Processing Operators.*

### Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 5.1. Write short note on PIG.**

**Answer**

1. PIG is a tool/platform which is used to analyze larger sets of data representing them as data flows.
2. PIG is generally used with Hadoop, we can perform all the data manipulation operations in Hadoop using Apache PIG.
3. To write data analysis programs, PIG provides a high-level language known as PIG Latin.
4. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.
5. To analyze data using Apache PIG, programmers need to write scripts using PIG Latin language.
6. All these scripts are internally converted to Map and Reduce tasks.
7. Apache PIG has a component known as PIG Engine that accepts the PIG Latin scripts as input and converts those scripts into Map Reduce jobs.

**Que 5.2. Discuss the execution modes/types of Pig.**

**Answer**

Pig has two execution types or modes : local mode and MapReduce mode.

**A. Local Mode :**

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

**B. MapReduce Mode :**

1. The MapReduce mode is also known as Hadoop Mode.

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

**Que 5.3. Write short note on Pig feature.**

**Answer**

Following are the feature of Pig :

1. **Rich set of operators :** It provides many operators to perform operations like join, sort, filer, etc.
2. **Ease of programming :** Pig Latin is similar to SQL and it is easy to write a Pig script if we are good at SQL.
3. **Optimization opportunities :** The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
4. **Extensibility :** Using the existing operators, users can develop their own functions to read, process, and write data.
5. **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.
6. **Handles all kinds of data :** Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

**Que 5.4. Differentiate between Pig Latin and SQL.**

**Answer**

| S. No. | Pig Latin | SQL |
|---|---|---|
| 1. | Pig Latin is a procedural language. | SQL is a declarative language. |
| 2. | 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. |
| 3. | The data model in Apache Pig is nested relational. | The data model used in SQL is flat relational. |
| 4. | Apache Pig provides limited opportunity for Query optimization. | There is more opportunity for query optimization in SQL. |

**Que 5.5. Write a short note on : Grunt.**

**Answer**

1. Grunt is Pig's interactive shell.
2. It enables users to enter Pig Latin interactively and provides a shell for users to interact with HDFS.
3. Grunt is started when no file is specified for Pig to run, and the -e option is not used.
4. It is also possible to run Pig scripts from within Grunt using run and exec.
5. Grunt provides command-line history and editing, as well as Tab completion.
6. Grunt is not a full-featured shell.
7. It does not provide a number of commands found in standard Unix shells, such as pipes, redirection, and background execution.
8. You can get a list of commands using the help command.
9. To exit Grunt you can type quit or enter Ctrl-D.

**Que 5.6. What are the various syntax and semantics of the Pig Latin programming language ?**

**Answer**

Following are the various syntax and semantics of the Pig Latin programming language :

**A. Statements :**

1. A Pig Latin program consists of a collection of statements.
2. A statement can be thought of as an operation, or a command.
3. Statements are usually terminated with a semicolon.
4. Pig Latin has two forms of comments: Double hyphen comments and C-style comments.
5. Pig Latin has a list of keywords that have a special meaning in the language and cannot be used as identifiers.
6. Pig Latin has mixed rules on case sensitivity. Operators and commands are not case sensitive; however, aliases and function names are case sensitive.

**B. Expressions :**

1. An expression is something that is evaluated to yield a value.
2. Expressions can be used in Pig as a part of a statement containing a relational operator.
3. Pig has a rich variety of expressions.

**C. Types :**

1. Pig has four numeric types: int, long, float, and double.
2. There is also a bytearray type for representing a blob of binary data, and chararray which represents textual data in UTF-16 format.
3. The numeric, textual, and binary types are simple atomic types.
4. Pig also has three complex types for representing nested structures : tuple, bag, and map.

**D. Schemas :**

1. A relation in Pig may have an associated schema, which gives the fields in the relation names and types.
2. Pig's flexibility in the degree to which schemas are declared contrasts with traditional SQL databases, which are declared before the data is loaded into the system.
3. Pig is designed for analyzing plain input files with no associated type information, so the types for fields are chosen later.

**E. Functions :**

1. Functions in Pig come in four types: eval function, filter function, load function, and store function.
2. If the function you need is not available, you can look in the Piggy Bank or write your own.
3. Piggy Bank is a repository of Pig functions shared by the Pig community.
4. If the Piggy Bank doesn't have what you need, you can write your own function.
5. These are known as user-defined functions, or UDFs.

**F. Macros :**

1. Macros provide a way to package reusable pieces of Pig Latin code from within Pig Latin itself.
2. To encourage reuse, macros can be defined in separate files to Pig scripts.
3. In this case they need to be imported into any script that uses them.

**Que 5.7. Write a short note on : User-defined functions (UDFs) in Pig.**

**Answer**

1. In Pig the ability to plug-in custom code is crucial for all data processing jobs.
2. For this reason, Pig's designers made it easy to define and use user-defined functions.
3. User-defined functions (UDFs) can be written in Java, Python or JavaScript.



4. They are run using the Java Scripting API.
5. Following are various user-defined functions (UDFs) :

**A. A Filter UDF :**

1. Filter UDFs are all subclasses of FilterFunc, which itself is a subclass of EvalFunc.
2. It makes the Pig script more concise, and it encapsulates the logic in one place so that it can be easily reused in other scripts.
3. For an ad hoc query, we wouldn't write a UDF.
4. It's when you start doing the same kind of processing over and over again that you use reusable UDFs.

**B. An Eval UDF :**

1. Writing an eval function is a small step up from writing a filter function.
2. An eval function extends the EvalFunc class, parameterized by the type of the return value.
3. When you write an eval function, you need to consider what the output's schema looks like.

**C. A Load UDF :**

1. The load UDFs control how data goes into Pig and comes out of Pig.
2. The Pig load API is aligned with Hadoop's InputFormat and OutputFormat classes.
3. This enables you to create new LoadFunc implementations based on existing Hadoop InputFormat and OutputFormat classes with minimal code.
4. The complexity of reading the data and creating a record lies in the InputFormat while the complexity of writing the data lies in the OutputFormat.
5. This enables Pig to easily read/write data in new storage formats as and when an Hadoop InputFormat and OutputFormat is available for them.

**Que 5.8. What are various data processing operators in Pig ?**

**Answer**

1. Data processing operators are the main tools Pig Latin provides to operate on the data.
2. They allow you to transform it by sorting, grouping, joining, projecting, and filtering.
3. Pig data processing operators can be classified as :

**A. Loading and Storing Data :**

1. The data is loaded from external storage for processing in Pig.

2. Storing the results is also straightforward.
3. There are many built-in storage functions to store data.

**B. Filtering Data :**

1. Once you have some data loaded into a relation, the next step is to filter it.
2. Filtering removes the data that you are not interested in.
3. By filtering early we minimize the amount of data flowing through the system.
4. This can improve efficiency.
5. There are various operators used for filtering data, for example, FOREACH...GENERATE, STREAM, etc.

**C. Grouping and Joining Data :**

1. Pig has very good built-in support for join operations, making it much more approachable.
2. Since the large datasets that are suitable for analysis by Pig are not normalized, joins are used infrequently in Pig.
3. There are various operators used for grouping and joining data, for example, JOIN, COGROUP, CROSS, GROUP, etc.

**D. Sorting Data :**

1. Relations are unordered in Pig.
2. There is no guarantee in which order the rows will be processed.
3. If you want to impose an order on the output, you can use the ORDER operator to sort a relation by one or more fields.

**E. Combining and Splitting Data :**

1. Sometimes you have several relations that you would like to combine into one.
2. For this, the UNION statement is used.
3. Pig attempts to merge the schemas from the relations that UNION is operating on.
4. Sometimes we need to partition a relation into two or more relations.
5. For this, the SPLIT operator is used.

## PART-2

*Hive, Apache Hive Architecture and Installation, Hive Shell, Hive Services, Hive Metastore, Comparison with Traditional Database, HiveQL Tables, Querying Data and User Defined Function, Sorting and Aggregating, Map Reduce Scripts, Join and Subqueries.*



## Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 5.9. Write short note on Hive.**

**Answer**

1. Hive is a data warehouse infrastructure tool used to process structured data in Hadoop.
2. It resides on top of Hadoop to summarize Big data, and makes querying and analyzing easy.
3. Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.
4. It is used by different companies *i.e.*, Amazon uses it in Amazon Elastic MapReduce.
5. It stores schema in a database and processed data into HDFS.
6. It is designed for OLAP.
7. It provides SQL type language for querying called HiveQL or HQL.
8. It is familiar, fast, scalable, and extensible.

**Que 5.10. Explain the architecture of Apache Hive.**

**Answer**

Fig. 5.10.1 shows the architecture of Apache Hive :

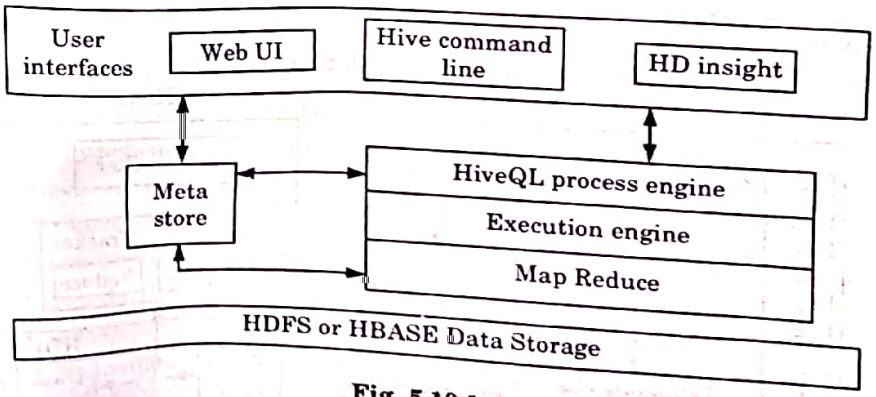



**Fig. 5.10.1.**

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

**Que 5.11. Explain the working of HIVE with Hadoop.**

**Answer**

Fig. 5.11.1 shows the working of Hive with Hadoop :

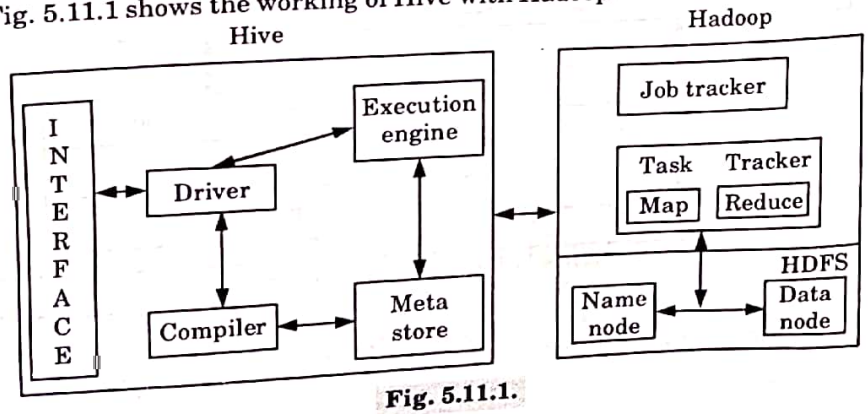



**Fig. 5.11.1.**



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. 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 :**
   i. Internally, the process of execution job is a MapReduce job.
   ii. The execution engine sends the job to JobTracker, which is i0n Name node and it assigns this job to TaskTracker, which is in Data node.
   iii. Here, the query executes MapReduce job.
8. **Metadata ops :** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
9. **Fetch result :** The execution engine receives the results from Data nodes.
10. **Send result :** The execution engine sends those resultant values to the driver. The driver sends the results to Hive Interfaces.

**Que 5.12. Explain the process of Apache Hive installation.**

**Answer**

**Step 1 : Verifying JAVA installation :**

i. Java must be installed on our system before installing Hive.
ii. Verify java installation using the following command :

$ java -version

**Step 2 : Verifying Hadoop installation :**

i. Hadoop must be installed on your system before installing Hive.
ii. Verify the Hadoop installation using the following command :

$ hadoop version

**Step 3 : Downloading Hive :** We can download it by visiting the following link http://apache.petsads.us/hive/hive-0.14.0/.

**Step 4 : Installing Hive :**

i. The following steps are required for installing Hive on your system.

ii. Assume the Hive archive is downloaded onto the /Downloads directory.

**Step 6 : Downloading and Installing Apache Derby :** Follow the steps given below to download and install Apache Derby :

i. Downloading Apache Derby :

The following command is used to download Apache Derby.

```
$ cd ~
$ wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz
```

ii. The following command is used to verify the download :

```
$ ls
```

**Step 7 : Configuring Metastore of Hive :**

i. Configuring Metastore means specifying to Hive where the database is stored.

ii. We can do this by editing the hive-site.xml file, which is in the $HIVE_HOME/conf directory.

**Step 8 : Verifying Hive Installation :**

i. Before running Hive, we need to create the /tmp folder and a separate Hive folder in HDFS. Here, we use the /user/hive/warehouse folder.

ii. We need to set write permission for these newly created folders as :

```
chmod g + w
```

iii. Now set them in HDFS before verifying Hive.

iv. The following commands are used to verify Hive installation :

```
$ cd $HIVE_HOME
$ bin/hive
```

**Que 5.13. Write a short note on : Hive shell.**

**Answer**

1. Hive shell is a primary way to interact with Hive.
2. It is a default service in Hive.
3. It is also called as CLI (Command Line Interference).
4. The shell interacts with Hive by issuing commands in HiveQL.
5. HiveQL is Hive's query language.
6. HiveQL is generally case insensitive.
7. You can run the Hive shell in both interactive and non-interactive mode.



8. In Hive shell up and down arrow keys are used to scroll previous commands.
9. Other useful Hive shell features include the ability to run commands on the host operating system and the ability to access Hadoop filesystems.

**Que 5.14. Explain Hive services.**

**Answer**

The following are the services provided by Hive :

1. **Hive CLI :** The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands.
2. **Hive Web User Interface :**
   i. The Hive Web UI is just an alternative of Hive CLI.
   ii. It provides a web-based GUI for executing Hive queries and commands.
3. **Hive MetaStore :**
   i. It is a central repository that stores all the structure information of various tables and partitions in the warehouse.
   ii. 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.
4. **Hive Server :**
   i. It is referred to as Apache Thrift Server.
   ii. It accepts the request from different clients and provides it to Hive Driver.
5. **Hive Driver :**
   i. It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver.
   ii. It transfers the queries to the compiler.
6. **Hive Compiler :**
   i. The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions.
   ii. It converts HiveQL statements into MapReduce jobs.
7. **Hive Execution Engine :**
   i. Optimizer generates the logical plan in the form of DAG of MapReduce tasks and HDFS tasks.
   ii. In the end, the execution engine executes the incoming tasks in the order of their dependencies.

**Que 5.15. What is Hive Metastore ?**

**Answer**

1. Metastore is the central repository of Apache Hive metadata.
2. It stores metadata for Hive tables and partitions in a relational database.
3. It provides client access to this information by using metastore service API.
4. Hive metastore consists of two fundamental units :
   i. A service that provides metastore access to other Apache Hive services.
   ii. Disk storage for the Hive metadata which is separate from HDFS storage.
5. Following are three modes for Hive Metastore deployment :

**A. Embedded Metastore :**

1. In Hive by default, metastore service runs in the same JVM as the Hive service.
2. It uses embedded Derby Database stored on the local file system in this mode.
3. Thus both metastore service and hive service runs in the same JVM by using embedded Derby Database.
4. But, this mode has a limitation. As only one embedded Derby Database can access the database files at any one time, so only one Hive session could be open at a time. If we try to start the second session it produces an error.

**B. Local Metastore :**

1. To overcome the limitation of Embedded Metastore, Local Metastore was introduced.
2. This mode allows us to have many Hive sessions *i.e.*, many users can use the metastore at the same time.
3. We can achieve this by using any JDBC compliant like MySQL which runs in a separate JVM or different machines than that of the Hive service and metastore service which are running in the same JVM.

**C. Remote Metastore :**

1. In this mode, metastore runs on its own separate JVM, not in the Hive service JVM.
2. If other processes want to communicate with the metastore server they can communicate using Thrift Network APIs.
3. We can also have one more metastore servers in this case to provide more availability.
4. This brings better manageability/security because the database tier can be completely firewalled off.



5. Also the client is no longer required to share database credentials with each Hive user to access the metastore database.

**Que 5.16. Give comparison of Hive with traditional databases.**

**Answer**

| S. No. | Hive | Traditional Databases |
|---|---|---|
| 1. | Hive does not verify data when it is loaded, but rather when a query is issued. This design is called schema on read. | In traditional database the data is checked when it is written into the database. This design is called schema on write. |
| 2. | It is very easily scalable at low cost. | Not much scalable. Also costly scale up. |
| 3. | It is based on Hadoop notation *i.e.*, write once and read many time. | In traditional database we can read and write many time. |
| 4. | Record level updates is not possible in Hive. | Record level updates, insertions and deletes, transactions and indexes are possible. |
| 5. | OLTP (On-line Transaction Processing) is not yet supported in Hive but it's supported OLAP (On-line Analytical Processing). | Both OLTP (On-line Transaction Processing) and OLAP (On-line Analytical Processing) are supported in RDBMS. |

**Que 5.17. Write a short note on : HiveQL.**

**Answer**

1. Hive Query Language (HiveQL) is for querying the large datasets which reside in the HDFS environment.
2. HiveQL script commands enable data definition, data manipulation and query processing.
3. HiveQL supports a large base of SQL users who are acquainted with SQL to extract information from data warehouses.
4. Hive provides a CLI for Hive query writing using Hive Query Language (HiveQL).
5. Hive supports four file formats which are: TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).
6. Hive supports both primitive and complex data types.

7. Primitives include numeric, boolean, string, and timestamp types.
8. The complex data types include arrays, maps, and structs.
9. The usual set of SQL operators is provided by Hive : relational operators, arithmetic operators, and logical operators.
10. Hive comes with a large number of built-in functions including mathematical and statistical functions, string functions, date functions, conditional functions, etc.

**Que 5.18. What is a Hive table ?**

**Answer**

1. A Hive table is logically made up of the data being stored and the associated metadata describing the layout of the data in the table.
2. There are two types of table in Hive : managed table and external table.
3. When you create a table in Hive, by default Hive will manage the data. This is known as managed table.
4. When you load data into a managed table, it is moved into Hive's warehouse directory.
5. In an external table, Hive refers to the data that is existing at location outside the warehouse directory.
6. If you are doing all your processing with Hive, then use managed tables.
7. If you are using Hive and other tools on the same dataset, then use external tables.

**Que 5.19. Write a short note on : Partitions and Buckets.**

**Answer**

**A. Partitions :**

1. Hive allows us to organize the table into multiple partitions where we can group the same kind of data together.
2. It is used for distributing the load horizontally.
3. Partition helps to do faster queries on slices of the data.
4. A table may be partitioned in multiple dimensions.
5. Partitions are used when the column with a high search query has low cardinality.
6. It is effective when the data volume in each partition is not very high.

**B. Buckets :**

1. Bucket is a technique to divide the data in a manageable form.
2. Tables or partitions may further be subdivided into buckets, to give extra structure to the data.



3. This data then may be used for more efficient queries.
4. There are two reasons to organize tables (or partitions) into buckets.
5. The first is to enable more efficient queries.
6. The second reason is to make sampling more efficient.

**Que 5.20. Write a short note on : Storage formats in Hive.**

**Answer**

1. There are two storage formats in Hive : row format and file format.
2. The row format dictates how rows, and the fields in a particular row, are stored.
3. The file format dictates the container format for fields in a row.
4. The default storage format is delimited text, with a row per line.
5. We can use Hadoop's sequence file format in Hive.
6. Hadoop's sequence file format is a general purpose binary format for sequences of records (key-value pairs).
7. Sequence files are row-oriented.
8. Hive provides another binary storage format called *RCFile, short for* Record Columnar File.
9. RCFiles store data in a column-oriented fashion.
10. A column-oriented layout permits columns that are not accessed in a query to be skipped.

**Que 5.21. What do you understand by sorting and aggregating in Hive ?**

**Answer**

1. Hive has both an ORDER BY and a SORT BY clause to sort the output of a query.
2. The difference between the two is that ORDER BY imposes a total order on query results, but SORT BY imposes order only on the rows in a Reduce task.
3. If there are multiple Reduce tasks, the output data will only have a partial order when SORT BY is used.
4. The ORDER BY clause requires a single Reduce task to achieve total order on the query, which is a bottleneck to sort large datasets.
5. Hive enforces the need for a LIMIT operator by default when an ORDER BY clause is used.
6. To perform aggregation we want to control which reducer a particular row goes to.

7. This is achieved by using Hive's DISTRIBUTE BY clause.

**Que 5.22. Write a short note on : Joins and Subqueries in Hive.**

**Answer**

**Joins in Hive :**

1. Hive makes performing commonly used operations like join very simple.
2. Following are the various kind of join used in Hive :

**A. Inner joins :**

1. The simplest kind of join is the inner join, where each match in the input tables results in a row in the output.
2. In Hive, you can join on multiple columns in the join predicate by specifying a series of expressions, separated by AND keywords.
3. You can also join more than two tables by supplying additional JOIN...ON... clauses in the query.

**B. Outer joins :**

1. Outer joins allow you to find nonmatches in the tables being joined.
2. Hive supports right outer joins, which reverses the roles of the tables relative to the left join.

**C. Map joins :**

1. If one table is small enough to fit in memory, then Hive can load the smaller table into memory to perform the join in each of the mappers.
2. Map joins can take advantage of bucketed tables, since a mapper working on a bucket of the left table only needs to load the corresponding buckets of the right table to perform the join.

**Subqueries in Hive :**

1. A subquery is a SELECT statement that is embedded in another SQL statement.
2. Hive has limited support for subqueries.
3. It only permit a subquery in the FROM clause of a SELECT statement.
4. Other databases allow subqueries almost anywhere that an expression is valid.
5. Many uses of subqueries can be rewritten as joins.
6. So when Hive does not support a subquery, then we see if it can be expressed as a join.



**Que 5.23. What do you understand by user-defined function (UDF) in Hive ?**

**Answer**

1. The built-in functions of Hive sometimes can't express the query that we want to write.
2. In such case we can write a user-defined function (UDF).
3. Hive makes it easy to plug UDF in the processing code and invoke it from a Hive query.
4. UDFs have to be written in Java language.
5. There are three types of UDF in Hive :
   - i. **UDF :** A UDF operates on a single row and produces a single row as its output.
   - ii. **UDAF (user-defined aggregate function) :** A UDAF works on multiple input rows and creates a single output row.
   - iii. **UDTF (user-defined table-generating function) :** A UDTF operates on a single row and produces multiple rows—a table—as output.

## PART-3

*HBase, HBase Concepts, Clients Example, HBase vs RDBMS, Advanced Usage, Schema Design, Advanced Indexing, Zookeeper, How it Helps in Monitoring a Cluster, How to Build Application with Zookeeper.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 5.24. What is HBase ?**

**Answer**

1. HBase is a column-oriented non-relational database management system that runs on top of Hadoop Distributed File System (HDFS).
2. HBase provides a fault-tolerant way of storing sparse data sets, which are common in many big data use cases.
3. It is well suited for real-time data processing or random read/write access to large volumes of data.

4. Unlike relational database systems, HBase does not support a structured query language like SQL.
5. HBase applications are written in Java.
6. An HBase system is designed to scale linearly.
7. It comprises a set of standard tables with rows and columns.
8. Each table must have an element defined as a primary key, and all access attempts to HBase tables must use this primary key.
9. HBase relies on ZooKeeper for high-performance coordination.
10. HBase works well with Hive to enable fault-tolerant big data applications.

**Que 5.25. Explain different features of HBase.**

**Answer**

Following are the features of HBase :

1. HBase is built for low latency operations.
2. It is used extensively for random read and write operations.
3. It stores a large amount of data in terms of tables.
4. It provides linear and modular scalability over cluster environment.
5. It strictly consistent to read and write operations.
6. It has automatic and configurable sharding of tables.
7. It has automatic failover supports between Region Servers.
8. Convenient base classes for backing Hadoop MapReduce jobs in HBase tables.
9. Easy to use Java API for client access.
10. Block cache and Bloom Filters for real-time queries.
11. Query predicate pushes down via server-side filters.

**Que 5.26. Provide overview of HBase data model.**

**OR**

**Write in detail about Hbase data model.**

**Answer**

1. The HBase Data Model is designed to handle semi-structured data that may differ in field size, which is a form of data and columns.
2. The data model's layout partitions the data into simpler components and spread them across the cluster.
3. HBase's Data Model consists of following logical components :



| Row Key | Customer | | Sales | |
|---|---|---|---|---|
| Customer id | Name | City | Product | Amount |
| 101 | Ram | Delhi | Chairs | 4000.00 |
| 102 | Shyam | Lucknow | Lamps | 2000.00 |
| 103 | Gita | Ghaziabad | Desk | 5000.00 |
| 104 | Sita | Kanpur | Bed | 2600.00 |

Column Families

i. **Table :**
   1. An HBase table is made up of several columns.
   2. The tables in HBase define upfront during the time of the schema specification.

ii. **Row :**
   1. An HBase row consists of a row key and one or more associated value columns.
   2. Row keys are the bytes that are not interpreted.
   3. Rows are ordered lexicographically, with the first row appearing in a table in the lowest order.
   4. The layout of the row key is very critical for this purpose.

iii. **Column :**
   1. A column in HBase consists of a family of columns and a qualifier of columns, which is identified by a character : (colon).

iv. **Column Family :**
   1. Apache HBase columns are separated into the families of columns.
   2. The column families physically position a group of columns and their values to increase its performance.
   3. The same prefix is granted to all column members of a column family.
   4. The character of the colon (:) distinguishes the family of columns from the qualifier of the family of columns.
   5. During schema definition time, column families must be declared upfront.
   6. While columns can be conjured on the fly when the table is up and running.

v. **Column Qualifier :**
   1. The column qualifier is added to a column family.
   2. Although column families are set up at table formation, column qualifiers are mutable and can vary significantly from row to row.

**vi. Cell :**

1. A Cell store data and is quite a unique combination of row key, Column Family, and the Column.
2. The data stored in a cell call its value and data types, which is every time treated as a byte[ ].

**vii. Timestamp :**

1. The timestamp reflects the time when the data is written on the Region Server.
2. But when we put data into the cell, we can assign a different timestamp value.

**Que 5.27. What are various client options for interacting with an HBase cluster ?**

**Answer**

Following are various client options for interacting with an HBase cluster :

**A. Java :**

1. HBase is written in Java.
2. To interact with an HBase cluster in Java we create instances of two classes HBaseAdmin and HTable.
3. HBaseAdmin is used for administering HBase cluster, for adding and dropping tables.
4. HTable is used to access a specific table.

**B. Avro :**

1. HBase is shipped with Avro interface.
2. Avro interface is useful when the interacting application is developed in a language other than Java.
3. Here, a Java server hosts an instance of the HBase client brokering application Avro requests in and out of the HBase cluster.
4. This extra work of proxying requests and responses makes these interfaces slower.
5. The Avro server is started by putting up a server to field Avro clients.
6. The Avro server by default uses port 9090.

**C. REST :**

1. HBase is shipped with REST interface.
2. REST interface is useful when the interacting application is developed in a language other than Java.
3. Here, a Java server hosts an instance of the HBase client brokering application REST requests in and out of the HBase cluster.



4. This extra work of proxying requests and responses makes these interfaces slower.
5. The REST server is started by putting up a stargate instance (stargate is the name for the HBase REST service).
6. The REST server by default uses port 8080.

**D. Thrift :**

1. HBase is shipped with Thrift interface.
2. Thrift interface is useful when the interacting application is developed in a language other than Java.
3. Here, a Java server hosts an instance of the HBase client brokering application Thrift requests in and out of the HBase cluster.
4. This extra work of proxying requests and responses makes these interfaces slower.
5. We start a Thrift service by putting up a server to field Thrift clients.
6. The Thrift server by default uses port 9090.

**Que 5.28. Differentiate between HBase and RDBMS.**

**Answer**

| S. No. | HBase | RDBMS |
|---|---|---|
| 1. | HBase does not require SQL. | RDBMS requires SQL. |
| 2. | HBase has no fixed schema. | RDBMS has a fixed schema. |
| 3. | HBase is column-oriented. | It is row-oriented. |
| 4. | It is highly scalable. | It is low scalable. |
| 5. | It is dynamic in nature. | It is static in nature. |
| 6. | It follows CAP Rule *i.e.*, consistency, availability, partition-tolerance. | It follows ACID rule *i.e.*, atomicity, consistency, isolation and durability. |

**Que 5.29. Explain schema design for HBase.**

**Answer**

1. HBase tables are similar to those in an RDBMS.
2. An exception is that the cells are versioned and rows are sorted.
3. Columns can be added on the fly as long as the column family they belong to preexists.

4. These factors should be considered when designing schemas for HBase.
5. Also another important concern in designing schemas is consideration of how the data will be accessed.
6. In HBase all access is via primary key so the key design should lend itself to how the data is going to be queried.
7. The other property to keep in mind when designing schemas is that HBase can host wide and sparsely populated tables at no incurred cost.

**Que 5.30. Write short note on advanced indexing.**

**Answer**

1. Advanced indexing is triggered when it object is a non-tuple sequence object or a tuple with at least one sequence object.
2. It returns a copy of data rather than a view of it.
3. There are two types of advanced indexing : Integer and Boolean.
   a. **Integer Indexing :**
      i. This mechanism helps in selecting any arbitrary item in an array based on its N-dimensional index.
      ii. Each integer array represents the number of indexes into that dimension.
      iii. When the index consists of as many integer arrays as the dimensions of the targeted arrays, it becomes straightforward.
   b. **Boolean Array Indexing :**
      i. This type of advanced indexing is used when the resultant object is meant to be the result of Boolean operations, such as comparison operators.

**Que 5.31. Write short note on ZooKeeper.**

**Answer**

1. Apache ZooKeeper is a client-server system for distributed coordination that exposes an interface similar to a filesystem, where each node (called a znode) may contain data and a set of children.
2. Each znode has a name and can be identified using a filesystem-like path.
3. Zookeeper coordinates, communicates, and shares state between the Masters and RegionServers.
4. HBase has a design policy of using ZooKeeper only for transient data (that is, for coordination and state communication).



5. Thus if the HBase's ZooKeeper data is removed, only the transient operations are affected, data can continue to be written and read from HBase.
6. ZooKeeper provides an interactive shell that allows us to explore the ZooKeeper state, run it by using HBase and walk through the znode in a filesystem.
7. HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning.
10. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization.

**Que 5.32. How ZooKeeper helps in monitoring a cluster ?**

**Answer**

1. Apache ZooKeeper is an open-source server that reliably coordinates distributed processes and applications.
2. It allows distributed processes to coordinate with each other through a shared hierarchal namespace which is organized similarly to a standard file system.
3. Apache ZooKeeper provides a hierarchical file system (with ZNodes as the system files) that helps with the discovery, registration, configuration, locking, leader selection, queuing, etc of services working in different machines.
4. ZooKeeper server maintains configuration information, naming, providing distributed synchronization, and providing group services, used by distributed applications.
5. Applications Manager aims to help administrators manage their ZooKeeper server-collect all the metrics that can help with troubleshooting, display performance graphs and be alerted automatically of potential issues.

**Monitoring a cluster :**

1. Track the number of ZNodes, the watcher setup over the nodes and the number of followers within the ensemble.
2. Keep an eye on the leader selection stats and client session times.
3. Know where the Leader is for a quorum, and when there is a change in Leaders.
4. Get alerts on the number of active, connected sessions, and measure the growth rate over a specific time period.

## PART-4

*IBM Big Data Strategy, Introduction to Infosphere, BigInsights and Big Sheets, Introduction to Big SQL.*

### Questions-Answers

### Long Answer Type and Medium Answer Type Questions

**Que 5.33. Explain IBM big data strategy.**

**Answer**

1. The Big data strategy of the company was to combine a wide array of the Big data analytic solutions and conquer the Big data market. The company's goal was to offer the broadest portfolio of products and solutions with the depth and breadth that no other company could match.
2. In 2013, IBM was awarded the contract to support Thames Water Utilities Limited's (Thames Water) Big data project.
3. The UK government planned to install smart meters in every home by 2020. Using these meters, the company would be able to collect a lot of data about the consumption patterns of its customers.
4. As a part of its next five-year plan, Thames Water planned to invest in Big data analytics to improve its operations, customer communication, services, and customer satisfaction using this data.
5. It chose IBM as an alliance partner for the project to support technology and innovation.
6. IBM had brought in new systems, software, and services to complement its Big data platform.
7. With these products it helped its customers to access and analyze data and use it to make informed decisions for the betterment of their businesses.
8. The Big data solutions were also meant to protect data and identify and restrict suspicious activity and block access to company data.

**Que 5.34. Write a short note on : InfoSphere.**



**Answer**

1. InfoSphere Information Server is a leading data integration platform with offerings that help you understand, cleanse, monitor, and transform data.
2. InfoSphere Information Server provides massively parallel processing (MPP) capabilities for a highly scalable and flexible integration platform that handles all data volumes, big and small.
3. This powerful, scalable extract, transform, load (ETL) platform helps you get flexible and near real-time integration of all types of data, deployable on premises or in the cloud.
4. It uses a standardized approach to discover your IT assets and define a common business language for your data.
5. It helps us to get a better understanding of current data assets, while improving integration with related products.
6. It helps us to derive more meaning from your enterprise data through integrated rules analysis on a scalable platform that supports heterogeneous data.

**Que 5.35. Write a short note on : BigInsights.**

**Answer**

1. BigInsights is a software platform for discovering analyzing and visualizing data from disparate sources.
2. By using BigInsights users can extract new insights from this data to enhance knowledge of business.
3. IBM BigInsights on cloud is a fully managed Hadoop service.
4. IBM developed BigInsights to help firms process and analyze the increasing volume, variety, and velocity of data of interest to many enterprises.
5. To make sifting through large volumes of diverse data practical, BigInsights provides built-in analytic technologies and exploits shared-nothing hardware clusters.
6. It transparently distributes data stored in files across disks attached to various nodes in a cluster, directing sub-tasks of applications to processors that are close to the target subsets of your data.
7. This approach minimizes network traffic and improves runtime performance.
8. For fault tolerance, BigInsights automatically replicates each portion of your data on multiple disks based on parameters specified by an administrator.

9. BigInsights doesn't replace a relational database management system (DBMS) or a traditional data warehouse.
10. Rather, it's a platform that can augment your existing analytic infrastructure, enabling you to filter through high volumes of raw data and combine the results with structured data stored in your DBMS or warehouse.

**Que 5.36. What is BigSheets ?**

**Answer**

1. Big Sheets is a spreadsheet style data manipulation and visualization tool that allows business users to access and analyze data in Hadoop without the need to be knowledgeable in Hadoop scripting languages or MapReduce programming.
2. It is a browser-based analytics tool for business users.
3. Using built-in line readers, BigSheets can import data in multiple formats (JSON, CSV, TSV, ...).
4. Spreadsheet-like interface enables business users to gather and analyze data easily.
5. Users can combine and explore various types of data to identify "hidden" insights.
6. Following are various applications of BigSheets :
   i. Model "big data" collected from various sources in spreadsheet like structures.
   ii. Filter and enrich content with built-in functions.
   iii. Combine data in different workbooks.
   iv. Visualize results through spreadsheets, charts.
   v. Export data into common formats (if desired).

**Que 5.37. What is Big SQL ? How Big SQL works ?**

**Answer**

1. Big SQL is a high performance massively parallel processing SQL engine for Hadoop that makes querying enterprise data from across the organization an easy and secure experience.
2. It includes a unique smart scan service that minimizes data movement and minimizes performance by parsing and intelligently filtering data where it resides.



**Working of Big SQL :**

1. Big SQL's robust engine executes complex queries for relational data and Hadoop data.
2. Big SQL provides an advanced SQL compiler and a cost-based optimizer for efficient query execution.
3. Combining these with a massive parallel processing (MPP) engine helps distribute query execution across nodes in a cluster.

☺☺☺

1 UNIT

# Introduction (2 Marks Questions)

**1.1. What do you mean by Big data ?**
**OR**
**How you can define the term big data ?**

**Ans.** Big data is a collection of data that is huge in volume and 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.

**1.2. What are the advantages of Big data ?**

**Ans.** Following are the advantages of Big data :

1. It helps in optimizing business processes.
2. It helps in improving science and research.
3. It helps in financial tradings, sports, polling, security/law enforcement.

**1.3. What are the disadvantages of Big data ?**

**Ans.** Following are the disadvantages of Big data :

1. Traditional storage can cost lot of money to store big data.
2. Lots of Big data is unstructured.
3. Big data analysis violates principles of privacy.
4. It may increase social stratification.

**1.4. Explain major challenges of Big data.**

**Ans.** Following are the major challenges of Big data :

1. Data complexity
2. Data security
3. Data mobility

**1.5. Define digital data.**

**Ans.** Digital data is the information stored on a computer system as a series of 0's and 1's in a binary language.



**1.6. What is the strength of digital data ?**

**Ans.** The strengths of digital data is that all sorts of complex analog input can be represented with the binary system.

**1.7. What are the different types of digital data ?**

**Ans.** Following are the different types of digital data :

1. Structured digital data
2. Semi-Structured digital data
3. Unstructured digital data

**1.8. Define structured data.**

**Ans.** Structured data is data whose elements are addressable for effective analysis. It has been organized into a formatted repository *i.e.*, a database.

**1.9. Define semi-structured data.**

**Ans.** Semi-structured data is information that does not reside in a relational database but that have some organizational properties that make it easier to analyze.

**1.10. What is unstructured data ?**

**Ans.** Unstructured data is a data which is not organized in a predefined manner or does not have a predefined data model.

**1.11. What is Big data platform ?**

**Ans.** Big data platform is a type of IT solution that combines the features and capabilities of several Big data application and utilities within a single solution.

**1.12. What are the 5Vs of Big data ?**
**OR**
**List the characteristics of Big data.**

**Ans.** Following are the 5Vs of Big data :

1. Volume
2. Variety
3. Variability
4. Value
5. Veracity

**1.13. Define veracity.**

**Ans.** Veracity 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.

**1.14. What are different Big data technology ?**

**Ans.** Following are different Big data technology :

1. Operational Big data technology.
2. Analytical Big data technology.

**1.15. What are the components of Big data technology ?**

**Ans.** Following are the components of Big data technology :

1. Machine learning
2. Natural Language Processing (NLP)
3. Business intelligence
4. Cloud computing

**1.16. What are the application of Big data ?**

**OR**

**What comes under Big data application ?**

**Ans.** Following are the application of Big data :

1. Health care
2. Education
3. Weather
4. Communication, media and entertainment

**1.17. What are Big data risks ?**

**Ans.** Following are the Big data risks :

1. Data breaches
2. Data brokerage
3. Data discrimination

**1.18. Define intelligent data analysis.**

**Ans.** Intelligent Data Analysis (IDA) is a process to extract useful knowledge and reveals implicit, previously unknown and potentially valuable information or knowledge from large amount of data.

**1.19. What are the tools used for data analytics ?**

**Ans.** Following are the tools used for data analytics :

1. **Hadoop :** It helps in storing and analyzing data.
2. **MongoDB :** It is used on datasets that change frequently.
3. **Talend :** It is used for data integration and management.
4. **Cassandra :** A distributed database used to handle chunks of data.
5. **Spark :** It is used for real-time processing and analyzing large amounts of data.



**1.20. Explain the difference between operational and analytical system.**

**Ans.**

| S. No. | Operational system | Analytical system |
|---|---|---|
| 1. | It is based on relational paradigm. | It is based on dimensional paradigms. |
| 2. | It store real-time transactional data. | It store historical transactional data. |

2 UNIT

# Hadoop (2 Marks Questions)

**2.1. What do you mean by Hadoop ?**

**Ans.** 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.

**2.2. What is Hadoop architecture ?**

**Ans.** Hadoop architecture is a package of the file system, MapReduce engine and HDFS.

**2.3. What are the layers of Hadoop ?**

**Ans.** Following are the layers of Hadoop :

1. Processing/Computation layer
2. Storage layer

**2.4. What are the advantages of Hadoop ?**

**Ans.** Following are the advantages of Hadoop :

1. It allows the user to quickly write and test distributed systems.
2. It does not rely on hardware to provide Fault-Tolerance and High Availability (FTHA).
3. Servers can be added or removed from the cluster dynamically.

**2.5. What are the disadvantages of Hadoop ?**

**Ans.** Following are the disadvantages of Hadoop :

1. Issue with small files
2. Vulnerable by nature
3. Processing overhead
4. Supports only batch processing
5. Iterative processing



**2.6. Define Apache Hadoop.**

**Ans.** Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data.

**2.7. Define Hadoop Distributed File System.**

**Ans.** The Hadoop Distributed File System (HDFS) is Google File System (GFS) that provides a distributed file system that is designed to run on commodity hardware.

**2.8. What are the features of HDFS ?**

**Ans.** Following are the features of HDFS :

1. It is suitable for the distributed storage and processing.
2. Hadoop provides a command interface to interact with HDFS.
3. The built-in servers of namenode and datanode help users to easily check the status of cluster.
4. Streaming access to file system data.

**2.9. What are the goals of HDFS ?**

**Ans.** Following are the Goals of HDFS :

1. Fault detection and recovery
2. Huge datasets
3. Hardware at data

**2.10. What are the components of Hadoop data format ?**

**Ans.** Following are the components of Hadoop data format :

1. Text/CSV
2. SequenceFile
3. Avro
4. Parquet

**2.11. What is Hadoop streaming ?**

**Ans.** Hadoop streaming is a utility that allows us to create and run Map Reduce jobs with any executable or script as the mapper and the reducer.

**2.12. Define Hadoop pipes ?**

**OR**

**State the usage of Hadoop pipes.**

**OR**

**State the purpose of Hadoop pipes.**

**Ans.** Hadoop Pipes is the name of the C++ interface to Hadoop MapReduce. Hadoop Pipes allows C++ code to use Hadoop DFS and map/reduce function.

**2.13. Define Hadoop Ecosystem.**

**Ans.** 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.

**2.14. What are the elements of Hadoop ?**

**Ans.** Following are the elements of Hadoop :

1. HDFS
2. MapReduce
3. YARN
4. PIG
5. HIVE

**2.15. What is MapReduce ?**

**Ans.** MapReduce is a framework that allows us to write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on Java.

**2.16. What are the different phases of MapReduce ?**

**Ans.** Following are the different phases of MapReduce :

1. Input splits
2. Mapping
3. Shuffling and
4. Reducing

**2.17. What are the types of task used in Hadoop ?**

**Ans.** Hadoop divides the job into tasks. There are two types of tasks :

1. Map tasks (Splits and Mapping)
2. Reduce tasks (Shuffling, Reducing)

**2.18. What are the types of schedulers in Hadoop ?**

**Ans.** Following are the scheduler in Hadoop :

1. FIFO scheduler
2. Capacity scheduler
3. Fair scheduler



**2.19. What are the features of MapReduce ?**

**Ans.** Following are the features of MapReduce :

1. Scalability
2. Flexibility
3. Security and authentication
4. Cost-effective solution

**2.20. List down the tools related with Hadoop.**

**Ans.** Following are the tools related with Hadoop :

1. Apache Hive
2. MapReduce
3. Apache Spark
4. Apache Flume

3 UNIT

# HDFS (Hadoop Distributed File System) (2 Marks Questions)

**3.1. Define HDFS.**

**OR**

**Explain Hadoop distributed file system.**

**Ans.** HDFS is a distributed file system that handles large data sets running on commodity hardware. It is used to scale a single Apache Hadoop cluster to more number of nodes.

**3.2. What are the featues of HDFS ?**

**Ans.** Following are the Features of HDFS :

1. Distributed data storage.
2. Blocks reduce seek time.
3. The data is highly available as the same block history server at multiple datanodes.
4. High fault tolerance.

**3.3. Define NameNode.**

**Ans.** The nameNode is the commodity hardware that contains the GNU/Linux operating system and the namenode software.

**3.4. Define the task of namenode.**

**Ans.** NameNode performs the following task :

1. It manages the file system namespace.
2. It regulates client's access to files.
3. It also executes file system operations such as renaming, closing, and opening files and directories.

**3.5. Define Datanode.**

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

**3.6. What is a block ?**



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

**3.7. Define block abstraction memoization.**

**Ans.** Block Abstraction Bemoization (BAM) is a technique in program verification that divides a program into blocks (like functions or loops) and analyses them separately.

**3.8. Define data replication.**

**Ans.** Data replication in HDFS increases the availability of Data at any point of time. It ensures the availability of the data. Replication is making a copy of something and the number of times we make a copy of that particular thing can be expressed as it is Replication Factor.

**3.9. Define command line interface.**

**Ans.** A command line interface (CLI) is a text-based user interface (UI) used to view and manage computer files. Command line interfaces are also called command-line user interfaces, console user interfaces and character user interfaces.

**3.10. Define Apache SQOOP.**

**Ans.** Apache SQOOP (SQL-to-Hadoop) is a tool designed to support bulk export and import of data into HDFS from structured data stores such as relational databases, enterprise data warehouses, and NoSQL systems.

**3.11. What are the tools used for data ingestion ?**

**Ans.** Following are the tools used for data ingestion :

1. Apache NiFi
2. StreamSets Data Collector (SDC)
3. Gobblin
4. SQOOP

**3.12. What is Hadoop Archives ?**

**Ans.** Hadoop Archives (HAR) is used to address the namespace limitations associated with storing many small files. Hadoop Archive is also compatible with Map Reduce, it allows parallel access to the original files by Map Reduce jobs.

**3.13. What is Hadoop I/O ?**

**Ans.** A Hadoop FormatIO is a transform for reading data from any source or writing data to any sink that implements Hadoop's InputFormat or OurputFormat accordingly. HadoopFormatIO allows us to

connect to many data sources/sinks that do not yet have a Beam IO transform.

**3.14. What are the compression format in Hadoop ?**

**Ans.** Following are the compression format in Hadoop :
1. GZIP
2. BZIP2
3. LZO
4. SNAPPY

**3.15. Define serialization.**

**Ans.** Serialization refers to the conversion of structured objects into byte streams for transmission over the network or permanent storage on disk.

**3.16. Define Auditing.**

**Ans.** Auditing is the process of keeping track of what an authenticated, authorized user did once he gets access to the cluster. It records all the activity of the authenticated user, including what data was accessed, added, changed, and what analysis occurred by the user from the period when he login to the cluster.

**3.17. What are the tools used in Hadoop security ?**

**Ans.** Following are the tools used in Hadoop security :
1. Knox
2. Ranger





# 4 UNIT

# Hadoop Ecosystem and YARN (2 Marks Questions)

**4.1. Define YARN.**

**Ans.** Yet Another Resource Negotiator (YARN) is the one who helps to manage the resources across the clusters.

**4.2. List down the entity of YARN.**

**Ans.** Following are the entity of YARN :
1. Client
2. Resource Manager
3. Node Manager
4. Application Manager
5. Container

**4.3. What are the components of YARN ?**

**Ans.** Fcllowing are the components of YARN :
1. Resource manager
2. Nodes manager
3. Application manager

**4.4. Define MRv2.**

**Ans.** The MapReduce v2 (MRv2) or YARN architecture splits the two primary responsibilities of the JobTracker. *i.e.*, resource management and job scheduling/monitoring into separate daemons a global ResourceManager and per-application ApplicationMasters.

**4.5. Define NoSQL database.**

**Ans.** A NoSQL originally referring to non SQL or non relational is a database that provides a mechanism for storage and retrieval of data. This data is modeled in means other than the tabular relations used in relational databases.

**4.6. What are the advantages of NoSQL ?**

**Ans.** Following are the advantages of NoSQL :
1. High scalability
2. High availability

**4.7. What are the types of NoSQL ?**

**Ans.** Following are the types of NoSQL :
1. Key Value Pair Based
2. Column-based
3. Document-Oriented
4. Graph-Based

**4.8. Define Mongo database.**

**Ans.** MongoDB is a No SQL database. It is an open-source, cross-platform, document-oriented database written in C++. MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

**4.9. What are the data types used in Mongo database.**

**Ans.** MongoDB supports many data types such as :
1. String
2. Integer
3. Boolean
4. Double

**4.10. Define capped collection.**

**Ans.** Capped collections are fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order.

**4.11. Define Spark.**

**Ans.** Spark is a fast and general processing engine compatible with Hadoop data. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop InputFormat.

**4.12. What are the application of Spark ?**

**Ans.** Following are the application of Spark :
1. Machine Learning
2. Fog computing
3. Event detection
4. Interactive analysis

**4.13. Define resilient distributed database.**

**Ans.** Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects.

**4.14. What are the components of Spark ?**

**Ans.** Following are the components of Spark :
1. Spark Core
2. Spark SQL
3. Spark Streaming
4. GraphX
5. MLlib

**4.15. Compare master-slave and peer-to-peer architecutre in NoSql.**



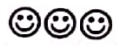
**Ans.**

| S. No. | Master-slave architecture | Peer-to-peer architecture |
|---|---|---|
| 1. | Master-slave architecture makes one node the authoritative copy that handles writes while slaves synchronize with the master and may handle reads. | Peer-to-peer architecture allows writes to any node, the nodes coordinate to synchronize their copies of data. |

**4.16. Compare the classic MapReduce with YARN.**

**Ans.**

| S. No. | MapReduce | YARN |
|---|---|---|
| 1. | It support batch processing application only. | It support variety of processing engines and application. |
| 2. | It provides static allocation of resources. | It provides dynamic allocation of resources. |



UNIT 5

# Applications on Big Data using Pig, Hive and HBase (2 Marks Questions)

**5.1. What is a PIG ?**

**Ans.** PIG is an abstraction over Map Reduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows.

**5.2. Define Grunt.**

**Ans.** Grunt Shell is a Shell Command. The Grunt shell of Apache Pig is mainly used to write Pig Latin scripts.

**5.3. Mention the usage of Grunt.**

**Ans.** Grunt is a JavaScript task runner, a tool used to automatically perform frequent tasks such as minification, compilation and unit testing.

**5.4. What are user defined function used in PIG ?**

**Ans.** Following are the functions used in PIG :

1. Filter Functions
2. Eval Functions
3. Algebraic Functions

**5.5. What are different operators used in PIG ?**

**Ans.** Following are the operators used in PIG :

1. Dump operator
2. Describe operators
3. Explain operators
4. Illustrate operators
5. Group operator

**5.6. Define HIVE.**

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



**5.7. What is a Hive shell ?**

**Ans.** Hive shell is a primary way to interact with Hive. It is a default service in hive. It is also called as CLI (command line interference).

**5.8. What are the Hive services ?**

**Ans.** The following are the services provided by Hive :

1. Hive CLI
2. Hive web user interface
3. Hive metastore
4. Hive server
5. Hive driver
6. Hive compiler
7. Hive execution engine

**5.9. How Date and Time data types are used in Hive ?**

**Ans.** Date data types are represented in the form YYYY-MM-DD. Time data types uses the format yyyy-mm-dd.

**5.10. Why Hive is preferred instead of Pig Latin ?**

**Ans.** Hive is preferred instead of Pig Latin as Hive suits the specific demands of analytics.

**5.11. Define function aggregating.**

**Ans.** An aggregate function is a function that summarizes the rows of a group into a single value. COUNT, MIN and MAX are examples of aggregate functions.

**5.12. Define subquery.**

**Ans.** A Subquery or Inner query or Nested query is a query within SQL query and embedded within the WHERE clause. A Subquery is a SELECT statement that is embedded in a clause of another SQL statement.

**5.13. What is Hbase ?**

**Ans.** 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.