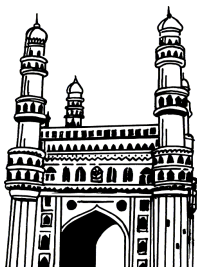
Rahul's ✔
Topper's Voice



M.C.A.

II Year IV Semester

Latest 2022 Edition

BIG DATA ANALYTICS

- ☞ **Study Manual**
- ☞ **Important Questions**
- ☞ **Solved Model Papers**

~~₹225/-~~
₹169/-

- by -

WELL EXPERIENCED LECTURER

***Rahul Publications*™**
Hyderabad. Ph : 66550071, 9391018098

M.C.A.

II Year IV Semester

BIG DATA ANALYTICS

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

Price : ~~` 225/-~~
` 169/-

Sole Distributors :

☎ : 66550071, Cell : 9391018098

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.
Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

BIG DATA ANALYTICS

CONTENTS

STUDY MANUAL

SOLVED MODEL PAPERS

SYLLABUS

UNIT - I

Getting an overview of Big Data: Introduction to Big Data, Structuring Big Data, Types of Data, Elements of Big Data, Big Data Analytics, Advantages of Big Data Analytics.

Introducing Technologies for Handling Big Data: Distributed and Parallel Computing for Big Data, Cloud Computing and Big Data, Features of Cloud Computing, Cloud Deployment Models, Cloud Services for Big Data, Cloud Providers in Big Data Market.

UNIT - II

Understanding Hadoop Ecosystem: Introducing Hadoop, HDFS and MapReduce, Hadoop functions, Hadoop Ecosystem.

Hadoop Distributed File System- HDFS Architecture, Concept of Blocks in HDFS Architecture, Namenodes and Datanodes, Features of HDFS. MapReduce.

Introducing HBase - HBase Architecture, Regions, Storing Big Data with HBase, Combining HBase and HDFS, Features of HBase, Hive, Pig and Pig Latin, Sqoop, ZooKeeper, Flume, Oozie.

UNIT - III

Understanding MapReduce Fundamentals and HBase: The MapReduce Framework ,Exploring the features of MapReduce, Working of MapReduce, Techniques to optimize MapReduce Jobs, Hardware/Network Topology, Synchronization, File system, Uses of MapReduce, Role of HBase in Big Data Processing- Characteristics of HBase.

Understanding Big Data Technology Foundations: Exploring the Big Data Stack, Data Sources Layer, Ingestion Layer, Storage Layer, Physical Infrastructure Layer, Platform Management Layer, Security Layer, Monitoring Layer, Visualization Layer.

UNIT - IV

Storing Data in Databases and Data Warehouses: RDBMS and Big Data, Issues with Relational Model, Non – Relational Database, Issues with Non Relational Database, Polyglot Persistence, Integrating Big Data with Traditional Data Warehouse, Big Data Analysis and Data Warehouse.

UNIT - V

NoSQL Data Management: Introduction to NoSQL, Characteristics of NoSQL, History of NoSQL, Types of NoSQL Data Models- Key Value Data Model, Column Oriented Data Model, Document Data Model, Graph Databases, Schema-Less Databases, Materialized Views, CAP Theorem.

Contents

UNIT - III

Important Questions

UNIT - I

1. Explain the evolution of Big Data.

Ans :

Refer Unit-I, Q.No. 2

2. What are various types of big data ?

Ans :

Refer Unit-I, Q.No. 4

3. What are the characteristics of Big Data?

Ans :

Refer Unit-I, Q.No. 7

4. What are the applications of Big Data?

Ans :

Refer Unit-I, Q.No. 11

5. Explain the relationship of big data with other areas?

Ans :

Refer Unit-I, Q.No. 13

6. Describe Distributed and Parallel Computing for Big Data.

Ans :

Refer Unit-I, Q.No. 14

7. Discuss the techniques of parallel computing.

Ans :

Refer Unit-I, Q.No. 15

8. Define Cloud Computing and Mention the advantages and disadvantages of cloud computing.

Ans :

Refer Unit-I, Q.No. 17

9. Explain about various web services involved in cloud computing.

Ans :

Refer Unit-I, Q.No. 23

10. Discuss the role cloud providers play in handling big data market.

Ans :

Refer Unit-I, Q.No. 28

UNIT - II

1. What is HDFS ? Explain HDFS architecture with neat diagram.

Ans :

Refer Unit-II, Q.No. 2

2. Explain about Mapreduce programming model.

Ans :

Refer Unit-II, Q.No. 3

3. How does Hadoop Function work in big data?

Ans :

Refer Unit-II, Q.No. 4

4. Explain briefly about Hadoop Ecosystem.

Ans :

Refer Unit-II, Q.No. 6

5. Explain concept of Blocks in HDFS Architecture.

Ans :

Refer Unit-II, Q.No. 9

6. Discuss the Features of HDFS.

Ans :

Refer Unit-II, Q.No. 11

7. Explain the HBase Architecture.

Ans :

Refer Unit-II, Q.No. 14

8. Explain about Combining HBase and HDFS.

Ans :

Refer Unit-II, Q.No. 15

9. Write the step by step installation process of Hive.

Ans :

Refer Unit-II, Q.No. 19

10. Define OOZIE. State the components of OOZIE.

Ans :

Refer Unit-II, Q.No. 27

11. State the benefits of OOZIE.

Ans :

Refer Unit-II, Q.No. 28

UNIT - III

1. List the features of MapReduce.

Ans :

Refer Unit-III, Q.No. 2

2. Describe the working of the MapReduce Algorithm.

Ans :

Refer Unit-III, Q.No. 3

3. Explain various Techniques to optimize MapReduce Jobs.

Ans :

Refer Unit-III, Q.No. 4

4. Explain the Characteristics of HBase.

Ans :

Refer Unit-III, Q.No. 7

5. Discuss the various layers in big data architecture.

Ans :

Refer Unit-III, Q.No. 8

6. Explain the functioning of Ingestion Layer in the big data architecture.

Ans :

Refer Unit-III, Q.No. 10

7. Explain briefly about Physical Infrastructure Layer in big data architecture.

Ans :

Refer Unit-III, Q.No. 12

8. Explain briefly about Security Layer.

Ans :

Refer Unit-III, Q.No. 14

UNIT - IV

1. Explain the relationship between RDBMS and Big Data.

Ans :

Refer Unit-IV, Q.No. 7

2. Explain the issues with Relational Model.

Ans :

Refer Unit-IV, Q.No. 9

3. What are called Non-Relational Database?

Ans :

Refer Unit-IV, Q.No. 11

4. Explain the advantages and disadvantges of Non Relational Database.

Ans :

Refer Unit-IV, Q.No. 12

5. What is the Difference Between Relational and Non-Relational Database?

Ans :

Refer Unit-IV, Q.No. 13

6. Discuss various issues with Non Relational Database.

Ans :

Refer Unit-IV, Q.No. 14

7. How big data integrating with traditional warehouse.

Ans :

Refer Unit-IV, Q.No. 16

8. Explain briefly about Big Data Analysis and Data Warehouse.

Ans :

Refer Unit-IV, Q.No. 17

UNIT - V

1. Explain the evolution of NoSQL.

Ans :

Refer Unit-V, Q.No. 3

2. What are the advantages and disadvantages of NoSQL?

Ans :

Refer Unit-V, Q.No. 4

3. Explain the Use of NoSQL in industry sector.

Ans :

Refer Unit-V, Q.No. 6

4. Discuss about Key Value Data Model.

Ans :

Refer Unit-V, Q.No. 9

5. Explain briefly about Column Oriented Data Model.

Ans :

Refer Unit-V, Q.No. 10

6. Discuss in detail about Graph Database.

Ans :

Refer Unit-V, Q.No. 13

7. What are Materialized Views in NoSQL.

Ans :

Refer Unit-V, Q.No. 15

8. Discuss about CAP Theorem.

Ans :

Refer Unit-V, Q.No. 16

UNIT I

Getting an overview of Big Data: Introduction to Big Data, Structuring Big Data, Types of Data, Elements of Big Data, Big Data Analytics, Advantages of Big Data Analytics.

Introducing Technologies for Handling Big Data: Distributed and Parallel Computing for Big Data, Cloud Computing and Big Data, Features of Cloud Computing, Cloud Deployment Models, Cloud Services for Big Data, Cloud Providers in Big Data Market.

1.1 Introduction to Big Data

Q1. What is Big Data?

Ans :

Definition

According to Gartner:

"Big data" is high-volume, velocity, and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."

Big Data refers to complex and large data sets that have to be processed and analyzed to uncover valuable information that can benefit businesses and organizations.

However, there are certain basic tenets of Big Data that will make it even simpler to answer what is Big Data:

- It refers to a massive amount of data that keeps on growing exponentially with time.
- It is so voluminous that it cannot be processed or analyzed using conventional data processing techniques.
- It includes data mining, data storage, data analysis, data sharing, and data visualization.
- The term is an all-comprehensive one including data, data frameworks, along with the tools and techniques used to process and analyze the data.

Q2. Explain the evolution of Big Data.

Ans : **(Imp.)**

The advent of IT, the Internet, and globalization has facilitated increased volumes of data and information generation at an exponential rate, which has led to 'information explosion.' This, in turn, fueled the evolution of Big Data that started in 1940s and continues till date.

Information explosion is described as a continuous increase in the volume of the published information or data and the effects of this abundant information.

Table lists some major milestones in the evolution of Big Data

Year	Milestone
1940s	An American librarian speculated the potential shortfall of shelves and cataloging staff, realizing the rapid increase in information and limited storage.
1960s	Automatic Data Compression was published in the Communications of the ACM. It states that the explosion of information in the past few years makes it necessary that requirements for storing information should be minimized. The paper described 'Automatic Data Compression' as a complete automatic and fast three-part compressor that can be used for any kind of information in order to reduce the slow external storage requirements and increase the rate of transmission from a computer system.
1970s	In Japan, the Ministry of Posts and Telecommunications initiated a project to study information flow in order to track the volume of information circulating in the country.
1980s	A research project was started by the Hungarian Central Statistics Office to account for the country's information industry. It measured the volume of information in bits.
1990s	Digital storage systems became more economical than paper storage. Challenges related to the amount of data and the presence of obsolete data became apparent. Some papers that discussed this concern are as follows: ➢ Michael Lesk published How much information is there in the world? ➢ John R. Masey presented a paper titled Big Data... and the Next Wave of InfraStress. ➢ K.G. Coffman and Andrew Odlyzko published The Size and Growth Rate of the Internet. ➢ Steve Bryson, David Kenwright, Michael Cox, David Ellsworth, and Robert Haimes published Visually Exploring Gigabyte Datasets in Real Time.
2000s	Many researchers and scientists published papers raising similar concerns and onwards discussing ways to solve them. Various methods were introduced to streamline information. Techniques for controlling the Volume, Velocity, and Variety of data emerged, thus introducing 3D data management. A study was carried out in order to estimate the new and original information created and stored worldwide in four types of physical media: paper, film, optical media, and magnetic media.

1.1.1 Structuring Big Data

Q3. Explain the Structuring of Big Data.

Ans :

Structuring of data, in simple terms, is arranging the available data in a manner such that it becomes easy to study, analyze, and derive conclusion from it.

In daily life, you may have come across questions like:

➢ How do I use to my advantage the vast amount of data and information I come across?

- Which news articles should I read of the thousands I come across?
- How do I choose a book of the millions available on my favorite sites or stores?
- How do I keep myself updated about new events, sports, inventions, and discoveries taking place across the globe?

These systems can analyze and structure a large amount of data specifically for you on the basis of what you searched, what you looked at, and for how long you remained at a particular page or website, thus scanning and presenting you with the customized information as per your behavior and habits.

In other words, structuring data helps in understanding user behaviors, requirements, and preferences to make personalized recommendations for every individual.

When a user regularly visits or purchases from online shopping sites, say eBay, each time he/she logs in, the system can present a recommended list of products that may interest the user on the basis of his/her earlier purchases or searches, thus presenting a specially customized recommendation set for every user.

1.1.2 Types of Data

Q4. What are various types of big data ?

Ans : **(Imp.)**

Big data could be found in three forms:

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

1. Structured data

Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. Over the period of time, talent in computer science have achieved greater success in developing techniques for working with such kind of data (where the format is well known in advance) and also deriving value out of it. However, now days, we are foreseeing issues when size of such data grows to a huge extent, typical sizes are being in the rage of multiple zettabyte.

Looking at these figures one can easily understand why the name 'Big Data' is given and imagine the challenges involved in its storage and processing.

Examples

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

2. Unstructured data

Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it. Typical example of unstructured data is, a heterogeneous data source containing a combination of simple text files, images, videos etc. Now a day organizations have wealth of data available with them but unfortunately they don't know how to derive value out of it since this data is in its raw form or unstructured format.

Examples of Un-structured Data

Output returned by 'Google Search'

3. Semi-structured data

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

Examples of Semi-structured Data

Personal data stored in a XML file-

Q5. Discuss the challenges of unstructured data.

Ans :

The challenges of unstructured data run the gamut from gathering to storing, to using it to make decisions:

1. Relevance

One way in which relevance comes into play is lack of insight into "backstory" of certain pieces of data. For instance, a student might do a search on a particular topic or product to gather information for a school paper, and then never search for those keywords again. If so, that search would be irrelevant to any subsequent consumer behaviour, but the computers doing Big Data analysis wouldn't know that. The system assumes a relationship that simply wasn't there.

Another big challenge in working with unstructured data comes into play with machine learning and highlights the importance of knowing which factors actually drive consumer behaviour. It's the classic "correlation or causation" dilemma on steroids. An analytic model could give too much weight to factors that are merely correlated, and, thanks to machine learning, the more the correlation is noted, the more weight it's given. But, since there is no actual causation, the conclusions are inaccurate, and they become more so as time goes on.

2. Volume

For many businesses, that's more than they can keep up with, and they may be collecting information they're not even aware of. That presents challenges for both using and securing the data. The lack of awareness makes it more likely for enterprises to run afoul of the increasing number of regulations addressing data privacy. Such a large volume of data also requires infrastructure that many businesses don't currently have, or haven't budgeted for.

3. Quality

By nature, a large volume of unstructured data is unverified. There are plenty of jokes about "Facebook lives," in which a person's Facebook updates are more fantasy than reality. One effect of growing privacy concerns is the tendency for people to make up details for their profiles, in which even the hard "facts" – like marital status and hometown – can be completely false. This presents serious challenges for consumers and enterprises. On a consumer level, people could be negatively impacted by companies that make decisions

based on flimsy unstructured data, like using a person's social media posts to help determine insurance rates. On an enterprise level, making business decisions based on inaccurate data could be extremely costly.

4. Usability

For unstructured data to be usable, businesses will have to come up with a way to locate, extract, organise, and store the data. This means coming up with an entirely new type of database to store information that doesn't fit the mould.

Unstructured Big Data isn't going away. And that's a good thing, because it holds the opportunity for greatly enhanced planning and decision-making. Contact us to develop and execute a plan for using Big Data rather than falling back on the "drinking from a fir hose" when much data is coming so fast that is becomes useless.

Q6. Explain the importance of Big Data.

Ans : **(Imp.)**

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

1. **Cost Savings:** Some 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 and these tools also help in identifying more efficient ways of doing business.
2. **Time Reductions:** 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**: By analyzing big data you can get a better understanding of current market conditions. For example, by analyzing customers' purchasing behaviors, a company can find out the products that are sold the most and produce products according to this trend. By this, it can get ahead of its competitors.
4. **Control online reputation:** Big data tools can do sentiment analysis. Therefore, you can get feedback about who is saying what about your company. If you want to monitor and improve the online presence of your business, then, big data tools can help in all this.
5. **Using Big Data Analytics to Boost Customer Acquisition and Retention:** The customer is the most important asset any business depends on. There is no single business that can claim success without first having to establish a solid customer base. However, even with a customer base, a business cannot afford to disregard the high competition it faces. If a business is slow to learn what customers are looking for, then it is very easy to begin offering poor quality products. In the end, loss of clientele will result, and this creates an adverse overall effect on business success. The use of big data allows businesses to observe various customer related patterns and trends. Observing customer behavior is important to trigger loyalty.
6. **Using Big Data Analytics to Solve Advertisers Problem and Offer Marketing Insights:** Big data analytics can help change all business operations. This includes the ability to match customer expectation, changing company's product line and of course ensuring that the marketing campaigns are powerful.
7. **Big Data Analytics As a Driver of Innovations and Product Development:** Another huge advantage of big data is the ability to help companies innovate and redevelop their products.

1.1.3 Elements of Big Data

Q7. Write about the elements of Big Data.

(OR)

What are the characteristics of Big Data?

Ans : **(Imp.)**

The underlying characteristics of big data include:

i) Volume

Big data is a form of data whose volume is so large that it would not fit on a single machine therefore specialized tools and frameworks are required to store process and analyze such data. For example, social media applications process billions of messages everyday, industrial and energy systems can generate terabytes of sensor data everyday, cab aggregation applications can process millions of transactions in a day, etc. The volumes of data generated by modern

IT, industrial, healthcare, Internet of Things, and other systems is growing exponentially driven by the lowering costs of data storage and processing architectures and the need to extract valuable insights from the data to improve business processes, efficiency and service to consumers. Though there is no fixed threshold for the volume of data to be considered as big data, however, typically, the term big data is used for massive scale data that is difficult to store, manage and process using traditional databases and data processing architectures.

ii) Velocity

Velocity of data refers to how fast the data is generated. Data generated by certain sources can arrive at very high velocities, for example, social media data or sensor data. Velocity is another important characteristic of big data and the primary reason for the exponential growth of data. High velocity of data results in the volume of data accumulated to become very large, in short span of time. Some applications can have strict deadlines for data analysis (such as trading or online fraud detection) and the data needs to be analyzed in real-time. Specialized tools are required to ingest such high velocity data into the big data infrastructure and analyze the data in real-time.

iii) Variety

Variety refers to the forms of the data. Big data comes in different forms such as structured, unstructured or semi-structured, including text data, image, audio, video and sensor data. Big data systems need to be flexible enough to handle such variety of data.

iv) Veracity

Veracity refers to how accurate is the data. To extract value from the data, the data needs to be cleaned to remove noise. Data-driven applications can reap the benefits of big data only when the data is meaningful and accurate. Therefore, cleansing of data is important so that incorrect and faulty data can be filtered out.

v) Value

Value of data refers to the usefulness of data for the intended purpose. The end goal of any big data analytics system is to extract value from the data. The value of the data is also related to the veracity or accuracy of the data. For some applications value also depends on how fast we are able to process the data.

1.1.4 Big Data Analytics

Q8. What is Analytics?

Ans :

- Analytics is a broad term that encompasses the processes, technologies, frameworks and algorithms to extract meaningful insights from data.

- Raw data in itself does not have a meaning until it is contextualized and processed into useful information. Analytics is this process of extracting and creating information from raw data by filtering, processing, categorizing, condensing and contextualizing the data.
- This information obtained is then organized and structured to infer knowledge about the system and/or its users, its environment, and its operations and progress towards its objectives, thus making the systems smarter and more efficient.

The choice of the technologies, algorithms, and frameworks for analytics is driven by the analytics goals of the application.

For example, the goals of the analytics task may be:

1. To predict something (for example whether a transaction is a fraud or not, whether it will rain on a particular day, or whether a tumor is benign or malignant).
2. To find patterns in the data (for example, finding the top 10 coldest days in the year, finding which pages are visited the most on a particular website, or finding the most searched celebrity in a particular year).
3. Finding relationships in the data (for example, finding similar news articles, finding similar patients in an electronic health record system, finding related products on an eCommerce website, finding similar images, or finding correlation between news items and stock prices).

Q9. Explain briefly about big data analytics?

Ans :

Big data analytics reformed the ways to conduct business in many ways, such as it improves, decisions making, business process management, etc. Business analytics uses the data and different other techniques like information technology, features of statistics, quantitative methods, and different models to provide results. There are three main types of business analytics: descriptive analytics, predictive analytics, and prescriptive analytics.

There are mainly three types of analytics:

i) Descriptive Analytics

Descriptive analytics is the most prevalent form of analytics, and it serves as a base for advanced analytics. It answers the question 'What happened in the business?' Descriptive analytics analyses a database to provide information on the trends of past or current business events that can help managers, planners, leaders, etc. to develop a road map for future actions. Descriptive analytics performs an in-depth analysis of data to reveal details such as frequency of events, operation costs, and the underlying reason for failures. It helps in identifying the root cause of the problem?

ii) Predictive Analytics

Predictive analytics is about understanding and predicting the future and answers the question 'What could happen?' By using statistical models and different forecast techniques. It predicts the near future probabilities and trends and helps in what-if analysis. In predictive analytics, we use statistics, data mining techniques, and machine learning to analyze the future. Figure shows the steps involved in predictive analytics:

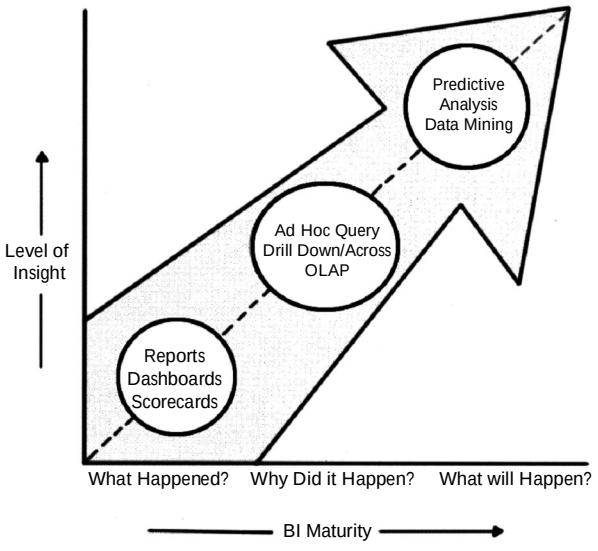


Fig.: Predictive Analytics

iii) Prescriptive Analytics

Prescriptive analysis answers 'What should we do? On the basis of complex data obtained from descriptive and predictive analyses. By using the optimization technique, prescriptive analytics determines the finest substitute to minimize or maximize some equitable finance, marketing, and many other areas. For example, if we have to find the best way of shipping goods from a factory to a destination, to minimize costs, we will use the prescriptive analytics. Figure shows a diagrammatic representation of the stages involved in the prescriptive analytics:

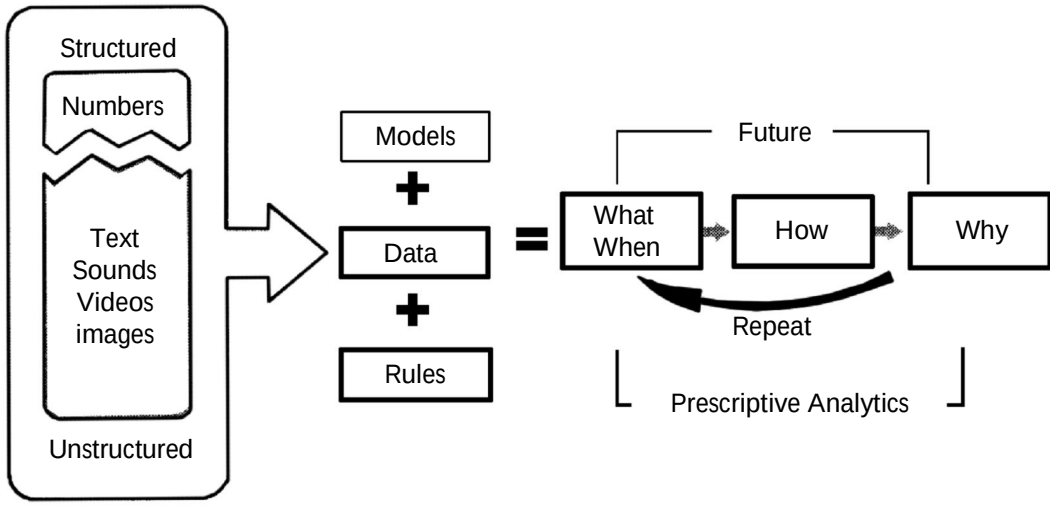


Fig.: Prescriptive Analytics

Data, which is available in abundance, can be streamlined for growth and expansion in technology as well as business. When data is analyzed successfully, it can become the answer to one of the most important questions: how can businesses acquire more customers and gain business insight? The key to this problem lies in being able to source, link, understand, and analyze data.

1.1.5 Advantages of Big Data Analytics

Q10. What are the Advantages of Big Data Analytics?

Ans :

The following are the advantages of Big Data Analytics:

- **Procurement:** To find out which suppliers are more efficient and cost-effective in delivering products on time.
- **Product Development:** To draw insights on innovative product and service formats and designs for enhancing the development process and coming up with demanded products.
- **Manufacturing:** To identify machinery and process variations that may be indicators of quality problems.
- **Distribution:** To enhance supply chain activities and standardize optimal inventory levels vis- a-vis various external factors such as weather, holidays, economy, etc.
- **Marketing:** To identify which marketing campaigns will be the most effective in driving and engaging customers and understanding customer behaviors and channel behaviors.
- **Price Management:** To optimize prices based on the analysis of external factors.

- **Merchandising:** To improve merchandise breakdown on the basis of current buying patterns and increase inventory levels and product interest insights on the basis of the analysis of various customer behaviors.
- **Sales:** To optimize assignment of sales resources and accounts, product mix, and other operations
- **Store Operations:** To adjust inventory levels on the basis of predicted buying patterns, study of demographics, weather, key events, and other factors
- **Human Resources:** To find you the characteristics and behaviors of successful and effective employees, as well as other employee insights for managing talent better.

Every business and industry today is affected by and benefitted from Big Data analytics in multiple ways.

Q11. What are the applications of Big Data?

Ans : **(Imp.)**

The following are the applications of Big Data:

i) Transportation

Big Data has greatly improved transportation services. The data containing traffic information is analyzed to identify traffic jam areas. Suitable steps can then be taken, on the basis of this analysis, to keep the traffic moving in such areas. Distributed sensors are installed in handled devices, on the roads and on vehicles to provide real-time traffic information. This information is analyzed and disseminated to commuters and also to the traffic control authority.

ii) Education

Big Data has transformed the modern-day education processes through innovative approaches, such as e-learning for teachers to analyze the students' ability to comprehend and thus impart education effectively in accordance with each student's needs. The analysis is done by studying the responses to questions, recording the time consumed in attempting those questions, and analyzing other behavioral signals of the students. Big Data also assists in analyzing the requirements and finding easy and innovative ways of imparting education, especially distance learning over vast geographical areas.

iii) Travel

The travel industry also uses Big Data to conduct business. It maintains complete details of all the customer records that are then analyzed to determine certain behavioral patterns in customers. For example, in the airline industry, Big Data is analyzed for identifying personal preferences or spotting which passengers like to have window seats for short-haul flights and aisle seats for long-haul flights. This helps airlines to offer the similar seats to customers when they make a fresh booking with the airways.

Big Data also helps air lines to track customers who regularly fly between specific routes so that they can make the right cross-sell and up-sell offers. Some airlines also apply analytics to pricing, inventory, and advertising for improving customer experiences, leading to more customer satisfaction, and hence, more business. Some airlines even go to the length of evaluating customers who tend to miss their flights. They try to help such customers by delaying the flights or booking them on another flight.

iv) Government

Big Data has come to play an important role in almost all the undertaking and processes of government.

- Taking timely and informed decisions about various issues.
- Identifying flaws and loopholes in processes and taking preventive or corrective measures on time.
- Assessing the areas of improvement in various sectors, such as education, health, defense, and research.
- Using budgets more judiciously and reducing unnecessary wastage and costs.
- Preventing fraudulent practices in various sectors.

v) Healthcare

In healthcare, the pharmacy and medical device companies use Big Data to improve their research and development practices, while health insurance companies use it to determine patient-specific treatment therapy modes that promise the best results. Big Data also helps researchers to work towards eliminating healthcare-related challenges before they become real problems. Big Data helps doctors to analyze the requirement and medical history of every patient and provide individualistic services to them, depending on their medical condition.

vi) Telecom

The mobile revolution and the Internet usage on mobile phones have led to a tremendous increase in the amount of data generated in the telecom sector. Managing this huge pool of data has almost become a challenge for the telecom industry. For example, in Europe, there is a compulsion on the telecom companies to keep data of their customers for at least six months and maximum up to two years. Now, all this collection, storage, and maintenance of data would just be a waste of time and resources unless we could derive any significant benefits from this data. Big Data analytics allows telecom industries to utilize this data for extracting meaningful information that could be used to gain crucial business insights that help industries in enhancing their performance, improving customer services, maintaining their hold on the market, and generating more business opportunities.

Q12. Explain how data can be collected for research ?

Ans :

Primary Research (Surveys, Experiences, Abservations),

Secondary Research

(Competitive and Market place data, Industry reports, Consumer data, Business data), Location data (Mobile device data, Geospatial data), Image data (Video, Satellite image, Surveillance), Supply Chain data (vendor Catalogs, Pricing etc), Device data (Sensor data, RF device, Telemetry)

Structured Data

They have predefined data model and fit into relational database. Especially, operational data is ¯structured as it is put into a database based on the type of data (i.e., character, numeric, floating point, etc.)

Semi-structured data

These are data that do not fit into a formal structure of data models. Semi-structured data is often a combination of different types of data that has some pattern or structure that is not as strictly defined as structured data. Semi-structured data contain tags that separate semantic elements which includes the capability to enforce hierarchies within the data.

Unstructured data

Do not have a predefined data model and / or do not fit into a relational database. Oftentimes, text, audio, video, image, geospatial, and Internet data (including click streams and log files) are considered unstructured data.

Q13. Explain the relationship of big data with other areas?

Ans : **(Imp.)**

Big data models Improve Operational Efficiencies Increase Revenues Achieve Competitive Differentiation Reduce risks and costs Sell to microtrends Offer new services Save time Enable self service Seize market share Lower complexity Improve customer experience Incubate new ventures Enable self service Detect fraud

i) Digital Marketing.

ii) Financial Services

iii) Big data and Advances in health care

iv) Advertising and Big Data

I) Digital Marketing

Introduction Database Marketers, Pioneers of Big Data Big Data & New School of Marketing Cross Channel Life cycle Marketing Social and Affiliate Marketing Empowering marketing with Social Intelligence Introduction Digital marketing encompasses using any sort of online media (profit or non-profit) for driving people to a website, a mobile app etc. and retaining them and interacting with them to understand what consumers really want.

Digital \marketing is easy when consumers interact with corporate' primary platform (i.e., The one the corporate own) because corporate get good information about them. But corporate get very little information once people start interacting with other platforms (e.g.,. Face book, Twitter, Google +). One of the problems that have to be dealt with in a digital existence, is that corporate do not have access to data for all consumers (i.e.,. There is very little visibility about people when they interact in social networking sites). So corporate lose control of their ability to access the data they need, in order to make smart, timely decisions. Big data on the web will completely transform a company's ability to understand the effectiveness of its marketing and the ability to understand how its competitors are behaving. Rather than a few people making big decisions, the organizations will be able to make hundreds and thousands of smart decisions every day

II) Financial Services

Fraud & Big Data - Fraud is intentional deception made for personal gain or to damage another individual. - One of the most common forms of fraudulent activity is credit card fraud. - Social media and mobile phones are forming new frontiers fraud. - Capegemini financial services team believes that due to the nature of data streams and processing required BIG Data Technologies provide an optimal technology solution based on the following three Vs : 1. High volume: Years of consumer records and transactions (150 billion + 2. records per year). 3. High velocity: Dynamic transactions and social media info. 4. High variety: Social media plus other unstructured data such as customer E-mails, call center conversations as well as transactional structured data.

III) Big data and Healthcare

Big data promises enormous revolution in healthcare, with the advancements in everything from the management of chronic disease to the delivery of personalized medicine. In addition to saving and improving lives, Big Data has the potential to transform the entire healthcare system by replacing guesswork and institution with objective data-driven science.-The healthcare industry now has huge amount of data: from biological data such as gene expression, Special Needs Plans (SNPs), proteomics, metabolomics, and next-generation gene sequence data etc. The exponential growth in data is further accelerated by the digitization of patient level data stored in Electronic Medical Records (EMRs) or Electronic Health Records (EHRs) and Health Information Exchanges (HIEs) enhanced with data from imaging and test

results, medical and prescription claims and personal health devices. - In addition to saving and improving lives, Big Data has the potential to transform the entire health care system by replacing guesswork and intuition with objective, data-driven science

IV) Advertising and Big Data

Big Data is changing the way advertisers address three related needs. (i) How much to spend on advertisements. (ii) How to allocate amount across all the marketing communication touch points. (iii) How to optimize advertising effectiveness. Given these needs advertisers need to measure their advertising end to end in terms of Reach, Response & Reaction. Reach, Resonance, and Reaction Reach: First part of reach is to identify the people who are most volumetrically responsive to their advertising and then answer questions such as what do those people watch? What do they do online? How to develop media plan against intended audience. The second part of reach is delivering advertisements to the right audience. That is, to understand if we are actually reaching our desired audience. If we think about the online world, it's a world where we can deliver 100 million impressions but we never really know for sure who our campaign was actually delivered to. If our intended audience is women aged 18 to 35, of our 100 million impressions, what percentage of impressions were actually delivered to the intended audience? What was the reach, what was the frequency, what was the delivery against the intended audience? Resonance: If we know whom we want to reach and we're reaching them efficiently with your media spend, the next question is, are our ads breaking through? Do people know they're from our brand? Are they changing attitudes? Are they making consumers more likely to want to buy our brand? This is what is called resonance. Reaction: Advertising must drive a behavioral reaction or it isn't really working. We have to measure the actual behavioral impact.

1.2 Introducing Technologies for Handling Big Data

1.2.1 Distributed and Parallel Computing for Big Data

Q14. Describe Distributed and Parallel Computing for Big Data.

Ans : **(Imp.)**

In distributed computing, multiple computing resources are connected in a network and computing "tasks are distributed across these resources. This sharing of tasks increases the speed as well as the efficiency of the system. Because of reason, the distributed computing is considered faster and much more efficient than traditional methods of computing. It is also more suitable to process huge amounts of data in a limited time.

Another way to improve the processing capability of a computer system is to add additional computational resources to it. This will help in dividing complex computations into subtasks, which can be handled individually by processing units that are running in parallel. We call such systems as parallel systems in which multiple parallel computing resources are involved to carry out calculations simultaneously. The concept behind involving multiple parallel resources is that the processing capability will increase with the increase in the level of parallelism.

Organizations use both parallel and distributed computing techniques to process Big Data. The most important constraint for businesses today is time. In case there was no restriction on time, every organization would hire outside (or third-party) sources to perform the analysis of its complex data. The direct benefit of adopting this method is that the organization would not require any resources and data sources to process and analyze complex data. These third parties are usually specialized agencies in the field of data manipulation, processing, and analysis. Apart from being effective, hiring third party agencies also reduces the storage and processing costs of handling large amounts of data.

Figure shows a comparison between distributed and parallel processing techniques:

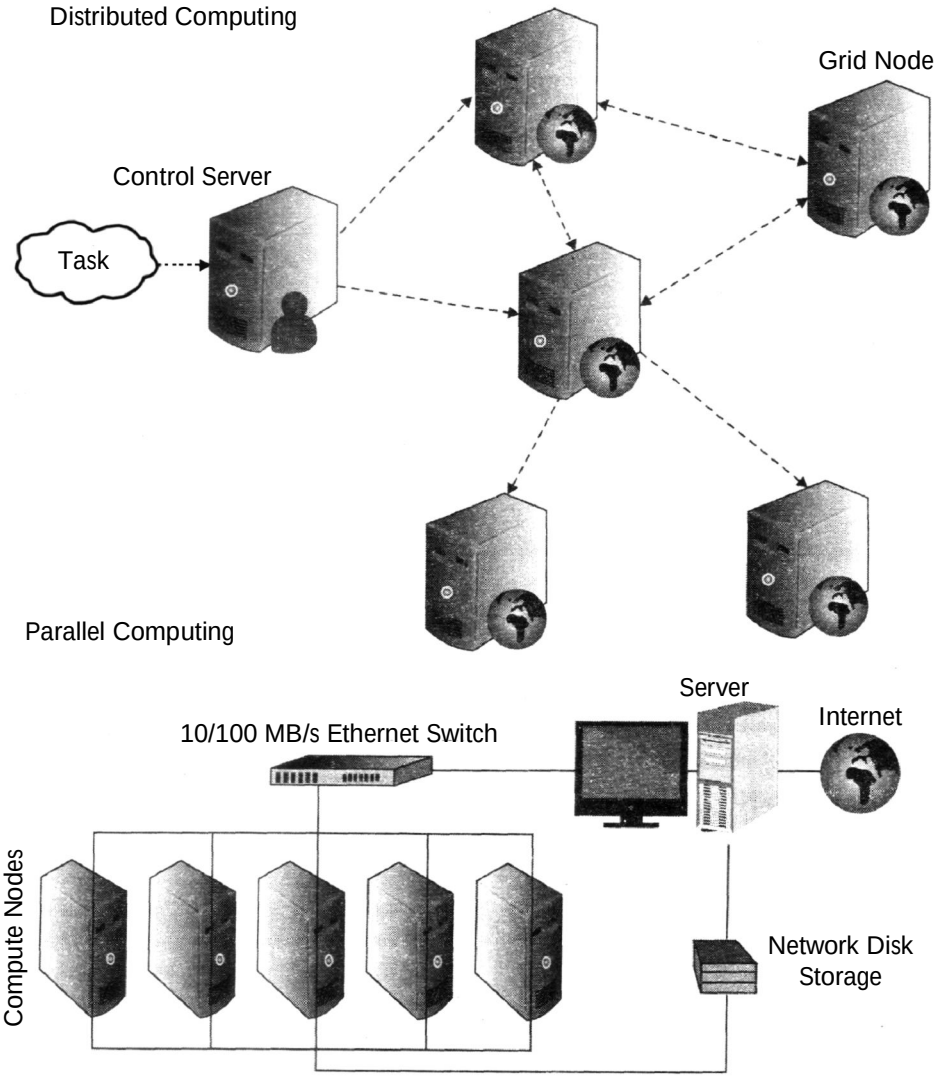


Fig.: Distributed Computing and Parallel Computing

Moreover, data processing and analysis activities carried out within an organization would require the organization to capture and analyze only a sample dataset rather than the entire data thus, reducing the processing load significantly.

The growing competition in the market and the astronomical increase in the volume, velocity, variety, and veracity of data collected from different sources, at the same time, are forcing organizations to adopt a data analysis strategy that can be used for analysing the entire data available with the organization in a very short time. This is done with the help of powerful hardware components and new software programs and/or applications written specifically to fulfill the need of advanced technological developments. The following procedure is followed by these software applications:

1. Breaking up the given task into smaller components
2. Surveying the available resources at hand
3. Assigning the subtasks to the nodes or computers that are interconnected in a network.

Sometimes, resources develop some technical problems and fail to respond. Such situations can be handled by virtualization, where some processing and analytical tasks are delegated to other resources. Another problem that often hampers data storage and processing activities is latency.

Latency can be defined as the aggregate delay in the system because of delays in the completion of individual tasks. Such a delay automatically leads to the slowdown in system performance as a whole and is often termed as system delay. It also affects data management and communication within and across various business units there by, affecting the productivity and profitability of an organization. Implementing distributed and parallel computing methodologies helps in handling both latency and data-related problems.

Figure shows elaborates the processing of a large dataset in a distributed computing environment:

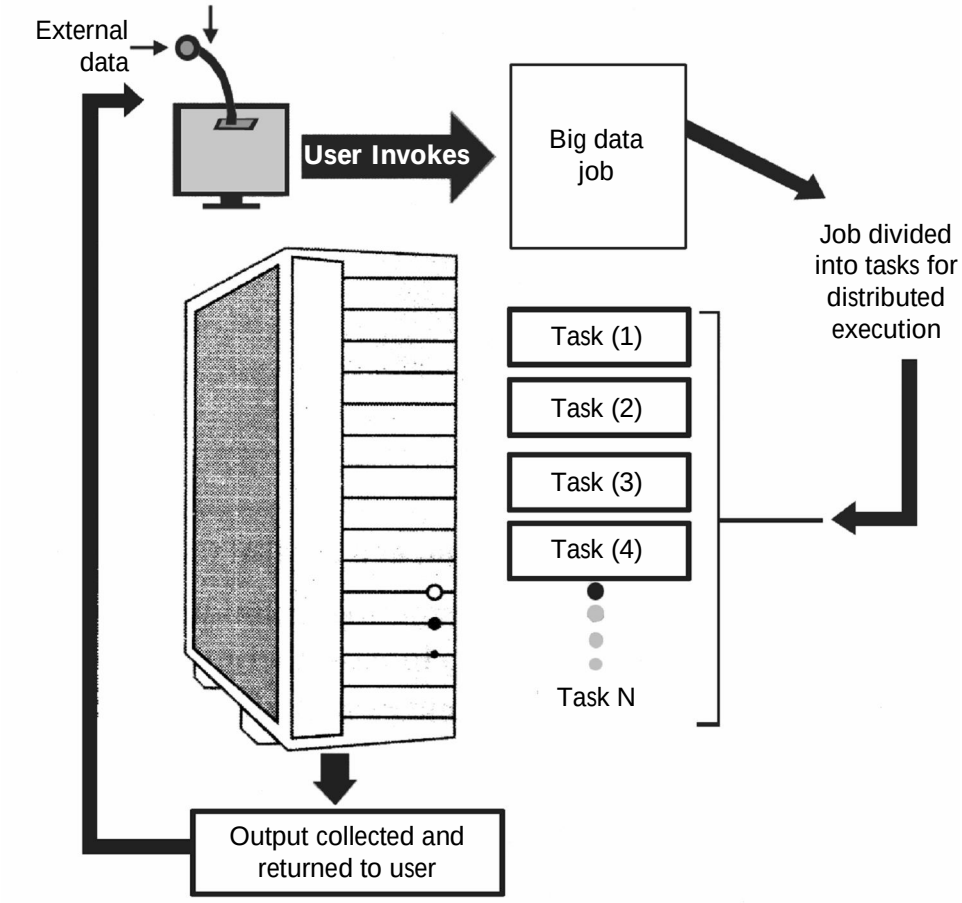


Fig.: Distributed Computing Technique for Processing Large Data

As you can notice in Figure, the nodes are arranged within a system along with the elements that form the core of computing resources. These resources include CPU, memory, disks, etc. Big Data systems, usually, have higher scaling requirements. Therefore, these nodes are more beneficial for adding scalability to the Big Data environment, as and when required. The system, with added scalability, can accommodate the growing amounts of data more efficiently and flexibly. Distributed computing technique also makes use of virtualization and load balancing features. The sharing of workload across various systems throughout the network to manage the load is known as load balancing. The virtualization feature creates a virtual environment in which hardware platform, storage device, and Operating System (OS) are included.

Q15. Discuss the techniques of parallel computing.

Ans : **(Imp.)**

Parallel computing technology uses a number of techniques to process and manage huge amounts of data produced at a high velocity. Some of these techniques.

i) Cluster or Grid Computing (Primarily used in Hadoop)

Cluster or grid computing is based on a connection of multiple servers in a network. This network is known as a cluster in which the servers share the workload among them. A cluster can be either homogeneous (comprising the same type of commodity hardware) or heterogeneous (consisting of different types of hardware).

Uses

A cluster can be created even by using hardware components that were acquired a long time back to provide cost-effective storage options. The overall cost may be very high in cluster computing.

ii) Massively Parallel Processing (MPP) (Used in data warehouses)

A single machine working as a grid is used in the MPP platform, which is capable of handling the activities of storage, memory, and computing. Software written specifically for MPP platform is used for the optimization of MPP capabilities.

Uses

MPP platforms, such as EMC Greenplum, and ParAccel are most suited for high-value use cases.

iii) High-Performance Computing (HPC)

HPC environments are known to offer high performance and scalability by using IMC. This technology is especially suitable for processing floating-point data at high speeds.

Uses

HPC environments can be used to develop specialty and custom applications for research and business organizations where the result is more valuable, than the cost or where strategic importance of the project is of high priority'.

Q16. List out the differences between distributing and parallel system.

Ans :

Sl. No.	Distributed System	Parallel System
1.	An independent, autonomous system connected in a network for accomplishing specific tasks	A computer system with several processing units attached to it
2.	Coordination is possible between connected computers that have their own memory and CPU processing unit in a network	A common shared memory can be directly accessed by every
3.	Loose coupling of computers connected in a network, providing access to data and remotely located resources	Tight coupling of processing resources that are used for solving a single, complex problem

1.3 Cloud Computing and Big Data

Q17. Define Cloud Computing and Mention the advantages and disadvantages of cloud computing.

Ans : **(Imp.)**

Cloud computing is a general term for the delivery of hosted services over the internet. Cloud computing enables companies to consume a compute resource, such as a virtual machine (VM), storage or an application, as a utility - just like electricity - rather than having to build and maintain computing infrastructures in house.

Advantages

- Lower upfront costs and reduced infrastruc-ture costs.
- Easy to grow your applications.
- Scale up or down at short notice.
- Only pay for what you use.
- Everything managed under SLAs.
- Overall environmental benefit (lower carbon emissions) of many users efficiently sharing large systems. (But see the box below.)

Disadvantages

- Higher ongoing operating costs. Could cloud systems work out more expensive?
- Greater dependency on service providers. Can you get problems resolved quickly, even with SLAs?
- Risk of being locked into proprietary or vendor-recommended systems? How easily can you migrate to another system or service provider if you need to?
- What happens if your supplier suddenly decides to stop supporting a product or system you've come to depend on?
- Potential privacy and security risks of putting valuable data on someone else's system in an unknown location?
- If lots of people migrate to the cloud, where they're no longer free to develop neat and whizzy new things.
- Dependency on a reliable Internet connection.

Q18. Explain about the layers of computing.

Ans : **(Imp.)**

Three Layers of Computing

Different categories of computing facilities. These computing facilities can be segmented into three categories:

1. Infrastructure
2. Platform
3. Application

These three categories of computing facilities form three layers in the basic architecture of computing. Figure represents the relationships between these three entities.

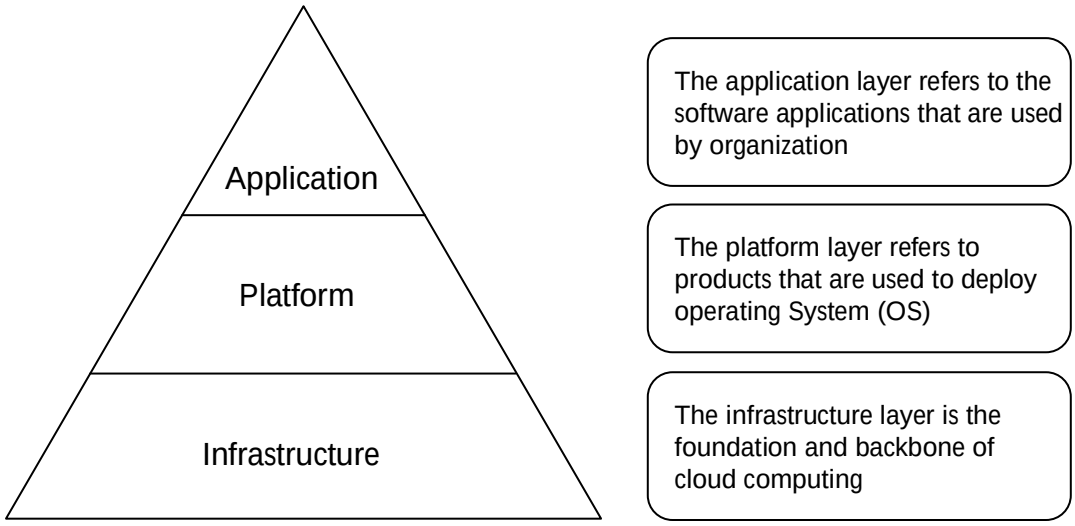


Fig.: Relationship between entities

1. **Infrastructure:** The bottom layer or the foundation is the 'computing infrastructure' facility. This includes all physical computing devices or hardware components like the processor, memory, network, storage devices and other hardware appliances. Infrastructure refers to computing resources in their bare-metal form. This layer needs basic amenities like electric supply, cooling system etc.

2. **Platform:** In computing, platform consists of the physical computing device (hardware) loaded with layer(s) of software where the program or application can run. The term 'computing platform' refers to different abstract levels. It consists of:

   - Certain hardware components, only.
   - Hardware loaded with an operating system (OS).
   - Hardware and OS, additionally, loaded with run-time libraries.

   Hardware alone can be considered as the platform in case of embedded systems, where physical resources can be accessed without any operating system.

   A fully configured physical computer loaded with an operating system is considered as a platform for computing. Different platforms can be installed over the same computing infrastructure. Linux or Windows operating systems installed over the same physical computer (computing infrastructure) can provide two different computing platforms.

   The platform layer is also the place where software developers work. Hence Integrated Development Environments (IDEs) and runtimes are part of this layer. Java Development Kit (JDK) or .NET are examples of popular computing frameworks. Software applications can be developed and run over these platforms.

3. **Application:** Applications (application software) constitute the topmost layer of this layered architecture. This layer generally provides interfaces for interaction with external systems (human or machine) and is accessed by end users of computing. A user actually works on the application layer to edit a document, play a game or use the calculator in a computer. At this layer, organizations access enterprise applications using application interfaces to run their business.

Different types of people work at different layers of computing. They need to have different skill-sets and knowledge.

Three Layers in Traditional Computing

In the traditional approach, the above three layers acts like the following:

1. Traditional Infrastructure Layer

Traditionally enterprises arrange computing infrastructure themselves by forming an internal computing system management team or they depend upon some third-party vendors. These vendors or internal system management departments are responsible for managing the whole computing infrastructure of the enterprise, including planning, designing, assembling of physical servers, network or data storage infrastructure build ups and other services like load balancing of electricity supply.

The whole process of setting up of such an infrastructure is quite complex. For a new office, this process generally takes weeks and sometimes months.. Protection and security of the infrastructure appears as an extra burden for any organization.

2. Traditional Platform Layer

In the traditional model, the platform building and management tasks need special expertise. Enterprises either need to maintain an internal team or can outsource these jobs. *Computing platform* building activities involve installation of an operating system (OS), application runtimes or application development environments on computing infrastructure. For example, Java Runtime Environment is essential to run any java based application.

Traditionally the entire responsibility of platform installation, configuration, updates and other staffing at appropriate levels often fall on the shoulders of the people concerned with application development or even on users who are interested only in using applications.

3. Traditional Application Layer

At application layers, the users are the end users of computing. They need not have any knowledge about a complex computing system. Their only interests are to access or use different software applications to fulfill requirements of personal or business related works.

Software applications provide solutions for various needs. In traditional model, computing infrastructure or computing platform or both often become concerns of application subscribers, which is a critical scenario.

Q19. Write about the challenges of cloud computing.

Ans :

1. Data Security and Privacy

Data security is a major concern when switching to cloud computing. User or organizational data stored in the cloud is critical and private. Even if the cloud service provider assures data integrity, it is your responsibility to carry out user authentication and authorization, identity management, data encryption, and access control. Security issues on the cloud include identity theft, data breaches, malware infections, and a lot more which eventually decrease the trust amongst the users of your applications.

2. Cost Management

When there is under optimization of the resources, let's say that the servers are not being used to their full potential, add up to the hidden costs. If there is a degraded application performance or sudden spikes or overages in the usage, it adds up to the overall cost. Unused resources are one of the other main reasons why the costs go up. If you turn on the services or an instance of cloud and forget to turn it off during the weekend or when there is no current use of it, it will increase the cost without even using the resources.

3. Multi-Cloud Environments

Due to an increase in the options available to the companies, enterprises not only use a single cloud but depend on multiple cloud service providers. Most of these companies use hybrid cloud tactics and close to 84% are dependent on multiple clouds. This often ends up being hindered and difficult to manage for the infrastructure team. The process most of the time ends up being highly complex for the IT team due to the differences between multiple cloud providers.

4. Performance Challenges

Performance is an important factor while considering cloud-based solutions. If the performance of the cloud is not satisfactory, it can drive away users and decrease profits. Even a little latency while loading an app or a web page can result in a huge drop in the percentage of users. This latency can be a product of inefficient load balancing, which means that the server cannot efficiently split the incoming traffic so as to provide the best user experience. Challenges also arise in the case of fault tolerance, which means the operations continue as required even when one or more of the components fail.

5. Interoperability and Flexibility

When an organization uses a specific cloud service provider and wants to switch to another cloud-based solution, it often turns up to be a tedious procedure since applications written for one cloud with the application stack are required to be re-written for the other cloud. There is a lack of flexibility from switching from one cloud to another due to the complexities involved. Handling data movement, setting up the security from scratch and network also add up to the issues encountered when changing cloud solutions, thereby reducing flexibility.

6. High Dependence on Network

Since cloud computing deals with provi-sioning resources in real-time, it deals with enormous amounts of data transfer to and from the servers. This is only made possible due to the availability of the high-speed network. Although these data and resources are exchanged over the network, this can prove to be highly vulnerable in case of limited bandwidth or cases when there is a sudden outage. It is therefore a major challenge for smaller enterprises that have to maintain network bandwidth that comes with a high cost.

7. Lack of Knowledge and Expertise

Due to the complex nature and the high demand for research working with the cloud often ends up being a highly tedious task. It requires immense knowledge and wide expertise on the subject. Although there are a lot of professionals in the field they need to constantly update themselves. Cloud computing is a highly paid job due to the extensive gap between demand and supply. Therefore, there is a need for upskilling so these professionals can actively understand, manage and develop cloud-based applications with minimum issues and maximum reliability.

Q20. Discuss about cloud computing model.

Ans : **(Imp.)**

One of the vital issues that organizations face with the storage and management of Big Data is the nuge amount of investment to get the required hardware setup and software packages. Some of these resources may be over utilized or underutilized with the varying requirements overtime. We can overcome these challenges by providing a set of computing resources that can be shared through cloud computing. These shared resources comprise applications, storage solutions, computational units, networking solutions, development and deployment platforms, business processes, etc. The cloud computing environment saves costs related to infrastructure in an organization by providing a framework that can be optimized and expanded horizontally. In order to operate in the real world, cloud implementation requires common standardized processes and their automation.

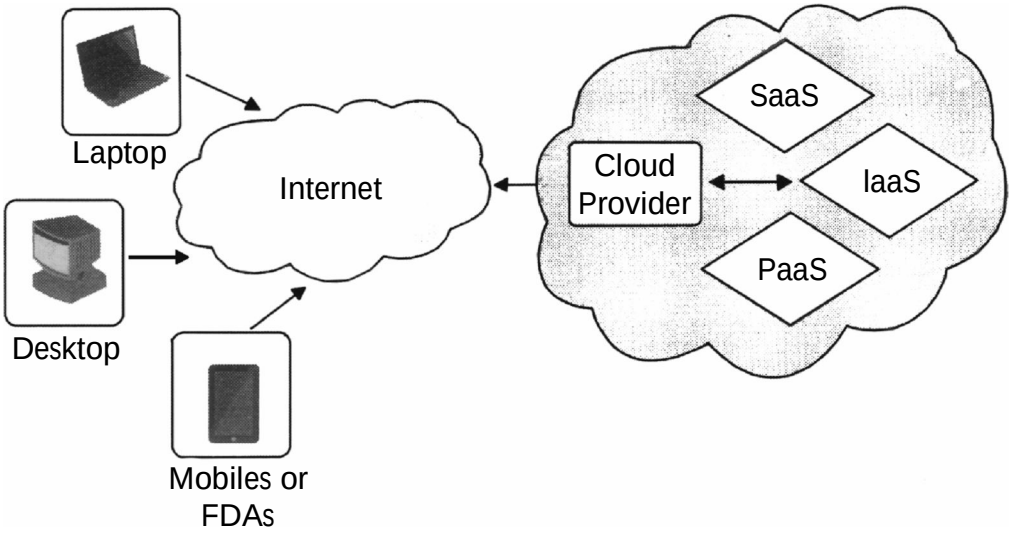


Fig.: Cloud Computing Model

In cloud-based platforms, applications can easily obtain the resources to perform computing tasks. The costs of acquiring these resources need to be paid as per the acquired resources and their use. In cloud computing, this feature of resource acquisition is in accordance with the requirements and payment of cost and is known as elasticity. Cloud computing makes it possible for organizations to dynamically regulate the use of computing resources and access them as per the need while paying only for those resources that are used. This facility of dynamic use of resources provides flexibility; however, an organization needs to plan, monitor, and control its resource utilization carefully. Careless resource monitoring and control can result in unexpectedly high costs.

The cloud computing technique uses data centers to collect data and ensures that data backup and recovery are automatically performed to cater to the requirements of the business community. Both cloud computing and Big Data analytics use the distributed computing model in a similar manner and hence, are complementary to each other.

1.3.1 Features of Cloud Computing

Q21. Discuss the features of cloud computing that can be used handle big data.

Ans : **(Imp.)**

The following are some features of cloud computing that can be used to handle Big Data:

i) Scalability

Scalability means addition of new resources to an existing infrastructure. The increase in the amount of data being collected and analyzed requires organizations to improve their hardware components' processing ability. These organizations may, at times, need to replace the existing hardware with a new set of hardware components in order to improve data management and processing activities. The new hardware may not provide complete support to the software that used to run properly on the earlier set of hardware. We can solve such issues by using cloud services that employ the distributed computing technique to provide scalability' to the architecture.

ii) Elasticity

Elasticity in cloud means hiring certain resources, as and when required, and paying ' for the resources that have been used. No extra payment is required for acquiring specific cloud services. For example, a business expecting the use of more data during in-store promotion could hire more resources to provide high processing power. Moreover, a cloud does not require customers to declare their resource requirements in advance.

iii) Resource Pooling

Resource pooling is an important aspect of cloud services for Big Data analytics. In resource pooling, multiple organizations, which use similar kinds of resources to carry out computing practices, have no need to individually hire all the resources. The sharing of resources is allowed in a cloud, which facilitates cost cutting through resource pooling.

iv) Self Service

Cloud computing involves a simple user interface that helps customers to directly access the cloud services they want. The process of selecting the needed services requires no intervention from human beings and can be accessed automatically.

v) Low Cost

A careful planning, use, management, and control of resources help organizations to reduce the cost of acquiring hardware significantly. Also, cloud offers customized solutions, especially to organizations that cannot afford too much initial investment in purchasing the resources that are used for computation in Big Data analytics. The cloud provides them the pay-: as-you-use option in which organizations need to sign for those resources only that are essential. This also helps the cloud provider in harnessing benefits of economies of scale and providing benefit to their customers in terms of cost reduction.

vi) Fault Tolerance

Cloud computing provides fault tolerance by offering uninterrupted services to customers, especially in cases of component failure. The responsibility of handling the workload is shifted to other components of the cloud.

Q22. Explain about, how cloud computing addresses business challenges.

Ans :

1. Importance of data and where it is stored (GDPR)

Your business should have a clear concept of the value (and sensitive nature) of the data that is critical for operations. The inherent need to undertake efforts to assess risks and costs involved with current data storage practices is real (GDPR). Especially in an international business organization, deciding where to house data is a complex question that is largely determined by how that data will be utilized.

2. Hosting

A hybrid cloud portfolio can support locally hosted options in either the UK or elsewhere in the EU, and cost-effective cloud options will help mitigate the risks associated with long-term investments or expensive migrations.

3. Security

Cloud technology has advanced greatly and now it is actually more secure and reliable than traditional on premise solutions. Many business owners who are accustomed to using local servers hesitate to transition to the cloud for fear of security risks. They worry that having their information "out there" on the cloud will make it more susceptible to hackers.

4. Vulnerability to disasters

If you're only storing your data on local servers, you may be more susceptible to having your data affected by a natural disaster. Certain precautions may help alleviate this risk - such as backing up data, for example - but utilizing the cloud can provide even greater protection.

5. Benefit for disaster recovery

Hosting systems and storing documents on the cloud provides a smart safeguard in case of an emergency. Man-made and natural disasters can damage equipment, shut off power and incapacitate critical IT functions. Supporting disaster recovery efforts is one of the important advantages of cloud computing for companies.

These improvements in security can also come with an attractive reduction in cost.

6. Increased long-term costs

Not moving to the cloud could cost your company money in the long run. While you do need to pay for equipment with the cloud, costs are often more flexible because you can pay as you go depending on how much storage space you need, 'On Demand'.

7. Boosts cost efficiency

Cloud computing reduces or eliminates the need for businesses to purchase equipment and build out and operate data centers. This presents a significant savings on hardware, facilities, utilities and other expenses required from traditional computing. Also, reducing the need for on-site servers, software and staff can trim the IT budget further.

8. Lack of flexibility

Businesses have historically been tethered to wherever their equipment is located, because that's where they need to access all their information. This becomes a problem, though, when employees need to work outside the office because it may limit or eliminate their ability to work from home, meet with clients out in the field, or network away from their workspace.

9. Promotes collaboration

If your employees are outside the office or your clients are not physically accessible, it can be difficult to work on the same task when everyone is limited to their own local workspace.

With the cloud, however, your business can use file-sharing applications to collaborate effectively, even if everyone is geographically separated. Clients, vendors, and employees can all work together in real time, making enhanced communication one of the best ways to combat the risks of not moving to the cloud.

10. Increases mobility

One of the advantages of cloud computing for businesses is how easily team members can work from anywhere. Businesses that operate on the cloud can provide staff with options to work on the go or at home, from their desktops, laptops, smart phones and tablets.

11. Frequent disturbances

Disasters aren't the only things putting your data at risk. Power outages, hardware problems, or general network issues can prevent you from getting your work done. Even disruptions like installing an update can cause downtime, which costs your business money. While these issues can affect the cloud as well as bare metal servers, a hybrid approach can help minimize these risks by backing up your data in multiple locations.

12. Limited technical support

Outside the cloud, your organization is limited to whoever is working inside your office. In the case of an emergency, you either have to hope your local professionals can get the job done or hire a third-party company to help, which could be costly.

Q23. Explain about various web services involved in cloud computing.

Ans : **(Imp.)**

Web Services in Cloud Computing

A web service is a set of open protocols and standards that allow data exchange between different applications or systems. Web services can be used by software programs written in different programming languages and on different platforms to exchange data through computer networks such as the Internet. In the same way, communication on a computer can be inter-processed.

Any software, application, or cloud technology that uses a standardized Web protocol (**HTTP or HTTPS**) to connect, interoperate, and exchange data messages over the Internet-usually XML (Extensible Markup Language) is considered a Web service Is.

Web services allow programs developed in different languages to be connected between a client and a server by exchanging data over a web service. A client invokes a web service by submitting an XML request, to which the service responds with an XML response.

Web Service Components

XML and HTTP is the most fundamental web service platform. All typical web services use the following components:

1. SOAP (Simple Object Access Protocol)

SOAP stands for "Simple Object Access Protocol". It is a transport-independent messaging protocol. SOAP is built on sending XML data in the form of SOAP messages. A document known as an XML document is attached to each message.

Only the structure of an XML document, not the content, follows a pattern. The great thing about web services and SOAP is that everything is sent through HTTP, the standard web protocol.

Every SOAP document requires a root element known as an element. In an XML document, the root element is the first element.

The "envelope" is divided into two halves. The header comes first, followed by the body. Routing data, or information that directs the XML document to which client it should be sent, is contained in the header. The real message will be in the body.

2. UDDI (Universal Description, Search, and Integration)

UDDI is a standard for specifying, publishing and searching online service providers. It provides a specification that helps in hosting the data through web services. UDDI provides a repository where WSDL files can be hosted so that a client application can search the WSDL file to learn about the various actions provided by the web service. As a result, the client application will have full access to UDDI, which acts as the database for all WSDL files.

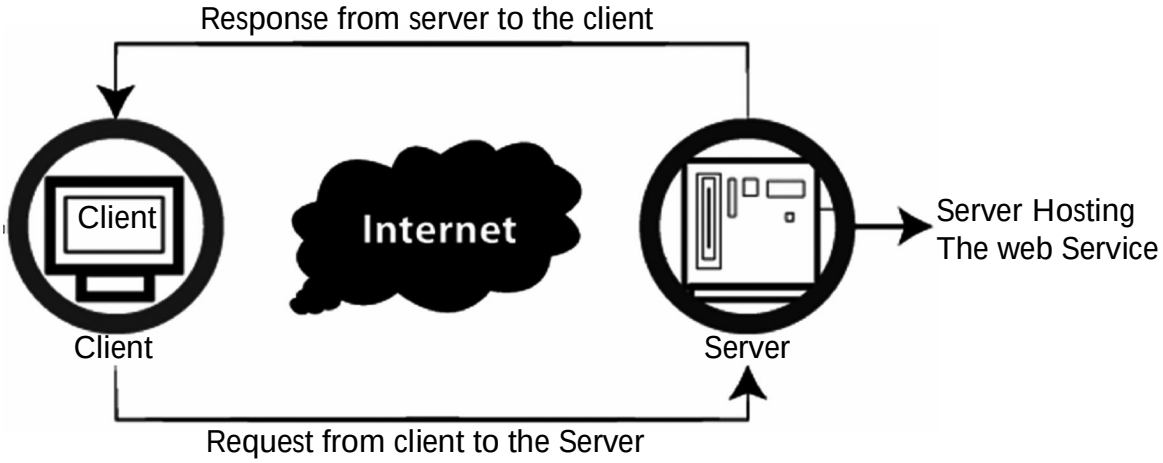
The UDDI Registry will keep the information needed for online services, such as a telephone directory containing the name, address, and phone number of a certain person so that client applications can find where it is.

3. WSDL (Web Services Description Language)

The client implementing the web service must be aware of the location of the web service. If a web service cannot be found, it cannot be used. Second, the client application must understand what the web service does to implement the correct web service. WSDL, or Web Service Description Language, is used to accomplish this. A WSDL file is another XML-based file that describes what a web service does with a client application. The client application will understand where the web service is located and how to access it using the WSDL document.

How does web service work?

The diagram shows a simplified version of how a web service would function. The client will use requests to send a sequence of web service calls to the server hosting the actual web service.



Remote procedure calls are used to perform these requests. The calls to the methods hosted by the respective web service are known as Remote Procedure Calls (RPC). Example: Flipkart provides a web service that displays the prices of items offered on Flipkart.com. The front end or presentation layer can be written in .NET or Java, but the web service can be communicated using a programming language.

The data exchanged between the client and the server, XML, is the most important part of web service design. XML (Extensible Markup Language) is a simple, intermediate language understood by various programming zrvice can be built in any programming language as the content is written in XML.

Features of Web Service

Web services have the following characteristics:

(a) XML-based: A web service's information representation and record transport layers employ XML. There is no need for networking, operating system, or platform bindings when using XML. At the mid-level, web offering-based applications are highly interactive.

(b) L oosely Coupled: The subscriber of an Internet service provider may not necessarily be directly connected to that service provider. The user interface for a web service provider may change over time without affecting the user's ability to interact with the service provider. A strongly coupled system means that the decisions of the mentor and the server are inextricably linked, indicating that if one interface changes, the other must be updated.

A loosely connected architecture makes software systems more manageable and easier to integrate between different structures.

(c) Ability to be synchronous or asynchronous: Synchronicity refers to the client's connection to the execution of the function. Asynchronous operations allow the client to initiate a task and continue with other tasks. The client is blocked, and the client must wait for the service to complete its operation before continuing in synchronous invocation.

Asynchronous clients get their results later, but synchronous clients get their effect immediately when the service is complete. The ability to enable loosely connected systems requires asynchronous capabilities.

(d) Coarse Grain: Object-oriented systems, such as Java, make their services available differently. At the corporate level, an operation is too great for a character technique to be useful. Building a Java application from the ground up requires the development of several granular strategies, which are then combined into a coarse grain provider that is consumed by the buyer or service.

Corporations should be coarse-grained, as should the interfaces they expose. Building web services is an easy way to define coarse-grained services that have access to substantial business enterprise logic.

(e) Supports remote procedural calls: Consumers can use XML-based protocols to call procedures, functions, and methods on remote objects that use web services. A web service must support the input and output framework of the remote system.

Enterprise-wide component development Over the years, JavaBeans (EJBs) and .NET components have become more prevalent in architectural and enterprise deployments. Several RPC techniques are used to both allocate and access them.

A web function can support RPC by providing its services, similar to a traditional role, or translating incoming invocations into an EJB or .NET component invocation.

(f) Supports document exchanges: One of the most attractive features of XML for communicating with data and complex entities.

1.3.2 Cloud Deployment Models

Q24. State various Cloud Deployment Models.

Ans :

Depending upon the architecture used in forming the network, services and applications used, and the target consumers, cloud services are offered in the form of various deployment models. The following are the most-commonly used cloud deployment models:

- Public Cloud Model
- Private Cloud Model
- Community Cloud Model
- Hybrid Cloud Model

Q25. Explain briefly about public cloud model.

(OR)

Discuss about End user cloud level.

Ans : **(Imp.)**

A cloud that is owned and managed by a company than the one (which can be either an individual "user or a company) using it is known as a public cloud. In this cloud, there is no need for the organizations (customers) to control or manage the resources; instead, they are being administered by a third party. Some examples of public cloud providers are Sawis, Verizon, Amazon Web Services, and Rackspace. You should understand that in case of a public cloud, the resources are owned or hosted by the cloud service providers (a company), and the services are sold to other companies. Companies or individuals can obtain various services in a public cloud. The workload is categorized on the basis of service category, and therefore, in this cloud, the hardware customization is possible to provide optimized performance. The process of computing becomes very flexible and scalable through customized hardware resources. For example, a cloud can be used specifically for video storage that can be streamed live on YouTube or Vimeo. You can also optimize this cloud for handling large traffic volumes.

Businesses can obtain economical cloud storage solutions in a public cloud, which provides efficient mechanisms for complex data handling. The primary concerns with a public cloud include security and latency, which can be overlooked citing the benefits of this cloud.

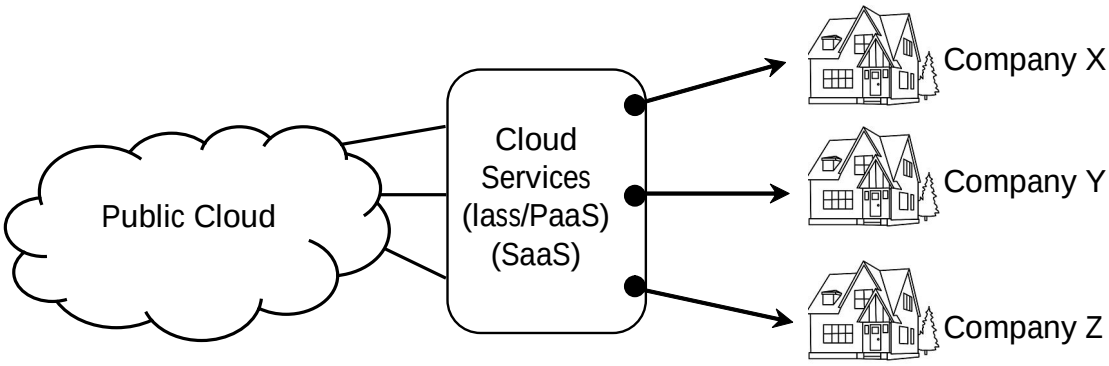


Fig.: Level of Accessibility in a Public Cloud

Q26. Explain briefly about :

(i) Private Cloud

(ii) Community Cloud

(iii) Hybrid Cloud

Ans :

i) Private Cloud (Enterprise Level Cloud)

The cloud that remains entirely in the ownership of the organization using it is known as a private cloud. In other words, in this cloud, the cloud computing infrastructure is solely designed for a single organization and cannot be accessed by other organizations. However, the organization may allow this cloud to be used by its employees, partners, and customers. The primary feature of a private cloud is that an organization installs the cloud for its own requirements. These requirements are customary to the organization that plans and manages the resources and their use.

A private cloud integrates all the processes, systems, rules, policies, compliance checks, etc. of the organization at a place. In a private cloud, you can automate several processes and operations that require manual handling in a public cloud. Moreover, you can also provide firewall protection to the cloud; thereby, solving many latency and security concerns. A private cloud can be either on-premises or hosted externally. In case of on-premises private clouds, the service is exclusively used and hosted by a single organization.

However, the private clouds that are hosted externally are used by a single organization and are not shared with other organizations. Moreover, the cloud services are hosted by a third party that specializes in cloud infrastructure. Note that on-premises private clouds are costlier as compared to the externally hosted private clouds. In case of a private cloud, security is kept in mind at every level of design. The general objective of a private cloud is not to sell the cloud services (IaaS/PaaS/SaaS) to the external organizations but to get the advantages of cloud architecture by not providing the privilege to manage your own data center.

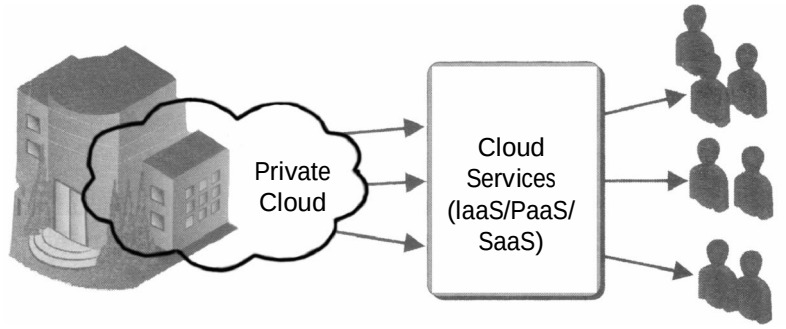
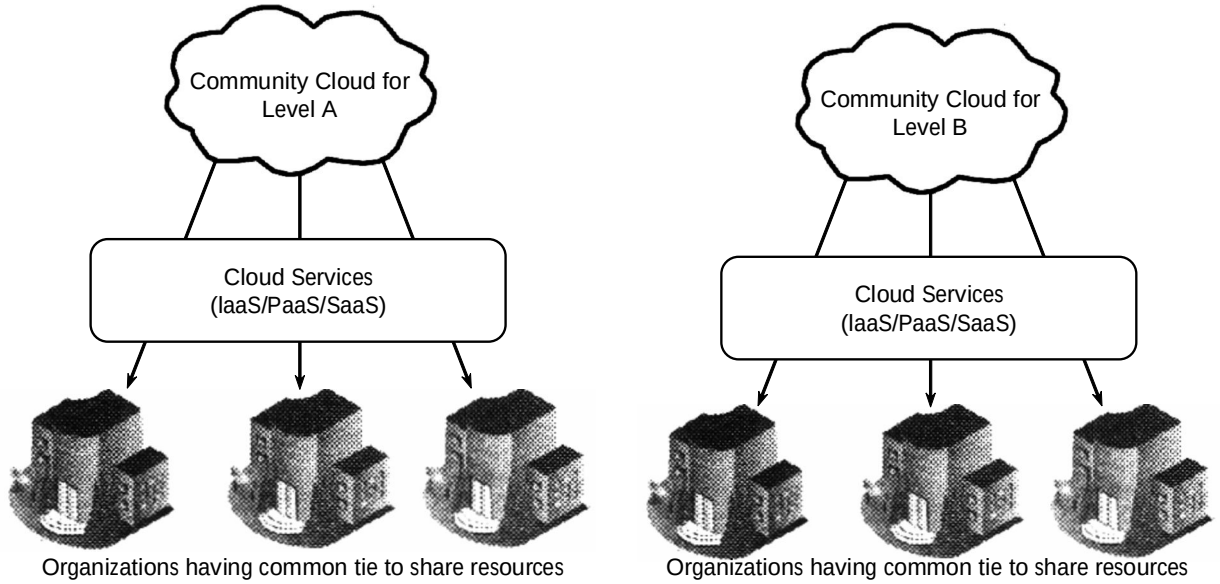


Fig.: Level of Accessibility in a Private Cloud

ii) Community Cloud

Community cloud is a type of cloud that is shared among various organizations with a common tie. This type of cloud is generally managed by a third party offering the cloud service and can be made available on or off premises. To make the concept of community cloud clear and to explain when community clouds can be designed, let's take an example. In any state or country, say England, the community cloud can be provided so that almost all government organizations of that state can share the resources available on the cloud. Because of the sharing of cloud resources on community cloud, the data of all citizens of that state can be easily managed by the government organizations.

Figure shows the use of community clouds:



iii) Hybrid Cloud

The cloud environment in which various internal or extemal' service providers offer services to many organizations is known as a hybrid cloud. Generally, it is observed that an organization hosts applications, which require high level of security and are critical, on the private cloud. It is also possible that the applications that are not so important or confidential can be hosted on the public cloud. In hybrid clouds, an organization can use both types of cloud, i.e. public and private together. Such type of cloud is generally used in situations such as cloud bursting. In case of cloud bursting,

an organization generally uses its own computing infrastructure; however, in high load requirements, the organization can access clouds. In other words, the organization using the hybrid cloud can manage an internal private cloud for general use and migrate the entire or a part of an application to the public cloud during the peak periods.

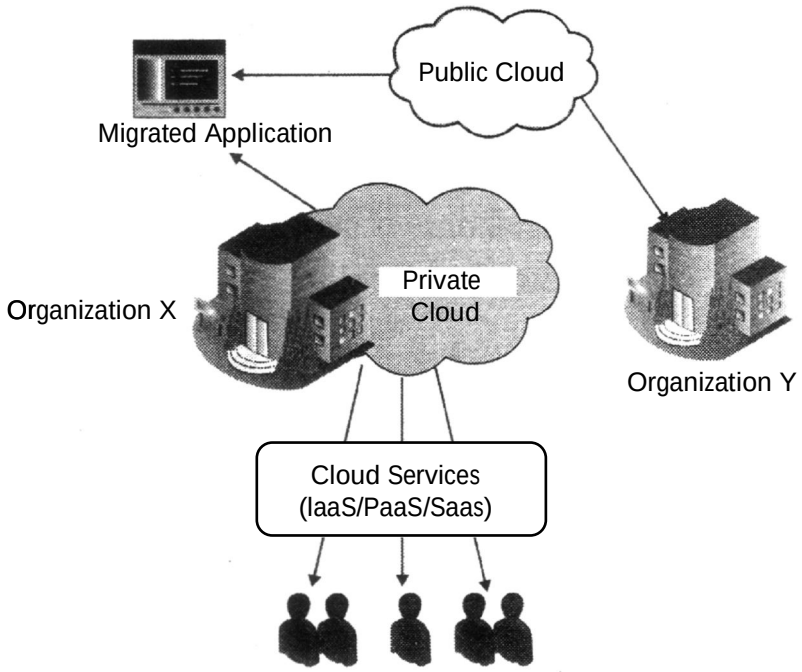


Fig.: Implementation of a Hybrid Cloud

1.3.3 Cloud Services for Big Data

Q27. Discuss the role cloud services play in handling big data.

Ans : **(Imp.)**

Cloud services are associated with various models that are used for delivery and deployment. Cloud follows the same architecture as Big Data, both requiring distributed clusters of computing devices. Big Data systems include different specifications of cloud as an integral part; therefore, a cloud is termed as an ideal computing environment for handling Big Data.

In Big Data, the IaaS, PaaS, and SaaS clouds are used in the following manner:

i) **IaaS:** The huge storage and computational power requirements for Big Data are fulfilled by the limitless storage space and computing ability obtained by the IaaS cloud.

ii) **PaaS:** PaaS offerings of various vendors have started adding various popular Big Data platforms that include MapReduce and Hadoop. These offerings save organizations from a lot of hassle, which may occur in managing individual hardware components and software applications.

iii) **SaaS:** Various organizations require identifying and analyzing the voice of customers, particularly on social media platforms. The social media data and the platform for analyzing the data are provided by SaaS vendors. In addition, private cloud facilitates the access to enterprise CRM data, which enables these analyses.

1.3.4 Cloud Providers in Big Data Market

Q28. Discuss the role cloud providers play in handling big data market.

Ans : **(Imp.)**

Big Data cloud providers have been gearing up to bring the most advanced technologies at competitive prices in the market. Some providers are established, whereas some of them are relatively new to the field of cloud services. Some of these providers are rendering services that are relevant to Big Data analytics only. Some such providers are discussed as follows:

1. Amazon

Amazon is one of the largest cloud service provider, and it offers its cloud services as Amazon Web Services (AWS).AWS includes some of the most popular cloud services, such as Elastic Compute Cloud (EC2), Elastic MapReduce, Simple Storage Service (S3), etc. Some of these services are discussed as follows:

i) EC2: Refers to a Web service that employs a large set of computing resources to perform its business operations. These resources are not properly utilized by Amazon, and therefore, they are pooled in the form of an IaaS cloud so that other organizations can take the benefit of these resources, ultimately benefitting Amazon through other rental cost. Organizations can use these resources elastically in a way that the hiring of resources is possible on an hourly basis. ,

ii) Elastic MapReduce: It is a Web service that uses Amazon EC2 computation and Amazon S3 storage for storing and processing large amounts of data so that the cost of processing and storage is reduced significantly.

iii) DynamoDB: Refers to a NoSQL database system in which data storage is done on Solid State Devices (SSDs). DynamoDB allows data replication for high availability and durability.

iv) Amazon S3: Amazon Simple Storage Service (Amazon S3) is a Web interface that allows data storage over the Internet and makes web-scale computing-possible.

v) High Performance Computing (HPC): Refers to a network that is replete with high bandwidth, low latency, and high computational abilities, which are required for processing Big Data, especially for solving issues related to education and business domains.

vi) RedShift: Refers to a data warehouse service that is used to analyze data with the help of existing tools of business intelligence in an economical manner. You can scale Amazon RedShift for handling data up to a petabyte.

2. Google

The cloud services that are provided by Google for handling Big Data include the following:

i) Google Compute Engine is a computing environment, which is secure, flexible, and based on virtual machine.'

ii) Google BigQuery is a Desktop as a Service (DaaS), which is used for searching huge amounts of data at a faster pace on the basis of SQL-format queries.

iii) Google Prediction API, which is used for the identifying patterns in data, storing of patterns, and improving the patterns with successive utilizations.

3. Windows Azure

Microsoft offers a PaaS cloud that is based on Windows and SQL abstractions and consists of a set of development tools, virtual machine support, management and media services, and mobile device services. Windows Azure PaaS is easy-to-adopt for people who are well equipped with the operations of .NET, SQL Server, and Windows. In addition, the Windows Azure HD Insight option added to the PaaS cloud makes it possible for the cloud users to address the emerging requirements to for integrating Big Data into Windows Azure solutions. The platform used for building the Windows Azure PaaS is Horton works Data Platform (HDP) that, as stated by Microsoft, is fully compatible with Apache Hadoop. Moreover, Microsoft Excel and various other Business Intelligence (BI) tools can be connected to Windows Azure with support from HD Insight, which can be developed on the Windows Server also.

UNIT II

Understanding Hadoop Ecosystem: Introducing Hadoop, HDFS and MapReduce, Hadoop functions, Hadoop Ecosystem.

Hadoop Distributed File System- HDFS Architecture, Concept of Blocks in HDFS Architecture, Namenodes and Datanodes, Features of HDFS. MapReduce.

Introducing HBase - HBase Architecture, Regions, Storing Big Data with HBase, Combining HBase and HDFS, Features of HBase, Hive, Pig and Pig Latin, Sqoop, ZooKeeper, Flume, Oozie.

2.1 INTRODUCING HADOOP

Q1. Define Hadoop.

Ans :

Meaning

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop helps to execute large amount of processing where the user can connect together multiple commodity computers to a single-CPU, as a single functional distributed system and have the particular set of clustered machines that reads the dataset in parallel and provide intermediate, and after integration gets the desired output.

Hadoop runs code across a cluster of computers and performs the following tasks:

- Data are initially divided into files and directories. Files are divided into consistent sized blocks ranging from 128M and 64M.
- Then the files are distributed across various cluster nodes for further processing of data.
- Job tracker starts its scheduling programs on individual nodes.
- Once all the nodes are done with scheduling then the output is return back.

2.1.1 HDFS

Q2. What is HDFS ? Explain HDFS architecture with neat diagram.

Ans : **(Imp.)**

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

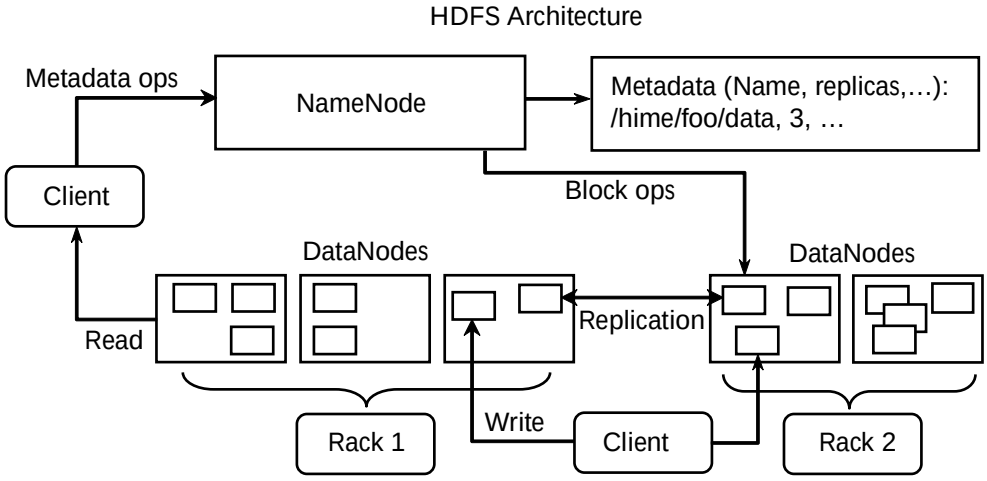
Features

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

Architecture

Apache HDFS or Hadoop Distributed File System is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a Master/Slave Architecture, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

i) NameNode



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

Functions of NameNode

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata :
- **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.

- **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.
- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the replication factor of all the blocks which we will discuss in detail later in this HDFS tutorial blog.
- In case of the DataNode failure, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

ii) DataNode

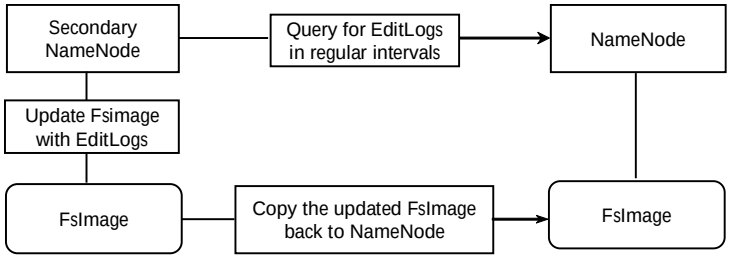
DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

Functions of DataNode

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

iii) Secondary NameNode

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



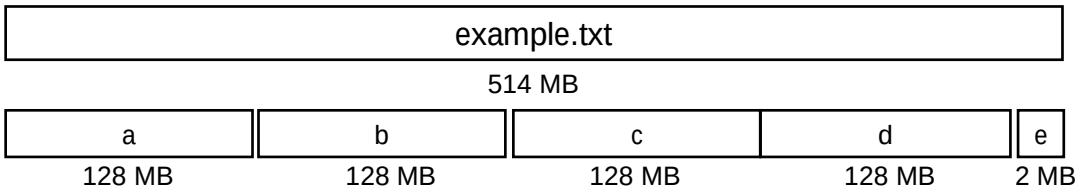
Functions

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

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

iv) Blocks

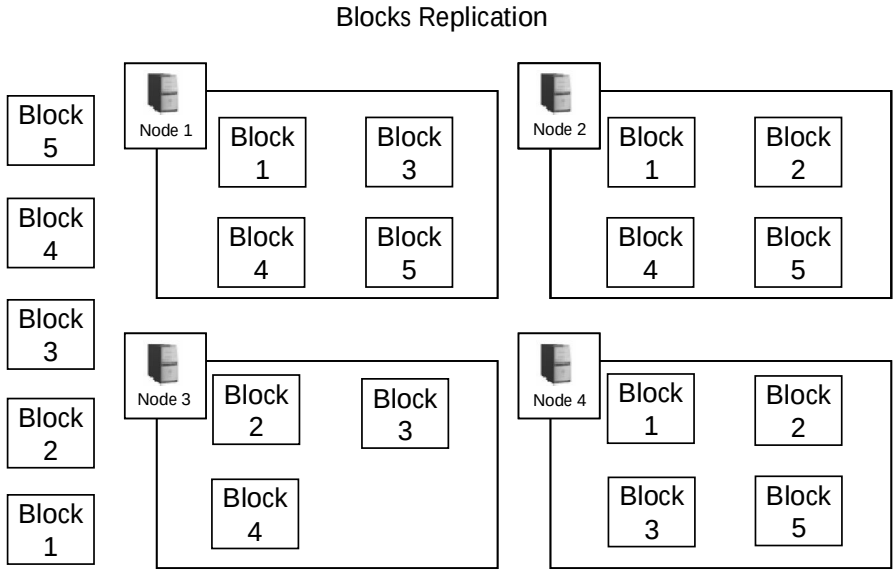
Blocks are the nothing but the smallest continuous location on your hard drive where data is stored. In general, in any of the File System, you store the data as a collection of blocks. Similarly, HDFS stores each file as blocks which are scattered throughout the Apache Hadoop cluster. The default size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



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

v) Replication Management

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



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

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

2.1.2 Mapreduce

Q3. Explain about Mapreduce programming model.

Ans : **(Imp.)**

Apache Hadoop is an open source framework for distributed batch processing of big data. Similarly, MapReduce is a parallel programming model suitable analysis of big data. MapReduce algorithms allow large-scale computations to be automatically parallelized across a large cluster of servers.

MapReduce Programming Model

MapReduce is a parallel data processing model for processing and analysis of massive scale data. MapReduce model has two phases: Map and Reduce. MapReduce programs are written in a functional programming style to create Map and Reduce functions. The input data to the map and reduce phases is in the form of key-value pairs. Run-time systems for MapReduce are typically large clusters built of commodity hardware. The MapReduce run-time systems take care of tasks such partitioning the data, scheduling of jobs and communication between nodes in the cluster. This makes it easier for programmers to analyze massive scale data without worrying about tasks such as data partitioning and scheduling.

In the Map phase, data is read from a distributed file system, partitioned among a set of computing nodes in the cluster, and sent to the nodes as a set of key-value pairs. The Map tasks process the input records independently of each other and produce intermediate results as key-value pairs. The intermediate results are stored on the local disk of the node running the Map task. When all the Map tasks are completed, the Reduce phase begins in which the intermediate data with the same key is aggregated. An optional Combine task can be used to perform data aggregation on the intermediate data of the same key for the output of the mapper before transferring the output to the Reduce task.

MapReduce programs take advantage of locality of data and the data processing takes place on the nodes where the data resides. In traditional approaches for data analysis, data is moved to the compute nodes which results in the delay in data transmission between the nodes in a cluster. However, the MapReduce programming model moves the computation to where the data resides thus decreasing the transmission of data and improving efficiency. The MapReduce programming model is well suited for parallel processing of massive scale data in which the data analysis tasks can be accomplished by independent map and reduce operations.

Figures show the execution flow of word count and inverted index MapReduce jobs. As seen from these figures, the sort and shuffle phase begins as soon as a map task completes and the reduce phase begins after the intermediate key-value pairs from all the map tasks are shuffled to the reducer.

Hadoop YARN7

Hadoop YARN is the next generation architecture of Hadoop (version 2.x). In the YARN architecture, the original processing engine of Hadoop (MapReduce) has been separated from the resource management component (which is now part of YARN) as shown in Figures. This makes YARN effectively an operating system for Hadoop that supports different processing engines on a Hadoop cluster such as MapReduce for batch processing, Apache Tez [60] for interactive queries, Apache Storm [65] for stream processing, for instance.

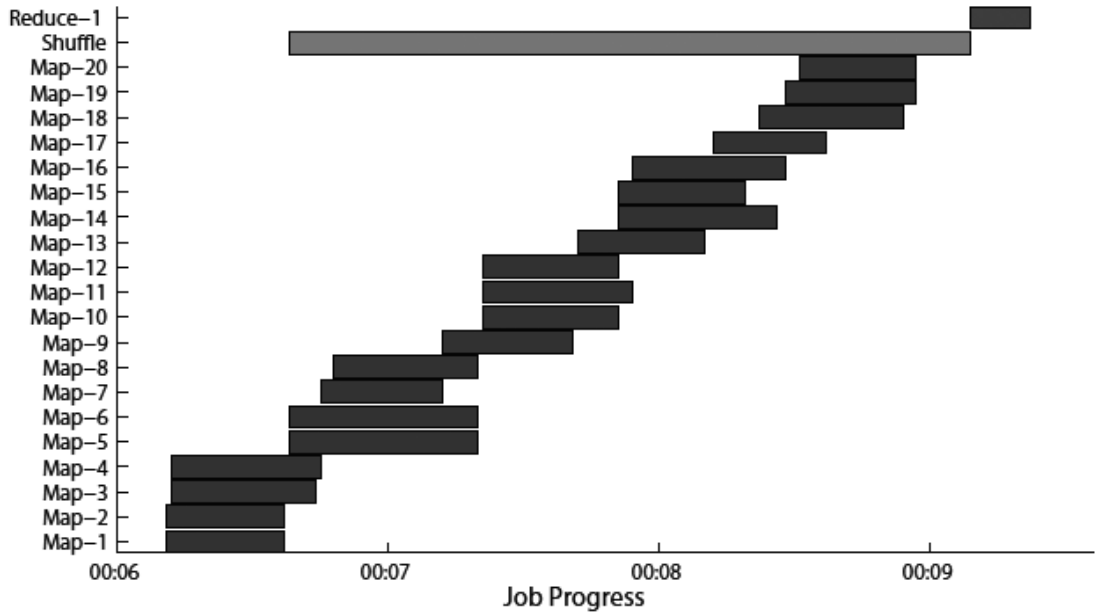


Fig.: Map/Reduce slot assignments for a word count MapReduce job

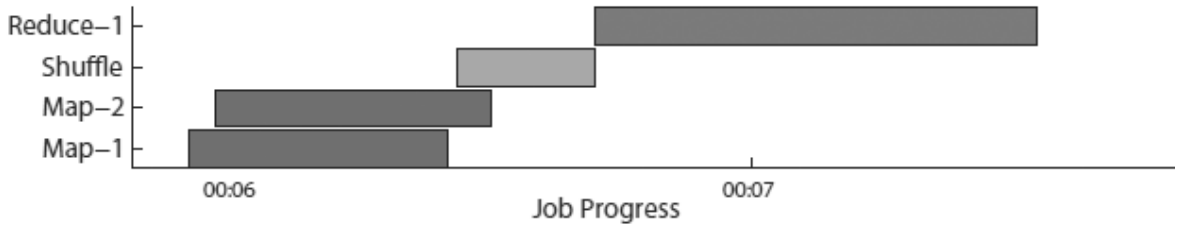


Fig.: Map/Reduce slot assignments for an inverted index MapReduce job

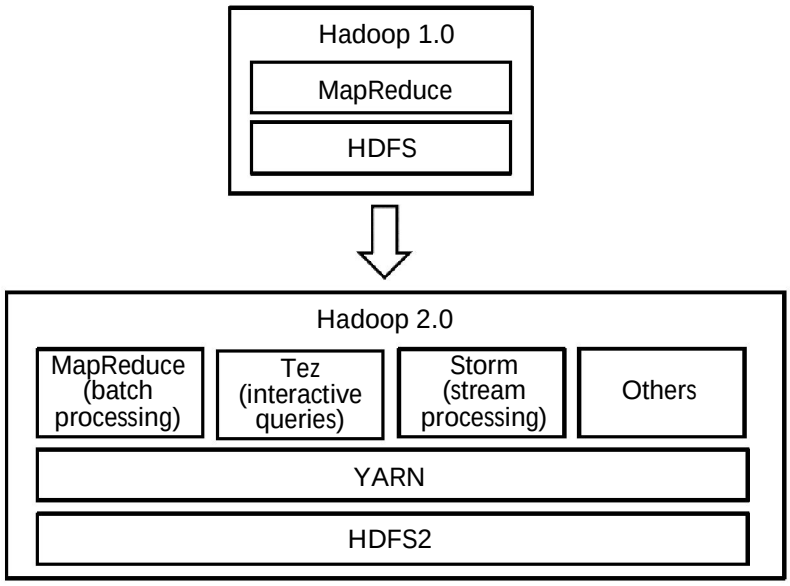


Fig.: Comparison of Hadoop 1.x and 2.x architectures

Figure shows the MapReduce job execution workflow for the next generation Hadoop MapReduce framework (MR2). The next-generation MapReduce architecture divides the two major functions of the JobTracker in Hadoop 1.x - resource management and job life-cycle management - into separate components - ResourceManager and ApplicationMaster. The key components of YARN are described as follows:

Resource Manager (RM)

RM manages the global assignment of compute resources to applications. RM consists of two main services:

- **Scheduler:** Scheduler is a pluggable service that manages and enforces the resource scheduling policy in the cluster.
- **Applications Manager (AsM):** AsM manages the running Application Masters in the cluster. AsM is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.

Application Master (AM)

A per-application AM manages the application's life cycle. AM is responsible for negotiating resources from the RM and working with the NMs to execute and monitor the tasks.

Node Manager (NM)

A per-machine NM manages the user processes on that machine.

Containers

Container is a bundle of resources allocated by RM (memory, CPU and network). A container is a conceptual entity that grants an application the privilege to use a certain amount of resources on a given machine to run a task. Each node has an NM that spawns multiple containers based on the resource allocations made by the RM.

Figure shows a YARN cluster with a Resource Manager node and three Node Manager nodes. There are as many Application Masters running as there are applications (jobs). Each application's AM manages the application tasks such as starting, monitoring and restarting tasks in case of failures. Each application has multiple tasks. Each task runs in a separate container. Containers in YARN architecture are similar to task slots in Hadoop MapReduce 1.x (MR1). However, unlike MR1 which differentiates between map and reduce slots, each container in YARN can be used for both map and reduce tasks. The resource allocation model in MR1 consists of a predefined number of map slots and reduce slots. This static allocation of slots results in low cluster utilization. The resource allocation model of YARN is more flexible with the introduction of resource containers which improve cluster utilization.

To better understand the YARN job execution workflow let us analyze the interactions between the main components on YARN. Figure 7.5 shows the interactions between a Client and Resource Manager. Job execution begins with the submission of a new application request by the client to the RM. The RM then responds with a unique application ID and information about cluster resource capabilities that the client will need in requesting resources for running the application's AM. Using the information received from the RM, the client constructs and submits an Application Submission Context which contains information such as scheduler queue, priority and user information. The Application Submission Context also contains a Container Launch Context which contains the application's jar, job files, security tokens and any resource requirements. The client can query the RM for application reports.

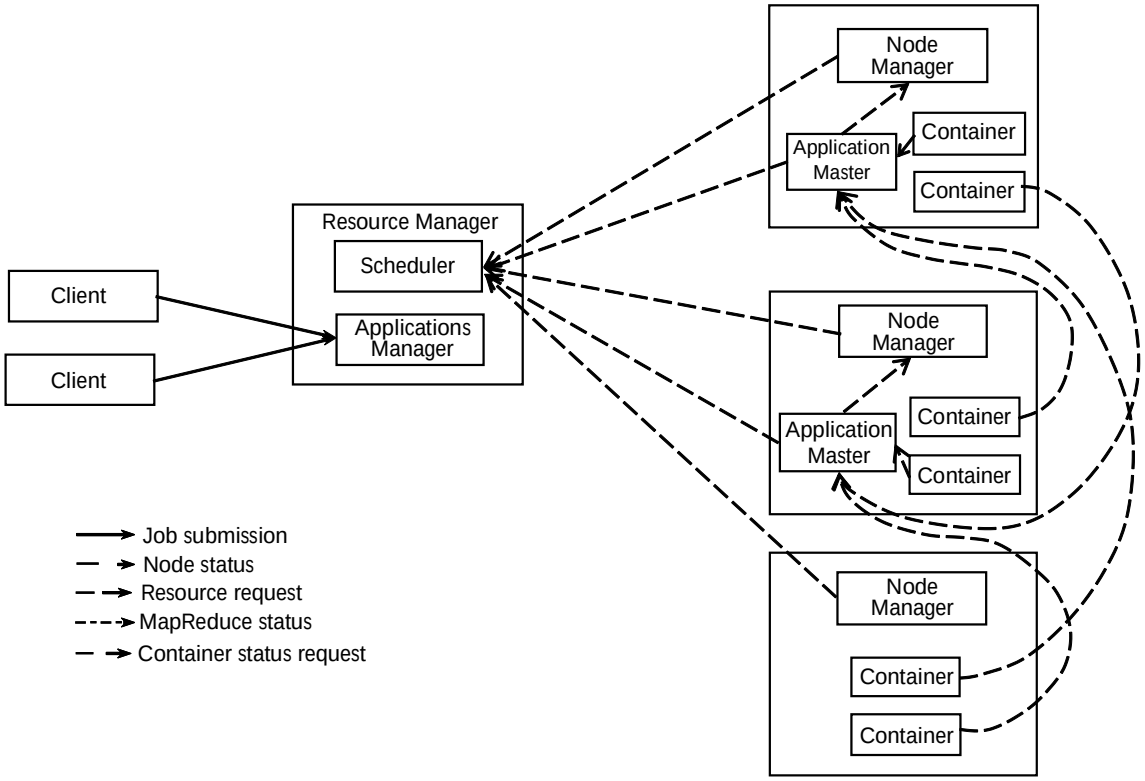


Fig.: Hadoop MapReduce Next Generation (YARN) job execution

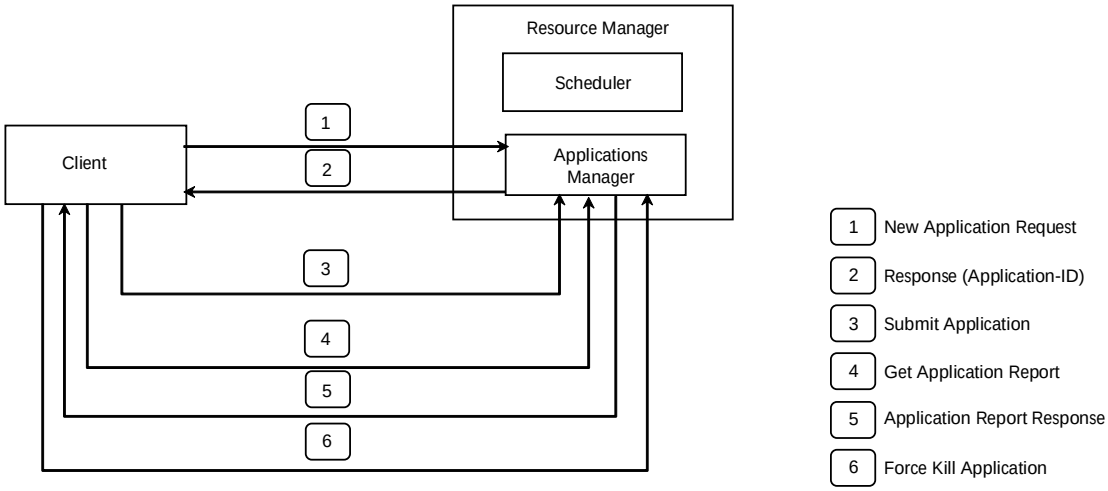


Fig.: Client - Resource Manager interaction

The client can also "force kill" an application by sending a request to the RM. Figure shows the interactions between Resource Manager and Application Master. Upon receiving an application submission context from a client, the RM finds an available container meeting the resource requirements for running the AM for the application. On finding a suitable container, the RM contacts the NM for the container to start the AM process on its node. When the AM is launched it registers itself with the RM. The registration process consists of handshaking that conveys information such as the RPC port that the AM will be listening on, the tracking URL for monitoring the application's status and progress, etc. The registration

response from the RM contains information for the AM that is used in calculating and requesting any resource requests for the application's individual tasks (such as minimum and maximum resource capabilities for the cluster). The AM relays heartbeat and progress information to the RM. The AM sends resource allocation requests to the RM that contains a list of requested containers, and may also contain a list of released containers by the AM. Upon receiving the allocation request, the scheduler component of the RM computes a list of containers that satisfy the request and sends back an allocation response. Upon receiving the resource list, the AM contacts the associated NMs for starting the containers. When the job finishes, the AM sends a Finish Application message to the RM.

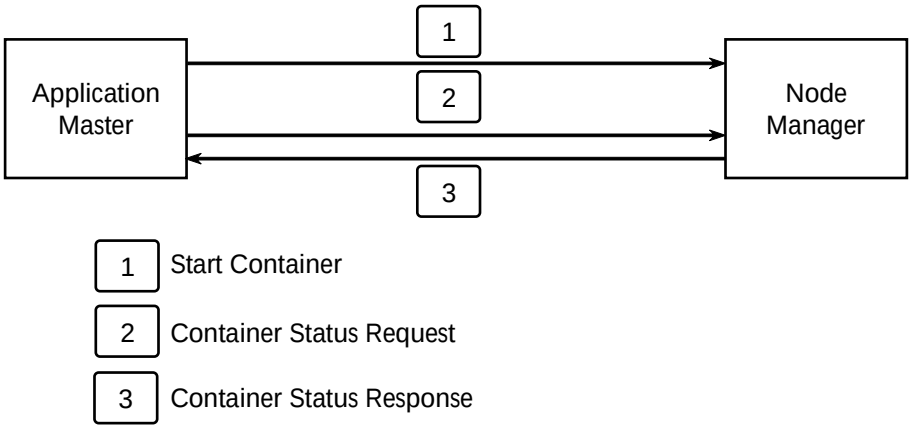


Fig.: Application Master - Node Manager interaction

Figure shows the interactions between the an Application Master and the Node Manager. Based on the resource list received from the RM, the AM requests the hosting NM for each container to start the container. The AM can request and receive a container status report from the Node Manager. Figure 7.8 shows the MapReduce job execution within a YARN cluster.

2.1.3 Hadoop Function

Q4. How does Hadoop Function work in big data?

Ans : **(Imp.)**

The first thing to know about the functioning of Hadoop is the way it utilizes multiple computing '"resources for executing a particular task. The core components of Hadoop include the following: Hadoop Distributed File System (HDFS)- It is a cluster of storage solutions that is highly reliable, more efficient, and economical and provides facilities to manage files containing related data across machines.

Hadoop MapReduce

It is a computational framework used in Hadoop to perform all the mathematical computations. It is based on a parallel and distributed implementation of MapReduce algorithm that provides high performance.

Hadoop facilitates the processing of large amounts of data present in both structured and unstructured forms. Hadoop clusters are created from the racks of commodity machines. Tasks are distributed across these machines (also known as nodes), which are allowed to work independently and provide their responses to the starting node. Moreover, it is possible to add or remove nodes dynamically in a Hadoop cluster on the basis of varying workloads. Hadoop has an ability to detect changes (which also include server failure) in the cluster and adjust to them, without causing any interruption in the system.

Hadoop accomplishes its operations (of dividing the computing tasks into subtasks that are handled bv individual nodes) with the help of the MapReduce model, which comprises two functions, namely, a

mapper and a reducer. The mapper function is responsible for mapping the computational subtasks to different nodes, and the reducer function takes the responsibility of reducing the responses from compute nodes, to a single result. The MapReduce model implements the MapReduce algorithm, as discussed earlier, to incorporate the capability of breaking data into manageable subtasks, processing the data on the distributed cluster simultaneously, and making the data available for additional processing or user consumption.

In the MapReduce algorithm, the operations of distributing task across various systems, handling the task placement for load balancing, and managing the failure recovery are accomplished by the map component (or the mapper function). The reduce component (or the reducer function), on the other hand, has the responsibility to aggregate all the elements together after the completion of the distributed computation.

When an indexing job is provided to Hadoop, it requires the organizational data to be loaded first. Xext, the data is divided into various pieces, and each piece is forwarded to different individual servers. Each server has a job code with the piece of data it is required to process. The job code helps Hadoop to track the current state of data processing. Once the server completes operations on the data provided to it, the response is forwarded with the job code being appended to the result.

In the end, results from all the nodes are integrated by the Hadoop software and provided to the user, as shown in Figure.

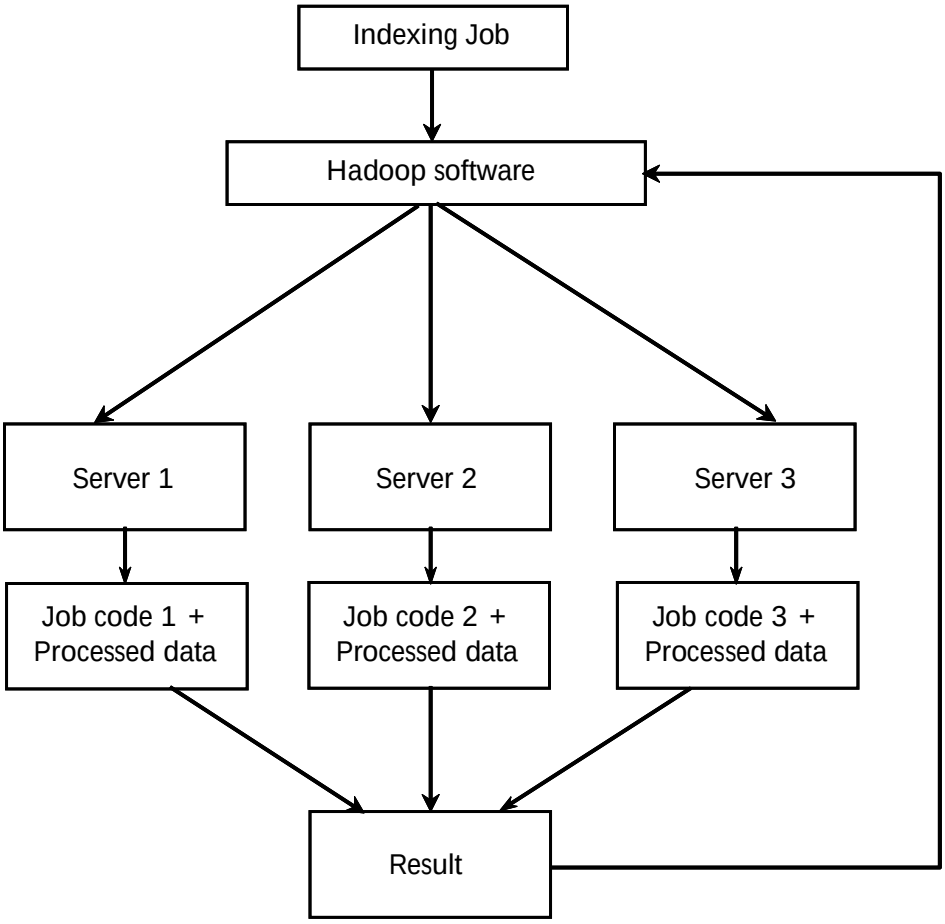


Fig.: Application Master - Node Manager interaction

We can understand the working of Hadoop by referring to the following example:

The call records of all the telephones in a city are being examined by a researcher who wants to know about those calls that are made by college students on the occasion of an event. The fields required as a result of the analysis carried out by the researcher include the timing of the event and the relevant information of the user. The query is fired on every machine to search the results from the call records stored with the machine, which will return the relevant results. Finally, a single result will be generated by aggregating individual results obtained from all the machines. We are considering that the records are collected in a Comma Separated Value (CSV) file.

- The processing starts by first loading the data into Hadoop and then applying the MapReduce programming model. Let us consider the following five columns to be contained in the CSV file:
  - u_id
  - u_name
  - c_name
  - sp_name
  - call_time

To identify individual users who made phone calls at a specific time, the u_id field is used. It helps us to determine the number of users who have called at the time. We can, thus, get the final output in terms of the number of users by whom calls were made during the specified time.

To obtain the final output, each mapper receives data line by line. Once the mapper completes its job, the results are shuffled or sorted by the Hadoop framework, which then combines the data in groups that are forwarded to the reducer. Ultimately, the final output is obtained from the reducer.

Data in Hadoop can be stored across multiple machines. Businesses can take the advantage of this storage facility to use multiple commodity machines. These machines are capable of hosting the Hadoop software. In this case, businesses will not need to create integrated systems.

Today, Hadoop is the most popular and used platform in businesses for Big Data processing. It helps in Big Data analytics by overcoming the obstacles that are usually faced in handling Big Data. In Hadoop, we can break down large computational problems into smaller tasks as smaller Dements can be analyzed economically and quickly. All these parts are analyzed in parallel, and the results of the analysis are regrouped to produce the final outputs.

Q5. What are the advantages of Hadoop?

Ans :

The following are the advantages of Hadoop:

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

2.1.4 Hadoop Ecosystem

Q6. Explain briefly about Hadoop Ecosystem.

Ans : **(Imp.)**

Hadoop ecosystem is a framework of various types of complex and evolving tools and components. Some of these elements may be very different from each other in terms of their architecture; however, what keeps them all together under a single roof is that they all derive their functionalities from the scalability and power of Hadoop.

In simple words, the Hadoop ecosystem can be defined as a comprehensive collection of tools and technologies that can be effectively implemented and deployed to provide Big Data solutions in a cost-effective manner. MapReduce and Hadoop Distributed File System (HDFS) are two core components of the Hadoop ecosystem that provide a great starting point to manage Big Data; however, they are not sufficient to deal with the Big Data challenges. Along with these two, the Hadoop ecosystem provides a collection of various elements to support the complete development and deployment of Big Data solutions.

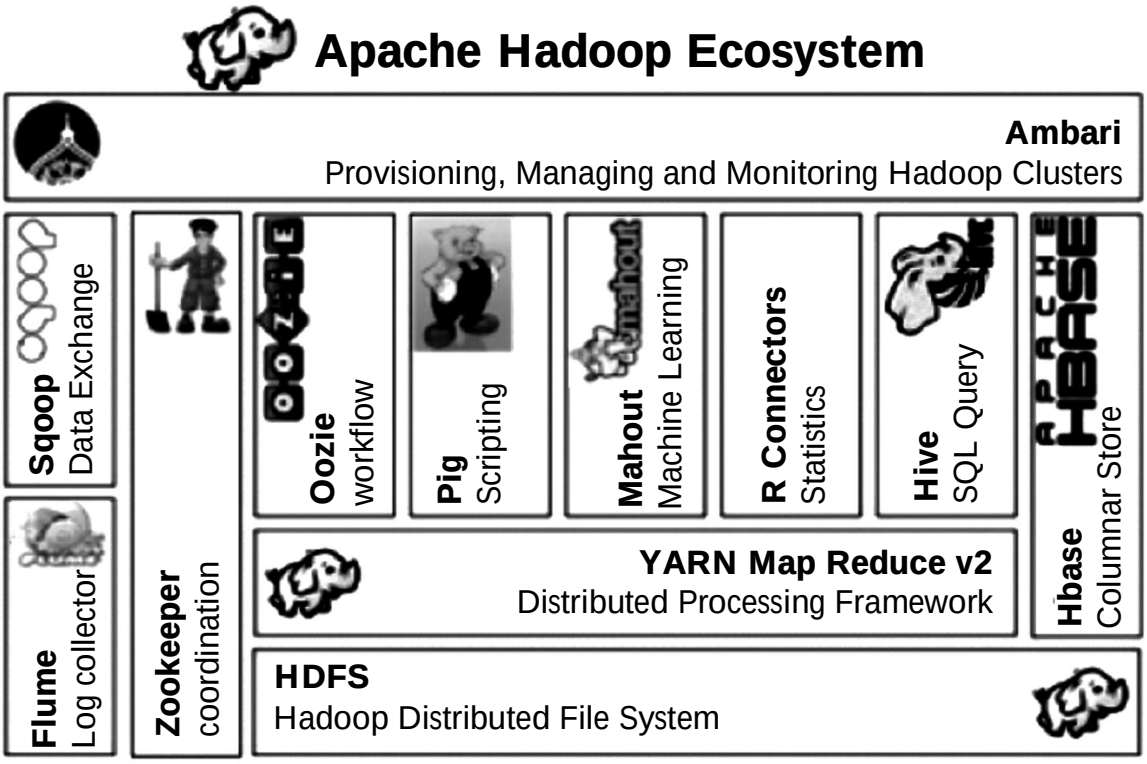


Fig.: Hadoop Ecosystem

All these elements enable users to process large datasets in real time and provide tools to support various types of Hadoop projects, schedule jobs, and manage cluster resources.

Figure depicts how the various elements of Hadoop involve at various stages of processing data:

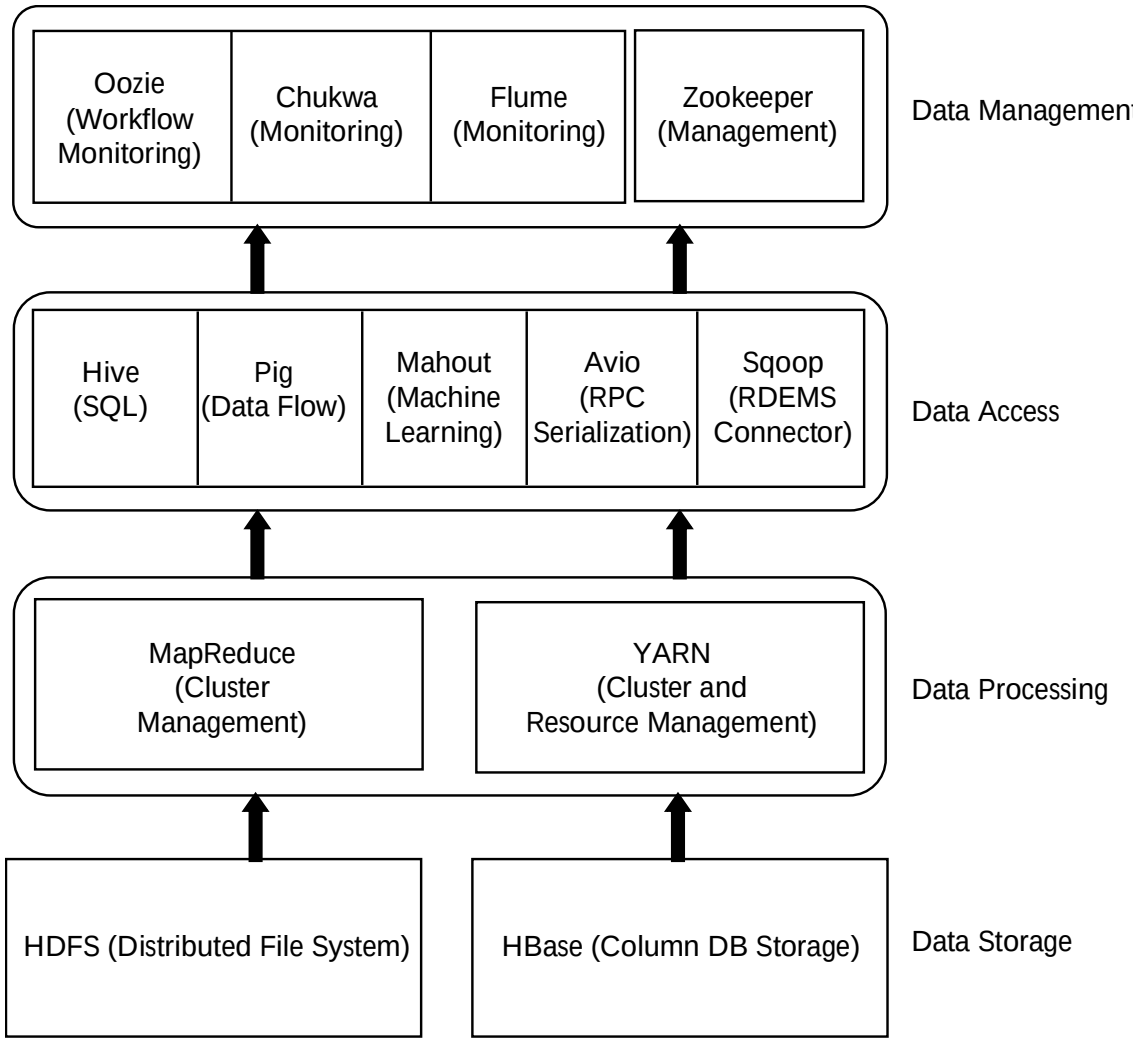


Fig.: Hadoop Ecosystem Elements at Various Stages of Data Processing

MapReduce and HDFS provide the necessary services and basic structure to deal with the core requirements of Big Data solutions. Other services and tools of the ecosystem provide the environment and components required to build and manage purpose-driven Big Data applications. In the absence of an ecosystem, the developers, database administrators, system and network managers will have to implement separate sets of technologies to create Big Data solutions. However, such an approach would prove to be expensive in terms of both time and money.

Q7. What are the components of Hadoop Ecosystem?

Ans :

Components of Hadoop Ecosystem

Let us start with the first component HDFS of Hadoop Ecosystem.

i) HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

- HDFS is a storage layer for Hadoop.
- HDFS is suitable for distributed storage and processing, that is, while the data is being stored, it first gets distributed and then it is processed.

- HDFS provides Streaming access to file system data.
- HDFS provides file permission and authentication.
- HDFS uses a command line interface to interact with Hadoop.

So what stores data in HDFS? It is the HBase which stores data in HDFS.

ii) HBase

- HBase is a NoSQL database or non-relational database .
- HBase is important and mainly used when you need random, real-time, read, or write access to your Big Data.
- It provides support to a high volume of data and high throughput.
- In an HBase, a table can have thousands of columns.

2.2 HADOOP DISTRIBUTED FILE SYSTEM

2.2.1 HDFS Architecture

Q8. Explain briefly about HDFS Architecture.

Ans : **(Imp.)**

HDFS has a master-slave architecture. It comprises a NameNode and a number of DataNodes. The NameNode is the master that manages the various DataNodes, as shown in Figure.

The NameNode manages HDFS cluster metadata, whereas DataNodes store the data. Records and directories are presented by clients to the NameNode. These records and directories are managed on the NameNode. Operations on them, such as their modification or opening and closing them are performed by the NameNode. On the other hand, internally, a file is divided into one or more blocks, which are stored in a group of DataNodes. DataNodes read and write requests from the clients. DataNodes can also execute operations like the creation, deletion, and replication of blocks, depending on the instructions from the NameNode.

2.2.2 Concept of Blocks in HDFS Architecture

Q9. Explain concept of Blocks in HDFS Architecture.

Ans : **(Imp.)**

W-disk has a certain block size, which is the basic measure of information that it can read or compose. File systems expand by managing information in pieces, which are an indispensable part of the disk block size. File system blocks are commonly a couple of kilobytes in size, while disk blocks are regularly 512 bytes. This is by and large straightforward for the file system client that is essentially perusing or composing a record of whatever length.

HDFS blocks are huge in contrast to disk blocks because they have to minimize the expense of the seek operation. Consequently, the time to transfer a huge record made of multiple blocks operates at the disk exchange rate. A quick computation demonstrates that if the seek time is around 10ms, and the exchange rate is 100 MB/s, then to assign the seek time that is 1% of the exchange time, we have to create a block size of around 100 MB. The default size is 64 MB, although numerous HDFS installations utilize 128 MB blocks. Map tasks of MapReduce (component of HDFS) typically work on one block at once, so if you have fewer assignments (fewer than the nodes in the group), your tasks will run slower.

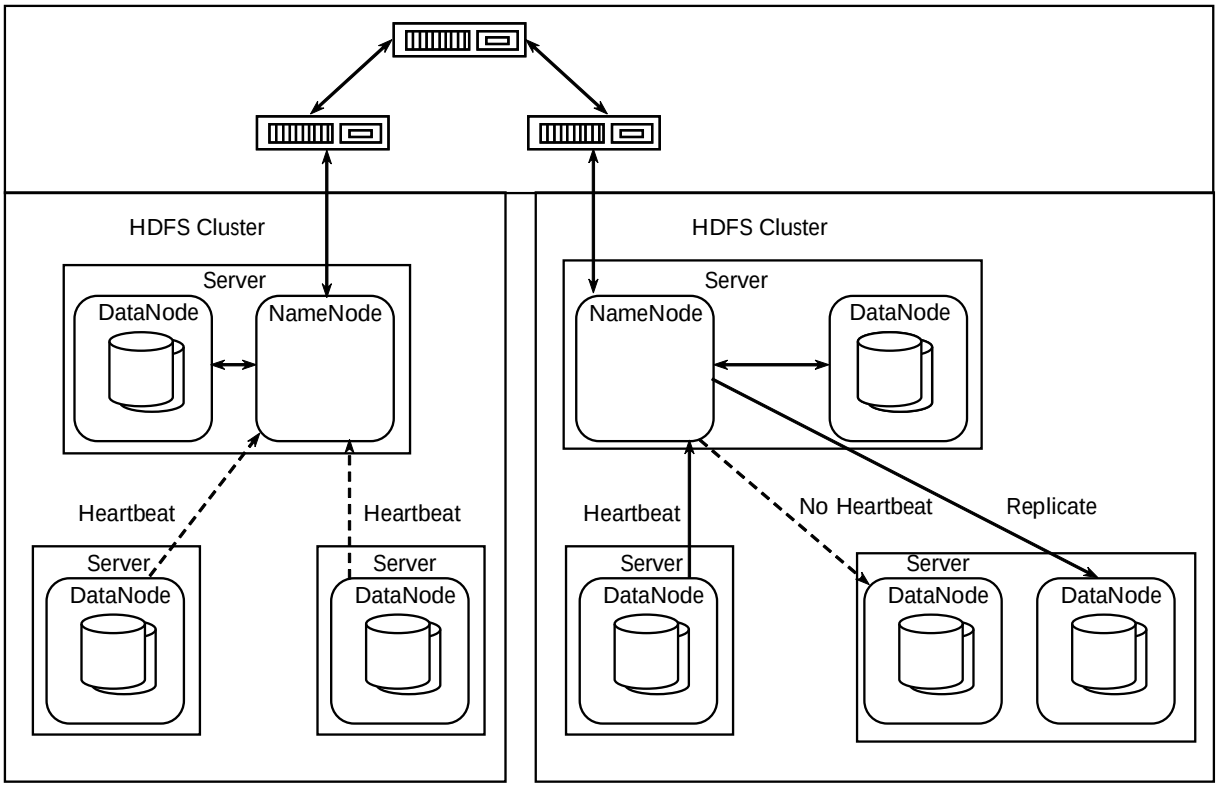


Fig.: Illustration of Hadoop Heartbeat Message

When a heartbeat message reappears or a new heartbeat message is received, the respective DataNode sending the message is added to the cluster.

HDFS performance is calculated through distribution of data and fault tolerance by detecting faults and quickly recurring the data. This recovery is completed through replication and resents in a reliable file system. It results in a reliable huge file storage.

To enable this task of reliability, one should facilitate number of tasks for failure management, some of which are utilized within HDFS and others are still in a process to be implemented:

- **Monitoring**

  DataNode and NameNode communicate through continuous signals ("Heartbeat"). If signal is not heard by either of the two, the node is considered to have failed and would be no longer available. The failed node is replaced by the replica and replication scheme is also changed.

- **Rebalancing**

  According to this process, the blocks are shifted from one to another location where ever the free space is available. Better performance can be judged by the increase in demand of data as well as the increase in demand for replication towards frequent node failures.

- **Metadata replication**

  These files are prove to failures; however, they maintain the replica of the corresponding file on the same HDFS.

There are a few advantages of abstracting a block for a distributed file system. The principal advantage is most evident: a record can be bigger than any single disk in a system. There is nothing that requires the blocks from a record to be put away on the same disk; so, they can exploit any of the disks in the cluster. It would still be possible to store a single document on a HDFS cluster with the entire cluster disks filled by its blocks.

Second, making the abstraction unit a block instead of a file improves the storage subsystem. The storage subsystem manages blocks, improving storage management (since blocks are of a fixed size, it is not difficult to compute the number of blocks that can be stored on a disk), and dispensing with metadata concerns.

To protect against corrupt blocks and disks and machine failure, each block is recreated on a few separate machines (usually three). In the event that a block gets occupied, a copy or duplicate block can be read from an alternate location in a straightforward manner to the client. A block that is no more accessible because of such issues can be replicated from its alternative location to other live machines.

Just like a disk file system, HDFS's fsck command can issue directives to blocks. For example, running the *% hadoop fsck -files -blocks* command lists the blocks that create each file in a file system.

2.2.3 Namenodes and Datanodes

Q10. Discuss about Namenodes and Datanodes in HDFS.

Ans : **(Imp.)**

An HDFS cluster has two node types working in a slave master design: a NameNode (the master) "and various DataNodes (slaves). The NameNode deals with the file system. It stores the metadata for all the documents and indexes in the file system. This metadata is stored on the local disk as two files: the file system image and the edit log. The NameNode is aware of the DataNodes on which all the pieces of a given document are found; however, it doesn't store block locations necessarily, since this data is recreated from DataNodes.

A client accesses the file system on behalf of the user by communicating with the DataNodes and NameNode. The client provides a file system, like the POSIX interface, so the user code does not require the NameNode and DataNodes in order to execute.

DataNodes are the workhorses of a file system. They store and recover blocks when they are asked to (by clients or the NameNode), and they report back to the NameNode occasionally with a list of blocks that they store externally.

Without the NameNode, the file system cannot be used. In fact, if the machine using the NameNode crashes, all files on the file system would be lost since there would be no way of knowing how to reconstruct the files from the blocks on DataNodes. This is why it is important to make the NameNode robust to cope with failures, and Hadoop provides two ways of doing this.

The first way is to take the back up of file documents. Hadoop can be set in a way that NameNode creates its state for various file systems. The normal setup choice is to write to the local disk and to a remote NFS mount.

Another way is to run a secondary NameNode, which does not operate like a normal NameNode.

Secondary NameNode periodically reads the filesystem, changes the log, and apply them into the fsimage file. When the NameNode is down, secondary NameNode will be online but this node will only have read permissions to fsimage and editlog file.

DataNodes ensure connectivity with the NameNode by sending heartbeat messages. Whenever the NameNode ceases to receive a heartbeat message from a DataNode, it unmaps the DataNode from the cluster and proceeds with further operations.

2.2.4 Features of HDFS

Q11. Discuss the Features of HDFS.

Ans : **(Imp.)**

HDFS ensures data integrity throughout the cluster with the help of the following features:

- **Maintaining Transaction Logs:** HDFS maintains transaction logs in order to monitor every operation and carry out effective auditing and recovery of data in case something goes wrong.
- **Validating Checksum:** Checksum is an effective error-detection technique wherein a numerical value is assigned to a transmitted message on the basis of the number of bits contained in the message. FIDFS uses checksum validations for verification of the content of a file. The validations are carried out as follows:
  - i) When a file is requested by a client, the contents are verified using checksum.
  - ii) If the checksums of the received and sent messages match, the file operations proceed further; otherwise, an error is reported.
  - iii) The message receiver verifies the checksum of the message to ensure that it is the same as in the sent message. If a difference is identified in the two values, the message is discarded assuming that it has been tempered with in transition. Checksum files are hidden to avoid tempering.
- **Creating Data Blocks:** HDFS maintains replicated copies of data blocks to avoid corruption of a file due to failure of a server. The degree of replication, the number of DataNodes in the cluster, and the specifications of the HDFS namespace are identified and implemented during the initial implementation of the cluster. However, these parameters can be adjusted any time during the operation of the cluster.

Data blocks are sometimes, also called block servers. A block server primarily stores data in a file system and maintains the metadata of a block. A block server carries out the following functions:

- Storage (and retrieval) of data on a local file system. HDFS supports different operating systems and provides similar performance on all of them.
- Storage of metadata of a block on the local file system on the basis of a similar template on the NameNode.
- Conduct of periodic validations for file checksums.
- Intimation about the availability of blocks to the NameNode by sending reports regularly.
- On-demand supply of metadata and data to the clients where client application programs can directly access DataNodes.
- Movement of data to connected nodes on the basis of the pipelining model.

2.2.5 Map Reduce

Q12. Discuss the working of MapReduce.

Ans :

Refer to Unit-II, Q.No. 3.

2.3 INTRODUCING HBASE

Q13. What is HBase?

Ans :

Meaning

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

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

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

Uses

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

Applications

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

2.3.1 HBase Architecture, Regions, Storing Big Data with HBase

Q14. Explain the HBase Architecture.

Ans : **(Imp.)**

Applications store information in labeled tables, and the values stored in table cells get updated from dime to time. A cells value is an unread array of bytes.

Keys in table rows are also byte arrays, so hypothetically anything can act as a row key. Table rows are categorized by the row key, which is the table's primary key. By default, the data type of the primary key is byte-ordered. All table access is via the table primary key.

Row/columns are clustered into column families. All family members of a column share a common prefix, e.g., the columns j*ava:android* and *java servlets* are both members of the *java* family. The column family should consist of printable characters. The suffix can be made of any random bytes.

A table's column family members must be specified beforehand as an aspect of the table schema meaning, but new column family members can be added on a need basis. For example, a new column address:pincode can be provided by a client as part of an upgrade, and its value continued, as long as the column family exists in place on the targeted table.

Physically, all family members of a column are saved together on the file system. So, previously we described HBase as a column-oriented entity, but it would be more precise if it is described as a *column-family* oriented entity. Because storage requirements are fulfilled at the column family level, all column family members are recommended to have the same general access design and size features. In short, HBase tables are like the ones in RDBMS, with cells being versioned, sorted rows, and on-the-fly addition of columns as per the client's requirements.

i) Regions

Tables are automatically partitioned horizontally into regions by HBase. Each region consists of a subset of rows of a table. Initially, a table comprises a single region but as the size of the region grows, after it crosses a configurable size limit, it splits at the boundary of a row into two new regions of almost equal size, as shown in Figure.

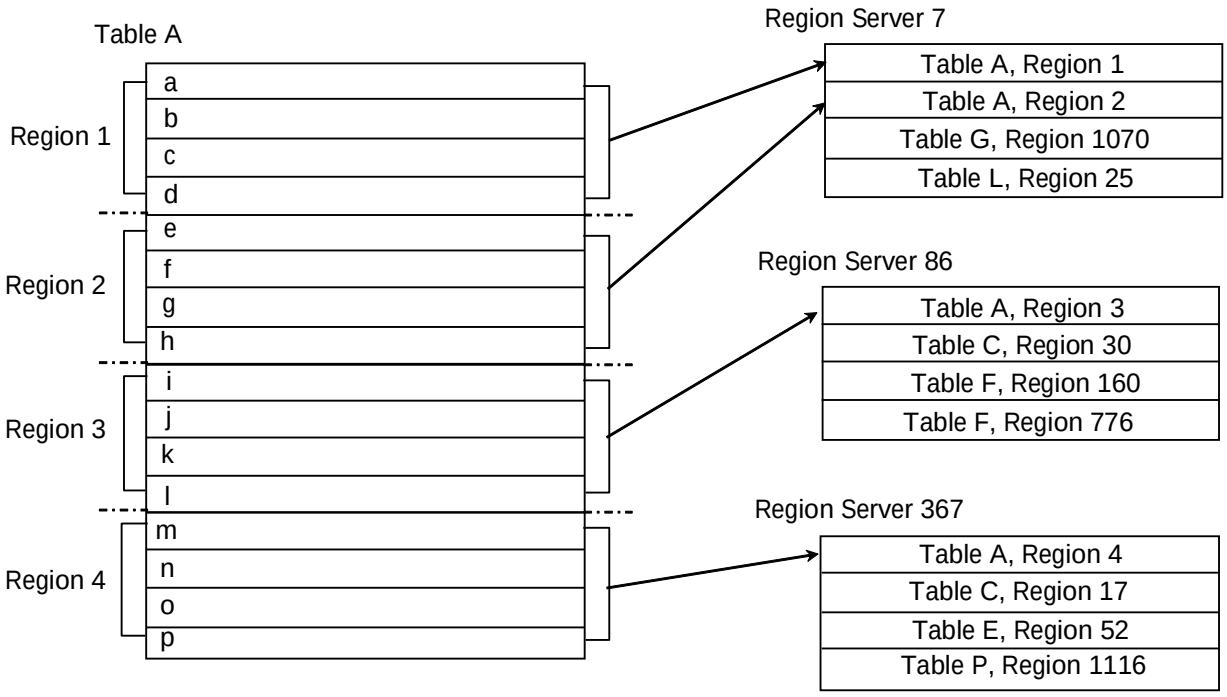


Fig.: Displaying Regions in HBsse

Regions are units that get spread over a cluster in HBase. Hence, a table too big for any single server can be carried by a cluster of servers with each node hosting a subset of all the regions of a table. This is also the medium by which the loading on a table gets spread. At a given time, the online group of sorted regions comprises the table's total content.

HBase persists data via the Hadoop file system API. There are multiple implementations of the file system interface-one each for the local file system, the KFS file system, Amazon's S3, and HDFS. HBase can persist to any of these implementations. By default, unless set otherwise, HBase writes into the local file system. The local file system is fine for experimenting with your initial HBase install, but thereafter, usually the first configuration made in an HBase cluster involves pointing HBase at the HDFS cluster to use.

ii) Storing Big Data with HBase

HBase is a dispersed, non-relational (columnar) database that uses HDFS. It is designed according to Google Bigtable (a compressed, high performance proprietary data storage system built on the Google file system) and is equipped for facilitating extensive tables (billions of sections/columns) on the grounds that it is layered on Hadoop clusters of appliance hardware. HBase gives arbitrary, continuous read/write access to enormous information. It is exceptionally configurable, giving a lot of flexibility to address immense measures of information proficiently.

In HBase, all information is put away into tables with columns and sections like relational frameworks (RDBMS). The cell is known as an intersection of a row and column. One essential distinction between HBase tables and RDBMS tables is versioning. Each ceil value consists of a version property, which is just a timestamp uniquely distinguishing the cell. Versioning tracks changes in the cell and makes it possible to retrieve any version of the contents, if required. HBase stores the information in cells in descending order (utilizing the timestamp), so a read will obviously find the newer values first.

iii) Interacting with the Hadoop Ecosystem

Writing programs or using specialty query languages is not the only way you interact with the Hadoop ecosystem. IT teams that manage infrastructures need to control Hadoop and the Big Data applications created for it. As Big Data becomes mainstream, non-technical professionals will want to try to solve business problems with it.

Hadoop-supported business distributions are always showing signs of change. New tools and technologies are presented, existing technologies are enhanced, and a few innovations are replaced by better substitutions. This is one of the key advantages of open source. An alternate is the selection of open-source innovations or applications by business organizations. These organizations upgrade the applications, making them better for everybody at an economical cost. This is how the Hadoop ecosystem has developed and why it is a good decision for serving to tackle your enormous information challenges.

iv) HBase in Operation - Programming with HBase

HBase keeps individual catalog tables internally, called -ROOT- and .META., within which it maintains the current list, state, recent history, and location of all regions at the top of the cluster. The -ROOT- table has the list of . META. table regions. The .META. table has a list of all the regions in it. Entries in these tables are keyed utilizing the region's starting row. The Columns are sorted so discovering the region that has a specific row is a matter of a lookup to discover the first entry whose key is more noteworthy than or equivalent to that of the asked for row key. Figure displays an HBase cluster:

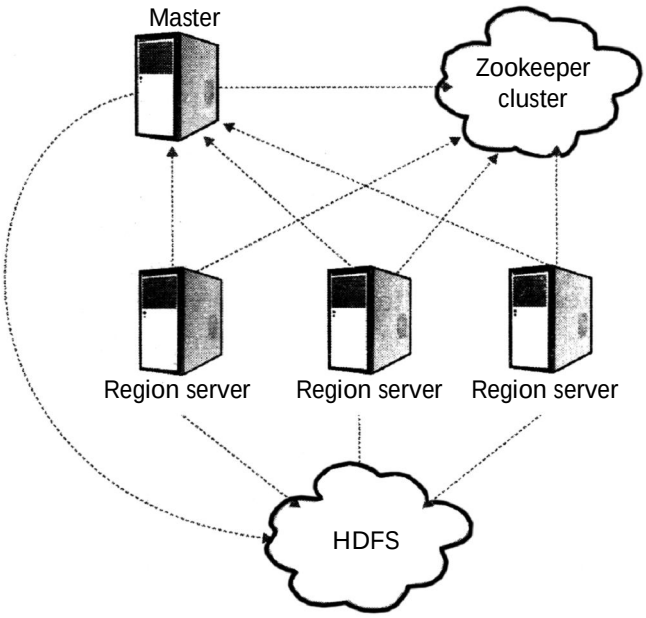


Fig.: Displaying an HBase Cluster

Fresh clients that connect to the ZooKeeper cluster need to first learn the location of the -ROOT- table. Clients consult -ROOT- to elicit the location of the .META, region, the scope of which covers the requested row. The client searches for the located .META. region to know the hosting region and its location. Thereafter, the client interacts directly with the hosting region-server.

To avoid making three round-trips per row operation, clients store all they learn while traversing -ROOT- and .META, caching locations as well as user-space region rows, so they can find hosting regions themselves without looking at the .META. table. As the clients work, they carry on using the cached entry, until there is a fault. When this happens, it means that the region has moved, and the client consults the .META. again to learn the new location. Therefore, if the contacted .META. region has moved, then -ROOT- is reconsulted.

Writes incoming at a region-server are first connected to a commit log file and then added to a cache. When the cache fills up, its content is moved to the file system. The log is stored on HDFS, so it stays available.

2.3.2 Combining HBase and HDFS

Q15. Explain about Combining HBase and HDFS.

Ans : **(Imp.)**

HBase's utilization of HDFS is different from how it is utilized by MapReduce. In MapReduce, by _and large, HDFS documents are opened, their content streamed through a map task (which takes a set of data and converts it into another set of data), and are closed. In HBase, data files are opened on group startup and kept open so that the user does not have to pay the file open expenses on each access. Hence, HBase has a tendency to handle issues not commonly experienced by MapReduce clients, which are as follows:

i) File descriptors shortage

Since we keep documents open on a loaded cluster, it doesn't take too long to run the documents on a framework. For example, we have a cluster having three hubs, each running an instance of a DataNode and region server, and we are performing an upload on a table that presently has 100 regions and 10 column families. Suppose every column family has two file records. In that case, we will have 100 x 10 x 2 = 2000 records open at any given time. Add to this the collective random descriptors consumed by good scanners, and Java libraries. Each open record devours no less than one descriptor on the remote DataNode. The default quantity of record descriptors is 1024.

ii) Not many DataNode threads

Essentially, the Hadoop DataNode has a higher limit of 256 threads it can run at any given time. Suppose we use the same table measurements cited earlier. It is not difficult to perceive how we can surpass this figure given in the DataNode since each open connection with a record piece consumes a thread. If you look in the DataNode log, you will see an error like xceivercount 258 limit exceeds point of simultaneous xcievers 256; therefore, you need to be careful in using the threads.

iii) Bad blocks

The Dfsclientclass in the region server will have a tendency to mark document blocks as bad if the server is heavily loaded. Blocks can be recreated only three times. Therefore, the region server will proceed onward for the recreation of the blocks. But if this recreation is performed during a period of heavy loading, we will have two of the three blocks marked as bad. In the event that the third block is discovered to be bad, we will see an error, stating, No live hubs contain current block in region server logs. During startup, you may face many issues as regions are opened and deployed. At the worst case, set the dfs.datanode.socket.write.timeout property to zero. Do note that this configuration needs to be set in a location accessible to HBase Dfsclient; set it in the hbase-site.xml file or by linking the hadoop-site.xml (or hdfssite.xml) file to your HBase conf directory.

iv) UI

HBase runs a Web server on the master to present a perspective on the condition of your running cluster. It listens on port 60010 by default. The master UI shows a list of basic functions, e.g., software renditions, cluster load, request rates, list of group tables, and participant region servers. In the master UI, click a region server and you will be directed to the Web server running the individual region server. A list of regions carried by this server and metrics like consumed resources and request rate would be displayed.

v) Schema design

HBase tables are similar to RDBMS tables, except that HBase tables have versioned cells, sorted rows, and columns. The other thing to keep in mind is that an important attribute of the column (-family)-oriented database, like HBase, is that it can host extensive and lightly populated tables at no extra incurred cost.

vi) Row keys

Spend a good time in defining your row key. It can be utilized for grouping information as a part of routes. If your keys are integers, utilize a binary representation instead of a persistent string form of a number as it requires lesser space.

Now, let's create a table in Hbase, insert data in it, and then perform a cleanup.

To create a table, you must define its schema. The schema has table attributes and a table column family list. Column families have properties that you set at the time of schema definition. The schemas can be revised later by putting the table off line using the disable command (shell), making the changes using the alter command, and then restoring the table with the enable command.

You can create a table with the name test having a single column family -data- by using the following command:

hbase(main):007:0> create 'test', 'data' 0 row(s) in 4.3066 seconds

To view if the creation of the new table was successful, run the list command. This displays all the existing tables, as follows:

hbase(main):019:0> list

test 1 row(s) in 0.1112 seconds

Insert data into different rows, and columns at the data column family, and then list the content, by using the following commands:

hbase(main):021:0> put 'test', 'rowl', 'data:l', 'valuel'

0 row(s) in 0.0454 seconds

hbase(main):022:0> put 'test', 'row2', 'data:2', 'value2'

0 row(s) in 0.0035 seconds

hbase(main):023:0> put 'test', 'row3', 'data:3', 'value3'

0 row(s) in 0.0090 seconds hbase(main):024:0> scan 'test'

ROW	COLUMN+CELL		
rowl	column=data:1,	timestamp=1240148026198,	value=valuel
row2	column=data:2,	timestamp=1240148040035,	value=value2
row3	column=data:3,	timestamp=1240148047497,	value=value3
	3 row(s) in 0.0825 seconds		

To remove a table, you must first disable it, as follows:

hbase(main):025:0> disable 'test'

15/02/15 06:40:13 INFO client.HBaseAdmin: Disabled test

0 row(s) in 6.0426 seconds hbase(roain):026:0> drop 'test'

15 /02/10 06:40:17 INFO client.HBaseAdmin: Deleted test

0 row(s) in 0.0210 seconds

hbase(main):027:0> list 0 row(s) in 2.0645 seconds

Shut down your HBase instance by running the following command:

% stop-hbase.sh

HBase is written in Java. Its classes and utilities are included in the org.apache.hadoop.hbase.mapred package, aided by using HBase in MapReduce jobs as a source. The TableInputFormat class splits the region borders, so mars are assigned a single region to work. The TableOutputFormat class writes the reduce result in HBase. The RowCounter class runs a map task using TableInputFormat, for counting rows. This class implements the Tool and HBase TableMap interfaces, as well as the org.apache, hadoop.mapred.Mapper interface, which sets the map inputs types passed by using the table input format. The createSubmittable Job () method analyses the added arguments command line configuration and estimates the table and columns we are to run using the RowCounter class. It also invokes the TableMapReduceUtil. initTableMap Job () utility method and sets the input format to TableInputFormat. MapReduce checks all the columns. In case some are found to be empty, it does not count the related rows; otherwise, it increments the Counters. ROWS property by one.

2.3.3 Features of HBase

Q16. State the Features of HBase.

Ans :

Some of the main features of HBase are:

i) **Consistency:** In spite of not being an ACID implementation, HBase supports consistent read and write operations. This makes HBase suitable for high-speed requirements where RDBMS- supported extra features, such as full transaction support or typed columns, are not required.

ii) **Sharding:** HBase allows distribution of data using an underlying file system and supports transparent, and automatic splitting and redistribution of content.

iii) **High availability:** HBase implements region servers to ensure the recovery of LAN and WAN operations in case of a failure. The master server at the core monitors the regional servers and manages all the metadata for the cluster.

iv) **Client API:** HBase supports programmatic access using Java APIs.

v) **Support for IT operations:** HBase provides a set of built-in Web pages to view detailed operational insights about the system)

2.4 HIVE

Q17. What is HIVE ? Write its features and characteristics.

Ans :

Meaning

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.

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. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Characteristics

- In Hive, tables and databases are created first and then data is loaded into these tables.
- Hive as data warehouse designed for managing and querying only structured data that is stored in tables.
- While dealing with structured data, Map Reduce doesn't have optimization and usability features like UDFs but Hive framework does. Query optimization refers to an effective way of query execution in terms of performance.
- Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming. It reuses familiar concepts from the relational database world, such as tables, rows, columns and schema, etc. for ease of learning.
- Hadoop's programming works on flat files. So, Hive can use directory structures to "partition" data to improve performance on certain queries.
- A new and important component of Hive i.e. Metastore used for storing schema information. This Metastore typically resides in a relational database. We can interact with Hive using methods like
  - Web GUI
  - Java Database Connectivity (JDBC) interface
- Most interactions tend to take place over a command line interface (CLI). Hive provides a CLI to write Hive queries using Hive Query Language(HQL)
- Generally, HQL syntax is similar to the SQL syntax that most data analysts are familiar with. The Sample query below display all the records present in mentioned table name.
  - **Sample query** : Select * from <TableName>
  - Hive supports four file formats those are **TEXTFILE, SEQUENCEFILE, ORC and RCFILE**(Record Columnar File).

Q18. Describe the components of Hive Architecture.

Ans : **(Imp.)**

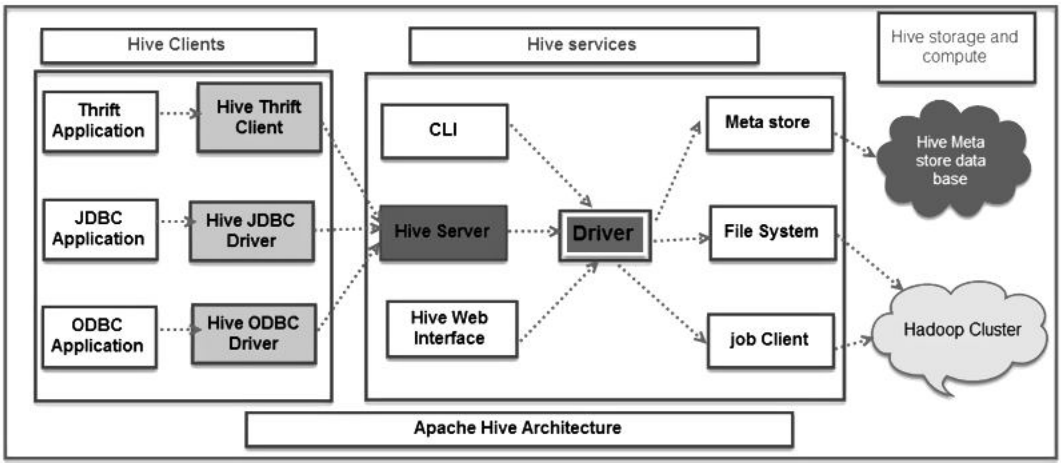


Fig.: Components of Hive Architecture

Hive Consists of Mainly 3 core parts

1. Hive Clients
2. Hive Services
3. Hive Storage and Computing

1. Hive Clients

Hive provides different drivers for communication with a different type of applications. For Thrift based applications, it will provide Thrift client for communication.

For Java related applications, it provides JDBC Drivers. Other than any type of applications provided ODBC drivers. These Clients and drivers in turn again communicate with Hive server in the Hive services.

2. Hive Services

Client interactions with Hive can be performed through Hive Services. If the client wants to perform any query related operations in Hive, it has to communicate through Hive Services.

CLI is the command line interface acts as Hive service for DDL (Data definition Language) operations. All drivers communicate with Hive server and to the main driver in Hive services as shown in above architecture diagram.

Driver present in the Hive services represents the main driver, and it communicates all type of JDBC, ODBC, and other client specific applications. Driver will process those requests from different applications to meta store and field systems for further processing.

3. Hive Storage and Computing:

Hive services such as Meta store, File system, and Job Client in turn communicates with Hive storage and performs the following actions.

- Metadata information of tables created in Hive is stored in Hive "Meta storage database".
- Query results and data loaded in the tables are going to be stored in Hadoop cluster on HDFS.

Q19. Write the step by step installation process of Hive.

Ans : **(Imp.)**

HIVE Installation

Step 1: The JAVA and Hadoop must be preinstalled on your system.

Step 2: For Hive Download Hive from http://apache.petsads.us/hive/hive-0.14.0/.

It gets downloaded in /user/download folder.

Check for the files

$ cd /usr/download

If the download is successful you will find the below file by typing ls command.

$ ls

apache-hive-0.14.0-bin.tar.gz

Unzip it

$ tar zxvf apache-hive-0.14.0-bin.tar.gz

Copy File

Copy the file to /usr/local/hive directory with root user

$ su -

passwd:

Copying files to /usr/local/hive directory

$ mv /user/download/apache-hive-0.14.0-bin /usr/local/hive

Environment for Hive

Add the below line in ./bashrc file

export HIVE_HOME=/usr/local/hive**</br>**
export PATH=$PATH:$HIVE_HOME/bin**</br>**
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.**</br>**
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.**</br></br>**

Now run the ./bashrc file to reflect those changes.

$ source ~/.bashrc**</br>**

Configuring Hive

For Configuring Hive hive-env.sh is edited. This file is present in HIVE_HOME/conf.

$ cd $HIVE_HOME/conf
$ cp hive-env.sh.template hive-env.sh

Add the below line to hive-env.sh.

export HADOOP_HOME=/usr/local/hadoop

Step 3 : Derby Database

Hive uses external Database server to configure Metastore.

Now Download and install Apache Derby

Follow the steps given below to download and install Apache Derby.

Downloading Apache Derby:

The following command is used to download Apache Derby. It takes some time to download.

$ cd ~

$ wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz**</br>**

Unzip it:

$tar zxvf db-derby-10.4.2.0-bin.tar.gz

Copy File:

Move file to /usr/local/derby directory

$mv /user/download/db-derby-10.4.2.0-bin /usr/local/derby**</br>**

Set up the environment

Add the below line to ./bashrc file

export DERBY_HOME=/usr/local/derby

export PATH=$PATH:$DERBY_HOME/bin

Apache Hive

export CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/

derbytools.jar

To reflect the changes type

$ source ~/.bashrc

Create a directory to store Metastore

Create a directory named data in $DERBY_HOME directory to store Metastore data.

$ mkdir $DERBY_HOME/data**</br>**

Step 4: Configuring Metastore of Hive

Edit hive-site.xml and append the following lines between the <configuration> and </configuration> tags :

<property>

<name>javax.jdo.option.ConnectionURL**</name>**

<value>jdbc:derby://localhost:1527/metastore_db;create=true **</value>**

<description>JDBC connect string for a JDBC metastore**</description>**

</property>

Create a file named jpox.properties and add the following lines into it :

javax.jdo.PersistenceManagerFactoryClass =

org.jpox.PersistenceManagerFactoryImpl

org.jpox.autoCreateSchema = false

org.jpox.validateTables = false

org.jpox.validateColumns = false

org.jpox.validateConstraints = false

org.jpox.storeManagerType = rdbms

org.jpox.autoCreateSchema = true

org.jpox.autoStartMechanismMode = checked

org.jpox.transactionIsolation = read_committed

javax.jdo.option.DetachAllOnCommit = true

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create = true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine

Step 5: Verify Hive Installation

Create the /tmp folder and a separate Hive folder in HDFS. Here, we use the /user/hive/warehouse folder. You need to set write permission for these newly created folders as shown below :

chmodg+w

Set them in HDFS using the following commands :

$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp

$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$ $HADOOP_HOME/bin/hadoop fs -chmodg+w /tmp

$ $HADOOP_HOME/bin/hadoop fs -chmodg+w /user/hive/warehouse

The following commands are used to verify Hive installation :

$ cd $HIVE_HOME

$ bin/hive

hive

2.5 PIG

Q20. What is Apache Pig. State the features of Apache.

Ans :

Meaning

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

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark. Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

Features of Apache Pig

Let's see the various uses of Pig technology.

1. **Ease of programming:** Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.
2. **Optimization opportunities:** It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
3. **Extensibility:** A user-defined function is written in which the user can write their logic to execute over the data set.
4. **Flexible:** It can easily handle structured as well as unstructured data.
5. **In-built operators:** It contains various type of operators such as sort, filter and joins.

Q21. Explain the Architecture of Apache Pig.

Ans : **(Imp.)**

For writing a Pig script, we need Pig Latin language and to execute them, we need an execution environment. The architecture of Apache Pig is shown in the below image.

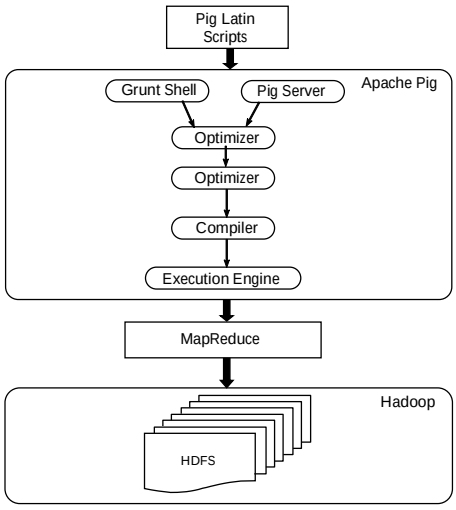


Fig.: Apache Pig Architecture

i) Pig Latin Scripts

Initially as illustrated in the above image, we submit Pig scripts to the Apache Pig execution environment which can be written in Pig Latin using built-in operators.

There are three ways to execute the Pig script :

- **Grunt Shell :** This is Pig's interactive shell provided to execute all Pig Scripts.
- **Script File :** Write all the Pig commands in a script file and execute the Pig script file. This is executed by the Pig Server.
- **Embedded Script :** If some functions are unavailable in built-in operators, we can programmatically create User Defined Functions to bring that functionalities using other languages like Java, Python, Ruby, etc. and embed it in Pig Latin Script file. Then, execute that script file.

ii) Parser

From the above image you can see, after passing through Grunt or Pig Server, Pig Scripts are passed to the Parser. The Parser does type checking and checks the syntax of the script. The parser outputs a DAG (directed acyclic graph). DAG represents the Pig Latin statements and logical operators. The logical operators are represented as the nodes and the data flows are represented as edges.

iii) Optimizer

Then the DAG is submitted to the optimizer. The Optimizer performs the optimization activities like split, merge, transform, and reorder operators etc. This optimizer provides the automatic optimization feature to Apache Pig. The optimizer basically aims to reduce the amount of data in the pipeline at any instant of time while processing the extracted data, and for that it performs functions like:

- **PushUpFilter:** If there are multiple conditions in the filter and the filter can be split, Pig splits the conditions and pushes up each condition separately. Selecting these conditions earlier, helps in reducing the number of records remaining in the pipeline.
- **PushDownForEachFlatten:** Applying flatten, which produces a cross product between a complex type such as a tuple or a bag and the other fields in the record, as late as possible in the plan. This keeps the number of records low in the pipeline.
- **ColumnPruner:** Omitting columns that are never used or no longer needed, reducing the size of the record. This can be applied after each operator, so that fields can be pruned as aggressively as possible.
- **MapKeyPruner:** Omitting map keys that are never used, reducing the size of the record.
- **LimitOptimizer:** If the limit operator is immediately applied after a load or sort operator, Pig converts the load or sort operator into a limit-sensitive implementation, which does not require processing the whole data set. Applying the limit earlier, reduces the number of records.

This is just a flavor of the optimization process. Over that it also performs **Join**, **Order By** and **Group By**functions.

To shutdown, automatic optimization, you can execute this command:

```
pig -optimizer_off [opt_rule | all ]
```

iv) Compiler

After the optimization process, the compiler compiles the optimized code into a series of MapReduce jobs. The compiler is the one who is responsible for converting Pig jobs automatically into MapReduce jobs.

v) Execution engine

Finally, as shown in the figure: Apache Pig Architecture, these MapReduce jobs are submitted for execution to the execution engine. Then the MapReduce jobs are executed and gives the required result. The results can be displayed on the screen using "**DUMP**" statement and can be stored in the HDFS using "**STORE**" statement.

2.6 PIG LATIN

Q22. Explain briefly about Pig Latin.

Ans :

Meaning

The Pig Latin is a data flow language used by Apache Pig to analyze the data in Hadoop. It is a textual language that abstracts the programming from the Java MapReduce idiom into a notation.

Pig Latin Statements

The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.

- It can span multiple lines.
- Each statement must end with a semi-colon.
- It may include expression and schemas.
- By default, these statements are processed using multi-query execution.

Pig Latin Conventions

S.No.	Convention	Description
i)	( )	The parenthesis can enclose one or more items. It can also be used to indicate the tuple data type. **Example** - (10, xyz, (3,6,9))
ii)	[ ]	The straight brackets can enclose one or more items. It can also be used to indicate the map data type. **Example** - [INNER | OUTER]
iii)	{ }	The curly brackets enclose two or more items. It can also be used to indicate the bag data type **Example** - { block | nested_block }
iv)	...	The horizontal ellipsis points indicate that you can repeat a portion of the code. **Example** - cat path [path ...]

Q23. Explain various data types of Pig Latin.

Ans :

Latin Data Types

Simple Data Types

Type	Description	Example
int	It defines the signed 32-bit integer.	2
long	It defines the signed 64-bit integer.	2L or 2l
float	It defines 32-bit floating point number.	2.5F or 2.5f or 2.5e2f or 2.5.E2F
double	It defines 64-bit floating point number.	2.5 or 2.5 or 2.5e2f or 2.5.E2F
chararray	It defines character array in Unicode UTF-8 format.	javatpoint
bytearray	It defines the byte array.	
boolean	It defines the boolean type values.	true/false
datetime	It defines the values in datetime order.	1970-01- 01T00:00:00.000 +00:00
biginteger	It defines Java BigInteger values.	5000000000000
bigdecimal	It defines Java BigDecimal values.	52.232344535345

Pig Data Types

Apache Pig supports many data types. A list of Apache Pig Data Types with description and examples are given below.

Type	Description	Example
Int	Signed 32 bit integer	2
Long	Signed 64 bit integer	15L or 15l
Float	32 bit floating point	2.5f or 2.5F
Double	32 bit floating point	1.5 or 1.5e2 or 1.5E2
charArray	Character array	hello javatpoint
byteArray	BLOB(Byte array)	
tuple	Ordered set of fields	(12,43)
bag	Collection of tuples	{(12,43),(54,28)}
map	collection of tuples	[open#apache]

2.7 SQOOP

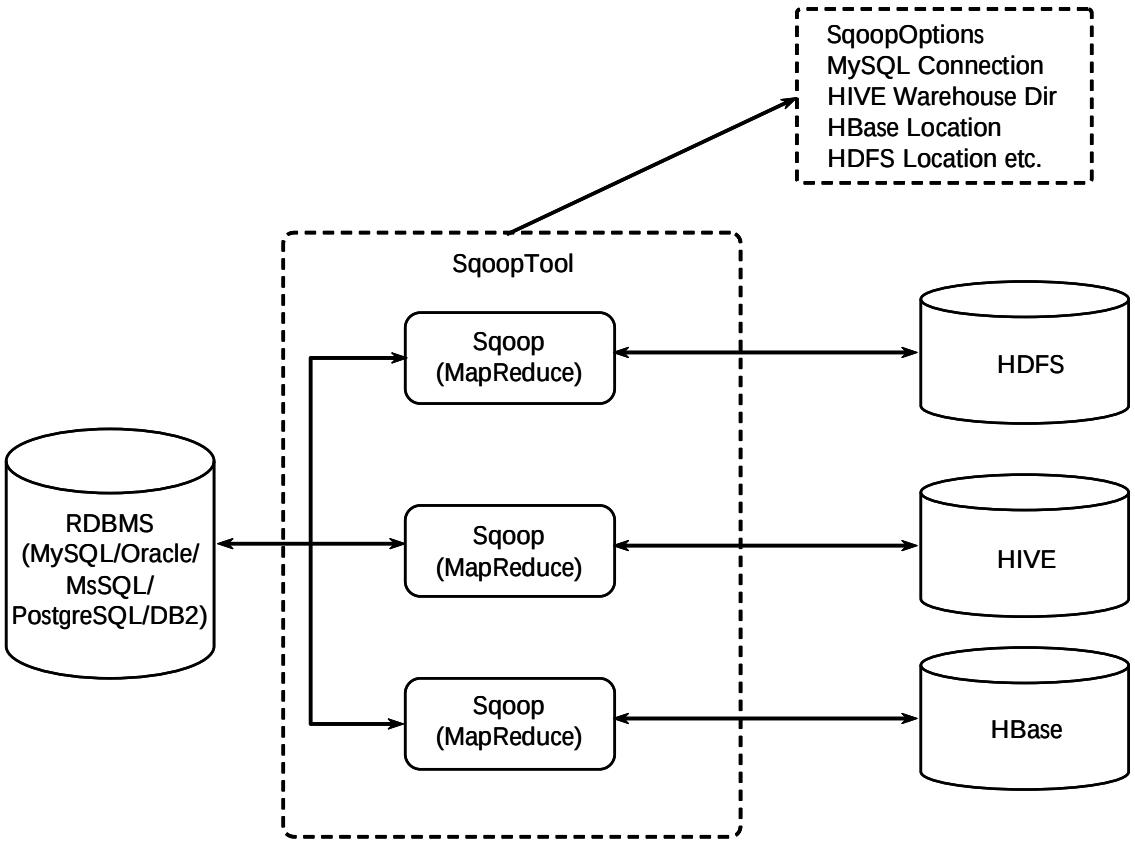
Q24. What is Sqoop ?

Ans :

Meaning

Sqoop is a command-line interface application for transferring data between relational databases and Hadoop.

It supports incremental loads of a single table or a free form SQL query as well as saved jobs which can be run multiple times to import updates made to a database since the last import.Using Sqoop, Data can be moved into HDFS/hive/hbase from MySQL/ PostgreSQL/Oracle/SQL Server/DB2 and vise versa.



Sqoop Working

Step 1: Sqoop send the request to Relational DB to send the return the metadata information about the table (Metadata here is the data about the table in relational DB).

Step 2: From the received information it will generate the java classes (Reason why you should have Java configured before get it working-Sqoop internally uses JDBC API to generate data).

Step 3: Now Sqoop (As its written in java ?tries to package the compiled classes to beable to generate table structure) , post compiling creates jar file (Java packaging standard).

2.8 ZOOKEEPER

Q25. Explain about Zookeeper ?

Ans :

ZooKeeper is a distributed co-ordination service to manage large set of hosts. Co-ordinating and managing a service in a distributed environment is a complicated process. ZooKeeper solves this issue with its simple architecture and API. ZooKeeper allows developers to focus on core application logic without worrying about the distributed nature of the application.

The ZooKeeper framework was originally built at "Yahoo!" for accessing their applications in an easy and robust manner. Later, Apache ZooKeeper became a standard for organized service used by Hadoop, HBase, and other distributed frameworks. For example, Apache HBase uses ZooKeeper to track the status of distributed data.

Benefits

Here are the benefits of using ZooKeeper :

- **Simple distributed coordination process**
- **Synchronization**

  Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.

- **Ordered Messages**
- **Serialization**

  Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.

- **Reliability**
- **Atomicity**

  Data transfer either succeed or fail completely, but no transaction is partial.

2.9 FLUME

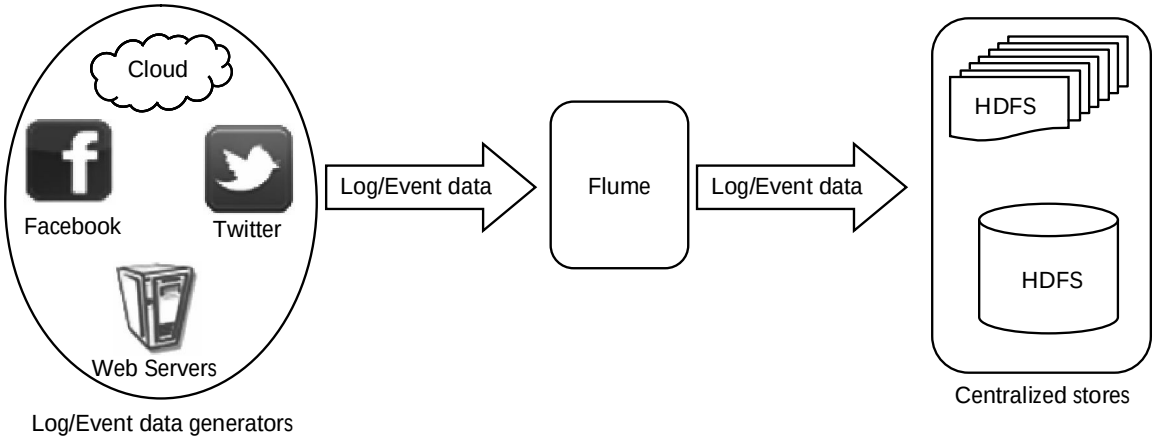
Q26. Explain about Flume.

Ans :

Meaning

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a centralized data store.

Flume is a highly reliable, distributed, and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.



Applications

Assume an e-commerce web application wants to analyze the customer behavior from a particular region. To do so, they would need to move the available log data in to Hadoop for analysis. Here, Apache Flume comes to our rescue.

Flume is used to move the log data generated by application servers into HDFS at a higher speed.

Advantages

Here are the advantages of using Flume

- Using Apache Flume we can store the data in to any of the centralized stores (HBase, HDFS).
- When the rate of incoming data exceeds the rate at which data can be written to the destination, Flume acts as a mediator between data producers and the centralized stores and provides a steady flow of data between them.
- Flume provides the feature of contextual routing.
- The transactions in Flume are channel-based where two transactions (one sender and one receiver) are maintained for each message. It guarantees reliable message delivery.
- Flume is reliable, fault tolerant, scalable, manageable, and customizable.

Features

Some of the notable features of Flume are as follows :

- Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.
- Using Flume, we can get the data from multiple servers immediately into Hadoop.
- Along with the log files, Flume is also used to import huge volumes of event data produced by social networking sites like Facebook and Twitter, and e-commerce websites like Amazon and Flipkart.
- Flume supports a large set of sources and destinations types.
- Flume supports multi-hop flows, fan-in fan-out flows, contextual routing, etc.
- Flume can be scaled horizontally.

2.10 OOZIE

Q27. Define OOZIE. State the components of OOZIE.

Ans : **(Imp.)**

Meaning

Apachel Oozie is a Java Web application that runs in a Java servlet container (i.e. Tomcat). Oozie is a server-based workflow engine and is used to schedule Apache Hadoop jobs.

Depending on the version, Oozie performs different functions as an engine. The three types of Oozie engines are as follows:

- A server-based workflow engine (Oozie vl)
- A server-based coordinator engine (Oozie v2)
- A server-based bundle engine (Oozie v3)

Oozie can store and run various Hadoop jobs like hive, pig, and so on. This essentially means it has a database to store workflow definitions and states as well as variables of any instances of a running workflow.

Oozie workflow is a set of actions specified by the control dependency DAG (Directed Acyclic Graph). This graph is specified in an XML Process Definition Language (xPDL).

Oozie performs the following jobs

- **Oozie workflow jobs**

  Workflow jobs are nodes in DAG. In other words, each node in DAG represents a job and the edges of the DAG represent jobs dependencies.

- **Oozie coordinator jobs**

  These are recurrent Oozie workflow jobs that are triggered by availability of dependent data.

  Oozie is distributed under the Apache license.

Main Functional Components

Oozie is a workflow scheduler and coordination system to manage Apache-Hadoop jobs. Oozie is only responsible for triggering (initiating) a task, while the execution of the task is done by Hadoop MapReduce. Oozie has four functional components:

- **Oozie workflow**

  This component is used to specify the sequence of actions (or jobs) to be executed in the form of DAG.

- **Oozie coordinator**

  This component automatically executes Oozie workflows on the basis of specified events and available system resources.

- **Oozie bundles**

  This component 7defines and executes a bundle of applications with a coordinator application.

- **Oozie Service-Level Agreement (SLA)**

  This component provides a way of tracking the whole execution process of workflows.

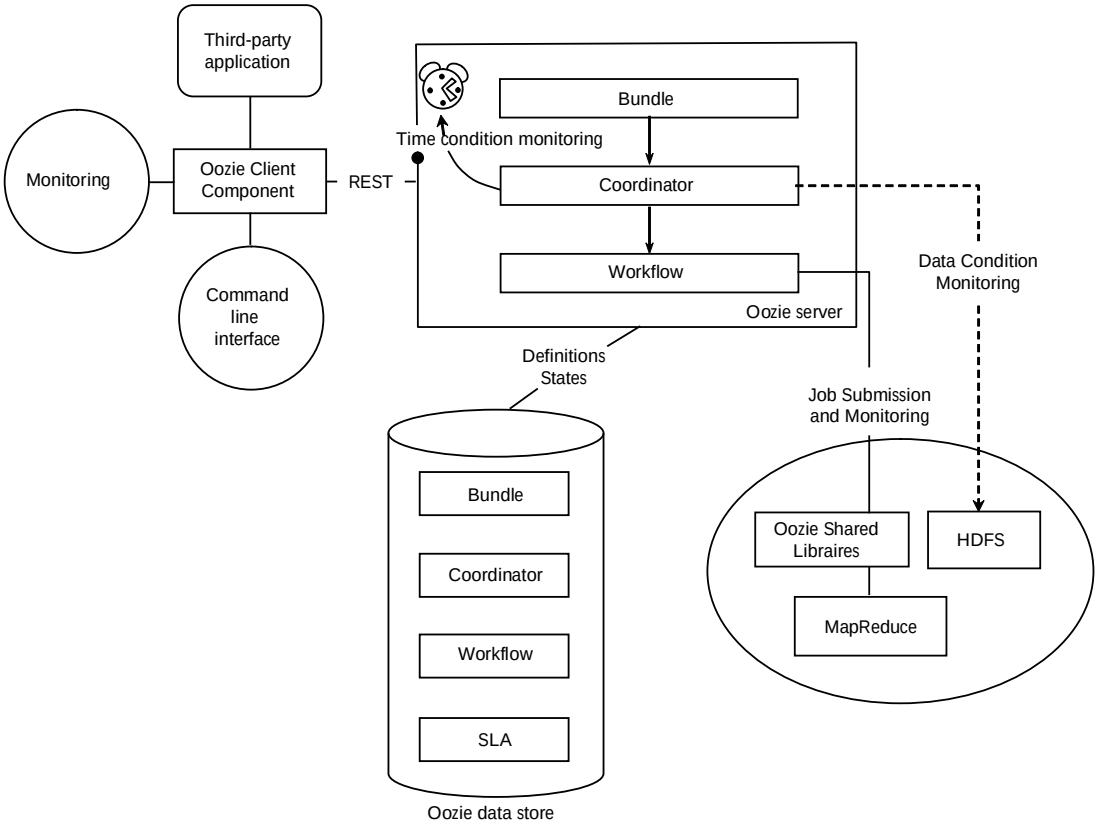


Fig.: Components of Oozie

Q28. State the benefits of OOZIE.

Ans : **(Imp.)**

The benefits of Oozie are as follows:

i) **Complex workflow action dependencies:** The Oozie workflow contains actions and dependencies. Users create DAG to model their workflow. At runtime, Oozie manages dependencies and executes actions when the dependencies identified in the DAG are satisfied,

ii) **Reduces Time-To-Market (TTM):** The DAG specification enables users to specify the workflow. Users save on the time to build and maintain custom solutions for dependency and workflow management.

iii) **Frequency execution:** Oozie workflow specification supports both data and time triggers. User-can specify execution frequency and wait for data arrival to trigger an action in the workflow.

iv) **Native Hadoop stack integration:** Oozie supports all types of Hadoop jobs and is integrated with the Hadoop stack. Yahoo! distribution of Oozie is validated against the Hadoop stack.

v) **Mechanism to manage a variety of complex data analysis:** Oozie was designed for Yahoos complex workflows and data pipelines at a global scale. It is integrated with the Yahoo! distribution of Hadoop with security and is a primary mechanism to manage a variety of complex data analysis workloads across Yahoo.

UNIT III

Understanding MapReduce Fundamentals and HBase: The MapReduce Framework ,Exploring the features of MapReduce, Working of MapReduce, Techniques to optimize MapReduce Jobs, Hardware/Network Topology, Synchronization, File system, Uses of MapReduce, Role of HBase in Big Data Processing- Characteristics of HBase.

Understanding Big Data Technology Foundations: Exploring the Big Data Stack, Data Sources Layer, Ingestion Layer, Storage Layer, Physical Infrastructure Layer, Platform Management Layer, Security Layer, Monitoring Layer, Visualization Layer.

3.1 THE MAPREDUCE FRAMEWORK

Q1. Explain framework of MapReduce.

Ans :

At the start of the 21st century, the team of engineers working with Google concluded that because of the increasing number of Internet users, the resources and solutions available would be inadequate to fulfill the future requirements. As a preparation to the upcoming issue, Google engineers established that the concept of task distribution across economical resources, and their interconnectivity as a cluster over the network, can be presented as a solution. The concept of task distribution, though, could not be a complete answer to the issue, which requires the tasks to be distributed in parallel. A parallel distribution of tasks:

- Helps in automatic expansion and contraction of processes.
- Enables continuation of processes without being affected by network failures or individual system failures.
- Empowers developers with rights to access the services that other developers have created in context of multiple usage scenarios.

In addition to the task distribution concept, location independence in terms of data and applications working on the data was also a necessity. A generic implementation to the entire concept was, therefore, provided with the development of the MapReduce programming model. The name, MapReduce, is effectively and practically a combination of the two capabilities of the existing computer languages. These capabilities are map and reduce. The MapReduce implementations have evolved over the years to be available both as commercial products and open-source models.

3.1.1 Exploring the features of MapReduce

Q2. List the features of MapReduce.

Ans : **(Imp.)**

The principal features of MapReduce include the following:

1. Scheduling

MapReduce involves two operations: map and reduce, which are executed by dividing large problems into smaller chunks. These chunks are run in parallel by different computing resources. The operation

of breaking tasks into subtasks and running these subtasks independently in parallel is called mapping, which is performed ahead of the reduce operation. The mapping operation requires task prioritization based on the number of nodes in the cluster. In case nodes are fewer than tasks, then tasks are executed on a priority basis. The reduction operation cannot be performed until the entire mapping operation is completed. The reduction operation then merges independent results on the basis of priority. Hence, the MapReduce programming model requires scheduling of tasks.

2. Synchronization

Execution of several concurrent processes requires synchronization. The MapReduce program execution framework is aware of the mapping and reducing operations that are taking place in the program. The framework tracks all the tasks along with their timings and starts the reduction process after the completion of mapping. A method known as shuffle and sort produces the intermediate data, which is transmitted across the network. The shuffle and sort mechanism is used for collecting the mapped data and preparing it for reduction.

3. Co-location of Code/Data (Data Locality)

The effectiveness of a data processing mechanism depends largely on the location of the code and the data required for the code to execute. The best result is obtained when both code and data reside on the same machine. This means that the co-location of the code and data produces the most effective processing outcome.

4. Handling of Errors/Faults

MapReduce engines usually provide a high level of fault tolerance and robustness in handling errors. The reason for providing robustness to these engines is their high propensity to make errors or faults. There are high chances of failure in clustered nodes on which different parts of a program are running. Therefore, the MapReduce engine must have the capability of recognizing the fault and rectify it. Moreover, the MapReduce engine design involves the ability to find out the tasks that are incomplete and eventually assign them to different nodes.

5. Scale-Out Architecture

MapReduce engines are built in such a way that they can accommodate more machines, as and when required. This possibility of introducing more computing resources to the architecture makes the MapReduce programming model more suited to the higher computational demands of Big Data.

3.1.2 Working of MapReduce

Q3. Describe the working of the MapReduce Algorithm.

Ans : **(Imp.)**

The data analysis operations must not destroy the data in any case, because the input data is not accessible to the user all the time. Any changes in the input data might disable the user from using the same dataset for other computations or purposes.

Applications to handle data are designed by software professionals on the basis of algorithms, which are stepwise processes to solve a problem or achieve a goal. The MapReduce programming model also works on an algorithm to execute the map and reduce operations. This algorithm can be depicted as follows:

1. Take a large dataset or set of records.
2. Perform iteration over the data.
3. Extract some interesting patterns to prepare an output list by using the map function.
4. Arrange the output list properly to enable optimization for further processing.
5. Compute a set of results by using the reduce function.
6. Provide the final output.

The MapReduce approach of analyzing data can be used by programmers for implementing various kinds of applications. This algorithm can also work with extremely large datasets, ranging from hundreds of gigabytes to few terabytes, or even beyond.

To sum up the things we have learned till now, the MapReduce model executes a given task by dividing it into two functions, map and reduce. The map function is executed first in parallel on different machines. The reduce function takes the output of the map function to present the final output in an aggregate form. The working of the MapReduce approach is shown in Figure :

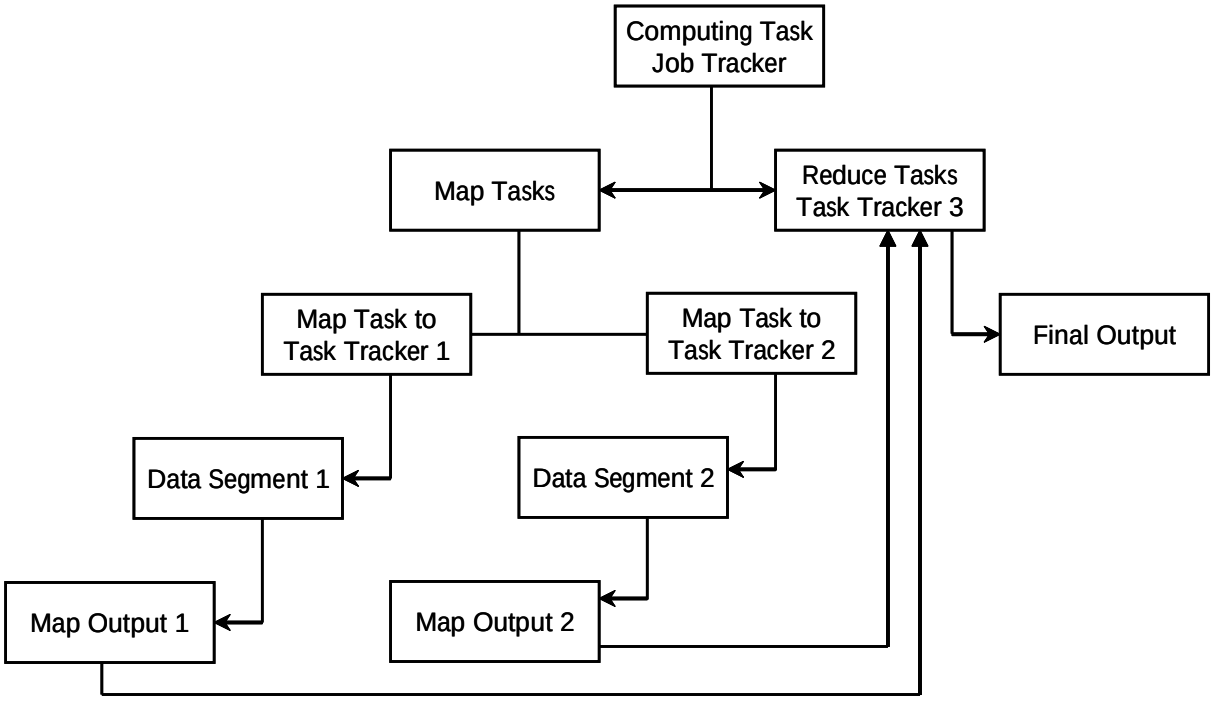


Fig. : Working of the MapReduce Approach

The MapReduce framework, shown in Figure, is a combination of a master and three slaves. The master monitors the entire job assigned to the MapReduce algorithm and is given the name of JobTracker. Slaves, on the other hand, are responsible for keeping track of individual tasks and are called TaskTrackers. First, the given job is divided into a number of tasks by the master, i.e., the JobTracker, which then distributes these tasks into slaves. It is the responsibility of the JobTracker to further keep an eye on the processing activities and the re-execution of the failed tasks. Slaves coordinate with the master by executing the tasks they are given by the master.

The JobTracker receives jobs from client applications to process large information. These jobs are assigned in the forms of individual tasks (after a job is divided into smaller parts) to various TaskTrackers. The task distribution operation is completed by the JobTracker. The data after being processed by TaskTrackers is transmitted to the reduce function so that the final, integrated output, which is an aggregate of the data processed by the map function, can be provided.

A cluster uses commodity servers to store nodes. The data processing job is accomplished through MapReduce and Hadoop Distributed File System (HDFS), which are based on these nodes.

The logical flow of data in the MapReduce programming framework is shown in Figure 5.2:

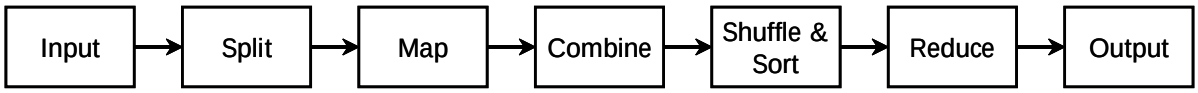


Fig. 5.2 : Logical Flow of Data in MapReduce

Operations performed in the MapReduce model, according to the data flow (as shown in Figure) are as follows:

- The input is provided from large data files in the form of key-value pair (KVP), which is the standard input format in a Hadoop MapReduce programming model.
- The input data is divided into small pieces, and master and slave nodes are created. The master node usually executes on the machine where the data is present, and slaves are made to work remotely on the data.
- The map operation is performed simultaneously on all the data pieces, which are read by the map function. The map function extracts the relevant data and generates the KVP for it.

The input/output operations of the map function are shown in Figure:

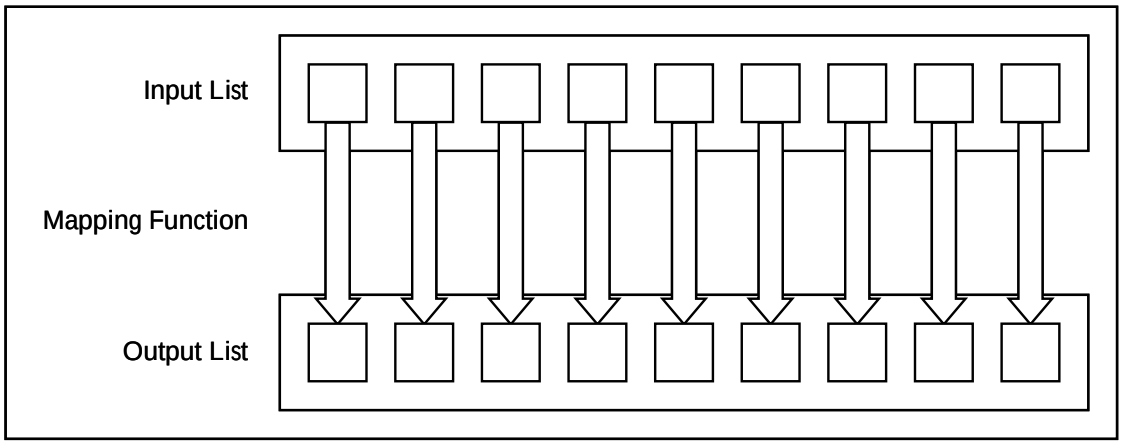


Fig. : Working of the Map Function in the MapReduce Model

The output list is generated from the map operation, and the master instructs the reduce function about further actions that it needs to take. The list of KVPs obtained from the map function is passed on to the reduce function.

The reduce function sorts the data on the basis of the KVP list. The process of collecting the map output list from the map function and then sorting it as per the keys is known as shuffling.

Every unique key is then taken by the reduce function. These keys are called, as required, for producing the final output to be sent to the file. The input/output operations of the reduce function are shown in Figure :

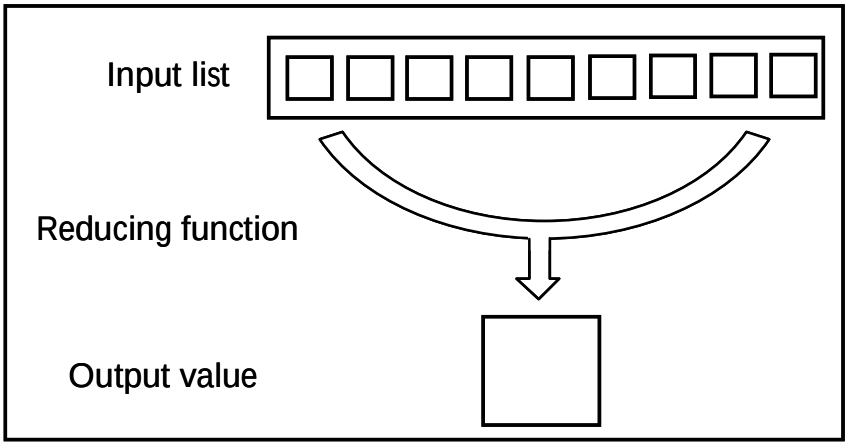


Fig. : Working of the Reduce Function in the MapReduce Model

The output is finally generated by the reduce function, and the control is handed over to the user program by the master.

Figure shows the entire process of data analysis conducted in the MapReduce programming model.

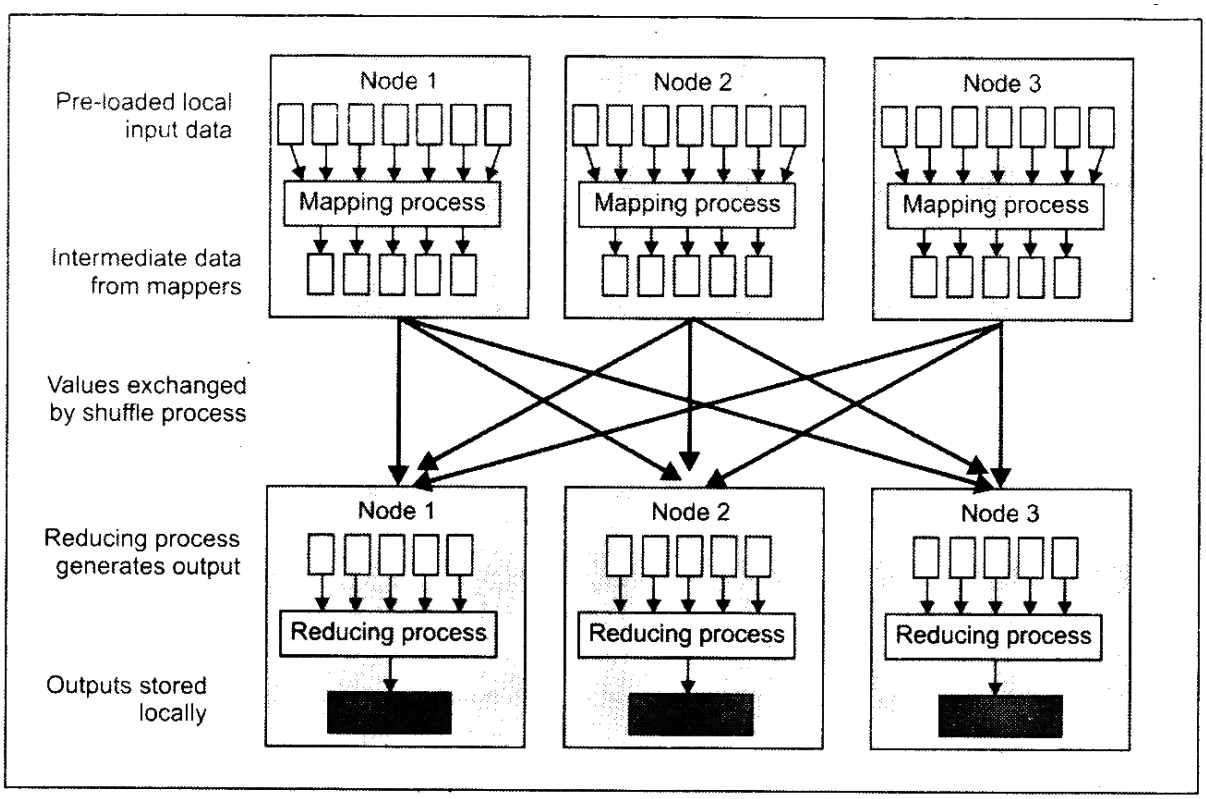


Fig. : Data Analysis in the MapReduce Programming Model

3.1.3 Techniques to optimize MapReduce Jobs

3.1.3.1 Hardware / Network Topology, Synchronization, File system

Q4. Explain various Techniques to optimize MapReduce Jobs.

Ans : **(Imp.)**

An analysis of the MapReduce program execution shows that it involves a series of steps in which every step has its own set of resource requirements. In addition, you must avoid any bottlenecks of resources in order to draw the maximum benefits from the MapReduce resources. These resources, if utilized to the fullest, can help you to reduce the response time of your MapReduce job to a minimum level.

In case of short jobs, especially the ones where the user requires a quick response to his/her query, the response time becomes more important. Hence, we need the MapReduce job to be optimized. Encountering a deadlock for even a single resource, during the execution of the program, slows down the execution process.

The performance of MapReduce jobs and their reliability, in addition to the code written for the main application, can be optimized by using some techniques. We can organize the MapReduce optimization techniques in the following categories:

1. Hardware or network topology
2. Synchronization
3. File system

1. Hardware/Network Topology

MapReduce makes it possible for hardware to run the MapReduce tasks on inexpensive clusters of commodity computers. These computers can be connected through standard networks. The performance and fault tolerance required for Big Data operations are also influenced by the physical location of servers. Usually, the data center arranges the hardware in racks. The performance offered by hardware systems that are located in the same rack where the data is stored will be higher than the performance of hardware systems that are located in a different rack than the one containing the data. The reason for the low performance of the hardware that is located away from the data is the requirement to move the data and/or application code.

When the program is being executed, we can configure the MapReduce engine to exploit the advantages of its closeness to the data. The best way to optimize the performance of your MapReduce engine is to keep the application code and data together. You can, thus, minimize the latency in MapReduce processing by keeping the hardware elements close to each other.

2. Synchronization

The completion of map processing enables the reduce function to combine the various outputs for providing the final result. However, the performance will degrade if the results of mapping are contained within the same nodes where the data processing began. In order to improve the

performance, we should copy the results from mapping nodes to the reducing nodes, which will start their processing tasks immediately. In addition, the hashing function sends all the values marked with the same key to the same reducer node. Thus, the overall performance of the system enhances with an increased efficiency. You can extract the best possible outcomes by writing the outputs of reduction operations directly to the file system. This entire process is synchronized to provide efficiency to the MapReduce programming model.

3. File System

A distributed file system is used to support the implementation of the MapReduce operation. Distributed file systems are different from local file systems in a manner that local file systems have less capability of storing and arranging data. The Big Data world has immense information to be processed and requires data to be distributed among a number of systems or nodes in a cluster so that the data can be handled efficiently. The distribution model followed in implementing the MapReduce programming approach is to use the master and slaves model. All the metadata and access rights, apart from the mapping, block, and file locations are stored with the master. On the other hand, the data on which the application code will run is kept with the slaves. The master node receives all the requests, which are forwarded to appropriate slaves for performing the required actions.

You need to keep the following points in mind while designing a file that supports MapReduce implementation:

- **Keep it Warm**

  The master handles various operations, which may lead to a stage of overworking for it. In case of the failure of the master node, you will be unable to access the entire file system unless the master becomes active again. In order to optimize the file system, you can develop a standby master node that can remain warm and, whenever the primary master fails to deliver, can take the responsibility.

- **The Bigger the Better**

  In Big Data environment, files with a size less than 100 MB are not preferred, so you need to avoid using them. The best results are obtained when the distributed file systems that are used with MapReduce are loaded with a small number of large-sized files.

- **The Long View**

  MapReduce handles the workload by keeping large jobs in small data batches. Hence, MapReduce needs a network bandwidth that remains available for a long time instead of having quick execution times of mappers and reducers. To use the network optimally, a long stream of data should be sent by the application code when it is reading from or writing to the file system

- **Right Degree of Security**

  The increasing number of security layers hampers the performance of distributed file systems. The permissions associated with file systems are meant to protect files from any unauthorized access and damage. It is always advisable to allow only authorized users to access the data center environment and protect the distributed file system.

3.1.4 Uses of MapReduce

Q5. Discuss the uses of MapReduce.

Ans :

MapReduce is used to process various types of data obtained from various sectors. Some of the fields benefitted by the use of MapReduce are:

- **Web Page Visits:** Suppose a researcher wants to know the number of times the websitp of a particular newspaper was accessed. The map task would be to read the logs of the WeL page requests and make a complete list. The map outputs may look similar to the following:

  <emai1URL, 1>
  <newspaperURL, 1>
  <socialmediaURL, 1>
  <sportsnewsURL, 1>
  <newspaperURL, 1>
  <emai1URL, 1>
  <newspaperURL, 1>

  The reduce function would find the results for the newspaper URL and add them. The output of the preceding code is:

  <newspaperuRL, 3>

- **Web Page Visitor Paths:** Consider a situation in which an advocacy group wishes to know how visitors get to know about its website. To determine this, they designed a link known as "source," and the Web page to which the link transfers the information is known as "target." The map function scans the Web links for returning the results of the type <target, source>. The reduce function scans this list for determining the results where the "target" is the Web page. The reduce function output, which is the final output, will be of the form <advocacy group page, list (source) >.

- **Word Frequency:** A researcher wishes to read articles about flood but, he does not want those articles in which the flood is discussed as a minor topic. Therefore, he decided that an article basically dealing with earthquakes and floods should have the word "tectonic plate" in it more than 10 times. The map function will count the number of times the specified word occurred in each document and provide the result as <document, frequency>. The reduce function will count and select only the results that have a frequency of more than 10 words.

- **Word Count:** Suppose a researcher wishes to determine the number of times celebrities talk about the present bestseller. The data to be analyzed comprises written blogs, posts, and tweets of the celebrities. The map function will make a list of all the words. This list will be in the KVP format, in which the key is each word, and the value is 1 for every appearance of that word.

The output from the map function would be obtained somewhat as follows:

<global warning, 1>

<food, 1>

<global warning, 1>

<bestseller, 1>

<Afgbanistan, 1>

<bestseller, 1>

The preceding output will be converted in the following form by the reduce function:

<global warning, 2>

<food, 1>

<bestseller, 2>

<Afghanistan, 1>

The MapReduce programming model can utilize other components of the Hadoop ecosystem to perform its operations better. One such components is HBase.

3.1.5 Role of HBase in Big Data Processing

Q6. Explain the Role of HBase in Big Data Processing.

Ans :

HBase is an open source, non-relational, distributed database developed as a part of Apache software Foundation's Hadoop project. It is beneficial when large amounts of information is required to be stored, updated, and processed at a fast speed.

Because of the vast size of Big Data, its storage and processing are challenging tasks. While MapReduce enhances Big Data processing, HBase takes care of its storage and access requirements.

HBase helps programmers to store large quantities of data in such a way that it can be accessed easily and quickly, as and when required. It stores data in a compressed format and thus, occupies less memory space.

HBase has low latency time and is, therefore, beneficial for lookups and scanning of large amounts of data. HBase saves data in cells in the descending order (with the help of timestamp); therefore, a read will always first determine the most recent values. Columns in HBase relate to a column family. The column family name is utilized as a prefix for determining members of its family; for instance, Cars: Wagon R and Cars: ilO are the members of the Cars column family.

3.1.6 Characteristics of HBase

Q7. Explain the Characteristics of HBase.

Ans : **(Imp.)**

HBase helps programmers to store large quantities of data in such a way that it can be accessed [0] easily and quickly, as and when required.

- It stores data in a compressed format and thus, occupies less memory space. HBase has low latency time and is, therefore, beneficial for lookups and scanning of large amounts of data.
- HBase saves data in cells in the descending order (with the help of timestamp); therefore, a read will always first determine the most recent values.
- Columns in HBase relate to a column family. The column family name is utilized as a prefix for determining members of its family; for instance, Cars: Wagon R and Cars: ilO are the members of the Cars column family.
- A key is associated with rows in HBase tables. The structure of the key is very flexible. It can be a calculated value, a string, or any other data structure.
- The key is used for controlling the retrieval of data to the cells in the row. All these characteristics help build the schema of the HBase data structure before the storage of any data. Moreover, tables can be modified and new column families can be added once the database is up and running.
- This extensibility is beneficial while dealing with Big Data, because every time you do not always know the diversity of your data streams.
- Relational databases are row-oriented, because the data in each row of a table is saved together. In a columnar or column-oriented database, the data is saved across rows.
- The columns can be added very easily and are added row-by-row, providing great flexibility, performance, and scalability.
- In case you have large volume and variety of data, you can use a columnar database. HBase is suitable in conditions where the data changes gradually and rapidly. In addition, HBase can save the data that has a slow-changing rate and ensure its availability for Hadoop tasks.
- Examples of data that changes at a very slow pace include demographic data, IP addresses, geolocation lookup tables, and product dimensional data.
- Application logs, clickstream data, and in-game usage data are developed rapidly and are used in real-time log ingestion and batch log analytics. HBase can obtain this data and quickly update the database to enable instant processing of the updated data.

3.2 Understanding Big Data Technology Foundations

3.2.1 Exploring the Big Data Stack

Q8. Discuss the various layers in big data architecture.

Ans : **(Imp.)**

The first step in the process of designing any data architecture is to create a model that should give a 'complete view of all the required elements. Although, initially, creating a model may seem to be a time-consuming task; however, it can save a significant amount of time, effort, and rework during the subsequent implementations.

Big Data analysis also needs the creation of a model or architecture, commonly known as the Big Data architecture. Here, you must note that the configuration of the model may vary depending upon the specific needs of the organization; however, the basic layers and components, more or less, remain the same. To create a Big Data architecture model, you need to think of Big Data as a strategy and not a project. This strategy also includes certain design principles related to the creation of an environment to support Big Data. Basically, these principles deal with the storage of data, analytics, reporting, or applications. However, while creating Big Data environment, we must also take hardware, infrastructure software, operational software, management software, Application Programming Interfaces (APIs), and software developer tools into consideration. In short, we can say that the architecture of the Big Data environment must fulfill all the foundational requirements and must be able to perform the following functions:

- Capturing data from different sources
- Cleaning and integrating data of different types of formats
- Sorting and organizing data
- Analyzing data
- Identifying relationships and patterns
- Deriving conclusions based on the data analysis results

Figure shows the arrangement of various layers in the Big Data architecture:

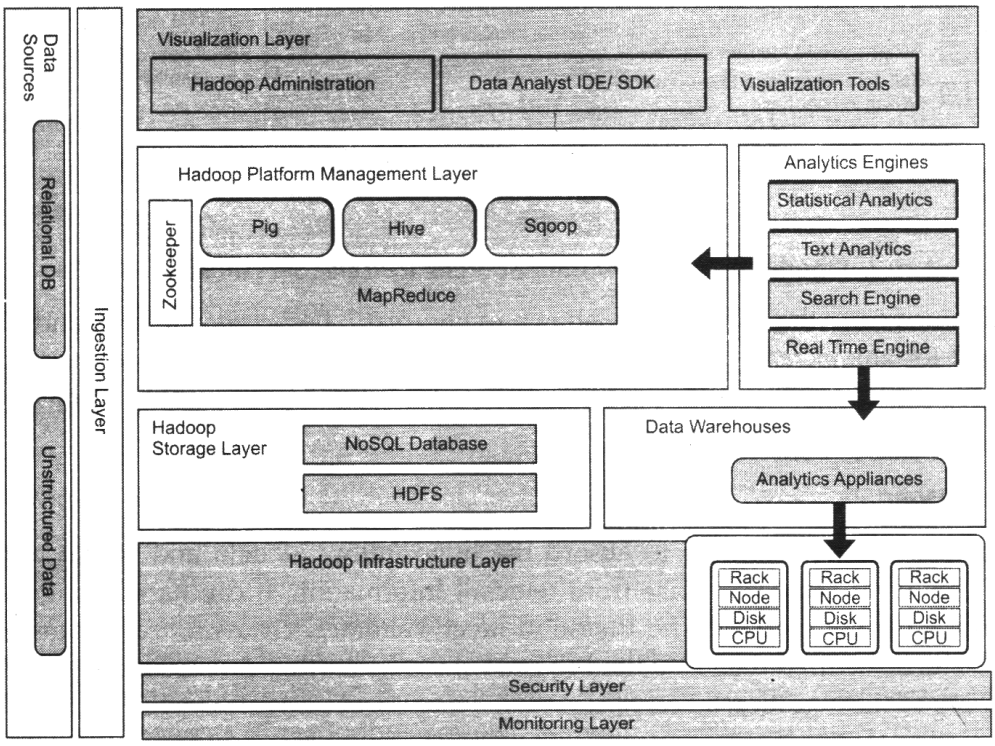


Fig. : Stack of Layers in Big Data Architecture

Figure shows a sample diagram of the Big Data architecture, comprising the following basic layers and components:

- Data Sources layer
- Ingestion layer
- Storage layer
- Physical Infrastructure layer
- Platform Management layer
- Security layer
- Monitoring layer
- Analytics engine
- Visualization layer.

3.2.1.1 Data Sources Layer

Q9. Explain the functioning of Data Sources Layer in the big data architecture.

Ans :

Organizations generate a huge amount of data on a daily basis. The basic function of the data sources layer is to absorb and integrate the data coming from various sources, at varying velocity and in different formats. Before this data is considered for big datastack, we have to differentiate between the noise and relevant information.

Figure shows different data sources from where the telecom industry obtains its data:

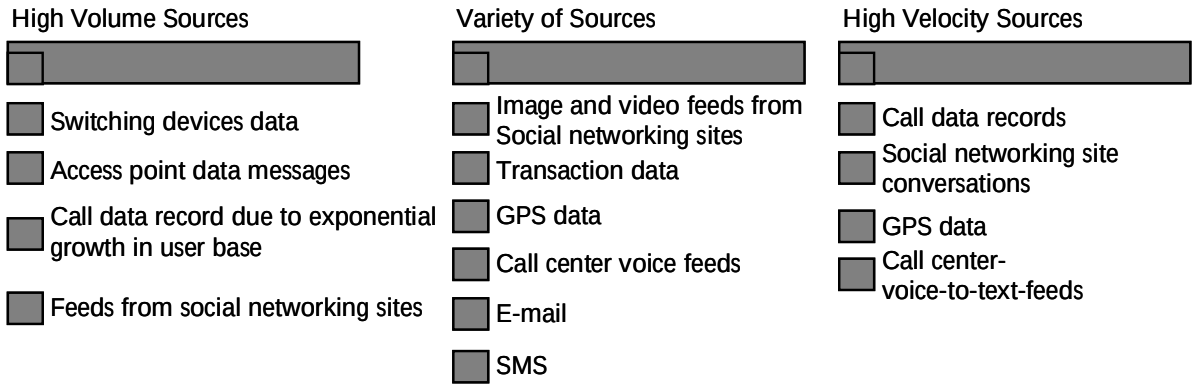


Fig.: Variety of Data Sources for Telecom Industry

The data obtained from the data sources, shown in Figure, has to be validated and cleaned before introducing it for any logical use in the enterprise. The task of validating, sorting, and cleaning data is done by the ingestion layer. The removal of noise from the data also takes place in the ingestion layer.

3.2.1.2 Ingestion Layer

Q10. Explain the functioning of Ingestion Layer in the big data architecture.

Ans : **(Imp.)**

The role of the ingestion layer is to absorb the huge inflow of data and sort it out in different categories. This layer separates noise from relevant information. It can handle huge volume, high velocity, and a variety of data. The ingestion layer validates, cleanses, transforms, reduces, and integrates the unstructured data into the Big Data stack for further processing. Figure illustrates the functioning of the ingestion layer:

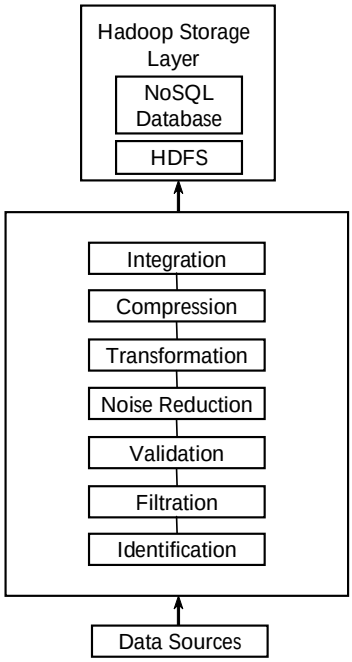


Fig. 6.3: Functioning of the Ingestion Layer

In the ingestion layer, the data passes through the following stages:

- **Identification:** At this stage, data is categorized into various known data formats, or we can say that unstructured data is assigned with default formats.
- **Filtration:** At this stage, the information relevant for the enterprise is filtered on the basis of the Enterprise Master Data Management (MDM) repository.
- **Validation:** At this stage, the filtered data is analyzed against MDM metadata.
- **Noise reduction:** At this stage, data is cleaned by removing the noise and minimizing the related disturbances.
- **Transformation:** At this stage, data is split or combined on the basis of its type, contents, and the requirements of the organization.
- **Compression:** At this stage, the size of the data is reduced without affecting its relevance for the required process. It should be noted that compression does not affect the analysis results.
- **Integration:** At this stage, the refined dataset is integrated with the Hadoop storage layer, which consists of Hadoop Distributed File System (HDFS) and NoSQL databases.

3.2.1.3 Storage Layer

Q11. Explain the functioning of Storage Layer in the big data architecture.

Ans :

Hadoop is an open source framework used to store large volumes of data in a distributed manner across multiple machines. The Hadoop storage layer supports fault-tolerance and parallelization, which enable high-speed distributed processing algorithms to execute over large-scale data. There are two major components of Hadoop: a scalable Hadoop Distributed File System (HDFS) that can support petabytes of data and a MapReduce engine that computes results in batches.

HDFS is a file system that is used to store huge volumes of data across a large number of commodity machines in a cluster. The data can be in terabytes or petabytes. HDFS stores data in the form of blocks of files and follows the write-once-read-many model to access data from these blocks of files. The files stored in the HDFS are operated upon by many complex programs, as per the requirement.

Consider an example of a hospital that used to perform a periodic review of the data obtained from the sensors and machines attached to the patients. This review helped doctors to keep a check on the condition of terminal patients as well as analyze the effects of various medicines on them. With time, the growing volume of data made it difficult for the hospital staff to store and handle it. To find a solution, the hospital consulted a data analyst who suggested the implementation of HDFS as an answer to this problem. HDFS can be implemented in an organization at comparatively low costs than other advanced technologies and can easily handle the continuous streaming of data.

Earlier, we needed to have different types of databases, such as relational and non-relational, for storing different types of data. However, now, all these types of data storage requirements can be addressed by a single concept known as Not Only SQL (NoSQL) databases. Some examples of NoSQL databases include HBASE, MongoDB, AllegroGraph, and InfiniteGraph.

Different NoSQL databases used for different types of business applications are shown in Figure.

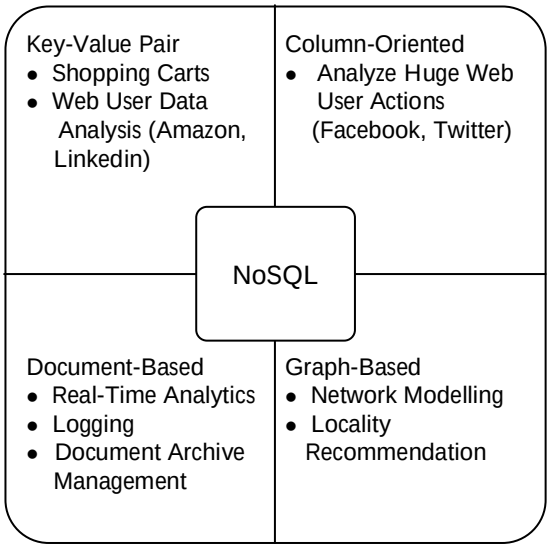


Fig. : Different NoSQL Databases for Different Business Applications

Figure shows that different types of NoSQL databases, such as graph-based, document-based, key-value pair, and column-oriented, can be used to store different types of data.

3.2.1.4 Physical Infrastructure Layer

Q12. Explain briefly about Physical Infrastructure Layer in big data architecture.

Ans : **(Imp.)**

Before learning about the physical infrastructure layer, you need to know about the principles on which Big Data implementation is based. Some of these principles are:

- **Performance**

  High-end infrastructure is required to deliver high performance with low latency. Performance is measured end to end, on the basis of a single transaction or query request. It would be rated high if the total time taken in traversing a query request is low. The total time taken by a data packet to travel from one node to another is described as latency. Generally, the setups that provide high performance and low latency are quite expensive than normal infrastructure setups.

- **Availability**

  The infrastructure setup must be available at all times to ensure nearly a 100 percent uptime guarantee of service. It is obvious that businesses cannot wait in case of a service interruption or failure; therefore, an alternative of the main system must also be maintained.

- **Scalability**

  The Big Data infrastructure should be scalable enough to accommodate varying storage and computing requirements. They must also be capable to deal with any unexpected challenges.

- **Flexibility**

  Flexible infrastructures facilitate adding more resources to the setup and promote failure recovery. It should be noted that flexible infrastructure is also costly; however, costs can be controlled with the use of cloud services, where you need to pay for what you actually use.

- **Cost**

  You must select the infrastructure that you can afford. This includes all the hardware, networking, and storage requirements. You must consider all the above parameters in the context of your overall budget and then make trade-offs, where necessary.

From the above points, it can be concluded that a robust and inexpensive physical infrastructure can be implemented to handle Big Data. This requirement is addressed by the Hadoop physical infrastructure layer. This layer is based on a distributed computing model, which allows the physical storage of data in many different locations by linking them through networks and the distributed file system. The Hadoop physical infrastructure layer also supports redundancy of data, because data is collected from so many different sources.

Figure shows the hardware topology used for Big Data implementation:

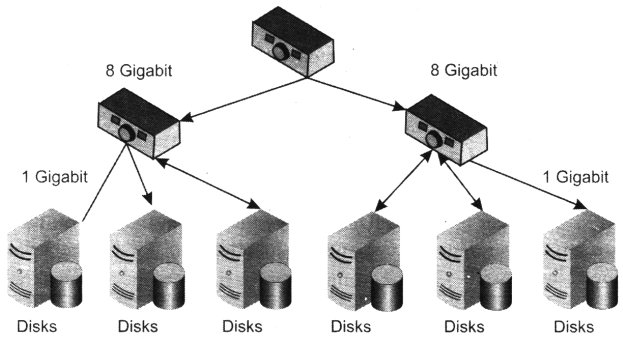


Fig.: Hardware Topology Used for Big Data Implementation

Hadoop infrastructure layer takes care of the hardware and network requirements. It can provide a virtualized cloud environment or a distributed grid of commodity servers over a fast gigabit network. Following are the main components of a Hadoop infrastructure:

- N commodity servers (8-core, 24 GBs RAM, 4 to 12 TBs, gig-E)
- 2-level network (20 to 40 nodes per rack)

3.2.1.5 Platform Management Layer

Q13. Describe the key building blocks of Hadoop Platform Management Layer.

Ans :

The role of the platform management layer is to provide tools and query languages for accessing NoSQL databases. This layer uses the HDFS storage file system that lies on the top of the Hadoop physical infrastructure layer. Figure shows the interaction of the Hadoop platform management layer with its lower layers:

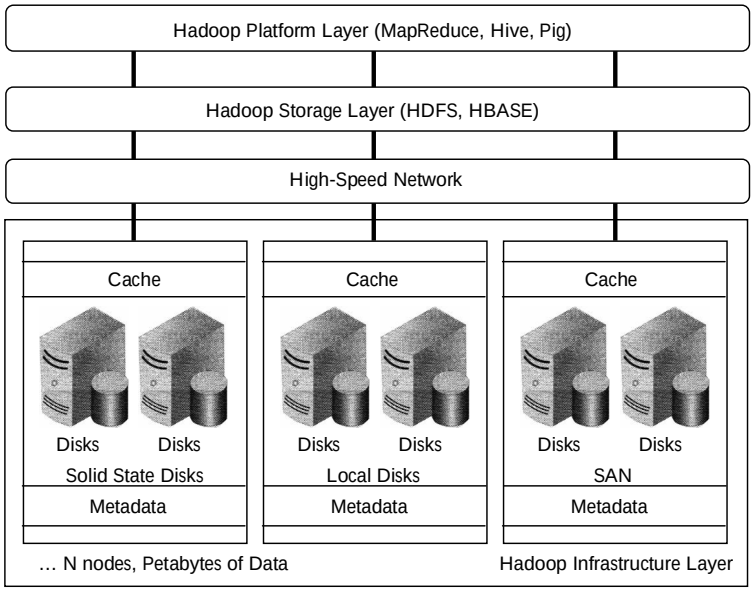


Fig. : Hadoop Platform Management Layer with Other Layers

There is a new technology called Hadoop; it consists of two core components: HDFS and Map-Reduce, and different types of tools that work on Hadoop framework to store, access, and analyze large amounts of data by using real-time analysis. These technologies handle the most fundamental problem of processing huge amounts of data timely, efficiently, and cost-effectively.

The following are the key building blocks of the Hadoop platform management layer:

- **MapReduce:** Refers to a technology that simplifies the creation of processes for analyzing huge amounts of unstructured and structured data. It is a combination of map and reduce features. Map is a component that distributes a problem (of multiple tasks) across a large number of systems and also handles the task of distributing the load for recovery management against failure. When the task of distributed computation is completed, the reduce function combines all the elements back together to provide an aggregate result.
- **Hive:** Refers to a data warehousing package built over Hadoop architecture. Hive provides SQL type query language, called Hive Query Language (HQL) for querying data stored in a Hadoop cluster.
- **Pig:** Refers to a scripting language that is used for batch processing huge amounts of data and allows us to process the data in HDFS in parallel. Pig is not suitable to perform queries on a small portion of a dataset because it scans the entire dataset in one go.
- **HBase:** Refers to a column-oriented database that provides fast access for handling Big Data. It is Hadoop compliant and suitable for batch processing.
- **Sqoop:** Refers to a command-line tool that can import individual tables, specific columns, or entire database files directly in the distributed file system or data warehouse.
- **ZooKeeper:** Refers to a coordinator that keeps multiple Hadoop instances and nodes in synchronization and provides protection to every node from failing because of the overload of data.

3.2.1.6 Security Layer

Q14. Explain briefly about Security Layer.

Ans : **(Imp.)**

The security layer handles the basic security principles that Big Data architecture should follow. Big Data projects are full of security issues because of using the distributed architecture, a simple programming model, and the open framework of services. Therefore, the following security checks must be considered while designing a Big Data stack:

- It must authenticate nodes by using protocols, such as Kerberos.
- It must enable file-layer encryption.
- It must subscribe a key management service for trusted keys and certificates.
- It must use tools such as Chef or Puppet for validating data during the deployment of datasets or while applying service patches on virtual nodes.
- It must maintain logs of the communication that occurs between nodes and trace any anomalies across layers by using distributed logging mechanisms.
- It must ensure a secure communication between nodes by using the Secure Sockets Layer (SSLV).

3.2.1.7 Monitoring Layer

Q15. Explain briefly about Monitoring Layer.

Ans :

The monitoring layer consists of a number of monitoring systems. These systems remain automatically aware of all the configurations and functions of different operating systems and hardware. They also provide the facility of machine communication with the help of a monitoring tool through high-level protocols, such as Extensible Markup Language (XML). Monitoring systems also provide tools for data storage and visualization. Some examples of open source tools for monitoring Big Data stacks are Ganglia and Nagios.

3.2.1.8 Visualization Layer

Q16. Explain briefly about Visualization Layer

Ans :

The visualization layer handles the task of interpreting and visualizing Big Data. Visualization of data is done by data analysts and scientists to have a look at the different aspects of the data in various visual modes. It can be described as viewing a piece of information from different perspectives, interpreting it in different manners, trying to fit it in different types of situations, and deriving different types of conclusions from it.

Figure shows the role of the visualization layer:

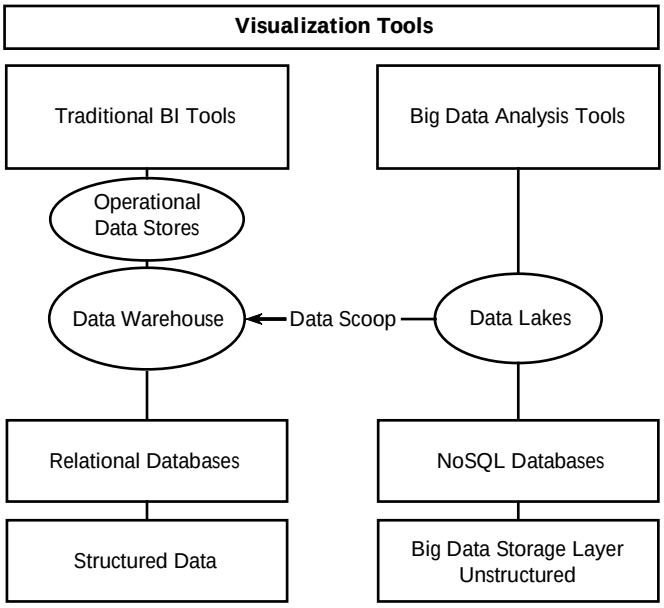


Fig.: Role of the Visualization Layer

The visualization layer works on top of the aggregated data stored in traditional Operational Data Stores (ODS), data warehouse, and data marts. These ODSs get the aggregated data through the data scoop, as shown in Figure. Some examples of visualization tools axe Tableau, Clickview, Spotfire, MapR, and revolution R. These tools work on top of the traditional components such as reports, dashboards, and queries.

UNIT IV

Storing Data in Databases and Data Warehouses: RDBMS and Big Data, Issues with Relational Model, Non – Relational Database, Issues with Non Relational Database, Polyglot Persistence, Integrating Big Data with Traditional Data Warehouse, Big Data Analysis and Data Warehouse.

4.1 Storing Data in Databases and Data Warehouses

Q1. Define Database state the component of data base.

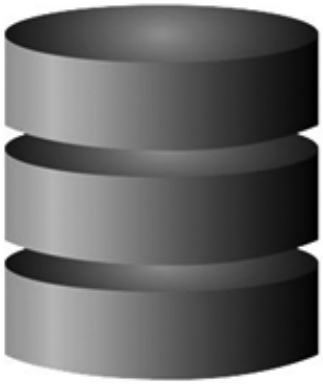
Ans :

Database

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. **Data** is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information.

A database is a collection of **information** that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.



Alternatively referred to as a databank or a datastore, and sometimes abbreviated as a DB, a database is a large quantity of indexed digital information. It can be searched, referenced, compared, changed or otherwise manipulated with optimal speed and minimal processing expense.

A database is built and maintained by using a database programming language. The most common database language is SQL, but there are multiple "flavors" of SQL, depending on the type of database being used. Each flavor of SQL has differences in the SQL syntax and are designed to be used with a specific type of database. For example, an Oracle database uses PL/SQL and Oracle SQL (Oracle's version of SQL). A Microsoft database uses T-SQL (Transact-SQL).

Components

A database is made up of several main components.

- **Schema:** A database contains one or more **schemes**, which is basically a collection of one or more **tables** of data.
- **Table:** Each table contains multiple columns, which are similar to columns in a spreadsheet. A table can have as little as two columns and as many as one hundred or more columns, depending on the type of data being stored in the table.
- **Column:** Each column contains one of several types of data or values, like dates, numeric or integer values, and alphanumeric values (also known as varchar).
- **Row:** Data in a table is listed in rows, which are like rows of data in a spreadsheet. Often there are hundreds or thousands of rows of data in a table.

Q2. What is a data warehouse? How does it differ from a database?

Ans : **(Imp.)**

A data warehouse is a type of contemporary database system designed to fulfil decision-support needs. However, a data warehouse differs from a conventional decision-support database in a number of ways:

- **Volume of data:** A data warehouse is likely to hold far more data than a decision-support database. Volumes of the order of over 400 gigabytes of data are commonplace
- **Diverse data sources:** The data stored in a warehouse is likely to have been extracted from a diverse range of application systems, only some of which may be database systems. These systems are described as data sources
- **Dimensional access:** A warehouse is designed to fulfil a number of distinct ways (dimensions) in which users may wish to retrieve data. This is some times referred to as the need to facilitate ad-hoc query.

A data warehouse as being 'a subject-oriented, integrated, time-variant, and non-volatile collection of data used in support of management decision-making':

- **Subject-oriented:** A data warehouse is structured in terms of the major subject are as of the organisation such as, in the case of a university, students, lecturers and modules, rather than in terms of application areas such as enrolment, payroll and time tabling.
- **Integrated:** A data warehouse provides a data repository which integrates data from different systems with data frequently in different formats. The objective is to provide a unified view of data for users.
- **Time-variant:** A data warehouse explicitly associates time with data. Data in a warehouse is only valid for some point or period in time
- **Non-volatile:** The data in a data warehouse is not updated in real-time. Instead, it is refreshed from data in operational systems on a regular basis. A consequence of this is that the management of data integrity is not a critical issue for data warehouses.

Q3. What are the benefits and goals of a data warehouse?

Ans :

Benefits

A data warehouse is seen to deliver three major benefits for organisations:

- A data warehouse provides a single manageable structure for decision support data
- A data warehouse enables organisational users to run complex queries ondata that traverses a number of business areas.
- A data warehouse enables a number of business intelligence applications such as on-line analytical processing and data mining.

The overall objective for a data warehouse is to increase the productivity and effectiveness of decision-making in organisations. This, in turn, is expected to deliver competitive advantage to organisations.

Challenges of Data Warehousing

Data warehousing projects are large-scale development projects. Typically a data warehousing project may take of the order of three years. Some of the challenges experienced in such projects are indicated below:

- Knowing in advance what the data users require, and determining the ownership and responsibilities in terms of data sources
- Selecting installing and integrating different hardware and software required to set up the warehouse. The large volume of data needed in terms of a data warehouse requires large amounts of disk space. This means that estimation of storage volume is a significant activity.
- Identifying reconciling and cleaning existing production data and loading it into the warehouse. The diverse sources of data feeding a data ware house introduces problems of design in terms of creating a homogeneous data store. Problems are also introduced in terms of the effort required to extract, clean and load data into the warehouse.

Q4. What are characteristics of data warehouse?

Ans :

Data ware-houses have the following distinct characteristics:

- Multidimensional conceptual view.
- Generic dimensionality.
- Unlimited dimensions and aggregation levels.
- Unrestricted cross-dimensional operations.
- Dynamic sparse matrix handling.
- Client/server architecture.

- Multi-user support.
- Accessibility.
- Transparency.
- Intuitive data manipulation.
- Consistent reporting performance.
- Flexible reporting.

Q5. List the steps used in building the Data warehouse?

Ans :

The key steps involved in a data warehousing project are outlined below:

- Users specify information needs
- Analysts and users create a logical and physical design
- Sources of data are identified in operational systems, external sources etc.
- Source data is scrubbed, extracted and transformed
- Data is transferred and loaded into the warehouse periodically
- Users are given access to the warehouse data
- The warehouse is maintained in terms of changing requirements.

Q6. List and explain in brief the components of Data Warehouse?

(OR)

What are the different components of a data warehouse?

Ans :

Some of the major components of a data warehouse:

- **Operational data**

Data for the warehouse may be sourced in a number of ways, e.g. from mainframe-based hierarchical or network databases, from relational databases and from data in proprietary file systems.

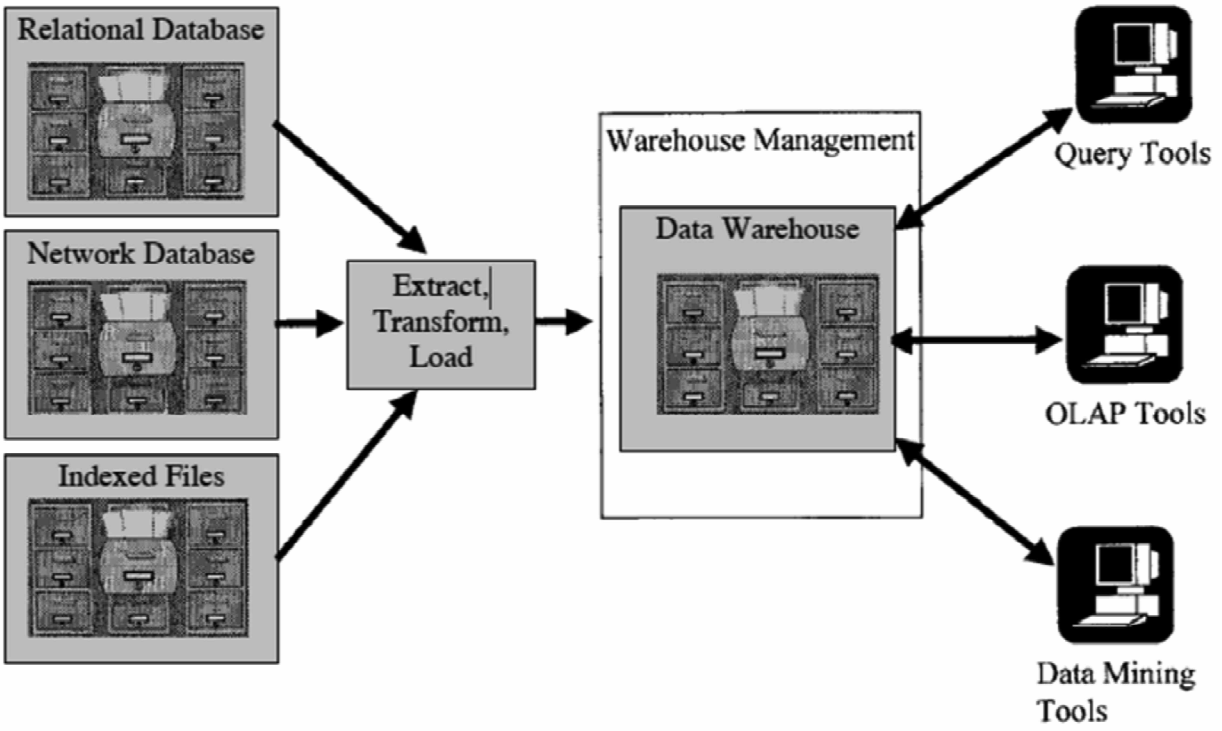
- **Extraction, transformation and loading functions**

These ETL operations or functions are concerned with extracting data from source systems, transforming into a suitable form and loading the transformed data into the data warehouse

- **Warehouse management**

A series of functions must be provided to manage the warehouse: consistency analysis, indexing, denormalisation, aggregation, backup and archiving

- **Query management:** The warehouse must perform a series of operations concerned with the management of queries for use by a variety of actors. reporting and query tools, OLAP tools or tools for data mining.



4.1.1 RDBMS and Big Data

Q7. Explain the relationship between RDBMS and Big Data.

Ans : **(Imp.)**

An RDBMS uses a relational model where all the data is stored using preset schemas. These schemas are linked using the values in specific columns of each table. The data is hierarchical, which means for data to be stored or transacted it needs to adhere to ACID standards, namely:

- **Atomicity:** Ensures full completion of a database operation.
- **Consistency:** Ensures that data abides by the schema (table) standards, such as correct data type entry, constraints, and keys.
- **Isolation:** Refers to the encapsulation of information. Makes only necessary information visible.
- **Durability:** Ensures that transactions stay valid even after a power failure or errors.

In traditional database systems, every time data is accessed or modified, it requires to be moved (indexed) to a central location for processing. There in lies a major limitation of hardware upgradation. however, depending on the hardware platform, there is a limitation on the number of processors and system memory that can be used to concurrently perform database operations. Besides the processing power restraint, network latency can also occur during data transfer to the central node.

RDBMS presumes your data to be essentially correct beforehand. Mapping real-world problems to a relational database model comprises many improvised strategies, but the textbook approach recommends following a three-step process: conceptual, logical, and physical modeling. But simply mapping the problem in such a way doesn't work. It makes some incorrect assumptions, such as your information system staying consistent and static all the time. However in the real world, data is only partially legible.

For example, a bank cannot maintain its Web server access logs using the same database that it uses for financial transactions. This is because it is more important for financial transactions to be highly consistent than the other requirement of maintaining logs. But since one of the operations (transactions) happens to be absolute and the other not, we clearly defy the presumption of information staying consistent all the time, which we had about our information systems.

The Internet has grown by leaps and bounds in the past two decades. Numerous domains have been registered, more than a billion gigabytes of Web-space has been reserved. With the digital revolution of the early 1990s aiding the personal computer segment, Web transactions have grown rapidly. With the advent of various search engines, lure of easily available information, and freely distributable information, the social media platform has recorded a threefold increase in the volume of transactions. Although solutions for such situations occurring in corporate scenarios have been there, coping with Web-based transactions has turned out to be a compelling factor while addressing database-related issues with Big Data solutions.

Big Data mainly takes three Vs into account: Volume, Variety, and Velocity. These three terms can be briefly explained as follows:

- **Volume of Data**

  Big Data is designed to store and process a few hundred terabytes, or even petabytes or zettabytes of data.

- **Variety of Data**

  Collection of data, different from a format suiting relational database systems, is stored in a semi-structured or an unstructured format.

- **Velocity of Data**

  The rate of data arrival might make an enterprise data warehouse problematic, particularly where formal data preparation processes like conforming, examining, transforming, and cleansing of the data needs to be accomplished before it is stored in data warehouse tables.

Big Data solutions are designed for storing and managing enormous amounts of data using a simple file structure, formats, and highly distributed storage mechanisms with the initial handling of the data occurring at each storage node, unlike RDBMS. This obviates the need for data to be moved every time over the network for even a simple processing.

One of the biggest difficulties with RDBMS is that it is not yet near the demand levels of Big Data. The volume of data handling today is rising at a fast rate.

Q8. Explain the differences between RDBMS and Big Data.

Ans :

Sl.No.	Nature	Relational Database Systems	Big Data Solutions
1.	Data types and formats	Structured	Structured, semi-structured and unstructured
2.	Data integrity updates	High transactional used - often follows consistent models	Depends on the technology
3.	Schema	Static	Dynamic
	Read and write pattern	Fully repeatable read/write	Write once, repeatable read
4.	Storage volume	Gigabytes to terabytes beyond	Terabytes, petabytes, and
5.	Scalability powerful hardware	Scale up with more servers	Scale out with additional
6.	Data processing distribution	Limited or none	Distributed across clusters
7.	Economics software	Expensive hardware and open-source software	Commodity hardware and

Table : Feature of Relational Databases and Big Data Solutions

4.1.2 Issues with Relational Model

Q9. Explain the issues with Relational Model.

Ans : **(Imp.)**

The following are the issues with relational model :

1. Maintenance Problem

The maintenance of the relational database becomes difficult over time due to the increase in the data. Developers and programmers have to spend a lot of time maintaining the database.

2. Cost

The relational database system is costly to set up and maintain. The initial cost of the software alone can be quite pricey for smaller businesses, but it gets worse when you factor in hiring a professional technician who must also have expertise with that specific kind of program.

3. Physical Storage

A relational database is comprised of rows and columns, which requires a lot of physical memory because each operation performed depends on separate storage. The requirements of physical memory may increase along with the increase of data.

4. Lack of Scalability

While using the relational database over multiple servers, its structure changes and becomes difficult to handle, especially when the quantity of the data is large. Due to this, the data is not scalable on different physical storage servers. Ultimately, its performance is affected i.e. lack of availability of data and load time etc. As the database becomes larger or more distributed with a greater number of servers, this will have negative effects like latency and availability issues affecting overall performance.

5. Complexity in Structure

Relational databases can only store data in tabular form which makes it difficult to represent complex relationships between objects. This is an issue because many applications require more than one table to store all the necessary data required by their application logic.

6. Decrease in performance over time

The relational database can become slower, not just because of its reliance on multiple tables. When there is a large number of tables and data in the system, it causes an increase in complexity. It can lead to slow response times over queries or even complete failure for them depending on how many people are logged into the server at a given time.

Q10. What are the advantages of relational database.

Ans :

The relational database benefits are discussed briefly.

1. Simplicity of Model

In contrast to other types of database models, the relational database model is much simpler. It does not require any complex queries because it has no query processing or structuring so simple SQL queries are enough to handle the data.

2. Ease of Use

Users can easily access/retrieve their required information within seconds without indulging in the complexity of the database. Structured Query Language (SQL) is used to execute complex queries.

3. Accuracy

A key feature of relational databases is that they're strictly defined and well-organized, so data doesn't get duplicated. Relational databases have accuracy because of their structure with no data duplication.

4. Data Integrity

RDBMS databases are also widely used for data integrity as they provide consistency across all tables. The data integrity ensures the features like accuracy and ease of use.

5. Normalization

As data becomes more and more complex, the need for efficient ways of storing it increases. Normalization is a method that breaks down information into manageable chunks to reduce storage

size. Data can be broken up into different levels with any level requiring preparation before moving onto another level of normalizing your data.

Database normalization also ensures that a relational database has no variety or variance in its structure and can be manipulated accurately. This ensures that integrity is maintained when using data from this database for your business decisions.

6. Collaboration

Multiple users can access the database to retrieve information at the same time and even if data is being updated.

7. Security

Data is secure as Relational Database Management System allows only authorized users to directly access the data. No unauthorized user can access the information.

4.2 NON-RELATIONAL DATABASE

Q11. Explain about Non-Relational Database.

(OR)

What are called Non-Relational Database?

Ans : **(Imp.)**

Non-relational databases do not rely on the table/key model endemic to RDBMSs. In short, specialty data in the big data world requires specialty persistence and data manipulation techniques.

One emerging, popular class of non-relational database is called not only SQL (NoSQL). Originally the originators envisioned databases that did not require the relational model and SQL. As these products were introduced into the market, the definition softened a bit and now they are thought of as "not only SQL," again bowing to the ubiquity of SQL.

The other class is databases that do not support the relational model, but rely on SQL as a primary means of manipulating the data within. Even though relational and non-relational databases have similar fundamentals, how the fundamentals are accomplished creates the differentiation.

Non-relational database technologies have the following characteristics in common :

- **Scalability**

  In this instance, this refers to the capability to write data across multiple data stores simultaneously without regard to physical limitations of the underlying infrastructure. Another important dimension is seamlessness. The databases must be able to expand and contract in response to data flows and do so invisibly to the end users.

- **Data and Query model**

  Instead of the row, column, key structure, nonrelational databases use specialty frameworks to store data with a requisite set of specialty query APIs to intelligently access the data.

- **Persistence design**

  Persistence is still a critical element in nonrelational databases. Due to the high velocity, variety, and volume of big data, these databases use difference mechanisms for persisting the data. The highest performance option is "in memory," where the entire database is kept in the very fast memory system of your servers.

- **Interface diversity:** Although most of these technologies support RESTful APIs as their "go to" interface, they also offer a wide variety of connection mechanisms for programmers and database managers, including analysis tools and reporting/visualization.

- **Eventual Consistency:** While RDBMS uses ACID (Atomicity, Consistency, Isolation, Durability) for ensuring the consistency of data, non-relational DBMS use BASE. BASE stands for Basically Available, Soft state, and Eventual Consistency. Eventual consistency is most important because it is responsible for conflict resolution when data is in motion between nodes in a distributed implementation. The data state is maintained by the software and the access model relies on basic availability.

4.2.1 Issues with Non Relational Database

Q12. Explain the advantages and disadvantges of Non Relational Database.

Ans : **(Imp.)**

Advantages

The following are the advantages of a Non-relational database:

- Object-oriented programming makes it easy to use and flexible
- It can deal with large volumes of structured, unstructured, and semi-structured data
- It provides high availability
- No schema required
- Provides efficiency, scale-out with a monolithic architecture

Disadvantages

The following are the advantages of the Non-relational database:

- The non-relational database is still growing, and many features are yet being implemented
- It requires some technical skills to install and maintain
- As it is open-source, it will provide less support
- Does not have in-built data integrity
- Weak or Eventual consistency BASE (Basically Available, Soft state, Eventually consistent)
- Differences between Relational and Non-relational database

Q13. What is the Difference Between Relational and Non-Relational Database?

Ans :

Sl. No.	Relational	Non-relational
1.	The relational database is a structured database.	The non-relational database is a document oriented database.
2.	It has a predefined schema.	It uses a dynamic schema for the unstructured database.
3.	The relational database is vertically scalable.	The non-relational database is horizontally scalable.
4.	It is not suitable for hierarchical data storage.	It is ideal for hierarchical data storage as it supports key-value method.
5.	The query language is Structured Query Language(SQL).	No declarative query language.
6.	It is used when the data validity is important.	It is used when it is important to have fast data than correct data.
7.	It supports ACID(Atomicity, Consistency, Isolation, Durability) set of properties.	It supports BASE(Basically Available, Soft state, Eventually consistent) model.
8.	It is configured for durable consistency.	It depends on the Database management system(DBMS) that offers strong consistency such as MongoDB, and some other offers eventual consistency such as Cassandra.

Q14. Discuss various issues with Non Relational Database.

Ans :

- The non-relational data model looks more like a concept of one entity, and all the data that relates to that one entity is called a document.
- The way your blogging software interacts with the blog post is theoretically a lot simpler in a non-relational database.
- The request comes in, and the blogging software queries the database and says “Return a specific post and everything related to it.”
- The bottom line is that you should be well aware of what you are doing at the software level.
- Imagine a grocery list. You can make a grocery list in a simple way —you write down the items you want on a piece of paper. But say, after few months, you want to know how many bread loaves you bought last year.
- In such a case, with the non-relational database model you have to go through each list and count every loaf separately, whereas in a relational database modeling, you can get that count instantly. Yes, it means a bit of extra work on the front end, but modeled correctly, it is indeed an efficient way.
- In situations where you want to be careful about your data and its structure - bank accounts, for example - A relational database is an absolute must, theoretically better, and a tried and proven technology.

- To sum up, individual limitations of the relational and non-relational data models may appear as trivial at a lower level but going up in the Big Data ecosystem, things become too evident to be overlooked.
- Some popular non-relational databases are CouchDB, MongoDB, Cassandra, Bigtable).

4.3 POLYGLOT PERSISTENCE

Q15. Discuss about Polyglot Persistence.

Ans : **(Imp.)**

The term polyglot in Big Data is applied to a set of applications that use several core database 'technologies. A polyglot database is often used to solve a complex problem by breaking it into fragments and applying different database models. Then, the results of different sets are aggregated into a data storage and analysis solution. We can attain the results by using multiple data storage technologies for a particular application, which is chosen based upon the best way data can be stored and retrieved for an application. In Big Data, it means picking the right NoSQL DB for the right application.

For examples, Disney in addition to RDBMS also uses Cassandra and MongoDB. Netflix uses Cassandra, Hbase, and SimpleDB. Twitter uses Cassandra, FlockDB, Hbase, and MYSQL. Mendeley uses Hbase, MongoDB, Solr, and Voldemort.

A lot of corporations still use relational databases for some data, but the increasing persistence requirements of dynamic applications are growing from predominantly relational to a mixture of data sources.

Polyglot persistence is quite a common and popular paradigm among database professionals, but an important reason is that users, like applications, come studded with rich experiences – where they can easily search for what they are looking, without caring about rest of the frills – such as finding friends, nearby restaurants, accurate information, etc.

Figure shows an example of a Web application using different types of databases for handling complex user queries:

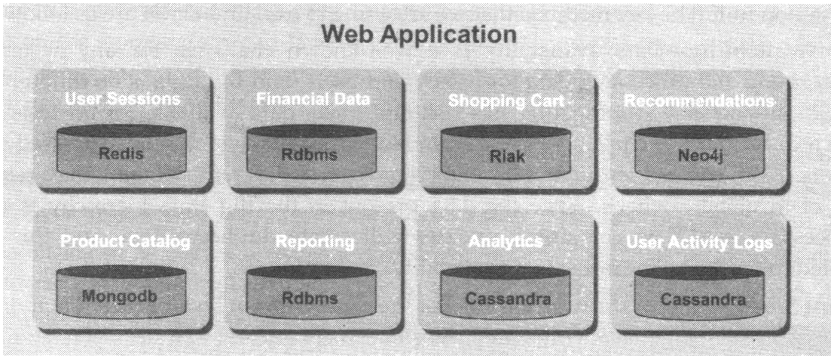


Fig. : Displaying a Web Application with Multiple Databases

To build a useful app for a user, developers are required to make clever improvisations with the existing ways of storing and querying data. They need tools to provide the required context-based content to the users.

4.4 Integrating Big Data with Traditional Data Warehouse

Q16. How big data integrating with traditional warehouse.

Ans : **(Imp.)**

After a feasibility study of Big Data requirement, the identification and transition phase commences. Planning a heterogeneous data environment is a task that involves multiple phases of undo-redo, quality checks, and integrity assurances, whereas implementing a Big Data solution to a previously non-existent environment includes key challenges to be taken care of, before any further commitments ensue. While Big Data and the traditional data warehouse confront each other often, they are more likely to stay homogenous than to come up with a new version of Big Data. Data warehouses are highly structured and adjusted for custom purposes. Besides, these structures of records happen to be highly centralized as well.

The relational data model is here to stay for a long time, since organizations will continue to use data warehouses to manage organized and operational type of data that characterize record systems. For business analysts, these data warehouses will have the ability to analyze trends in Big Data. A relationship between Big Data and a data warehouse can be best described as a hybrid structure in which the well-structured, optimized, and operational data remains in the heavily guarded data warehouse, while the highly distributed data subject to variations in real time is managed by a Hadoop- or NoSQL-based infrastructure.

Gradually, organizations are beginning to realize that they have an inevitable business requirement of combining traditional data warehouses with their archived business data sources to less structured and scrutinized Big Data sources. A hybrid approach helps in accomplishing these business goals, which supports a cross between traditional and Big Data sources.

The main challenges confronting the physical architecture of the next-gen data warehouse platform include data availability, loading, storage performance, data volume, scalability, assorted and varying query demands against the data, and operational costs of maintaining the environment. To cope with any issues that might hamper the overall implementation and integration process, a number of challenges are pre-studied and dealt with before actually assigning bandwidth to the data implementation unit. The key methods that we are going to explain in brief are as follows:

- **Data Availability**

  Data availability is a well-known challenge for any system related to transforming and processing data for use by end-users, and Big Data is no different. Hadoop or NoSQL is beneficial in mitigating this risk and make data available for analysis immediately upon acquisition. The challenge is to sort and load the data, which is unstructured and in varied formats. Also, context-sensitive data involving several different domains may require another level of availability check. Since the data present in the Big Data hierarchy is not updated, reprocessing new data containing updates will create duplicate data, and this needs to be handled to minimize the impact on availability.

- **Pattern Study**

  Pattern study is nothing but the centralization and localization of data according to the demands. A global e-commerce website can centralize requests and fetch directives and results on the basis of end user locations, so as to return only meaningful contextual knowledge than to impart the entire data to the user. Trending topics are one of the pattern-based data study models that are a popular mode of knowledge gathering for all platforms. The trending pattern to be found for a given regional location is matched for occurrence in the massive data stream, in terms of keywords or popularity of links as per the hits they receive, and based on the geographical diversifications and stream filtering methods, data with similar characteristics is conjoined to form a pattern.

- **Data Incorporation and Integration:** Since no guidebook format or schema metadata exists, the data incorporation process for Big Data is about just acquiring the data and storing as files. However, continuous data processing on a platform can create a conflict for resources over a given period of time, often leading to deadlocks. Especially in case of big documents, images or videos, if such requirements happen to be the sole architecture driver, a dedicated machine can be allocated for this task, bypassing the guesswork involved in the configuration and setup process.

- **Data Volumes and Exploration:** Traffic spikes and volatile surge in data volumes can easily dislocate the functional architecture of corporate infrastructure due to the fundamental nature of the data streams. On each cycle of data acquisition completion, retention requirements for data can vary depending on the nature and the freshness of the data and its core relevance to the business. Data exploration and mining is an activity responsible for Big Data procurements across organizations and also yields large data sets as processing output. These data sets are required to be preserved in the system by occasional optimization of intermediary data sets. This is an important area normally ignored by organizations, which can be a potential performance drain over a given period of time.

- **Compliance and Localized Legal Requirements:** Various compliance standards such as Safe Harbor, GLBA, and PCI regulations can have some impact on data security and storage. Therefore, these standards should be judiciously planned and executed. Moreover, there are several cases of transactional data sets not being stored online required by the courts of law. Big Data infrastructure can be used as a storage engine for such data types, but the data needs to comply with certain standards and additional security. Large volumes of data can affect overall performance, and if such data sets are processed on the Big Data platform, the appliance configurator can provide administrators with tools/tips to mark the data in its own area of infrastructure zoning, minimizing both risk and performance impact.

- **Storage Performance:** In all these years, storage-based solutions didn't advance as rapidly as their counterparts, processors, memories, or cores did. Disk performance is a vital point to be taken care of while developing Big Data systems, and appliance architecture can throw better light on the storage class and layered architecture. If a combination of Solid State Drive (SSD), in-memory, and traditional storage architecture is intended for Big Data processing, the exchange and persistence of data across the different layers can be time-consuming and vapid.

An often misunderstood conviction among corporations is that once a Big Data solution is implemented, the existing relational data warehousing becomes redundant and are not required anymore, since they already have the best technology of recent times.

4.5 Big Data Analysis and Data Warehouse

Q17. Explain briefly about Big Data Analysis and Data Warehouse.

Ans : **(Imp.)**

Big Data is analyzed to know the present behavior or trends and make future predictions. Various Big Data solutions are used for extracting useful information from Big Data. A simple working definition of a Big Data solution is that it is a technology that:

- Enables the storage of very large amounts of heterogeneous data
- Holds data in low-cost storage devices
- Keeps data in a raw or unstructured format

Big Data analytics performed by a Big Data solution helps organizations in the following ways:

- Brings improvement in the tourism industry by analyzing the behavior of tourists and their bends
- Enables improvement in technological research
- Enables improvement in the medical field for diagnosing diseases quickly
- Helps the defense sector by enabling better monitoring
- Helps the insurance industry by better customer relationship management

Data warehousing, on the other hand, is a group of methods and software to enable data collection from functional systems, integration, and synchronization of that data into a centralized database, and then the analytical visualization and report-hacking of key performance indicators in a dashboard-driven environment. There are different interpretations of the meaning of data warehousing. However, it is generally agreed that data warehouse means something that is achieved by incorporating data from multiple heterogeneous sources that support logical reporting, structured decision making, and random queries. The data warehousing process comprises cleaning, integration, and consolidation of data. A data warehouse acts as a single point of reference for a granular and integrated data of the company.

A Big Data solution is preferred because there is a lot of data that has to be manually and relationally handled. In organizations handling Big Data, if the data is potentially used, it can provide much valuable information leading to superior decision making which, in turn, can lead to more profitability, revenue, and happier customers. Isn't this what most organizations want?

Such organizations require a data warehouse in order to make rational decisions. In order to have a good knowledge of what actually is going on in your company, you need your data to be reliable, credible, and available to everyone.

On comparing a data warehouse to a Big Data solution, we find that a Big Data solution is a technology and data warehousing is an architecture. They are two distinct things.

Technology is just a medium to store and operate huge amounts of data. A data warehouse is a way of organizing data so that when someone queries or fetches data from it, that person knows about other people having the same data for different purposes. There is a scope of data reconcilability in the case of a data warehouse.

For example, can an organization;

- Have a Big Data solution and no data warehouse or vice versa? Yes.
- Have a Big Data solution and data warehouse or vice versa? Yes.

There is hardly a correlation between a Big Data analysis and a data warehouse. They are not the same thing.

In a typical data warehouse, you will find a combination of flat files, relational database tables, and non-relational sources. A well-constructed data warehouse will be architected so that the data is converted into a common format, allowing queries to be correctly and consistently processed.

Big Data is not a Substitute for a Data Warehouse

Data warehouses work with abstracted data that has been operated, filtered, and transformed into a separate database, using analytics such as sales trend analysis or compliance reporting. That database is updated gradually with the same filtered data, either at a weekly or monthly basis. But Big Data systems have raw data whether from operations, for example, reporting, user activity such as session hacking, or other real-world norms. That data is left untouched since the actual usage pattern or requirement is not known or predetermined.

Actually, organizations that use data warehousing technology will continue to do so and those that use both Big Data and data warehousing are future-proof from any further technological advancements only up till the point where the thin line of separation starts depleting. Big Data systems are normally used to understand strategic issues, for example, for inventory maintenance or target-based individual performance reports.

Enterprises still use data warehousing for reports and visualizations for management purposes to report a clearer and more lucid picture about company finances. Conventional data warehouse systems are proven systems and with investments to the tune of millions of dollars for their development, those systems, are not going anywhere, soon. Regardless of how good and profitable Big Data analytics is or turns out, data warehousing will still continue to provide crucial database support to many enterprises, and in all circumstances, will complete the lifecycle of current systems.

UNIT V

NoSQL Data Management: Introduction to NoSQL, Characteristics of NoSQL, History of NoSQL, Types of NoSQL Data Models- Key Value Data Model, Column Oriented Data Model, Document Data Model, Graph Databases, Schema-Less Databases, Materialized Views, CAP Theorem.

5.1 NoSQL Data Management

5.1.1 Introduction to NoSQL

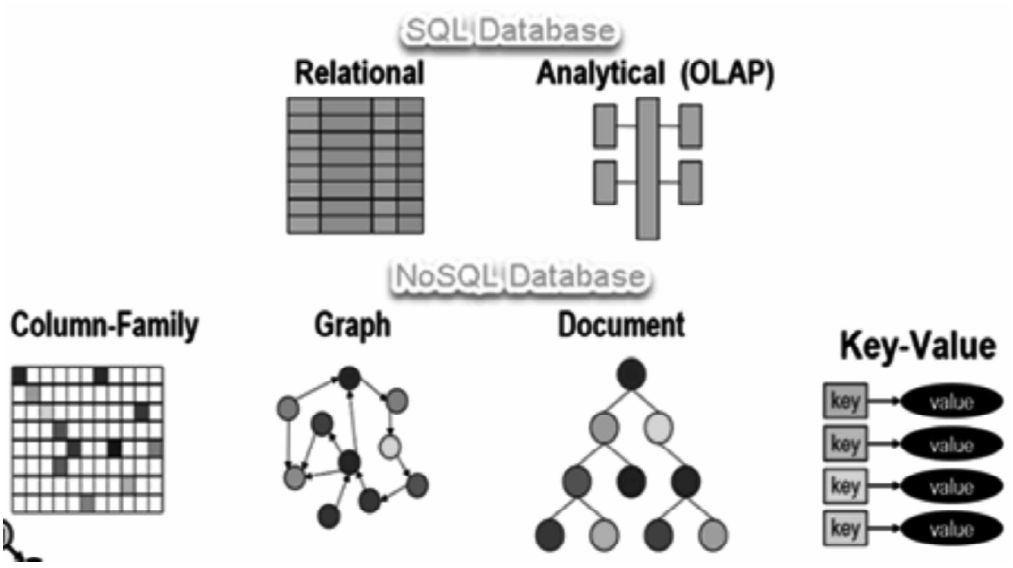
Q1. What is NoSQL?

Ans :

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would be "NoREL", NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.



5.1.2 Characteristics of NoSQL

Q2. What are the Characteristics of NoSQL?

Ans :

Although there are different ways that can be incorporated to understand how NoSQL databases work, we will now look at some of the most common features that define a basic NoSQL database.

1. **Complex-free Working:** Unlike SQL databases, NoSQL databases are not complicated. They store data in an unstructured or a semi-structured form that requires no relational or tabular arrangement. Perhaps they are easier to use and can be accomplished by all.

2. **Independent of Schema:** Secondly, NoSQL databases are independent of schemas which implies that they can be run over without any predetermined schemas.

   That said, they are far more efficient to work with and perhaps this particular feature works well for young programmers and organizations handling large amounts of heterogeneous data that requires no schemas to structure it.

3. **Better Scalability:** One of the most prominent features of such a database is that it has high scalability that makes it suitable for large amounts of data.

   Needless to mention that the contemporary data scientists often prefer to work with NoSQL databases due to this feature since it allows them to accommodate humongous data without rupturing its efficacy.

4. **Flexible to Accommodate:** Since such databases can accommodate heterogeneous data that requires no structuring, they are claimed to be flexible in terms of their usage and reliability.

   For beginners intending to try their hands in the field, NoSQL databases are easy to handle yet very useful.

5. **Durable:** If durability is not one of its most striking features, then what is? NoSQL databases are highly durable as they can accommodate data ranging from heterogeneous to homogeneous.

   Not only can they accommodate structured data, but they can also incorporate unstructured data that requires no query language. Undoubtedly, these databases are durable and efficient.

5.1.3 History of NoSQL

Q3. Explain the evolution of NoSQL.

Ans :

The acronym NoSQL was first used in 1998 by Carlo Strozzi while naming his lightweight, open-source “relational” database that did not use SQL. The name came up again in 2009 when Eric Evans and Johan Oskarsson used it to describe non-relational databases.

Relational databases are often referred to as SQL systems. The term NoSQL can mean either “No SQL systems” or the more commonly accepted translation of “Not only SQL,” to emphasize the fact some systems might support SQL-like query languages.

NoSQL developed at least in the beginning as a response to web data, the need for processing unstructured data, and the need for faster processing. The NoSQL model uses a distributed database system, meaning a system with multiple computers.

The non-relational system is quicker, uses an ad-hoc approach for organizing data, and processes large amounts of differing kinds of data. For general research, NoSQL databases are the better choice for large, unstructured data sets compared with relational databases due to their speed and flexibility.

Not only can NoSQL systems handle both structured and unstructured data, but they can also process unstructured Big Data quickly. This led to organizations such as Facebook, Twitter, LinkedIn, and Google adopting NoSQL systems.

These organizations process tremendous amounts of unstructured data, coordinating it to find patterns and gain business insights. Big Data became an official term in 2005.

Q4. What are the advantages and disadvantages of NoSQL?

Ans : **(Imp.)**

Major advantages of NoSQL

(i) Flexible Data Model: NoSQL databases are highly flexible as they can store and combine any type of data, both structured and unstructured, unlike relational databases that can store data in a structured way only.

(ii) Evolving Data Model: NoSQL databases allow you to dynamically update the schema to evolve with changing requirements while ensuring that it would cause no interruption or downtime to your application.

(iii) Elastic Scalability: NoSQL databases can scale to accommodate any type of data growth while maintaining low cost.

(iv) High Performance: NoSQL databases are built for great performance, measured in terms of both throughput (it is a measure of overall performance) and latency (it is the delay between request and actual response).

(v) Open-source: NoSQL databases don't require expensive licensing fees and can run on inexpensive hardware, rendering their deployment cost-effective.

Major disadvantages of NoSQL

(i) Lack of Standardization: There is no standard that defines rules and roles of NoSQL databases. The design and query languages of NoSQL databases vary widely between different NoSQL products – much more widely than they do among traditional SQL databases.

(ii) Backup of Database: Backups are a drawback in NoSQL databases. Though some NoSQL databases like MongoDB provide some tools for backup, these tools are not mature enough to ensure proper complete data backup solution.

(iii) Consistency: NoSQL puts a scalability and performance first but when it comes to a consistency of the data NoSQL doesn't take much consideration so it makes it little insecure as compared to the relational database e.g., in NoSQL databases if you enter same set of data again, it will take it without issuing any error whereas relational databases ensure that no duplicate rows get entry in databases.

Q5. What are the Applications of NoSQL Databases

Ans :

1. **Data Mining:** When it comes to data mining, NoSQL databases are useful in retrieving information for data mining uses. Particularly when it's about large amounts of data, NoSQL databases store data points in both structured and unstructured formats leading to efficient storage of big data.

   Perhaps when a user wishes to mine a particular dataset from large amounts of data, one can make use of NoSQL databases, to begin with. Data is the building block of technology that has led mankind to such great heights.

   Therefore, one of the most essential fields where NoSQL databases can be put to use is data mining and data storage.

2. **Social Media Networking Sites:** Social media is full of data, both structured and unstructured. A field that is loaded with tons of data to be discovered, social media is one of the most effective applications of NoSQL databases.

   From comments to posts, user-related information to advertising, social media marketing requires NoSQL databases to be implemented in certain ways to retrieve useful information that can be helpful in certain ways.

   Social media sites like Facebook and Instagram often approach open-source NoSQL databases to extract data that helps them keep track of their users and the activities going on around their platforms.

3. **Software Development:** The third application that we will be looking at is software development. Software development requires extensive research on users and the needs of the masses that are met through software development.

   However, a developer must be able to scan through data that is available.

   Perhaps NoSQL databases are always useful in helping software developers keep a tab on their users, their details, and other user-related data that is important to be noted. That said, NoSQL databases are surely helpful in software development.

Q6. Explain the Use of NoSQL in industry sector.

Ans : **(Imp.)**

1. **Session Store:** Managing session data using relational database is very difficult, especially in case where applications are grown very much.

   In such cases the right approach is to use a global session store, which manages session information for every user who visits the site.

   NOSQL is suitable for storing such web application session information very is large in size.

   Since the session data is unstructured in form, so it is easy to store it in schema less documents rather than in relation database record.

2. **User Profile Store:** To enable online transactions, user preferences, authentication of user and more, it is required to store the user profile by web and mobile application.

   In recent time users of web and mobile application are grown very rapidly. The relational database could not handle such large volume of user profile data which growing rapidly, as it is limited to single server.

Using NOSQL capacity can be easily increased by adding server, which makes scaling cost effective

3. **Content and Metadata Store:** Many companies like publication houses require a place where they can store large amount of data, which include articles, digital content and e-books, in order to merge various tools for learning in single platform

   The applications which are content based, for such application metadata is very frequently accessed data which need less response times.

   For building applications based on content, use of NoSQL provide flexibility in faster access to data and to store different types of contents

4. **Mobile Applications:** Since the smartphone users are increasing very rapidly, mobile applications face problems related to growth and volume.

   Using NoSQL database mobile application development can be started with small size and can be easily expanded as the number of user increases, which is very difficult if you consider relational databases.

   Since NoSQL database store the data in schema-less for the application developer can update the apps without having to do major modification in database.

   The mobile app companies like Kobo and Playtika, uses NOSQL and serving millions of users across the world.

5. **Third-Party Data Aggregation:** Frequently a business require to access data produced by third party. For instance, a consumer packaged goods company may require to get sales data from stores as well as shopper's purchase history.

   In such scenarios, NoSQL databases are suitable, since NoSQL databases can manage huge amount of data which is generating at high speed from various data sources.

6. **Internet of Things:** Today, billions of devices are connected to internet, such as smartphones, tablets, home appliances, systems installed in hospitals, cars and warehouses. For such devices large volume and variety of data is generated and keep on generating.

   Relational databases are unable to store such data. The NOSQL permits organizations to expand concurrent access to data from billions of devices and systems which are connected, store huge amount of data and meet the required performance.

7. **E-Commerce:** E-commerce companies use NoSQL for store huge volume of data and large amount of request from user.

8. **Social Gaming:** Data-intensive applications such as social games which can grow users to millions. Such a growth in number of users as well as amount of data requires a database system which can store such data and can be scaled to incorporate number of growing users NOSQL is suitable for such applications.

   NOSQL has been used by some of the mobile gaming companies like, electronic arts, zynga and tencent.

9. **Ad Targeting:** Displaying ads or offers on the current web page is a decision with direct income To determine what group of users to target, on web page where to display ads, the platforms gathers behavioral and demographic characteristics of users.

   A NoSQL database enables ad companies to track user details and also place the very quickly and increases the probability of clicks.

   AOL, Mediamind and PayPal are some of the ad targeting companies which uses NoSQL.

Q7. Compare relational databases with NoSQL databases.

Ans :

A comparison of relational databases and NoSQL databases is shown in the following table:

Sl.No.	Nature	Relational Database	NoSQL Database
1.	**Dataset size**	Relational databases are designed to handle similar type of data and large datasets, for example, transactional data.	NoSQL databases are designed to handle different types of data as well as very large datasets, for example, Google or Facebook data.
2.	**Scalability**	It provides less scalability.	It provides more scalability.
3.	**Data Consistency Model**	Relational databases use ACID properties. As a result, a consistent and equivalent view of data for all users is guaranteed. So, the data which is committed will be available to the customer. For example, in case of online banking, consistency is a key criterion. Therefore, we have to use relational databases.	NoSQL follows Basic Availability, Soft State and Eventual Consistency (BASE). NoSQL is designed according to the client's requirements. For example, in case of a social media website, consistency is not an issue. Therefore, we can use NoSQL databases here.
4.	**CAP Theorem**	RDBMS are also called CA systems, because they possess the features of consistency and availability.	NoSQL can be called CP or AP systems, which means that they have consistency and partitioning or availability and partitioning.
5.	**Data Model**	RDBMS contains a data model in the form of rows and columns.	NoSQL does not have a fixed data model. No SQL can therefore be key/value, column-oriented, or document-oriented.
6.	**Data Integrity**	Data integrity can be ensured in relational databases with the help of foreign keys and by defining fields like unique or not-null, etc.	The design of NoSQL is such that it does not provide data integrity; instead, it is the responsibility of an application to handle it.
7.	**Querying**	Relational databases use SQL for querying. One of the advantages of using these databases is that they support joins.	NoSQL does not use the concept of joins. For the purpose of querying, these databases require APIs, a proprietary language, or object graph navigation APIs or Map Reduce APIs, etc.
8.	**Schema Definition**	RDBMS schema is fixed. We also need to define the database schema before using it.	NoSQL has no schemabased restrictions.

5.2 Types of NoSQL Data Models

Q8. State the various types of NoSQL Data Models.

Ans :

There are generally four types of NoSQL database. Each of them has there own attributes and limitations

1. Key Value Data Model
2. Column Oriented Data Model
3. Document Data Model
4. Graph Databases.

5.2.1 Key Value Data Model

Q9. Discuss about Key Value Data Model.

Ans : **(Imp.)**

In the key value data model, the client can get the value for a key, put a value for a key, or delete a key from the data store. Here, the value is a Binary Large Object (BLOB) that is only concerned about data and not what is inside; it is the responsibility of the application to understand what exactly is stored. BLOB is also scalable because it uses primary key access for the different purposes.

A sample key/value database is shown in Figure:

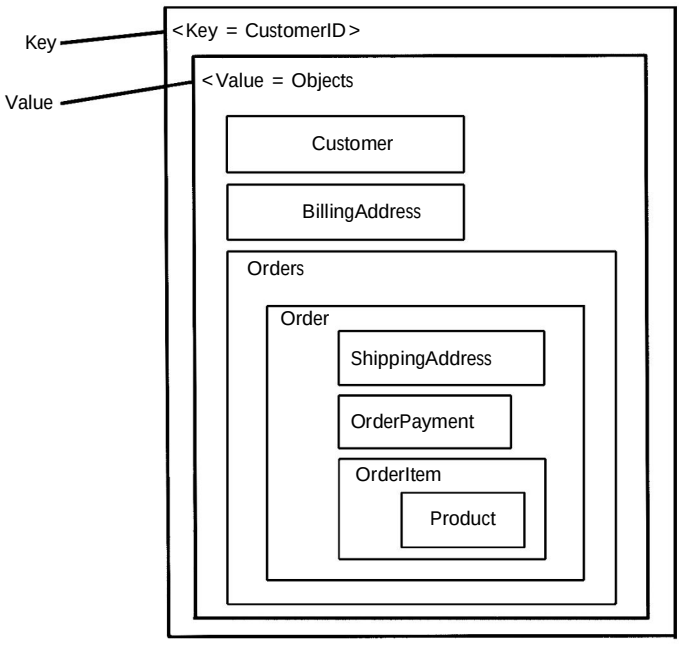


Fig.: A Key Value Database

Some examples of popular key/value databases are Riak; Redis; Memcached and its versions; Berkeley DB; HamsterDB; Amazon DynamoDB; Project Voldemort; and Couchbase. All key/value databases are not the same; there are major differences between these products; for example, Memcached data is not persistent, but Riak data is.

Therefore, we need to cache user preferences in Memcached data. If we implement user preferences in Memcached, they would need to be refreshed entirely in case any of the nodes are lost. On the other hand, we may not need to worry about losing data if it is stored in the Riak database. However, we need to consider how to update stale data. So, it is also important to choose the type of key/value databases as per our requirement.

5.2.2 Column Oriented Data Model

Q10. Explain briefly about Column Oriented Data Model.

Ans : **(Imp.)**

At a very surface level, column-store databases do exactly what is advertised on the tin: namely, that instead of organizing information into rows, it does so in columns. This essentially makes them function the same way that tables work in relational databases. Of course, since this is a NoSQL database, this data model makes them much more flexible.

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

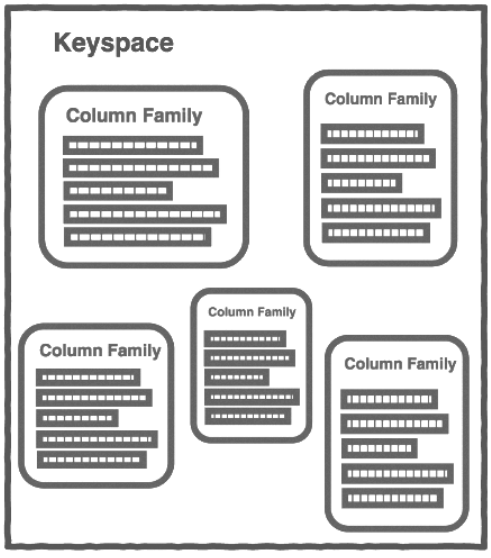
Column-oriented

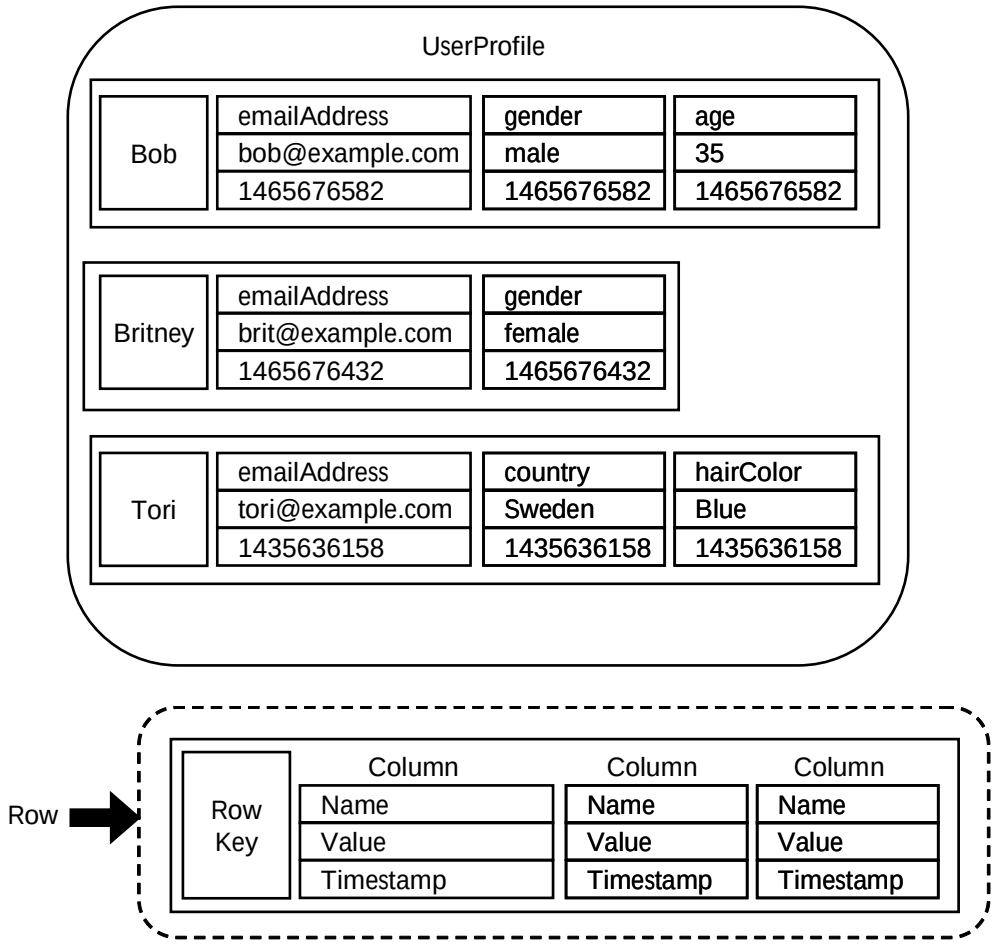
Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

More specifically, column databases use the concept of keyspace, which is sort of like a schema in relational models. This keyspace contains all the column families, which then contain rows, which then contain columns. It's a bit tricky to wrap your head around at first but it's relatively straightforward.





By taking a quick look, we can see that a column family has several rows. Within each row, there can be several different columns, with different names, links, and even sizes (meaning they don't need to adhere to a standard). Furthermore, these columns only exist within their own row and can contain a value pair, name, and a timestamp

If we take a specific row as an example:

The Row Key is exactly that: the specific identifier of that row and is always unique. The column contains the name, value, and timestamp, so that's straightforward. The name/value pair is also straight forward, and the timestamp is the date and time the data was entered into the database.

Some examples of column-store databases include Casandra, CosmoDB, Bigtable, and HBase.

Benefits of Column Databases

There are several benefits that go along with columnar databases:

- Column stores are excellent at compression and therefore are efficient in terms of storage. This means you can reduce disk resources while holding massive amounts of information in a single column Since a majority of the information is stored in a column, aggregation queries are quite fast, which is important for projects that require large amounts of queries in a small amount of time.

- Scalability is excellent with column-store databases. They can be expanded nearly infinitely, and are often spread across large clusters of machines, even numbering in thousands. That also means that they are great for Massive Parallel Processing Load times are similarly excellent, as you can easily load a billion-row table in a few seconds. That means you can load and query nearly instantly.

- Large amounts of flexibility as columns do not necessarily have to look like each other. That means you can add new and different columns without disrupting the whole database. That being said, entering completely new record queries requires a change to all tables.
- Overall, column-store databases are great for analytics and reporting: fast querying speeds and abilities to hold large amounts of data without adding a lot of overhead make it ideal.

Disadvantages of Column Databases

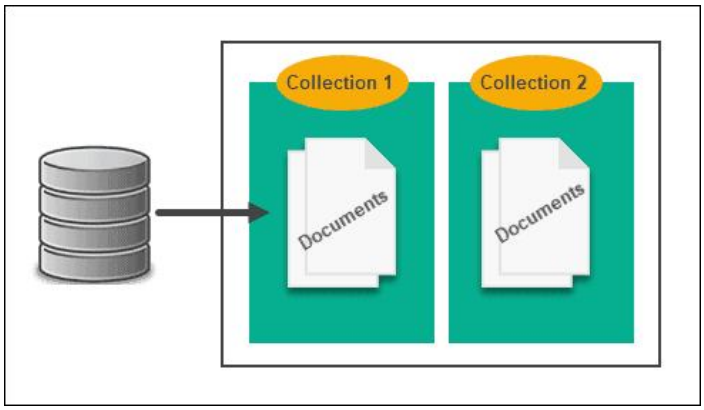
- As it usually is in life, nothing is perfect and there are a couple of disadvantages to using column-oriented databases as well:
- Designing an indexing schema that's effective is difficult and time consuming. Even then, the said schema would still not be as effective as simple relational database schemas.
- While this may not be an issue for some users, incremental data loading is suboptimal and should be avoided if possible.
- This goes for all NoSQL database types and not just columnar ones. Security vulnerabilities in web applications are ever present and the fact that NoSQL databases lack inbuilt security features doesn't help. If security is your number one priority, you should either look into relational databases you could employ or employ a well-defined schema if possible.
- Online Transaction Processing (OLTP) applications are also not compatible with columnar databases due to the way data is stored.

5.2.3 Document Data Model

Q11. Explain in detail Document Data Model.

Ans : **(Imp.)**

A document database is a type of NoSQL database which stores data as JSON documents instead of columns and rows. JSON is a native language used to both store and query data. These documents can be grouped together into collections to form database systems.



Each document consists of a number of key-value pairs. Here is an example of a document that consists of 4 key value pairs:

```
{
    "ID" : "001",
    "Book" : "Java: The Complete Reference",
    "Genre" : "Reference work",
    "Author" : "Herbert Schildt",
}
```

Using JSON enables app developers to store and query data in the same document-model format that they use to organize their app's code. The object model can be converted into other formats, such as JSON, BSON and XML.

Advantages

1. **Schema-less:** There are no restrictions in the format and structure of data storage. This is good for retaining existing data at massive volumes and different structural states, especially in a continuously transforming system.
2. **Faster creation and care:** Minimal maintenance is required once you create the document, which can be as simple as adding your complex object once.
3. **No foreign keys:** With the absence of this relationship dynamic, documents can be independent of one another.
4. **Open formats:** A clean build process that uses XML, JSON and other derivatives to describe documents.
5. **Built-in versioning:** As your documents grow in size they can also grow in complexity. Versioning decreases conflicts.

Disadvantages

1. **Consistency-Check Limitations:** In the book database use case example above, it would be possible to search for books from a non-existent author. You could search the book collection and find documents that are not connected to an author collection.
2. **Each listing may also duplicate author information for each book:** These inconsistencies aren't significant in some contexts, but at upper-tier standards of RDB consistency audits, they seriously hamper database performance.
3. **Atomicity weaknesses:** Relational systems also let you modify data from one place without the need for JOINs. All new reading queries will inherit changes made to your data via a single command (such as updating or deleting a row).

   For document databases, a change involving two collections will require you to run two separate queries (per collection). This breaks atomicity requirements.
4. **Security.** Nearly half of web applications today actively leak sensitive data. Owners of NoSQL databases, therefore, need to pay careful attention to web app vulnerabilities.

Q12. Compare and contrast RDBMS and Document Data Model

Ans :

Sl. No.	RDBMS	Document Database System
1.	Structured around the concept of relationships.	Focused on data rather than relationships.
2.	Organizes data into tuples (or rows). theoretical definitions, instead of rows.	Documents have properties without
3.	Defines data (forms relationships) via constraints and foreign keys (e.g., a child table references to the master table via its ID).	No DDL language for defining schemas.
4.	Uses DDL (Data Definition Language) to create relationships. may contain others nested inside of it, leading to an N:1 or 1:N relationship between the two document entities).	Relationships represented via nested data, not foreign keys (any document
5.	Offeres extreme consistency, critical for soem use cases such as daily banking.	Offers eventual consistency with a period of inconsistency.

Catalogs

Document databases are much more efficient than relational databases when it comes to storing and reading catalog files. Catalogs may have thousands of attributes stored and document databases provide fast reading times. In document databases, attributes related to a single product are stored in a single document. Modifying one product's attributes does not affect other documents.

5.2.4 Graph Databases

Q13. Discuss in detail about Graph Database.

Ans : **(Imp.)**

A graph database is a type of database that makes use of graph structures for semantic queries having nodes, edges, and properties to display and store data. Graph databases are mainly used for storing entities and the relationships between these entities. An entity is nothing but a node with its own properties. A node can be perceived as an instance of an object in an application. Relations are also known as edges, and they can have properties as well. Edges are directions, and each direction has its own significance. The graph is organized, and the data in it are stored once but interpreted in different aspects based on the relationships. Figure shows an example of a graph database:

Whenever we store a graph-like structure in a relational database, it is stored for a single type of relationship. If we add another relationship to it, it means a lot of schema changes and data movement. However, this doesn't happen in the case of graph databases. Additionally, in relational databases, we model the graph based on the traversal. So, if the traversal changes, the data will have to change accordingly.

In graph databases, traversing the joins or relationships can be done very quickly. Moreover, the relationship between nodes is not calculated at the time of querying but is actually persisted as a relationship. The point is that we can have nodes with different types of relationships between them. This allows us to represent relationships between the major domain entities and secondary relationships for category, path, or linked lists for accessing sorted data. Moreover there is no limitation to the number and kind of relationships that can be formed, so relationships can be represented in the same graph database.

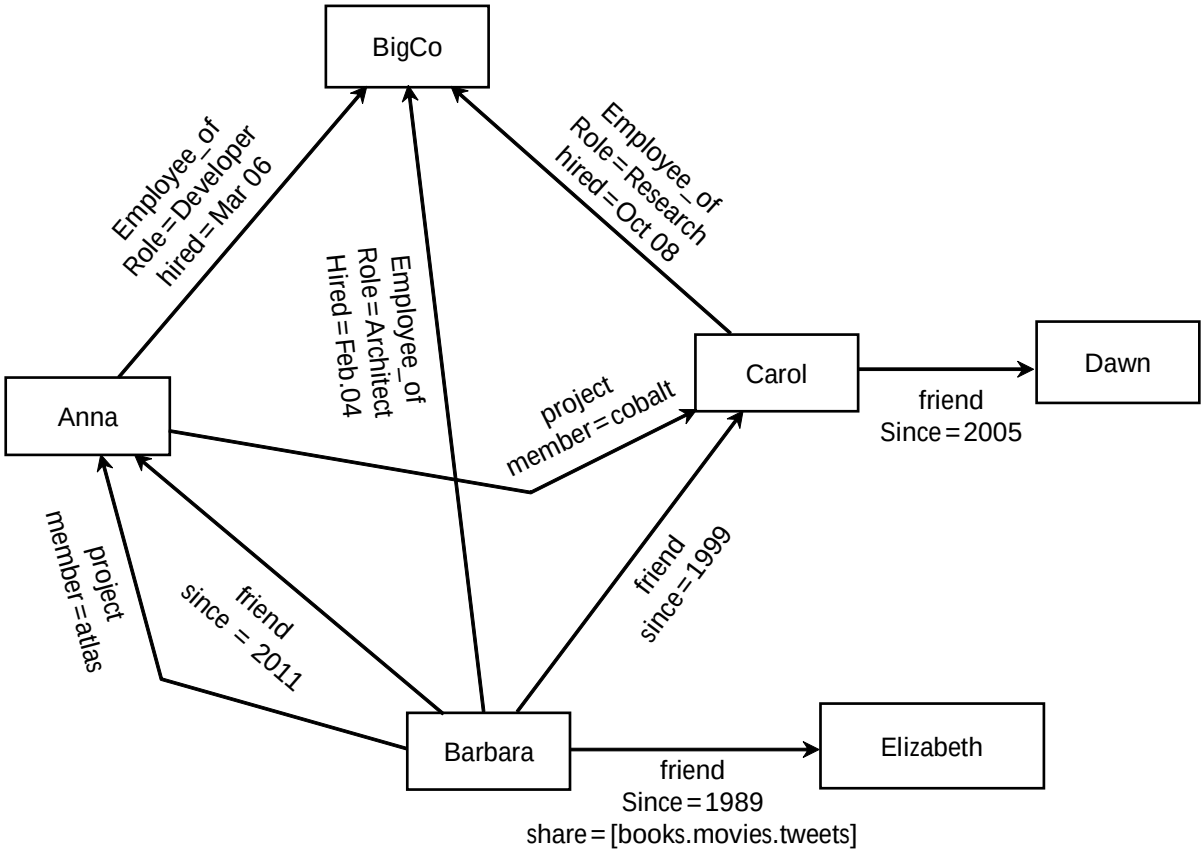


Fig.: A Graph Database

Relationships are of paramount importance in graph databases. They help in deriving most of the values in these databases. Relationships can have properties in addition to a start node and an end node. The most important concept of a database is relations among nodes.

Since relationships and their properties are integral to graph databases, you must invest a lot of time, thought, and design expertise to model the relationships in the domain you are trying to work with. Addition of new relationship types is easy. Changing existing nodes and their relationships is similar to migrating data, because the major changes must be done on each node and each relationship in your data. Some examples of popular graph databases are Neo 4J, Infinite Graph, and FlockDB.

5.3 SCHEMA-LESS DATABASES

Q14. What is a Schema-Less Database?

Ans :

Schema-less databases store data as Key/Value pairs (also known as KV) or as JSON documents. Based on the use cases users have the choice to either store data as KV pairs or as JSON documents. JSON documents are generally very rich in the way data is represented and allows users to very closely model the entity relationship model that we are all very familiar with and have found very useful. For e.g.: An Account entity can be modeled in a JSON document with all the required attributes and the nested values that go with a typical Account object – the multiple address, emails, aliases etc. JSON documents also provide the added advantage of being able to index individual values making the access much more performant, allowing pieces of data from different documents to be joined together.

Schema-Less Databases are :

1. Do not require any modelling (3NF form of normalization) that has a made the careers of lot of database architects including yours truly
2. Do not require pre-setting data types in your repository which greatly reduces time required to stand-up a data repository
3. Can store data with different characteristics and can tolerate change to that definition without having to plan ahead for complex outages and changes and eliminates complex schema migrations
4. Can be easily transformed e.g. an Account Number could start as being all characters or it could be numbers or a combination of both. It is limited only by the user's definition and imagination!

Benefits of Schema-less Databases

1. Database Design or Normalization

Schema-less database eliminates this activity to a very large extent and greatly reduces the complexity of this activity. The only decisions that a user needs to pre-determine is which attributes of an entity they want to keep collocated.

2. ETL/ELT of required to reformat and store structured/unstructured data

3. Schema-less database does not need long drawn transforming and cleansing processes due to the fact that the model is flexible and user is not pigeon-holed into conforming to a strict schema.

Ongoing change management including schema changes

Schema-less databases eliminates complex migrations activities and change synchronization due to it flexible data storage model.

4. Real-time analytics on streaming data

5. A schema-less database lets you define your view of data rather than create a schema that you have to fight to extract value. This lets your non-technical IT dependent staff extract information quickly and easily and do what they are hired to do rather than understand complex data models and spend hours creating their universe of the data. This again translates to huge costs savings.

5.4 MATERIALIZED VIEWS

Q15. What are Materialized Views in NoSQL.

Ans : **(Imp.)**

The Materialized View pattern describes generating prepopulated views of data in environments where the source data isn't in a suitable format for querying, where generating a suitable query is difficult, or where query performance is poor due to the nature of the data or the data store.

These materialized views, which only contain data required by a query, allow applications to quickly obtain the information they need. In addition to joining tables or combining data entities, materialized views can include the current values of calculated columns or data items, the results of combining values or executing transformations on the data items, and values specified as part of the query. A materialized view can even be optimized for just a single query.

A key point is that a materialized view and the data it contains is completely disposable because it can be entirely rebuilt from the source data stores. A materialized view is never updated directly by an application, and so it's a specialized cache.

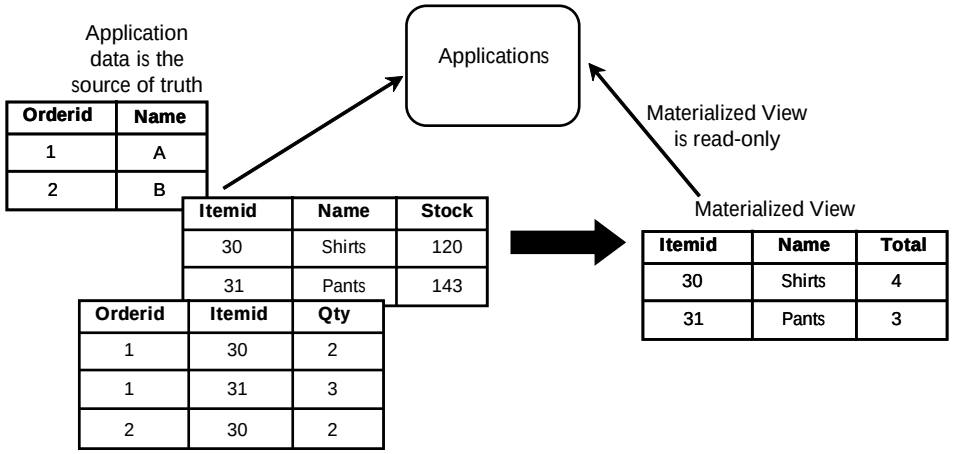


Fig : Materialized View

This pattern is useful when:

- Creating materialized views over data that's difficult to query directly, or where queries must be very complex to extract data that's stored in a normalized, semi-structured, or unstructured way.
- Creating temporary views that can dramatically improve query performance, or can act directly as source views or data transfer objects for the UI, for reporting, or for display.
- Supporting occasionally connected or disconnected scenarios where connection to the data store isn't always available. The view can be cached locally in this case.
- Simplifying queries and exposing data for experimentation in a way that doesn't require knowledge of the source data format. For example, by joining different tables in one or more databases, or one or more domains in NoSQL stores, and then formatting the data to fit its eventual use.
- Providing access to specific subsets of the source data that, for security or privacy reasons, shouldn't be generally accessible, open to modification, or fully exposed to users.

- Bridging different data stores, to take advantage of their individual capabilities. For example, using a cloud store that's efficient for writing as the reference data store, and a relational database that offers good query and read performance to hold the materialized views.

5.5 CAP THEOREM

Q16. Discuss about CAP Theorem.

Ans : **(Imp.)**

The CAP theorem, originally introduced as the CAP principle, can be used to explain some of the competing requirements in a distributed system with replication. It is a tool used to make system designers aware of the trade-offs while designing networked shared-data systems.

The three letters in CAP refer to three desirable properties of distributed systems with replicated data: consistency (among replicated copies), availability (of the system for read and write operations) and partition tolerance (in the face of the nodes in the system being partitioned by a network fault).

The CAP theorem states that it is not possible to guarantee all three of the desirable properties – consistency, availability, and partition tolerance at the same time in a distributed system with data replication.

The theorem states that networked shared-data systems can only strongly support two of the following three properties:

1. **Consistency:** Consistency means that the nodes will have the same copies of a replicated data item visible for various transactions. A guarantee that every node in a distributed cluster returns the same, most recent and a successful write. Consistency refers to every client having the same view of the data. There are various types of consistency models. Consistency in CAP refers to sequential consistency, a very strong form of consistency.

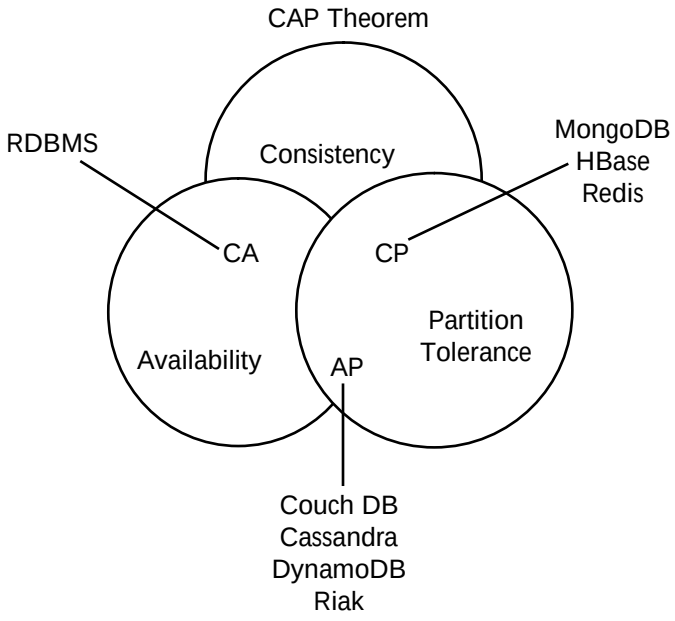


Fig.: Aspects of the CAP Theorem

2. **Availability:** Availability means that each read or write request for a data item will either be processed successfully or will receive a message that the operation cannot be completed. Every non-failing node returns a response for all the read and write requests in a reasonable amount of time. The key word here is "every". In simple terms, every node (on either side of a network partition) must be able to respond in a reasonable amount of time.

3. **Partition Tolerance:** Partition tolerance means that the system can continue operating even if the network connecting the nodes has a fault that results in two or more partitions, where the nodes in each partition can only communicate among each other. That means, the system continues to function and upholds its consistency guarantees in spite of network partitions. Network partitions are a fact of life. Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.

   The use of the word consistency in CAP and its use in ACID do not refer to the same identical concept.

   In CAP, the term consistency refers to the consistency of the values in different copies of the same data item in a replicated distributed system. In ACID, it refers to the fact that a transaction will not violate the integrity constraints specified on the database schema.

Q17. Explain about

i) Acid property

ii) Sharding

Ans :

i) Acid property

ACID (Atomicity, Consistency, Isolation, and Durability) transactions guarantee four properties:

- **Atomicity:** In this either all the operations in transaction will complete or not a single one. If any part of transaction fails, the entire transaction will fail.
- **Consistency:** A transaction must leave the database in an inconsistent state. This ensures that any transaction which is done will change the database to another valid state.
- **Isolation:** Transactions will not interfere with each other.
- **Durability**: The Successful completion of transaction will not be reversed.

Eric's CAP theorem says that if you want consistency, availability, and partition tolerance, then we can look out for two out of three. In a distributed system, partition tolerance means that system will continue to work unless there is a complete network failure.

An alternative to ACID is BASE:

- **Basic Availability:** System does not guarantee availability.
- **Soft-state**: System changes over time, even without input.
- **Eventual consistency**: The system will become consistent over time.

The NoSQL database provides options for database developers to fulfill their specific requirements. This, however, is both good and bad at the same time. This is good because it can design the system according to the requirements, and bad because there are several options available to it and it can make an incorrect choice in designing the system.

We all know that transactions are a major feature of relational databases. Our development methods are used in such a way to this feature that we cannot imagine databases without transactions. However, most NoSQL databases do not provide transaction support by default, which means the developers have to think of ways of implementing transactions.

ii) Sharding

Database sharding can be defined as a partitioning scheme for large databases distributed across various servers, and is responsible for new levels of database performance and scalability. It divides a database into smaller parts called 'shards' and replicates those across a number of distributed servers. Figure displays the technique of sharding:

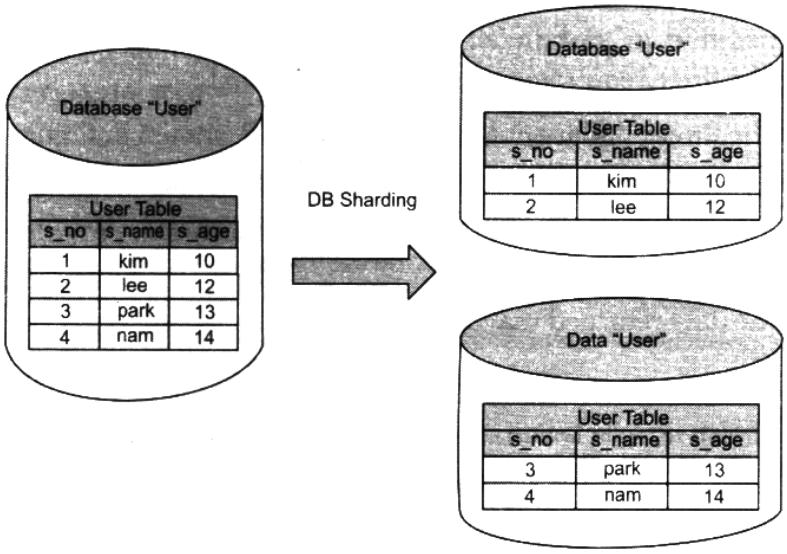


Fig: Database Sharding

The term 'sharding' was coined by Google engineers, and gained popularity through their publication of Big Table architecture. The concept of 'shared-nothing' database partitioning has been in use for more than a decade, and many implementations were provided for it over this period, specifically as high profile developed as in-house solutions by Internet leaders such as eBay. Amazon, Digg, YouTube, Facebook, Friendster, and Wikipedia. Database sharding has been gaining popularity due to the extensive growth in transactional volume and size of business application databases. It is popular among many successful e-service providers, Software as a Service (SaaS) companies, and social networking websites.

Database sharding employs a scalable approach for improving the throughput and overall performance of high-transaction, large database-centric business applications. Since the inception of RDBMS, application engineers and architects have been looking for ways to increase performance and capacity of the systems.

The fact is that business databases are continuously growing in terms of size, especially due to the commencement of the Internet economy, the Information Age, and the introduction of high-volume electronic commerce. As the size and transaction volume of databases increase in a linear pattern, the response time also grows logarithmically, as shown in Figure.

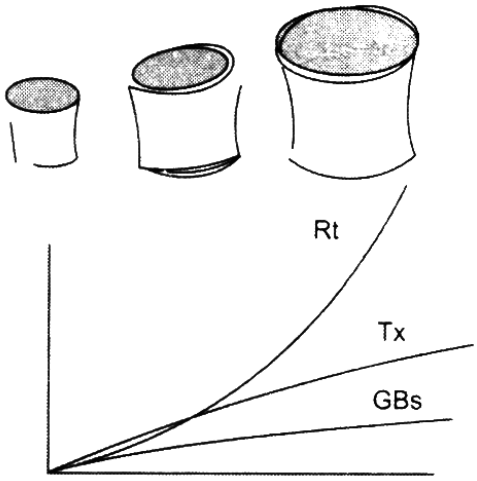


Fig.: Growth in Database Transactions and Volumes

The fundamental design of a database management system is based on two factors —performance and scalability. Databases are heavily dependent on the three components of a computer, i.e. CPU, memory, and disk space.

We know that each of these components on a single server can only scale up to a certain point, and then other measures have to be taken. While it is dear that disk I/O is a primary bottleneck as database management systems have improved, they also continue to take greater advantage of CPU and memory.

Therefore, as business applications gain sophistication and continue to grow in demand, architects, developers, and database administrators are faced with the constant challenge of maintaining database performance for critical systems. This is one of the major aspects that drive the need for database sharding.

FACULTY OF INFORMATICS

M.C.A II Year IV-Semester Examination

MODEL PAPER - I

BIG DATA ANALYTICS

Time : 3 Hours] [Max. Marks : 70

ANSWERS

1. (a) What is Big Data? **(Unit-I, Q.No.1)**

   (b) Explain about the layers of computing. **(Unit-I, Q.No.18)**

OR

2. (a) What are the Advantages of Big Data Analytics? **(Unit-I, Q.No.10)**

   (b) Discuss about cloud computing model. **(Unit-I, Q.No.20)**

3. (a) Explain about Mapreduce programming model. **(Unit-II, Q.No.3)**

   (b) Explain about Zookeeper. **(Unit-II, Q.No.25)**

OR

4. (a) What is Sqoop? **(Unit-II, Q.No.24)**

   (b) How does Hadoop Function work in big data? **(Unit-II, Q.No.4)**

5. List the features of MapReduce. **(Unit-III, Q.No.2)**

OR

6. Discuss the various layers in big data architecture. **(Unit-III, Q.No.8)**

7. Explain the relationship between RDBMS and Big Data. **(Unit-IV, Q.No.7)**

OR

8. Discuss about Polyglot Persistence. **(Unit-IV, Q.No.15)**

9. Compare relational databases with NoSQL databases. **(Unit-V, Q.No.7)**

OR

10. Discuss about CAP Theorem. **(Unit-V, Q.No.16)**

FACULTY OF INFORMATICS

M.C.A II Year IV-Semester Examination

MODEL PAPER - II

BIG DATA ANALYTICS

Time : 3 Hours] [Max. Marks : 70

ANSWERS

1. (a) What is Analytics? **(Unit-I, Q.No.8)**

   (b) Discuss the challenges with unstructured data. **(Unit-I, Q.No.5)**

OR

2. (a) Discuss the role cloud providers play in handling big data market. **(Unit-I, Q.No.28)**

   (b) Explain about various web services involved in cloud computing. **(Unit-I, Q.No.23)**

3. (a) Define Hadoop. **(Unit-II, Q.No.1)**

   (b) Explain briefly about Pig Latin. **(Unit-II, Q.No.22)**

OR

4. (a) Explain briefly about HDFS Architecture. **(Unit-II, Q.No.8)**

   (b) What is HIVE ? Write its features and characteristics. **(Unit-II, Q.No.17)**

5. Explain various Techniques to optimize MapReduce Jobs. **(Unit-III, Q.No.4)**

OR

6. Explain the functioning of Ingestion Layer in the big data architecture. **(Unit-III, Q.No.10)**

7. Explain the issues with Relational Model. **(Unit-IV, Q.No.9)**

OR

8. How big data integrating with traditional warehouse. **(Unit-IV, Q.No.16)**

9. Explain the Use of NoSQL in industry sector. **(Unit-V, Q.No.6)**

OR

10. Discuss in detail about Graph Database. **(Unit-V, Q.No.13)**

FACULTY OF INFORMATICS

M.C.A II Year IV-Semester Examination

MODEL PAPER - III

BIG DATA ANALYTICS

Time : 3 Hours] [Max. Marks : 70

ANSWERS

1. (a) Explain the relationship of big data with other areas? **(Unit-I, Q.No.13)**
   (b) Discuss the role cloud services play in handling big data. **(Unit-I, Q.No.27)**

OR

2. (a) List out the differences between distributing and parallel system. **(Unit-I, Q.No.16)**
   (b) What are the applications of Big Data? **(Unit-I, Q.No.11)**
3. (a) What are the advantages of Hadoop? **(Unit-II, Q.No.5)**
   (b) State the benefits of OOZIE. **(Unit-II, Q.No.28)**

OR

4. (a) Explain about Flume. **(Unit-II, Q.No.26)**
   (b) Explain about Combining HBase and HDFS. **(Unit-II, Q.No.15)**
5. Explain the Characteristics of HBase. **(Unit-III, Q.No.7)**

OR

6. Explain briefly about Physical Infrastructure Layer in big data architecture. **(Unit-III, Q.No.12)**
7. What are called Non-Relational Database? **(Unit-IV, Q.No.11)**

OR

8. Explain briefly about Big Data Analysis and Data Warehouse. **(Unit-IV, Q.No.17)**
9. Explain in detail Document Data Model. **(Unit-V, Q.No.11)**

OR

10. (a) What is NoSQL? **(Unit-V, Q.No.1)**
    (b) What are the Characteristics of NoSQL? **(Unit-V, Q.No.2)**