Shivani®

EDUCATING PEOPLE®

Complete Book®

For

Engineering Students

July 2020

As Per New
Scheme & Syllabus
(AICTE Flexible Curricula)

New Edition

VII Semester

R.G.P.V. Examination Papers Completely Solved

Big Data

Smart Accurate Comprehensive

om others

y R.G.P.V.

ions.

nt friendly.

uage with

e without

approach

degree of

included

iting of
ns for
eet all
ons.

Syllabus

BIG DATA

UNIT-I – Introduction to big data, Big data characteristics, Types of big data, Traditional versus big data, Evolution of big data, challenges with Big Data, Technologies available for Big Data, Infrastructure for Big data, Use of Data Analytics, Desired properties of Big Data system.

UNIT-II – Introduction to Hadoop, Core Hadoop components, Hadoop Ecosystem, Hive Physical Architecture, Hadoop limitations, RDBMS Versus Hadoop, Hadoop Distributed FIle sytem, Pocessing Data with Hadoop, Managing Resources and Application with Hadoop YARN, MapReduce programming.

UNIT-III – Introduction to Hive, Hive Architecture, Hive Data types, Hive Query Language, Introduction to Pig, Anatomy of Pig, Pig on Hadoop, Use Case for Pig, ETL Processing, Data types in Pig, running Pig, Execution model of Pig, Operators, functions, Data types of Pig.

UNIT-IV – Introduction to NoSQL, NoSQL Business Drivers, NoSQL Data architectural patterns, Variations of NoSQL architectural patterns, using NoSQL to Manage Big Data. Introduction to MongoDB.

UNIT-V – Mining social Network Graphs – Introduction, Applications of social Network mining, Social Networks as a Graph, types of social Networks, Clustering of social Graphs, Direct Discovery of communities in a social graph, Introduction to recommender system.

Price : Rs. 90.00 (Rs. Ninty Only)

Edition : 2020

Contents

BIG DATA

UNIT 1

INTRODUCTION OF BIG DATA

INTRODUCTION TO BIG DATA, BIG DATA CHARACTERISTICS, TYPES OF BIG DATA, TRADITIONAL VERSUS BIG DATA, EVOLUTION OF BIG DATA, CHALLENGES WITH BIG DATA

Q.1. What is big data ? Explain.

Ans. "Data of a very large size and typically to the extent that its manipulation and management present significant logistical challenges" is known as big data.

The technologies and initiatives that involve data that is too diverse, fast-changing or massive for conventional technologies, skills and infrastructure to address efficiently is referred to as big data. The data sets that are so large, complex, and impractical to manage with traditional software tools are described by big data.

But now the information from big data can be analyzed by using new technologies e.g., user web clicks can be tracked by retailers to identify behavioural trends that improve campaigns, pricing and stockage.

Major web companies such as Google, Amazon, and Facebook pioneered businesses built on monetizing massive data volumes over the last decade. The new paradigms not only for extracting value from data but also for managing data and compute resources from data center design, to hardware, to software, to application provisioning were invented by them

Another definition of big data is as follows –

"The collection, processing, discovery, analysis and storage of large volumes and disparate types of data is enabled by the emerging technologies and practices, very quickly and cost effectively".

Q.2. What is the importance of big data ?

Ans. The importance of big data depends upon its utilization. Data can be fetched from any source and analyzed to solve that enable us in

terms of –

(i) Cost reductions

(ii) Time reductions

(iii) New product development and optimized offerings, and

(iv) Smart decision making.

Combination of big data with high powered analytics, can have great impact on a business strategy in following ways –

(i) Finding the root cause of failures, issues and defects in real time operations.

(ii) Generating coupons at the point of sale seeing the customer's habit of buying goods.

(iii) Recalculating entire risk portfolios in just minutes.

(iv) Detecting fraudlent behaviour before it affects and risks our organization.

Q.3. Write short note on Drivers for big data.

Ans. There are three contributing factors or drivers for big data. These drivers are consumers, automation and monetization.

More than each of these contributing factors, their interaction is speeding the creation of big data. With increasing automation, it is easier to offer big data creation and consumption opportunities to the consumers and the monetization process is increasingly providing an efficient marketplace for big data. These drivers are explained below –

(i) Sophisticated Consumers – The increase in information level and the associated tools has created a new breed of sophisticated consumers. These consumers are far more analytic, far savvier at using statistics, and far more connected, using social media to rapidly collect and collate opinion from others.

(ii) Automation – Marketing and sales have received their biggest boost in instrumentation from Internet-driven automation over the past 10 years.

Browsing, shopping, ordering, and customer service on the web not only has provided tremendous control to user but also has created an enormous flood of information to the marketing, product and sales organizations in understanding the buyer's behavior. Each sequence of web clicks can be collected, collated and analyzed for customer delight, puzzlement, dysphoria, or outright defection. More information can also be obtained about sequence leading upto a decision.

(iii) Monetization – A big data analytics perspectiver, a "data bazar" is the biggest enabler to create an external market place where we collect,

exhange, and sell customer information. We are seeing a new trend in the market place, in which customer experience from one industry is anonymized, packaged, and sold to other industries.

Q.4. Discuss four V's and fiveV's characteristics of big data with suitable examples.

Ans. Big data is important because it enables organizations to gather, store, manage, and manipulate vast amounts of data at the right speed, at the right time, to gain the right insights. In addition, Big data generators must create scalable data (volume) of different types (variety) under controllable generation rates (velocity), while maintaining important characteristics of the raw data (veracity), the collected data can bring to the intended process, activity or predictive analysis/hypothesis. Indeed, there is no clear definition for 'Big Data'. It has been defined based on some of its characteristics. Therefore, these five characteristics have been used to define Big Data, earlier known as 4V's (volume, variety, velocity and veracity), as illustrated in fig. 1.1.

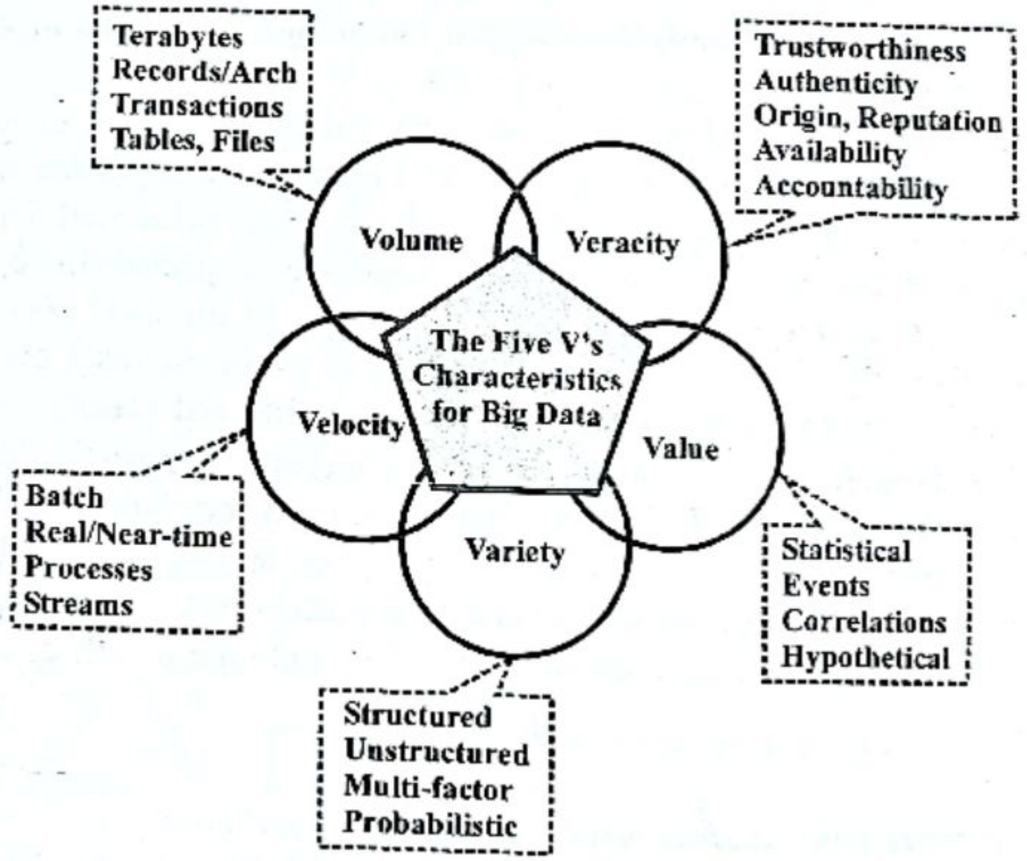

Fig. 1.1 Five V's Big Data Characteristics

(i) Volume – It refers to the quantity of data gathered by a company. This data must be used further to gain important knowledge. Enterprises are awash with ever-growing data of all types, easily amassing terabytes even petabytes of information (e.g., turning 12 terabytes of tweets per day into improved product sentiment analysis; or converting 350 billion annual meter readings to better predict power consumption).

Moreover, Demchenko, Grosso, de Laat and Membrey stated that volume is the most important and distinctive feature of Big Data, imposing specific requirements to all traditional technologies and tools currently used.

(ii) Velocity – It refers to the time in which Big Data can be processed. Some activities are very important and need immediate responses, which is why fast processing maximizes efficiency. For time-sensitive processes like fraud detection, Big data flows must be analyzed and used as they stream into the organizations in order to maximize the value of the information (e.g., scrutinize 5 million trade events created each day to identify potential fraud; analyze 500 million daily call detail records in real-time to predict customer churn faster).

(iii) Variety – It refers to the type of data that big data can comprise. This data may be structured or unstructured. Big data consists different types of data, including structured and unstructured data such as text, sensor data, audio, video, click streams, log files and so on. The analysis of combined data types brings new problems, situations, and so on, such as monitoring hundreds of live video feeds from surveillance cameras to target points of interest, exploiting the 80% data growth in images, video and documents to improve customer satisfaction.

(iv) Value – It refers to the important feature of the data which is defined by the added-value that the collected data can bring to the intended process, activity or predictive analysis/hypothesis. Data value will depend on the events or processes they represent such as stochastic, probabilistic, regular or random. Depending on this the requirements may be imposed to collect all data, store for longer period (for some possible event of interest), etc. In this respect data value is closely related to the data volume and variety.

(v) Veracity – It refers to the degree in which a leader trusts information in order to make a decision. Therefore, finding the right correlations in Big Data is very important for the business future. However, as one in three business leaders do not trust the information used to reach decisions, generating trust in big data presents a huge challenge as the number and type of sources grows.

Q.5. Explain big data types with examples.

Or

Write short note on structured and unstructured data.

[(R.G.P.V., May 2019 (VIII-Sem.)]

Ans. Big data encompasses everything, from dollar transactions to tweets to images to audio. Therefore, taking advantage of big data requires that all this information to be integrated for analysis and data management. This is more difficult than it appears. Big data includes huge volume, high velocity, and extensible variety of data. There are three types of data concerned here –

(i) Structured Data – This is the data stored in relational databases table in the format of row and column. They have fixed structures and these

structures are defined by organizations by creating a model. The model allows to store, process as well as gives permission to operate the data. The model defines the characteristics of data including data type and some restrictions on the data. Analysis and storing of structured data is very easy. Because of high cost, limited storage space and techniques used for processing, causes RDBMS the only path to store and process the data effectively. Programming language called structured query language (SQL) is used for managing this type of data.

(ii) Semi-structured Data – Data which is in the form of structured data but does not fit the data model is semi-structured data. It cannot be stored in the form of data table, but it can be stored in some particular types of files which hold some specific markers or tags. These markers are distinguished by some specific rule and the data is enforced to be stored with a ranking. This form of data increased rapidly after the introduction of the World Wide Web where various form of data need medium for interchanging the information like XML and JSON.

Example – CSV, XML and JSON documents are semi-structured documents, NoSQL databases are considered as semi-structured.

(iii) Unstructured Data – Data without any specific structure and due to this could not be stored in a row and column format is unstructured data. This data is contradictory to that of structured data. It cannot be stored in a databank. Volume of this data is growing extremely fast which is very tough to manage and analyze it completely. To analyze the unstructured data advanced technology knowledge is needed.

Fig. 1.2, depict these types of big data along with example.

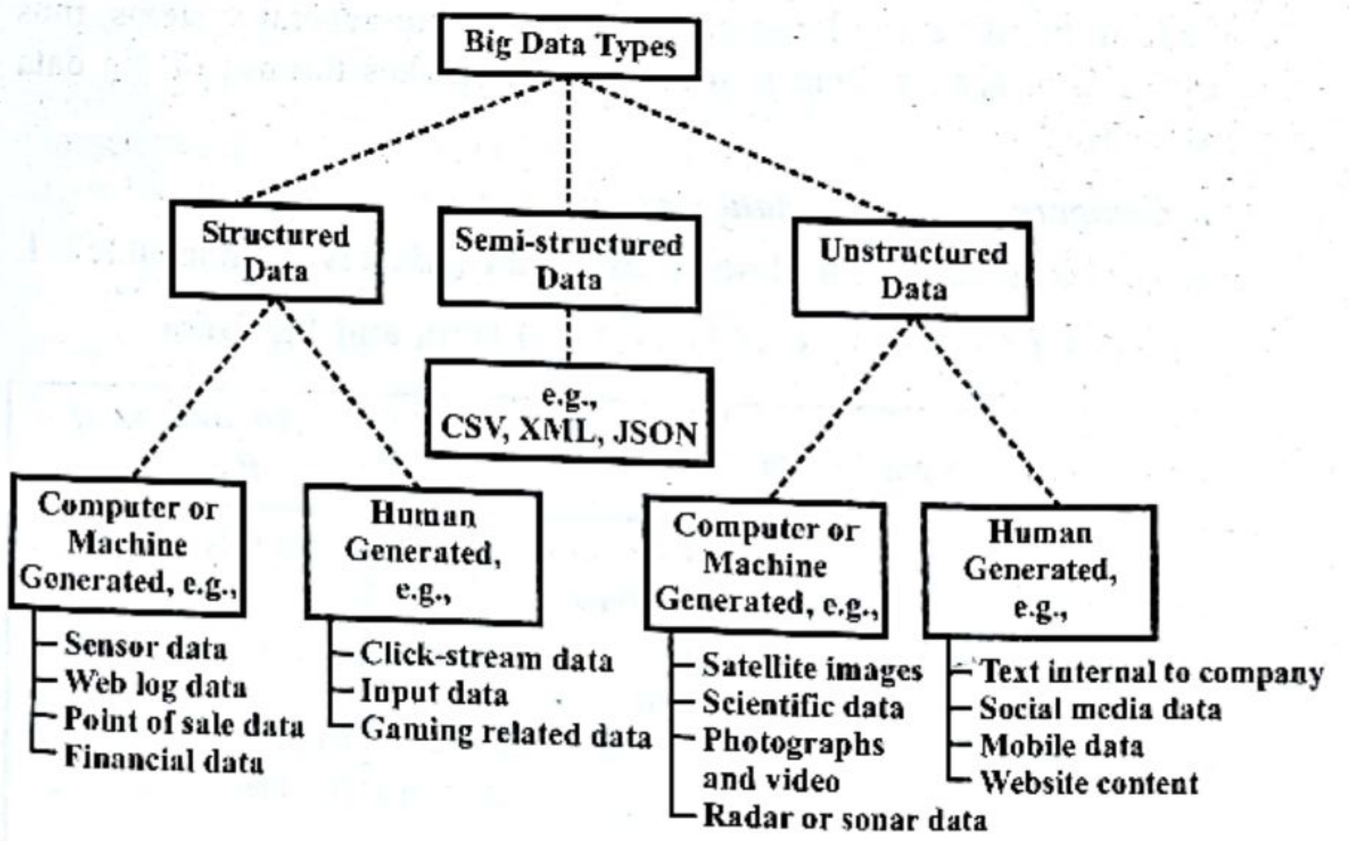

Fig. 1.2 Big Data Types

Q.6. Give advantages of big data over traditional data.

Ans. Traditional data systems such as relational databases were used as major source for storing and analyzing data of business and organizations about 30-40 years back. The systems were designed primarily for handling structured data and the main characteristics of that system was that it was highly organized data. Traditional data solves large and complex problems in a single computer. It used centralized architecture which is costly and ineffective for large data sets, whereas big data uses distributed database architecture. In this architecture large blocks of datasets are divided in small sets and solved, solution to the given problem is calculated by different computers present in a given computer networks. These computers communicate with each other and find the solution to the given problem. Distributed database is in lower price, improves performance as well as provide better computing. Distributed architecture is based on microprocessors which is economical as compared to centralized database which is based on mainframe and distributed database has more computational power as compared to traditional. Traditional database systems are based on structured data whereas big data uses semi as well as unstructured data. Traditional database store small amount of data which range from some giga-bytes to terabyte however big data can store and analyze data ranging from hundreds of terabytes or petabytes and more. Storing large amount of data reduces the cost which will help the business intelligence (BI). Big data uses dynamic schema, whereas traditional database uses fixed schema, which cannot be changed once saved. Traditional database system requires complex and expensive software and hardware for managing large amount of data. While in big data, the large data is divided into several systems, thus amount of data in each system is reduced. This makes the use of big data simple and cheap.

Q.7. Compare traditional data and big data.

Ans. The comparison of traditional data and big data is given in table 1.1.

Table 1.1 Comparison of Traditional Data and Big Data

	Traditional Data	*Big Data*	*Advantage of Big Data*
Data architecture	Centralized database	Distributed database	Cost effective
Types of data	Structured data	Unstructured and semi-structured	Improves variety
Volume	Small amount of data. Range – Giga-byte-terabytes	Large amount of data. Range – <petabytes	Cost reduces and help business intelligence

Data schema	Fixed schema	Dynamic schema	Preserves the information in data
Data relationship	Relationship with data is explored easily.	Difficulty in relationship between data items	—
Scaling	More than one server for computing	Single server for computing	Cost effective
Accuracy	Less accurate results	High accurate results	Confident results and reliable

Q.8. Describe history of big data.

Ans. Big data is a long evolution of capturing and using of data and not a new phenomenon. Big data is the future act that will bring change in the way we run society, just like the other developments in storage of data, processing of data and internet. The ancient history of data is when humans used tally sticks for storing and analysis of data about C 1800 BCE. The tribal peoples used to mark notches into bones or sticks for calculations, which would make them predict about how long their food would last. One of the earliest prehistoric data storage is Ishango Bone now known as Uganda which was discovered in 1960. Then in C 2400 BCE came the very first device particularly for performing calculations – Abacus. Our first libraries also appeared in this time period which represented our initial step towards mass storage. Then in the period of 300 BC-48 AD the library containing largest collection of data of the historic world which covered pretty much everything which we learned so far was destroyed by Romans accidentally. Then the earliest mechanical computer was developed by Greek from C 100-200 AD whose CPU consist of 30 bronze gears. It was designed for astrological purposes and tracking cycle of Olympic Games. After this many small discoveries laid the foundation to emergence of statistics like first recorded experiment in statistical data analysis. In 1880, Hollerith Tabulating Machine was developed that used punch cards for calculation purposes that completed 10 years of work in 3 months designed by Herman Hollerith known as the father of automated computation etc. Then started the early stage of modern data storage. In 1928 a German-Austrian engineer Fritz Pfleumer invented a magnetic tape which stored information magnetically. Then came the Business Intelligence and start of large data centers where ideas of relational database and Material Requirement Planning systems were out forward.

In 1989 the first use of the term big data was made by Erik Larson in the Harpers Magazine where he said that "The keepers of big data say they are doing it for the consumer's benefit. But data have a way of being used for purposes other originally intended". The birth of World Wide Web took place that kicked internet into gear in 1991. Google search engine started in the year

1997. After a couple of years in 1999 big data term appeared in a research paper published by the Association for Computing Machinery. In that storing large amounts of data and inadequate space for storage as well as analysis difficulties were highlighted. In 2001, characteristics of big data like volume, velocity, variety, were defined by Dough Laney. In 2005, creation of an open source framework called Hadoop took place for storing and analyzing big data sets. Hadoop was famous for its flexibility and management of both structured and unstructured data. Life on earth evolved around 4 billion years ago, in which over last 6 million years human evolution occurred, out of which about 100000 years ago human language evolution started then about 70000 years ago cognitive evolution started and then finally was the scientist evolution which happened about 500 years ago and fortunately analytics evolution is about just 30 years old but still unfold. It was started in 1990s as Analytics 1.0 also known as 'Business Intelligence'. In this data about production process, interaction of customer etc. were collected, combined and analyzed by traditional databases where data used to fit neatly and stored in rows and columns. In Analytics 1.0 era, more time was spent on preparing data for analysis than analytics itself by IT & Business Analytic. Then in 2000s came the Analytics 2.0 or well known as big data which had complex queries that had views of both structured as well as unstructured data. To deal with such fast processing across parallel servers software like Hadoop, NoSQL etc., have been developed.

Q.9. Explain in detail about challenges in big data.

Ans. There are six major challenges areas in big data; those areas are shown in fig. 1.3.

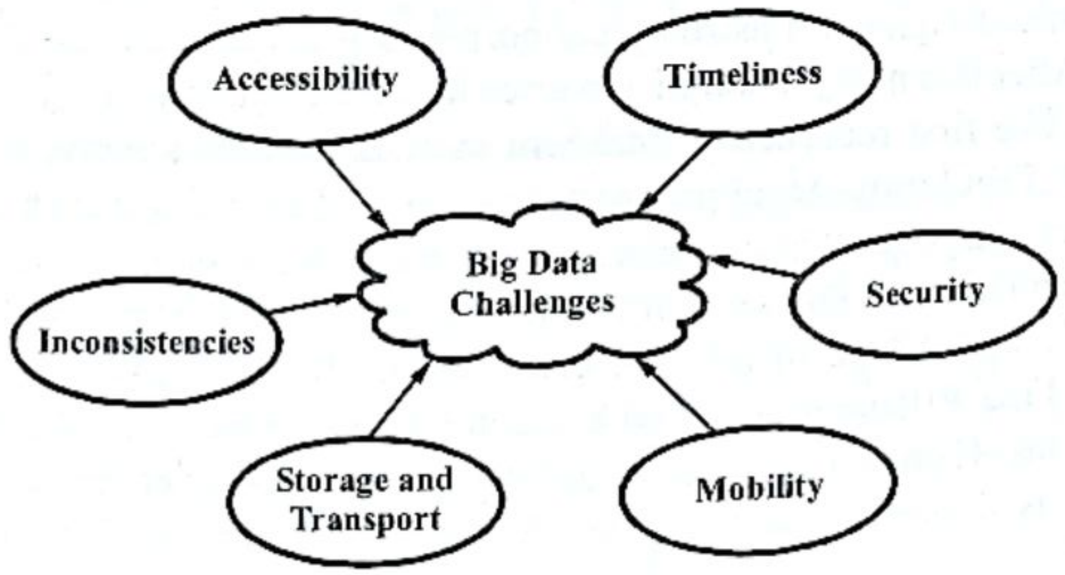

Fig. 1.3 Major Challenges in Big Data

(i) Timeliness – Timeliness is one of the challenges of big data. As volume of the data increases, the time taken to analyze the data will also increase. Some cases like analyzing fraudulent activity need very fast processing. But it is not possible to get a full analysis of the fraudulent activity

before the fraudulent transactions can occur. So, more the volume and variety of data is, the more time is needed for a complete analysis of big data. So companies like bank and security companies face timeliness challenges of analyzing the big data.

(ii) Security – Big data is worried with a considerable measure of utilization cases such as staging, pre-handling, processing, meta information stockpiling and to store fleeting and long haul truth information. For serving every usage case multi-elements of establishment required. Secure and private trades are the two significant stress of IT. Be that as it may, the security and assurance transforms into a question mark as the truths volume of gignatic data rapidly create. When we think the security perspective, the accessible cryptography benchmarks cannot gather the requests of enormous information. Subsequently, security insurance is still one additionally difficult issue in enormous information.

(iii) Storage and Transport – Big data stores and oversee information in various courses from the customary information distribution centers. It envelops substantial sensor information; crude and semi-organized enlist information of IT businesses and the detonated measure of information from online networking. In addition, data is being made by it is conceivable that one or by all (i.e. from PDAs to supercomputers and by specialists, analysts, editorialists, researches, etc.).

(iv) Accessibility – The rapid pace of development in information on the web challenges the scientists to advance proficient calculations and preparing advances. The strategy for getting to enormous information has two appearances. First, the information in the source side and communicate comes about. In this way, the upgrade of scripting innovations on the program side is required to bring fundamental code from the server. Second, communicating just the genuine information in the wake of applying legitimate channels.

(v) Inconsistencies – The range of this review i.e., big data is encompassed with multi-dimensional, specialized and precise spaces. The target of big data likewise changes over partner to partner. The big data analytics is a rising edge for development and progression of innovations. In this way, its effect on society ought not to be maintained a strategic distance from. This appears to be evident that sooner the big data would withdraw to wrap all these areas and parts like special sciences, lite and physical sciences, communique, capital and so forth. Big data includes every single space; in this manner irregularity exists either in information level, data level, or learning level. This inconsistency in every level must be tended to. Irregularity has been separated in four categories i.e., temporal, textual, spatial and functional inconsistency.

(vi) Mobility – Organizations are ready to expand more resources in applications which are being acknowledged by cell phones. The incorporation of versatile processing advancements and endeavor advancement in technology has an awesome potential to swell efficiency in business settings. The heightening area based datasets, invasion of information from versatile applications, their measurement and assortment surpasses the ability of spatial and portable figuring advancements. The online conduct of various versatile clients has contributed a considerable measure to the data analytics. This union of custom controlling organizations (checking GPS and spatial data) into the gigantic data model could defy a couple of significant issues; in any case, the development in computational cost since its augmentation impacts the guiding request to PDAs, secondly, it uses geographical thinking in remotely identifying and conclusion after some time and space. The inalienable development locators in mobile phones begin a colossal measure of data from every customer's life.

Q.10. Discuss the issues in big data.

Ans. The issues in big data are very few and while adopting the technology competently, one should clearly know by its organization. These data issues are discussed below –

(i) Issues Related to the Characteristics –

(a) Data Volume – As the data size increases, the estimation of different data records diminish in ratio to age, sort, riches, and amount amidst different elements. The long range interpersonal communication destinations accessible are themselves delivering information in terabytes over and over this measure of information is obviously difficult to handle with the current customary frameworks.

(b) Data Velocity – The customary frameworks are not sufficiently skilled to play out the investigation on the information which is persistently changing or expanding. The rising online business has immediately expanded the speed and fortune of information.

(c) Data Variety – The data comes in various arrangements, for example, crude, organized, semiorganized, and unstructured data, these distinctive configurations are difficult to handle by the reachable customary explanatory frameworks. From the investigative perspective the disappointment of customary scientific framework is a principle snag to successfully utilize the immense volume of information. In any case, jumbled data designs, unbiased data structures, and confused information semantics speak to unimportant difficulties that can prompt to investigative fall.

(d) Data Value – As the stored data is utilized by various associations for data analytics, which made a sort of crack between business pioneers and IT specialists, as business pioneers needs to build the estimation

of their business and pick up progressively advantage not in any manner like the IT pioneers who have stress with the subtle elements of the limit and planning.

(ii) Storage and Transport Issues – The measure of information has exploded every time we have fanciful once more stockpiling medium. The uniqueness about the most current data explosion, primarily because of web-based social networking, is that no new storage medium has been introduced. Exabyte of data could be set up on a single PC edge work; it is unfit to clearly join the basic number of circles. Access to that data would overcome current correspondence frameworks. Tolerating that a 1 gigabyte for consistently framework has an accommodating viable conversion standard of 80%, the reasonable transmission limit is around 100 Megabytes.

(iii) Data Management Issues – The data information management is the most complex issue while cooking enormous information. Settling issues of perfect to use, utilize, redesigning, organization, and reference. The well springs of the information are varied by size, by setup, and by technique for social occurrence.

Main data management issues are –

(a) Data privacy (b) Security
(c) Ethical (d) Governance.

(iv) Processing Issues – Let an exabyte of data ought to be prepared completely and arranged orderly. For simplicity, consider the data is pieced into squares of 8 words, so 1 exabyte = 1 K petabytes. Expecting a processor utilizes 100 rules on one square at 5 gigahertz, the time required for end-to-end get ready would be 20 nanoseconds. To manipulate 1K petabytes would require a total pier-to-pier preparation time of around 232000 days. In this way, capable get ready of exabyte of data will require wide parallel taking care of and new examination computations remembering the true objective to give fortunate and huge.

(v) Processing Major Issues –

(a) Gathering required data/information

(b) Arranging data from different resources (e.g., resolution when two entities are same)

(c) Changing the data into a form suitable for inspection

(d) Modelling it, whether arithmetically, or through some form of simulation

(e) Understanding the output, visualize and distribute the results, think for a second how to display multifaceted analytics on an iPhone or a mobile device.

Q.11. Discuss the security challenges of big data.

Ans. The exponential growth of the data has opened many opportunities for researchers, students, and industries as well as cyber criminals. Cyber criminals can destroy both the industry and its customers with a single opportunity. The effect of weak securities can lead to the destruction of industry's reputation and may be subject to millions of dollars loss as a data breach settlement. So, the industries collecting large volume of data should be aware of security challenges while storing and processing the big data. Some of the security challenges are explained below –

(i) Privacy – Privacy is considered as one of the primary security challenges of big data. These days lots of companies trades customer sensitive information based on the user's location and preferences. Hackers can easily track the identity of the users by analyzing the location and pattern of user's activity. Once they are able to track the personal information, they can use that information to create duplicate debit and credit cards to be sold online. When the data are transferred from one company to another, there should be some guarantee that the company that receives the data will fairly use the data. Sometimes the fair use of data results personal harm too.

One of the retailers was tracking the shopping habits of customers and concluded the teenage customer was pregnant. The retailer started sending deals and coupons related with the pregnancy products in her mailing address thinking it might be useful to that customer. These deals and coupons unintentionally disclosed her father about the pregnancy. Even though the retailer used an accurate result from the analysis, it violated the privacy of the teenage customer. Future data scientists should be aware of consequences of using information from the analysis of big data.

(ii) Quantity of Loss Affected from the Security Breaches – Other potential security challenges of big data are the amount of loss affected from the security breach. The more data is generated, the more devastating consequences will result from the data compromised than that we have from the normal data.

Another security challenge of big data is maintaining the granular access of the big data. There will be lots of users trying to access to the big data. Being large in size, big data needs more work to classify the importance of the data and decide whom to give access of it.

Most companies collect data from various sources and store at the central storing site. Data contains very sensitive information since it may contain the information of people, employee, financial information, and trade secret. This sensitive information might be potentially vulnerable from attacker since they can easily attack to the central database.

Additional security challenges of big data are maintaining the compliance with the law. Some law restricts where the data should be stored and processed. Since company has to store and transfer data over the Internet, it is potentially vulnerable for companies to maintain the law. They might face security issues, or they might be violating the privacy of individuals without noticing it.

TECHNOLOGIES AVAILABLE FOR BIG DATA, INFRASTRUCTURE FOR BIG DATA, USE OF DATA ANALYTICS, DESIRED PROPERTIES OF BIG DATA SYSTEM

Q.12. Explain big data technologies.

Ans. Using modern computing technology, businesses may now manage immense volumes of data previously could dealt with using expensive supercomputers. These are now much cheaper. As a result, new techniques for distributed computing are main stream. Big data became paramount as companies such as Yahoo!, Google, and Facebook came to the realization that they required help in monetizing the massive amounts of data their offerings were creating. Thus, these new companies must search for new technologies to store, access, and analyze huge amounts of data in near real time. Such real-time analysis is required in order to profit from so much data from users. Their resulting solutions have affected the larger data management market. In particular, the innovations MapReduce, Hadoop, and Big Table have proven lead to a new generation of data management. These technologies will allow businesses to address one of the most fundamental problems, namely the compability to process massive amounts of data efficiently, cost-effectively, and quickly.

(i) MapReduce – MapReduce was designed by Google to efficiently carry out a set of functions against a large amount of data in batch mode. The "map" component distributes the programming problem or task across a large number of systems while managing placement to balance the load and allow recovery from failures. After the distributed computation is complete, another function called "reduce" aggregates all the elements back together to provide a result. An example of MapReduce would be determining the number of pages in a book that are written in each of 50 different languages.

(ii) Big Table – Big Table was developed by Google to be a distributed storage system to manage highly scalable structured data. Data is organized into tables with rows and columns. Unlike typical relational

database models, Big Table is a sparse, distributed, persistent multidimensional sorted map. It has been designed to keep large volumes of data across commodity servers.

(iii) Hadoop – Hadoop is an Apache-managed software framework created using MapReduce and Big Table. Hadoop allows applications based on MapReduce to run on large clusters of commodity hardware. The project has become the basis for the computing architecture underlying Yahoo!'s business. Hadoop is designed to parallelize data processing across computing nodes to speed computations and diminish latency. Two major components of Hadoop exist – a massively scalable distributed file system that can support petabytes of data, and a massively scalable MapReduce engine that computes results in batches.

Q.13. Describe big data tools and techniques.

Ans. Organizations use various techniques and technologies to aggregate, manipulate, analyze and visualize big data. They come from various fields such as statistics, computer science, applied mathematics, and economics. Some of them have been developed intentionally and some of them have been adapted for this purpose.

To capture the value from big data, we need to develop new techniques and technologies for analyzing it. Until now, scientists have developed a wide variety of techniques and technologies to capture, curate, analyze and visualize big data.

Big Data Techniques – Big data needs extraordinary techniques to efficiently process large volume of data within limited run times. Reasonably, big data techniques involve a number of disciplines, including statistics, data mining, machine learning, neural networks, social network analysis, signal processing, pattern recognition, optimization methods and visualization approaches. There are many specific techniques in these disciplines, and they overlap with each other.

(i) Optimization Methods – Optimization methods have been applied to solve quantitative problems in a lot of fields, such as physics, biology, engineering, and economics.

(ii) Statistics – Statistics is the science to collect, organize, and interpret data. Statistical techniques are used to exploit correlation and causal relationships between different objectives. Numerical descriptions are also provided by statistics. However, standard statistical techniques are usually not well suited to manage big data.

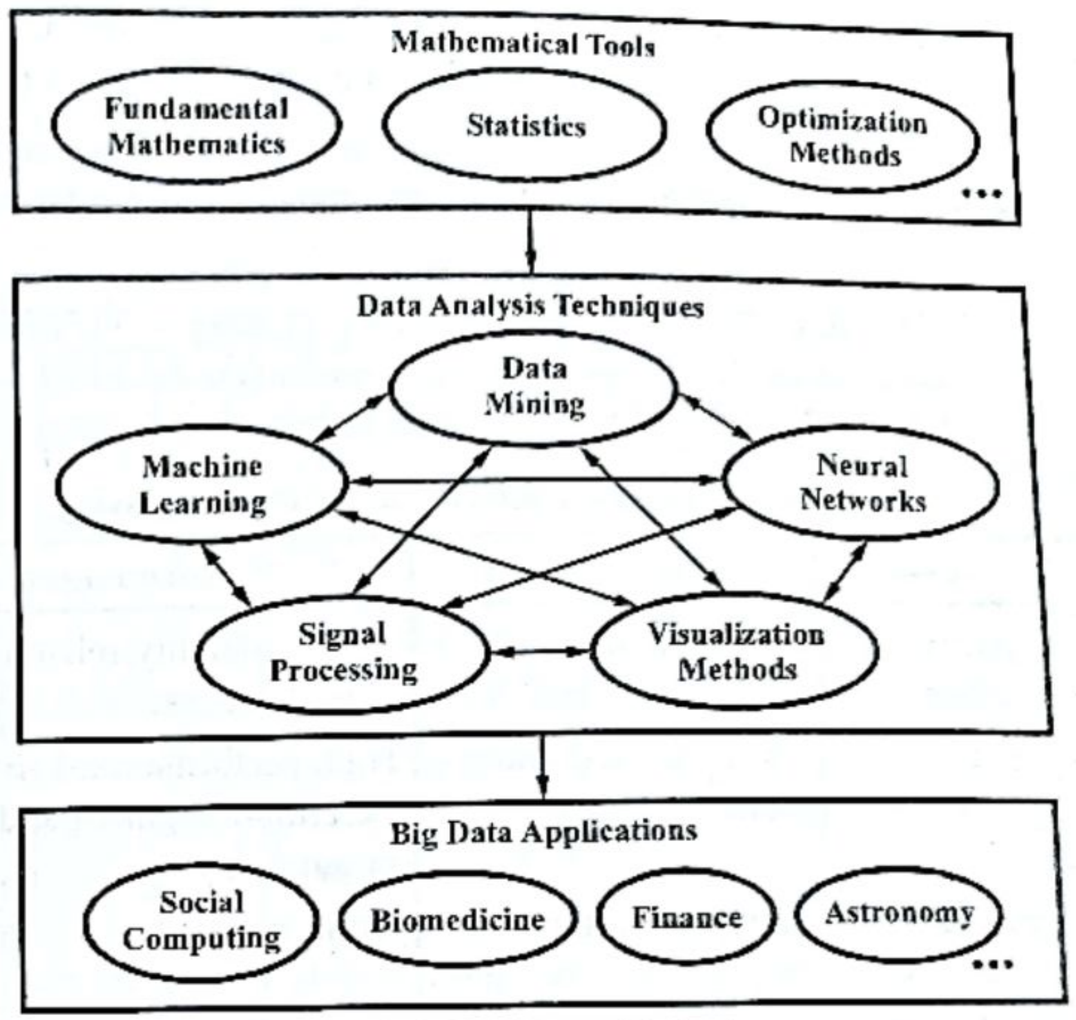

Fig. 1.4 Big Data Techniques

(iii) Data Mining – Data mining is a set of techniques to extract valuable information (patterns) from data, including clustering analysis, classification, regression and association rule learning.

(iv) Machine Learning – Machine learning is an important application of artificial intelligence which is aimed to design algorithms that allow computers to evolve behaviours based on empirical data. The most obvious characteristic of machine learning is to discover knowledge and make intelligent decisions automatically.

(v) Artificial Neural Network – Artificial Neural Network (ANN) is a mature technique and has a wide range of application coverage. Its successful applications can be found in pattern recognition, image analysis, adaptive control, and other areas.

(vi) Visualization Approaches – Visualization approaches are the techniques used to create tables, images, diagrams and other intuitive display ways to understand data.

(vii) Social Network Analysis – Social network analysis (SNA) which has emerged as a key technique in modern sociology, views social relationships in terms of network theory, and it consists of nodes and ties.

Higher level big data technologies include distributed file systems, distributed computational systems, massively parallel-processing (MPP)

systems, data mining based on grid computing, cloud-based storage an computing resources, as well as granular computing and biological computing

Big Data Tools – Current big data tools concentrate on three classes namely, batch processing tools, stream processing tools, and interactive analysi tools.

(i) Big Data Tools Based on Batch Processing – One of the mos famous and powerful batch process based big data tools is Apache Hadoop. I provides infrastructures and platforms for other specific big data applications

Table 1.2 Big Data Tools Based on Batch Processing

S.No.	*Name*	*Specified Use*	*Advantages*
(i)	Apache Hadoop	Infrastructure and platform	High scalability, reliability, completeness.
(ii)	Dryad	Infrastructure and platform	High performance distributed execution engine, good programmability.
(iii)	Apache mahout	Machine learning algorithms in business	Good maturity.
(iv)	Jaspersoft BI suite	Business intelligence software	Cost-effective, self-service BI at scale.
(v)	Pentaho business analytics	Business analytics platform	Robustness, scalability, flexibility in knowledge discovery.
(vi)	Skytree server	Machine learning and advanced analytics	Process massive datasets accurately at high speeds.
(vii)	Tableau	Data visualization, business analytics	Faster, smart, fit, beautiful and easy to use dashboards.
(viii)	Karmasphere studio and analyst	Big data workspace	Collaborative and standards-based unconstrained analytics and self service.
(ix)	Talend open studio	Data management and application integration	Easy-to-use, eclipse-based graphical environment.

(ii) Stream Processing Big Data Tools – Hadoop does well in processing large amount of data in parallel. It provides a general partitioning mechanism to distribute aggregate workload across different machines. Nevertheless, Hadoop is designed for batch processing. It is a multi-purpose engine but not a real-time and high performance engine, since there are high throughout latency in its implementations. Stream big data has high volume,

high velocity and complex data types. Indeed, when the high velocity and time dimension are concerned in applications that involve real-time processing, there are a number of different challenges to map/reduce framework. Therefore, the real-time big data platforms, like SQL stream, storm and stream cloud, are designed especially for real-time stream data analytic.

Table 1.3 Big Data Tools Based on Stream Processing

S.No.	*Name*	*Specified Use*	*Advantages*
(i)	Storm	Real-time computation system	Scalable, fault-tolerant, and is easy to set up and operate.
(ii)	S4	Processing continuous unbounded streams of data	Proven, distributed, scalable, fault-tolerant, pluggable platform.
(iii)	SQL stream s-server	Sensor, M2M, and telematics applications	SQL-based, real-time streaming big data platform.
(iv)	Splunk	Collect and harness machine data	Fast and easy to use, dynamic environments, scales from laptop to datacenter.
(v)	Apache kafka	Distributed publish subscribe messaging system	High-throughput stream of immutable activity data.
(vi)	SAP Hana	Platform for real-time business	Fast in-memory computing and real-time analytic.

(iii) Big Data Tools Based on Interactive Analysis – The interactive analysis presents the data in an interactive environment, allowing users to undertake their own analysis of information. Users are directly connected to the computer and hence can interact with it in real time. The data can be reviewed, compared and analyzed in tabular or graphic format or both at the same time.

(a) In 2010, Google proposed an interactive analysis system, named Dremel, which is scalable for processing nested data. Dremel has a very different architecture compared with well-known Apache Hadoop, and acts as a successful complement of map/reduce based computations. It has capability to run aggregation queries over trillion-row tables in seconds by means of combining multi-level execution trees and columnar data layout.

(b) Apache drill is another distributed system for interactive analysis of big data. It is similar to Google's Dremel. For drill, there is more flexibility to support a various different query languages, data formats and data sources.

Q.14. Write short note on processing of big data.

Ans. For big data processing, a large number of big data technologies have been developed and these are categorized according to data processing concepts. In processing a lot of content must be extracted and analyzed from the collected information to serve the knowledge requirements of various business organizations, political parties and scientific research departments. The process is initiated with the retrieval of information, which can come from various sources such as database, websites, documents or content management system. Hadoop, is used for storing this massive amount of data.

Before processing big information it must be recorded from different information creating sources. In the order of its happening it must be filtered and optimized. Just the pertinent information ought to be recorded by methods for channels that dispose of futile data. This can be done by specific instruments such as ETL. ETL method commonly combines data from multiple systems in which the data is actually loaded into the data warehouse.

Q.15. Describe the architecture for big data.

Ans. The architecture for big data is shown in fig. 1.5.

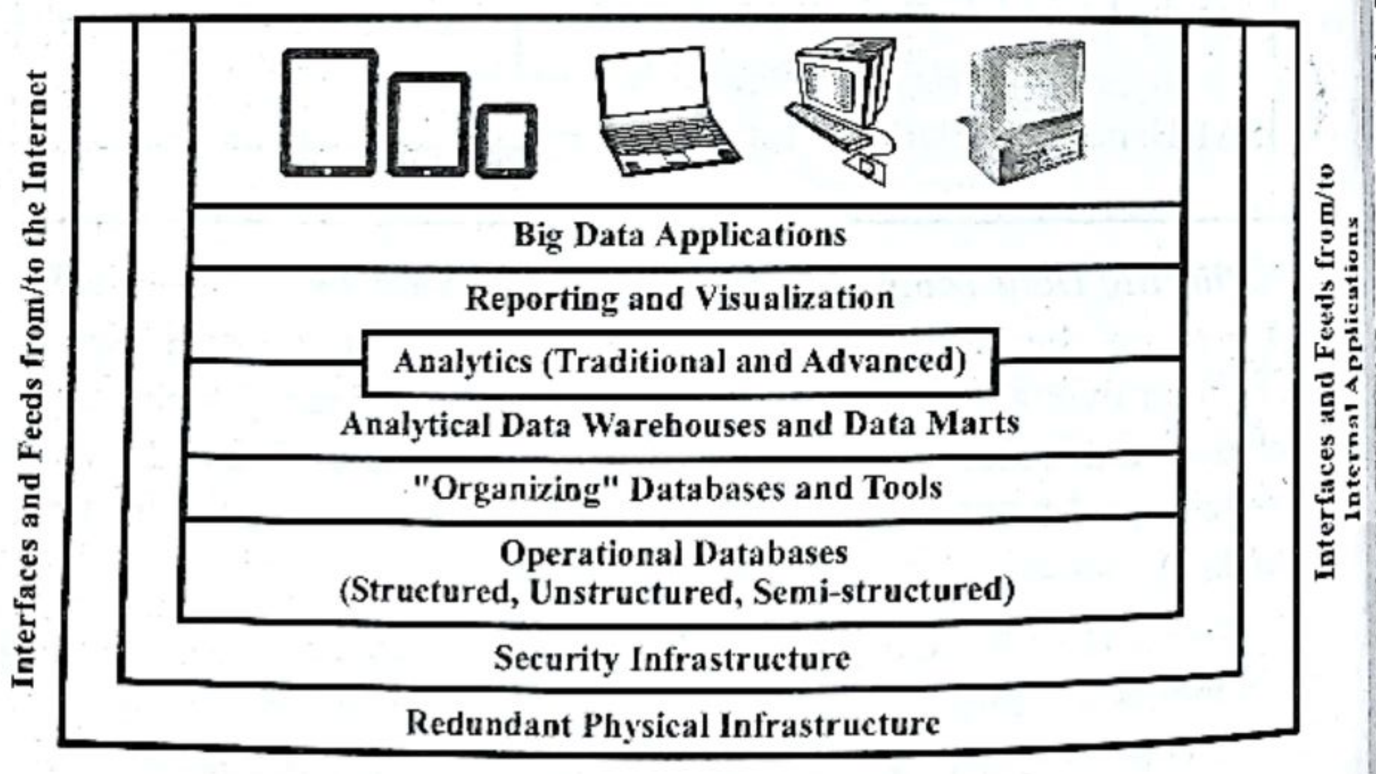

Fig. 1.5 Big Data Architecture

(i) Interfaces and Feeds – What makes big data big is the fact that it relies on picking up lots of data from lots of sources. Therefore, open application programming interfaces (APIs) are a core part of any big data architecture.

In addition, interfaces exist at every level and between every layer of the stack. Without integration services, big data cannot happen. Other important operational database approaches include columnar databases that store

information efficiently in columns, and not rows. This approach leads to faster performance, as input/output is extremely fast. When geographic data storage is part of the equation, a spatial database is optimized to store and query data based on how objects are related in real terms.

(ii) Redundant Physical Infrastructure – The supporting physical infrastructure is fundamental to the operation and scalability of a big data architecture. In fact, without the availability of robust physical infrastructures, big data would likely not have become such a strong trend. To support an unanticipated or unpredictable volume of data, a physical infrastructure for big data has to be different than that for traditional data. The physical infrastructure has been based on a distributed computing model. This means that data may be physically stored in many different locations, allowing it to be linked through networks, the use of a distributed file system, and various big data analytic tools and applications.

Redundancy is important, as companies must handle a great deal of data from many sources. Redundancy comes in many forms. For instance, if the company has created a private cloud, company may want to create redundancy within private areas so that it can scale out to support changing workloads. If a company needs to limit internal IT growth, it may use external cloud services to add to its own resources. In some cases, this redundancy may come in the form of a Software as a Service (SaaS), allowing companies to carry out advanced data analysis as a service. The SaaS approach allows for a faster start at reduced costs.

(iii) Security Infrastructure – As big data analysis becomes part of workflow, it becomes vital to secure that data. For example, a healthcare company probably wants to use big data applications to determine changes in demographics or shifts in patient needs. This data about patients needs to be protected, both to meet compliance requirements and to protect patient privacy. The company needs to consider who is allowed to see the data and when they may see it. Also, the company need to be able to verify the identity of users, as well as protect the identity of patients. These types of security requirements must be part of the big data fabric from the out set, and not an after thought.

(iv) Operational Data Sources – Concerning big data, a company must ensure that all sources of data will provide a better viewpoint about the business and allow it to understand how data effects the operational methods of that company. Traditionally, an operational data source consisted of highly structured data, managed by the line of business in a relational database. However, operational data now has to consider a broader set of data sources, including unstructured sources like social media or customer data.

(v) Performance Matters – Data architecture also must work to perform according to the supporting infrastructure of organization or company.

For instance, the company might be interested in running models to determin whether it is safe to drill for oil in an offshore area, provided with real-tim data of temperature, salinity, sediment resuspension, and many other biologica chemical, and physical properties of the water column. It might take days t run this model using a traditional server configuration. However, using distributed computing model, a day's long task may take minutes. Performanc might also determine the kind of database that company would use. Und certain circumstances, stakeholders may want to understand how two ver distinct data elements are related, or the relationship between social netwo activity and growth in sales. This is not the typical query the company coul ask of a structured, relational database. A graphical database might be a bett choice, as it may be tailored to separate the "nodes" or entities from i "properties" or the information that defines that entity, and the "edge" relationship between nodes and properties. Using the right database may als improve performance. Typically, a graph database may be used in scientif and technical applications.

(vi) Organizing Data Services and Tools – Indeed, not all the da that organizations use is operational. A growing amount of data comes from number of sources that are not quite as organized or straightforward, includin data that comes from machines or sensors, and massive public and priva data sources. In the past, most companies were not able to either capture store this vast amount of data. It was simply too expensive or too overwhelmin Even if companies are able to capture the data, they do not have the tools to anything about it. Very few tools can make sense of these vast amounts data. The tools that did exist were complex to use and did not produce result within a reasonable time frame. In the end, companies who really wanted do the enormous effort of analyzing this data were forced to work wit snapshots of data. This means that stakeholders may miss out on relevan events as they may not have been captured in a certain snapshot.

(vii) Analytical Data Warehouses and Data Marts – After a compar sorts through the massive amounts of data available, it is often important take the subset of data that reveals patterns and put it into a form that's availabl to the business. Such so-called 'warehouses' provide compression, multilev partitioning, and a massively parallel processing architecture.

(viii) Reporting and Visualization – Companies have always relie on the capability to create reports to give them an understanding of what th data tells them about everything from monthly sales figures to projections growth. Big data changes the way the data is managed and used. If a compan is able to collect, manage, and analyze enough data, it may use a new generatio of tools to help management truly understand the impact not just of a collectio

of data elements, but also how these data elements offer context based on the business problem being addressed. With big data, reporting and data visualization have become tools for looking at the context of how data is related and the impact of those relationships on the future.

(ix) Big Data Applications – Traditionally, business has anticipated that data would be used to answer questions about what to do and when to do it. Data has often been integrated into general-purpose business applications.

With the advent of big data, this is changing. Now, the companies are seeing the development of applications that are designed specifically to take advantage of the unique characteristics of big data. Specific emerging applications include areas like healthcare, manufacturing management and traffic management. All of these applications rely on huge volumes, velocities, and varieties of data to transform the behaviour of a market. For example, in healthcare, a big data application might be able to monitor premature infants to determine if data indicates when intervention is needed. In manufacturing, a big data application can be used to prevent a machine from shutting down during a production run. A big data traffic management application may reduce the number of traffic jams on busy city highways, decreasing the number of accidents while saving fuel and reducing pollution.

Q.16. What do you mean by big data analytics ? Explain various types of analytics.

Ans. Big data analytics, is the process of examining large data sets that containing a variety of data types i.e., big data to uncover all hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. Then analytical findings can lead to more effective marketing, new revenue opportunities, better customer service, improved operational efficiency, competitive advantages over rival organizations and other business benefits.

The primary goal of big data analytics is to help companies make more informative business decisions by enabling data scientists, predictive modellers and other analytics professionals to analyse large volumes of transactional data, as well as other forms of data that may be untapped by more conventional business intelligence (BI) programs. That could include web server logs and Internet click stream data, social media content and social network activity reports, text from customer e-mails and survey responses, mobile phone call detail records and machine data captured by sensors and connected to the Internet of Things.

Big data burst upon the scene in the first decade of the 21st century, and the first organizations to embrace it were online and start-up firms. Arguably, firms like Google, LinkedIn, eBay and Facebook were built around big data

from the beginning. They did not have to reconcile or integrate big data w more traditional sources of data and the analytics performed upon them, becau they did not have that much of traditional forms. They did not have to mes big data technologies with their traditional IT infrastructures because the infrastructures did not exist. Big data could stand alone, big data analyt could be the only focus of analytics, and big data technology architectur could be the only architecture.

Analytics can be classified into following three types –

(i) Predictive analytics

(ii) Descriptive analytics

(iii) Prescriptive analytics.

(i) Predictive Analytics – Predictive analysis establish previous da patterns and gives list of solutions which may come for given situatio Predictive analysis study the present as well as past data and predict what m happen in future, give probabilities of what would happen. It is used to yo big data to forecast other data which we do not have. This analytical meth is one of the most commonly used methods used for sales lead scoring, soci media and consumer relationship management data.

Three basic elements of predictive analytics are as follows –

(a) Predictive modelling

(b) Decision analysis and optimization

(c) Transaction profiling.

For example, predictive analytics is used for optimizing custom relationship management systems. They can help enable an organization t analyze all customer data therefore exposing patterns that predict custom behaviour.

(ii) Descriptive Analytics – Descriptive analytics also known as da mining, operates what is happening in real-time. It is one of the simplest typ of analytics as it converts big data into small bytes. The result is monitore through e-mails or dashboard. It is used by majority of organizations.

For example, descriptive analytics examines historical electricity usage da to help plan power needs and allow electric companies to set optimal prices.

(iii) Prescriptive Analytics – Prescriptive analytics reveals actio and recommend of what step should be taken. It gives answer to the situatio in a focused way. Prescriptive data analytics goes one step forward predictive as it provides multiple actions with likely outcomes for each decisio This method of analytics is not preferred much by organizations, but its da can show impressive result if used correctly.

For example, prescriptive analytics can benefit healthcare strategic planning by using analytics to leverage operational and usage data combined with data of external factor such as economic data, population demographic trends and population health trends, to more accurately plan for future capital investments such as new facilities and equipment utilization as well as understand the trade-offs between adding additional beds and expanding an existing facility versus building a new one.

Q.17. Explain core components of analytical data architecture.
[R.G.P.V., May 2019 (VIII-Sem.)]

Ans. The big data storage and analytics platform provides resources and functionalities for storage as well as for batch and real-time processing of the big data. It provides main integration interfaces between the site operational platform and the cloud data lab platform and the programming interfaces for the implementation of the data mining processes. The internal structure of the big data storage and analytics platform is given in fig. 1.6.

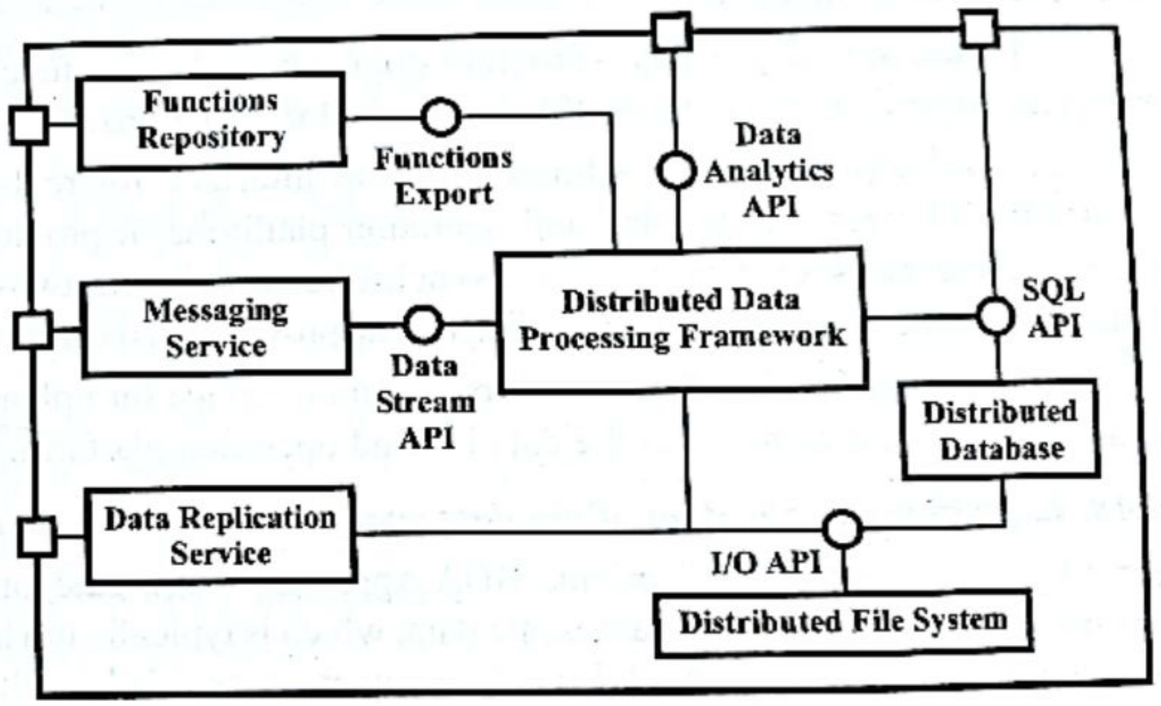

Fig. 1.6 The Internal Architecture of the Big Data Storage and Analytics Platform

Data are primarily stored in the distributed file system, which is responsible for the distribution and replication of large datasets across the multiple servers (data nodes). A unified access to the structured data is provided by the distributed database using the standard SQL interface. The main component responsible for data processing is the distributed data processing framework, which provides high-level API for the implementation of the data pre-processing tasks and for the building and validation of the predictive functions. Predictive functions are stored in the functions repository, where they are available for production deployment or for the simulations and overall optimization of the production

processes. The rest of the components (messaging service and data replicat service) provide data communication interfaces, and connect the operation platform to the data lab platform.

The big data storage and analytics platform consist of the following s components –

(i) ***Distributed File System*** – Provides a reliable, scalable file syst with similar interfaces and semantics to access data as local file systems.

(ii) ***Distributed Database*** – Provides a structured view of the d stored in the data lab platform using the standard SQL language, and suppo standard RDBMS programming interfaces such as JDBC for Java or ODB for .Net platforms.

(iii) ***Distributed Data Processing Framework*** – Allows the executi of applications in multiple nodes in order to retrieve, classify or transform t arriving data. The framework provides data analytics APIs for two ma paradigms for processing large datasets – API for parallel computation a API for distributed computation.

(iv) ***Functions Repository*** – Provides storage for predictive functio together with all settings required for the deployment of functions.

(v) ***Messaging Service*** – Implements an interface for real-ti communication between the data lab and operation platforms. It provides publish-subscribe messaging system for asynchronous real-time two-w communication, which allows to the decoupling of data providers and consume

(vi) ***Data Replication Service*** – Provides an interface for uploadi of the historical batch data between the data lab and operation platform.

Q.18. Explain the application of big data analytics in various fields.

Ans. Big data analytics applications (BDA Apps) are a new category software applications that leverage largescale data, which is typically too larg to fit in memory or even on one hard drive, to uncover actionable knowledg using large scale parallel-processing infrastructures. The big data can com from sources such as runtime information about traffic, tweets during th Olympic Games, stock market updates, usage information of an online game or the data from any other rapidly growing data intensive software system.

(i) ***In Clustering*** – Using clustering (K-means algorithm) throug a simple point and click dialog, users can automatically find groups within data based on specific data dimensions. With clustering, it is then simple t identify and address groups by customer type, text documents, products patient records, click path, behaviour, purchasing patterns, etc.

(ii) ***In Data Mining*** – Datameer's decision trees automatically help users understand what combination of data attributes result in a desired outcome

Decision trees illustrate the strengths of relationships and dependencies within data and are often used to determine what common attributes influence outcomes such as disease risk, fraud risk, purchases and online signups. The structure of the decision tree reflects the structure that is possibly hidden in big data.

(iii) ***In Banking*** – The use of customer data invariably raises privacy issues. By uncovering hidden connections between seemingly unrelated pieces of data, big data analytics could potentially reveal sensitive personal information. Research indicates that 62% of bankers are cautious in their use of big data due to privacy issues. Further, outsourcing of data analysis activities or distribution of customer data across departments for the generation of richer insights also amplifies security risks. For instance, a recent security breach at a leading UK-based bank exposed databases of thousands of customer files. Although this bank launched an urgent investigation, files containing highly sensitive information such as customers' earnings, savings, mortgages, and insurance policies ended up in the wrong hands. Such incidents reinforce concerns about data privacy and discourage customers from sharing personal information in exchange for customized offers.

(iv) ***In Marketing*** – Marketers have begun to use facial recognition software to learn how well their advertising succeeds or fails at stimulating interest in their products. A recent study published in the Harvard Business Review looked at what kinds of advertisements compelled viewers to continue watching and what turned viewers off. Among their tools was "a system that analyses facial expressions to reveal what viewers are feeling." The research was designed to discover what kinds of promotions induced watchers to share the ads with their social network, helping marketers create ads most likely to "go viral" and improve sales.

(v) ***In Smart Phones*** – Perhaps more impressive, people now carry facial recognition technology in their pockets. Users of iPhone and Android smart phones have applications at their fingerprints that use facial recognition technology for various tasks. For example, Android users with the remember app, can snap a photo of someone, then bring up stored information about that person based on their image when their own memory lets them down a potential boon for salespeople, iPhone users can unlock their device with recognize me, an app that uses facial recognition in lieu of a password. If deployed across a large enterprise, this app could save an average of $2.5 million a year in help-desk costs for handling forgotten passwords.

(vi) ***In Telecom*** – Now-a-days big data is used in different fields. In telecom also it plays a very good role. Service providers are trying to compete in the cut-throat world of telecom services. Where more and more subscribers rely on over-the-top (OTT) players as providers of value-added services are

focused on increasing revenue, reducing opex, chum and enhancing
customer experience as key business objectives.

Operators believe that big data and advanced analytics will play a crit
role in helping them meet their business objectives. In the same sur
respondents indicate critical use case scenarios in the context of big data
advanced analytics where they are investing now and where they plan
invest in the next three years.

Operators face an uphill challenge when they need to deliver
compelling, revenue generating services without overloading their netwo

(vii) In Agriculture – A biotechnology firm uses sensor data
optimize crop efficiency. It plants test crops and runs simulations to meas
how plants react to various changes in condition. Its data environment consta
adjusts to changes in the attributes of various data it collects, includ
temperature, water levels, soil composition, growth, output, and ge
sequencing of each plant in the test bed. These simulations allow it to disco
the optimal environmental conditions for specific gene types.

***Q.19. Explain advantages of using big data analytics in healthc
sector and banking sector.***

Ans. **Advantages of using Big Data Analytics in Healthcare Secto**
Advantages of using big data analytics in healthcare sector are as follows

(i) For Research – The large amount of data produced, gives gr
opportunity to researchers in fields of health informatics, by using tools t
techniques for unlocking the hidden patterns.

(ii) For Individual's/Patients – In deciding any line of treatment
a patient, historical data about the symptoms, drugs, outcomes, responses
different patients is taken into account. The move is towards formulating
(patient on personalized treatment) on the genomic data, locality, area and lifesty
response to certain medicines, allergy, and family history. When genome data
known completely, some kinds of relations are established between the DN
and the disease. Then specific treatment is formulated for such small groups
individual. The patient gets advantage by various ways such as correct as w
as effective line of treatment, better health related decisions, preventive steps
time, continuous health monitoring of patients by wireless devices, designi
personal line treatment, increase life quality and expectancy.

(iii) For Hospitals – By various techniques and tools of BDA data availab
in hospitals gain various advantages such as to predict patients which like to stay
longer time or get readmit after treatment, identification of patients that are risky
hospitalization, provider could develop pre health plans to prevent hospitalizatio
Various queries that could be answered using these BDA tools, include – will a patie

respond positively to the given treatment ? If surgery done, will the patient respond to it ? Will the patient get prone to catch disease in near future ?

The hospital management and administration can take better decisions like number of patients not getting cured at early stage, number of readmission increasing because of patient getting ill again after treatment, increasing number of staffs on floors for efficient working, plans for frequent post treatment follow ups, etc.

(iv) For Insurance Companies – Government for giving medical claim to patients do large amount of expenditure. By using BDA analysis, prediction and minimizing fraud medical claims can be done.

(v) For Pharmaceuticals – BDA techniques help R&D to produce drugs, instruments, tools etc. in shorter period of time, which are effective in treating specific diseases.

(vi) For Government – The demographic data, historical data of disease outbreak, weather data, and data from social media over diseases like cholera, flue etc. information is used by the government. Government analyzes this massive data to predict epidemics, by finding correlations between weather and disease and accordingly preventive measures are taken. Public health surveillance is improved as well as the response to disease outbreak is quick by using BDA.

(vii) For Pharma Companies – To improve workflow quality and quantity, like predictive modeling, statistical tool and algorithms. These improve the outcome of experiment and provide better understanding of developing drugs pharma companies need new tools. This tool successfully navigates the regulatory approval and marketing process.

Advantages of using Big Data Analytics in Banking Sector – Advantages of using big data analytics in banking sector are as follows –

(i) Sentiment Analytics – Continuously monitoring of customers opinion is needed from banks. Banks need to identify which are their key customers and by their feedback they need to improve their flaws in system. This lead to increase in their productivity and services.

(ii) Changes in Service Delivery – Whenever a reputation or account range enters into system, it checks through all the information and provide desired information. This allows banks to map work process, save time and prices. Huge information and its proper knowledge allow organization to identify and solve issues before they affect their customers.

(iii) Fraud Detection and Prevention – One of the most important obstacles faced by banking sectors is fraud. Big data ensures that no unauthorized transactions are done and provides security as well as safety to the entire system.

*(iv) **Enhanced Reporting*** – Getting access to huge amount of d
also contain different needs of different customers. Then banks offer th
needs in a meaningful way. Banking industry provides the exact informa
required by the customer by using big data.

*(v) **Risk Management*** – Early detection of fraud is a massive p
of risk management. Large amount of information does the maximum amo
of risk management as it will identify fraud. Massive information plays a m
role in desegregation of banks needs into a centralized practical platform
which possibilities of losing the information is reduced.

*(vi) **Customer Segmentation*** – By identifying usage of cards
customer, loyalty programs are created. Targeted marketing programs
made as well as relationships are build between valuable customers.

*(vii) **Examine Customer Feedback*** – Customers sentiments
collected in text form from various social media sites and after collecti
they are classified into positive and negative. This is used to provide servic
customer.

Q.20. Explain open-source technology for big data analytics.

Ans. Open-source software is computer software that is available in sou
code form under an open-source license that permits users to study, chan
and improve and at times also to distribute the software. The open-sou
name came out of a 1998 meeting in Palo Alto in reaction to Netscap
announcement of a source code release for Navigator (as Mozilla).

Although the source code is released, there are still governing bodies
agreements in place. The most prominent and popular example is the GNU Gen
Public License (GPL), which "allows free distribution under the condition t
further developments and applications are put under the same license." This ensu
that the products keep improving over time for the greater population of user

Some other open-source projects are managed and supported by commer
companies, such as Cloudera, that provide extra capabilities, training, a
professional services that support open-source projects such as Hadoop. This
similar to what Red Hat has done for the open-source project Linux.

One of the key attributes of the open-source analytics stack is that
not constrained by someone else's predetermined ideas or vision, says Da
Champagne, chief technology officer at Revolution Analytics, a provider
advanced analytics. The open-source stack does not put you into a straitjack
You can make it into what you want and what you need. If you come up w
an idea, you can put it to work immediately. That's the advantage of the op
source stack – flexibility, extensibility, and lower cost.

One of the great benefits of open-source lies in the flexibility of the adopt
model – you download and deploy it when you need it, said Yves de Montcher

vice president of marketing at Talend, a provider of open-source data integration solutions. You do not need to prove to a vendor that you have a million dollars in your budget. With open source, you can try it and adopt it at your own pace.

One disadvantage of open-source is that it has to coexist with the proprietary solution for a long time for many reasons. For example, getting data from Hadoop to a database required a Hadoop expert in the middle to do the data cleansing and the data type translation. If the data was not 100% (clean which is the case with most circumstances) a developer was needed to get it to a consistent, proper form. Besides wasting the valuable time of that expert, this process meant that business analysts could not directly access and analyze data in Hadoop clusters. SQL-H is software that is developed to solve this problem.

Q.21. What are the desired properties of big data system ?

Ans. The desired properties of big data system are as follows –

*(i) **Error Tolerance and Robustness*** – Because of the challenges encountered in distributed system, it is very much difficult to build a system that "do the right thing". Systems are required to behave in a right manner despite machines going down randomly, the complex semantics of uniformity in distributed databases, redundancy, concurrency, and many more. These challenges make it complicate even to reason about the functioning of the system. Robustness of big data system is needed to overcome the complexities associated with it.

*(ii) **Scalability*** – It is the ability to maintain the performance with the growing data and load by adding resources to the system. The lambda architecture is horizontally scalable across all layers of the system stack i.e. scaling is achieved by including more number of machines.

*(iii) **Generalization*** – A wide range of applications can be function in a general system. As lambda architecture is based on function of all data, it generalizes to all applications, whether financial management systems, social media analytics etc. because the lambda architecture is based functions of all data.

*(iv) **Debuggability*** – A big data system must provide the information required to debug the system when things go wrong. We should be able to trace, for each value in the system, exactly what caused it to have that value.

Debuggability is achieved in the lambda architecture through the functional nature of the batch layer and by preferring to use recomputation algorithms when possible.

*(v) **Ad hoc Queries*** – The ability to perform ad hoc queries on the data is significant. Every large dataset contains unanticipated value in it. Having

the ability of data mining arbitrarily provides opportunities for new applicati and business optimization.

(vi) Extensibility – Extensible system enables function to be add cost effectively. Sometimes, a new feature or a change to an already existi system feature needs to reallocate pre-existing data into a new data form Large scale transfer of data becomes easy in an extensible system.

(vii) Minimal Maintenance – Maintenance is the work required keep a system running smoothly. This includes anticipating when to a machines to scale, keeping processes up and running and debugging anythi that goes wrong in production. In order to have minimum maintenan components with low implementation complexity should be selected.

(viii) Low Latency Reads and Updates – Large number of applicatio need low latency reads, within a few milliseconds to a hundred millisecon While, the update latency requirements may vary widely. In some applicatio updates need to propagate immediately, but in other applications, update laten of few hours is allowed.

UNIT 2

HADOOP

INTRODUCTION TO HADOOP, CORE HADOOP COMPONENTS, HADOOP ECOSYSTEM, HIVE PHYSICAL ARCHITECTURE, HADOOP LIMITATIONS, RDBMS VERSUS HADOOP

Q.1. What is Hadoop ?

Ans. Hadoop was developed in the year of 2005 by Doug Cutting and Mike Caferella. It is the Apache open source software which allows to store nd process the huge volume of data in a distributed environment and it is ritten in java. Hadoop is also called MR1. The major social networking sites uch as Facebook, Yahoo, Google, Twitter and LinkedIn uses the Hadoop echnology to process their huge volume of data.

It is mainly designed to scale up from a single machine to thousands of achine; each offers the local computation and storage. Here, a single ame node manages the whole namespace in the hadoop cluster. The major ses of this technology are fast, scalable, inexpensive hardware and resilient failure. Hadoop consist of two main frame work Map reduce layer and DFS layer. Map reduce layer is used for processing the big data (where the ser application executes) and HDFS is used to store the big data (where the ser data resides).

Q.2. Explain main components of Hadoop.

Ans. Two main components of Hadoop are as follows –

(i) The Hadoop Distributed File System (HDFS) – HDFS is the torage system for a cluster. When data lands in the cluster, HDFS breaks it nto pieces and distribute those pieces among the different servers participating the cluster. Each server stores just a small fragment of the complete data set nd each piece of data is replicated on more than one server.

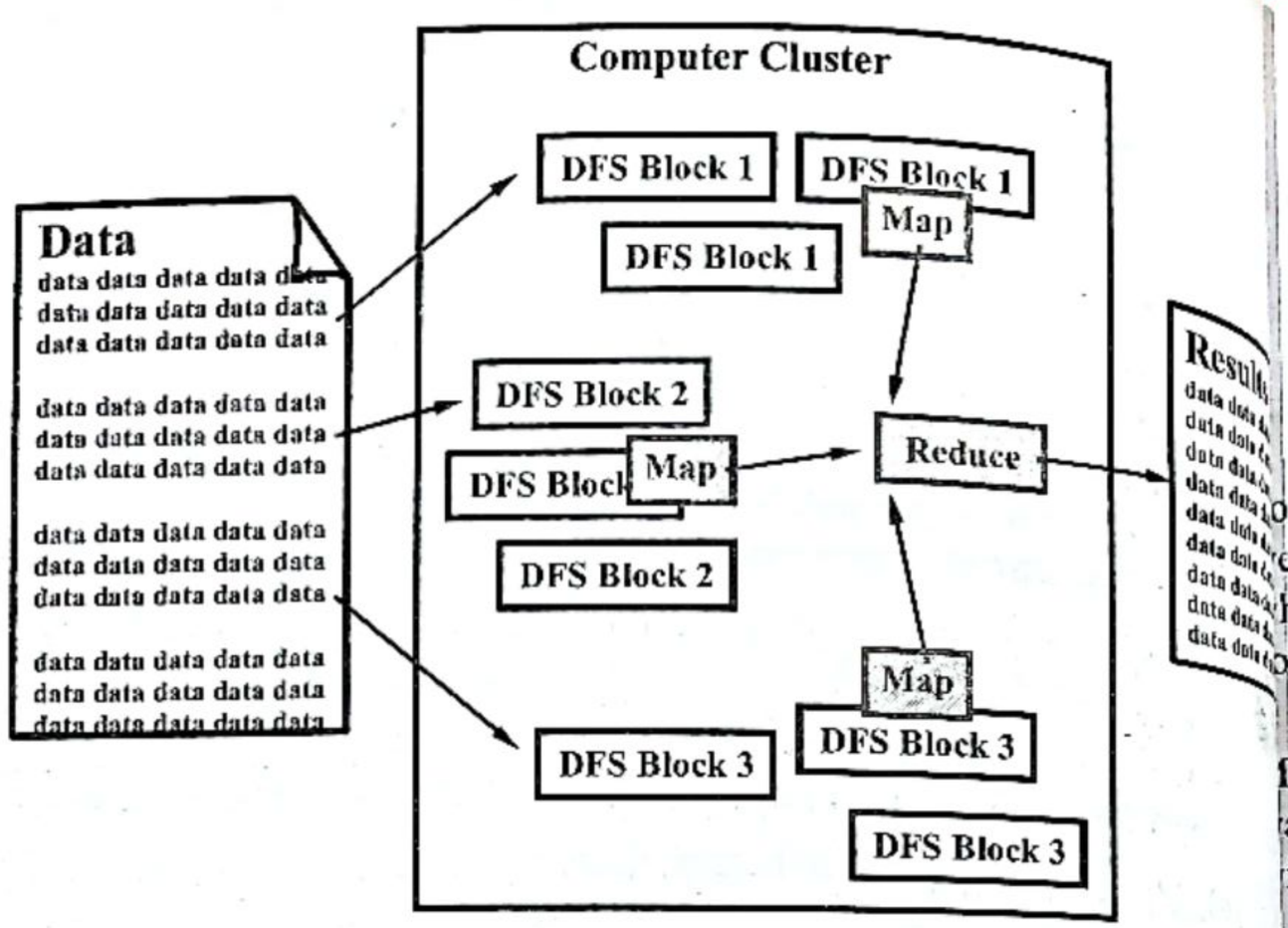

Fig. 2.1 HDFS & Map Reduce

(ii) Map Reduce – Because Hadoop stores the entire data
small pieces across a number of servers, analytical jobs can be distribu
parallel to each of the servers storing part of the data. Each server eva
the question against its local fragment simultaneously and reports its
back for collation into a comprehensive answer. Map Reduce is the
that distributes the work and collects the results. Both HDFS and Map R
are designed to continue to work even if there are failures. HDFS contin
monitors the data stored on the cluster. If a server becomes unavaila
disk drive fails or data is damaged due to hardware or software prob
HDFS automatically restores the data from one of the known good re
stored elsewhere on the cluster. MapReduce monitors the progress o
of the servers participating in the job, when an analysis job is running
of them is slow in returning an answer or fails before completing its
MapReduce automatically starts another instance of the task on a
server that has a copy of the data.

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

Q.3. Write short note on Hadoop's parallel world.

Ans. The Hadoop framework application works in an environme
provides distributed storage and computation across clusters of comp
Hadoop is designed to scale up from single server to thousands of mac
each offering local computation and storage.

The Hadoop's parallel world has following two major layers –

(i) Processing/computation layer is called MapReduce.

(ii) Storage layer is called Hadoop Distributed File System (HDFS).

Q.4. Explain the ecosystem of Hadoop.

Ans. Hadoop is an open source framework maintained by the Apache
undation for reliable, scalable and distributed computing. According to the
ebsite hadoop.apache.org, the components of Hadoop are defined as projects
hich function different to each other's. Some of the widely used Hadoop
mponents are as follows –

(i) Pig – It is a platform for HDFS. It consists of a compiler for
apReduce programs and a high-level language called Pig Latin. It provides a
ay to perform data extractions, transformations and loading, and basic analysis
ithout having to write MapReduce programs.

(ii) Hive – It is a distributed data warehouse. A data warehouse and
QL-like query language that presents data in the form of tables. Hive
rogramming is similar to database programming. (It was initially developed
y Facebook).

(iii) HBase – It is a non-relational, distributed database that runs on
p of Hadoop. HBase tables can serve as input and output for MapReduce jobs.

(iv) Zookeeper – It is an application that coordinates distributed
rocesses.

(v) Mahout – Mahout is a data mining software that can be easily
alable. Mahout offers java libraries or scalable machine learning algorithm
hich can be used for analyzing the data. These machine learning algorithms
low user to perform a task such as classification, clustering, association rule
nalysis, and predictive analysis.

(vi) Cassandra – Hadoop Cassandra provides database that can be
asily scalable and highly available without interruption in the job performance.

(vii) Chukwa – Chukwa is a data collections system which is mainly
ed for displaying, monitoring, and analyzing the outcomes of the collected data.

(viii) Spark – Spark is a computing system whicl. is used for
nfiguring the Hadoop cluster for fast processing of Hadoop data. Spark
oes not use MapReduce job of execution engine to run the job. It uses its
wn distributed runtime to complete the job.

(ix) Tez – Tez is a data-flow programming language build in the
adoop Yarn to execute an arbitrary DAG of tasks to process data for both
atch and interactive use-case.

(x) Avro – Avro is used for data serialization which provides a
ontainer file for storing persistent data. Avro was created by Doug Cutting

for making Hadoop to be writable in many programming languages such C++, C#, Java, JavaScript, Python, Ruby.

(xi) Ambari – It is a web interface for managing, configuring testing Hadoop services and components.

(xii) Flume – It is a software that collects, aggregates and large amounts of streaming data into HDFS.

(xiii) Sqoop – It is a connection and transfer mechanism that data between Hadoop and relational databases.

(xiv) Oozie – It is a Hadoop job scheduler.

The Hadoop ecosystem is shown in fig. 2.2.

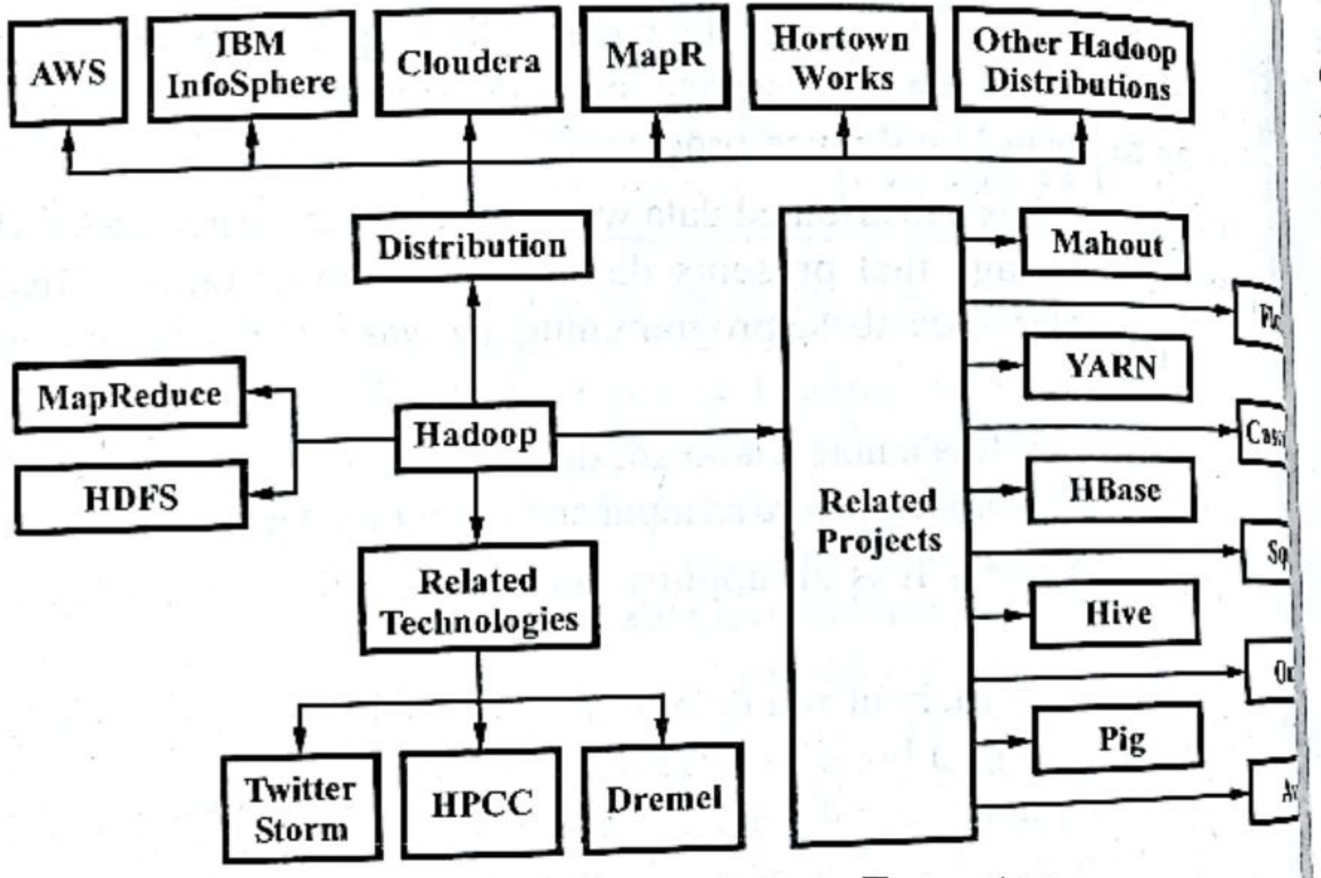

Fig. 2.2 The Hadoop Ecosystem

Q.5. What are the system requirements for installing Hadoop ?

Ans. There is various application which can be used for the analyzi data. In order to address the challenges of big data, Hadoop software used as an alternative solution. Hadoop is an open source software whic be downloaded from apache.org for free of cost. To get started using Ha software there are some recommended hardware and software details pro in the hadoop.apache.org website. According to the hadoop.apache.or hardware and software requirement for using single node cluster Hado listed below –

(i) Hardware Requirements –

(a) Operating System – Hadoop project can be perform the Linux or Windows operating system. Windows 10 version of ope system has been found to be most efficient.

(b) RAM Size – For using Hadoop the recommended RAM size is 4 GB or higher.

(c) Processor – Two or more core processors are needed for installing Hadoop.

(ii) Software Requirements –

(a) Java 7 or higher
(b) Hadoop 2.7 or higher
(c) Eclipse or IntelliJ community version
(d) Virtual machine and Cloudera (optional).

Hadoop is owned by Apache Foundation, and is available for free for downloading. Java, Eclipse, and IntelliJ are also available for downloading for free. Students and researchers can use these software for free. The software is also available from commercial vendors which provide support when needed. Cloudera and Horton works are the companies which provides Hadoop supports, but they charge for service. Their free version can be used but the software support is not available when needed.

Once the requirements are meet, the Hadoop software can be installed for free of cost to get started with the simple project. Later the software and hardware can be upgraded to work on more complex project with bigger volumes and variety of big data.

Q.6. What is Sqoop ? Also write its advantages.

Ans. Sqoop is mainly used to transfer the huge amount of data between Hadoop and relational database. Sqoop refers "SQL to Hadoop and Hadoop to SQL". It imports the data from the relational database such as Mysql, Oracle, postgreSQL to the Hadoop (HDFS, Hive, HBase) and exports the data from HDFS to relational database. The non-Hadoop data store can also be extracted and transformed to Hadoop data store. The Extraction, Transformation and Loading (ETL) can be performed by using Sqoop. It is the open source framework of cloudera Inc. The data can be imported and exported in a parallel manner.

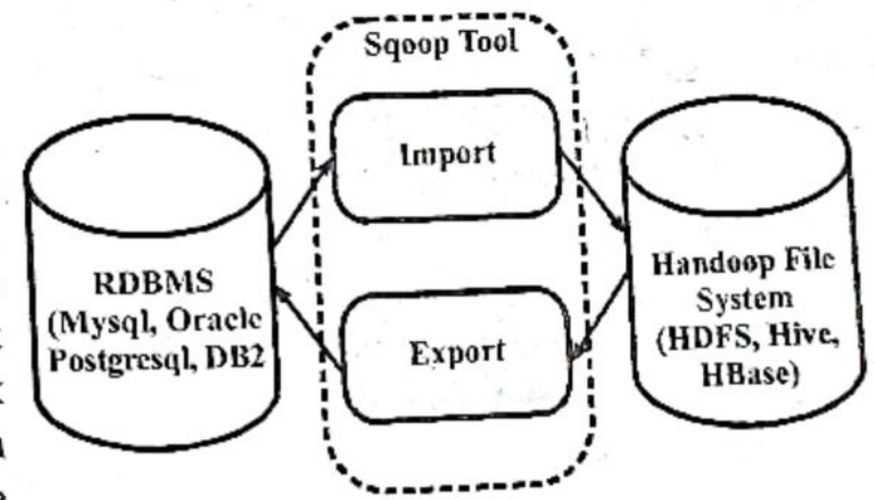

Fig. 2.3 SQOOP Transformation

Advantages of Sqoop are as follows –

(i) It offers the migration of heterogeneous data.
(ii) It offers easy integration with Hive, HBase and oozie.
(iii) We can import the whole database or the single table in to HDFS.

Q.7. What is zookeeper ? Also write its advantages and disadvantage

Ans. In a traditional distributed environment coordinating and managing task is quite complex and complicated. But the zookeeper overcomes this proble with the help of simple architecture and its API. In the cluster (group of node to maintain the shared data and coordinating among themselves they u zookeeper as a service to provide robust synchronization. It provides servic such as naming service (identify the name of the node in the cluster), configurati management (Up to date information is maintained), cluster management (t status of the node leaving or joining in the cluster), leader election (for coordinati among themselves they elect a single node as leader in the cluster), Locking a synchronizing service (when any data can be modified in the cluster it locks th particular data to provide consistency) etc. The inconsistency of data, ra condition and deadlock problem in the traditional distributed environment can b solved easily with the help of zookeeper mechanism such as Atomicity, serializati property and synchronization respectively.

Advantages of Zookeeper –

(i) It provides reliability and availability of data.

(ii) It offers high synchronization and serialization.

(iii) The atomicity eliminates the inconsistency of data among cluste

(iv) It is fast and simple.

Disadvantages of Zookeeper –

(i) The large number of stacks needs to be maintained.

Q.8. What is Mahout ? Give its advantages and disadvantages.

Ans. Mahout is the Apache open source software framework and provides data mining library. The processing task can be split in to multip segments and each segment can be computed on a different machine in ord to speed up the computation process. The primary goals of the Mahout a data clustering, classification, regression testing, statistical modeling an collaborative filtering. It provides scalable data mining and machine learnin (it makes the decision based on the current and previous history of dat approaches for the data.

Advantages of Mahout –

(i) It supports complementary and distributed naive bay classification.

(ii) It mines the huge volume of data.

(iii) The companies such as Adobe, Twitter, Foursquare, Faceboo and LinkedIn internally uses Mahout for data mining.

(iv) Yahoo uses it for pattern mining.

Disadvantages of Mahout –

(i) It doesn't support scala version in the development.

(ii) It has no decision tree algorithm.

Q.9. What is Oozie ? Also write its advantages and disadvantages.

Ans. Oozie was initially developed at Yahoo for their complex workflow search engine. Later it was acquired by open source Apache incubator. It is a workflow scheduler for managing Hadoop jobs. There are two major types of oozie jobs are available, i.e. oozie workflow and oozie coordination. In the oozie workflow it follows Directed Acyclic Graph (DAG) for parallel and sequential execution of jobs in the Hadoop. It contains the control flow node. The control flow node controls the beginning and end of the wo.kflow execution. In the oozie coordination, workflow jobs are triggered by time.

Advantages of Oozie –

(i) It allows the workflow of execution can be restarted from the failure.

(ii) It provide web service API (i.e. we can control the jobs from anywhere).

Disadvantages of Oozie –

(i) It is not a resource scheduler.

(ii) It is not suitable for off grid scheduling.

Q.10. Give some applications of Hadoop.

Ans. Now-a-days, with the rapid growth of the data volume, the storage and processing of Big Data has become the most pressing needs of the enterprises. Hadoop as the open source distributed computing platform has become a brilliant choice for the business. The users can develop their own distributed applications on Hadoop and processing Big Data even if they do not know the bottom-level details of the system. Due to the high performance of Hadoop, it has been widely used in many companies. Some applications of Hadoop are given below –

(i) Hadoop in Yahoo! – Yahoo! is the leader in Hadoop technology research and applications. It applies Hadoop on various products, which include the data analysis, content optimization, anti-spam e-mail system, and advertising optimization. Hadoop has also been fully used in user interests' prediction, searching ranking, and advertising location.

In the Yahoo! home page personalization, the real-time service system will read the data from the database to the interest mapping through the Apache. Every 5 minutes, the system will rearrange the contents based on Hadoop cluster and update the contents every 7 minutes.

For spam e-mails, Yahoo! uses the Hadoop cluster to score the e-ma
Every couple of hours, the Yahoo! will improve the anti-spam e-mail mode
the Hadoop clusters and the clusters will push 5 billion times of e-mails deli
every day. At present, the largest application of the Hadoop is the Sea
Webmap of Yahoo!. It has been run on more than 10000 Linux cluster machi

(ii) Hadoop in Facebook – Facebook is the largest social netw
in the world. From 2004 to 2009, Facebook has over 800 million active us
The data created everyday is huge. This means that Facebook is facing
problem with big data processing which contains content maintainance, pho
sharing, comments, and users access histories. These data are not easy
process so Facebook has adopted the Hadoop and Hbase to handle it.

Q.11. Why Facebook has chosen Hadoop ?

Ans. As Facebook is developing, it discovered that MySQL cannot meet
its requirements. After long-term research and experiments, Facebook fina
choose Hadoop and Hbase as the data processing system. The reason w
Facebook choose the Hadoop and Hbase has the two aspects. On the one ha
Hbase meets the requirements of Facebook. Hbase can support the rapid acce
to the data. Although Hbase does not support the traditional outer form operatio
the Hbase column oriented storage model brings high flexibility search in t
inner form. Hbase is also a good choice for intensive data. It is able to maint
huge data, support the complex index with the flexible scalability and guarant
the speed of data access. On the other hand, Facebook has the confidence
solve the Hadoop problems in real use. For now, Hbase has already been able
provide high consistency and high throughput key-value storage but t
Namenode as the only manager node in the HDFS may become the bottlene
of the system. Then, Facebook has designed a high availability Namenode, cal
AvatarNode to solve this problem. In the aspect of the fault tolerance, HDF
can tolerate and isolate faults in the subsystem of the disk. The failures of t
whole clusters of Hbase and HDFS are part of fault tolerance system.

Overall, according to the improvements by the Facebook, Hadoop c
meet the Facebook most requirements and can provide a stable, efficient, a
safe service for the Facebook users.

Q.12. What are the advantages of Hadoop ? Explain Hadoop architecture and its components with proper diagram. [R.G.P.V., May 2019 (VII-Sem

Ans. **Advantages of Hadoop –**

(i) The scalability and elasticity of free open source Hadoop runnin
on standard hardware allow organizations to hold onto more data and tak
advantage of all their data to increase operational efficiency and gain competiti
edge. Hadoop supports complex analysis across large collections of data
one tenth the cost of traditional solutions.

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

(iii) Apache Hadoop is an open-source project by the Apache Software foundations. The software was originally developed by the world's largest Internet companies to capture and analyze the data that they generate. Unlike traditional, structured platforms Hadoop is able to store any kind of data in its native format and to perform a wide variety of analyses and transformation on that data. Hadoop stores terabytes and even petabytes of data inexpensively. It is robust and reliable and handles hardware and system failures automatically without losing data analyses.

(iv) Hadoop runs on clusters of commodity servers and each of those servers has local CPUs and disk storage that can be leveraged by the system.

Hadoop Architecture – Hadoop is an open-source framework that allows users to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines with high degree of fault tolerance. Data in a Hadoop cluster is broken down into smaller pieces and distributed throughout the cluster like the Map and Reduce functions that are executed on smaller subsets of larger data sets, and this provides the scalability needed for big data processing.

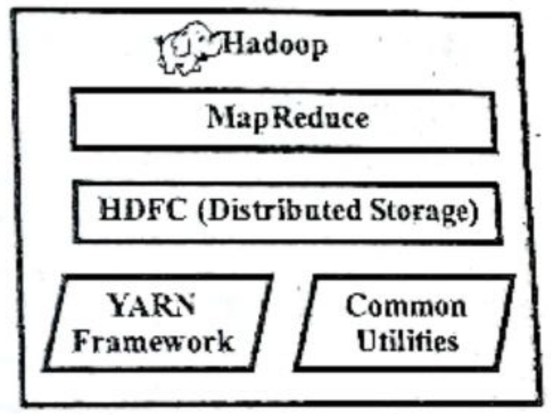

Fig. 2.4 Hadoop Architecture

Hadoop framework includes four models –

(i) Hadoop Common – They contain Java libraries and utilities that are required by other Hadoop modules. The Java libraries provide file system and OS level abstraction. It contains necessary Java files and scripts that are required to start Hadoop.

(ii) Hadoop Yarn – YARN is a cluster management technology. It is one of the key features in second-generation of Hadoop, designed from the experience gained from the first generation of Hadoop. YARN provides resource management and a central platform to deliver consistent operations, security and data governance tools across Hadoop clusters.

(iii) HDFS (Hadoop Distributed File System) – It is a distributed file system that provides high throughput computing access to application data.

(iv) Hadoop MapReduce – For large scale data processing this is programming model.

Components of Hadoop – Refer to Q.4.

Q.13. Explain the Hive physical architecture.

Ans. Fig. 2.5 shows the major components of Hive and its interaction with Hadoop. The main components of Hive are –

External Interfaces – Hive provides both user interfaces like command line (CLI) and web UI, and application programming interfaces (API) like JDBC and ODBC.

The Hive Thrift Server exposes a very simple client API to execute HiveQL statements. Thrift is a framework for cross-language services, where a server written in one language (like Java) can also support clients in other languages. The Thrift Hive clients generated in different languages are used to build common drivers like JDBC (java), ODBC (C++), and scripting drivers written in php, perl etc.

The Driver manages the life cycle of a HiveQL statement during compilation, optimization and execution. On receiving the HiveQL statement, from the thrift server or other interfaces, it creates a session handle which is later used to keep track of statistics like execution time, number of output rows, etc.

Hive > load data localInPath '/home/hadoop/file.txt' into table students

Select Command:

Hive>select*from students;

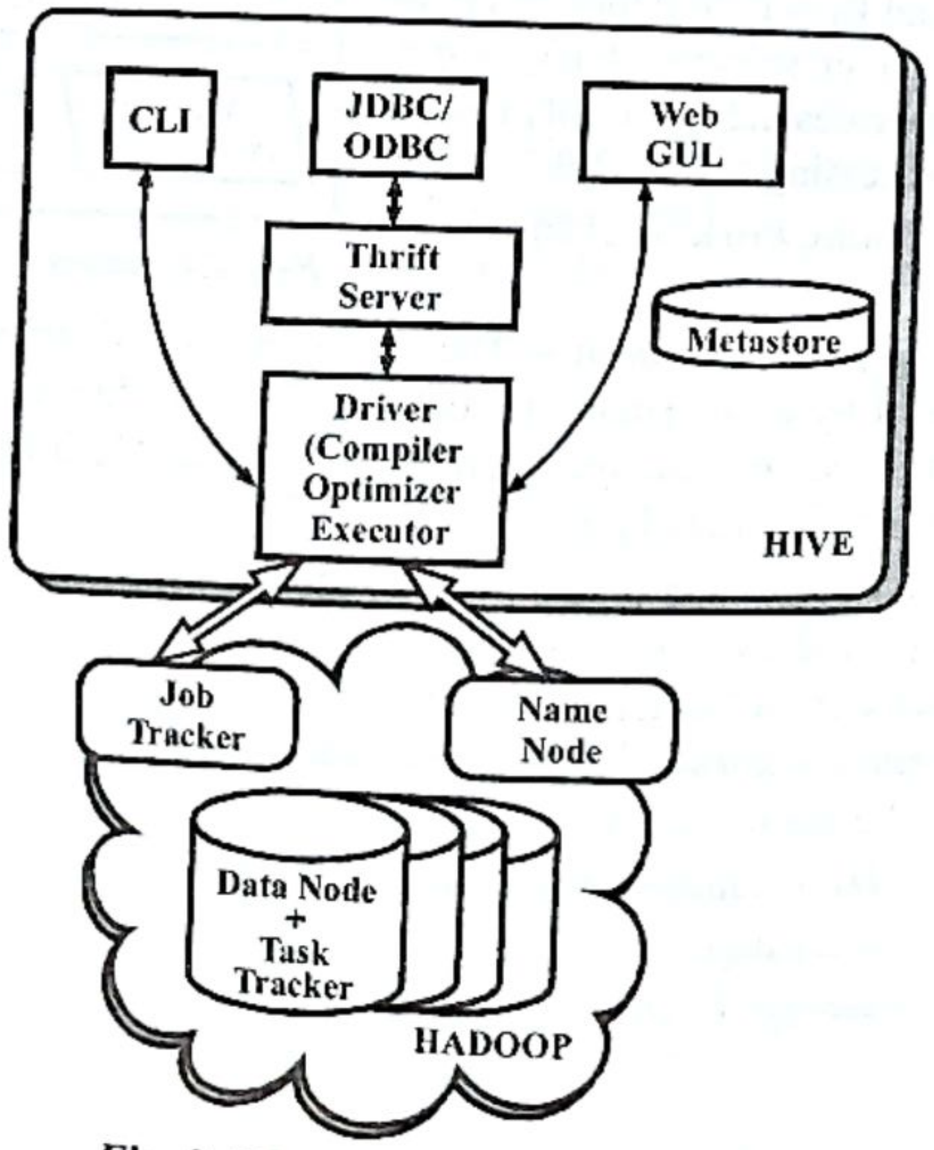

Fig. 2.5 Hive Physical Architecture

There are 2 types of tables in Hive viz, managed table and external table.

(i) Managed Table – Managed table is like a normal database table where data can be stored and queried on. On dropping these tables, data also lost forever.

Creating an Internal Table –

```
CREATE TABLE STUDENTS (roll_number INT,
name STRING, age INT, address STRING)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ',';
```

(ii) External Table –

```
CREATE EXTERNAL TABLE STUDENTS (roll number
INT, name STRING, age INT, address STRING)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ','
LOCATION'
```

ROW FORMAT should have delimiters used to terminate the fields and lines like in the above example the fields are terminated with comma (",").

Q.14. Give the limitations of Hadoop.

Ans. The limitations of Hadoop are as follows –

(i) Security Concerns – Hadoop is missing encryption at the storage and network levels, which is a major limitation from government agencies and others organizations point of view that prefer to keep their data under wraps.

(ii) Vulnerable by Nature – Speaking of security, the very makeup of Hadoop makes running it a risky proposition. The framework is written almost entirely in Java. It has been heavily exploited by cyber criminals and as a result, implicated in numerous security breaches. For this reason, several experts have suggested dumping it in favor of safer, more efficient alternatives.

(iii) Not Fit for Small Data – Due to its high capacity design, the Hadoop distributed file system lacks the ability to efficiently support the random reading of small files. As a result, it is not recommended for organizations with small quantities of data.

Q.15. Differentiate Hadoop vs distributed data base.

[R.G.P.V., May 2019 (VIII-Sem.)]

Ans. Differences between Hadoop and distributed data base are as follows –

Parameter	***RDBMS***	***Hadoop***
Type of data	Structured data with known schemes	Unstructured and structured
Data groups	Records, long fields, objects, XML	Files

Data modification	Updates allowed	Only inserts and deletes (hdfs)
Programs	SQL & XQuery	Hive, Pig, Jaql
Access	Quick response, random access	Random access (indexing)
Data loss	Data loss is not acceptable	Data loss can happen sometimes
Security and auditing	Yes	Not yet
Encryption	Yes	Not yet
Compression method	Sophisticated data compression	Simple file compression
Hardware requirement	Enterprise hardware	Commodity hardware
Evolution (for 2016)	30+ years of innovation	5-8 years old technology
Data processing	Batch processing	Streaming access to full files
Acceptance	Large DBA and application development community, widely used.	Small number of companies using it in production, many startups.

HADOOP DISTRIBUTED FILE SYSTEM, PROCESSING DATA WITH HADOOP, MANAGING RESOURCES AND APPLICATION WITH HADOOP YARN, MAPREDUCE PROGRAMMING

Q.16. Discuss in detail about Hadoop Distributed File System (HDFS).
[R.G.P.V., May 2019 (VII-Sem.)]

Ans. HDFS also known as Hadoop Distributed File System is one of the Hadoop components which handles the storage of big data. When users need to add more storage in the system, then they can easily increase the storage capacity by adding servers. HDFS consist of number of clusters depending upon the user configurations. The cluster consists of Master and Slave nodes. The data in the Hadoop cluster are broken into many small blocks which are 128 MB sizes by default. These blocks are

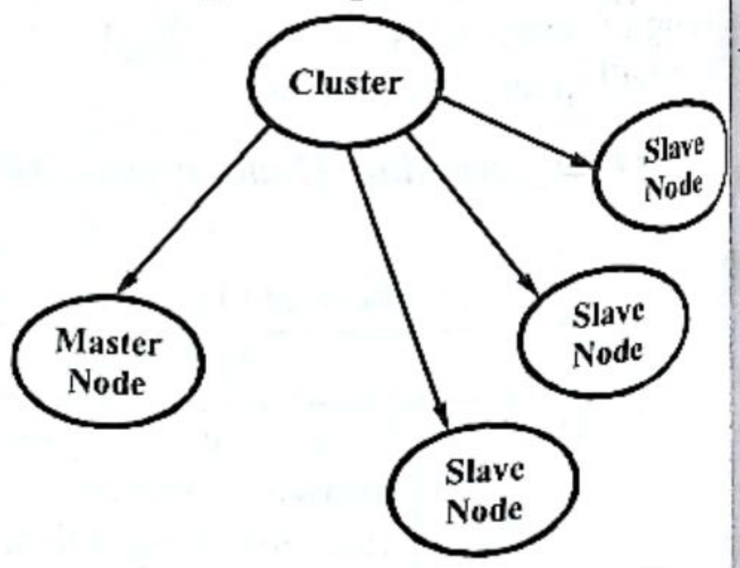

Fig. 2.6 Hadoop Cluster Node

stored in the different slaves' nodes in the Hadoop clusters. These blocks are highly scalable and can be increased when needed.

In HDFS, users can create new file, append content to the end of file, delete or rename the file, and modify file attributes. In comparison to traditional method of handling data, Hadoop's storage can be scalable at a very low cost because Hadoop uses commodity hardware.

Hadoop is composed of clusters. And cluster have Master node and Slave node as shown in the fig. 2.6. Master node is also known as name node which assigns jobs to the slave nodes. Beside assigning jobs to the slave nodes, master node manages the file system namespace. All the details are store in the form of namespace image and edit log. Cluster have only one master node, where as it may have multiple slave nodes. The function of slave nodes is to store data in the form of blocks and performed a job assigned by the master node.

HDFS is a distributed system which is suitable for running on the commodity hardware. There are many common characteristics in the existing distributed systems but the differences between them are also obvious. HDFS is a high fault-tolerant system and relaxed the parts of the POSIX constraints to provide high throughput access to the data so that it can be suitable to applying on the big data.

Q.17. What are the features of HDFS ? List out the characteristics of HDFS.

Ans. **The Features of HDFS –** HDFS is not a general-purpose file system, as it only executes specific types of applications, it does not need all the requirements of a general distributed file system. For example, security has never been supported for HDFS systems.

Characteristics – The characteristics of HDFS are as follows –

(i) HDFS fault tolerance
(ii) Block replication
(iii) Replica placement
(iv) Heartbeat and block report messages
(v) HDFS high throughput access to large dataset.

Q.18. Why is a block in HDFS so large ? List out the areas where HDFS cannot be used.

Ans. HDFS blocks are large compared to disk blocks, in order to minimize the cost of seeks. By making a block large enough, the time to transfer the data from the disk can be made significantly larger than the time to seek to the start of the block. Thus the time to transfer a large file made of multiple blocks operates at the disk transfer rate.

HDFS cannot be used in following areas –

(i) Low-latency data access
(ii) Lots of small files
(iii) Multiple writers, arbitrary file modifications.

Q.19. Explain the architecture of Hadoop distributed file (HDFS).

Ans. HDFS is the master/slave structure. The Namenode is the node, while the Datanode is the slave node. Documents are stored blocks in the Datanode. The default size of a data block is 64M and be changed. If the files are less than a block data size, HDFS will not the whole block storage space. The Namenode and the Datanode nor as Java programs in the Linux operating system.

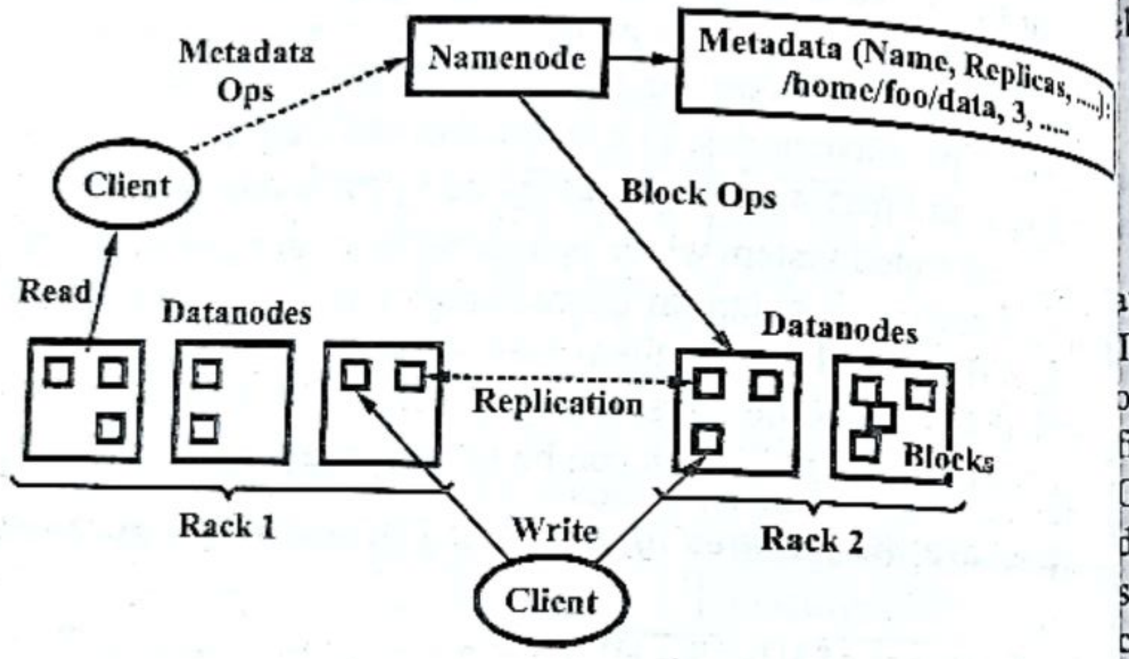

Fig. 2.7 HDFS Architecture

The Namenode which is the manager of the HDFS is responsible f management of the namespace in the file system. It will put all the folde files metadata into a file system tree which maintains all the metadata files directories. At the same time, Namenode also saves the corresp relations between each file and the location of the data block. Datanode place to store the real data in the system. However, all the data is not stor the hard drives but will be collected when the system starts to find the rea data server of the required documents.

The Secondary Namenode is a backup node for the Namenode. If is only one Namenode in the Hadoop cluster environment, the Namenod obviously become the weakest point of the process in the HDFS. Once failure of the Namenode occurs, it will affect the whole operation of the sy This is the reason why Hadoop designed the Secondary Namenode a alternative backup. The Secondary Namenode usually runs on a separate phy computer and keeps communication at certain time interval to keep the snap of the file system metadata with the Namenode so that it can recover the immediately in case some error happens.

The Datanode is the place where the real data is saved and handles of the fault-tolerant mechanism. The files in HDFS are usually divided

ultiple data blocks stored in the form of redundancy backup in the Datanode. he Datanode reports the data storage lists to the Namenode regularly so that he user can obtain the data by direct access to the Datanode.

The client is the HDFS user. It can read and write the data though calling he API provided by HDFS. While in the read and write process, the client first eeds to obtain the metadata information from the Namenode, and then the lient can perform the corresponding read and write operations.

Q.20. Write short note on the followings –
(i) Authority management of HDFS.
(ii) Limitations of HDFS.

Ans. (i) Authority Management of HDFS – HDFS shares a similar uthority system to POSIX. Each file or directory has an owner and a group. The authority permissions for the files or the directories are different to the wner, users in the same group, and other users. On the one hand, for the iles, users are required the –r authority to read and the –w authority to write. On the other hand, for the directories, users need the –r authority to list the irectory content and –w authority to create or delete. Unlike the POSIX system, there is no sticky, setuid or setgid of directories because there is no concept of executable files in HDFS.

(ii) Limitations of HDFS – HDFS as the open source implementation of GFS (Google File System) is an excellent distributed file system and has many advantages. HDFS was designed to run on the cheap commodity hardware not on expensive machines. This means that the probabilities of node failure are slightly high. To give a full consideration to the design of HDFS, we may find that HDFS has not only advantages but also limits for dealing with some specific problems. The limitations of HDFS are as follows –

(a) High Access Latency – HDFS does not fit for the requests which should be applied in a short time. The HDFS was designed for the Big Data storage and it is mainly used for it high throughput abilities. This may cost the high latency instead. Because HDFS has only one single Master system, all the file requests need to be processed by the Master. When there is a huge number of requests, there is an inevitable delay. Currently, there are some additional projects to address this limitation, such as Hbase uses the Upper Data Management project to manage the data.

(b) Poor Small Files Performance – HDFS needs to use the Namenode to manage the metadata of the file system to respond to the client and return the locations so that the limitation of a file size is determined by the Namenode. In general, each file, folder, and block need to occupy the 150 bytes' space. In other words, if there are one million files and each file occupies one block, it will take 300 MB space. Based on the current technology, it is

possible to manage millions of files. However, when the files extend to billions, the work pressures on the Namenode is heavier and the time of retrieving data is unacceptable.

(c) Unsupported Multiple Users Write Permissions – In HDFS, one file just has one writer because multiple users' writer permissions are not supported yet. The write operations can only be added at the end of the file not at the any positions of the file by using the Append method.

Q.21. Describe in detail about dataflow of file read in HDFS.

Ans. To get an idea of how data flows between the client interacting with HDFS, the Namenode and the Datanode, consider the fig. 2.8, which shows the main sequence of events when reading a file.

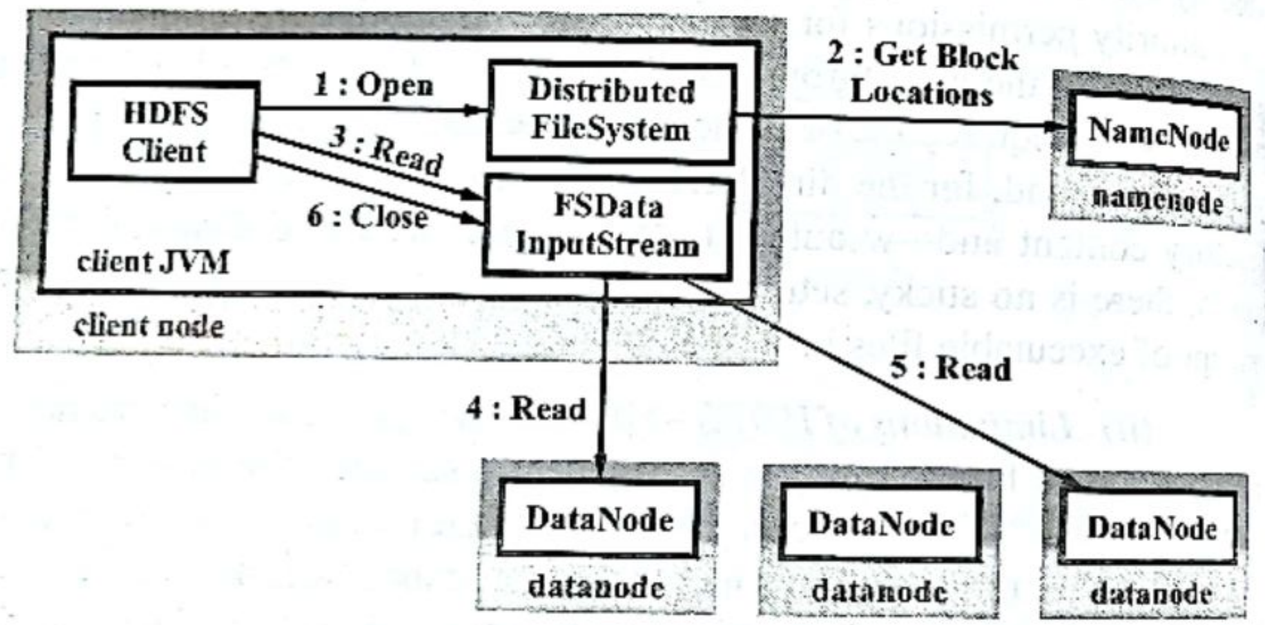

Fig. 2.8 Client Reading Data from HDFS

The client opens the file it wishes to read by calling open( ) on the FileSystem object, which for HDFS is an instance of DistributedFileSystem (step 1). DistributedFileSystem calls the Namenode, using RPC, to determine the locations of the blocks for the first few blocks in the file (step 2). For each block, the Namenode returns the addresses of the Datanodes that have a copy of that block. Furthermore, the Datanodes are sorted according to their proximity to the client. If the client is itself a Datanode (in the case of a MapReduce task, for instance), then it will read from the local Datanode.

The DistributedFileSystem returns a FSDataInputStream to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the Datanode and Namenode I/O. The client then calls read( ) on the stream (step 3). DFSInputStream, which has stored the Datanode addresses for the first few blocks in the file, then connects to the first (closest) Datanode for the first block in the file. Data is streamed from the Datanode

back to the client, which calls read( ) repeatedly on the stream (step 4). When the end of the block is reached, DFSInputStream will close the connection to the Datanode, then find the best Datanode for the next block (step 5). This happens transparently to the client, which from its point of view is just reading a continuous stream. Blocks are read in order with the DFSInputStream opening new connections to Datanodes as the client reads through the stream. It will also call the Namenode to retrieve the Datanode locations for the next batch of blocks as needed. When the client has finished reading, it calls close( ) on the FSDataInputStream (step 6).

One important aspect of this design is that the client contacts Datanodes directly to retrieve data, and is guided by the Namenode to the best Datanode for each block. This design allows HDFS to scale to large number of concurrent clients, since the data traffic is spread across all the Datanodes in the cluster. The Namenode meanwhile merely has to service block location requests (which it stores in memory, making them very efficient), and does not, for example, serve data, which would quickly become a bottleneck as the number of clients grew.

Q.22. Describe in detail about dataflow of file write in HDFS.

Ans. The case we are going to consider is the case of creating a new file, writing data to it, then closing the file.

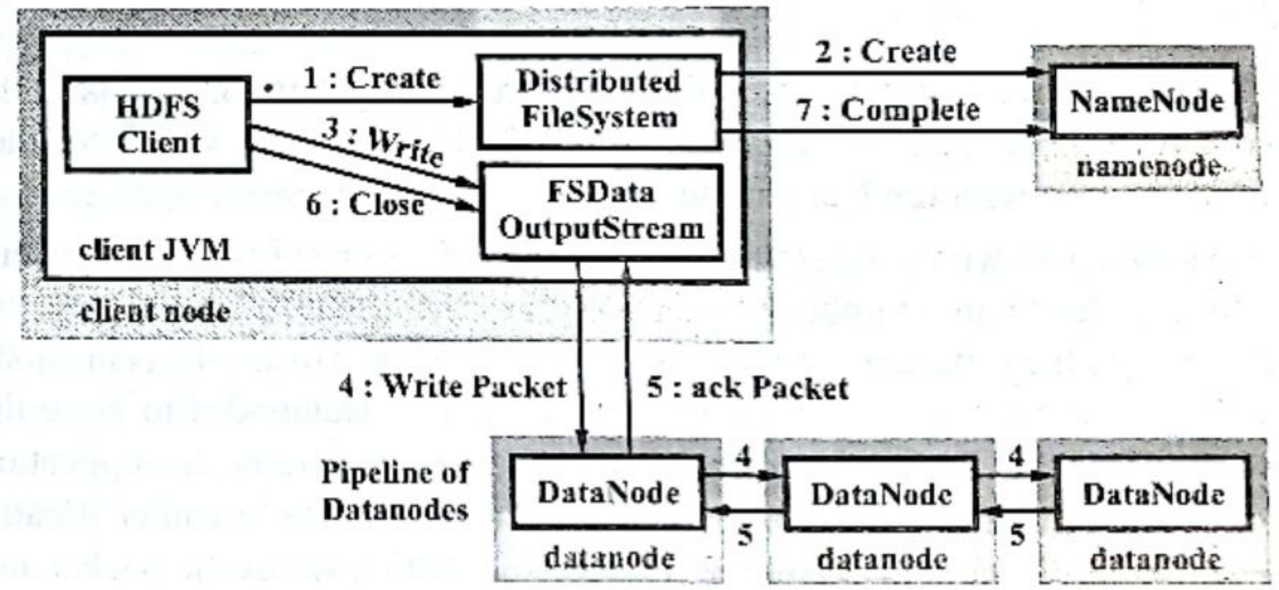

Fig. 2.9 Client Writing Data to HDFS

The client creates the file by calling create( ) on DistributedFileSystem (step 1). DistributedFileSystem makes an RPC call to the Namenode to create a new file in the filesystem's namespace, with no blocks associated with it (step 2). The Namenode performs various checks to make sure the file does not already exist, and that the client has the right permissions to create the file. If these checks pass, the Namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException. The

DistributedFileSystem returns a InputFormat is also responsible for creating the input splits and dividing them into records. The data is divided into number of splits (typically 64/128Mb) in HDFS. An input split is a chunk of the input that is processed by a single map.

InputFormat class calls the getSplits( ) function and computes splits for each file and then sends them to the jobtracker, which uses their storage locations to schedule map taks to process them on the tasktrackers. On a tasktracker, the map task passes the split to the createRecordReader( ) method on InputFormat to obtain a RecordReader for that split. The RecordReader loads data from its source and converts into key-value pairs suitable for reading by mapper. The default InputFormat is TextInputFormat which treats each value of input a new value and the associated key is byte offset.

A RecordReader is little more than an iterator over records, and the map task uses one to generate record key-value pairs, which it passes to the map function. We can see this by looking at the Mapper's run( ) method –

```
public void run(Context context) throws IOException, InterruptedException{
setup(context);
while(context.nextKeyValue( )) {
map(context.getCurrentKey( ), context.getCurrentValue( ), context);
}
cleanup(context);
}
```

SDataOutputStream for the client to start writing data to. Just as in the read case, FSDataOutputStream wraps a DFSOutputStream, which handles communication with the Namenode.

As the client writes data (step 3), DFSOutputStream splits it into packets, which it writes to an internal queue, called the data queue. The data queue is consumed by the DataStreamer, whose responsibility it is to ask the Namenode to allocate new blocks by picking a list of suitable Datanodes to store the replicas. The list of Datanodes forms a pipeline – we will assume the replication level is 3, so there are three nodes in the pipeline. The Data Streamer streams the packets to the first Datanode in the pipeline, which stores the packet and forwards it to the second data node in the pipeline. Similarly, the second data node stores the packet and forwards it to the third (and last) Datanode in the pipeline (step 4). DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by Datanodes, called the ack queue. A packet is removed from the ack queue only when it has been acknowledged by all the Datanodes in the pipeline (step 5).

If a Datanode fails while data is being written to it, then the following actions are taken, which are transparent to the client writing the data. First the

pipeline is closed, and any packets in the ack queue are added to the front of the data queue so that Datanodes that are downstream from the failed node will not miss any packets. The current block on the good Datanodes is given a new identity, which is communicated to the Namenode, so that the partial block on the failed Datanode will be deleted if the failed Datanode recovers later on. The failed Datanode is removed from the pipeline and the remainder of the block's data is written to the two good Datanodes in the pipeline. The Namenode notices that the block is under-replicated, and it arranges for a further replica to be created on another node. Subsequent blocks are then treated as normal.

When the client has finished writing data it calls close( ) on the stream (step 6). This action flushes all the remaining packets to the Datanode pipeline and waits for acknowledgements before contacting the Namenode to signal that the file is complete (step 7). The Namenode already knows which blocks the file is made up of (via Data Streamer asking for block allocations), so it only has to wait for blocks to be minimally replicated before returning successfully.

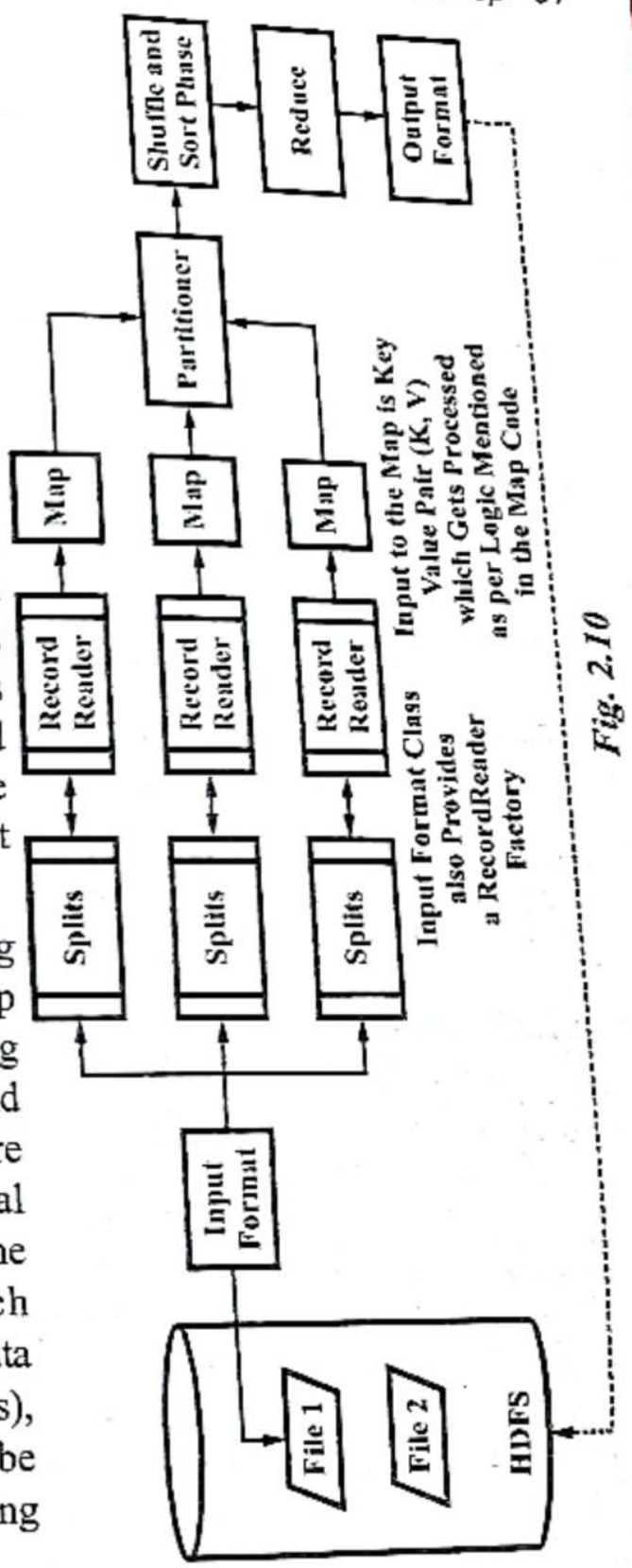

Fig. 2.10

Q.23. What is the Google file system ? Explain architecture of GFS.

Ans. The Google File System (GFS) is a scalable distributed file system for large distributed data intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients. GFS provides a familiar file system interface, though it does not implement a standard API such as POSIX. Files are organized hierarchically in directories and identified by path-names. GFS support the usual operations such as create, delete, open, close, read, and write files.

GFS has snapshot and record append operations. Snapshot creates a copy of a file or a directory tree at low cost. Record append allows multiple clients to append data to the same file concurrently while guaranteeing the atomicity of each individual clients append.

Architecture – A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients, as shown in the fig. 2.11.

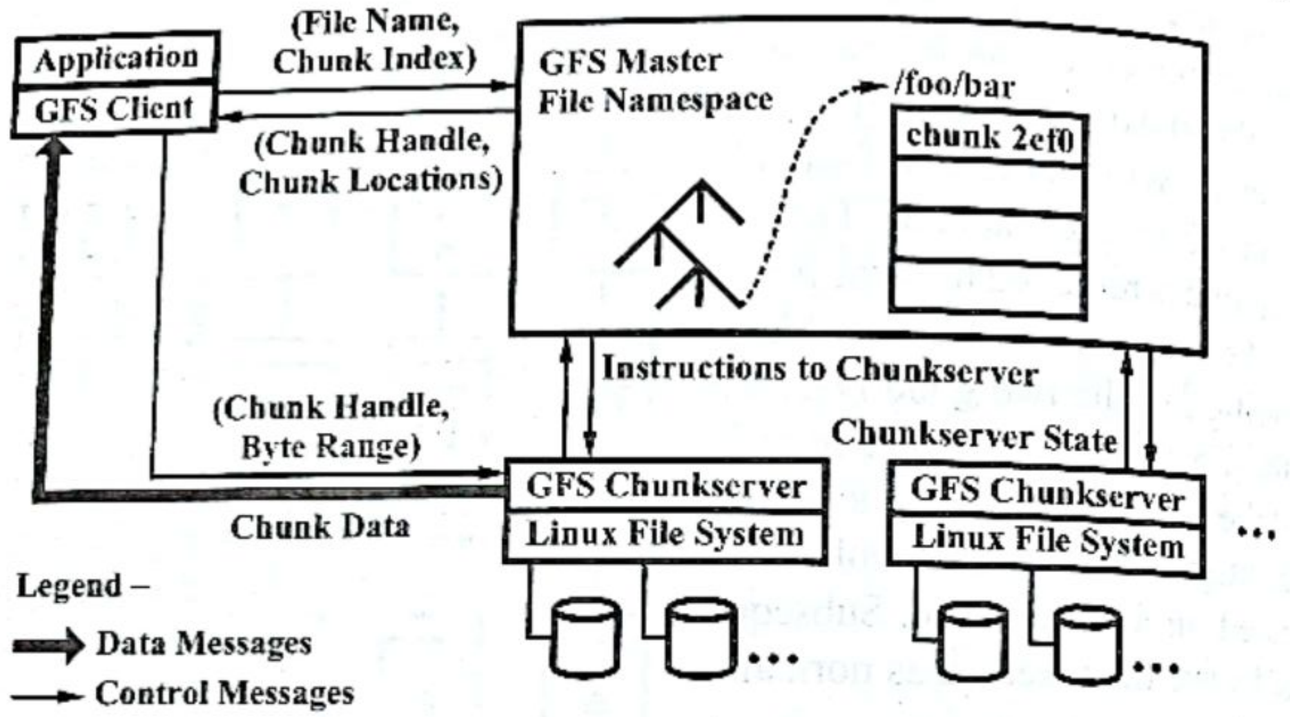

Fig. 2.11 GFS Architecture

Each of these is typically a commodity Linux machine running a user-level server process. Files are divided into fixed-size chunks. Each chunk is identified by a fixed and globally unique 64-bit chunk handle assigned by the master at the time of chunk creation. Chunk servers store chunks on local disks as Linux files. For reliability, each chunk is replicated on multiple chunk servers. By default, there will three replicas and this value can be changed by user. The master maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks. It also controls system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunk servers. The master periodically communicates with each chunk server in Heart Beat messages to give it instructions and collect its state.

GFS client code linked into each application implements the file system API and communicates with the master and chunk servers to read or write data on behalf of the application. Clients interact with the master for metadata operations, but all data-bearing communication goes directly to the chunk servers.

Q.24. Explain the basic building blocks of Hadoop with a neat sketch. [R.G.P.V., May 2019 (VIII-Sem.)]

Ans. A fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in your network.

These daemons have specific roles; some exist only on one server, some exist across multiple servers. The daemons include –

(i) NameNode (ii) DataNode
(iii) Secondary NameNode (iv) JobTracker
(v) TaskTracker.

(i) NameNode – Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called the Hadoop Distributed File System, or HDFS. The NameNode is the master of HDFS that directs the slave DataNode daemons to perform the low-level I/O tasks. The NameNode is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed file system. The server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program. The negative aspect of NameNode is that if the NameNode fail then the entire Hadoop cluster will fail.

(ii) DataNode– Each slave machine in HDFS cluster will host a DataNode daemon to perform the reading and writing HDFS blocks to actual files on the local file system. When we want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in. Your client communicates directly with the DataNode daemons to process the local files corresponding to the blocks. A DataNode may communicate with other DataNode to replicate its data blocks for redundancy. Fig. 2.12 illustrates the roles of the NameNode and DataNodes.

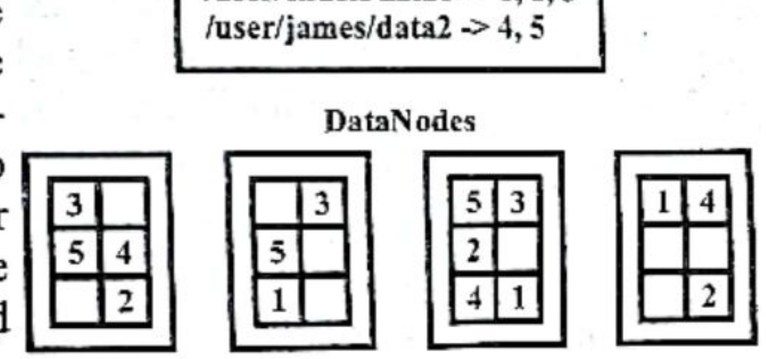

Fig. 2.12

Fig. 2.12, shows two data files, one at/user/chuck/data1 and another at/user/james/data2. The data1 file takes up three blocks, which we denote 1, 2, and 3, and the data2 file consists of blocks 4 and 5. The content of the files are distributed among the DataNodes. In the fig. 2.12 each block each has three replicas to ensure that if any one DataNode crashes or becomes inaccessible over the network, we will still be able to read the files. DataNodes are constantly reporting to the NameNode. The DataNodes continually communicate with the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete blocks from the local disk.

(iii) Secondary NameNode – The Secondary NameNode (SNN) is an assistant daemon for monitoring the state of the cluster HDFS. Each cluster has one SNN. The SNN communicates with the NameNode to take snapshots

of the HDFS metadata at intervals defined by the cluster configuration. The NameNode is a single point of failure for a Hadoop cluster, and the SNN snapshots help minimize the downtime and loss of data. A NameNode failure requires human involvement to reconfigure the cluster to use the SNN as the primary NameNode.

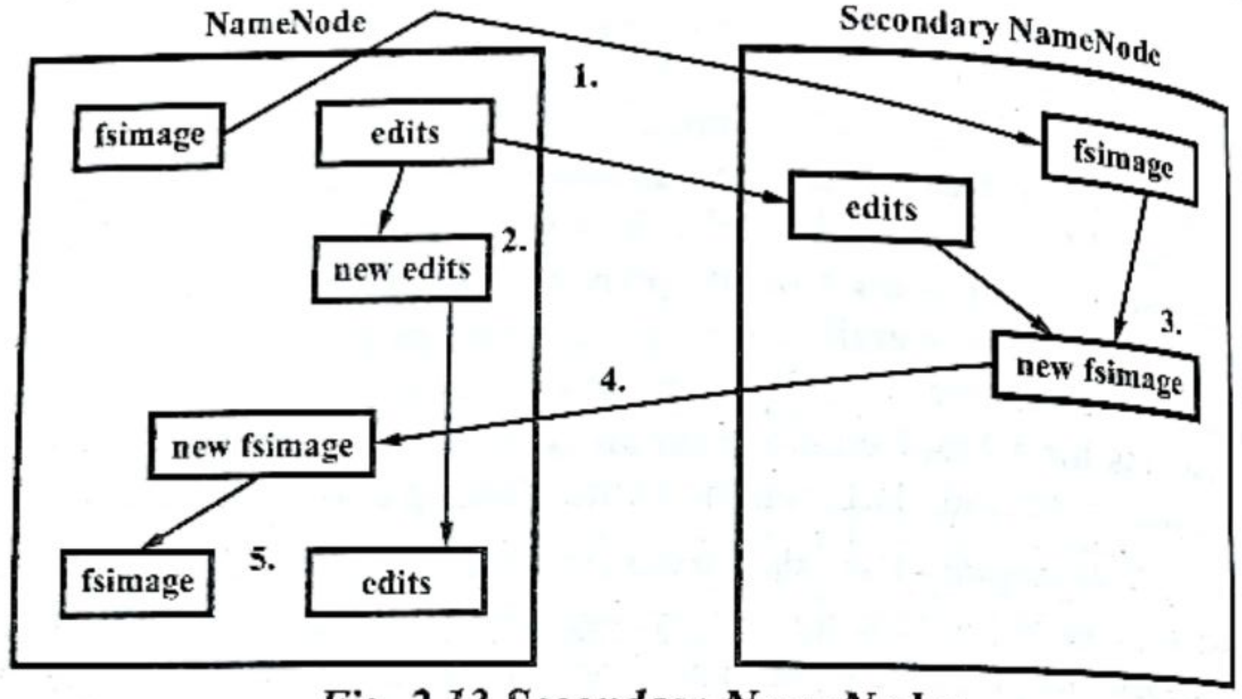

Fig. 2.13 Secondary NameNodes

(iv) JobTracker – The JobTracker daemon is the link between your application and Hadoop. Once we submit our code to the cluster, the JobTracker determines the execution plan by determining which files to process, assigns nodes to different tasks, and monitors all tasks as they are running. If a task fail, the JobTracker will automatically re-launch the task, possibly on a different node, upto a predefined limit of retries. There is only one JobTracker daemon per Hadoop cluster. It's typically run on a server as a master node of the cluster.

(v) TaskTracker – Just like the storage daemons, the computing daemons also follow a master/slave architecture – the JobTracker is the master overseeing the overall execution of a MapReduce job and the TaskTrackers

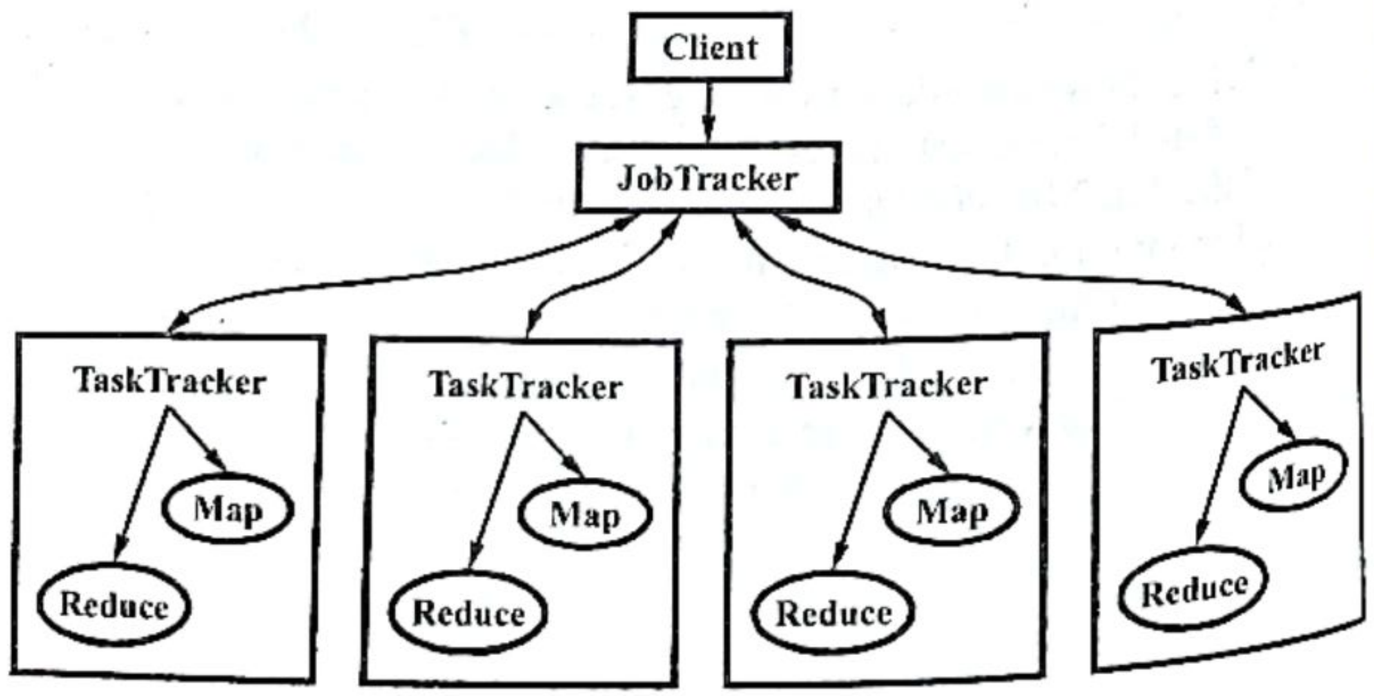

Fig. 2.14

manage the execution of individual tasks on each slave node. Fig. 2.14 illustrates this interaction.

One responsibility of the TaskTracker is to constantly communicate with the JobTracker. If the JobTracker fails to receive a heartbeat from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster. The following fig. 2.15 shows the topology of one typical Hadoop cluster.

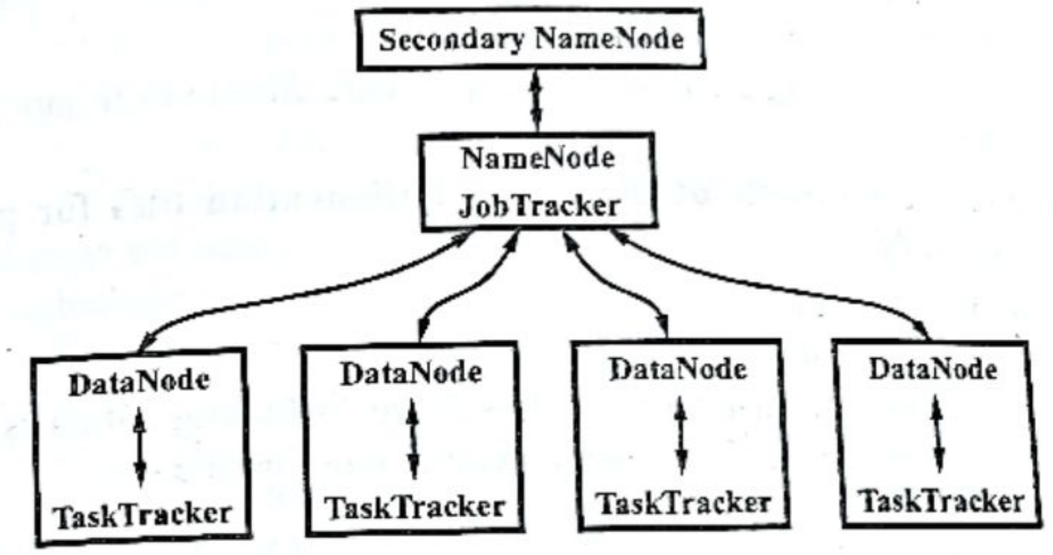

Fig. 2.15

This topology features a master node running the NameNode and JobTracker daemons and a standalone node with the SNN in case the master node fails. The slave machines each host a DataNode and TaskTracker, for running tasks on the same node where their data is stored.

Q.25. Explain the Hadoop distributed file system selecting appropriate execution modes.

Ans. A Hadoop cluster has following three types of modes –

(i) Local (standalone) mode (ii) Pseudo-distributed mode

(iii) Fully distributed mode

(i) Local/Standalone Mode – After downloading Hadoop in your system, by default, it is configured in a standalone mode and can be run as a single java process.

(a) When we first uncompressed the Hadoop source package, it does not consider our hardware setup. Hadoop chooses to be conservative and assumes a minimal configuration.

(b) All three XML files (or hadoop-site.xml before version 0.20) are empty under this default mode –

```
<?xml version = "1.0" ?>
<?xml-stylesheet type = "text/xsl" href = "configuration.xsl"?>
<!--Put site-specific property overrides in this file.-->
<configuration>
</configuration>
```

(c) Its primary use is for developing and debugging the application logic of a MapReduce program without the additional complexity of interacting with the daemons.

(ii) Pseudo-distributed Mode – It is a distributed simulation on single machine. Each Hadoop daemon such as HDFS,Yarn, MapReduce etc., will run as a separate java process. This mode is useful for development.

(a) This mode allowing us to examine memory usage, HDFS input/output issues, and other daemon interactions.

(b) Listing 2.1 provides simple XML files to configure a single server in this mode.

Listing 2.1 Example of the three configuration files for pseudo-distributed mode

core-site.xml

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xs1" href = "configuration.xs1"?>
<!--Put site-specific property overrides in this file.-->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost : 9000</value >
<description> The name of the default file system. A URI whose scheme and authority determine the FileSystem implementation.
</description>
</property>
</configuration>
```

mapred-site.xml

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xs1" href = "configuration.xs1"?>
<!--Put site-specific property overrides in the file.-->
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost : 9001</value>
<description> The host and port that the MapReduce job tracker runs at.</description>
</property>
</configuration>
```

hdfs-site.xml

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xs1" href = "configuration.xs1"?>
<!--Put site-specific property overrides in this file.-->
```



```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
<description> The actual number of replications can be specified when the file is created.</description>
</property>
</configuration>
```

(c) In core-site.xml and mapred-site.xml we specify the hostname and port of the NameNode and the JobTracker, respectively.

(d) In hdfs-site.xml we specify the default replication factor for HDFS.

(e) We must also specify the location of the Secondary NameNode in the masters file and the slave nodes in the slaves file –

```
[hadoop-user@master] $ cat masters
localhost
[hadoop-user@master] $ cat slaves
localhost
```

(f) While all the daemons are running on the same machine, they still communicate with each other using the same SSH protocol as if they were distributed over a cluster.

(g) We need to check the machine allows you to ssh back to itself.

```
[hadoop-user@master] $ ssh localhost
```

(h) If the above command results an error then set up shh by using the following commands

```
[hadoop-user@master] $ ssh-keygen -t dsa -P " –f ~/.ssh/id_dsa
[hadoop-user@master] $ cat ~/.ssh/id_dsa.pub>>~/.ssh/authorized_keys
```

(i) Next, we need to format the namenode by using the following command

```
[hadoop-user@master] $ bin/hadoop namenode-format
```

(j) To launch the daemons by use of the start-all.sh script. The Java jps command will list all daemons to verify the setup was successful.

```
[hadoop-user@master] $ bin/start-all.sh
[hadoop-user@master] $ jps
26893 Jps
26832 TaskTracker
26620 SecondaryNameNode
26333 NameNode
26484 DataNode
26703 JobTracker
```

(k) We can shut down all the daemons using the command

```
[hadoop-user@master] $ bin/stop-all.sh
```

(iii) Fully Distributed Mode –

(a) An actual Hadoop cluster runs in the third mode, the fully distributed mode that emphasizing the benefits of distributed storage and distributed computation.

(b) In the discussion below we will use the following server names –

(1) master – The master node of the cluster and host of the NameNode and JobTracker daemons.

(2) backup – The server that hosts the Secondary NameNode daemon.

(3) hadoop1, hadoop2, hadoop3 – The slave boxes of the cluster running both Datanode and TaskTracker daemons.

Q.26. What is parallel and distributed processing ? Explain with example. [R.G.P.V., May 2019 (VIII-Sem.)]

Ans. **Parallel Data Processing –** Parallel data processing involves the simultaneous execution of multiple sub-tasks that collectively comprise a large task. The goal is to reduce the execution time by dividing a single larger task into multiple smaller tasks that run concurrently.

Although parallel data processing can be achieved through multiple networked machines, it is more typically achieved within the confines of a single machine with multiple processors or cores. A task can be divided into three sub-tasks that are executed in parallel on three different processors within the same machine as shown in fig. 2.16.

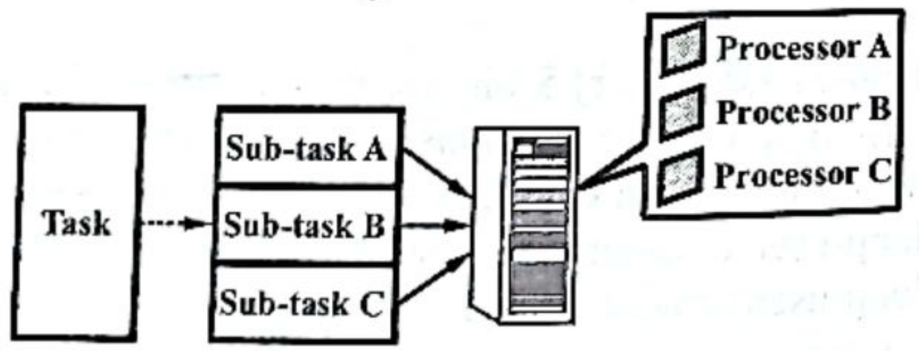

Fig. 2.16 An Example of Parallel Data Processing

Distributed Data Processing – Distributed data processing is closely related to parallel data processing in that the same principle of "divide-and-conquer" is applied. However, distributed data processing is always achieved through physically separate machines that are networked together as a cluster. In fig. 2.17, a task is divided into three sub-tasks that are then executed on three different machines sharing one physical switch.

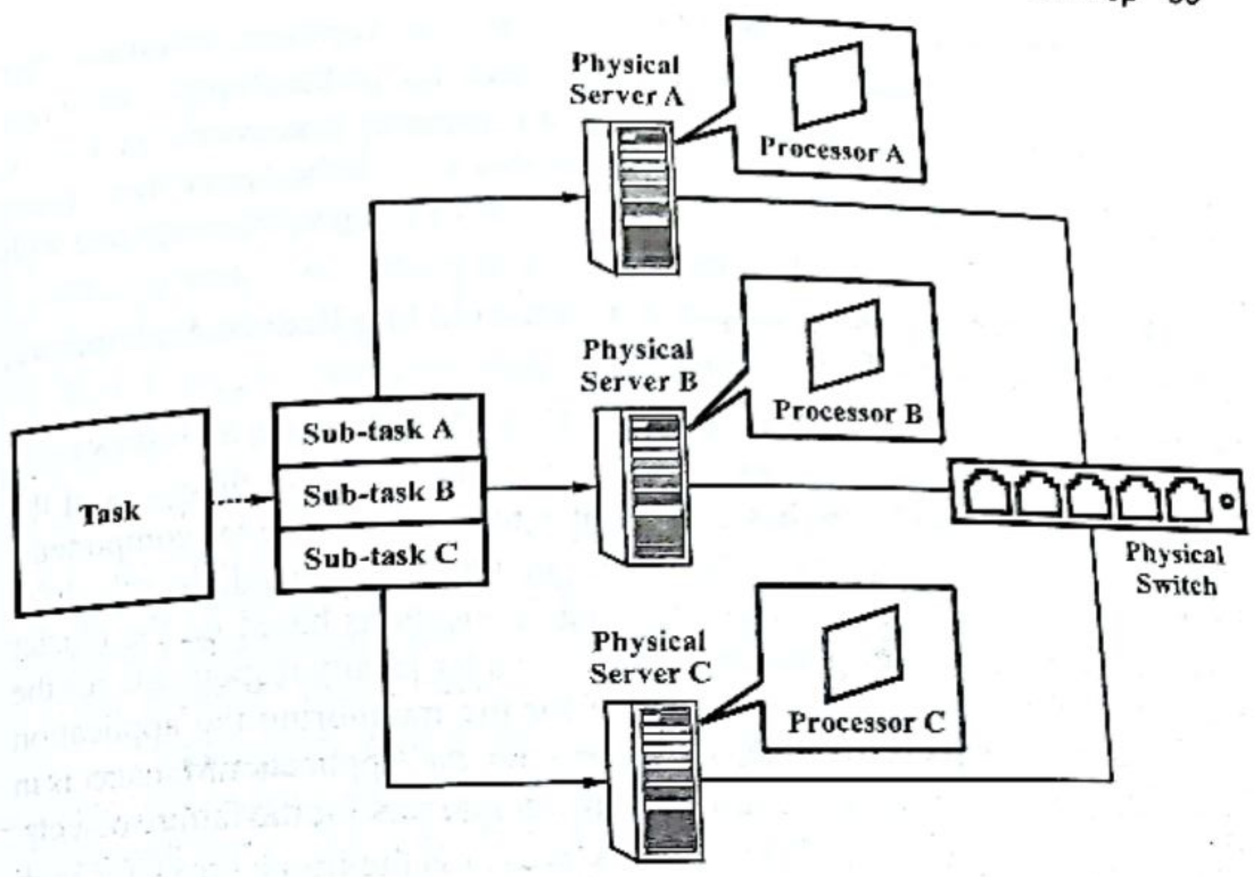

Fig. 2.17 An Example of Distributed Data Processing

Q.27. What is YARN ? Explain its architecture with the help of diagram.

Ans. To overcome limitations of MapReduce the designers have put forward the next generation of MapReduce – YARN. The main purpose of YARN is to divide the tasks for the JobTracker. In YARN, the resources are managed by the

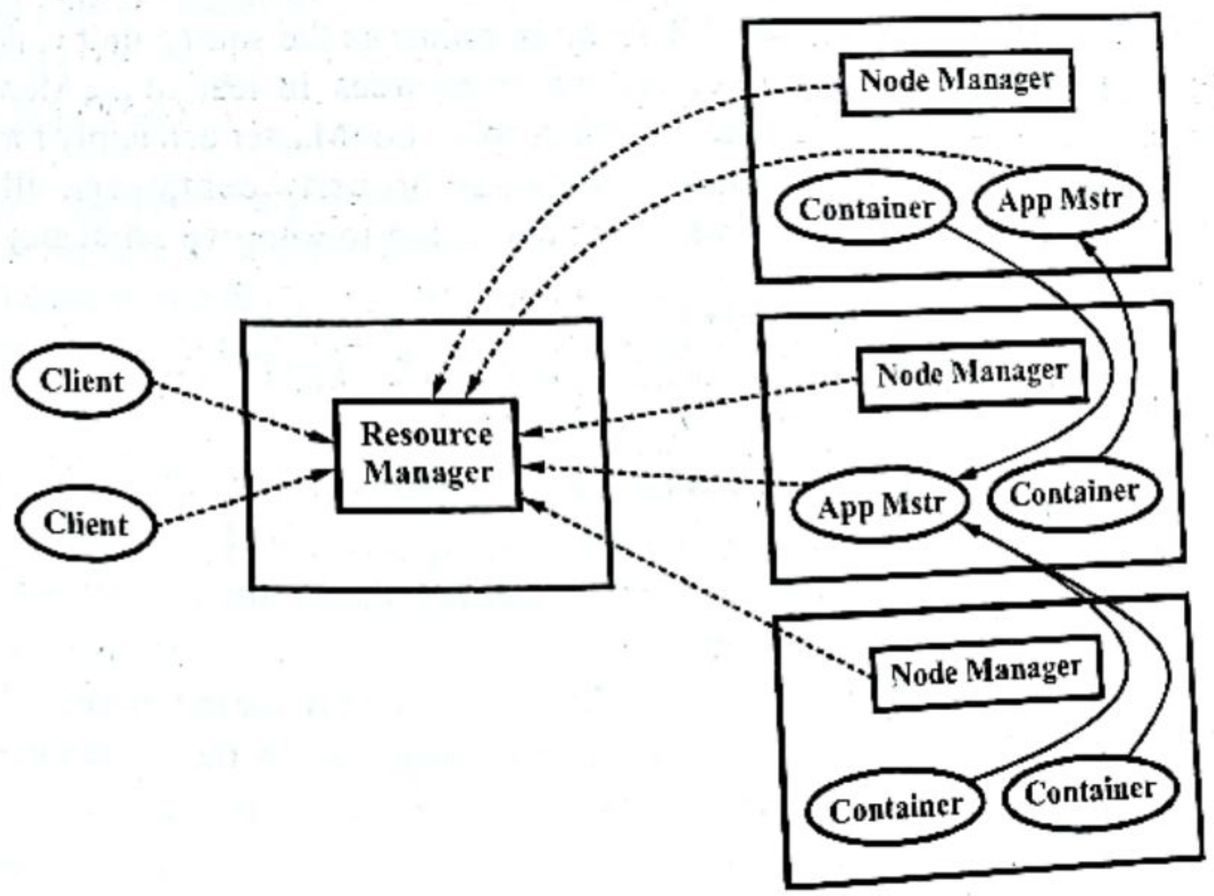

Fig. 2.18 YARN Architecture

ResourceManager and the jobs are traced by the ApplicationMaster. The TaskTracker has become the NodeManager. Hence, the global ResourceManager and the local NodeManager compose the data computing framework. In YARN, the ResourceManager will be the resources distributor while the ApplicationMaster is responsible for the communication with ResourceManager and cooperate with the NodeManager to complete the tasks.

YARN Architecture – Compared with the old MapReduce Architecture, YARN is more structured and simple.

The four core components of the YARN architecture are as follows –

(i) ResourceManager – According to the different functions of the ResourceManager, designers has divided it into two lower level components viz., the scheduler and the ApplicationManager. On the one hand, the scheduler assigns the resource to the different running applications based on the cluster size, queues, and resource constraints. The scheduler is only responsible for the resources allocation but is not responsible for the monitoring the application implementation and task failure. On the other hand, the ApplicationManager is in charge of receiving jobs and redistributing the containers for the failure objects.

(ii) NodeManager – The NodeManager is the frame proxy for each node. It is responsible for launching the application container, monitoring the usage of the resource, and reporting all the information to the Scheduler.

(iii) ApplicationMaster – The ApplicationMaster is cooperating with the NodeManager to put tasks in the suitable containers to run the tasks and monitor the tasks. When the container has errors, the ApplicationMaster will apply for another resource from the Scheduler to continue the process.

(iv) Container – In YARN, the container is the source unit which is the available node splitting the organization resources. Instead of the Map and Reduce source pools in MapReduce, the ApplicationMaster can apply for any numbers of the container. Due to the same property containers, all the containers can be exchanged in the task execution to improve efficiency.

Q.28. What are the advantages of YARN ?

Ans. There are four main advantages of YARN compared to the MapReduce –

(i) YARN greatly enhances the scalability and availability of the cluster by distributing the tasks to the JobTracker. The ResourceManager and the ApplicationMaster greatly relieves the bottleneck of the JobTracker and the safety problems in the MapReduce.

(ii) In YARN, the ApplicationMaster is a customized component. That means that the users can write their own program based on the programming model. This makes the YARN more flexible and suitable for wide use.

(iii) YARN, on the one hand, supports the program to have a specific checkpoint. It can ensure that the ApplicationMaster can reboot immediately

based on the status which was stored on HDFS. On the other hand, it uses the ZooKeeper on the ResourceManager to implement the failover. When the ResourceManager receives errors, the backup ResourceManager will reboot quickly. These two measures improve the availability of YARN.

The cluster has the same containers are the Reduce and Map pools in MapReduce. Once there is a request for resources, the Scheduler will assign the available resources in the cluster to the tasks and regard the resource type. It will increase the utilization of the cluster resources.

Q.29. Define MapReduce. What is the role of map function and reduce function ?

Ans. MapReduce is widely used in logs analysis, data sorting, and specific data searching. MapReduce is the core technology of Hadoop. It provides a parallel computing model for the big data and supports a set of programming interfaces for the developers.

MapReduce is a standard functional programming model. This kind of model has been used in the early programming languages, such as Lisp. The core of the calculation model is that can pass the function as the parameter to another function. Through multiple concatenations of functions, the data processing can turn into a series of function execution. MapReduce has two stage of processing. The first one is Map and the other one is Reduce. The reason why the MapReduce is popular is that it is very simple, easy to implement, and offers strong expansibility. MapReduce is suitable for processing the big data because it can be processed by the multiple hosts at the same time to gain a faster speed.

Map Function – Each map function receives the input data split as a set of (key, value) pairs to process and produce the intermediated (key, value) pairs.

Reduce Function – The reduce worker iterates over the grouped (key, value) pairs, and for each unique key, it sends the key and corresponding values to the Reduce function. Then this function processes its input data and stores the output results in predetermined files in the user's program.

Q.30. Write short note on employing Hadoop MapReduce. Also describe it's features and applications.

Ans. A distributed data processing framework is called MapReduce. In other words, MapReduce is a framework for processing parallelizable problems across large datasets using a large number of computers, collectively referred to as a cluster or a grid. Processing can occur on data stored either in a filesystem or in a database (unstructured & structured).

The features of Hadoop MapReduce are as follows –

(i) The programming model is simple yet expressive. A large number of tasks can be expressed as MapReduce jobs. The model is independent of

the underlying storage system and is able to process both structured and unstructured data.

(ii) It achieves scalability through block-level scheduling. The run-time system automatically splits the input data into even-sized blocks and dynamically schedules the data blocks to the available nodes for processing.

(iii) It offers fault tolerance whereby only tasks on failed nodes have to be restarted.

The applications of MapReduce are as follows –

(i) Large scale machine learning problems
(ii) Clustering problems for Google News
(iii) Extracting data for reports of popular queries
(iv) Extracting properties of Web pages for various purposes
(v) Processing satellite image data
(vi) Language model processing for statistical machine translation
(vii) Large-scale graph computations
(viii) Index building for various search operations
(ix) Spam detection
(x) Various data mining applications.

Q.31. Discuss in detail about Hadoop MapReduce.

Ans. A Hadoop MapReduce job mainly consists of two user-defined functions – Map and Reduce. The input of a Hadoop's MapReduce job is a set of key-value pairs (k, v) and the map function is called for each of these pairs. The map function produces zero or more intermediate key-value pairs (k', v'). Then, the Hadoop's MapReduce framework groups these intermediate key-value pairs by intermediate key k' and calls the reduce function for each group. Finally, the reduce function produces zero or more aggregated results.

The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform they are Mappers and Reducers –

Map Job – The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

The map function produces zero or more intermediate key-value pairs (key', value'). Map function takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain. After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, creating one group for each key.

$$\text{Map}(k1, v1) \rightarrow \text{list}(k2, v2)$$

Reduce Job – The second is the reduce job, which takes the output of a map job as input and combines those data tuples into a smaller set of tuples. The Reduce function is then applied in parallel to each group, which in turn produces a collection of values in the same domain.

$$\text{Reduce}(k2, \text{list}(v2)) \rightarrow \text{list}(v3)$$

Reduce stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

5 Step Process of MapReduce –

Step 1 – Prepare the Map( ) Input – Set of key-value pairs (k, v)

Step 2 – Run the User-provided Map( ) Code – Generate intermediate key-value pairs (key', value') and lists $\text{Map}(k1, v1) \rightarrow \text{list}(k2, v2)$

Step 3 – "Shuffle" the Map Output to the Reduce Processors – The MapReduce system designates Reduce processors, assigns the k2 key-value each processor should work on. That is, worker nodes redistribute data based on the output keys (k2) such that all data belonging to one key is located on the same worker node.

Step 4 – Run the User-provided Reduce( ) Code – Reduce( ) is run exactly once for each k2 key value produced by the Map step.

Step 5 – Produce the Final Output – The MapReduce system collects all the Reduce output, and sorts it by k2 to produce the final outcome.

As the sequence of the name MapReduce implies, the reduce job is always performed after the map job. Below fig. 2.19 shows the MapReduce work process.

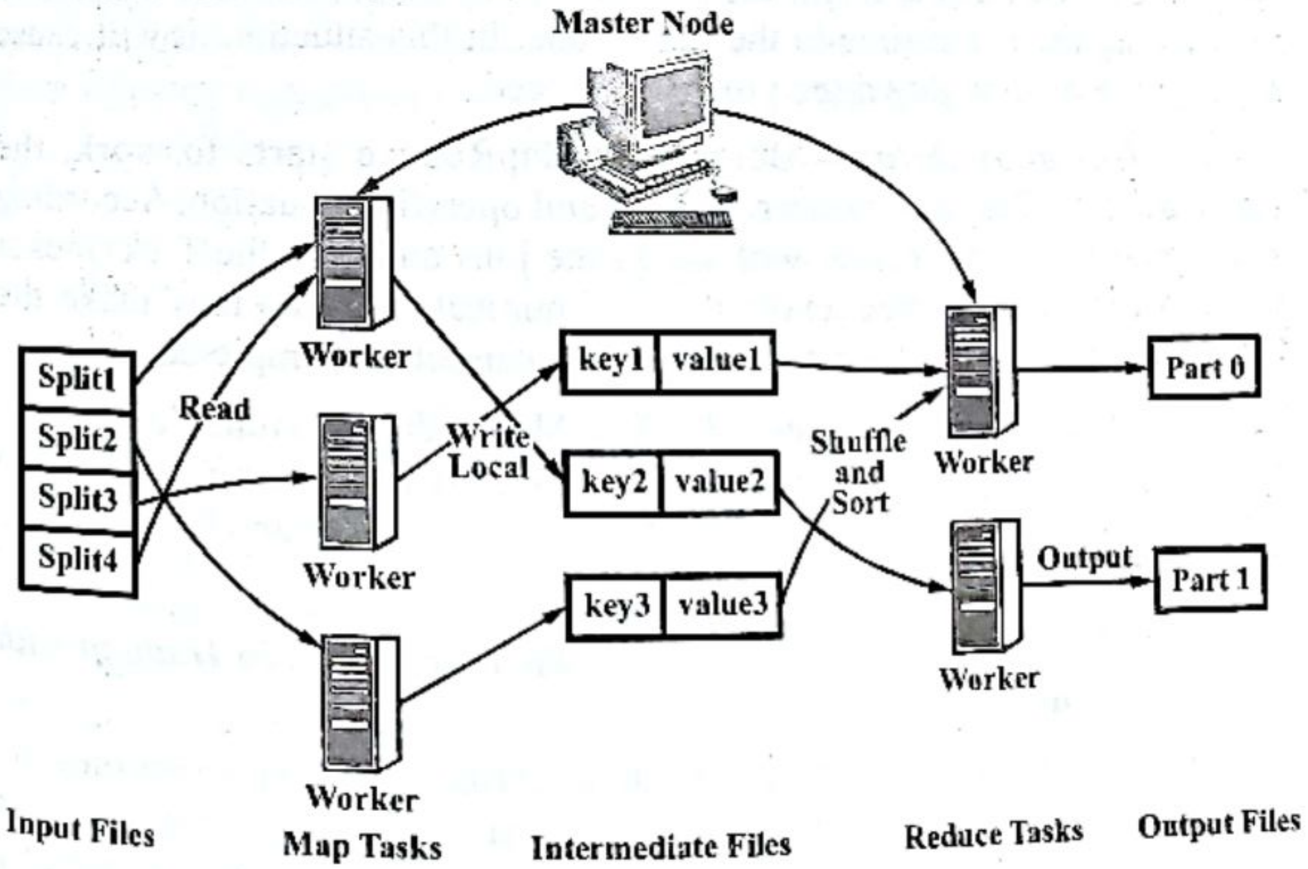

Fig. 2.19 MapReduce Working Process

Q.32. Write short note on uses of MapReduce.

Ans. Uses of MapReduce are as follows –

At Google –

(i) Index building for Google Search
(ii) Article clustering for Google News
(iii) Statistical machine translation.

At Yahoo! –

(i) Index building for Yahoo! Search
(ii) Spam detection for Yahoo! Mail.

At Facebook –

(i) Ad optimization
(ii) Spam detection.

Q.33. Give the limitations of MapReduce.

Ans. There are following four main limitations of the MapReduce –

(i) The Bottleneck of JobTracker – The JobTracker should be responsible for jobs allocation, management, and scheduling. It should also communicate with all the nodes to know the processing status. It is obvious that the JobTracker which is unique in the MapReduce, performs too many tasks. If the number of clusters and the submission jobs increase rapidly, it will cause network bandwidth consumption. As a result, the JobTracker will reach bottleneck and this is the core risk of MapReduce.

(ii) Node Failure – Because the jobs allocation information is too simple, the TaskTracker might assign a few tasks that need more sources or need a long execution time to the same node. In this situation, it will cause node failure or slow down the processing speed.

(iii) Jobs Delay – Before the MapReduce starts to work, the TaskTracker will report its own resources and operation situation. According to the report, the JobTracker will assign the jobs and then the TaskTracker starts to run. As a consequence, the communication delay may make the JobTracker to wait too long so that the jobs cannot be completed in time.

(iv) Inflexible Framework – The MapReduce currently allows the users to define its own functions for different processing stages, the MapReduce framework still limits the programming model and the resources allocation.

Q.34, Explain the overview of MapReduce execution in Hadoop with the help of example.

Ans. The map tasks are distributed across multiple machines by automatically partitioning the input data into a set of M splits. These splits can be processed in parallel by different machines. Reduce tasks are distributed by partitioning the intermediate key space into R pieces using a partitioning function

(e.g. hash (key) mod R). The number of partitions (R) and the partitioning function are specified by the user.

When the user program calls the MapReduce( ) function, the following occurs –

(i) The MapReduce library splits the input files into M pieces (usually 16-64 MB per piece) and starts up many copies of the program on a cluster of machines.

(ii) One of the copies of the program is the master as previously specified. The rest are workers that are assigned work by the master. There are M map tasks and R reduce tasks to assign. The master picks the idle workers and assigns each one either a map or a reduce task.

(iii) A worker assigned with a map task reads the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-defined map function. The intermediate key/value pairs produced by the function are buffered in memory.

(iv) Periodically, these buffered pairs are written to the local disk and partitioned into R regions by the partitioning function. The locations of these pairs are passed back to the master who is responsible for forwarding these locations to the reduce workers.

(v) When a reduce worker is notified about these locations, it uses remote procedure calls (RPCs) to read the buffered data from the disks of the map workers. When a reduce worker has read all intermediate data for its partition, it sorts it by the intermediate keys to group together all occurrences of the same key. If the amount of intermediate data is too large to fit in the memory, an external sort is used.

(vi) The reduce worker iterates over the sorted intermediate data and for each unique intermediate key, it passes the key and the corresponding set of intermediate values to the user's reduce function. The output of the reduce function is appended to a final output file for this reduce partition.

(vii) When all map and reduce tasks have completed, the master wakes up the user program. At this point, the MapReduce call in the user program returns back to the user code. After successful completion, the output of the MapReduce execution is available in the R output files.

To detect failure, the master pings every worker periodically. If no response is received from a worker in a certain amount of time, the master marks the worker as failed. Any map tasks completed by the worker are reset back to their initial idle state, and therefore become eligible for scheduling on other workers. Similarly, any map task or reduce task in progress on a failed worker is also reset to idle and becomes eligible for rescheduling.

Completed map tasks are re-executed when failure occurs because their output is stored on the local disk(s) of the failed machine and is therefore inaccessible. Completed reduce tasks do not need to be re-executed since their output is stored in a global file system.

Example of a MapReduce – Assume we have five files, and each file contains two columns, a key and a value in Hadoop terms that represent a city

and the corresponding temperature recorded in that city for the various measurement days. This example is made very simple so it's easy to follow. We can imagine that a real application contain millions or even billions of rows.

Delhi, 31
Mumbai, 32
Chennai, 33
Calcutta, 32
Delhi, 24
Calcutta, 34
Chennai, 38
Delhi, 27
Calcutta, 33
Chennai, 37.

Let out of all the data collected, we have to find the maximum temperature for each city across all of the data files (note that each file might have the same city represented multiple times). Using the MapReduce framework, we can break this down into five map tasks, where each mapper works on one of the five files and the mapper task goes through the data and returns the maximum temperature for each city. The results produced from mapper task for the above data would look like this –

(Delhi, 31)
(Mumbai, 32)
(Chennai, 38)
(Calcutta, 34)

Q.35. Explain the architecture of MapReduce in Hadoop.

Ans. The Hadoop MapReduce MRv1 framework is based on a centralized master/slave architecture. The architecture utilizes a single master server (JobTracker) and several slave servers (Task Tracker's). The JobTracker represents a centralized program that keeps track of the slave nodes, and provides an interface infrastructure for job submission. The Task Tracker executes on each of the slave nodes where the actual data is normally stored. Users submit MapReduce jobs to the JobTracker, which inserts the jobs into the pending

```
Map Function
map(input_record) {
...
emit(k1, v1)
...
emit(k2, v2)
...
}

Reduce Function
reduce(key, values) {
while(values.has_next) {
aggregate-merge(values.next)
  }
collect(key, aggregate)
}
```

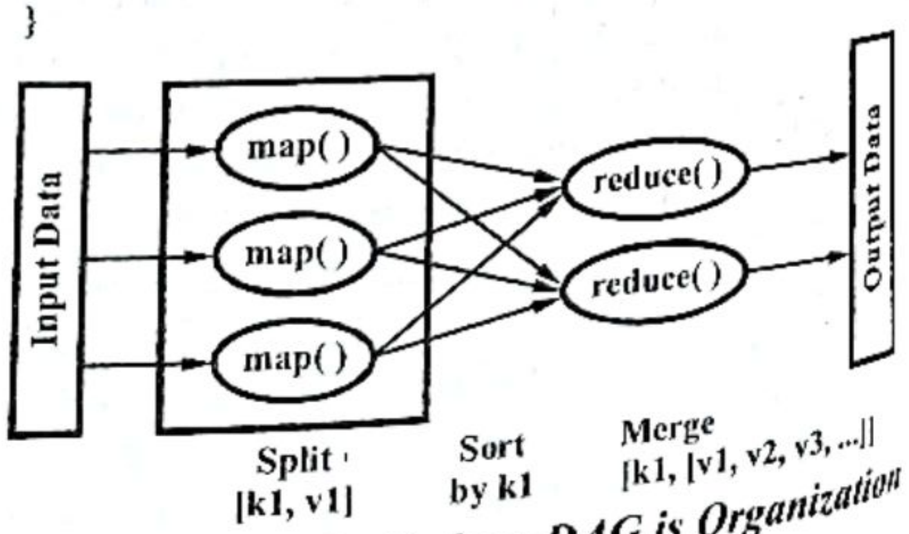

Fig. 2.20 The MapReduce DAG is Organization

jobs queue and executes them (normally) on a FIFO basis (it has to be pointed out that other job schedulers are available – see Hadoop Schedulers below). The JobTracker manages the map and reduce task assignments with the TaskTracker's. The TaskTracker's execute the jobs based on the instructions from the JobTracker and handle the data movement between the maps and reduce phases, respectively. Any map/reduce construct basically reflects a special form of a Directed Acyclic Graph (DAG). A DAG can execute anywhere in parallel, as long as one entity is not an ancestor of another entity. In other words, parallelism is achieved when there are no hidden dependencies among shared states. In the MapReduce model, the internal organization is based on the map function that transforms a piece of data into entities of [key, value] pairs. Each of these elements is sorted (via their key) and ultimately reaches the same cluster node where a reduce function is used to merge the values (with the same key) into a single result (see code below). The Map/Reduce DAG is organized as depicted in fig. 2.20.

The Hadoop MapReduce framework is based on a pull model, where multiple TaskTracker's communicate with the JobTracker requesting tasks (either map or reduce tasks). After an initial setup phase, the JobTracker is informed about a job submission. The JobTracker provides a job ID to the client program, and starts allocating map tasks to idle TaskTracker's requesting work items (see fig. 2.21) Each TaskTracker contains a defined number of task slots based on the capacity potential of the system. Via the heartbeat protocol, the JobTracker knows the number of free slots in the TaskTracker (the TaskTracker's send heartbeat messages indicating the free slots-true for the FIFO scheduler). Hence, the JobTracker can determine the appropriate job setup for a TaskTracker based on the actual availability behaviour. The assigned TaskTracker will fork a MapTask to execute the map processing cycle (the MapReduce framework spawns 1 MapTask for each InputSplit generated by the InputFormat). In other words, the MapTask extracts the input data from the splits by using the RecordReader and InputFormat for the job, and it invokes the user provided map function, which emits a number of [key, value] pairs in the memory buffer.

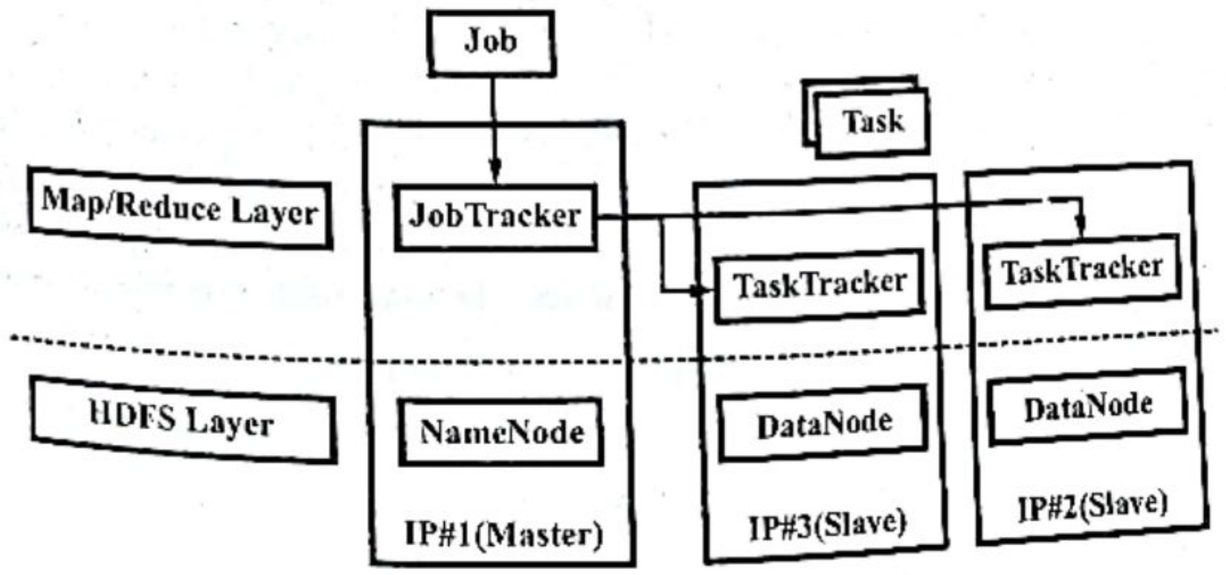

Fig. 2.21 Architecture of MapReduces

After the MapTask finished executing all input records, the commit process cycle is initiated by flushing the memory buffer to the index and data file pair. The next step consists of merging all the index and data file pairs into a single construct that is (once again) being divided up into local directories. As some map tasks are completed, the JobTracker starts initiating the reduce tasks phase. The TaskTracker's involved in this step download the completed files from the map task nodes, and basically concatenate the files into a single entity. As more map tasks are being completed, the JobTracker notifies the involved TaskTracker's, requesting the download of the additional region files and to merge the files with the previous target file. Based on this design, the process of downloading the region files is interleaved with the on-going map task procedures.

Eventually, all the map tasks will be completed, at which point the JobTracker notifies the involved TaskTracker's to proceed with the reduce phase. Each TaskTracker will fork a ReduceTask (separate JVM's are used), read the downloaded file (that is already sorted by key), and invoke the reduce function that assembles the key and aggregated value structure into the final output file (there is one file per reducer node). Each reduce task (or map task) is single threaded, and this thread invokes the reduce [key, values] function in either ascending or descending order. The output of each reducer task is written to a temp file in HDFS. When the reducer finishes processing all keys, the temp file is automatically renamed into its final destination file name.

As the MapReduce library is designed to process vast amounts of data by potentially utilizing hundreds or thousands of nodes, the library has to be able to gracefully handle any failure scenarios. The TaskTracker nodes periodically report their status to the JobTracker that oversees the overall job progress. In scenarios where the JobTracker has not been contacted by a TaskTracker for a certain amount of time, the JobTracker assumes a TaskTracker node failure and hence, reassigns the tasks to other available TaskTracker nodes. As the results of the map phase are stored locally, the data will no longer be available if a TaskTracker node goes offline.

In such a scenario, all the map tasks from the failed node (regardless of the actual completion percentage) will have to be reassigned to a different TaskTracker node that will re-execute all the newly assigned splits. The results of the reduce phase are stored in HDFS and hence, the data is globally available even if a TaskTracker node goes offline. Hence, in a scenario where during the reduce phase a TaskTracker node goes offline, only the set of incomplete reduce tasks have to be reassigned to a different TaskTracker node for re-execution.

Q.36. Explain the dataflow and control flow of MapReduce.

Ans. MapReduce is the heart of Hadoop. It is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. The framework processes the feature of data locality. Data locality means movement of algorithm to the data instead of data to algorithm.

When the processing is done on the data algorithm is moved across the DataNodes rather than data to the algorithm. The architecture is so constructed because Moving Computation is Cheaper than Moving Data. It is fault tolerant which is achieved by its daemons using the concept of replication. The daemons associated with the MapReduce phase are jobTracker and taskTrackers.

Map-Reduce jobs are submitted on JobTracker. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status whether the node is dead or alive. Whenever there is negative status, the JobTracker assigns the task to another node on the replicated data of the failed node stored in this node.

Let's see how the data flows –

MapReduce has a simple model of data processing – Inputs and outputs for the map and reduce functions are key-value pairs. The map and reduce functions in Hadoop MapReduce have the following general form –

map – (K1, V1) → list(K2, V2)

reduce – (K2, list(V2)) → list(K3, V3)

Now before processing it needs to know on which data to process, this is achieved with the InputFormat class. InputFormat is the class which selects file from HDFS that should be input to the map function. An after running setup( ), the nextKeyValue( ) is called repeatedly on the Context, (which delegates to the identically-named method on the RecordReader) to populate the key and value objects for the mapper. The key and value are retrieved from the Record Reader by way of the Context, and passed to the map( ) method for it to do its work. Input to the map function which is the key-value pair (K, V) gets processed as per the logic mentioned in the map code.

When the reader gets to the end of the stream, the nextKeyValue( ) method returns false, and the map task runs its cleanup( ) method.

The output of the mapper is sent to the partitioner. Partitioner controls the partitioning of the keys of the intermediate map-outputs. The key (or a subset of the key) is used to derive the partition, typically by a hash function. The total number of partitions is the same as the number of reduce tasks for the job. Hence this controls which of them reduce tasks the intermediate key (and hence the record) is sent for reduction. The use of partitioners is optional.

Q.37. Write short note on subdividing data in preparation for Hadoop MapReduce.

Ans. Central to the scalability of Apache Hadoop is the distributed processing framework known as MapReduce. MapReduce helps programmers solve data-parallel problems for which the data set can be sub-divided into small parts and processed independently. MapReduce is an important advance because it allows ordinary developers, not just those skilled in high-performance

computing, to use parallel programming constructs without worrying above the complex details of intra-cluster communication, task monitoring, and failure handling. MapReduce simplifies all that.

The system splits the input data-set into multiple chunks, each of which is assigned a map task that can process the data in parallel. Each map task reads the input as a set of (key, value) pairs and produces a transformed set of (key, value) pairs as the output. The framework shuffles and sorts outputs of the map tasks, sending the intermediate (key, value) pairs to the reduce tasks, which group them into final results. MapReduce uses JobTracker and TaskTracker mechanisms to schedule tasks, monitor them, and restart any that fail.

The Apache Hadoop platform also includes the Hadoop Distributed File System (HDFS), which is designed for scalability and fault-tolerance. HDFS stores large files by dividing them into blocks (usually 64 or 128 MB) and replicating the blocks on three or more servers. HDFS provides APIs for MapReduce applications to read and write data in parallel. Capacity and performance can be scaled by adding Data Nodes, and a single NameNode mechanism manages data placement and monitors server availability. HDFS clusters in production use today reliably hold petabytes of data on thousands of nodes.

Q.38. Discuss in detail about Map and Reduce operation ?

Ans. The Map operation applies computation of key/value pairs in an input and Reduce operation combines all the result value that is computed

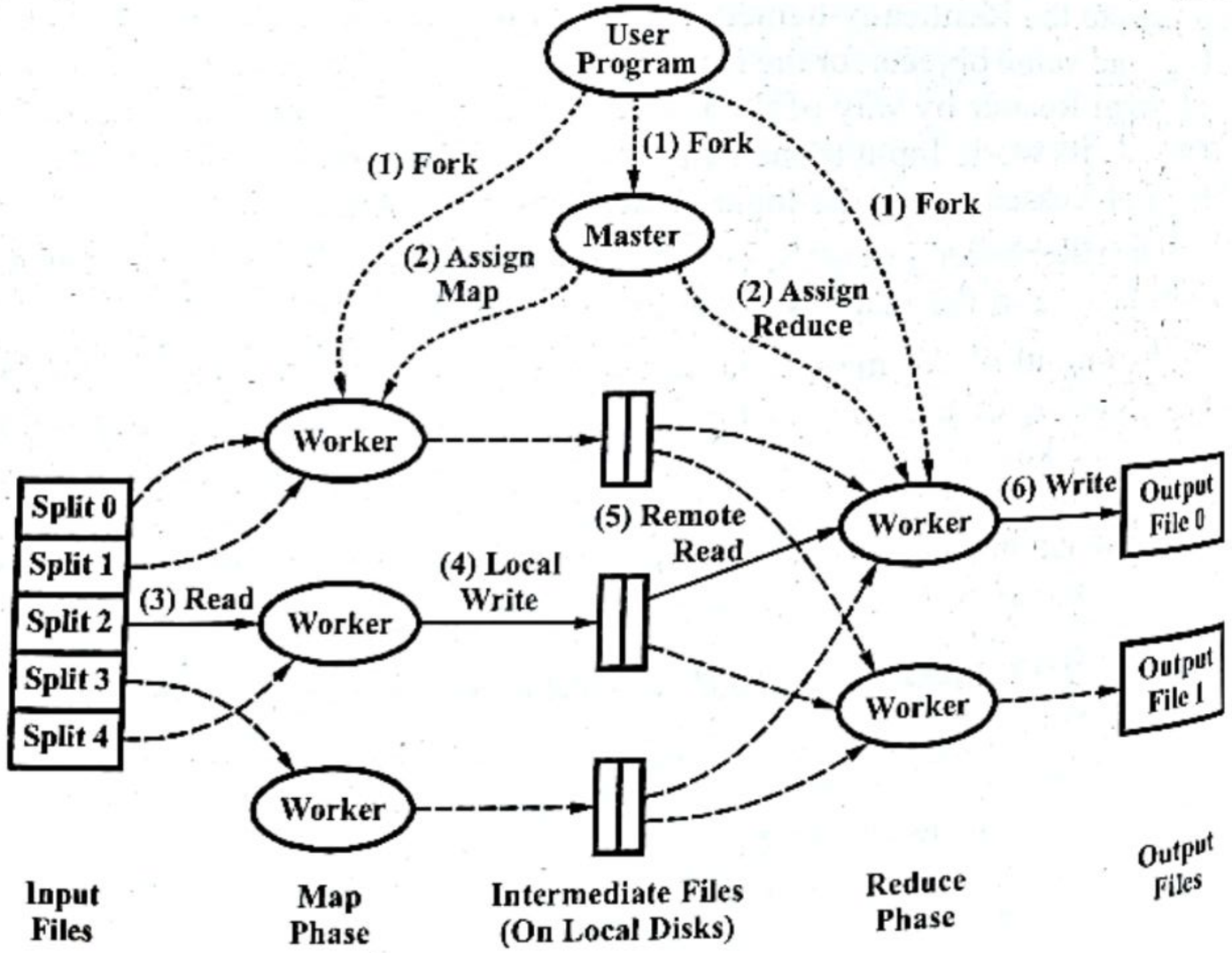

Fig. 2.22 Map and Reduce Operation

from the result of Map operation. As shown in fig. 2.22 below, the users divide the input files in different blocks of 128 MB size and these blocks generate the number of copies program in the clusters. Every cluster has different programs with one master node and several data nodes. Data nodes are also known as worker nodes and may be assigned Map work or Reduce work by the master node.

Once the user defines the input files, the master node assigns the worker node for Map function. Those worker nodes who are assigned for Map work reads files from different input files and writes the file in local disk. Once the Map worker nodes finished their work by writing the result in local disk, another sets of worker nodes are assigned for Reduce fur ction. The assigned worker nodes read the files from local disk and write it to the output files. In this way, the retrieved process is completed in the Hadoop MapReduce.

Q.39. Explain mapping data to the programming framework.

Ans. Many different higher-level programming frameworks have been developed. The most commonly implemented programming framework is the MapReduce framework. MapReduce is an emerging programming framework for data-intensive appli- cations proposed by Google. MapReduce borrows ideas from func- tional programming, where the programmer defines Map and Reduce tasks to process large sets of distributed data.

Implementations of MapReduce enable many of the most common calculations on large-scale data to be performed on computing clusters efficiently and in a way that is tolerant of hardware failures during computation. However MapReduce is not suit- able for online transactions.

The key strengths of the MapReduce progra- mming framework are the high degree of parallelism combined with the simpli- city of the programming framework and its applicability to a large variety of application domains. This requires dividing the workload across a large number of machines. The degree of parallelism depends on the input data size. The map function processes the input pairs (key1, value1) returning some other intermediary pairs (key2, value2). Then the intermediary pairs are grouped together according to their key. The reduce function will output some new key-value pairs of the form (key3, value3). Fig. 2.23 shows an example of a MapReduce algorithm used to count words in a file. In this example the map input key is the provided data chunk with a value of 1. The map output key is the word itself and the value is 1 every time the word exists in the processed data chunk. The reducers perform the aggregation of the key-values pair output from the maps and output a single value for every key, which in this case is a count for every word. Fig. 2.23 provides

further explanation of the generation of the key-value pairs produced during the processing phases of the WordCount MapReduce program.

High performance is achieved by breaking the processing into small units of work that can be run in parallel across potentially hundreds or thousands of nodes in the cluster. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

MapReduce programs are usually written in Java; however they can also be coded in languages such as C++, Perl, Python, Ruby, R, etc. These programs may process data stored in different file and database systems.

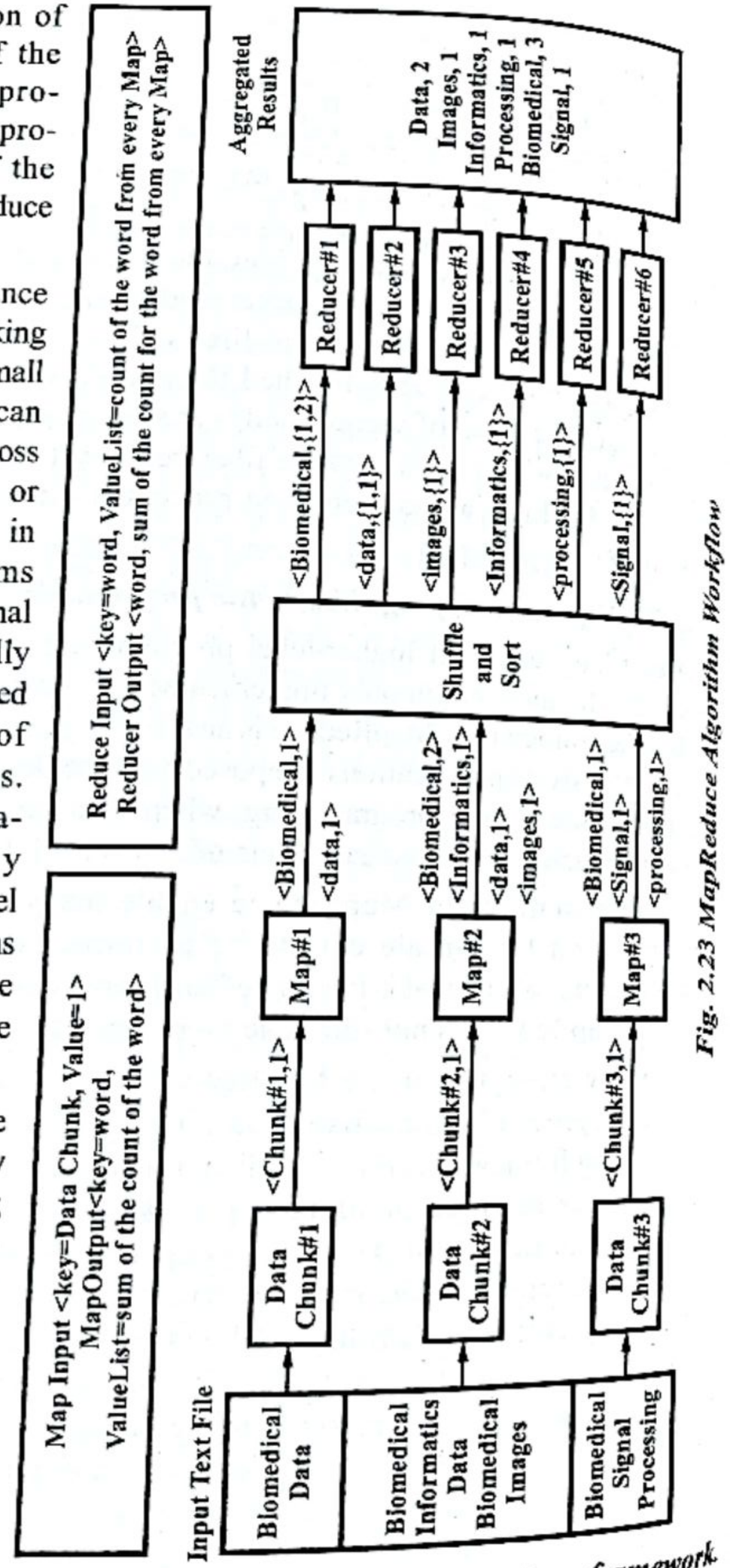

Fig. 2.23 MapReduce Algorithm Workflow

Q.40. Write a program to the sample data using MapReduce framework.

Ans. package hadoop;

```
import java.util.*;
import java.io.IOException;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class ProcessUnits
{
    //Mapper class
    public static class E_EMapper extends MapReduceBase implements
    Mapper<LongWritable,        /*Input key Type*/
    Text,                       /*Input value Type*/
    Text,                       /*Output key Type*/
    IntWritable>                /*Output value Type*/
    {
    //Map function
    public void map(LongWritable key, Text value, OutputCollector <Text,
IntWritable> output, Reporter reporter) throws IOException
    {
        String line = value.toString( );
        String lasttoken = null;
        StringTokenizer s = new StringTokenizer(line, "\t");
        String year = s.nextToken( );
            while(s.hasMoreTokens( ))
        {
        lasttoken = s.nextToken( );
    }
        int avgprice = Integer.parseInt(lasttoken);
    output.collect(new Text(year), new IntWritable(avgprice));
    }
}
    //Reducer class
  public static class E_EReduce extends MapReduceBase implements
Reducer<Text, IntWritable, Text, IntWritable>
  {
    //Reduce function
  public void reduce(Text key, Iterator <IntWritable> values,
    OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException
```

```
    {
        int maxavg=30;
        int val=Integer.MIN_VALUE;
        while(values.hasNext( ))
        {
            if((val=values.next( ).get( ))>maxavg)
            {
            output.collect(key, new IntWritable(val));
            }
        }
    }
}
```

//Main function

```
public static void main(String args[ ]) throws Exception
{
    JobConf conf = new JobConf(ProcessUnits.class);
    conf.setJobName("max_electricityunits");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(E_EMapper.class);
    conf.setCombinerClass(E_EReduce.class);
    conf.setReducerClass(E_EReduce.class);
    conf.setInputFormat(TextInputFromat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]);
    FileOutputFormat.setOutputPath(conf, new Path(args[1]);
    JobClient.runJob(conf);
    }
}
```

Output –

Below is the output generated by the MapReduce program.

```
1981  34
1984  40
1985  45
```

Q.41. Explain the wordcount program with example.

Ans. Wordcount is a Hello World program of the Hadoop world. In this program, the word is counted from the file in such a way that it will count how many times the word is repeated in the file. In wordcount program, the words are sorted in following steps –

(i) Input – In this step, the data are copied in the HDFS as an input. These inputs contain the words in any format.

(ii) Splitting – Once the data are copied as input in the HDFS, the data are split into different blocks depending on the sizes of the file and block. If the size of the file is 500 MB and the size of the block is 128 MB, the file will be splitted into four blocks. In these blocks, the file is arranged in such a way that every word is separated by space, comma, etc.

(iii) Mapping – The counting of words occurs in the phase. The words are counted and given a 1 value for every word. In this step, the results are arranged in the form of key and value pair. Each pair will have one key word and value 1. Suppose if the file has "This is a wordcount example", then the result in this phase will be following –

This, 1
Is, 1
A, 1
Wordcount, 1
Example, 1

(iv) Shuffle – The result obtained from the mapping steps are further sorted and arranged in such a way that repeated words are put together as a group. But still the key and value appear to be the same, and the repeated word are listed as many time as they appear in the file. For example, if a word "Apple" appearing 3 times and "Mango" 2 times in the file, it will be arranged as –

Apple, 1
Apple, 1
Apple, 1
Mango, 1
Mango, 1

(v) Reduce – In this step, the sorted group of words from the shuffle step is further sorted in such a way that the repeated words appear only one time and the number of time repeated will be added to the value. In this phase, the example from the shuffle phase after sorting will looks like following –

Apple, 3
Mango, 2

If there are more words in the file, then in the same way rest of the repeated words are sorted and the value are added to the number of times they are repeated.

(vi) Results – Finally, the sorted words from the Reduce phase are put together into the output where users can read the results.

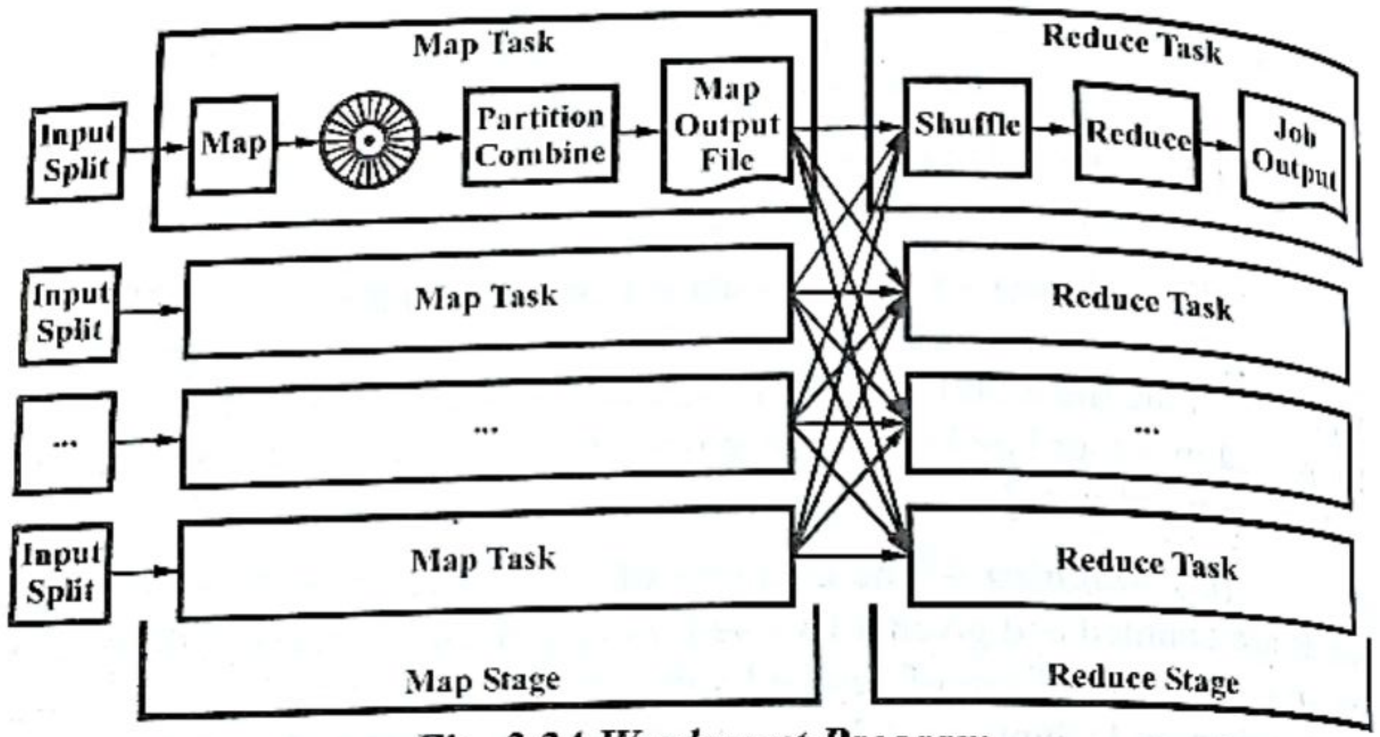

Fig. 2.24 Wordcount Program

This graphical representation of the entire process is shown in fig. 2.24. As shown in fig. 2.24, the wordcount process can be separated into three phases – Input Split, Map Task, and Reduce Task.

The example code that runs the MapReduce program shown in fig. 2.24 can be written in Scala programming language which is similar to Java : all of the steps involved in the above description is as follows –

```
package com.ims
import org.apache.spark.{SparkContext, SparkConf}
object WordCount {
def main(args : Array[String]) {
val conf = new SparkConf( ).setAppName("word count").setMaster
("local[2]")
val sc = new SparkContext(conf)
System.setProperty("hadoop.home.dir","C:\\winutil\\")
val input = sc.textFile("data/inputs/wordcount.txt")
val mapOutput = input.flatMap (line => line.split(" ")).map(x => (x, 1))
mapOutput.reduceByKey(_+_).collect.foreach(println)
mapOutput.reduceByKey(_+_, 1).saveAsTextFile("data\\output")
}
}
```

Q.42. Write the advantages of HDFS and MapReduce.

Ans. **Advantages of HDFS –**

(i) It has very high bandwidth to support map reduce jobs.
(ii) It is less expensive.
(iii) We can write the data once and read many times.

Advantages of MapReduce –

(i) It supports wide range of language, (ii) It is platform independent.

UNIT 3

HIVE AND PIG

INTRODUCTION TO HIVE, HIVE ARCHITECTURE, HIVE DATA TYPES, HIVE QUERY LANGUAGE

Q.1. What is Hive ? Write the advantages and disadvantages of Hive.

Ans. Hive was originally an internal Facebook project which eventually tenured into a full-blown Apache project, and it was created to simplify access to MapReduce (MR) by exposing a SQL-based language for data manipulation. Hive also maintains metadata in a metastore, which is stored in a relational database, as well as this metadata contains information about what tables exist, their columns, privileges, and more. Hive is an open source data warehousing solution built on top of Hadoop, and its particular strength is in offering ad-hoc querying of data, in contrast to the compilation requirement of Pig and cascading.

Hive is a natural starting point for more full-featured business intelligence systems which offer a user friendly interface for non-technical users.

Apache Hive supports analysis of large datasets stored in Hadoop's HDFS as well as easily compatible file systems like Amazon S3 (Simple Storage Service). Amazon S3 is a scalable, high-speed, low-cost, web-based service designed for online backup and archiving of data as well as application programs. Hive provides SQL-like language called HiveQL while maintaining full support for map/reduce, and to accelerate queries, it provides indexes, including bitmap indexes. Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, as well as analysis.

Advantages – The advantages of Hive are as follows –

(i) Perfectly fits low level interface requirement of Hadoop
(ii) Hive supports external tables and ODBC/JDBC
(iii) Having intelligence optimizer
(iv) Hive support of table-level partitioning to speed up the query times
(v) Metadata store is a big plus in the architecture that makes the lookup easy.

Disadvantages – The disadvantages of Hive are as follows –

(i) It does not support processing of unstructured data

(ii) The complicated jobs cannot be performed using Hive

(iii) The output of one job cannot to be used to query as input for another jobs.

Q.2. Explain installation and running Hive.

Ans. Hadoop must be installed on our system before installing Hive. Let us verify the Hadoop installation using the following command –

$hadoop version//It will display hadoop version information otherwise first we have to install hadoop.

Download Hive and rename its folder to/usr/local/hive

urp@localhost$ cd/home/user/Download

urp@localhost$ mv apache-hive-0.14.0-bin/usr/local/hive

#exit

You can set up the Hive environment by appending the following lines to ~/.bashrc file –

export HADOOP_HOME=/usr/local/hadoop

export HIVE_HOME=/usr/local/hive

export PATH=$PATH:$HIVE_HOME/bin

export PATH=$PATH:$HADOOP_HOME/bin

After installing Hive execute

$ cd $HIVE_HOME

$ bin/hive

Wordcount example for hive

Hive > select word, count(1) as count from (SELECT explode(split (sentence, ' ')) AS word FROM texttable) temp Table group by word.

Q.3. Explain architecture of Hive and its component. Also describe features of Hive.

Or

Explain the architecture of Hive and how it is useful for data analysis.

[R.G.P.V., May 2019, (VIII-Sem.)]

Ans. The architecture of Hive is shown in fig. 3.1 –

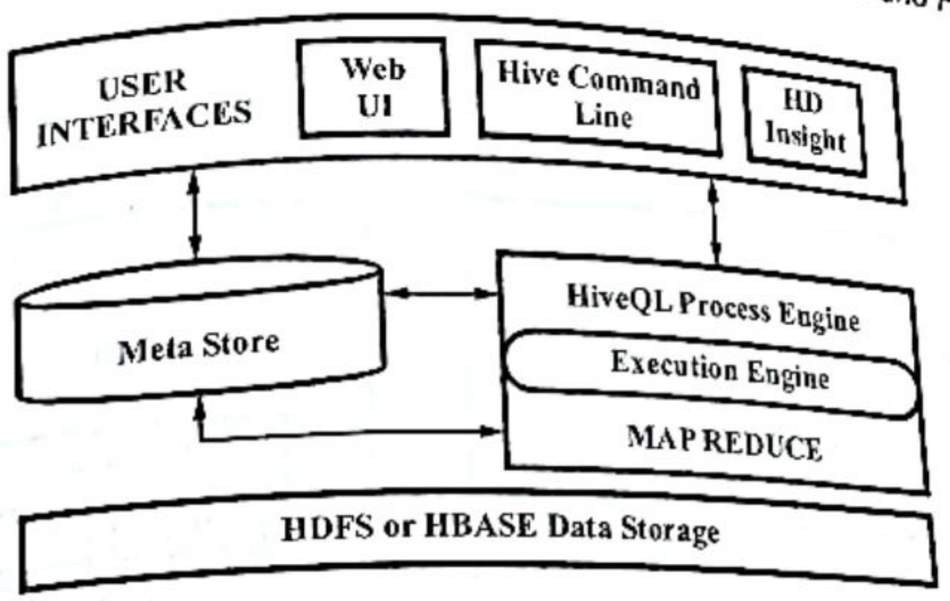

Fig. 3.1 The Architecture of Hive

This component diagram contains different units. Its components are discussed below –

(i) User Interface – Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

(ii) Meta Store – Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

(iii) HiveQL Process Engine – HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

(iv) Execution Engine – The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavour of MapReduce.

(v) HDFS or HBASE – Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Features of Hive –

(i) It stores schema in a database and processed data into HDFS.

(ii) It is designed for OLAP.

(iii) It provides SQL type language for querying called HiveQL or HQL.

(iv) It is familiar, fast, and extensible.

Q.4. Explain the working process of Hive.

Ans. The working process of Hive is shown in fig. 3.2.

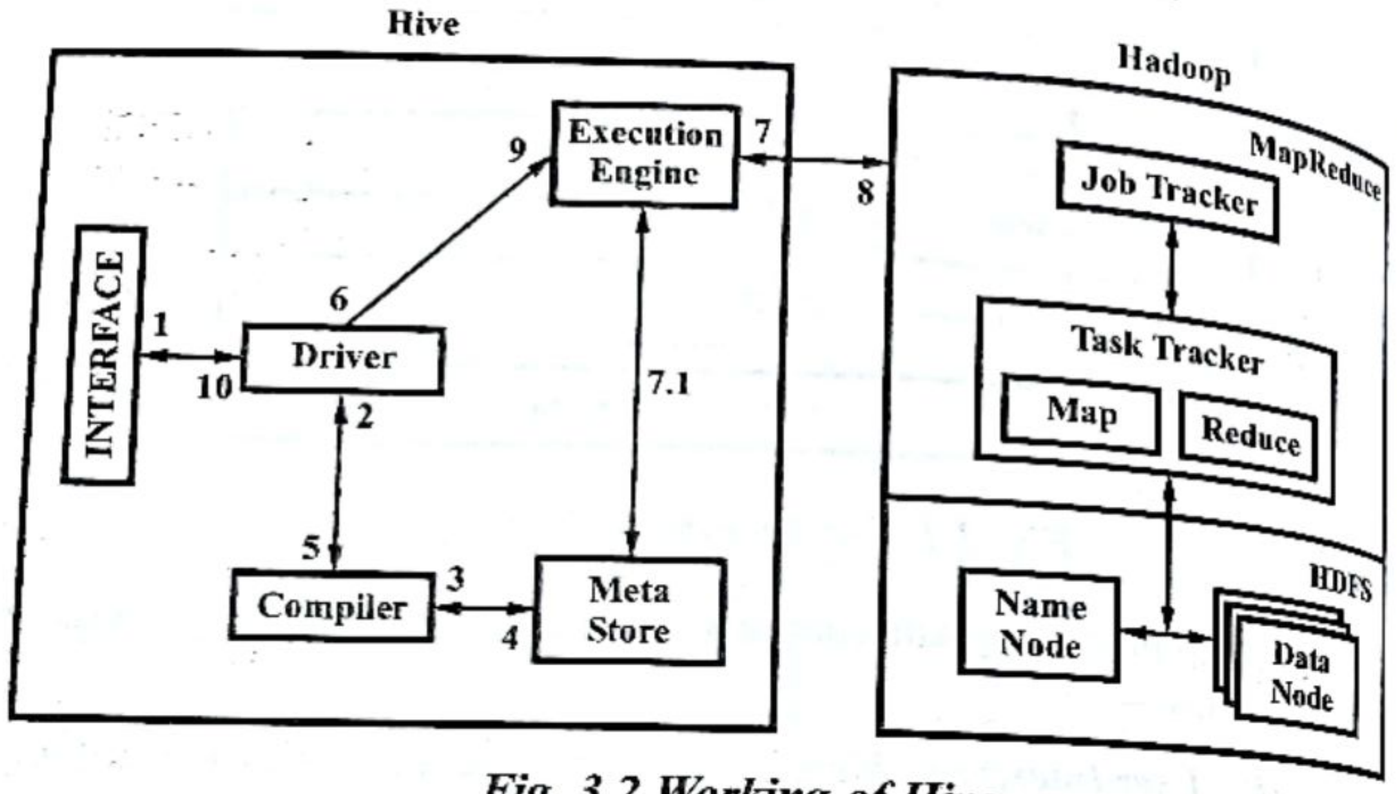

Fig. 3.2 Working of Hive

How Hive interacts with Hadoop framework is discussed below –

Step-1 Execute Query – The Hive interface such as Command Line or Web UI send query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.

Step-2 Get Plan – The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.

Step-3 Get Metadata – The compiler sends metadata request to Metastore (any database).

Step-4 Send Metadata – Metastore sends metadata as a response to the compiler.

Step-5 Send Plan – The compiler checks the requirement and resends the plan to the driver. At this point, the parsing and compiling of a query is complete.

Step-6 Execute Plan – The driver sends the execute plan to the execution engine.

Step-7 Execute Job – Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.

Step-7.1 Metadata Ops – Meanwhile, the execution engine can execute metatada operations with Metastore.

Step-8 Fetch Result – The execution engine receives the results from Data nodes.

Step-9 Send Results – The execution engine sends those resultant values to the driver.

Step-10 Send Results – The driver sends the results to Hive Interfaces.

Q.5. Write a short note on Hive file format.

Ans. Hive supports all the Hadoop file formats, plus Thrift encoding, as well as supporting pluggable SerDe (serializer/deserializer) classes to support custom formats.

There are several file formats supported by Hive.

(i) TEXTFILE – TEXTFILE is the easiest to use, but the least space efficient.

(ii) SEQUENCEFILE – SEQUENCEFILE format is more space efficient.

(iii) MAPFILE – MAPFILE adds an index to a SEQUENCE-FILE for faster retrieval of particular records.

Hive defaults to the following record and field delimiters, all of which are non-printable control characters and all of which can be customized.

Hive's default record and field delimiters are as follows –

Table 3.1

Delimiter	*Description*
\n	For text files, each line is a record, so the line feed character separates records.
^A("control"A)	Separates all fields (columns). Written using the octal code\001 when explicitly specified in CREATE TABLE statements.
^B	Separate the elements in an ARRAY or STRUCT, or the key-value pairs in a MAP. Written using the octal code \002 when explicitly specified in CREATE TABLE statements.
^C	Separate the key from the corresponding value in MAP key-value pairs. Written using the octal code \003 when explicitly specified in CREATE TABLE statements.

Q.6. Discuss about the Hive data types.

Ans. Hive support two types of data –

(i) Primitive data type

(ii) Collection data type.

(i) Primitive Data Type –

(a) TINYINT, SMALLINT, INT, BIGINT are four integer data types with only differences in their size.

(b) FLOAT and DOUBLE are two floating point data types. BOOLEAN is to store true or false.

(c) STRING is to store character strings. Note that, in hive we do not specify length for STRING like in other databases. It's more flexible and variable in length.

(d) TIMESTAMP can be an integer which is interpreted as seconds since UNIX epoch time. It may be a float where number after decimal is nanosecond. It may be string which is interpreted.

(e) According to the JDBC date string format i.e., YYYY-MM-DD hh:mm:ss.fffffffff. Time component is interpreted as UTC time.

(f) BINARY is used to place raw bytes which will not be interpreted by hive. It is suitable for binary data.

The primitive data types are shown in table 3.2.

Table 3.2 The Primitive Data Types

S.No.	*Data Type*	*Size*	*Example*
(i)	TINYINT	1 Byte	10, –10
(ii)	SMALLINT	2 Byte	10, –10
(iii)	INT	4 Byte	10, –10
(iv)	BIGINT	8 Byte	10, –10
(v)	FLOAT	Single precision float	10.8932
(vi)	DOUBLE	Double precision float	10.8932
(vii)	BOOLEAN	Boolean true or false	TRUE
(viii)	STRING	Sequence of characters	'Sample string'
(ix)	TIMESTAMP	Integer, float or string values	129357385 8929245.879395 '2013-01-01 12:00:00.123456789'
(x)	BINARY	Array of bytes	

(ii) Collection Data Types –

(a) STRUCT (b) MAP (c) ARRAY.

(a) STRUCT – Analogous to a C struct or an "object. "Fields can be accessed using the "dot" notation. For example, if a column name is of type STRUCT {first STRING; last STRING}, then the first name field can be referenced using name.first.

For e.g., struct('John', 'Doe')

(b) MAP– A collection of key-value tuples, where the fields are accessed using array notation (e.g., ['key']). For example, if a column name is of type MAP with key → value pairs 'first' → 'John' and 'last' → 'Doe', then the last name can be referenced using name['last'].

For e.g., map('first', 'John', 'last', 'Doe')

(c) ARRAY – Ordered sequences of the same type that are indexable using zero-based integers. For example, if a column name is of type ARRAY of strings with the value ['John', 'Doe'], then the second element can be referenced using name.

For e.g., array('John', 'Doe')

Q.7. What do you mean by HiveQL ?

Ans. HiveQL is the Hive query language. Hadoop is an open source framework for the distributed processing of large amounts of data across a cluster. It relies upon the MapReduce paradigm to reduce complex tasks into smaller parallel tasks that can be executed concurrently across multiple machines. However, writing MapReduce tasks on top of Hadoop for processing data is not for everyone since it requires learning a new framework and a new programming paradigm altogether. What is needed is an easy-to-use abstraction on top of Hadoop that allows people not familiar with it to use its capabilities as easily.

Hive aims to solve this problem by offering an SQL-like interface, called HiveQL, on top of Hadoop. Hive achieves this task by converting queries written in HiveQL into MapReduce tasks that are then run across the Hadoop cluster to fetch the desired results.

Hive is best suited for batch processing large amounts of data (such as in data warehousing) but is not ideally suitable as a routine transactional database because of its slow response times (it needs to fetch data from across cluster).

A common task for which Hive is used is the processing of logs of web servers. These logs have a regular structure and hence can be readily converted into a format that Hive can understand and process.

Hive query language (HiveQL) supports SQL features like CREATE tables, DROP tables, SELECT ..... FROM ..... WHERE clauses, Joins (inner, left outer, right outer and outer joins), Cartesian products, GROUP BY, SORT BY,

aggregations, union and many useful functions on primitive as well as complex data types. Metadata browsing features such as list databases, tables and so on are also provided. HiveQL does have limitations compared with traditional RDBMS SQL. HiveQL allows creation of new tables in accordance with partitions (Each table can have one or more partitions in Hive) as well as buckets (The data in partitions is further distributed as buckets) and allows insertion of data in single or multiple tables but does not allow deletion or updating of data.

Q.8. Explain the working with database in Hive.

Ans. (i) HiveQL : Data Definition – First open the hive console by typing –

```
$ hive
```

Once the hive console is opened, like

```
hive>
```

We need to run the query to create the table.

(ii) Create and Show Database – They are very useful for larger clusters with multiple teams and users, as a way of avoiding table name collisions. It's also common to use databases to organize production tables into logical groups. If we do not specify a database, the default database is used.

```
hive>CREATE DATABASE IF NOT EXISTS financials;
```

At any time, you can see the databases that already exist as follows –

```
hive>SHOW DATABASES;
```

output is

default

financials

```
hive>CREATE DATABASE human_resources;
```

```
hive>SHOW DATABASES;
```

output is

default

financials

human_resources

(iii) DESCRIBE Database – Shows the directory location for the database.

```
hive>DESCRIBE DATABASE financials;
```

output is

hdfs://master-server/user/hive/warehouse/financials.db

(a) USE Database – The USE command sets a database as your working database, analogous to changing working directories in a filesystem.

```
hive>USE financials;
```

(b) DROP Database – You can drop a database –

```
hive>DROP DATABASE IF EXISTS financials;
```

The IF EXISTS is optional and suppresses warnings if financials does not exist.

(c) Alter Database – We can set key-value pairs in the DBPROPERTIES associated with a database using the ALTER DATABASE command. No other metadata about the database can be changed, including its name and directory location.

```
hive>ALTER DATABASE financials SET DBPROPERTIES
('edited-by' = 'active steps');
```

(d) Create Tables – The CREATE TABLE statement follows SQL conventions, but Hive's version offers significant extensions to support a wide range of flexibility where the data files for tables are stored, the formats used, etc.

(1) Managed Tables –

(A) The tables we have created so far are called managed tables or sometimes called internal tables, because Hive controls the lifecycle of their data. As we have seen, Hive stores the data for these tables in a subdirectory under the directory defined by hive.metastore.warehouse.dir (e.g./user/hive/warehouse), by default.

(B) When we drop a managed table, Hive deletes the data in the table.

(C) Managed tables are less convenient for sharing with other tools.

(2) External Tables –

```
CREATE EXTERNAL TABLE IF NOT EXISTS stocks (
    exchange STRING,
    symbol STRING,
    ymd STRING,
    price_open FLOAT,
    price_high FLOAT,
    price_low FLOAT,
    price_close FLOAT,
    volume INT,
    price_adj_close FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/data/scocks/';
```

The EXTERNAL keyword tells Hive this table is external and the LOCATION ..... clause is required to tell Hive where it's located. Because it's external.

(3) Partitioned, Managed Tables – Partitioned tables help to organize data in a logical fashion, such as hierarchically.

Example – HR people often run queries with WHERE clauses that restrict the results to a particular country or to a particular first-level subdivision (e.g., state in the United States or province in Canada).

We have to use address.state to project the value inside the address. So, let's partition the data first by country and then by state –

```
CREATE TABLE IF NOT EXISTS mydb.employees (
name STRING,
salary FLOAT,
subordinates ARRAY<STRING>
deductions MAP<STRING, FLOAT>
address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
)
PARTITIONED BY (country STRING, state STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
COLLECTION ITEMS TERMINATED BY '\002'
MAP KEYS TERMINATED BY '\003'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Partitioning tables changes how Hive structures the data storage. If we create this table in the mydb database, there will still be an employees directory for the table –

```
LOAD DATA LOCAL INPATH '/path/to/employee.txt'
INTO TABLE employees
PARTITION (country = 'US', state = 'IL');
hdfs://master_server/user/hive/warehouse/mydb.db/employees
```

Once created, the partition keys (country and state, in this case) behave like regular columns.

```
hive>SHOW PARTITIONS employees;
```


output is

```
OK
country=US/state=IL
Time taken : 0.145 seconds
```

(e) Dropping Tables – The familiar DROP TABLE command from SQL is supported –

```
DROP TABLE IF EXISTS employees;
```

Q.9. Explain HiveQL data manipulation in brief.

Ans. **HiveQL Data Manipulation –**

(i) Loading Data into Managed Tables – Create stocks table –

```
CREATE EXTERNAL TABLE IF NOT EXISTS stocks (
    exchange STRING,
    symbol STRING,
    ymd STRING,
    price_open FLOAT,
    price_high FLOAT,
    price_low FLOAT,
    price_close FLOAT,
    volume INT,
    price_adj_close FLOAT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/data/stocks/';
Queries on Stock Data Set
```

Load the stocks –

```
LOAD DATA LOCAL INPATH '/path/to/employees.txt'
INTO TABLE stocks
PARTITION (exchange = 'NASDAQ', symbol = 'AAPL');
```

This command will first create the directory for the partition, if it doesn't already exist, then copy the data to it.

(ii) Inserting Data into Tables from Queries –

```
INSERT OVERWRITE TABLE employees PARTITION (country
= 'US', state = 'OR')
```

With OVERWRITE, any previous contents of the partition are replaced. If we drop the keyword OVERWRITE or replace it with INTO, Hive appends the data rather than replaces it.

(iii) Dynamic Partition Insert –

```
hive > INSERT OVERWRITE TABLE employees
>PARTITION (country, state)
>SELECT ....., a.count, a.state
>FROM staged_employees a;
```

Hive supports dynamic partition feature, where it can infer the partitions to create based on query partitions.

Q.10. Explain the HiveQL queries data.

Ans. **HiveQL Queries –**

(i) SELECT...FROM Clauses – SELECT is the projection operator in SQL. The FROM clause identifies from which table, view, or nested query we select records.

Create Employees –

```
CREATE EXTERNAL TABLE employees (
name STRING,
salary FLOAT,
subordinates ARRAY<STRING>,
deductions MAP<STRING, FLOAT>,
address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
COLLECTION ITEMS TERMINATED BY '\002'
MAP KEYS TERMINATED BY '\003'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/data/employees';
```


Load Data –

```
LOAD DATA LOCAL INPATH '/path/to/employees.txt'
INTO TABLE employees
PARTITION (country = 'US', state = 'IL');
```

Data in employee.txt is assumed as –

John Doe 100000.0 ["Mary Smith", "Todd Jones"] {"Federal Taxes" : 0.2, "State Taxes" : 0.05, "Insurance" : 0.3}

Mary Smith 80000.0 ["Bill King"] {"Federal Taxes" : 0.2, "State Taxes" : 0.05, "Insurance" : 0.1}

Todd Jones 70000.0 [ ] {"Federal Taxes" : 0.15, "State Taxes" : 0.03, "Insurance" : 0.1}

Bill King 60000.0 [ ] {"Federal Taxes" : 0.15, "State Taxes" : 0.03, "Insurance" : 0.1}

Boss Man 200000.0 ["John Doe", "Fred Finance"] {"Federal Taxes" : 0.3, "State Taxes" : 0.07, "Insurance" : 0.6}

Fred Finance 150000.0 ["Stacy Accountant"] {"Federal Taxes" : 0.3, "State Taxes" : 0.07, "Insurance" : 0.05}

Stacy Accountant 60000.0 [ ] {"Federal Taxes" : 0.15, "State Taxes" : 0.03, "Insurance" : 0.1}

Select Data –

```
hive > SELECT name, salary FROM employees;
```

output is

John Doe
100000.0
Mary Smith 80000.0
Todd Jones 70000.0
Bill King 60000.0

When you select columns that are one of the collection types, Hive uses JSON (Java-Script Object Notation) syntax for the output. First, let's select the subordinates, an ARRAY, where a comma-separated list surrounded with [...] is used.

```
hive>SELECT name, subordinates FROM employees;
```

output is

John Doe
["Mary Smith", "Todd Jones"]
Mary Smith ["Bill King"]
Todd Jones [ ]
Bill King [ ]

The deductions is a MAP, where the JSON representation for maps is used, namely a comma-separated list of key : value pairs, surrounded with {...}

```
hive>SELECT name, deductions FROM employees;
```

output is

John Doe
{"Federal Taxes" : 0.2, "State Taxes" : 0.05, "Insurance" : 0.1}
Mary Smith {"Federal Taxes" : 0.2, "State Taxes" : 0.05, "Insurance" : 0.1}
Todd Jones {"Federal Taxes" : 0.15, "State Taxes" : 0.03, "Insurance" : 0.1}
Bill King {"Federal Taxes" : 0.15, "State Taxes" : 0.03, "Insurance" : 0.1}

Finally, the address is a STRUCT, which is also written using the JSON map format :

```
hive > SELECT name, address FROM employees;
```

output is

John Doe
{"street" : "1 Michigan Ave.", "city" : "Chicago", "state" : "IL", "zip" : 60600}
Mary Smith {"street" : "100 Ontario St.", "city" : "Chicago", "state" : "IL", "zip" : 60601}
Todd Jones {"street" : "200 Chicago Ave.", "city" : "Oak Park", "state" : "IL", "zip" : 60700}
Bill King {"street" : "300 Obscure Dr.", "city" : "Obscuria", "state" : "IL", "zip" : 60100}

Q.11. Write a short note on sub queries.

Ans. The problem with this sampling approach is that it does not take into account the fact that the electricity usage for a given consumer on a given day is spread over 48 rows of the table. If a sample was taken using the above approach directly it is highly unlikely that there would be any complete days of usage for any of the consumers making any further useful analysis all but impossible.

However, by using this method to sample the geography table, which contains a single row only for each household instead of sampling the meter readings table directly it is possible to select a sample of unique households based on the anonid variable values. The anonid variable in the geography table represents the households and can be used to join the geography file data with the meter readings data.

In the query below we are selecting from the elec_all tables all of the rows associated with the anonids in bucket 3.


```
SELECT *
   FROM elec_all e
WHERE anon_id IN
      (SELECT anonid
         FROM geog_all
            TABLESAMPLE (BUCKET 3 OUT OF 150 ON rand (1)) s);
```

Q.12. Describe the table joins in querying.

Ans. In any relational database system, the ability to join tables together is a key querying requirement. Joins are used to combine the rows from two (or more) tables together to form a single table. A join between tables will only be possible if they have at least one column in common. The column doesn't need to have the same name in each table, and quite often they won't, but they do need to have a common usage. For example, in the geography table there is the anonid column and in the elec or gas tables there is the anon_id column in both cases they represent an anonymised household.

There are several different types of join possible as shown in table 3.3.

Table 3.3

Join Type	*What it Does*
Inner join	Matched rows in both tables are returned.
Left outer join	All row in the left hand table are returned along with the matches from the right hand table or NULLs if there is no match.
Right outer join	All row in the right hand table are returned along with the matches from the left hand table or NULLs if there is no match
Full outer join	All rows from both tables are returned, with NULLs where there are no matches.

By far the inner join is the most commonly used. The outer joins can be useful for exploring or discovering missing data. The cross join is rarely used and could create extremely large tables.

For Examples – Using the two small tables Animals and Animal_Eats as shown in table 3.4 and table 3.5.

Table 3.4

Id_A	*Name*
1	Elephant
2	Monkey
3	Cat
4	Dog
8	Goat
10	Pig
11	Mouse

Table 3.5

Id_E	*Name*
1	Hay
3	Fish
4	Meat
6	Goldfish food
7	Lettuce
8	Flowers
10	Anything

as **Inner Join** results is as shown in table 3.6.

Table 3.6

Id_A	*Name*	*Id_E*	*Eats*
1	Elephant	1	Hay
3	Cat	3	Fish
4	Dog	4	Meat
8	Goat	8	Flowers
10	Pig	10	Anything

a **Left Outer Join** results is as shown in table 3.7.

Table 3.7

Id_A	*Name*	*Id_E*	*Eats*
1	Elephant	1	Hay
2	Monkey	NULL	NULL
3	Cat	3	Fish
4	Dog	4	Meat
8	Goat	8	Flowers
10	Pig	10	Anything
11	Mouse	NULL	NULL

a **Right Outer Join** results is as shown in table 3.8.

Table 3.8

Id_A	*Name*	*Id_E*	*Eats*
1	Elephant	1	Hay
3	Cat	3	Fish
4	Dog	4	Meat
NULL	NULL	6	Goldfish food
NULL	NULL	7	Lettuce
8	Goat	8	Flowers
10	Pig	10	Anything

a ***Full Outer Join*** results is as shown in table 3.9.

Table 3.9

Id_A	*Name*	*Id_E*	*Eats*
1	Elephant	1	Hay
2	Monkey	NULL	NULL
3	Cat	3	Fish
4	Dog	4	Meat
NULL	NULL	6	Goldfish food
NULL	NULL	7	Lettuce
8	Goat	8	Flowers
10	Pig	10	Anything
11	Mouse	NULL	NULL

An example join statement –

```
SELECT e.anon_id, g.anon_id
    FROM distinct_elec_c AS e
    JOIN distinct_gas_C as g
    ON e.anon_id = g.anon_id;
```

Q.13. Define user defined functions in Hive.

Ans. There are multiple ways to extend Hive's functionality including writing custom user defined functions (UDF).

(i) Simple Functions **– Concat** can be used to add strings together

```
SELECT anonid,
    acorn_category,
    acorn_group,
    acorn_type,
    concat  (acorn_category,
             ",",
             acorn_group,
             ",",
             acorn_type)
        As acorn_code
    FROM geog_all;
```

substr can be used to extract a part of a string

```
SELECT  anon_id,
        advancedatetime,
        substr (advancedatetime, 1, 2) As day,
        substr (advancedatetime, 3, 3) As month,
        substr (advancedatetime, 6, 2) As year,
FROM elec_c;
```

examples of **length**, **instr** and **reverse**

```
SELECT   anonid,
         acorn_code,
         length (acorn_code),
         instr (acorn_code, ',') AS a_catpos,
         instr (reverse (acorn_code), "," ) AS reverse_a_type
FROM geog_all;
```

Where needed functions can be nested within each other.

cast and type conversions

```
SELECT   anonid,
         substr (acorn_code, 7, 2) AS ac_type_string,
         cast (substr (acorn_code, 7, 2) AS INT) AS ac_type_
         substr (acorn_code, 7, 2) + 1 AS ac_type_not_sure
FROM geog_all;
```

(ii) Aggregations – Aggregate functions are used to perform some k of mathematical or statistical calculation across a group of rows. The rows each group are determined by the different values in a specified column or colum A list of all of the available functions are given in the apache documentation.

```
SELECT   anon_id,
         count (eleckwh) As total_row_count,
         sum (eleckwh) As total_period_usage,
         min (eleckwh) As min_period_usage,
         avg (eleckwh) As avg_period_usage,
         max (eleckwh) As max_period_usage
    FROM elec_c
GROUP BY anon_id;
```

In the above example, the aggregation were performed over the sing column anon_id. It is possible to aggregate over multiple columns by specifyi them in both the select and the group by clause. The grouping will take pla based on the order of the columns listed in the group by clause. What is n allowed is specifying a non-aggregated column in the select clause which not mentioned in the group by clause.

```
SELECT anon_id,
       substr (advancedatetime, 6, 2) AS reading_year,
       count (eleckwh) AS total_row_count,
       sum (eleckwh) As total_period_usage,
       min (eleckwh) As min_period_usage,
       avg (eleckwh) As avg_period_usage,
       max (eleckwh) As max_period_usage
    FROM elec_c
GROUP BY anon_id, substr (advancedatetime, 6, 2);
```


Unfortunately, the group by clause will not accept alias'.

```
SELECT anon_id,
       substr (advancedatetime, 6, 2) AS reading_year,
       count (eleckwh) AS total_row_count,
       sum (eleckwh) AS total_period_usage,
       min (eleckwh) AS min_period_usage,
       avg (eleckwh) AS avg_period_usage,
       max (eleckwh) AS max_period_usage
    FROM elec_c
GROUP BY anon_id, substr (advancedatetime, 6, 2)
ORDER BY anon_id, reading_year;
```

But the Order by clause does.

The distinct keyword provides a set of unique combination of column values within a table without any kind of aggregation.

```
SELECT DISTINCT, eprofileclass, fueltypes
FROM geog_all;
```

(iii) Date Functions – In the elec_c and gas_c tables, the advancedate time column, although it contains a timestamp type information, it is defined as a string type. For much of the time this can be quite convenient, however there will be times when we really do need to be able to treat the column as a Timestamp. Perhaps the most obvious example is when you need to sort rows based on the advancedatetime column.

Hive provides a variety of date related functions to allow you to convert strings into Timestamp and to additionally extract parts of the Timestamp.

unix_timestamp returns the current data and time – as an integer!

from_unixtime takes an integer and converts in into a recognisable timestamp string

```
SELECT unix_timestamp ( ) AS currenttime
    FROM sample_07
LIMIT 1;
SELECT from_unixtime (unix_timestamp ( )) AS currenttime
    FROM sample_07
LIMIT 1;
```

There are various date part functions which will extract the relevant parts from a Timestamp string

```
SELECT anon_id,
    from_unixtime (UNIX_TIMESTAMP (reading_date, 'ddMMyy')
      AS proper_date,
    year (from_unixtime (UNIX_TIMESTAMP (reading_date,
'ddMMyy')))
      AS full year,
    month (from_unixtime (UNIX_TIMESTAMP (reading_date,
'ddMMyy')))
      AS full_month,
    day (from_unixtime (UNIX_TIMESTAMP (reading_date,
'ddMMyy')))
      AS full_day,
    last_day (from_unixtime (UNIX_TIMESTAMP (reading_date,
'ddMMyy')))
      AS last_day_of_month,
    date_add ((from_unixtime(UNIX_TIMESTAMP
(reading_date, 'ddMMyy'))), 10)
      AS added_days
  FROM elec_days_c
  ORDER BY proper_date;
```

INTRODUCTION TO PIG, ANATOMY OF PIG, PIG ON HADOOP, USE CASE FOR PIG, ETL PROCESSING

Q.14. What is Pig ?

Ans. Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization. An application that creates map-reduce jobs based on a language called Pig Latin which is workflow driven. It was originally created at Yahoo! (company). Apache Pig is good for structured data too, but its advantage is the ability to work with BAGs of data (all rows that are grouped on a key), it is simpler to implement things like –

(i) Get top N elements for each group

(ii) Calculate total per each group and than put that total against each row in the group

(iii) Use Bloom filters for JOIN optimisations

(iv) Multiquery support (it is when PIG tries to minimise the number on MapReduce Jobs by doing more stuff in a single job)

(v) Hive is better suited for ad-hoc queries, but its main advantage is that it has engine that stores and partitions data. But its tables can be read from Pig or Standard MapReduce.

(vi) One more thing, Hive and Pig are not well suited to work with hierarchical data.

Pig integration with streaming also makes it easy for researchers to take a perl or python script they have already debugged on a small data set and run it against a huge data set. PIG is best for semi structured data, for programming, used as procedural language, does not support partitions, do not have dedicated metadata of database.

Q.15. Explain the architecture of Apache Pig and its components.

Ans. The language used to analyze data in Hadoop using Pig is known as Pig Latin. It is a high-level data processing language which provides a rich set of data types and operators to perform various operations on the data.

To perform a particular task programmers using Pig, programmers need to write a Pig Script using the Pig Latin language, and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded). After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output. Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy. The architecture of Apache Pig is shown in fig. 3.3.

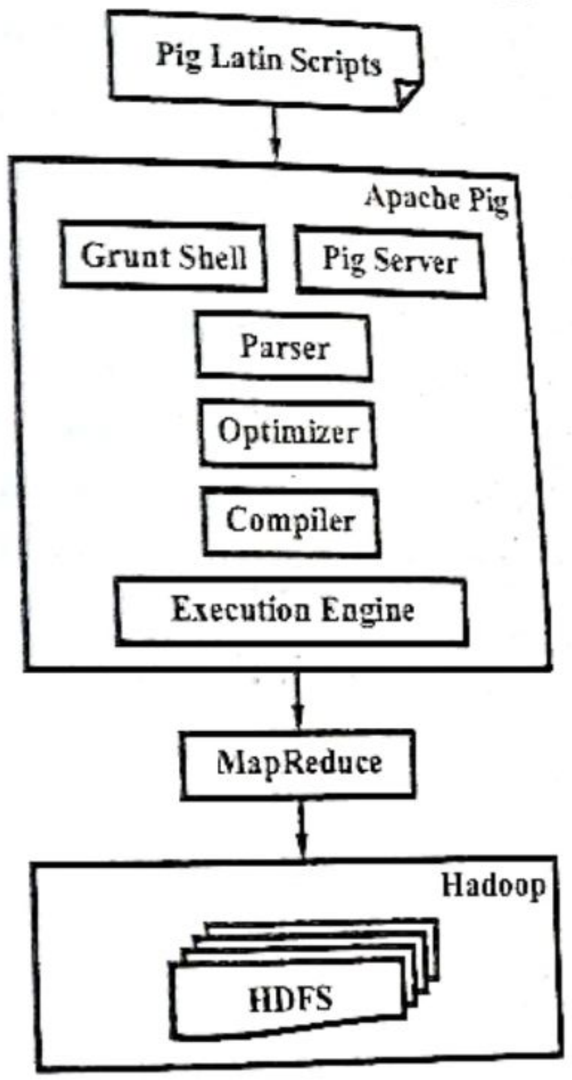

Fig. 3.3 Apache Pig Architecture

As shown in the fig. 3.3, there are various components in the Apache Pig framework. Let us take a look at the major components.

(i) Parser – Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.

In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.

(ii) Optimizer – The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.

(iii) Compiler – The compiler compiles the optimized logical plan into a series of MapReduce jobs.

(iv) Execution Engine – Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

(v) Grunt Shell – Grunt is Pig's interactive shell. It enables users to enter Pig Latin interactively and provides a shell for user to interact with HDFS.

In other words, Grunt is command interpreter we can type Pig Latin on the Grunt command line and Grunt will execute the command.

Q.16. Why we need Apache Pig ? Explain.

Ans. Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers –

(i) Using Pig Latin, programmers can perform MapReduce tasks easily without having to type complex codes in Java.

(ii) Apache Pig uses multi-query approach, thereby reducing the length of codes. For example, an operation that would require to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.

(iii) Pig Latin in SQL-like language and it is easy to learn Apache Pig when we are familiar with SQL.

(iv) Apache pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types such as tuples, bags, and maps that are missing from MapReduce.

Q.17. What are the applications and features of Apache Pig ?

Ans. **Applications of Apache Pig** – Apache Pig is generally used by data scientists for performing tasks involving ad-hoc processing and quick prototyping. Apache Pig is used as –

(i) To process huge data sources such as web logs.

(ii) To perform data processing for search platforms.

(iii) To process time sensitive data loads.

Features of Apache Pig – Apache Pig comes with the following features –

(i) Rich Set of Operators – It provides many operators to perform operations like join, sort, filer, etc.

(ii) Ease of Programming – Pig Latin is similar to SQL and it is easy to write a pig script if we are good at SQL.

(iii) Optimization Opportunities – The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.

(iv) Extensibility – Using the existing operators, users can develop their own functions to read, process, and write data.

(v) UDF's – Pig provides the facility to create user-defined functions in other programming languages such as Java and invoke or embed them in Pig Scripts.

(vi) Handles all Kinds of Data – Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Q.18. What are the major differences between Apache Pig and MapReduce ?

Ans. The major differences between Apache Pig and MapReduce are as follows –

S.No.	*Apache Pig*	*MapReduce*
(i)	Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
(ii)	It is a high level language.	MapReduce is low level and rigid.
(iii)	Performing a join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a join operation between datasets.
(iv)	Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce.
(v)	Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
(vi)	There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

Q.19. What do you mean by Pig Latin ?

Ans. Pig is a high-level platform for creating MapReduce Programs us with Hadoop. The language for this platform is called Pig Latin. Pig La abstracts the programming from the Java MapReduce idiom into a notati which makes MapReduce programming high level. Similar to that of SQL f RDBMS systems. Pig Latin can be extended using UDF (user defined functions which the user can write in java, Python, Javascript, Ruby or Groovy a then call directly from the language.

Q.20. What are the advantages of Pig as compared to SQL ?

Ans. In compare to SQL, Pig have following advantages –

(i) Uses lazy evaluation

(ii) Uses extract, transform, load (ETL)

(iii) Able to store data at any point during a pipeline

(iv) Declares execution plans

(v) Supports pipeline split, thus allowing workflows to proceed alo DAGs instead of stricty sequential pipelines.

Q.21. What are the major differences between Pig and SQL databas

Ans. The major differences between Apache Pig and SQL are as follow

S.No.	*Pig*	*SQL*
(i)	Pig Latin is a procedural language.	SQL is a declarative language.
(ii)	In Apache Pig, schema is optional. We can store data without designing a schema (values are stored as $01, $02, etc.)	Schema is mandatory in SQL.
(iii)	The data model in Apache Pig is nested relational.	The data model used in SQL is flat relational.
(iv)	Apache Pig provides limited opportunity for query optimization.	There is more opportunity for query optimization in SQL.

Q.22. What are the important uses of Pig.

Ans. Some of the important uses of Pig are as follows –

(i) Pig is a powerful tool for querying data in a Hadoop cluster. It is so powerful that Yahoo estimates that between 40% and 60% of its Hadoop workloads are generated from Pig Latin scripts.

(ii) Pig is also used at Twitter (processing logs, mining tweet data) at AOL and MapQuest (for analytics and batch data processing); and at LinkedIn, where Pig is used to discover people you might know.

(iii) With continually increasing population, crimes and crime rate analyzing related data is a huge issue for governments to make strategic decisions so as to maintain law and order. The benefit of using Pig for analysis is that fewer lines of code have to be written which reduces overall development and testing time.

(iv) Using Pig script a large scale data processing system for analyzing web log data through Map Reduce programming in Hadoop framework is efficient.

(v) Pig is used to evaluate the performance of a commercial RDBMS and Hadoop in astronomy simulation analysis tasks.

Q.23. What do you mean by ETL big data with Apache Hadoop ? Describe various stages in ETL.

Ans. The challenge of extracting value from big data is similar in many ways to the age-old problem of distilling business intelligence from transactional data. At the heart of this challenge is the process used to extract data from multiple sources, transform it to fit your analytical needs, and load it into a data warehouse for subsequent analysis, a process known as "Extract, Transform & Load" (ETL). The nature of big data requires that the infrastructure for this process can scale cost-effectively. Apache Hadoop has emerged as the de facto standard for managing big data.

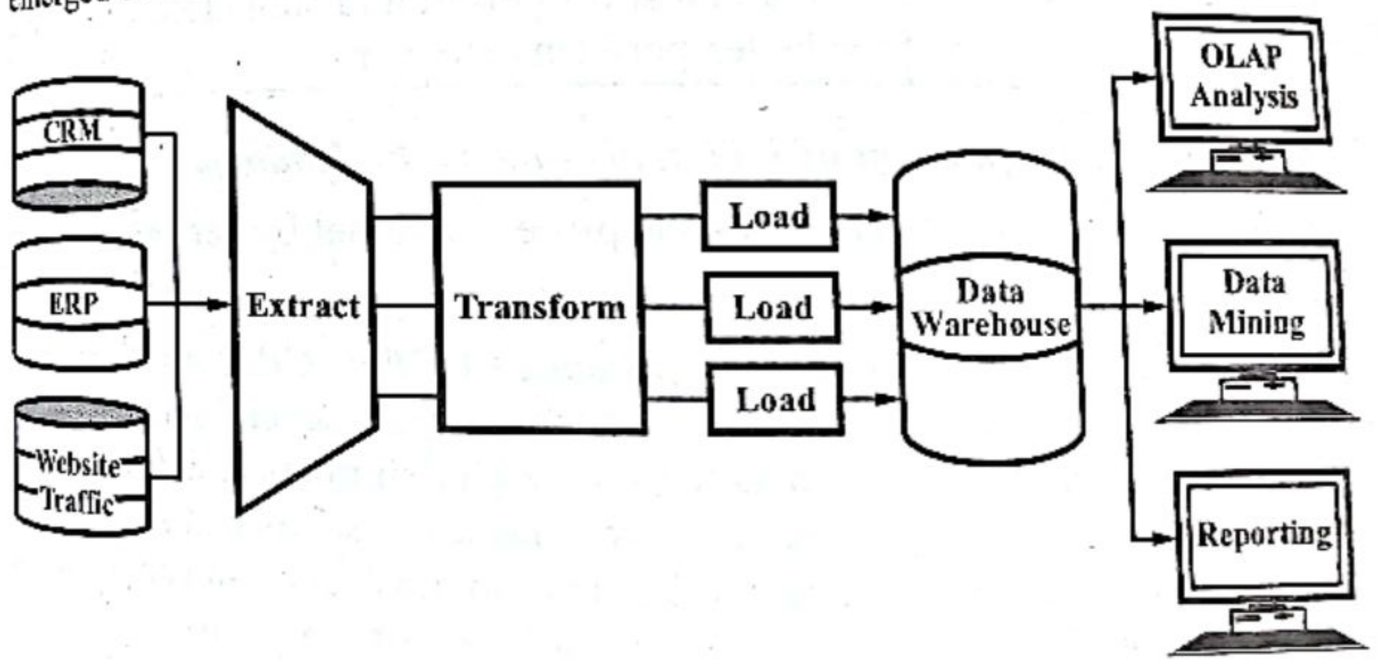

Fig. 3.4 ETL Process

A traditional ETL process extracts data from multiple sources, then cleanses, formats, and loads it into a data warehouse for analysis. When the source data sets are large, fast, and unstructured, traditional ETL can become the bottleneck, because it is too complex to develop, too expensive to operate, and takes too long to execute.

By most accounts, 80 percent of the development effort in a big data project goes into data integration and only 20 percent goes toward data analysis. Furthermore, a traditional EDW platform can cost upwards of USD 60 K per terabyte. Analyzing one petabyte – the amount of data Google processes in 1

hour would cost USD 60 M. Clearly "more of the same" is not a long-t strategy that any CIO can afford.

Various Stages in ETL –

Table 3.10 Stages in ETL Approach

S.No.	*Stage*	*Description of Stage*
(i)	Extraction	Extraction is the initial segment of an ETL proce dure which includes extracting the informatio from the source framework. In separating informa tion accurately sets the phase for the achievemen of ensuing procedures. The majority of the unic takings are to consolidate information with a fe distinctive source frameworks.
(ii)	Transformation	In this phase rules or principles are applied to th extracted information loads into the end target. S any information does not require any chang whatsoever such information is known a "immediate move" or go through information.
(iii)	Loading	The loading stage loads huge volume of data loaded in a short period and should be optimize for better performance.

Q.24. Explain function of ETL tools in Apache Hadoop.

Ans. ETL tools move data from one place to another by performing three functions –

(i) Extract Data from sources such as ERP or CRM Applications – During the extract step, you may need to collect data from several source systems and in multiple file formats, such as flat files with delimiters (CSV) and XML files. You may also need to collect data from lagacy systems that store data in arcane formats no one else uses anymore. This sounds easy, but can in fact be one of the main obstacles in getting an ETL solution of the ground.

(ii) Transform that Data into a common Format that Fits other Data in the Warehouse – The transform step may include multiple data manipulations, such as moving, splitting, translating, merging, sorting, pivoting and more. For example, a customer name might be split into first and last names, or dates might be changed to the standard ISO format (e.g., from 07-24-13 to 2013-07-24). Often this step also involves validating the data against data quality rules.

(iii) Load the Data into the Data Warehouse for Analysis – This step can be done in batch processes or row by row, more or less in real time.

Q.25. Explain offload ETL with Hadoop.

Ans. Once the data is in Hadoop (on a Hadoop-compatible file system), we can perform the traditional ETL tasks of cleansing, normalizing, aligning, and aggregating data for EDW by employing the massive scalability of MapReduce.

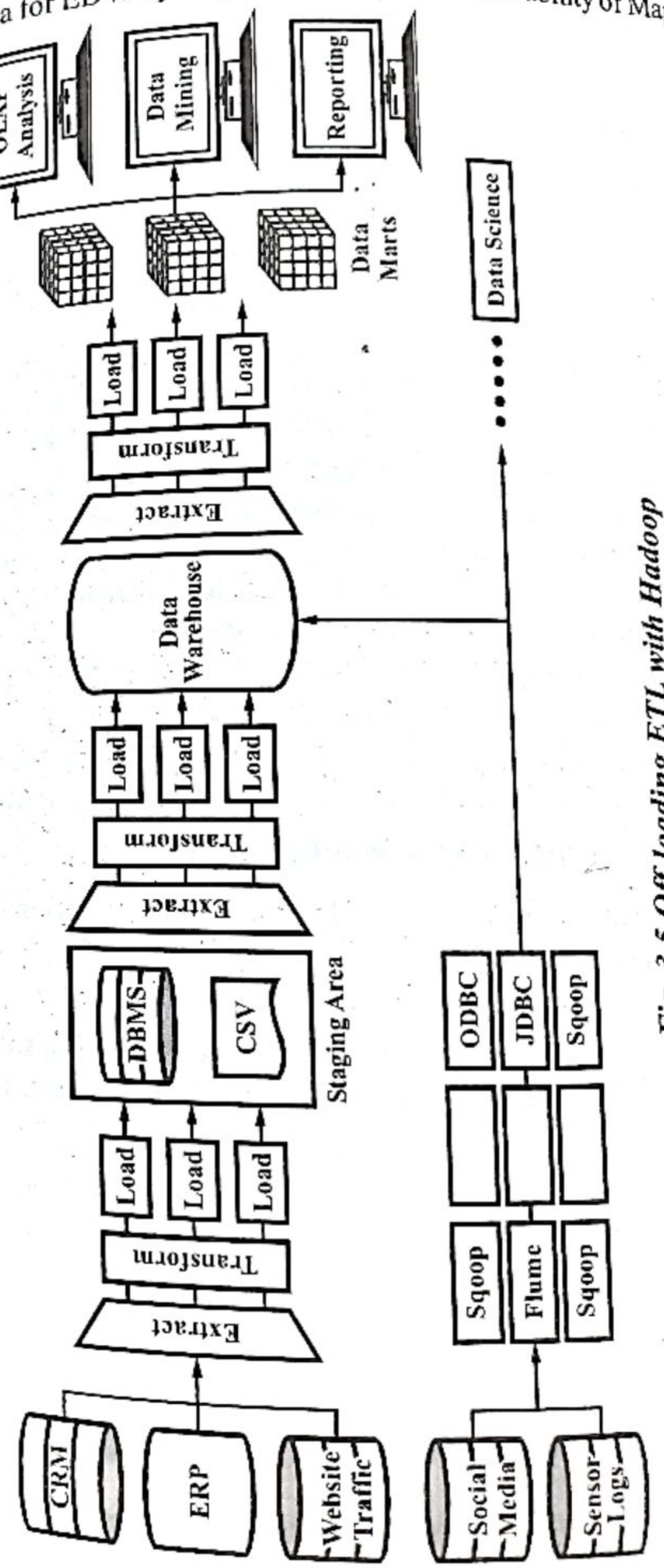

Fig. 3.5 Off-loading ETL with Hadoop

Using Hadoop allows to avoid the transformation bottleneck in traditional ETLT by off-loading the ingestion, transformation, and integration of unstructured data into data warehouse as shown in fig. 3.5. Because Hadoop enables to embrace more data types than ever before, it enriches data warehouse in ways that would otherwise be infeasible or prohibitive. Because of its scalable performance, the ETLT jobs can be accelerated significantly. Moreover, because data stored in Hadoop can persist over a much longer duration, we can provide more granular, detailed data through EDW for high-fidelity analysis.

Using Hadoop in this way, the organization gains an additional ability to store and access data that they "might" need, data that may never be loaded into the data warehouse. For example, data scientists might want to use the large amounts of source data from social media, web logs, or third-party stores (from curators, such as data.gov) stored on Hadoop to enhance new analytic models that drive research and discovery. They can store this data cost effectively in Hadoop, and retrieve it as needed (using Hive or other analytic tools native to the platform), without affecting the EDW environment.

Regardless of whether using the ETL, ELT or ETLT approach to data warehousing, the operational cost of overall BI/DW solution can be reduced by off-loading common transformation pipelines to Apache Hadoop and using MapReduce on HDFS to provide a scalable, fault-tolerant platform for processing large amounts of heterogeneous data.

DATA TYPES IN PIG, RUNNING PIG, EXECUTION MODEL OF PIG, OPERATORS, FUNCTIONS, DATA TYPES OF PIG

Q.26. Describe user defined data types in Pig.

Ans. Pig's data types can be divided into two categories –

(i) Scalar types

(ii) Complex types.

(i) Scalar Types – It contains a single value. Pig's scalar types are simple data types that appear in most programming languages. These include -

(a) int – An integer store a four-byte signed integer

(b) long – A long integer store an eight-byte signed integer

(c) float – A floating-point number uses four bytes to store their value.

(d) double – A double-precision floating-point number use eight bytes to store their value.

(e) chararray – A string or character array are expressed as string literals with single quotes.

(f) bytearray – A blob or array of bytes.

(ii) Complex Types – Pig has several complex data types such as maps, tuples, and bags. All of these types can contain data of any type, including other complex types. So it is possible to have a map where the value field is a bag, which contains a tuple where one of the fields is a map.

(a) Map – A map in Pig is a chararray to data element mapping, where that element can be any Pig type, including a complex type. The chararray is called a key and is used as an index to find the element, referred to as the value.

(b) Tuple – A tuple is a fixed-length, ordered collection of Pig data elements. Tuples are divided into fields, with each field containing one data element. These elements can be of any type – they do not all need to be the same type. A tuple is analogous to a row in SQL, with the fields being SQL columns.

Syntax – (Field [, field ....])

(c) Bag – A bag is an unordered collection of tuples. Because it has no order, it is not possible to reference tuples in a bag by position. Like tuples, a bag can, but is not required to, have a schema associated with it. In the case of a bag, the schema describes all tuples within the bag.

Syntax – {tuple[, type .....]}

(d) Nulls – Pig includes the concept of a data element being null. Data of any type can be null. It is important to understand that in Pig the concept of null is the same as in SQL, whic is completely different from the concept of null in C, Java, Python, etc. In Pig a null data element means the value is unknown.

(e) Casts – Indicates convert one type of content to any other type.

The data types scalar and complex table are as follows –

Table 3.11 Data Types in Pig

Scalar Types	*Description*	*Example*
int	Signed 32-bit integer	10
long	Signed 64-bit integer	Data : 10L or 101 Display : 10L
float	32-bit floating point	Data : 10.5F or 10.5f or 10.5e2f or 10.5E2F Display : 10.5F or 1050.0F
double	64-bit floating point	Data : 10.5 or 10.5e2 or 10.5E2 Display : 10.5 or 1050.0

chararray	Character Aarray (string) in Unicode UTF-8 format	hello world
bytearray	Byte array (blob)	
boolean	boolean	true/false (case insensitive)
Complex Types –		
tuple	An ordered set of fields	(19, 2)
bag	An collection of tuples	{(19, 2), (18, 1)}
map	A set of key value pairs	[open#apache]

Q.27. Explain in detail installation and running of Pig.

Ans. **Pig Installation** – Linux users need the following –

(i) Hadoop 0.20.2

(ii) Java 1.6 (set JAVA_HOME to the root of Java installation)

(iii) Ant 1.7 (optional, for builds)

(iv) JUnit 4.5 (optional, for unit tests).

To get a pig distribution, download a recent stable release from one of the Apache Download Mirrors.

Unpack the downloaded Pig distribution. The Pig script is located in the bin directory.

Add/pig-n.n.n/bin to your path. Use export (bash, sh, ksh) or setenv (tcsh, csh). For example –

urp@localhost$ export PATH=/usr/local/pig-0.14.0/bin:$PATH

Try the following command, to get a list of Pig commands

urp@localhost$ pig-help

Try the following command, to start the Grunt shell

urp@localhost$pig

(i) Run Modes – Pig has two run or execution modes i.e. local mode and map reduce mode.

(a) Local Mode – In this mode, pig run in a single JUM and makes use of local file system. This mode is suitable only for analysis of small data set using Pig.

(b) Map Reduce Mode – In this mode, queries written in Pig Latin are translated into MapReduce jobs and are run on a Hadoop cluster. MapReduce mode with fully distributed cluster is useful of running Pig on large dataset.

(ii) Grunt Shell – Grunt is a command interpreter. We can type Pig Latin on the Grunt command line and Grunt will execute the command.

Local Mode –

urp@localhost$ pig-x local

MapReduce Mode –

urp@localhost$ pig

or

urp@localhost$ pig-x mapreduce

For either mode, the Grunt shell is invoked and we can enter commands at the prompt. The results are displayed to terminal screen (if DUMP is used) or to a file (if STORE is used).

Wordcount Example for Pig –

```
grunt>a=LOAD '/piginput/pigexample.txt'USING PigStorage as (word:chararray);
grunt>dump a;
grunt>words=FOREACH a GENERATE FLATTEN (TOKENIZE(word));
grunt>dump word;
grunts>grouped=GROUP words by $0;
grunt>dump grouped;
grunt>word_counts=FOREACH grouped GENERATE group, COUNT(words);
grunt>store word_counts into '/pigoutput/word_count1' USING PigStorage;
```

Q.28. Explain data processing operators in Pig Latin with example.

Ans. Pig Latin has a very rich syntax. It supports operators for the following operations –

(i) Loading and storing of data

(ii) Streaming data

(iii) Filtering data

(iv) Grouping and joining data

(v) Sorting data

(vi) Combining and splitting data.

Pig Latin also supports a wide variety of types, expressions, functions, diagnostic operators, macros, and file system commands.

(i) DUMP – Dump directs the output of your script to your screen

Syntax – dump out.txt;

(ii) LOAD – Loads data from the file system.

Syntax – LOAD 'data' [USING function] [AS schema];

'data' is the name of the file or directory, in single quotes. USING, AS are keywords. If the USING clause is omitted, the default load function PigStorage is used. Schema – A schema using the AS keyword, enclosed in parentheses.

Usage – Use the LOAD operator to load data from the file system.

For Examples – Suppose we have a data file called myfile.txt. The fields are tab-delimited. The records are newline-separated.

123

421

834

In this example the default load function, PigStorage, loads data from myfile.txt to form relation A. The two LOAD statements are equivalent. Note that, because no schema is specified, the fields are not named and all fields default to type byte array.

```
A = LOAD 'myfile.txt' USING PigStorage('\t');
DUMP A;
```

Output –

(1, 2, 3)
(4, 2, 1)
(8, 3, 4)

Sample Code – The examples are based on these Pig commands, which extract all user IDs from the /etc/passwd file.

```
A = load 'passwd' using PigStorage(':');
B = foreach A generate $0 as id;
dump B;
store B into 'id.txt';
```


(iii) Storing Data – Stores or saves results to the file system.

Syntax – STORE alias INTO 'directory' [USING function];

For Examples – In this example data is stored using PigStorage and the asterisk character (*) as the field delimiter.

```
A = LOAD 'data';
DUMP A;
```

Output –

(1, 2, 3)
(4, 2, 1)
(8, 3, 4)
(4, 3, 3)
(7, 2, 5)
(8, 4, 3)

```
STORE A INTO 'myoutput' USING PigStorage ('*');
CAT myoutput;
```

Output –

1*2*3
4*2*1
8*3*4
4*3*3
7*2*5
8*4*3

(iv) Streaming Data – Sends data to an external script or program.

(v) Grouping and Joining Data –

(a) Group – Groups the data in one or more relations.

(b) JOIN (Inner) – Performs an inner join of two or more relations based on common field values.

(c) JOIN (Outer) – Performs an outer join of two relations based on common fields values.

Example – Suppose we have relations A and B.

```
A = LOAD 'data1' AS (a1 : int, a2 : int, a3 : int);
DUMP A;
(1, 2, 3)
(4, 2, 1)
(8, 3, 4)
(4, 3, 3)
(7, 2, 5)
(8, 4, 3)
B = LOAD 'data2' AS (b1 : int, b2 : int);
DUMP B;
(2, 4)
(8, 9)
(1, 3)
(2, 7)
(2, 9)
(4, 6)
(4, 9)
```

In this example relations A and B are joined by their first fields.

```
X = JOIN A BY a1, B BY b1;
DUMP X;
(1, 2, 3, 1, 3)
(4, 2, 1, 4, 6)
(4, 3, 3, 4, 6)
(4, 2, 1, 4, 9)
(4, 3, 3, 4, 9)
(8, 3, 4, 8, 9)
(8, 4, 3, 8, 9)
```

Q.29. What are the advantages and disadvantages of Pig ?

Ans. The advantages and disadvantages of Pig are as follows –

Advantages –

(i) It decreases the Duplication of data.

(ii) It reduces the number of lines of code and save the development time.

(iii) The user defined functions can be easily programmed for read and write operations.

(iv) It supports nested data models.

(v) The programmer who knows SQL language can easily able to learn and write Pig scripts.

Disadvantages –

(i) It doesn't provide JDBC and ODBC connectivity.

(ii) There is no dedicated metadata data base.

(iii) It doesn't offer web interface.

Q.30. Give the functions used in Pig.

Ans. Functions used in Pig are as follows –

(i) Eval – A function that takes one or more expression and returns another expressions.

(ii) Filter – A special type of eval function that returns a logical Boolean result.

(iii) Load – A function that specifies how to load data into a relation from external storage.

(iv) Store – A function which specifies how to save contents of a relation to external storage.

Q.31. What types of commands are used in Pig program ?

Ans. The commands used in Pig program are as follows –

Table 3.12 Pig Commands

Command	*Description*
Load	Read data from the file system
Store	Write data to the file system
Dump	Write output to stdout
Foreach Generate	Apply expression to each record and generate one or more records
Filter	Apply predicate to each record and remove records where false
Group/ Cogroup	Collect records with the same key from one or more inputs

Join	Join two or more inputs based on a key
Order	Sort records based on a key
Distinct	Remove duplicate records
Union	Merge two datasets
Limit	Limit the number of records
Split	Split data into two or more sets, based on filter conditions
Cross	Creates the cross product of two or more relations

UNIT 4

NoSQL

INTRODUCTION TO NoSQL, NoSQL BUSINESS DRIVERS, NoSQL DATA ARCHITECTURAL PATTERNS, VARIATIONS OF NoSQL ARCHITECTURAL PATTERNS, USING NoSQL TO MANAGE BIG DATA

Q.1. What is NoSQL ? Explain.

Ans. NoSQL, which means "Not only SQL" is a generic term of database management systems (DBMS), which provide a mechanism for storing and retrieving data different from that of relational DBMS, and hence, traditional SQL queries over the data cannot be applied to them. A basic feature of most NoSQL datastores is the "shared nothing" horizontal scaling, which allows them to execute a huge number of read/write operations per second. Non-relational databases are generally known for their schema-less data models, improved performance and scalability.

Conventional relational database system uses two-dimensional table for data creation, with properties like transactions, complex SQL queries, and multi-table related query. However, multi-table queries are not effective for huge data queries. Scalability in relational databases requires powerful servers that are both expensive and difficult to handle.

NoSQL provides the flexibility to store entire data in terms of documents instead of conventional method of table-row-column. NoSQL is extensively useful when we need to access and analyze huge amounts of unstructured data or data that is stored remotely on multiple virtual servers. There are four different NoSQL databases –

(i) Key-value Stores – It is a system that stores values indexed for retrieval by keys. These systems can hold structured or unstructured data and can easily be distributed to a cluster or a collection of nodes as in Amazon's DynamoDB and Project Voldemort.

(ii) Column-oriented Databases – It is a system that stores data in whole column instead of a row, which minimizes disk access compared to a

heavily structured table of columns and rows with uniform sized fields for each record as in HBase and Cassandra.

(iii) Document-based Stores – These databases store data and organize them as document collections, instead of structured tables with uniform sized fields for each record. With this database, users can add any number of fields of any length to a document as implemented in Couch DB, Mongo DB.

(iv) Graph Databases – These databases use nodes, edges, and properties to represent and store data in the form of graphs. It is possible to represent data as a graph-like structure, which can be easily traversed as in Allegro Graph and Neo4i.

Q.2. Write down the features of NoSQL.

Ans. NoSQL databases may not require a predefined table schema, typically scale horizontally and usually avoid join operations. Because of schema less nature and involvement of smaller subset analysis of NoSQL system, this database can be better described as structured data stores. Three important basic features of NoSQL databases are scale-out, flexible data structure and replication, which are explained below –

(i) Scale-out – This refers to achieve high performance in a distributed environment by using many general-purpose machines. NoSQL databases allow the distribution of the data over a large number of machines with a distributed processing load. Many NoSQL databases allow automatic distribution of data to new machines when they are added to the cluster. Scale-out is evaluated in terms of scalability and elasticity.

(ii) Flexibility – Flexibility in terms of data structure says that there is no need to define a schema for databases. NoSQL databases do not require a predefined schema. This allows the users to store data of various structures in the same database table. However, support for high-level query languages such as SQL is not supported by most of the NoSQL databases.

(iii) Data Replication – One of the features of NoSQL databases is data replication. In this process a copy of the data is distributed to different systems in order to achieve redundancy and load distribution. However there is a chance of losing data consistency among the replicas. But it is believed that sometimes this consistency may be achieved eventually. Consistency and availability are the factors for evaluating replication.

Q.3. Write short note on oracle big data.

Ans. Oracle is the first vendor to offer a complete and integrated solution to address the full spectrum of enterprise big data requirements. Oracle's big data strategy is centered on the idea that you can extend your current enterprise information architecture to incorporate big data. New big data technologies,

such as Hadoop and Oracle NoSQL database, run alongside your Oracle data warehouse to deliver business value and address your big data requirements.

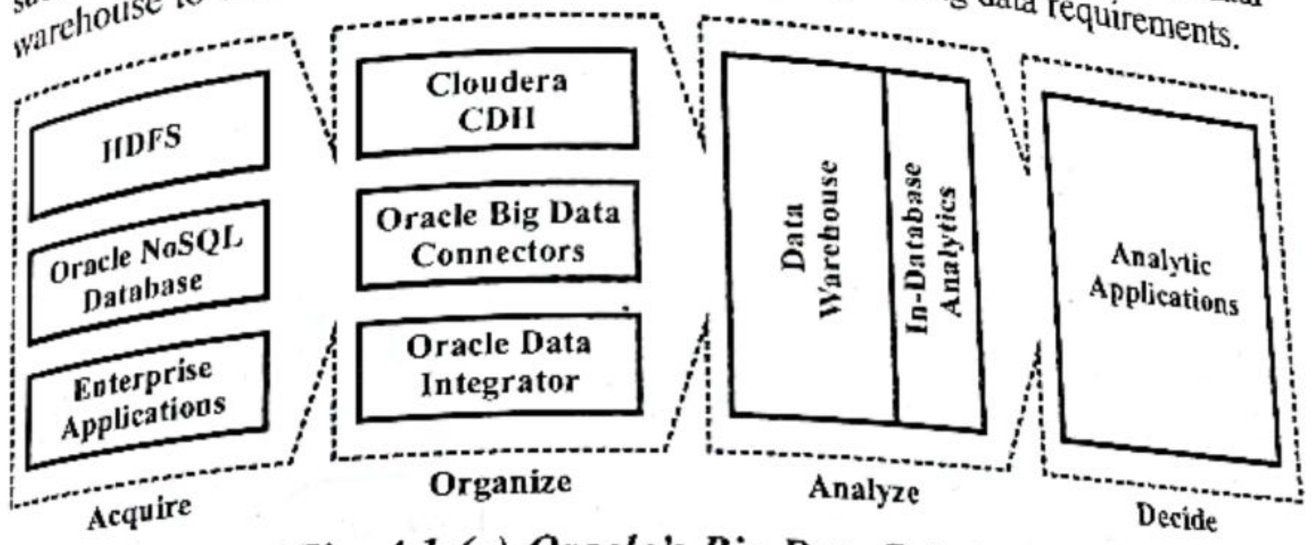

Fig. 4.1 (a) Oracle's Big Data Solutions

Oracle Big Data Appliance – Oracle big data appliance is an engineered system that combines optimized hardware with a comprehensive big data software stack to deliver a complete, easy-to-deploy solution for acquiring and organizing big data.

Oracle big data appliance comes in a full rack configuration with 18 sun servers for a total storage capacity of 648 TB. Every server in the rack has 2 CPUs, each with 8 cores for a total of 288 cores per full rack. Each server has 64 GB memory for a total of 1152 GB of memory per full rack.

Oracle big data appliance includes a combination of open source software and specialized software developed by oracle to address enterprise big data requirements.

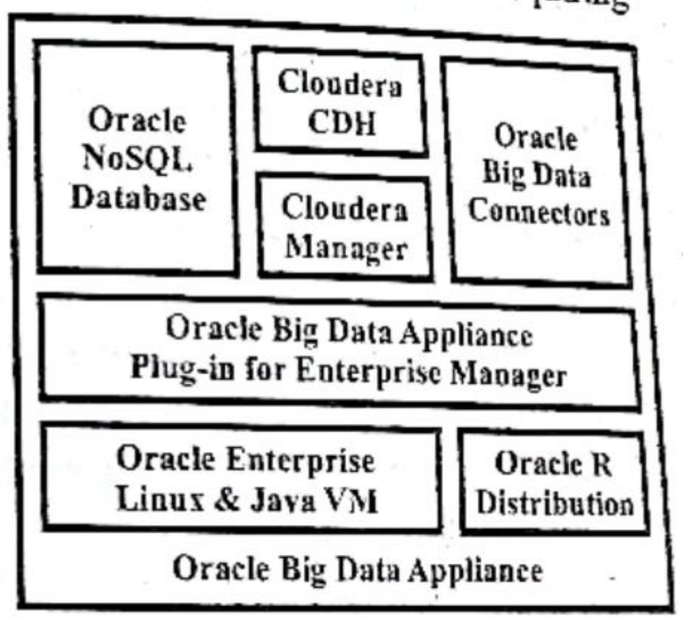

Fig. 4.1 (b) High-level Overview of Software on Big Data Appliance

The oracle big data appliance software includes –

(i) Full distribution of Cloudera's Distribution including Apache Hadoop (CDH4).

(ii) Oracle Big Data Appliance Plug-in for Enterprise Manager.

(iii) Cloudera Manager to administer all aspects of Cloudera CDH.

(iv) Oracle distribution of the statistical package R.

(v) Oracle NoSQL Database Community Edition.

(vi) Oracle Enterprise Linux operating system and Oracle Java VM.

Q.4. What is the role of NoSQL business drivers ?

Ans. Most of the organizations supporting single CPU relational systems have come to a cross roads. The need of these organizations are changing. Businesses have found value in rapidly capturing and analyzing large amounts of variable data, and making immediate changes in their businesses based on the information they receive.

The business drivers names are velocity, volume, variability and agility, these play a important role in the emergence of NoSQL solutions. As each of these drivers applies pressure to the single-processor relational model, its foundation becomes less stable and in time no longer meets the organization's needs. In short, volume and velocity refer to the ability to handle large data-sets that arrive quickly. Variability refers to how diverse data types don't fit into structured tables, and agility refers to how quickly an organization responds to business changes. The NoSQL business drivers are shown in fig. 4.2.

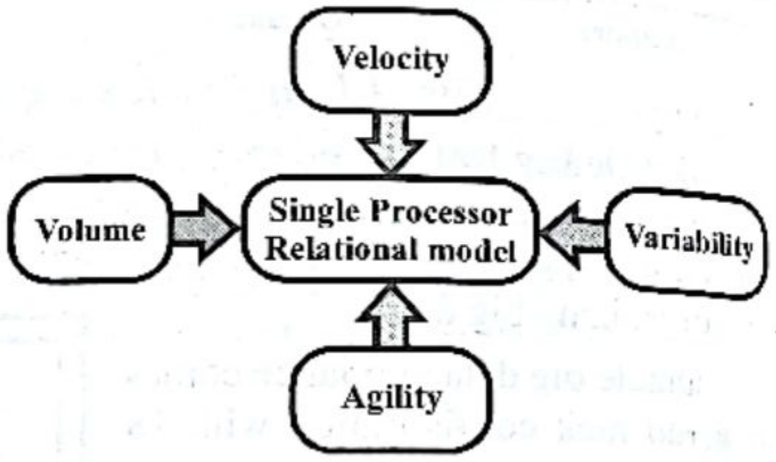

Fig. 4.2 NoSQL Business Drivers

(i) Velocity – Though big data problems are a consideration for many organizations moving away from RDBMSs, the ability of a single processor system to rapidly read and write data is also key. Many single-processor RDBMSs are unable to keep up with the demands of real-time inserts and online queries to the database made by public-facing websites. RDBMSs frequently index many columns of every new row, a process which decreases system performance. When single-processor RDBMSs are used as a back end to a web store front, the random bursts in web traffic slow down response for everyone, and tuning these systems can be costly when both high read and write throughput is desired.

(ii) Volume – Without a doubt, the important factor pushing organizations to found alternatives to their current RDBMSs is a need to query big data using clusters of commodity processors. Until around 2005, performance concerns were resovled by purchasing faster processors. In time, the ability to increase processing speed was no longer an option. As chip density increased, heat could no longer dissipate fast enough without chip overheating. This phenomenon, known as the power wall, forced systems designers to shift their focus from increasing speed on a single chip to using more processors working together. The need to scale out (also known as horizontal scaling), rather than scale up (faster processors), moved

organizations from serial to parallel processing where data problems are split into separate paths and sent to separate processors to divide and conquer the work.

(iii) Variability – Companies that want to capture and report on exception data struggle when attempting to use rigid database schema structures imposed by RDBMSs. For example, if a business unit wants to capture a few custom fields for a particular customer, all customer rows within the database need to store this information even though it doesn't apply. Adding new columns to an RDBMS requires the system to be shut down and ALTER TABLE commands to be run. When a database is large, this process can impact system availalbility, costing time and money.

(iv) Agility – The most complex part of building applications using RDBMSs is the process of putting data into and getting data out of the database. If your data has nested and repeated subgroups of data structures, you need to include an object-relational mapping layer. The responsibility of this layer is to generate the correct combination of INSERT, UPDATE, DELETE, and SELECT SQL statements to move object data to and from the RDBMS persistence layer. This process is not simple and is associated with the largest barrier to rapid change when developing new or modifying existing applications.

Q.5. Write short note on NoSQL data architectural patterns.

Ans. There are four basic types of NoSQL data architectural patterns in the broad sense, key-value, document oriented, Graph oriented and column oriented. A large number of cloud databases have been developed under each category. This implies the need to understand the differences among such data stores, and which is more suitable to any given data. Key-value model allows representing the data in a simple format. Document oriented model allows representing data via structured text. In graph oriented data is stored in the form of graph. Column oriented model allows representing data in columns and data is stored in tables.

Q.6. Explain the key-value data stores with example.

Ans. In order to handle highly concurrent access to database, the category of NoSQL designed is key-value stores. It is the simplest, still the most powerful data store. In a key-value store each data consists of a pair of a unique key and value. In order to save data a key gets generated by the application and value gets associated with the key. And this key-value pair gets submitted to the data store. The data values stored in key-value stores can have dynamic sets of attributes attached to it and is opaque to the database management system. Hence key is the only means to access the data values. The type of binding from the key to value depends on the programming language used in the application. An application needs to provide a key to the data stores in order to

retrieve data. Many key-value data stores use a hash function. The application hashes the key and find out the location of the data in the database. The key-value data stores are row focused. Which means it enables the application to retrieve data for complete entities.

Retrieval of data from a key-value database is shown in fig. 4.3. The application has specified a key 'Emp20' to the data store in order to retrieve data. Using the hash function the application hashes the key in order to trace the location of data in the data store. The design of the key should support the most frequent queries fired on the data store. Efficiency of the hash function, design of the key and size of the values being stored are the factors which affect the performance of a key-value data store. The operations performed on such data stores are mostly limited to read and write operations. Because of the simplicity of the key-value data store, it provides users with fastest means of storing and fetching data. All other categories of NoSQL are built upon the simplicity, scalability and performance of key-value data stores.

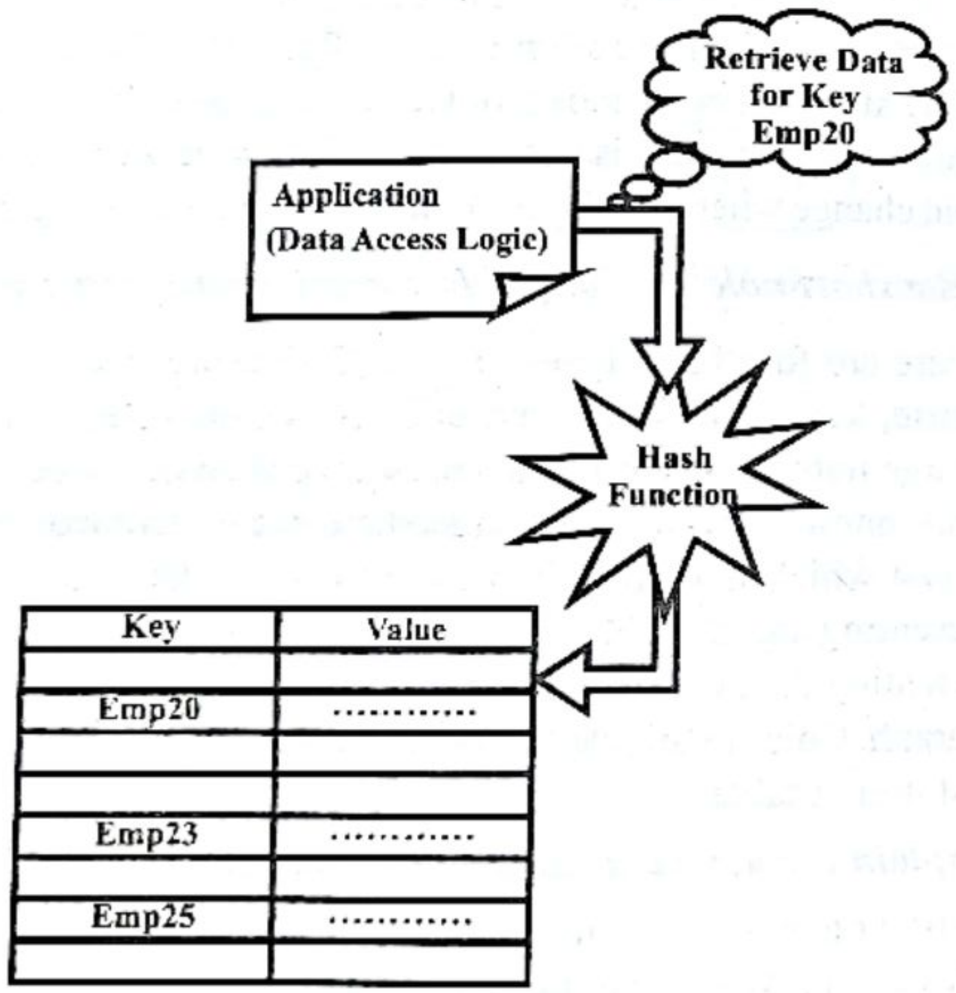

Fig. 4.3 A Key-value Data Store

Examples of key-value data stores are Berkeley DB, Redis, Memcache, Riak and DynamoDB.

Q.7. What are the characteristic features of key-value data store ?

Ans. The characteristic features of key-value data stores are as follows –

(i) Key-value data stores provides a single means of accessing data by primary key.

(ii) Utilize in memory storage to provide fast access with optional persistence.

(iii) Other data models built on top of this model to provide more complex objects.

(iv) Document databases give the richest query functionality, which allows them to address a wide variety of operational and real-time analytics applications.

(v) Can be used for applications that require fast changing data environments like mobile, gaming, online ads.

(vi) The simplest model where each object is retrieved with a unique key, with values having no inherent model.

Q.8. Give some popular use cases for key-value based data stores.

Ans. The typical usage for key-value based data stores are as follows –

(i) Distributing Information – They can be used to implement pub/sub.

(ii) Queueing – Some key-value data stores like Redis supports lists, queues and set etc.

(iii) Keeping Live Information – Applications which need to keep a state can use key-value data stores easily.

(iv) Caching – Quickly storing data for sometimes frequent future use.

Q.9. Explain the document oriented data stores with example.

Ans. Document oriented data stores are used to store and organize data in the form of document. At an abstract level document oriented database is similar to key-value data store. It also holds value, which an application can read or fetch by using a key. Several document databases automatically generate the unique key while creating a new document. A document in a document database is an entity, which is a collection of named fields. The feature which distinguishes the document oriented database from a key-value data store is transparency of the data held by the database. Hence the query possibility is not restricted with the key only. In order to support scenarios where the application requires querying the database not only based on its key but also with attribute values, can switch for document databases. A document needs to be self-describing in a document oriented database. Information is stored in a portable and well understood format such as XML, BSON or JSON.

As shown in fig. 4.4 the document database stores data in form of key-value pairs. But the data stored in the database is transparent to the system unlike key-value databases. The application can query the database not only

with the key i.e. 'Employee ID' but also with the defined fields in the document i.e. firstname (FN), lastname (LN), age etc. Document data stores are efficient approach to model data based on common software problems. But it comes at the cost of slightly lower performance and scalability in comparison to key-value data stores. Few examples of the most prominent document stores are Riak, MongoDB, CouchDB, Marklogic, exist-db, Berkeley DBXML, Counch base etc.

Key (Employee ID)	Document
Emp20	FN : Raj LN : Verma Age : 22
.....	
.....	
Emp100	FN : Veena LN : Sharma Age : 25

Fig. 4.4 A Document Data Store

Q.10. Write down some popular use case for document oriented data stores.

Ans. Some popular use cases for document oriented based data stores are as follows –

(i) JavaScript Friendly – One of the most critical functionalities of document based data stores are the way they interface with applications using javascript friendly Javascript object notation (JSON).

(ii) Nested Information – Document based data stores allow you to work with deeply nested complex data structures.

Q.11. Explain graph oriented data stores with example.

Ans. Graph databases are considered to be the specialists of highly linked data. Therefore it handles data involving a huge number of relationships. There are basically three core abstractions of graph database. These are nodes, edges which connect two different nodes, and properties. Each node holds information about an entity. The edges represent the existence of relationship between the entities. Each relationship is having a relationship type and is directional with a start point (node) and an end point. The end point can be some other node than that of the start node or possibly the same node. Key-value properties are associated not only with the nodes but also with the relationships. The properties of the relationships provide additional information about the relationships. The direction of the relationship determines the traversal path from one node to the other in a graph database.

Fig. 4.5 represents a part of the 'Employee' database structured as graph database. Each node in this graph database represents an employee entity. These entities are related with each other through a relationship of relationship type "knows". The property associated with the relationship is "Duration". The key difference between a graph and relational database is data querying.

Instead of using cost intensive process like recursive join as in relational database graph databases use traversal method. While querying through graph database a start node has to be specified by the application. Traversal starts from the start node and progresses via relationships to nodes connected to the start node, based upon some rule defined by the application logic. The traversal method involves only nodes which are relevant to the application not the entire data set. Hence, a huge increase in number of nodes does not affect the traversal rate much. Social networking, data mining, managing networks, and calculating routes are few of the fields where graph database has been used extensively. Neo4J, Apache graph, Allegro-graph are popular examples of graph databases.

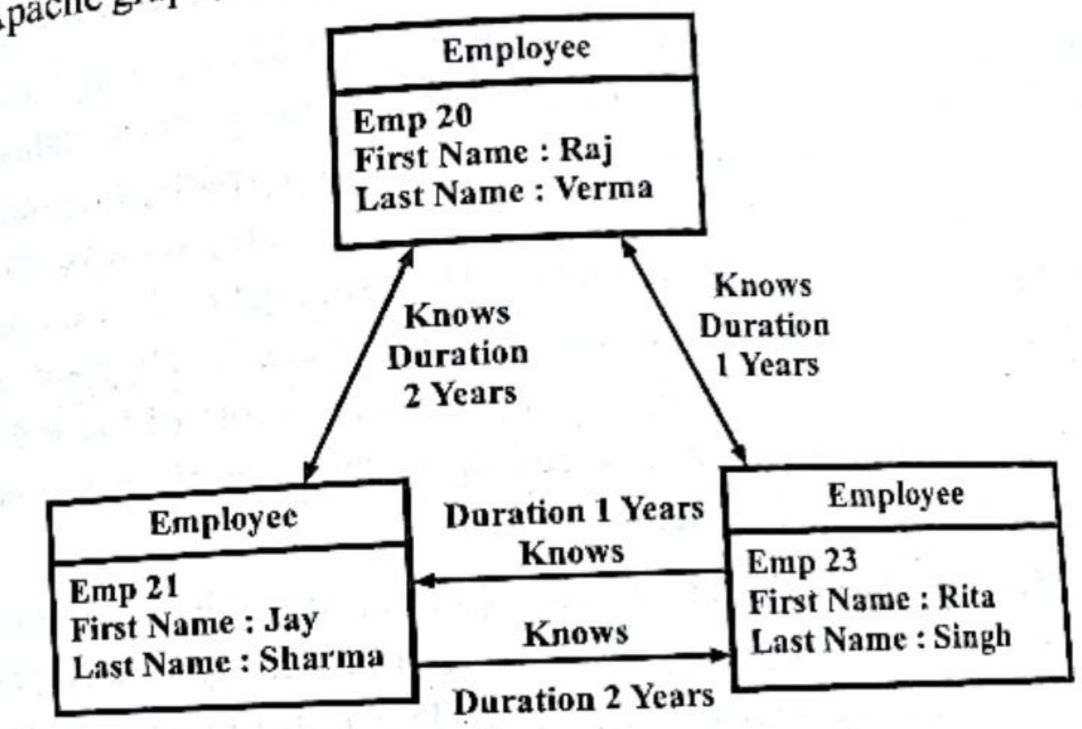

Fig. 4.5 A Graph Oriented Database

Q.12. What are the characteristic features of graph oriented data store ?

Ans. The characteristic features of graph oriented data stores are as follows –

(i) It implements very fast graph traversal operations.

(ii) Graph oriented database also supports indexing of meta data to enable graph traversal combined with search queries.

(iii) Application that deal with objects with a large number of inter-relations.

(iv) Models graphs consisting of nodes and edges with properties describing them.

(v) Applications like social networking friends networks, hierarchical role based permissions, maps and network topologies.

Q.13. Write down important use cases for graph oriented based data store.

Ans. Some important use cases for graph oriented based data stores are as follows –

(i) Modelling and Handling Classification – Graph oriented databases excel in any situation where relationships are involved. Modelling data and classifying various information in a relational way can be handled very well using these type of data stores.

(ii) Handling Complex Relational Information – Graph oriented database makes it extremely efficient and easy to handle with complex but relational information such as the connections between two entities and various degrees of other entities indirectly related to them.

Q.14. Explain the column oriented database store with example.

Ans. Column oriented datastores are designed to store huge numbers of columns. Data is stored based on column values. Though these datastores are the most similar to their traditional relational counterparts, they are able to overcome the drawbacks of the latter databases, as they remove null values from columns, when values are unknown. They support high scalability since column data can be distributed on several clusters easily. They are also most suitable for data mining and analytics applications. Most of these datastores employ MapReduce framework to speed up processing of large amounts of data distributed on numerous clusters.

Sometimes an application may want to read or fetch a subset of fields, similar to the SQL's projection operation. Column family data store enables storing data in column centric approach. The column family data store partitions the key space. In NoSQL a key space is considered to be an object which holds all column families of a design together. It is the outer most grouping of the data in the data store. Each partition of the key space is known to be a Table. Column families are declared by these tables. Each column family consists of number of columns. A row in a column family is structured as collections of arbitrary number of columns. Each column is a map of a key-value pair. In this map, keys are the names of columns and columns themselves are the values. Each of these mappings is called a cell. Each row in a column-family database is identified by a unique row key, defined by the application. Use of these row keys makes the data retrieval quicker. In order to avoid overwriting of the cell values few of the popular column-family databases add time stamp information automatically to individual columns.

A simple example data structure of a column datastore is shown in fig. 4.6. It stores information similar to that of the document datastore in fig. 4.6, but in a different column-oriented format.

One of the popular column data stores is Cassandra, which was developed by Apache Software Foundation, and implemented in Java. It is based on both Amazon's DynamoDB key-value datastore and Google's Bigtable column datastore, so it includes concepts of both datastore types. It supports high availability, partitioning tolerance, persistence and high scalability. It also has a dynamic schema. It can be used for a variety of applications like social networking websites, banking and finance, and real time data analytics. Some other examples of column data stores are Apache HBase, DynamoDB, Apache Accumulo, Hyper table.

Fig. 4.6 Simple Data Structure of a Column Datastore

Q.15. What are the characteristic features of column oriented data stores.

Ans. The characteristic features of column oriented data stores are as follows –

(i) A column can have multiple time stamped versions.

(ii) Extension of key-value model, where the value is a set of columns.

(iii) Storing a large number of time-stamped data like event logs and sensor data.

(iv) Columns can be generated at run time and not all rows need to have all columns.

(v) It provides more granular access to data than the key value datastore, but less flexibility than the document oriented data store.

Q.16. Write down some important use cases of column oriented based data stores.

Ans. Some important use cases of column oriented based data stores are as follows –

(i) Scaling – These are highly scalable by nature and also handle a huge amount of information.

(ii) Keeping Unstructured Non-volatile Information – If a large collection of attributes and values needs to be kept for long periods of time, column based data stores come in extremely handy.

Q.17. Explain the comparison of NoSQL data architecture patterns.

Ans. The comparison of NoSQL database stores is shown in table 4.1.

Table 4.1 Comparison of NoSQL Database Stores

Characteristic/ Data Model	*Column-oriented*	*Document-oriented*	*Graph-oriented*	*Key-value*
Concept	A model that allows representing data in columns	A model that allows representing data via structured text	A model that allows representing data and their connections	A model that allows representing the data in a simple format (key and values)
Structure	Data are stored in tables	Nesting of key-value pairs	Set of data objects (nodes)	Tuple of two strings (key and value)
		Each document identified by a unique identifier	Set of links between the objects (edges)	A key represents an entity's attribute
	Values in a column are stored consecutively	Any value can be a structured document		Values can be of any data type
		Key and value are separated by a colon ":"		
		Key-value pairs are separated by commas ","		
		Data enclosed in curly braces denotes documents		
		Data enclosed in square brackets denotes array collection		
Techniques	With compression : Lightweight encoding Bit-vector encoding Dictionary encoding Frame of reference encoding Differential encoding	Denormalized flat model	Simple direct graph, Undirected multigraph, Directed multigraph	NA
		Denormalized model with more structure (metadata)	Weighted graph	
	With join algorithm	Shattered, equivalent to normalization	Hypergraph	
	With late materialization		Nested graph	
	Tuple at a time			
Applications	Consumer data, Inventory data	JSON documents, XML documents	Social networks, Supply-chain, Medical records, IT operations, and Transports	User-profiles and their attributes
Advantages	High performance in loading and querying operations Efficient data compression and partitioning (both horizontally and vertically) Scalability Support for massive parallel processing Well-suited for Online Analytical Processing and Online Transaction Processing workloads	Support for multiple document types Support for atomicity, consistency, isolation and durability transactions Scalability Suitable for complex data, nested documents and arrays	Easy modeling Fast and simple querying Scalability	Easy design and implementation, Fault tolerance, Redundancy, Scalability, High speed
Disadvantages	Difficult to use wide-columns Delays in querying specific data	Information duplication across multiple documents Inconsistencies in complex designs	Lack of a standard declarative language Support to limited concurrency and parallelism	Very basic query language Some queries can only depend on the primary key

Q.18. What is Cassandra ? Write down it's advantages and disadvantages.

Ans. Cassandra was initially developed by face book to handle large volume of data and later it was acquired by Apache Software foundation in the year of 2009. It provides no single point of failure, high availability, high scalability, highly fault tolerant and high performance of data. Many servers are interconnected with each other and if one node goes down, another node can provide service to the end user. To allow all the nodes to communicate with each other and to detect the faulty node a gossip based protocol is used.

Advantages –

(i) The biggest companies such as face book, twitter, rack space and cisco uses the cassandra to handle their data.

(ii) It contains dynamo-style replications model to provide no single point of failure.

(iii) It provides high throughput and quick response time if the number of nodes in the cluster increases.

(iv) The ACID property is supported for transaction.

(v) It is a column oriented data base.

Disadvantages –

(i) It does not support sub query and join operation.

(ii) Limited support for data aggregation.

(iii) Limited storage space for a single column value.

Q.19. What do you understand by Hbase ? Explain the architecture of Hbase.

Ans. Hbase is the Hadoop database which can provide real-time access to the data and powerful scalability. Hbase was designed based on the Bigtable, a database was launched by Google. Hbase aims at storing and processing big data easily. More specifically, it uses a general hardware configuration to process millions of data. Hbase is an open source, distributed, has multiple versions, and uses the NoSQL database model. It can be applied on the local file systems and on HDFS. In addition, Hbase can use the MapReduce computing model to parallel process big data in Hadoop. This is also the core feature of Hbase. It can combine data storage with parallel computing perfectly.

Architecture of Hbase – Hbase is the storage layer in the Hadoop. Its underlying storage support is HDFS, using the MapReduce framework to process the data, and cooperate with the ZooKeeper. The architecture of Hbase is shown in fig. 4.7.

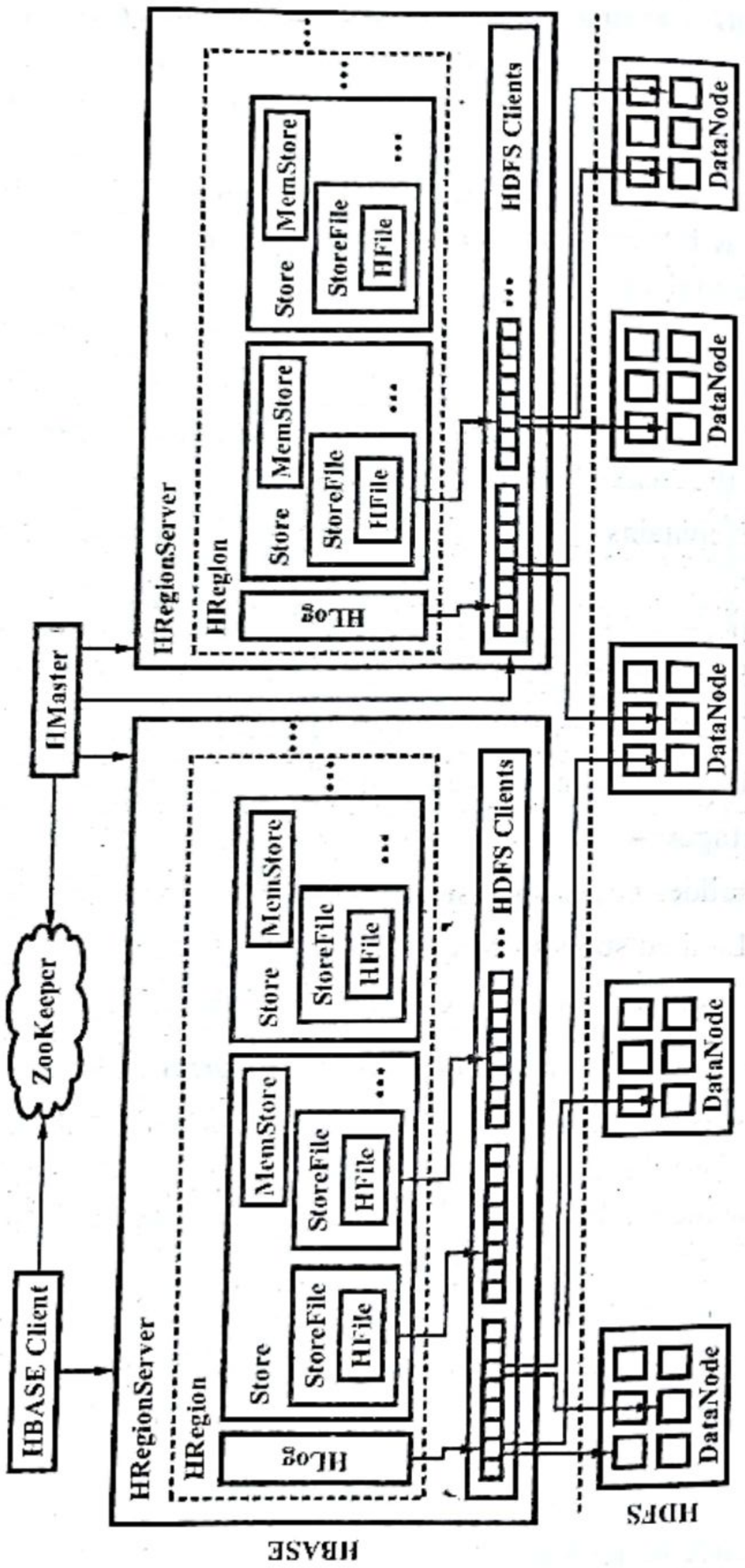

Fig. 4.7 Hbase Architecture

The four key components are as follows –

(i) Hbase Client – The client is the user of the Hbase. It takes part in the manage operations with HMaster and read/write operations with HRegionServer.

(ii) ZooKeeper – ZooKeeper is the collaborative management node of Hbase. It can provide distributed collaboration, distributed synchronization, and configuration functions. The ZooKeeper coordinates all the clusters of Hbase by using data which contains the HMaster address and HRegionServer status information.

(iii) HMaster – HMaster is the controller of the Hbase. It is responsible for adding, deleting, and queuing the data. It adjusts the HRegionServer load balance and the Region distribution to ensure that the region will move to the next Region when the HRegionServer suffers failure. An Hbase environment can launch multiple HMaster to avoid failure. At the same time, there is always a Master Election mechanism working in case of the node failure.

(iv) HRegionServer – HRegionServer is the core component of Hbase. It is responsible for handling the reading and writing requests for the users and performing the corresponding operations on HDFS.

Q.20. Explain the comparison of RDBMS and Hbase.

Ans. The comparison of RDBMS and Hbase is as follows –

S.No.	Criteria	Hbase	RDBMS
(i)	Changeable data	Yes	Yes
(ii)	Data layout	A sparse, distributed, persistent multidimensional sorted map.	Row-oriented or column-oriented.
(iii)	Data types	Bytes; data types are interpreted on query.	Rich data type support.
(iv)	Hardware	Hadoop-clustered commodity x86 servers; five or more is typical because the underlying storage technology is HDFS, which by default requires three replicas.	Typically large, scalable multi-processor systems.
(v)	High availability	Yes; built into the Hadoop architecture.	Yes, if the hardware and RDBMS are configured correctly.
(vi)	Indexes	Row-key only or special table required.	Yes
(vii)	Query language	Hbase API commands (get, put, scan, delete, increment, check), HiveQL	SQL

Hbase, as the representative database, is often compared with the traditional RDBMS. The design target, implementation mechanism, and running performance are different. Due to the reason that the Hbase and RDBMS can replace each other in some special situations, it is inevitable to compare RDBMS with Hbase. As mentioned before, Hbase is a distributed database system and the underlying physical storage uses the Hadoop distributed file system. It does not have particularly strict requirements on the hardware platform. However, RDBMS is a fixed structure database system. The difference between their design goals makes them have the greatest difference in the implementation mechanism.

Q.21. Explain in detail about challenges of NoSQL.

Ans. The promise of the NoSQL has generated a lot of enthusiasm, but there are many obstacles to overcome before they can appeal to mainstream enterprises. Here, the important challenges are as follows –

(i) Maturity – Relational database systems have been around for a long time, stable as well as richly functional. NoSQL (Not only SQL) advocates will argue that their advancing age is a sign of their obsolescence, but for most CIOs, the maturity of the RDBMS is reassuring. Most, NoSQL alternatives are in preproduction versions with large key features yet to be implemented. Living on the technological leading edge is a demanding prospect for large developers, but enterprises should approach it with extreme caution.

(ii) Support – Every organization wants the reassurance that if a key system fails, they will be able to get competent support as well as timely. All relational databases management system vendors go to great lengths to support a high level of enterprise. Many NoSQL systems are open source projects, and although there are usually one or more firms offering support for each NoSQL database, these companies often are small start-ups without the global reach, support resources, as well as credibility of a Microsoft, Oracle, or IBM.

(iii) Business Intelligence and Analytics – Not only SQL databases have evolved to meet the scaling demands of modern Web 2.0 applications as well as offer some facilities for ad-hoc query and analysis. Some relief is provided by the emergence of solutions like Hive or Pig that can provide easier access to data held in Hadoop clusters as well as perhaps eventually, other NoSQL databases.

(iv) Administration – The main goals for NoSQL may be to provide a zero admin solution, but the current reality falls well short of that goal. NoSQL today requires a lot of skill to install as well as a lot of effort to maintain.

(v) Expertise – Almost each NoSQL developer is in a learning mode and situation will address naturally over time, but for now, it is far easier to find experienced RDBMS programmers or administrators than a NoSQL expert.

Q.22. Discuss about the variations of NoSQL architectural patterns.

Ans. The variations of NoSQL architectural patterns are discussed below –

(i) RAM or SSD Stores – A key-value store that only uses RAM is called a RAM cache, it is flexible and has general tools that application developers can use to store global variables, configuration files, or intermediate results of document transformations. A RAM cache is fast and reliable, and can be thought of as another programming construct like an array, a map or a lookup system.

Simple RAM resident key-value stores are generally empty when the server starts up and can only be populated with values on demand. RAM resident information must be saved to another storage system if we want it to persist between server restarts. We need to define the rules about how memory is partitioned between the RAM cache and the rest of our application.

SSD systems provide permanent storage and are almost as fast as RAM for read operations. The Amazon DynamoDB key value store services uses SSDs for all its storage, resulting in high-performance read operations. Write operations to SSDs can often be buffered in large RAM caches, resulting in fast write times until the RAM becomes full.

(ii) Distributed Stores – The ability to elegantly and transparently scale to a large number of processor is a core property of most NoSQL systems. Ideally the process of data distribution is transparent to the user meaning that the API doesn't require you to know how or where your data is stored. But knowing that your NoSQL software can scale and how it does this is critical in the software selection process.

If our application uses many web servers, each caching the result of a long-running query, it is most efficient to have a method that allows the servers to work together to avoid duplication. This mechanism is known as memcache. The memcache protocol shows that we can create simple communication protocols between distributed systems to make them work efficiently as a group. This type of information sharing can be extended to other NoSQL data architectures such as column stores (bigtable stores) and document stores. We can generalize the key-value pair to other patterns by referring to them as cached items.

The cached items need to be replicated automatically on multiple servers to give a seamless data service without interruption. If the cached items are stored on two servers and the first one becomes unavailable, the second server can quickly return the value without waiting for the first server to be rebooted or restored from backup.

(iii) Grouping – The implementation of a collection system can also vary dramatically based on what NoSQL data pattern we use. Key-value stores have several methods to group similar items based on attributes in their keys. Graph stores associate one or more group identifiers with each triple. Big data systems use column families to group similar columns. Document stores use a concept of a document collection.

Q.23. Discuss various ways in which NoSQL handles big data.

Ans The most popular four ways in which NoSQL systems handle and manage big data obstacles are as follows –

(i) By Moving Queries to the Data – When a client wants to send a general query to all nodes that hold data, it is more efficient to send the query to each node than it is to transfer large datasets to a central processor.

This simple rule helps us to understand how NoSQL databases can have dramatic performance advantages over systems that were not designed to distribute queries to the data nodes. Consider an RDBMS that has tables distributed over two different nodes. In order for the SQL query to work, information about rows on one table must all be moved across the network to the other node. Larger tables result in more data movement, which results in slower queries. Think of all the steps involved. The tables can be extracted, serialized, sent through the network interface, transmitted over networks, reassembled and then compared on the server with the SQL query. Keeping all the data within each data node in the form of logical documents means that only the query itself and the final result need to be moved over a network. This keeps our big data queries fast.

(ii) Using Hash Ring to Evenly Distribute the Load – Using a hash ring technique to evenly distribute big data loads over many servers with a randomly generated 40-character key is a good way to evenly distribute a network load. The hash rings are common in big data solutions because they consistently determine how to assign a piece of data to a specific processor. These rings take the leading bits of a document's hash value and use this to determine which node the document should be assigned. This allows any node in a cluster to know what node the data lives on and how to adapt to new assignment methods as our data grows. Partitioning keys into ranges and assigning different key ranges to specific nodes is known as keyspace management. Most NoSQL systems including MapReduce, use keyspace concepts to manage distributed computing problems.

(iii) Using Replication to Scale Reads – The replication strategy works well in most cases. Only a few times there is some lag time between a write to the read/write node and a client reading of that same record from a replica. If a client does a write and then an immediate read from that same node, there's no difficulty. The problem occurs if a read occurs from a replica node before the update happens. This is an example of an inconsistent read.

The best way to avoid this type of problem is to only allow reads to the same write node after a write has been done. This logic can be added to a session or state management system at the application layer. Almost all distributed databases relax database consistency rules when a large number of nodes permit writes.

(iv) Allowing the Database to Distribute Queries Evenly to Data Nodes – Fig. 4.8 shows the approach of moving the query to the data rather than moving the data to the query. This is an important part of NoSQL big data strategies. In this instance, moving the query is handled by the database server, and distribution of the query and waiting for all nodes to respond is the only responsibility of the database, not the application layer.

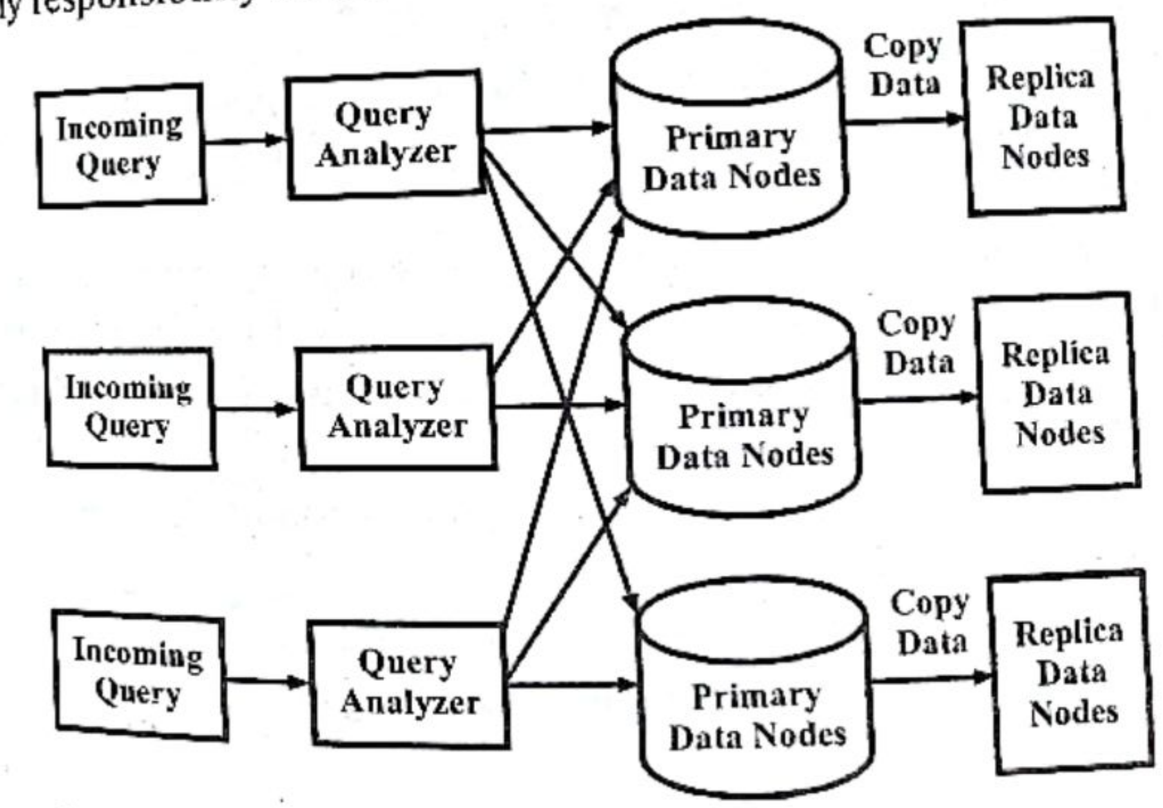

Fig. 4.8 NoSQL System Moving the Query to a Data Node

INTRODUCTION TO MONGODB

Q.24. What is MongoDB ?

Ans. MongoDB is a famous NoSQL database that is an open source, written in C++, cross-platform, high performance as well as document oriented database. MongoDB uses collections to store data as well as

represent relationships between them and data is in the format of BSON documents. It is a binary format in that zero or more key/value pairs are stored as a single entity i.e. as a document. BSON is based on JSON style documents. JSON (JavaScript Object Notation) is a format that is easy for computers to parse and generate. MongoDB works on concept of collection and document.

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Q.25. What are the features of MongoDB ?

Ans. MongoDB features include full index support, replication, high availability, and auto-sharding. Some important features of MongoDB are discussed below –

(i) Indexing – MongoDB supports secondary indexing, that makes retrieval faster as well as unique, compound and geospatial indexing is also possible.

(ii) Stored JavaScript – Users can also use JavaScript function as well as scripts on server side.

(iii) Aggregation – MongoDB supports MapReduce that is a very useful aggregation tool.

(iv) Horizontal Scaling – MongoDB scales horizontally. It scales out and up easily on a variety of platforms including in the cloud using services like Amazon EC2 and Rackspace.

(v) Sharding – This is a process in which large databases are broken down into different tables so that they can be processed on multiple machines and in MongoDB, this is automatic. MongoDB's sophisticated sharding keys make balancing your data across large clusters easy and powerful.

Q.26. How MongoDB used in BlogPost ?

Ans. Using MongoDB database, our blog posts can be stored in a single collection, with each entry looking like this – With a document type database data is stored almost exactly as it is represented in the program.


```
{_id:1,
 Author : {Name: "Rakesh Kumar", email : "rakesh@redinno.in"},
 Post : "I like you",
 Date: {$date: "2015-03-12:47UTC"},
 Location : [–121.2322, 42.1223222],
 Rating: 1.1,
 Comments: [
     {User: "rakesh@hotmail.com",
     Upvotes: 11,
     Downvotes : 04,
     Text : "I agree with you"},
     {User: "dolly@gmail.com",
     Upvotes: 321,
     Downvotes: 11,
     Text : "You are a man"}
 ],
 Tags: ["Politics", "Virginia"]}
```

Q.27. Give advantages and disadvantages of MongoDB.

Ans. **Advantages of MongoDB –**

(i) Schema-less (without scheme) design enables rapid introduction of new CDR (Call Detail Record) types of the System.

(ii) Scale BillRun production site already controls several TB in a single table, without being limited by adding new fields.

(iii) Rapid replica set easily enables meeting regulation with easy to setup multi data center DRP as well as HA solution.

(iv) Sharding enables linear as well as scale out growth without running out of budget.

(v) With over approximately 2,000/s CDR inserts, then MongoDB architecture is great for a system that must support high insert load. So you can easily guarantee transactions with findAndModify as well as two-phase commit.

(vi) Supports developer oriented queries.

(vii) Location based is being utilized to analyze users usage as well as determining where to invest in cellular infrastructure.

Disadvantages of MongoDB –

(i) The current database is locked when MongoDB is writing onto it; therefore this does not allow concurrent writes.

(ii) Mongo DB reports scalability constraints when the data exceeds hundreds of GB.

Q.28. Give reasons to use MongoDB.

Ans. Few of the reasons to use MongoDB are as follows –

(i) Document-oriented – Since MongoDB is a NoSQL type database, therefore instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.

(ii) Ad hoc Queries – MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.

(iii) Indexing – Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.

(iv) Replication – MongoDB can provide high availability with replica sets. A replica set consists of two or more MongoDB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.

(v) Load Balancing – MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

Q.29. Discuss about the MongoDB datatypes.

Ans. MongoDB supports following datatypes –

(i) String – This is most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.

(ii) Integer – This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.

(iii) Boolean – This type is used to store a Boolean (true/false) value.

(iv) Double – This type is used to store floating point values.

(v) Min/Max Keys – This type is used to compare a value against the lowest and highest BSON elements.

(vi) Arrays – This type is used to store arrays or list or multiple values into one key.

(vii) Timestamp – This can be handy for recording when a document has been modified or added.

(viii) Object – This datatype is used for embedded documents.

(ix) Null – This type is used to store a Null value.

(x) Symbol – This datatype is used identically to a string however, it is generally reserved for languages that use a specific symbol type.

(xi) Date – This datatype is used to store the current date or time in UNIX time format. We can specify our own date time by creating object of Date and passing day, month, year into it.

(xii) Object ID – This datatype is used to store the document's ID.

(xiii) Binary Data – This datatype is used to store binary data.

(xiv) Code – This datatype is used to store javascript code into document.

(xv) Regular Expression – This datatype is used to store regular expression.

Q.30. Give advantages of MongoDB over RDBMS.

Ans. Advantages of MongoDB over RDBBMS are as follows –

(i) Schema less : MongoDB is document database in which one collection holds documents. Number of fields, content and size of the document can differ from one document to another.

(ii) Structure of a single object is clear.

(iii) No complex joins.

(iv) Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that is nearly as powerful as SQL.

(v) Tuning.

(vi) MonogoDB is easy to scale.

(vii) Conversion/mapping of application objects to database objects not needed.

(viii) Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Q.31. What are the differences between MongoDB and RDBMS ?

Ans. The differences between MongoDB and RDBMS are as follows –

RDBMS	*MongoDB*	*Difference*
Table	Collection	In RDBMS, the table contains the columns and rows which are used to store the data. In MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs.
Row	Document	In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB the data is stored in documents.
Column	Field	In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields.
Joins	Embedded documents	In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using embedded documents. So there is no concept of joins in MongoDB.

■■

UNIT 5

MINING SOCIAL NETWORK GRAPHS

INTRODUCTION, APPLICATIONS OF SOCIAL NETWORK MINING, SOCIAL NETWORKS AS A GRAPH, TYPES OF SOCIAL NETWORKS

Q.1. What is social network ? Explain.

Ans. Social network is a term used to describe web-based services that allow individuals to create a public/semi-public profile within a domain such that they can communicatively connect with other users within the network. Social network has improved on the concept and technology of Web 2.0, by enabling the formation and exchange of User-Generated Content. Simply put, social network is a graph consisting of nodes and links used to represent social relations on social network sites. The nodes include entities and the relationships between them forms the links as shown in fig. 5.1.

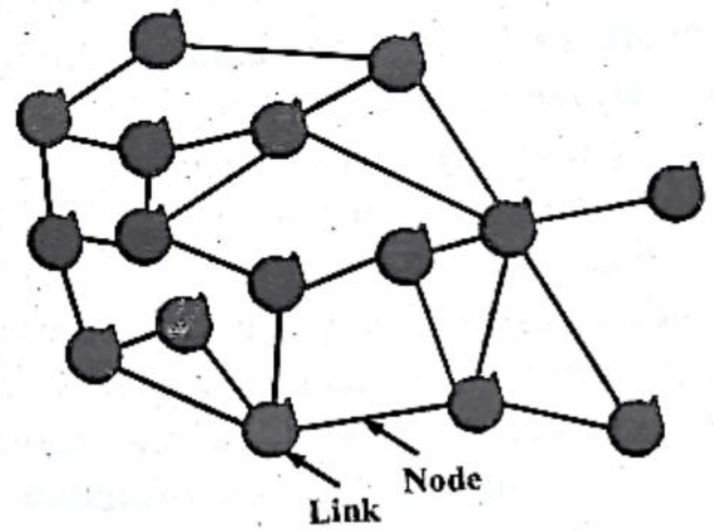

Fig. 5.1 Social Network Showing Nodes and Links

Social networks are important sources of online interactions and contents sharing, subjectivity, assessments, approaches, evaluation, influences, observations, feelings, opinions and sentiments expressions borne out in text, reviews, blogs, discussions, news, remarks, reactions, or some other documents. Before the advent of social network, the homepages were popularly used in the late 1990s which made it possible for average internet users to

share information. However, the activities on social network in recent times seem to have transformed the World Wide Web (www) into its intended original creation. Social network platforms enable rapid information exchange between users regardless of the location. Many organisations, individuals and even government of countries now follow the activities on social networks. These networks enable big organisations, celebrities, government official and government bodies to obtain knowledge on how their audience reacts to postings that concerns them out of the enormous data generated on social network.

Social networks need not be social in context. There are many real-world instances of technological, business, economic, and biologic social networks. For example electrical power grids, telephone call graphs, the spread of computer viruses, the World Wide Web, and citation networks of scientists. Customer networks and collaborative filtering problems (where product recommendations are made based on the preferences of other customers) are other examples. In biology, examples range from epidemiological networks, cellular and metabolic networks, and food webs, to the neural network of the nematode worm Caenorhabditis elegans (the only creature whose neural network has been completely mapped). The exchange of e-mail messages within corporations, newsgroups, chat rooms, friendships, sex webs (linking sexual partners), and the quintessential "old-boy" network (i.e., the overlapping boards of directors of the largest companies in the United States) are examples from sociology.

Q.2. Write short note on history of social network analysis.

Ans. Social network analysis (SNA) emerged from the social sciences branch as a very useful method for studying why and how social groups operate, behave, and interact in certain ways. A social network is comprised of individuals or organizations connected by kinship, friendship, beliefs, common interests, and financial exchange and many other things. These social relationships are considered as a graph where the entities form the nodes and the edges are the connections between them. Social networks operate at many levels, from within the families to nation-wide and also across nations and play a crucial role in understanding the behaviour of entities within the network.

George Simmel, writing at the turn of twentieth century, was the first scholar to directly think in the terms of social network. His essays focused on the nature of network size on interaction and to the likelihood of interaction in loosely-coupled networks rather than groups.

In the early twentieth century, three main traditions in social network appeared. J.L Moreno pioneered the systematic recording and analysis of social relationships in small groups, namely in work groups and classrooms.

Meanwhile, a Harvard group led by W. Lloyd Warner and Elton Mayo explored interpersonal links at work. In 1940, A.R. Radcliffe Brown's presidential address to British anthropologists stressed on the systematic study of networks. In the 1960s-1970s, an increasing number of researchers worked to combine different traditions and tracks. Significant independent work was also done by scholars from various universities like University of California, University of Chicago, University of Toronto, and Michigan State University. They critiqued methodological individualism and group-based analyses, stating that viewing the world as social networks offered more analytic advantage.

Social network analysis is an analytical tool. It just provides users with a starting point for areas that require further analysis. The SNA can be effective when used as a value addition to human analytical discretion, but as it is can be incomplete and may not have considered the wider context of these networks.

Q.3. Describe the essential characteristics of a social network.

Ans. The essential characteristics of a social network are as follows –

(i) There is a collection of entities that participate in the network. Typically, these entities are people, but they could be something else entirely.

(ii) There is at least one relationship between entities of the network. On Facebook or its ilk, this relationship is called friends. Sometimes the relationship is all-or-nothing; two people are either friends or they are not. However, in other examples of social networks, the relationship has a degree. This degree could be discrete, e.g., friends, family, acquaintances, or none as in Google+. It could be a real number, an example would be the fraction of the average day that two people spend talking to each other.

(iii) There is an assumption of nonrandomness or locality. This condition is the hardest to formalize, but the intuition is that relationships tend to cluster. That is, if entity A is related to both B and C, then there is a higher probability than average that B and C are related.

Q.4. Describe basic terminology of social network.

Ans. The basic terminology of social network is as follows –

(i) Betweenness – Betweenness of a node measures the number of paths that pass through each individual. This can identify the nodes which has the ability to control the flow of information between different parts of the network. These can be called as the gateway nodes. Gateway nodes channel information to most of the others in the network if they have many paths running through them. If they have a few paths running through them, still they play a powerful communication role if they exist between different clusters of the network.

(ii) Centrality – Centrality is a key term in SNA. A highly centralized network is dominated by one individual who controls the information and knowledge flow and may become a hub of communication failure. A less centralized network has no hub point of failures. So people can still pass on information even if some channels are blocked.

(iii) Degree – Degree of a node specifies the number of links to other individuals in the network. Higher the degree of a node, the more influential it is within the network.

(iv) Closeness – Closeness measures the extent to which an individual is near to all other individuals in a network either directly or indirectly. It exhibits the ability to access information through the network members.

Q.5. Give general applications of social network analysis.

Ans. The general applications of SNA are as follows –

(i) For improved customer targeting, for potential promotions based on their past purchase history.

(ii) In identifying loyal customers who are vocal, active and passionate and can be characterized as brand ambassadors.

(iii) In reducing average churn rates in the telecommunications industry by identifying central connectors and offering special rewards or customized experiences.

(iv) In combating terrorist activities by characterizing the network organizations to determine the likelihood and impact of terrorist activity.

(v) In detecting health care fraud by detecting patterns, establishing linkage between individuals, and to connect non-obvious relationships.

Q.6. Write short note on social network as a graph.

Ans. A social network is conceptualized as a graph, that is, a set of vertices (or nodes, units, points) representing social entities or objects and a set of lines representing one or more social relations among them. A network, however, is more than a graph because it contains additional information on the vertices and lines.

Formally, a network N can be defined as $N = (U, L, F_U, F_L)$ containing a graph $G = (U, L)$, which is an ordered pair of a unit or vertex set U and a line set L, extended with a function F_U specifying a vector of properties of the units ($f: U \rightarrow X$) and a function F_L specifying a vector of properties of the lines ($f: L \rightarrow Y$). The set of lines L may be regarded as the union of a set of undirected edges E and a set of directed arcs $A (L = E \cup A)$. Each element e of E (each edge) is an unordered pair of units u and v (vertices) from U, i.e. e(u: v), and each element a of A (each arc) is an ordered pair of units u and v (vertices) from U, i.e. a(u: v).

Based on the contents of the nodes network can be divided into two major types as follows –

(i) Social and Economic Network – It consists of a group of people connected with some sort of interactions or pattern of communication.

e.g. – Facebook, Twitter, business relation between companies and clients, interrelationship between families involved in a marriage etc.

(ii) Information Network – The connection between information objects.

e.g. – Semantic (links between various words and symbols), World Wide Web (link between various web pages; new page connecting to another through hyperlinks).

Q.7. Describe the varieties of social networks.

Ans. There are many varieties of social networks other than "friends" networks, such as –

(i) Telephone Networks – In these networks the nodes represent phone numbers, which are really individuals. There is an edge between two nodes if a call has been placed between those phones in some fixed period of time, such as last month, or "ever". The edges could be weighted by the number of calls made between these phones during the period. Communities in a telephone network will form from groups of people that communicate frequently. For example groups of friends, members of a club, or people working at the same company.

(ii) Email Networks – In these networks the nodes represent email addresses, which are again individuals. An edge represents the fact that there was at least one email in at least one direction between the two addresses. Alternatively, we may only place an edge if there were emails in both directions. In that way, we avoid viewing spammers as "friends" with all their victims. Another approach is to label edges as weak or strong. Strong edges represent communication in both directions, while weak edges indicate that the communication was in one direction only. The communities seen in email networks come from the same sorts of groupings as in case of telephone networks. A similar sort of network involves people who text other people through their cell phones.

(iii) Collaboration Networks – Here nodes represent individuals who have published research papers. There is an edge between two individuals who published one or more papers jointly. Optionally, we can label edges by

the number of joint publications. The communities in this network are authors working on a particular topic.

An alternative view of the same data is as a graph in which the nodes are papers. Two papers are connected by an edge if they have at least one author in common. Now, we form communities that are collections of papers on the same topic.

There are several other kinds of data that form two networks in a similar way. For example, we can look at the people who edit Wikipedia articles and the articles that they edit. Two editors are connected if they have edited an article in common. The communities are groups of editors that are interested in the same subject. Dually, we can build a network of articles, and connect articles if they have been edited by the same person. Here, we get communities of articles on similar or related subjects.

Q.8. How can we use social network analysis in health system ?

Ans. Social networks have a tremendous influence on the health behaviour of individuals. The results from social network analysis can be used by the government for designing health plans, benefits and to take preventive measures during some disease outbreaks. Pharmaceutical companies can target demographic groups and specific markets. Health insurance companies can design their insurance plans in a better way.

The health behaviour of people in a specific location is analyzed. One of the two graph search algorithms namely Breadth First Search (BFS) and Depth First Search (DFS) can be used to get the sub graph L(G) of a specified location from the main graph S(G). The graph L(G) can further be narrowed down to extract the sub graphs of health communities (HC(G)), health blogs (HB(G)), and users in the location (U(G)). Fig. 5.2 depicts the above process.

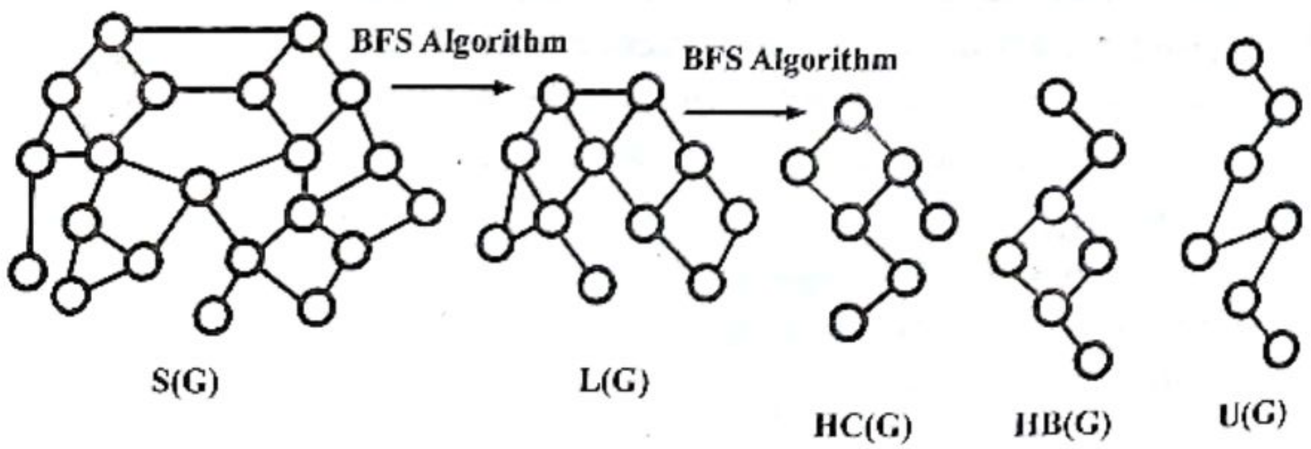

Fig. 5.2 Narrowing Down the Initial Graph using Breadth First Search Algorithm

Traversing these graphs using web based crawler, number of users belonging to various categories were identified and categorized under Healthy Behaviour, Risk Behaviour, and Preventive Health Behaviour and percentage of people under each category in that specific location is calculated. These results can be used to design targeted policy plans for different types of profiles – healthy, unhealthy etc. They can also be used by NGOs to identify the locations where "Health Awareness Programs" could be planned and conducted. They will also be able to target the right audience for preventive health programs.

Q.9. Discuss in brief various types of social networks.

Ans. The different types of social networks are as follows –

(i) One Mode Networks – One mode network considers relations only among a single set of similar nodes. For example, if we use demographic data for user-oriented nodes and combine similar nodes based on this demographic data into a set, this will be treated as a one mode network. Mostly, networks are considered as one mode networks with one set of nodes that are similar to each other. Fig. 5.3 (a) illustrates a one mode network consisting of a group of scientists that share similar features or characteristics.

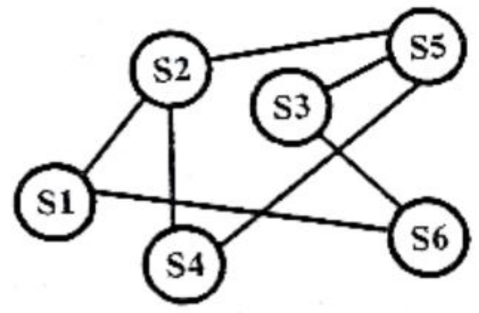

Fig. 5.3(a) One Mode Network Consisting of a Cluster of Scientists

(ii) Two Mode Networks – It considers relations among two different set of nodes. In community detection a node or actor may have strong connection with a set of nodes that form a community, but need to be analyzed with other sets of nodes with which it is considered to have weak ties. One classic example of two mode network is the scientific collaboration network, in which the two sets of nodes are scientists and papers. A scientist is connected to a paper only if he/she is an author of that paper. Fig. 5.3 (b) illustrates a two mode network consisting of a group of scientists that share some common papers being the authors of that paper.

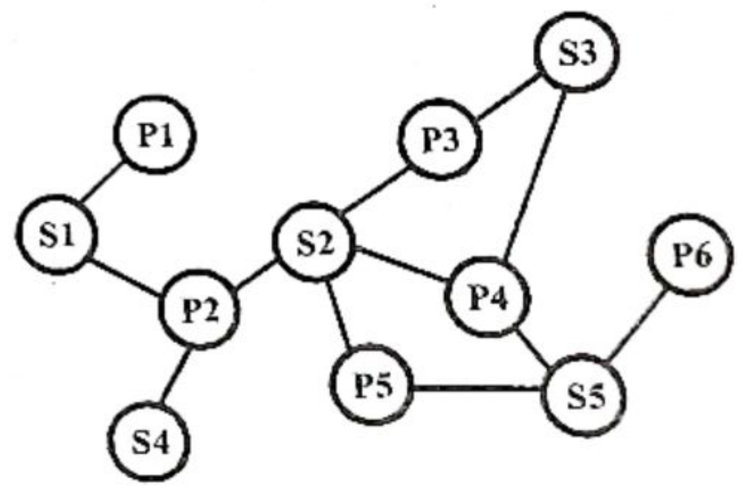

Fig. 5.3 (b) Two Mode Network Consisting of a Cluster of Scientists

(iii) Complete/Whole Networks – These networks comprises of a network that contains associations among members of a single, enclosed

community of nodes. For example, if we consider the relational ties among all of the experts of a social network for a particular topic, it is considered as a complete network. This kind of network focuses on the pattern of connections in the network as a whole rather than on individual nodes. Fig. 5.3 (c) illustrates a complete network consisting of a group of nodes U1 to U10 and the connections among all the nodes as a whole.

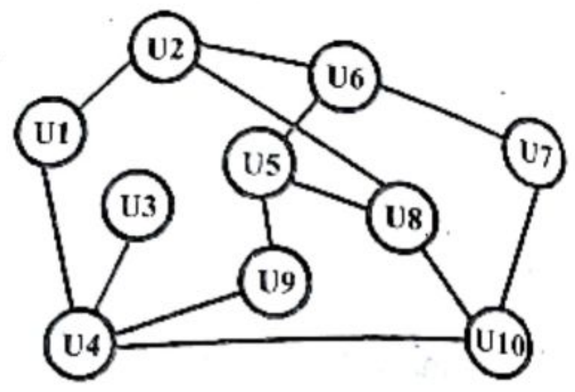

Fig. 5.3 (c) Complete Network Consisting of a Cluster of Ten Nodes

(iv) Ego Networks – Ego or ego-centric networks, comprises of a network that concentrates mainly on the connections directly associated with the focal actor (called as ego). For example, if we initially choose few nodes from a social network that can be considered as trusted nodes, these nodes will be served as egos in the network. Then, the egos of the network can be further studied to generate more trusted nodes in the social network. Ego networks are considered to be homophilous, i.e.,ego nodes will have the strongest ties with those nodes that are similar to ego nodes in terms of key attributes such as, age, gender, political views, occupation, etc. Fig. 5.3 (d) illustrates an ego network consisting of an ego node E and all other nodes (A1 to A5) referred to as alters, who are connected to ego node E. The dashed lines indicate the connections between alter nodes.

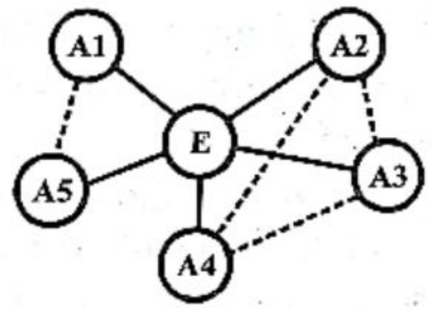

Fig. 5.3 (d) Ego Network Consisting of an Ego Node E and a Cluster of Alters A1 to A5

CLUSTERING OF SOCIAL GRAPHS, DIRECT DISCOVERY OF COMMUNITIES IN A SOCIAL GRAPH, INTRODUCTION TO RECOMMENDER SYSTEM

Q.10. Discuss the clustering of social network graphs.

Ans. An important aspect of social networks is that they contain communities of entities that are connected by many edges. These typically correspond to groups of friends at school or groups of researchers interested in the same topic. Let us consider clustering of the graph as a way to identify communities. The first step is to define distance measures for social network graph and then apply standard clustering methods.

Distance Measures for Social Network Graphs – Before applying standard clustering techniques to a social network graph, the first step is to define a distance measure. When the edges of the graph have labels, these labels might be usable as a distance measure, depending on what they represented. But when the edges are unlabeled, as in a "friends" graph, there is not much we can do to define a suitable distance.

First instinct is to assume that nodes are close if they have an edge between them and distant if not. Thus, we could say that the distance d(m, n) is 0, if there is an edge (m, n) and 1, if there is no such edge. We could use any other two values, such as 1 and ∞, as long as the distance is closer when there is an edge.

Neither of these two-valued "distance measures" – 0 and 1 or 1 and ∞ – is a true distance measure. The reason is that they violate the triangle inequality when there are three nodes, with two edges between them. That is, if there are edges (X, Y) and (Y, Z), but no edge (X, Z), then the distance from X to Z exceeds the sum of the distances from X to Y to Z. We could fix this problem by using, say, distance 1 for an edge and distance 1.5 for a missing edge.

Applying Standard Clustering Methods – There are two general approaches to clustering. The first is hierarchical (agglomerative) and second is point-assignment.

Hierarchical clustering of a social network graph starts by combining some two nodes that are connected by an edge. Successively, edges that are not between two nodes of the same cluster would be chosen randomly to combine the clusters to which their two nodes belong. The choices would be random, because all distances represented by an edge are the same.

For Example – Consider the graph of fig. 5.4. First, let us specify the communities. At the highest level, it appears that there are two communities {X, Y, Z} and {M, N, O, P}. However, we could also view {M, N, O} and {M, O, P} as two subcommunities of {M, N, O, P}. These two subcommunities overlap in two of their members, and thus could never be identified by a pure clustering algorithm. Finally, we could consider each pair of individuals that are connected by an edge as a community of size 2, although such communities are uninteresting.

The problem with hierarchical clustering of a graph like that of fig. 5.4 is that at some point we are likely to chose to combine Y and M, even though they surely belong in different clusters. The reason we are likely to combine Y and M is that M, and any cluster containing it, is

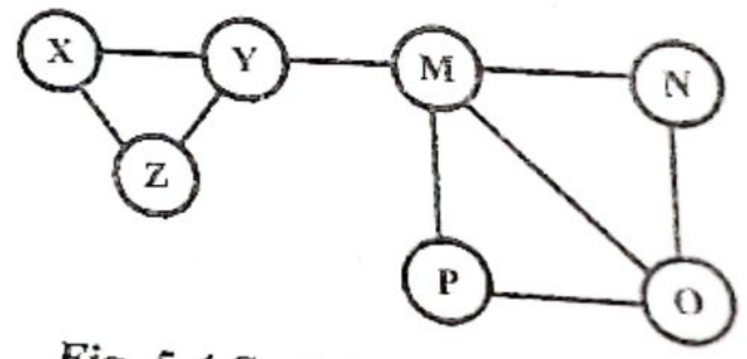

Fig. 5.4 Social Network Graph

as close to Y and any cluster containing it, as X and Z are to Y. There is even a 1/9 probability that the first thing we do is to combine Y and M into one cluster.

To reduce the probability of error, we can run hierarchical clustering several times and pick the run that gives the most coherent clusters. In a large graph with many comm..nities there is a significant chance that in the initial phases we shall use some edges that connect two nodes that do not belong together in any large community.

The point-assignment approach to clustering social networks is also based upon the fact that all edges which are at the same distance will introduce a number of random factors that will lead to some nodes being assigned to the wrong cluster.

Q.11. Discuss the direct discovery of communities in a social graph.

Ans. Searching communities by partitioning all the individuals is relatively efficient, however it does have several limitations. It is not possible to place an individual in two different communities, and everyone is assigned to a community. Let us consider a technique for discovering communities directly by looking for subsets of the nodes that have a relatively large number of edges among them.

Finding Cliques – To find sets of nodes with many edges between them we have to start by finding a large clique (a set of nodes with edges between any two of them). However, that task is not easy. Not only is finding maximal cliques NP-complete, but it is among the hardest of the NP-complete problems in the sense that even approximating the maximal clique is hard. Further, it is possible to have a set of nodes with almost all edges between them, and yet have only relatively small cliques. For example, suppose our graph has nodes numbered 1, 2,....., n and there is an edge between two nodes i and j unless i and j have the same remainder when divided by k. Then the fraction of possible edges that are actually present is approximately (k – 1)/k. There are many cliques of size k, of which {1, 2, ..., k} is but one example.

Yet there are no cliques larger than k. To see why, observe that any set of k + 1 nodes has two that leave the same remainder when divided by k. This point is an application of the "pigeonhole principle". Since there are only k different remainders possible, we cannot have distinct remainders for each of k + 1 nodes. Thus, no set of k 1 nodes can be a clique in this graph.

Complete Bipartite Graphs – A complete bipartite graph consists of s nodes on one side and t nodes on the other side, with all st possible edges between the nodes of one side and the other present. We denote this graph by $K_{s,t}$. We should draw an analogy between complete bipartite graphs as subgraphs of general bipartite graphs and cliques as subgraphs of general graphs. In fact, a clique of s nodes is often referred to as a complete graph and denoted as K_s, while a complete bipartite subgraph is sometimes called a bi-clique.

From above example it is clear that it is not possible to guarantee that a graph with many edges necessarily has a large clique, it is possible to guarantee that a bipartite graph with many edges has a large complete bipartite subgraph. We can consider a complete bipartite subgraph (or a clique if we discovered a large one) as the nucleus of a community and add to it nodes with many edges to existing members of the community. If the graph itself is k-partite then we can take nodes of two types and the edges between them to form a bipartite graph. In this bipartite graph, we can search for complete bipartite subgraphs as the nuclei of communities.

However, we can also use complete bipartite subgraphs for community finding in ordinary graphs where nodes all have the same type. Divide the nodes into two equal groups at random. If a community exists, then we would expect about half its nodes to fall into each group, and we would expect that about half its edges would go between groups. Thus, we still have a reasonable chance of identifying a large complete bipartite subgraph in the community. To this nucleus we can add nodes from either of the two groups, if they have edges to many of the nodes already identified as belonging to the community.

Q.12. How to use betweenness to find communities ? Explain with example.

Ans. The betweenness scores for the edges of a graph behave something like a distance measure on the nodes of the graph. It is not exactly a distance measure, because it is not defined for pairs of nodes that are unconnected by an edge, and might not satisfy the triangle inequality even when defined. However, we can cluster by taking the edges in order of increasing betweeness and add them to the graph one at a time. At each step, the connected components of the graph form some clusters. The higher the betweenness we allow, the more edges we get, and the larger the clusters become.

More commonly, this idea is expressed as a process of edge removal. Start with the graph and all its edges; then remove edges with the highest betweeness, until the graph has broken into a suitable number of connected components.

For example – Let us consider the graph of fig. 5.5. We see it with the betweenness for each edge in fig. 5.5. The calculation of the betweenness can be done in usual manner. The only tricky part of the count is to observe that

between N and O there are two shortest paths, one going through M and the other through P. Thus, each of the edges (M, N), (N, P), (M, O), and (O, P) are credited with half a shortest path.

Clearly, edge (Y, M) has the highest betweenness, so it is removed first. That leaves us with exactly the communities we observed make the most sense, namely: {X, Y, Z} and {M, N, P, O}. However, we can continue to remove edges. Next to leave are (X, Y) and (Y, Z) with a score of 6, followed by (M, N) and (M, O) with a score of 5.5. Then, (M, P), whose score is 5, would leave the graph. Remaining graph is shown in fig. 5.6.

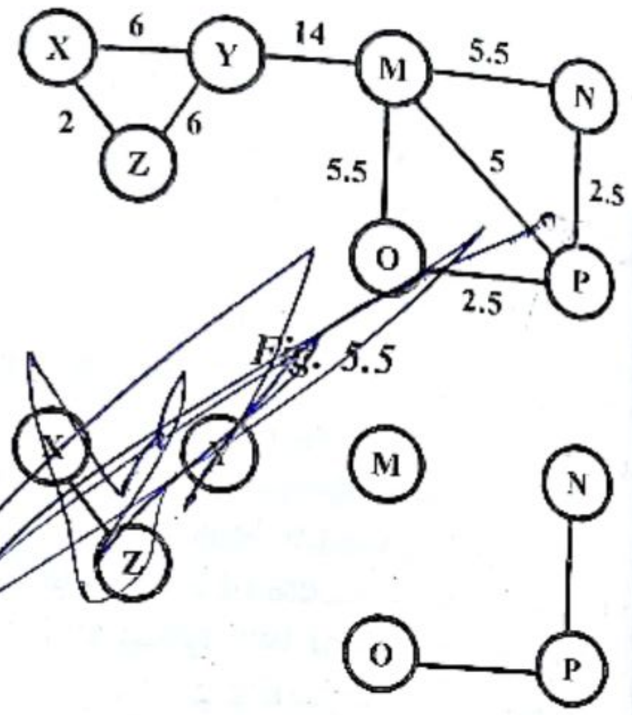

Fig. 5.5

Fig. 5.6 All the Edges with Betweenness 5 or More have been Removed

The "communities" of fig. 5.6 look strange. One implication is that X and Z are more closely knit to each other than to Y. That is, in some sense Y is a "traitor" to the community {X, Y, Z} because he has a friend M outside that community. Likewise, M can be seen as a "traitor" to the group {M, N, P, O}, which is why in fig. 5.6, only N, P, and O remain connected.

Q.13. How to find complete bipartite subgraphs ? Explain

Ans. Suppose we are given a large bipartite graph G, and we want to find instances of $K_{s,t}$ within it. It is possible to view the problem of finding instances of $K_{s,t}$ within G as one of finding frequent itemsets. For this purpose, let the "items" be the nodes on one side of G, which we shall call the left side. We assume that the instance of $K_{s,t}$ we are looking for has t nodes on the left side, and we shall also assume for efficiency that $t \leq s$. The "baskets" correspond to the nodes on the other side of G (the right side). The members of the basket for node v are the nodes of the left side to which v is connected. Finally, let the support threshold be s, the number of nodes that the instance of $K_{s,t}$ has on the right side.

We can now state the problem of finding instances of $K_{s,t}$ as that of finding frequent itemsets F of size t. That is, if a set of t nodes on the left side is frequent, then they all occur together in at least s baskets. But the baskets are the nodes on the right side. Each basket corresponds to a node that is connected to all t of the nodes in F. Thus, the frequent itemset of size t and s of the baskets in which all those items appear form an instance of $K_{s,t}$.

It is important to understand that we do not mean a generated subgraph one formed by selecting some nodes and including all edges. In this context, we only require that there be edges between any pair of nodes on different sides. It is also possible that some nodes on the same side are connected by edges as well.

For example – Let us consider the bipartite graph of fig. 5.7. The left side is the nodes {1, 2, 3, 4} and the right side is {m, n, o, p}. The latter are the baskets, so basket m consists of "items" 1 and 4; that is, m = {1, 4}. Similarly, n = {2, 3}, o = {1} and p = {3}.

If s = 2 and t = 1, we must find itemsets of size 1 that appear in at least two baskets. {1} is one such itemset, and {3} is another. However, in this tiny example there are no itemsets for larger, more interesting values of s and t, such as s = t = 2.

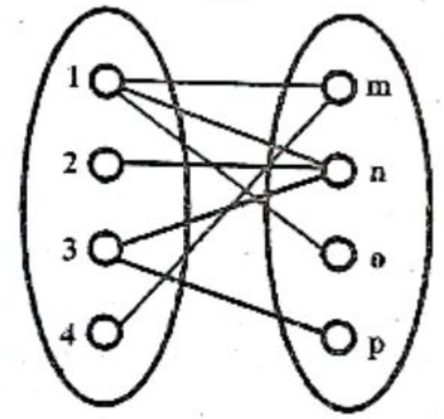

Fig. 5.7 The Bipartite Graph

Q.14. Discuss about the community or group detection.

Ans. Community or group detection in online social networks is based on studying the social network structure to find individual nodes in the network that correlate more with each other than with other related group of users. Discovery of such communities lead to intra-communities (users or nodes belonging to same community) that are more likely to be connected or associated compared to inter-communities (users or nodes belonging to different community). Such kind of clustering in groups helps to further make estimation about the users in the network, regarding his/her likes and interests, tastes and future activities. This is turn, will help in assessing the probability of which products he/she would buy, which songs or movies he/she would watch, which services he/she may be interested in, and so on.

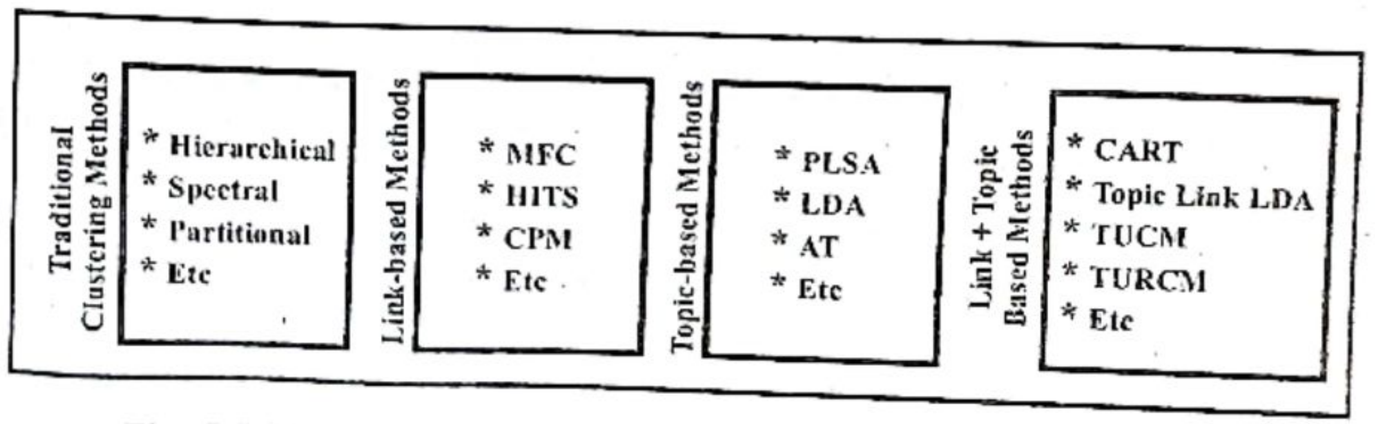

Fig. 5.8 Various Categories of Community Detection Methods

Detecting communities is not only restricted to social networks but also is of great significance in various other fields such as politics, economics,

biology, sociology and information networks where systems can be represented as graphs. However, detection of communities in vast, dynamic and complex networks is a challenging problem and has caught the attention of many researchers working in this area of SNAM.

Fig. 5.8 shows that there are several approaches that can be followed for community detection in social networks, namely the traditional clustering methods, the link-based methods, the topic-based methods and the topic-link based methods. A brief description of these community detection methods is given below –

(i) Traditional Clustering Methods – The hierarchical method of clustering constructs a group of nested clusters by progressively merging or splitting the clusters. Two well-known hierarchical clustering methods are Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) and Clustering Using Representatives (CURE). Partitioning clustering algorithms partitions the entire nodes into 'n' clusters, in which the value of 'n' is known in advance. Spectral clustering makes use of the spectral properties of the similarity matrix to construct clusters of communities.

(ii) Link-based Methods – The link-based community detection methods study the links or edges of the network to detect communities. This method, however, can only reflect the strength of connections and not the semantics such as the interesting topics shared by people. Two common link-based community detection methods are Hyperlink Induced Topic Search (HITS) and Maximum Flow Community (MFC).

(iii) Topic-based Methods – The topic-based community detection methods generate communities which are topically similar. This method, however, do not concentrate on nodes that may share some explicit communication via links. Two common link-based community detection methods are Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet allocation(LDA).

(iv) Topic-link Based Methods – The topic-link based community detection approach is the most recent approach used for detecting communities in social networks. This hybrid approach is a combination of link-based methods and topic-based methods that considers the disadvantages of using only one single method for community detection. Two common link-based community detection methods are Community-Author-Recipient-Topic (CART) and Topic-Link LDA.

Q.15. What do you understand by graph partitioning ?

Ans. Graphs are also used by various scientists or researchers for modeling an application program. Partitioning a graph is purely an algorithmic problem.

It helps to reduce complexity of big graphs and also introduces parallelizations. Graph partitioning is required in various application problems like social networks, road networks, air traffic controls, image analysis etc. Some applications of graph portioning problems are scientific computing, partitioning various stages of VLSI design circuit, task scheduling in multiprocessor systems etc. The aim of graph partitioning is to divide the nodes into several disjoint parts such that the predefined objective function is minimal. The optimal graph partitioning is NP-complete however various approximate algorithms are made to solve the problems.

Graph partitioning is divided into two groups –

(i) Constrained Partitioning – In this partitioning the parts are of equal size.

(ii) Unconstrained Partitioning – In this partitioning the parts are of different size.

Various algorithms are available for graph partitioning. Among them three principle algorithms are Geometric partitioning, Spectral partitioning and Multi-level graph partitioning.

In geometric partitioning the graph is bisected by utilizing those coordinates which are obtained if nodes of a graph are available in space. In this partition the vertices which are spatially near to each other are taken into one cluster. In spectral partitioning the associability of the graph is concluded by finding the Eigen vectors with respect to the second smallest Eigen value of Laplacian matrix L corresponding to graph. This bisection method is really demanding but not feasible for large graphs.

Multi-level partitioning is highly effective than the classical graph partitioning methods. Constrained graph partitioning problems are efficiently solved by this method. The main idea is to partition the large graphs into k – parts, group the vertices together in a group and deal with this group of vertices rather than independent vertices. It has three phases – coarsening phase, initial partitioning phase, and partition refinement phase.

Q.16. Explain the term structural analysis.

Ans. Graph is a mathematical structure which shows relation between some objects. Graph is made up of vertices, nodes, or points which are connected using lines, arcs or edges. A graph may be directed or undirected. A graph represented as G = (V, E) where V is set of vertices and E is set of edges. Graph is represented using different data structures like adjacency list, adjacency matrices, incidence matrices etc. In a social network the node represent individuals or organizations and the edges represent the relationship between individuals or organizations. Social network provides set of methods for analyzing the structure

of whole social entities. There are different levels of analysis that are not necessarily mutually exclusive. Generally there are three levels –

(i) Micro Level – At this level it typically begins with an individual or may begin with a small group of individuals.

(ii) Meso Level – It falls between micro and macro-level. It shows the connection between the micro and macro level. Meso-level networks are low density network.

(iii) Macro Level – These are large scale networks that traces the interactions over large population. These networks are more complex.

Q.17. What is recommender system ?

Ans. Recommender Systems (RS) provide recommendations to users about a set of articles or services they might be interested in. This facility in online social networks (OSNs) has become very popular due to the easy access of information on the internet. A few important applications of RS are its use in several e-commerce sites, such as Amazon, Flipkart and Firstcry, for recommendation of items such as movies, books, gadgets, and jewellery. The data required for providing recommendations can be obtained explicitly based on users' ratings or comments, or implicitly by monitoring user activities, such as items checked, books bought, audios heard, web sites visited, and so on. RS may also use demographic information of users like occupation, gender or age for clustering group of users that may have similar linkings. Fig. 5.9 shows a generic RS framework that takes the user profile as input, the item profile and/or the user-item rating matrix. These inputs are fed to a recommender system engine which computes and predicts the top-N recommendations for a user.

It is a challenge for every e-commerce site to correlate consumers with the most suitable products which, in turn, enhance customer satisfaction and loyalty. Hence, the majority of the e-commerce sites have become interested in RS, which provide personalized recommendations to each visited user of a site by examining patterns of user interests in products that suggests a user's taste.

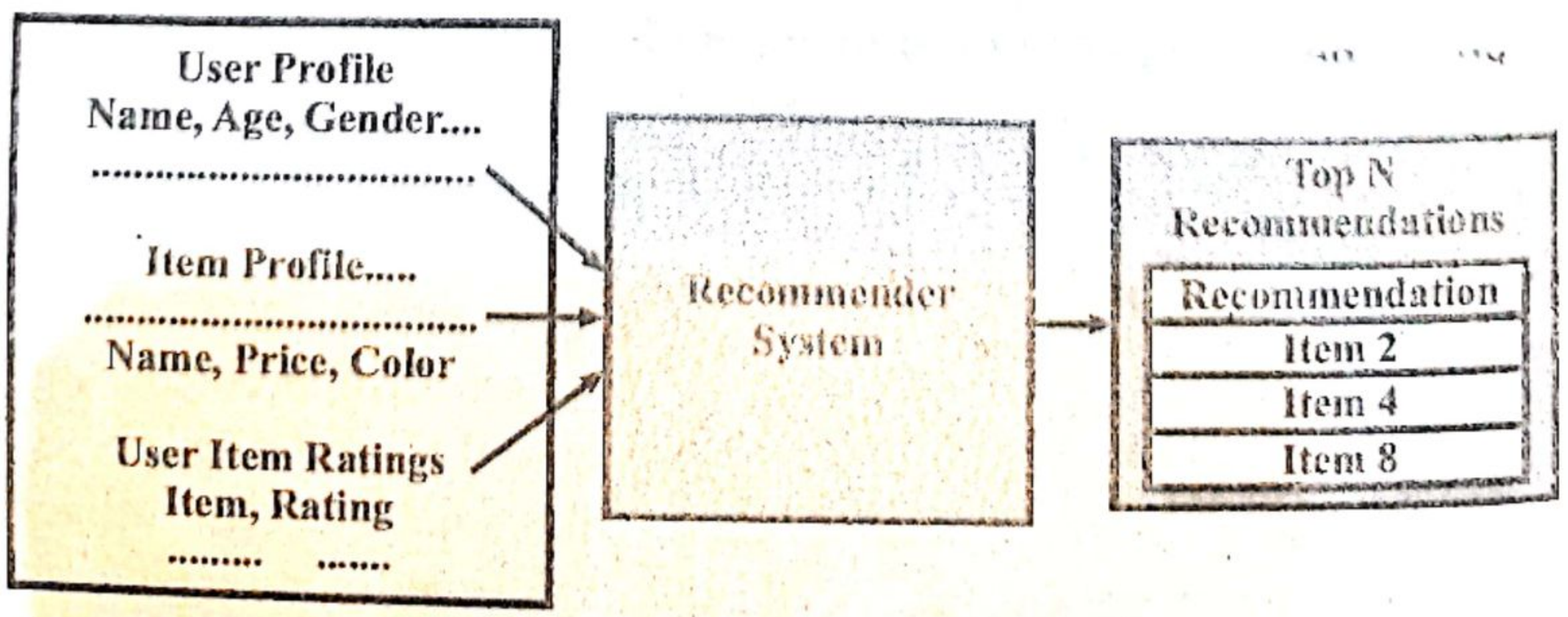

Fig. 5.9 The Generic Recommender System Framework