CONTENTS

Sl. No.	DETAILS		Pg. No.
1	INTRODUCTION TO BIG DATA	...	A.1-A.3
2	HADOOP AND MAP REDUCE	...	B.1-B.3
3	HDFS (HADOOP DISTRIBUTED FILE SYSTEM)	...	C.1-C.2
4	(HADOOP ECO SYSTEM AND YARN, NOSQL, SPARK AND SCALA)	...	D.1-D.3
5	(HADOOP ECO SYSTEM FRAMEWORK, PIG, HIVE AND HBASE)	...	E.1-E.2
6	MODEL PAPER	...	F.1-F.2

❑ Publishers : **KAMAL PUBLISHING HOUSE**
110/201, Laxmi Tower,
Near R.K. Mission School & Hospital,
R.K. Nagar, G.T. Road
Kanpur – 208 012
(Uttar Pradesh)
Contact No. : +91 9559401840
Email : kamalpubhouse@gmail.com
Website : www.kphindia.in

❑ *Composed by* : Komal Graphics
Kanpur

SYLLABUS

	Topic
1.	**Introduction to Big Data** : Types of digital data, history of Big Data innovation, introduction to Big Data platform, drivers for Big Data, Big Data architecture and characteristics, 5 Vs of Big Data, Big Data technology components, Big Data importance and applications, Big Data features – security, compliance, auditing and protection, Big Data privacy and ethics, Big Data Analytics, Challenges of conventional systems, intelligent data analysis, nature of data, analytic processes and tools, analysis vs reporting, modern data analytic tools.
2.	**Hadoop** : History of Hadoop, Apache Hadoop, the Hadoop Distributed File System, components of Hadoop, data format, analyzing data with Hadoop, scaling out, Hadoop streaming, Hadoop pipes, Hadoop Echo System. **Map-Reduce** : Map-Reduce framework and basics, how Map Reduce works, developing a Map Reduce application, unit tests with MR unit, test data and local tests, anatomy of a Map Reduce job run, failures, job scheduling, shuffle and sort, task execution, Map Reduce types, input formats, output formats, Map Reduce features, Real-world Map Reduce
3.	**HDFS (Hadoop Distributed File System)** : Design of HDFS, HDFS concepts, benefits and challenges, file sizes, block sizes and block abstraction in HDFS, data replication, how does HDFS store, read, and write files, Java interfaces to HDFS, command line interface, Hadoop file system interfaces, data flow, data ingest with Flume and Scoop, Hadoop archives, Hadoop I/O: Compression, serialization, Avro and file-based data structures. Hadoop Environment: Setting up a Hadoop cluster, cluster specification, cluster setup and installation, Hadoop configuration, security in Hadoop, administering Hadoop, HDFS monitoring & maintenance, Hadoop benchmarks, Hadoop in the cloud
4.	**Hadoop Eco System and YARN** : Hadoop ecosystem components, schedulers, fair and capacity, Hadoop 2.0 New Features – Name Node high availability, HDFS federation, MRv2, YARN, Running MRv1 in YARN. **NoSQL Databases** : Introduction to NoSQL MongoDB: Introduction, data types, creating, updating and deleing documents, querying, introduction to indexing, capped collections **Spark** : Installing spark, spark applications, jobs, stages and tasks, Resilient Distributed Databases, anatomy of a Spark job run, Spark on YARN **SCALA** : Introduction, classes and objects, basic types and operators, built-in control structures, functions and closures, inheritance.
5.	**Hadoop Eco System Frameworks** : Applications on Big Data using Pig, Hive and HBase **Pig** : Introduction to PIG, Execution Modes of Pig, Comparison of Pig with Databases, Grunt, Pig Latin, User Defined Functions, Data Processing operators, **Hive** : Apache Hive architecture and installation, Hive shell, Hive services, Hive metastore, comparison with traditional databases, HiveQL, tables, querying data and user defined functions, sorting and aggregating, Map Reduce scripts, joins & subqueries.

INTRODUCTION TO BIG DATA

1. *What is Big data? Define it in terms of volume and velocity* **(2018-2019)**

Big Data

Big data is data that contains greater variety, arriving in increasing volumes and with more velocity. Big data is large, more complex data sets, especially from new data sources. These data sets are so voluminous that traditional data processing software just can't manage them.

Volume : Volume refers to the quantity of data to be stored. For example, Walmart deals with big data. They handle more than 1 million customer transactions every hour.

Velocity : Velocity refers to the speed with which data is generated. For example data that is generated with high velocity would be Twitter messages or Facebook posts.

2. *Discuss about the three dimensions of Big data.* **(2016-2017)**

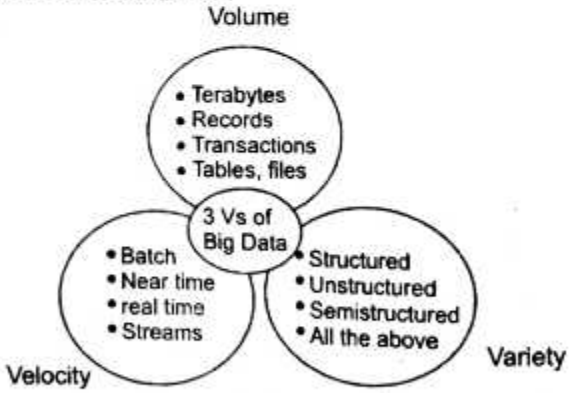
There are three dimensions of Big Data named –Volume, Velocity and Variety. It is also refered as 3 V's of Big Data.

Volume : Volume is the V most associated with big data because, well, volume can be big. quantities of data that reach almost incomprehensible proportions. Within the Social Media space for example, Volume refers to the amount of data generated through websites, portals and online applications. Especially for B2C companies, Volume encompasses the available data that are out there and need to be assessed for relevance. Consider the following - Facebook has 2 billion users, Youtube 1 billion users, Twitter 350 million users and Instagram 700 million users. Every day, these users contri bute to billions of images,

posts, videos, tweets etc. You can now imagine the insanely large amount -or Volume- of data that is generated every minute and every hour.

Velocity : With Velocity we refer to the speed with which data are being generated. Staying with our social media example, every day 900 million photos are uploaded on Facebook, 500 million tweets are posted on Twitter, 0.4 million hours of video are uploaded on Youtube and 3.5 billion searches are performed in Google. This is like a nuclear data explosion. Big Data helps the company to hold this explosion, accept the incoming flow of data and at the same time process it fast so that it does not create bottlenecks.

Variety : Variety in Big Data refers to all the structured and unstructured data that has the possibility of getting generated either by humans or by machines. The most commonly added data are structured -texts, tweets, pictures & videos. However, unstructured data like emails, voicemails, hand-written text, ECG reading, audio recordings etc, are also important elements under Variety. Variety is all about the ability to classify the incoming data into various categories.



(Figure)

3. *How Big data analytics can be useful in development of smart cities?* **(2016-2017)**

Big data offer the potential for cities to obtain valuable insights from a large amount of data collected through various sources. Big data systems are stored, processed, and mined in smart cities **efficiently to produce information to enhance different smart city services**. In addition, Big data can help decision makers plan for any expansion in smart city services, resources, or areas. These new challenges focus primarily on problems related to business and technology that enable cities to actualize the vision, principles, and requirements of the applications of smart cities by realizing the main smart environment characteristics. There are some fields where Big data analytics can be useful in developing the smart city.

Water Management : Real-time analysis and sensors can help detect the flow of water, pollution level, predict scarcity on the basis of usage, reduce areas of leakage, sewage overflow, etc.

In the Vietnamese city of Da Nang, sensors help keep track of the water's quality: its salinity, pH, etc., giving notification alerts in case there has been an unusual change. This eliminates the need to collect information manually and test the water quality, which would be a time-consuming process.

Transport Management : Big data can play an extremely important role in transport management, which is a major problem faced by all major cities across the world. Information on public buses and other modes of transport on your mobile phone. You might have sensors attached to cars which will indicate nearest parking lots.

For example, the Vietnamese city of Da Nang has teamed up with IBM, to ensure that economic development does not come at the price of the environment. The plan is to manage the traffic congestion with the help of sensors embedded on the roads and public transport, which can help them make the best out of available resources.

Controlling Pollution : Tracking illegal factories, or those releasing a heavy and unacceptable amount of toxic substance can become easier.

For example, currently, Beijing is facing a problem of acute air pollution. IBM has teamed with the Government for its project called 'Green Horizon' to combat this

problem. Big data can be helpful to identify the sources of pollution, its quantity; which will be of great help to tackle the issue. It can also be used to predict the occurrences of smog. By enhancing the information available and forecasting abilities, finding a solution will become much easier.

Garbage Management : An example can be: garbage bins might be placed with sensors that will be connected with garbage disposal centres. Perhaps, garbage cans might come with a feature that indicates when a garbage can have become full, which will, in turn, notify the trash can collectors, who can then predict and decide which route to focus and map their path

Law Enforcement & Security : Devices will help you to track high danger zones, make it easier for civilians to report crimes to security agencies. Resolving citizen's queries has become much easier since information is available at the tips of these agencies.

For example of Toronto, a smart city. It has now teamed up with a Chicago based firm "City Zenith", which is helping the city process raw data available into useful insights. This move has been made to make data available easily to the government to take appropriate steps for infrastructure, transportation, etc.

4. *What are the benefits of Big data.* **(2016-17)**

There is no denying the fact that in less than a decade, Big Data becomes a multi-billion-dollar industry. Today, Big data revolution has arrived with the growth of the internet, wireless networks, smartphones, social media and other technologies. It is beneficial for both big and small businesses. Some Key benefits are explained here.

(1) **Big Data to Improve Strategy and Pricing :** Business intelligence tools based on Big data analytics are used to evaluate finances, which helps to get a clearer picture of where your business stands. And based on that evaluation who can adopt the required strategy needed to improve pricing in business or organization.

(2) **Big Data Controls Online Reputation :** We can use Big data tools for sentimental analysis. Thus, you can use it to get feedback about your company or organization.

Feedback is like who is saying what about your company. And if you want to monitor and improve the online presence of your business, then Big data tools can be utilized for this purpose.

(3) **Advantages of Big Data in New Product Development :** Using Big data analytics, trends of customer needs and satisfaction can be analyzed. This can further help to develop a whole new product according to their requirements.

(4) **Big Data Advantages in Cost Saving :** Using Big data tools like Hadoop and Cloud-Based Analytics, cost saving in business can be done. In business, when large amounts of data is there, then these tools help to handle and maintain that data in more efficient ways.

(5) **Use Big Data to Ensure that we have Hired the Right Employees :** Recruiting companies scan candidate's resumes and LinkedIn profiles using Keywords. Keywords are such that which matches the job description. The hiring process is no longer based on how the candidate looks on paper and how they are perceived in person.

(6) **Big Data for Understanding the Market Conditions :** Better understanding of current market conditions is possible by analyzing the Big data. Let's take an example – by analyzing a customer's purchasing behavior, a company can find out the products which are sold most. It helps to analyze the trend and what customers want. Using this, a particular business can get ahead of its competitors.

(7) **Use Big Data to Increase your Efficiency :** Digital technology like Big data tools boosts your business's efficiency. From using tools such as Google Maps, Google Earth, and social media can do many

tasks right at your desk. These tools save a
amount of travel expenses and time too.

(8) **Big Data to Focus on Local Preferences** : B
data allows small businesses to focus on the lo
environment they cater to. It allows you to ha
insight into your local client's likes, dislikes a
preferences even more. Once you know yo
customer's preferences it becomes easy to compete
the market.

(9) **Big Data in Time Reduction** : High speed tools li
Hadoop and in-memory analytics can easily identi
new sources of data. As a result, this helps
immediate analysis of business data and makes qui
decisions based on the learning.

(10) **Big Data to Increase Sales and Loyalty** : B
Data helps businesses to grow digitally and to tail
their products and services to what customers exact
want. The digital footprints of business allow
customers to have insight of product and services
social media platforms. When this satisfies th
customers, they start to purchase and advertise
more and more. As a result, this makes a busine
more reliable and loyal among customers.

5. *Discuss key challenges under Big Data.*

Big data needs to be analyzed to enhance decisio
making. But, there are some challenges of Big Dat
encountered by companies. Let's take a look at some
these challenges:

(1) **Data Integration** : Normally, an organization wi
connect data from numerous sources, which makes
hard to monitor the effectiveness of the integratio
process. A lot of the problems pertaining
inaccurate insights may be tracked back all the wa
to how the data gets collected, verified, stored, an
utilized. The problem is, when working in data
sensitive and intensive industries, even the smalles
error may prove fatal to the success of the overa
process.

(2) **Data Complexity** : It is a known fact that the data flowing in is getting more complicated on a daily basis. This means that the systems have to account for a lot more factors and parameters than earlier. As the raw data flows in from various sources, such as salespeople, operations, consumers, and other users existing within the same organization, it becomes evident that the data structure is more complex than ever before. Moreover, there are lots of technologies that function across multiple channels, from offline to web to mobile, which intensifies the complexity to a whole new level and increases the challenges of big data.

(3) **Data Security** : Merely collecting and storing all the information from different sources isn't enough. The available data needs to be secured at all costs. The creation of technologies such as the cloud helps store data on a single platform so that it can be accessed anywhere, anytime, and make life easier for organizations and consumers alike. But, at the same time, storing all the vital data in one place has also given rise to several security issues. Thankfully, most of these broad-level problems may be handled if all the small steps are taken correctly and the data integration process flows smoothly without any hiccups. To achieve this sort of free flow of information and produce meaningful business insights, the accuracy of the data needs to be considered.

Even a small step in the right direction can pay off big time. Organizations must begin with the goal of working on methods to bring the disparate data sources together and find a way to harmonize the data so that it can impress clients

(4) **Data Capture** : Data capture is currently leading the charge when it comes to edge-to-core architectures found in datacenters. Not surprisingly, the new data is captured directly at the source, which might exist underneath the ocean as seen in gas and

oil exploration, or orbiting satellites as evident fro weather applications. Closer to home, the new form o Big Data is formed on your phone every time yo capture a video or image, or send out a tweet. Th amount of data obtained from the source is going t be considerably higher than what we've becom familiar with now.

(5) **Data Scale** : Data scale has been found to be responsible for data center automation. Right now, the scale of the top-tier cloud providers is such that they have no option but to invest considerably in intelligence and automation when it comes to managing all their infrastructure efficiently. As far as manual management is concerned, the process is too cost-prohibitive if the scale at which they operate is considered.

(6) **Data Mobility** : Worldwide networks are undergoing changes constantly, and most of it can be traced back to data mobility. With data being available everywhere, it needs to be moved so it can be collected and then analyzed. So, it turns out that networks came close to becoming faster than the requirements of Internet bandwidth at 40 to 100Gbps, but unfortunately, the data movement increased almost 100 times, and in some cases even 1,000 times.

(7) **Data Value** : The existing notions about storage are all being reevaluated thanks to data value. There is little doubt that data holds more value for organizations than ever before, which means that the usefulness of the data is increasing over longer periods of time thanks to the developments made in the fields of artificial intelligence-based analytics and machine learning. What this all implies is that a lot more data has to be stored for longer time periods and that the data needs to be addressable in aggregate, so analytics can have the desired effect.

(8) **Data Analytics** : When we come down to brass tacks, it becomes evident that data analytics will be the driver for future compute-intensive architectures. So, organizations need to come up with ways to drive more data so that they can later aggregate it into large data repositories, driven by the nature of analytics as well as machine learning. Such analytics are better equipped to handle questions and provide powerful solutions when introduced in various, Big Data sources. Being compute-intensive operations, both machine learning and analytics drive huge amounts of high-speed proc--ing during the analytics process on large datasets.

What's more, the compute-intensive form of analytics seems to be driving several new methods for storing and accessing data, from something as compact as in-memory databases to as large as 100 petabyte-scale object stores.

6. *What are structured, unstructured and semi structured data? Explain with suitable examples.* **(2018-19)**

Structured Data

Structured data is the easiest to explain but the most challenging to search through. Structured data is data that would be inside a database or some sort of data management application. These applications can track the usage and activity and provide versioning back to the beginning of the file's existence if managed from the start.

Database type applications such as SQL, Mongo, and Caché, to name a few of the popular ones, use an application to collect the data through various data entry points like a GUI or web-based portal. Data is added to the fields on the user interface and then inserted into various columns and rows in the database. Most websites or data entry applications will collect data into these various database formats.

Unstructured Data

Now let's look at unstructured data. Unstructur data makes up the majority of enterprise data–well ov 80%, in fact. This data is not usable in a tradition database application since single field entry is normal the mechanism to add data to the rows and columns Unstructured data types are vast; there are applicatio that can process over 1000 types of unstructured da formats.

Examples of unstructured data types include offi documents, text files, image files, PDFs, log files, an application data files like .ini or .dll. A typical user wi create and process primarily unstructured data. This is th data that Aparavi is going after.

Semi-Structured Data

Now that we understand structured vs. unstructure data, note that some data is considered semi-structure Semi-structured data is, as its name suggests, a mix structured and unstructured data. An example would be on-prem Exchange Server. Exchange stores all the emai and attachments data within its database. However, email file can be easily moved or duplicated from yo email client by simply dragging the email to the deskto This creates an .msg file and includes all attachment dat Attachments can be opened within this client and saved your local file share or desktop. Aparavi can also proce this type of data, provided the data has been exported fro the structured environment.

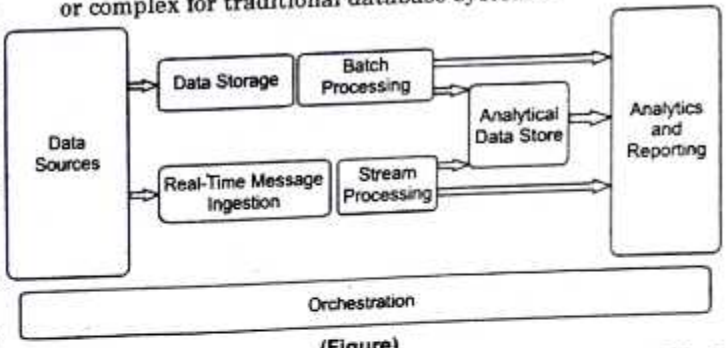
Differences between Structured, Semi-structure and Unstructured data:

Properties	Structured Data	Semi-Structured Data	Unstructure Data
Technology	It is based on Relational database table	It is based on XML/RDF(Resource Description Framework).	It is based c character a binary data
Transaction management	Matured transaction and various concurrency techniques	Transaction is adapted from DBMS not matured	No transactio management n and concurrency

Version management	Versioning over tuples,row,tables	Versioning over tuples or graph is possible	Versioned as a whole
Flexibility	It is schema dependent and less flexible	It is more flexible than structured data but less flexible than unstructured data	It is more flexible and there is absence of schema
Scalability	It is very difficult to scale DB schema	It's scaling is simpler than structured data	It is more scalable.
Robustness	Very robust	New technology, not very spread	—
Query performance	Structured query allow complex joining	Queries over anonymous nodes are possible	Only textual queries are possible

7. *Explain Architecture of Big data?*

A big data architecture is designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems.



(Figure)

Most Big Data Architectures Include Some or All of the Following Components :

(1) **Data Sources :** All big data solutions start with one or more data sources. Examples include:

- **(a)** Application data stores, such as relational databases.
- **(b)** Static files produced by applications, such as web server log files.
- **(c)** Real-time data sources, such as IoT devices.

(2) **Data Storage :** Data for batch processing operation is typically stored in a distributed file store that can hold high volumes of large files in various formats. This kind of store is often called a data lake. Options for implementing this storage include Azure Data Lake Store or blob containers in Azure Storage.

(3) **Batch Processing :** Because the data sets are so large, often a big data solution must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files, processing them, and writing the output to new files. Options include running U-SQL jobs in Azure Data Lake Analytics, using Hive, Pig, or custom Map/Reduce jobs in an HDInsight Hadoop cluster, or using Java, Scala, or Python programs in an HDInsight Spark cluster.

(4) **Real-time Message Ingestion :** If the solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing. This might be a simple data store, where incoming messages are dropped into a folder for processing. However, many solutions need a message ingestion store to act as a buffer for messages, and to support scale-out processing, reliable delivery, and other message queuing semantics. Options include Azure Event Hubs, Azure IoT Hubs, and Kafka.

(5) **Stream Processing :** After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis. The processed stream data is then written to an output sink. Azure Stream Analytics provides a managed stream processing service based on perpetually running SQL queries that operate on unbounded streams. You can also use open source Apache streaming technologies like Storm and Spark Streaming in an HD Insight cluster.

(6) **Analytical Data Store :** Many big data solutions prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools. The analytical data store used to serve these queries can be a Kimball-style relational data warehouse, as seen in most traditional business intelligence (BI) solutions. Alternatively, the data could be presented through a low-latency NoSQL technology such as HBase, or an interactive Hive database that provides a metadata abstraction over data files in the distributed data store. Azure Synapse Analytics provides a managed service for large-scale, cloud-based data warehousing. HDInsight supports Interactive Hive, HBase, and Spark SQL, which can also be used to serve data for analysis.

(7) **Analysis and Reporting :** The goal of most big data solutions is to provide insights into the data through analysis and reporting. To empower users to analyze the data, the architecture may include a data modeling layer, such as a multidimensional OLAP cube or tabular data model in Azure Analysis Services. It might also support self-service BI, using the modeling and visualization technologies in Microsoft Power BI or Microsoft Excel. Analysis and reporting can also take the form of interactive data exploration by data scientists or data analysts. For these scenarios, many Azure services support analytical notebooks, such as Jupyter, enabling these users to leverage their existing skills with Python or R. For large-scale data exploration, you can use Microsoft R Server, either standalone or with Spark.

(8) **Orchestration :** Most big data solutions consist of repeated data processing operations, encapsulated in workflows, that transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report or dashboard. To automate these workflows, you can use an

orchestration technology such Azure Data Factory or Apache Oozie and Sqoop.

7. *What are the different types of Big data technologies?*

New strategies and methods are explored to make the contemporary practice of Big data that is giving strength and consistency to upraise business to the next level.

The finest evolution in the digital era embraces big data technologies to reckon more spark in the conventional technologies. Now we shall discuss the leading-edge technologies that influence the market and IT industries in recent time.

(1) **Artificial Intelligence** : A broad bandwidth of computer science that deals in designing smart machines capable of accomplishing various tasks that typically demand human intelligence is known as Artificial Intelligence. The excellent aspect of AI is the strength to intellectualize and make decisions that can provide a acceptable likelihood in achieving a definite goal. AI is evolving consistently to make benefits in various industries. For example, AI can be used for drug treatment, healing patients, and conducting surgery in OT.

(2) **NoSQL Database** : NoSQL incorporates a broad range of separate database technologies that are developing to design modern applications. It depicts a non-SQL or non relational database that delivers a method for accumulation and retrieval of data. They are deployed in real-time web applications and big data analytics. It stores unstructured data and delivers faster performance, and proffers flexibility while dealing with varieties of data types at a huge scale. Examples included MongoDB, Redis, and Cassandra.

It uses data structures that are different from those accounted by default in relational databases, it makes computations quicker in NoSQL. For example, companies like Facebook, Google and Twitter store terabytes of user data every single day.

(3) **R Programming** : R is the programming language and an open-source project. It is free software highly used for statistical computing, visualization, unified developing environments like Eclipse and Visual Studio assistance communication. Expert says it has graced the most prominent language across the world. Along with it, being used by data miners and statisticians, it is widely implemented for designing statistical software and mainly in data analytics.

(4) **Data Lakes** : Data Lakes refers to a consolidated repository to stockpile all formats of data in terms of structured and unstructured data at any scale. In the process of data accumulation, data can be saved as it is, without transforming it into structured data and executing numerous kinds of data analytics from dashboard and data visualization to big data transformation, real-time analytics and machine learning for better business interferences. Organizations that use data lakes will be able to defeat their peers, new types of analytics can be conducted such as machine learning across new sources of log files, data from social media and click-streams and even IoT devices freeze in data lakes.

(5) **Predictive Analytics** : A subpart of big data analytics, it endeavors to predict future behavior via prior data. It works using machine learning technologies, data mining and statistical modeling and some mathematics, able to forecast future events. The science of predictive analytics generates upcoming inferences with a compelling degree of precision. With the tools and models of predictive analytics, any firm deploy prior and latest data to drag out trends and behaviors that could occur at a particular time. For example, to explore the relationships among various trending parameters. Such models are designed to assess the pledge or risk delivered by a specific set of possibilities.

(6) **Apache Spark** : With in-built features f streaming, SQL, machine learning and grap processing support, Apache Spark earns the cite the speedest and common generator for big da transformation. It supports majo languages of b data comprising Python, R, Scala, and Java.

(7) **Prescriptive Analytics** : Prescriptive Analyti gives guidance to companies about what they could d when to achieve aspired outcomes. For example, can give notice to a company that the borderline of product is expecting to decrease, then prescripti analytics can assist in investigating various factors i response to market changes and predict the mo favorable outcomes. Where it relates both descripti and predictive analytics but focuses on valuab insights over data monitoring and give the be solution for customer satisfaction, business profit and operational efficiency.

(8) **In-memory Database** : The in-memory databa (IMDB) is stored in the main memory of the compute (RAM) and controlled by the in-memory databa management system. In prior, conventiona databases are stored on disk drives. In-memor databases are built in order to achieve minimum tim by omitting the requirements to access disks. But, a all data is collected and controlled in the mai memory completely, there are high chances of losin the data upon a process or server failure.

(9) **Blockchain** : Blockchain is the assigned databa technology that carries Bitcoin digital currency with unique feature of secured data, once it gets written never be deleted or changed later on the fact. It is highly secure ecosystem and an amazing choice fo various applications of big data in industries banking, finance, insurance, healthcare, retailing etc. Blockchain technology is still in the process development, however, many merchants of variou organizations like AWS, IBM, Microsoft includin startups have tried multiple experiments

introduce the possible solutions in building blockchain technology.

(10) **Hadoop Ecosystem** : The Hadoop ecosystem comprises a platform that assists in resolving the challenges surrounding big data. It incorporates a variety of varied components and services namely ingesting, storing, analyzing, and maintaining inside it. Majority services prevalent in the Hadoop ecosystem are to complement its various components which include HDFS, YARN, MapReduce and Common.

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

Operational and Analytical Data Systems are both very similar in how they provide information on your organization, company, or non-profit, but the two are very structurally different, and provide different types of insights. Lets discuss one by one :

Operational Data Systems : Operational Data is exactly what it sounds like - data that is **produced by your organization's day to day operations**. Things like customer, inventory, and purchase data fall into this category. This type of data is pretty straightforward and will generally look the same for most organizations. If you want to know the most up to date information on something - you're using Operational Data. Operational Data Systems support high-volume low-latency access, called Online Transactional Processing tables, or OLTP, where you want to create, read, update, or delete one piece of data at a time.

Analytical Data Systems : Analytical Data is a little more complex and will look different for different types of organizations; however, at its core is an organization's Operational Data. Analytical Data is used to make business decisions, as opposed to recording the data from actual operational business processes. Examples include grouping customers for market segmentation or changes in

purchase volume over time. Every organization will hav different questions to answer and different decisions t make, so Analytical Data is definitely not one-size-fits-al by any stretch of the imagination! Analytical Data is bes stored in a Data System designed for heavy aggregation data mining, and ad hoc queries, called a Online Analytical Processing system, OLAP, or a Dat Warehouse.

9. *What is the importance of Big data?*

Big data is a combination of structured, sem structured and unstructured data collected b organizations that can be mined for information and use in machine learning projects, predictive modeling an other advanced analytics applications. Systems tha process and store big data have become a commo component of data management architectures i organizations, combined with tools that support big dat analytics uses.

Importance of Big Data

Companies use big data in their systems to improv operations, provide better customer service, creat personalized marketing campaigns and take other action that, ultimately, can increase revenue and profit Businesses that use it effectively hold a potentia competitive advantage over those that don't becaus they're able to make faster and more informed busines decisions.

For example, big data provides valuable insights int customers that companies can use to refine the marketing, advertising and promotions in order to increa customer engagement and conversion rates. Both historic and real-time data can be analyzed to assess the evolvin preferences of consumers or corporate buyers, enablin businesses to become more responsive to customer wan and needs.

Big data is also used by medical researchers t identify disease signs and risk factors and by doctors t

help diagnose illnesses and medical conditions in patients. In addition, a combination of data from electronic health records, social media sites, the web and other sources gives healthcare organizations and government agencies up-to-date information on infectious disease threats or outbreaks.

Here are some more examples of how big data is used by organizations :

(1) In the energy industry, big data helps oil and gas companies identify potential drilling locations and monitor pipeline operations; likewise, utilities use it to track electrical grids.

(2) Financial services firms use big data systems for risk management and real-time analysis of market data.

(3) Manufacturers and transportation companies rely on big data to manage their supply chains and optimize delivery routes.

Other government uses include emergency response, crime prevention and smart city initiatives.

10. *Explain Aggregate data model with example.* **(2015-16)**

Aggregate Data Models

Relational database modeling is vastly different than the types of data structures that application developers use. Using the data structures as modeled by the developers to solve different problem domains has given rise to movement away from relational modeling and towards aggregate models, most of this is driven by Domain Driven Design, An aggregate is a collection of data that we interact with as a unit. These units of data or aggregates form the boundaries for ACID operations with the database, Key-value, Document, and Column-family databases can all be seen as forms of aggregate-oriented database.

Aggregates make it easier for the database to manage data storage over clusters, since the unit of data now could

reside on any machine and when retrieved from the database gets all the related data along with it. Aggregate-oriented databases work best when most data interaction is done with the same aggregate, for example when there is need to get an order and all its details, it better to store order as an aggregate object but dealing with these aggregates to get item details on all the orders is not elegant.

Aggregate-oriented databases make inter-aggregate relationships more difficult to handle than intra-aggregate relationships. Aggregate-ignorant databases are better when interactions use data organized in many different formations. Aggregate-oriented databases often compute materialized views to provide data organized differently from their primary aggregates.

For example this is often done with map-reduce computations, such as a map-reduce job to get items sold per day.

11. *How to calculate risk in marketing?* (2016-17)

We have seen how the interdisciplinary use of big data affected many sectors. Different examples are contagion spreading, music albums success predictions or presidential election.

In financial markets, the sentiment analysis probably represents the major and most known implementation of machine learning techniques on big datasets). In spite of all the hype, though, risk management is still an exception. New information and stack of technologies did not bring as many benefits to the risk management as they did to trading for instance. Risk is indeed usually addressed from an operational perspective, from a customer relationship angle, or specifically for fraud prevention and credit scoring.

Traditionally, there are few techniques used to calculate market risk.

(1) Value-at-Risk (VaR) : It has been used for decades to assess market risk. In a nutshell, the VaR is a statistical technique used to measure the level of risk of a portfolio given a certain confidence interval and within a fixed timeframe.

(2) Conditional VaR (CVaR) : It is also called *expected shortfall*, computes the expected return of the portfolio in the worst scenarios for a certain probability level.

(3) Entropic VaR (EVaR) : It represents the upper bound for both VaR and CVaR, and its dual representation is related to the concept of relative entropy.

12. *What do you mean by shrading?*

Sharding is a method of splitting and storing a single logical dataset in multiple databases. By distributing the data among multiple machines, a cluster of database systems can store larger dataset and handle additional requests. Sharding allows a database cluster to scale along with its data and traffic growth.

It is generally used in Hadoop to increase the total storage capacity of the system.

13. *Why would you use inferential statistics in big data?*

In Inferential statistics, we make an inference from a sample about the population. The main aim of inferential statistics is to draw some conclusions from the sample and generalise them for the population data. For example we have to find the average salary of a data analyst across India. Descriptive statistics describe the important characteristics of data by using mean, median, mode, variance etc. It summarizes the data through numbers and graphs.

In Inferential statistics, we make an inference from a sample about the population. The main aim of inferential statistics is to draw some conclusions from the sample and generalize them for the population data. E.g. we have to find the average salary of a data analyst across India. There are two options

(1) The first option is to consider the data of da analysts across India and ask them their salari and take an average.

(2) The second option is to take a sample of da analysts from the major IT cities in India and tal their average and consider that for across India.

The first option is not possible as it is very difficu to collect all the data of data analysts across India. It time-consuming as well as costly. So, to overcome th issue, we will look into the second option to collect small sample of salaries of data analysts and take the average as India average. This is the inferenti statistics where we make an inference from a sampl about the population. In inferential statistics, w generally use the concepts of probability, distribution and hypothesis testing.

Importance of Inferential Statistics :

(1) Making conclusions from a sample about th population

(2) To conclude if a sample selected is statistically significant to the whole population or not

(3) Comparing two models to find which one is mor statistically significant as compared to the other.

(4) In feature selection, whether adding or removing variable helps in improving the model or not.

14. *Relate crowd sourcing and big data. Justify the relationship with an example.* (2016-17)

Big data is all the rage these days as various organizations dig through large datasets to enhance their operations and discover novel solutions to big data problems. Sorting through the flood of information is a challenge even for large IT organizations.

Crowd sourcing has gained significance as an interesting practice for yielding meaningful insights from big data. Crowd sourcing has eased the process of performing tasks that are hard to crack for computers including audio transcriptions, sentiment analysis,

document summaries, document editing, entity resolution and image annotation. Enterprises that completely crowd source data to make critical business decisions, definitely does have some loopholes.

When making business decisions using crowd sourcing alone, from diverse network sources, businesses need to judge the quality of various data points, find different ways to overcome geographical differences if any and then relate to the goals of an organization. This is when big data analytics proves to be highly beneficial in coagulating crowd sourcing success. Organizations can identify the real titbits in crowd sourced data that drive innovation, development decisions and market practices by making use of well-established big data principles. Big data analytics is a sternly intertwined trend to crowd sourcing.

Crowd sourcing involves ***obtaining, work, information, or opinions from a large group of people who submit their data via the Internet***, social media, and smartphone apps. We can justify it by the following points and examples.

Crowd Sourcing Helps ease the Process of Big Data Analytics

(1) Generally, a data scientist spends 78% of his time in preparing the data for big data analytics. Thus, an intelligent and cost effective strategy for big data companies would be to hand over the unstructured data sets to a well-managed crowd sourcing platform so that the crowd will tell more about the information contained within the data points collected. For example, before the analysis the crowd can tell whether the data points are a Tweet or updates from Facebook and whether it carries a negative, positive or neutral connotation.

(2) Crowd helps provide structure (document editing, audio transcription, image annotation) to big data thereby helping analysts improve their analysis predictive models by 25%.

(3) Crowd sourcing along with big data analytics ca help reveal hidden insights from dispersed bu connected information quickly.

(4) Big data problems can be solved with more accurac with crowd sourcing as a reliable medium.

(5) The results from the crowd can be used by dat scientists to enhance the efficiency of the machin learning algorithms.

15. *Write down the key features of Big data in detail.*

The following is a list of the features of Big Data t help you get on the right track with determining what you big data analytics requirements should be :

(1) **Data Processing :** Data processing features involv the collection and organization of raw data to produc meaning. Data modeling takes complex data sets and displays them in a visual diagram or chart. This makes it digestible and easy to interpret for user trying to utilize that data to make decisions.

Data mining tools allow users to extract and analyze data from different perspectives and summarize it into actionable insights. It is especially useful on large unstructured data sets collected over a period of time.

Big Data analytics tools should enable data import from sources such as Microsoft Access, Microsoft Excel, text files and other flat files. Being able to merge data from multiple sources and in multiple formats will reduce labor by preventing the need for data conversion and speed up the overall process by importing directly to the system.

(2) **Predictive Applications :** Identity management (or identity and access management) is the organizational process for controlling who has access to your data. Identity management functionality manages identifying data for everything that has access to a system including individual users, computer hardware and software applications.

Identity management also deals with issues including how users gain an identity with access, protection of those identities and support for other system protections such as network protocols and passwords. It determines whether a user has access to a system and the level of access that user has permission to utilize.

Identity management applications aim to ensure only authenticated users can access your system and, by extension, your data. It is a crucial element of any organization's security plan and will include real-time security and fraud analytics capabilities.

(3) **Analytics :** Big Data analytics tools offer a variety of analytics packages and modules to give users options. Risk analytics, for example, is the study of the uncertainty surrounding any given action. It can be used in combination with forecasting to minimize the negative impacts of future events. Risk analytics allow users to mitigate these risks by clearly defining and understanding their organization's tolerance for and exposure to risk.

Decision management involves the decision making processes of running a business. Decision management modules treat decisions as usable assets. It incorporates technology at key points to automate parts of that decision making process. Content analysis is very similar to text analysis but includes the analysis of all formats of documentation including audio, video, pictures, etc. Social media analytics is one form of content analysis that focuses on how your user base is interacting with your brand on social media. Statistical analytics collects and analyzes data sets composed of numbers. The goal is to draw a sample from the total data that is representative of a total population. Statistical analysis takes place in five steps: describing the nature of the data, exploring the relation of the data to the population that provided it, creating a model to summarize the connections, proving or disproving its

validity, and employing predictive analytics to guid decision-making.

(4) **Reporting Features** : Reporting functions kee users on top of their business. Real-time reportin gathers minute-by-minute data and relays it to you typically in an intuitive dashboard format. Thi allows users to make snap decisions in heavily time constrained situations and be both more prepare and more competitive in a society that moves at th speed of light.

(5) **Security Features** : Keeping your system safe i crucial to a successful business. Big Data analytic tools should offer security features to ensure securit and safety. One such feature is single sign-on. Als called SSO, it is an authentication service tha assigns users' a single set of login credentials t access multiple applications. It authenticates en user permissions and eliminates the need to logi multiple times during the same session. It can als log and monitor user activities and accounts to kee track of who is doing what in the system.

Another security feature offered by Big Dat analytics platforms is data encryption. Data encryptio involves changing electronic information into unreadabl formats by using algorithms or codes. While web browser offer automatic encryption, you want something a bit mor robust for your sensitive proprietary data. Make sure th system offers comprehensive encryption capabilities whe looking for a data analytics application.

16. *Discuss Big data in health care and medicine.* (2018-19)

'Big data' is massive amounts of information that ca work wonders. It has become a topic of special interest fo the past two decades because of a great potential that i hidden in it. Various public and private sector industrie generate, store, and analyze big data with an aim t improve the services they provide. In the healthcar industry, various sources for big data include hospital records, medical records of patients, results of medical examinations, and devices that are a part of internet of things. Biomedical research also generates a significant portion of big data relevant to public healthcare. This data requires proper management and analysis in order to derive meaningful information. Otherwise, seeking solution by analyzing big data quickly becomes comparable to finding a needle in the haystack. There are various challenges associated with each step of handling big data which can only be surpassed by using high-end computing solutions for big data analysis. That is why, to provide relevant solutions for improving public health, healthcare providers are required to be fully equipped with appropriate infrastructure to systematically generate and analyze big data.

(Figure)

An efficient management, analysis, and interpretation of big data can change the game by opening new avenues for modern healthcare. That is exactly why various industries, including the healthcare industry, are taking vigorous steps to convert this potential into better services and financial advantages. With a strong

integration of biomedical and healthcare data, mod healthcare organizations can possibly revolutionize medical therapies and personalized medicine. Some them are discussed below.

Electronic Health Records : It is important to note t the National Institutes of Health (NIH) rece announced the "All of Us" initiative that aims to collect million or more patients' data such as EHR, inclu medical imaging, socio-behavioral, and environmental over the next few years. EHRs have introduced m advantages for handling modern healthcare related dat

Digitization of Healthcare and Big Data : Simila EHR, an electronic medical record (EMR) stores standard medical and clinical data gathered from patients. The management and usage of such health data has been increasingly dependent on informa technology. The development and usage of wells monitoring devices and related software that can gener alerts and share the health related data of a patient w the respective health care providers has ga momentum, especially in establishing a real-t biomedical and health monitoring system. These dev are generating a huge amount of data that can be analy to provide real-time clinical or medical care. The use of data from healthcare shows promise for improving hea outcomes and controlling costs.

Big Data from Omics Studies : NGS has gre simplified the sequencing and decreased the costs generating whole genome sequence data. The cost complete genome sequencing has fallen from millions t couple of thousand dollars .NGS technology has resulte an increased volume of biomedical data that comes fr genomic and transcriptomic studies. According to estimate, the number of human genomes sequenced 2025 could be between 100 million to 2 billion.

Big Data in Biomedical Research : A biological syst such as a human cell, exhibits molecular and physi events of complex interplay. In order to underst interdependencies of various components and events such a complex system, a biomedical or biological experiment usually gathers data on a smaller and/or simpler component. Consequently, it requires multiple simplified experiments to generate a wide map of a given biological phenomenon of interest. This indicates that more the data we have, the better we understand the biological processes. With this idea, modern techniques have evolved at a great pace. For instance, one can imagine the amount of data generated since the integration of efficient technologies like next-generation sequencing (NGS) and Genome wide association studies (GWAS) to decode human genetics. NGS-based data provides information at depths that were previously inaccessible and takes the experimental scenario to a completely new dimension.

Internet of Things (IOT) : Healthcare industry has not been quick enough to adapt to the big data movement compared to other industries. Therefore, big data usage in the healthcare sector is still in its infancy. For example, healthcare and biomedical big data have not yet converged to enhance healthcare data with molecular pathology. Such convergence can help unravel various mechanisms of action or other aspects of predictive biology. Therefore, to assess an individual's health status, bio molecular and clinical datasets need to be married. One such source of clinical data in healthcare is 'internet of things' (IoT).

Mobile Computing And Mobile Health (mHealth) : In today's digital world, every individual seems to be obsessed to track their fitness and health statistics using the in-built pedometer of their portable and wearable devices such as, smartphones, smartwatches, fitness dashboards or tablets. With an increasingly mobile society in almost all aspects of life, the healthcare infrastructure needs remodeling to accommodate mobile devices .The practice of medicine and public health using mobile devices, known as mHealth or mobile health, pervades different degrees of health care especially for chronic diseases, such as diabetes and cancer .Healthcare organizations are increasingly using mobile health and wellness services for implementing novel and innovative ways to provide care and coordinate health as well as wellness.

17. *Briefly explain the evolution of of Big Data.*

The first major data project is created in 1937 an was ordered by the Franklin D. Roosevelt's administrati in the USA. After the Social Security Act became law 1937, the government had to keep track of contributi from 26 million Americans and more than 3 milli employers. IBM got the contract to develop punch car reading machine for this massive book keeping project.

The first data-processing machine appeared in 19 and was developed by the British to decipher Nazi cod during World War II. This device, named Colossu searched for patterns in intercepted messages at a rate 5.000 characters per second. Thereby reducing the tas from weeks to merely hours.

In 1952 the National Security Agency (NSA) created and within 10 years contract more than 12.00 cryptologists. They are confronted with informatio overload during the Cold War as they start collecting an processing intelligence signals automatically.

In 1965 the United States Government decided to build th first data centre to store over 742 million tax returns an 175 million sets of fingerprints by transferring all thos records onto magnetic computer tape that had to be store in a single location. The project was later dropped out o fear for 'Big Brother', but it is generally accepted that i was the beginning of the electronic data storage era.

In 1989 British computer scientist Tim Berners-Le invented eventually the World Wide Web. He wanted t facilitate the sharing of information via a 'hypertext system. Little could he know at the moment the impact o his invention.

As of the '90s the creation of data is spurred as mor and more devices are connected to the internet. In 1995 th first super-computer is built, which was able to do as muc work in a second than a calculator operated by a singl person can do in 30.000 years.

90% of the available data has been created in the las two years and the term Big Data has been around 2005

when it was launched by O'Reilly Media in 2005. However, the usage of Big Data and the need to understand all available data has been around much longer.

In 2005 Roger Mougalas from O'Reilly Media coined the term Big Data for the first time, only a year after they created the term Web 2.0. It refers to a large set of data that is almost impossible to manage and process using traditional business intelligence tools.

2005 is also the year that Hadoop was created by Yahoo! built on top of Google's MapReduce. It's goal was to index the entire World Wide Web and nowadays the open-source Hadoop is used by a lot organizations to crunch through huge amounts of data. As more and more social networks start appearing and the Web 2.0 takes flight, more and more data is created on a daily basis. Innovative startups slowly start to dig into this massive amount of data and also governments start working on Big Data projects.

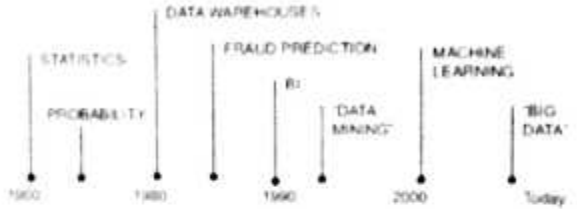
In 2009 the Indian government decides to take an iris scan, fingerprint and photograph of all of this 1.2 billion inhabitants. All this data is stored in the largest biometric database in the world.

In 2010 Eric Schmidt speaks at the Techonomy conference in Lake Tahoe in California and he states that "there were 5 exabytes of information created by the entire world between the dawn of civilization and 2003. Now that same amount is created every two days."

In 2011 the McKinsey report on Big Data: The next frontier for innovation, competition, and productivity, states that in 2018 the USA alone will face a shortage of 140.000 – 190.000 data scientist as well as 1.5 million data managers.

In the past few years, there has been a massive increase in Big Data startups, all trying to deal with Big Data and helping organizations to understand Big Data and more and more companies are slowly adopting and moving towards Big Data. However, while it looks like Big

Data is around for a long time already. in fact Big Data is as far as the internet was in 1993. The large Big Data revolution is still ahead of us so a lot will change in the coming years. Let the Big Data era begin.



(Figure : A Brief History of Data)

❑❑

HADOOP AND MAP REDUCE

1. *What is Hadoop also define its architecture* **(2015-16)**

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing. It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

Apache Hadoop consists of two sub-component :

(1) Hadoop MapReduce : MapReduce is a computational model and software framework for writing applications which are run on Hadoop. These MapReduce programs are capable of processing enormous data in parallel on large clusters of computation nodes.

(2) HDFS (Hadoop Distributed File System) : HDFS takes care of the storage part of Hadoop applications. MapReduce applications consume data from HDFS. HDFS creates multiple replicas of data blocks and distributes them on compute nodes in a cluster. This distribution enables reliable and extremely rapid computations.

Although Hadoop is best known for MapReduce and its distributed file system- HDFS, the term is also used for a family of related projects that fall under the umbrella of distributed computing and large-scale data processing. Hadoop has a Master-Slave Architecture for data storage and distributed data processing using MapReduce and HDFS methods.

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

(1) **Hadoop Common** : These are Java libraries and utilities required by other Hadoop modules.

(2) **Hadoop YARN** : This is a framework for job scheduling and cluster resource management.

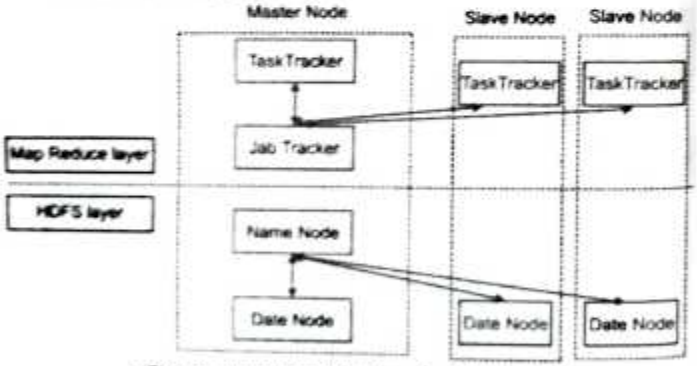
Name Node : NameNode represented every files and directory which is used in the namespace

Data Node : Data Node helps you to manage the state of an HDFS node and allows you to interacts with the blocks

Master Node : The master node allows you to conduct parallel processing of data using Hadoop MapReduce.

Slave Node : The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations. Moreover, all the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.

In Hadoop, master or slave system can be set up in the cloud or on-premise



(Figure : High Level Hadoop Architecture)

2. *How does Hadoop work? Explain the advantages of Hadoop?* **(2015-16)**

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing but as an alternative, you can tie together many

commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs :

(1) Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).

(2) These files are then distributed across various cluster nodes for further processing.

(3) HDFS, being on top of the local file system, supervises the processing.

(4) Blocks are replicated for handling hardware failure.

(5) Checking that the code was executed successfully.

(6) Performing the sort that takes place between the map and reduce stages.

(7) Sending the sorted data to a certain computer.

(8) Writing the debugging logs for each job.

Advantages of Hadoop

(1) **Varied Data Sources** : Hadoop accepts a variety of data. Data can come from a range of sources like email conversation, social media etc. and can be of structured or unstructured form. Hadoop can derive value from diverse data. Hadoop can accept data in a text file, XML file, images, CSV files etc.

(2) **Cost-Effective** : Hadoop is an economical solution as it uses a cluster of commodity hardware to store data. Commodity hardware is cheap machines hence the cost of adding nodes to the framework is not much high. In Hadoop 3.0 we have only 50% of storage overhead as opposed to 200% in Hadoop2.x. This requires less machine to store data as the redundant data decreased significantly.

(3) **Performance** : Hadoop with its distribute processing and distributed storage architectu processes huge amounts of data with high spee Hadoop even defeated supercomputer the faste machine in 2008. It divides the input data file into number of blocks and stores data in these blocks ov several nodes. It also divides the task that us submits into various sub-tasks which assign to the worker nodes containing required data an these sub-task run in parallel thereby improving th performance.

(4) **Fault-Tolerant** : In Hadoop 3.0 fault tolerance provided by erasure coding. For example, 6 dat blocks produce 3 parity blocks by using erasur coding technique, so HDFS stores a total of these blocks. In event of failure of any node the data bloc affected can be recovered by using these parity block and the remaining data blocks.

(5) **Highly Available** : In Hadoop 2.0, HDF architecture has a single active NameNode and single Standby NameNode, so if a NameNode goe down then we have standby NameNode to count on But Hadoop 3.0 supports multiple standby NameNode making the system even more highly available as it can continue functioning in case if tw or more NameNodes crashes.

(6) **Low Network Traffic** : In Hadoop, each job submitted by the user is split into a number of independent sub-tasks and these sub-tasks are assigned to the data nodes thereby moving a small amount of code to data rather than moving huge data to code which leads to low network traffic.

(7) **High Throughput** : Throughput means job done per unit time. Hadoop stores data in a distributed fashion which allows using distributed processing with ease. A given job gets divided into small jobs which work on chunks of data in parallel thereby giving high throughput.

(8) **Open Source** : Hadoop is an open source technology i.e. its source code is freely available. We can modify the source code to suit a specific requirement.

(9) **Scalable** : Hadoop works on the principle of horizontal scalability i.e. we need to add the entire machine to the cluster of nodes and not change the configuration of a machine like adding RAM, disk and so on which is known as vertical scalability. Nodes can be added to Hadoop cluster on the fly making it a scalable framework.

(10) **Ease of Use** : The Hadoop framework takes care of parallel processing. MapReduce programmers does not need to care for achieving distributed processing, it is done at the backend automatically.

(11) **Compatibility** : Most of the emerging technology of Big Data is compatible with Hadoop like Spark, Flink etc. They have got processing engines which work over Hadoop as a backend i.e. We use Hadoop as data storage platforms for them.

(12) **Multiple Languages Supported** : Developers can code using many languages on Hadoop like C, C++, Perl, Python, Ruby, and Groovy.

3. *Explain Hadoop Distributed File Sysytem?*

With growing data velocity the data size easily outgrows the storage limit of a machine. A solution would be to store the data across a network of machines. Such file systems are called distributed file systems. Since data is stored across a network all the complications of a network come in This is where Hadoop comes in. It provides one of the most reliable file systems. HDFS (Hadoop Distributed File System) is a unique design that provides storage for extremely large files with streaming data access pattern and it runs on commodity hardware. Let's elaborate the terms:

(1) **Extremely Large Files** : Here we are talking about the data in range of petabytes (1000 TB).

(2) **Streaming Data Access Pattern** : HDFS is designed on principle of write-once and read-many-

times. Once data is written large portions of dataset can be processed any number times.

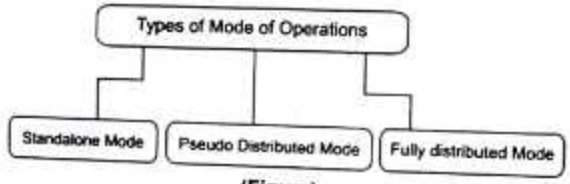
(3) **Commodity Hardware** : Hardware that is inexpensive and easily available in the market. This is one of feature which specially distinguishes HDFS from other file system.

Features :

(1) Distributed data storage.
(2) Blocks reduce seek time.
(3) The data is highly available as the same block is present at multiple datanodes.
(4) Even if multiple data nodes are down we can still do our work, thus making it highly reliable.
(5) High fault tolerance.

4. *What do you mean by Hadoop operation mode Or What are the different modes in which Hadoop can be installed and what is the use of each modes in application and developer point of view.* (2018-19)

Hadoop posses a scale-out storage property, which means that we can scale up or scale down the number of nodes as per are a requirement in the future which is really a cool feature.



(Figure)

Hadoop Mainly Works on 3 Different Modes :

(1) Standalone Mode
(2) Pseudo-distributed Mode
(3) Fully-Distributed Mode

(1) **Standalone Mode :** In Standalone Mode none of the Daemon will run (Deamons are the processes running in background) i.e. Namenode, Datanode, Secondary Name node, Job Tracker, and Task Tracker. We use job-tracker and task-tracker for processing purposes

in Hadoop1. For Hadoop2 we use Resource Manager and Node Manager. Standalone Mode also means that we are installing Hadoop only in a single system. By default, Hadoop is made to run in this Standalone Mode or we can also call it as the Local mode. We mainly use Hadoop in this Mode for the Purpose of Learning, testing, and debugging. Hadoop works very much Fastest in this mode among all of these 3 modes. As we all know HDFS (Hadoop distributed file system) is one of the major components for Hadoop which utilized for storage Permission is not utilized in this mode. You can think of HDFS as similar to the file system's available for windows i.e. NTFS (New Technology File System) and FAT32(File Allocation Table which stores the data in the blocks of 32 bits). when your Hadoop works in this mode there is no need to configure the files – hdfs-site.xml, mapred-site.xml, core-site.xml for Hadoop environment. In this Mode, all of your Processes will run on a single JVM (Java Virtual Machine) and this mode can only be used for small development purposes.

Here is the summarized view of the standalone mode :

(a) Used for debugging purpose
(b) HDFS is not being used
(c) Uses local file system for input and output
(d) No need to change any configuration files
(e) Default Hadoop Modes

(2) **Pseudo Distributed Mode (Single Node Cluster) :** In Pseudo-distributed Mode we also use only a single node, but the main thing is that the cluster is simulated, which means that all the processes inside the cluster will run independently to each other. All the daemons that are Namenode, Datanode, Secondary Name node, Resource Manager, Node Manager, etc. will be running as a separate process on separate JVM(Java Virtual Machine) or we can say run on different java processes that is why it is called a Pseudo-distributed.

One thing we should remember that as we are using only the single node set up so all the Master and Slave processes are handled by the single system. Namenode and Resource Manager are used as Master and Datanode and Node Manager is used as a slave. A secondary name node is also used as a Master. The purpose of the Secondary Name node is to just keep the hourly based backup of the Name node. In this Mode,

(a) Hadoop is used for development and for debugging purposes both.

(b) Our HDFS(Hadoop Distributed File System) is utilized for managing the Input and Output processes.

(c) We need to change the configuration files mapred-site.xml, core-site.xml, hdfs-site.xml for setting up the environment.

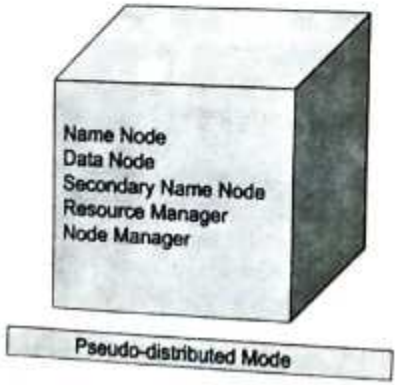
(d) Single Node Hadoop deployment running on Hadoop is considered as pseudo distributed mode

(e) All the master & slave daemons will be running on the same node

(f) Mainly used for testing purpose

(g) Replication Factor will be ONE for Block

(h) Changes in configuration files will be required for all the three files- mapred-site.xml, core-site.xml, hdfs-site.xml

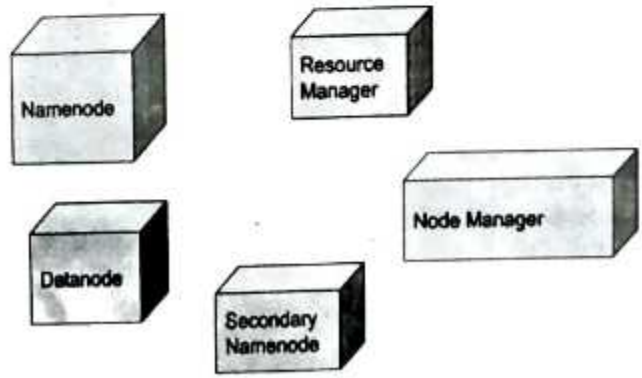


(3) Fully Distributed Mode (Multi-Node Cluster) : This is the most important one in which multiple nodes are used few of them run the Master Daemon's that are Name node and Resource Manager and the rest of them run the Slave Daemon's that are Data Node and Node Manager. Here Hadoop will run on the clusters of Machine or nodes. Here the data that is used is distributed across different nodes. This is actually the Production Mode of Hadoop let's clarify or understand this Mode in a better way in Physical Terminology. Once you download the Hadoop in a tar file format or zip file format then you install it in your system and you run all the processes in a single system but here in the fully distributed mode we are extracting this tar or zip file to each of the nodes in the Hadoop cluster and then we are using a particular node for a particular process. Once you distribute the process among the nodes then you'll define which nodes are working as a master or which one of them is working as a slave.

(a) Production phase of Hadoop

(b) Separate nodes for master and slave daemons

(c) Data are used and distributed across multiple nodes



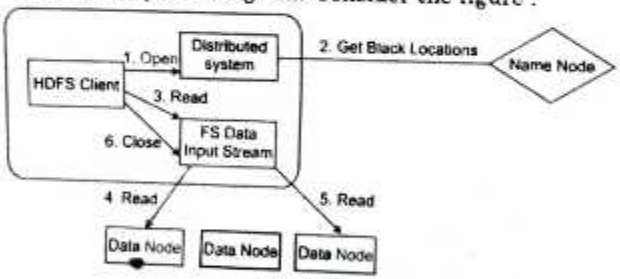
(Figure : Fully Distributed Mode)

5. ◆ ***Write the working procedure of HDFS and also explain the features of HDFS.***
◆ ***Show on How Client read and writes data in HDFS. Give an Example code.*** **(2016-17)**

With growing data velocity the data size easil outgrows the storage limit of a machine. A solution woul be to store the data across a network of machines. Such fil systems are called distributed file systems. Since data i stored across a network all the complications of a networ come in. HDFS (Hadoop Distributed File System) is a unique design that provides storage for extremely larg files with streaming data access pattern and it runs o commodity hardware. Lets discuss the working of HDF through read and write operation.

0Process of File Read in HDFS0

Let's get an idea of how data flows between the clien interacting with HDFS, the name node, and the data node with the help of a diagram. Consider the figure :



(Figure)

Step 1 : The client opens the file it wishes to read by calling open () on the File System Object (which for HDFS is an instance of Distributed File System).

Step 2 : Distributed File System (DFS) calls the name node, using remote procedure calls (RPCs), to determine the locations of the first few blocks in the file. For each block, the name node returns the addresses of the data nodes that have a copy of that block. The DFS returns an FS Data Input Stream to the client for it to read data from. FS Data Input Stream in turn wraps a DFS Input Stream, which manages the data node and name node I/O.

Step 3 : The client then calls read () on the stream. DFS Input Stream, which has stored the info node addresses for the primary few blocks within the file, then connects to the primary (closest) data node for the primary block in the file.

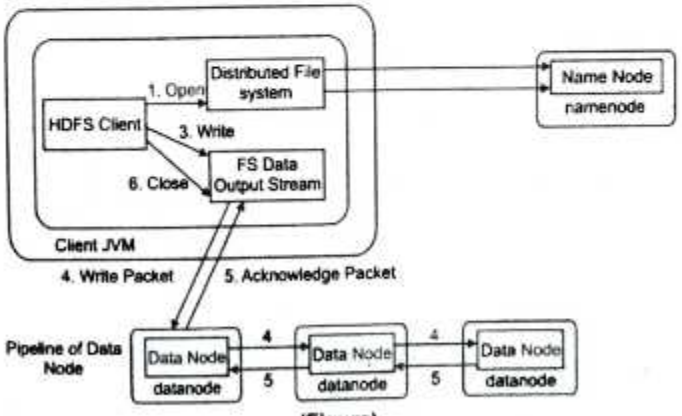
Step 4 : Data is streamed from the data node back to the client, which calls read() repeatedly on the stream.

Step 5 : When the end of the block is reached, DFS Input Stream will close the connection to the data node, then finds the best data node for the next block. This happens transparently to the client, which from its point of view is simply reading an endless stream. Blocks are read as, with the DFS Input Stream opening new connections to data nodes because the client reads through the stream. It will also call the name node to retrieve the data node locations for the next batch of blocks as needed.

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

Process of File Write in HDFS

Next, we'll check out how files are written to HDFS. Consider the figure 1.2 to get a better understanding of the concept.



(Figure)

Step 1 : The client creates the file by calling create () on Distributed File System(DFS).

Step 2 : DFS makes an RPC call to the name node to create a new file in the file system's namespace, with no blocks associated with it. The name node performs various checks to make sure the file doesn't already exist and that the client has the right permissions to create the file. If these checks pass, the name node prepares a record of the new file; otherwise, the file can't be created and therefore the client is thrown an error i.e. IO Exception. The DFS returns an FS Data Output Stream for the client to start out writing data to.

Step 3 : Because the client writes data, the DFS Output Stream splits it into packets, which it writes to an indoor queue called the info queue. The data queue is consumed by the Data Streamer, which is liable for asking the name node to allocate new blocks by picking an inventory of suitable data nodes to store the replicas. The list of data nodes forms a pipeline, and here we'll assume the replication level is three, so there are three nodes in the pipeline. The Data Streamer streams the packets to the primary data node within the pipeline, which stores each packet and forwards it to the second data node within the pipeline.

Step 4 : Similarly, the second data node stores the packet and forwards it to the third (and last) data node in the pipeline.

Step 5 : The DFS Output Stream sustains an internal queue of packets that are waiting to be acknowledged by data nodes, called an "ack queue".

Step 6 : This action sends up all the remaining packets to the data node pipeline and waits for acknowledgments before connecting to the name node to signal whether the file is complete or not.

HDFS follows Write Once Read Many models. So, we can't edit files that are already stored in HDFS, but we can include it by again reopening the file. This design allows HDFS to scale .to a large number of concurrent clients because the data traffic is spread across all the data nodes in the cluster. Thus, it increases the availability, scalability, and throughput of the system.

6. *Define the term Heartbeat, Balancing and Replication related to HDFS.*

Heartbeat : It is the signal that data node continuously sends to name node. If name node doesn't receive heartbeat from a data node then it will consider it dead.

Balancing : If a data node is crashed the blocks present on it will be gone too and the blocks will be under-replicated compared to the remaining blocks. Here master node(name node) will give a signal to data nodes containing replicas of those lost blocks to replicate so that overall distribution of blocks is balanced.

Replication : It is done by data node. No two replicas of the same block are present on the same data node. We perform replication to achieve fault tolerance.

7. *State the usage of Hadoop pipes.* **(2016-17)**

Hadoop Pipes is the name of the C++ interface to Hadoop MapReduce. Unlike Streaming, which uses standard input and output to communicate with the map and reduce code, Pipes uses sockets as the channel over which the task tracker communicates with the process running the C++ map or reduce function.

8. *Differentiate "Scale up and Scale out" Explain with an example How Hadoop uses Scale out feature to improve the Performance.*

The term "scaling up" means to use a more powerful single server to process the workload that fits within the server boundaries. One can easily scale up the cluster by adding more nodes. Scale-out is a different model which utilizes multiple processors as a single entity so a business can scale beyond the computer capacity of a single server. Scale out is a type of capacity expansion concentrating on the addition of new hardware resources instead of increasing the capacity of already available hardware resources such as storage.

Hadoop uses scale out feature to improve t performance by using two types of Scaling

Vertical Scaling : When we add more resources to single machine when the load increases. For example, y need 20 GB of ram but currently your server has 10 GB ram so you add extra ram to the same server to meet t needs.

Horizontal Scaling of Scaling Out : when you add mo machines to match the resources need it's called horizon scaling. So if I have a machine of already 10 GB I'll add extra machine with 10 GB ram.

9. *Explain in detail about Map-reduce Workflows.*

MapReduce is a programming model or patte within the Hadoop framework that is used to access l data stored in the Hadoop File System (HDF MapReduce facilitates concurrent processing by splittn petabytes of data into smaller chunks, and processing the in parallel on Hadoop commodity servers. MapReduce is software framework for processing large data sets in distributed fashion over a several machines. The core id behind MapReduce is mapping your data set into collection of <key, value> pairs, and then reduci overall pairs with the same key. The overall concept simple, but is actually quite expressive when you consid that :

(1) Almost all data can be mapped into <key, valu pairs somehow, and

(2) Your keys and values may be of any type: string integers, dummy types and, of course, <key,valu pairs the mselves. The canonical MapRedu use case is counting word frequencies in a large te what you can do in the MapReduce framewo include :

(a) Distributed sort
(b) Distributed search
(c) Web-link graph traversal
(d) Machine learning

A MapReduce Workflow

Map Reduce follow the given Steps of Job Execution flow :

Map Reduce processes the data in various phases with the help of different components. Let's discuss the steps of job execution in Hadoop.

(1) **Input Files :** In input files data for MapReduce job is stored. In HDFS, input files reside. Input files format is arbitrary. Line-based log files and binary format can also be used.

(2) **Input Format :** After that Input Format defines how to split and read these input files. It selects the files or other objects for input. Input Format creates Input Split.

(3) **Input Splits :** It represents the data which will be processed by an individual Mapper. For each split, one map task is created. Thus the number of map tasks is equal to the number of Input Splits. Framework divide split into records, which mapper process.

(4) **Record Reader :** It communicates with the input Split. And then converts the data into key-value pairs suitable for reading by the Mapper. Record Reader by default uses Text Input Format to convert data into a key-value pair.

It communicates to the Input Split until the completion of file reading. It assigns byte offset to each line present in the file. Then, these key-value pairs are further sent to the mapper for further processing.

(5) **Mapper :** It processes input record produced by the Record Reader and generates intermediate key-value pairs. The intermediate output is completely different from the input pair. The output of the mapper is the full collection of key-value pairs.

Hadoop framework doesn't store the output of mapper on HDFS. It doesn't store, as data is temporary and writing on HDFS will create unnecessary multiple copies. Then Mapper passes the output to the combiner for further processing.

(6) **Combiner** : Combiner is Mini-reducer wh performs local aggregation on the mapper's output. minimizes the data transfer between mapper a reducer. So, when the combiner functional completes, framework passes the output to t partitioner for further processing.

(7) **Partitioner** : Partitioner comes into the existence we are working with more than one reducer. It tak the output of the combiner and performs partitioni Partitioning of output takes place on the basis of t key in MapReduce. By hash function, key (or a subs of the key) derives the partition.

On the basis of key value in MapReduc partitioning of each combiner output takes place. Al then the record having the same key value goes in the same partition. After that, each partition is se to a reducer. Partitioning in MapReduce executi allows even distribution of the map output over th reducer.

(8) **Shuffling and Sorting** : After partitioning, th output is shuffled to the reduce node. The shuffling i the physical movement of the data which is done ov the network. As all the mappers finish and shuff the output on the reducer nodes.

Then framework merges this intermediat output and sort. This is then provided as input t reduce phase.

(9) **Reducer** : Reducer then takes set of intermediat key-value pairs produced by the mappers as th input. After that runs a reducer function on each i them to generate the output.

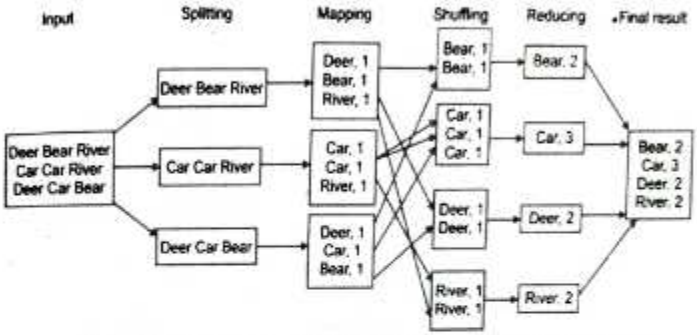
The output of the reducer is the fina output. Then framework stores the outpu on HDFS.

(10) **Record Writer** : It writes these outpu key-value pair from the Reducer phase to the outpu files.

(11) **Output Format** : Output Format defines th way how Record Reader writes these outpu

key-value pairs in output files. So, its instances provided by the Hadoop write files in HDFS. Thus Output Format instances write the final output of reducer on HDFS.

The diagram below illustrates the described scenario nicely



(Figure : The Overall Mapreduce Word Count Process)

10. *Explain in detail about configuring/installing the hadoop in local/standalone mode.* **(2016-17)**

Installing and Configuring Hadoop in Standalone Mode

You might want to create a dedicated user for running Apache Hadoop but it is not a prerequisite. In our demonstration, we will be using a default user for running Hadoop.

Environment :

Ubuntu 10.10

JDK 6 or above

Hadoop-1.1.2 (Any stable release)

Follow these steps for installing and configuring Hadoop on a single node :

Step 1-Install Java : Use the below command to begin the installation of Java

```
$ sudo apt-get install openjdk-6-jdk
```

This will install the full JDK under /usr/lib/jvm/java-6-sundirectory.

Step 2 : Verify java Installation :

(1) You can verify java installation using the following command

```
$ java – version
```

(2) On executing this command, you should see output similar to the following :

```
java version "1.6.0_27"
```

Step 3 : SSH Configuration :

(1) Install SSH using the command

```
Sudo apt-get install ssh
```

(2) Generate ssh key

(3) Now copy the public key

(4) Verify SSH configuration using following command

```
ssh localhost
```

Pressing yes will add localhost to known hosts

Step 4 : Download Hadoop :

(1) Download the latest stable release of Apache Hadoop from http://hadoop.apache.org/releases.html.

Unpack the release tar – zxvf hadoop-1.0.3.tar.gz

(2) Save the extracted folder to an appropriate location, HADOOP_HOME will be pointing to this directory.

Step 5 : Verify Hadoop :

(1) Check if the following directories exist under HADOOP_HOME: bin, conf, lib, bin

(2) Use the following command to create an environment variable that points to the Hadoop installation directory (HADOOP_HOME)

```
export HADOOP_HOME=/home/user/hadoop
```

Now place the Hadoop binary directory on your command-line path by executing the command

```
export PATH=$PATH:$HADOOP_HOME/bin
```

Use this command to verify your Hadoop installation:

```
hadoop version
```

The o/p should be similar to below one

```
Hadoop 1.1.2
```

Step 6 : Configure JAVA-HOME :

(1) Hadoop requires Java installation path to work on, for this we will be setting JAVA_HOME environment variable and this will point to our Java installation dir.

(2) Java_Home can be configured in ~/.bash_profile or ~/.bashrc file. Alternatively you can also let hadoop know this by setting Java_Home in hadoop conf/hadoop-env.sh file.

(3) Use the below command to set JAVA_HOME on Ubuntu

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

JAVA_HOME can be verified by command

```
echo $JAVA_HOME
```

Step 7 : Create Data Directory for Hadoop :

(1) An advantage of using Hadoop is that with just a limited number of directories you can set it up to work correctly. Let us create a directory with the name hdfs and three sub-directories name, data and tmp.

(2) Since a Hadoop user would require to read-write to these directories you would need to change the permissions of above directories to 755 or 777 for Hadoop user.

Step 8 : Configure Hadoop XML files : Next, we will configure Hadoop XML file. Hadoop configuration files are in the HADOOP_HOME/conf dir.

Step 9 : Format Hadoop Name Node : Execute the below command from hadoop home directory

```
$ ~/hadoop/bin/hadoop namenode -format
```

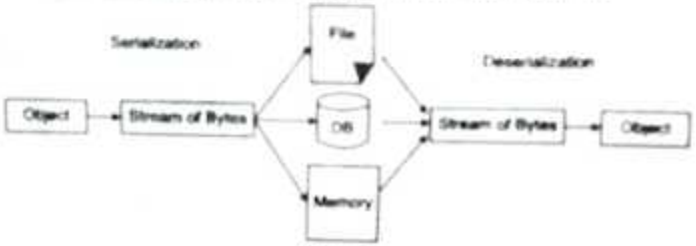
11. *Define serialization in Hadoop.*

Serialization is the process of turning structured objects into a byte stream for transmission over a network or for writing to persistent storage or we can say Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices.

Computer systems may vary in their hardware architecture, OS, addressing mechanisms. Internal binary representations of data also vary accordingly in every environment. Storing and exchanging data between such varying environments requires a platform-and-language-neutral data format that all systems understand.

Once the serialized data is transmitted from the source machine to the destination machine, the reverse process of creating objects from the byte sequence called deserialization is carried out. Deserialization is the reverse process of turning a byte stream back into a series of structured objects. Reconstructed objects are clones of the original object.

Choice of data serialization format for an application depends on factors such as data complexity, need for human readability, speed and storage space constraints. XML, JSON, BSON, YAML, Message Pack, and protobuf are some commonly used data serialization formats.



(Figure)

12. *How does HDFS ensure data Integrity in Hadoop Cluster?*

Data Integrity in Hadoop is achieved by maintaining the checksum of the data written to the block. Whenever data is written to HDFS blocks, HDFS calculate the checksum for all data written and verify checksum when it will read that data. The seperate checksum will create for every dfs. bytes. per. checksum bytes of data. The default size for this property is 512 bytes. Checksum is 4 Byte long.

All data nodes are responsible to check checksum of their data. When client read data from checksum, they also check checksum. To check the data block data nodes runs a Data Block Scanner periodically to verify Block. So if corrupt data found HDFS will take replica of actual data and replace the corrupt one.

(1) Data Integrity means to make sure that no data is lost or corrupted during storage or processing of the Data.

(2) Since in Hadoop, amount of data being written or read is large in Volume, chances of data corruption are more.

(3) So in Hadoop checksum is computed when data written to the disk for the first time and again checked while reading data from the disk. If checksum matches the original checksum then it is said that data is not corrupted otherwise it is said to be corrupted.

(4) It's just data detection error.

(5) It is possible that it's the checksum that is corrupt, not the data, but this is very unlikely, because the checksum is much smaller than the data.

(6) HDFS uses a more efficient variant called CRC-32C to calculate checksum.

(7) Data Nodes are responsible for verifying the data they receive before storing the data and its checksum. Checksum is computed for the data that they receive from clients and from other Data Nodes during replication.

(8) Hadoop can heal the corrupted data by copying one of the good replica to produce the new replica which is uncorrupt replica.

(9) If a client detects an error when reading a block, it reports the bad block and the Data Nodes it was trying to read from to the Name Node before throwing a Checksum Exception. "

(10) The Name Node marks the block replica as corrupt so it doesn't direct any more clients to it or try to copy this replica to another Data Nodes.

(11) It provides copy of the block another Data Nodes which is to be replicated, so its replication factor is back at the expected level.

(12) Once this has happened, the corrupt replica is deleted.

13. *Explain in Detail about input format and output format in MapReduce.*

There are different types of MapReduce Input Format in Hadoop which are used for different purpose. Let's discuss the Hadoop Input Format types below :

(1) **File Input Format** : It is the base class for all file-based Input Formats. File Input Format also specifies input directory which has data files location. When we start a MapReduce job execution, File Input Format provides a path containing files to read.

This Input Format will read all files. Then it divides these files into one or more Input Splits.

(2) **Text Input Format** : It is the default Input Format. This Input Format treats each line of each input file as a separate record. It performs no parsing. Text Input Format is useful for unformatted data or line-based records like log files. Hence,

(a) **Key** : It is the byte offset of the beginning of the line within the file (not whole file one split). So it will be unique if combined with the file name.

(b) **Value** : It is the contents of the line. It excludes line terminators.

(3) **Key Value Text Input Format** : It is similar to Text Input Format. This Input Format also treats each line of input as a separate record. While the difference is that Text Input Format treats entire line as the value, but the Key Value Text Input Format breaks the line itself into key and value by a tab character ('/t'). Hence,

(a) **Key** : Everything up to the tab character.

(b) **Value** : It is the remaining part of the line after tab character.

(4) **Sequence File Input Format** : It is an Input Format which reads sequence files. Sequence files are binary files. These files also store sequences of binary key-value pairs. These are block-compressed and provide direct serialization and deserialization of several arbitrary data. Hence, Key & Value both are user-defined.

(5) **Sequence File As Text Input Format** : It is the variant of Sequence File Input Format. This format converts the sequence file key values to Text objects. So, it performs conversion by calling 'to string()' on the keys and values. Hence, Sequence File As Text Input Format makes sequence files suitable input for streaming.

(6) **Sequence File As Binary Input Format** : By using Sequence File Input Format we can extract the sequence file's keys and values as an opaque binary object.

(7) **N line Input Format** : It is another form of Text Input Format where the keys are byte offset of the line. And values are contents of the line. So, each mapper receives a variable number of lines of input with Text Input Format and Key Value Text Input Format.

The number depends on the size of the split. Also, depends on the length of the lines. So, if want our mapper to receive a fixed number of lines of input, then we use N Line Input Format.

N- It is the number of lines of input that each mapper receives.

By default (N=1), each mapper receives exactly one line of input.

Suppose N = 2, then each split contains two lines. So, one mapper receives the first two Key-Value pairs. Another mapper receives the second two key-value pairs.

(8) **DB Input Format** : This Input Format reads data from a relational database, using JDBC. It also loads small datasets, perhaps for joining with large datasets from HDFS using Multiple Inputs. Hence,

Key : Long Writables

Value : DB Writables.

Output format of MapReduce

There are different types of MapReduce Output Format in Hadoop which are used for different purpose. Let's discuss the Hadoop Output Format types below :

(1) **Text Output Format :** MapReduce default Hadoop reducer Output Format is Text Output Format, which writes (key, value) pairs on individual lines of text files and its keys and values can be of any type since Text Output Format turns them to string by calling to String() on them. Each key-value pair is separated by a tab character, which can be changed using MapReduce.output.textoutputformat.separator property. Key Value Text Output Format is used for reading these output text files since it breaks lines into key-value pairs based on a configurable separator.

(2) **Sequence File Output Format :** It is an Output Format which writes sequences files for its output and it is intermediate format use between MapReduce jobs, which rapidly serialize arbitrary data types to the file; and the corresponding Sequence File Input Format will deserialize the file into the same types and presents the data to the next mapper in the same manner as it was emitted by the previous reducer, since these are compact and readily compressible. Compression is controlled by the static methods on Sequence File Output Format.

(3) **Sequence File As Binary Output Format :** It is another form of Sequence File Input Format which writes keys and values to sequence file in binary format.

(4) **Map File Output Format :** It is another form of File Output format in Hadoop Output Format, which is used to write output as map files. The key in a Map File must be added in order, so we need to ensure that reducer emits keys in sorted order.

(5) **Multiple Outputs :** It allows writing data to files whose names are derived from the output keys and values, or in fact from an arbitrary string.

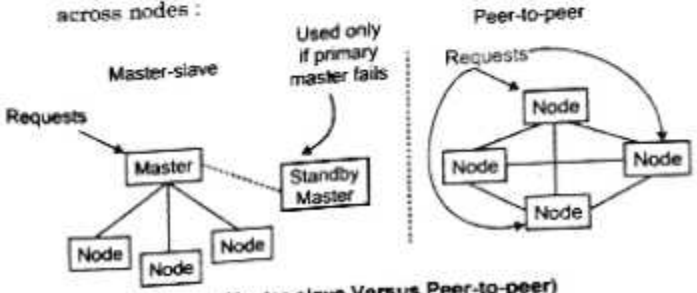
(6) **Lazy Output Format :** Sometimes File output Format will create output files, even if they are empty. the record is emitted for a given partition. Lazy Output Format is a wrapper Output Format which ensures that the output file will be created only when

(7) **DB output Format :** DB Output Format in Hadoop is an Output Format for writing to relational databases and HBase. It sends the reduce output to a SQL table. It accepts key-value pairs, where the key has a type extending DB writable. Returned Record Writer writes only the key to the database with a batch SQL query.

14. *Explain Master slave and peer- peer replication in detail.*

Replication

Means keeping a copy of the same data on multiple machines that are connected via a network. Replication can be synchronous or asynchronous. It keeps data geographically close to your users, thus reducing access latency. It allows the system to continue working even if some of its parts have failed, thus increasing availability. By Scales out the number of machines that can serve read queries, thus increasing read throughput. Here are the following algorithms for replicating changes across nodes :



(Figure : Master-slave Versus Peer-to-peer)

Master Slave Replication : When a client wants to rea from the database, it can query either the master or any o the slaves. However, writes are only accepted on th master. Often, leader-based replication is configured to b completely asynchronous. In this case, if the master fail and is not recoverable, any writes that have not yet bee replicated to slaves are lost. This means that a write is no guaranteed to be durable. Weakening durability is a bad trade-off.

Master and Slaves Posses the Following Features :

Master :

(1) Is the authoritative source for the data.

(2) Is responsible for processing any updates to that data.

(3) Can be appointed manually or automatically.

Slaves :

(1) A replication process synchronizes the slaves with the master.

(2) After a failure of the master, a slave can be appointed as new 6 master very quickly.

The Master slave algorithm has some advantages and disadvantages as well. Some of them are given below.

Advantages :

(1) More read requests:

(2) Add more slave nodes

(3) Ensure that all read requests are routed to the slaves

(4) Should the master fail, the slaves can still handle read requests

(5) Good for datasets with a read-intensive dataset

Disadvantages :

(1) The master is a bottleneck

(2) Limited by its ability to process updates and to pass those updates on

(3) Its failure does eliminate the ability to handle writes until :

(a) The master is restored or

(b) A new master is appointed

(c) Inconsistency due to slow propagation of changes to the slaves

(d) Bad for datasets with heavy write traffic

Peer- Peer Replication : Peer-to-peer replication provides a scale-out and high-availability solution by maintaining copies of data across multiple server instances, also referred to as nodes. It was introduced in SQL Server 2005. Peer-to-Peer Transactional Replication is typically used to support applications that distribute read operations across a number of server nodes. When we configure peer-to-peer between the servers, all the inserts, updates and deletes are propagated to the other nodes when the command is executed. Thus all the nodes are updated and the databases stay synchronized in near real-time. All the replicas have equal weight, they can all accept writes. The loss of any of them doesn't prevent access to the data store. Because data is maintained across the nodes in near real-time, peer-to-peer replication provides data redundancy, which increases the availability of data.

Peer-Peer method also has some advantages and disadvantages as well just like Master slave method. Some of them are listed given below.

Advantages :

(1) You can ride over node failures without losing access to data

(2) You can easily add nodes to improve your performance

Disadvantages :

(1) Inconsistency!

(2) Slow propagation of changes to copies on different nodes

(3) Inconsistencies on read lead to problems but are relatively transient

(4) Two people can update different copies of the same record stored on different nodes at the same time - a write-write conflict

(5) Inconsistent writes are forever.

15. *What are the applications of Data Serialization?*

Serialization allows a program to save the state of an object and recreate it when needed. Its common uses are :

(1) **Persisting Data Onto Files :** Happens mostly language-neutral formats such as CSV or XM However, most languages allow objects to serialized directly into binary using APIs such as t Serializable interface in Java, fstream class in C, or Pickle module in Python.

(2) **Storing Data into Databases :** When progra objects are converted into byte streams and th stored into DBs, such as in Java JDBC.

(3) **Transferring Data Through the Network :** Su as web applications and mobile apps passing objects from client to server and vice versa.

(4) **Remote Method Invocation (RMI) :** By passin serialized objects as parameters to functions runnin on a remote machine as if invoked on a local machin This data can be transmitted across domains throug firewalls.

(5) **Sharing data in a Distributed Object Model** When programs written in different language (running on diverse platforms) need to share objec data over a distributed network using framework such as COM and CORBA. However, SOAP, REST and other web services have replaced these applications now.

16. *What are the potential risks that affect data due to serialization?*

Naive use of object serialization may allow a malicious party with access to the serialization byte stream to read private data, create objects with illegal or dangerous state, or obtain references to the private fields of deserialized objects. Workarounds are tedious, not guaranteed. Oracle has promised to do away with object serialization and allow programmers to choose from open formats such as JSON, XML, etc.

Open formats too have their security issues. XML might be tampered using external entities like macros or unverified DTD schema files. JSON data is vulnerable to

attack when directly passed to a JavaScript engine due to features like JSONP requests. Programmers need to take care of such security vulnerabilities in their code, before deploying their applications.

17. *Which are the programming languages that offer good serialization support?*

Platform-independent languages offer better support for data serialization. Java, the pioneer of object serialization, supports in-built serialization (Serializable Interface) and custom serialization (Externalizable Interface). Python has newly introduced the Pickle module for the same.

However, real-world environment is diverse and cross-platform. Most web applications are opting for platform-neutral data serialization formats. Open source formats JSON, XML, YAML, protobuf, Message Pack have support for all major programming languages such as C, C++, Java, Python, Perl, Go, etc. So the choice of language depends more on other factors, not on serialization any more.

18. ◆ *Explain Big data and algorithmic trading.* (2015-16)

◆ ***What is algorithmic Trading and How Big Data can help that?***

Algorithmic Trading is also called as automated trading or the black-box trading, or Algorithm based Trading, which mean you can write automated algorithms which can place trade for you, totally depends on how you write the algorithms so that trade can be perfect as your algorithm is. The trade can generate profits at a speed and frequency that is impossible for a human trader to trade or take the decision.

It involves a lot of mathematics calculations on the historical data which can be anything, lets consider the data for the stock market for a particular share named "HDFC" we can get the historical data from the API's

available that will help us to know the behaviour of that particular share like when it was low, what was high and opening price and all the things, that unstructured data will help us to analyse the data so that we can predict that how will it be in future what will be the highs and lows of that share. So for those mathematical concepts like moving mean, stochastic etc can help us to tell us what will be the behaviour of that share. Lets understand it with a better example :

Suppose a trader follows these simple trade criteria :

Buy 50 shares of a stock when its 50-day moving average goes above the 200-day moving average. (A moving average is an average of past data points that smooths out day-to-day price fluctuations and thereby identifies trends.)Sell shares of the stock when its 50-day moving average goes below the 200-day moving average. Using above two simple instructions, a computer program will automatically monitor the stock price (and the moving average indicators) and place the buy and sell orders when the defined conditions are met. The trader no longer needs to monitor live prices and graphs or put in the orders manually. The algorithmic trading system does this automatically by correctly identifying the trading opportunity.

19. *Define data locality optimization of Map Reduce in detail?* (2016-17)

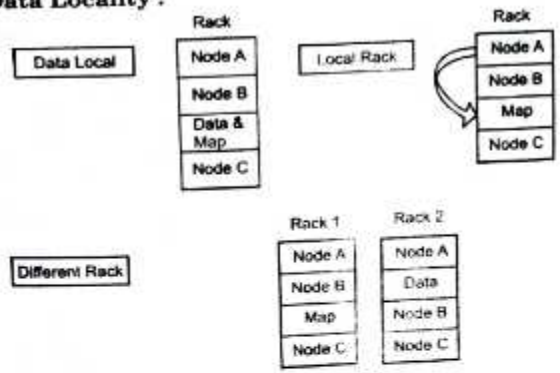
Data locality in Hadoop is the process of moving the computation close to where the actual data resides instead of moving large data to computation. This minimizes overall network congestion. This also increases the overall throughput of the system. The main drawback of Hadoop was cross-switch network traffic due to the huge amount of data. To overcome this drawback, Data Locality came into existence. Since Data locality is the main advantage of Hadoop MapReduce. But this is not always beneficial in practice due to various reasons like Heterogeneous cluster, speculative execution, Data distribution and placement,

and Data Layout. In large clusters challenges become more prevalent. As in large cluster more the number of data nodes and data, the less is the locality. In larger clusters, some nodes are newer and faster than the other, creating the data to compute ratio out of balance. Thus, large clusters tend not be completely homogenous. In Hadoop speculative execution since the data might not be local, but it uses the compute power. The main cause also lies in the data layout/placement. Also non-local data processing puts a strain on the network, which creates problem to scalability. Therefore the network becomes the bottleneck.

We can also improve data locality by first detecting which jobs have degrade over time or data locality problem. Problem-solving is more complex and involves changing the data placement and data layout using a different scheduler.

After that we have to verify whether a new execution of the same workload has a better data locality ratio.

Data Locality :



(Figure)

Types of Data Locality in Hadoop : The various categories in Hadoop Data Locality are as follows:

(1) Data Local Data Locality in Hadoop : In this, data is located on the same node as the mapper working on the data. In this, the proximity of data is very near to computation. Data local data locality is the most preferred scenario.

(2) **Intra-Rack Data Locality in Hadoop :** As
know that it's not always possible to execute t
mapper on the same datanode due to resou
constraints. In this case, it is preferred to run t
mapper on the different node but on the same rack.

(3) **Inter-Rack Data Locality in Hadoop :** Sometim
it is also not possible to execute mapper on a differe
node in the same rack. In such situation, we w
execute the mapper on the nodes on different rac
Inter -rack data locality is the least prefer
scenario.

Requirement for Hadoop Data Locality : Hado
architecture needs to satisfy below conditions to get t
benefits of all the advantages of data locality:

First, Hadoop cluster should have the appropria
topology. The Hadoop code should have the ability to rea
data locality.

Second, Apache Hadoop should be aware of th
topology of the nodes where tasks are executed. Als
Hadoop should know where the data is located

Advantages of Data Locality in Hadoop :

High Throughput : Data locality in Hadoop increases th
overall throughput of the system.

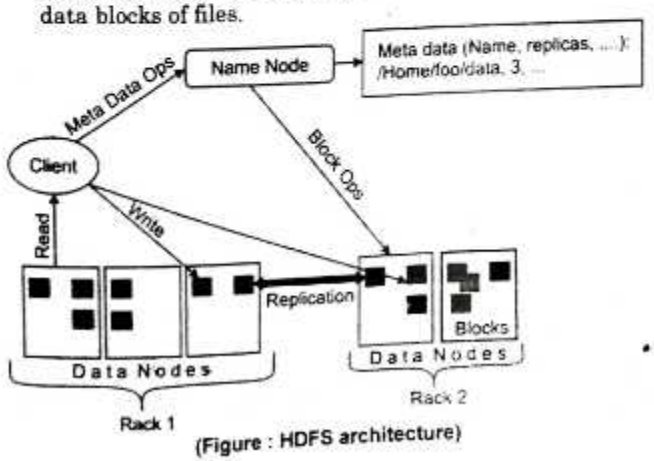
Faster Execution : In data locality, framework move cod
to the node where data resides instead of moving larg
data to the node. Thus, this makes Hadoop faster. Becaus
the size of the program is always lesser than the size o
data, so moving data is a bottleneck of network transfer.

HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

1. ***Draw and explain HDFS architecture. Explain the function of Name node, Data node. What is secondary Name node? Is it a substitute of Name node?*** **(2018-19)**

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

Hadoop Distributed File System follows the master-slave architecture. Each cluster comprises a single master node and multiple slave nodes. Internally the files get divided into one or more blocks, and each block is stored on different slave machines depending on the replication factor .The master node stores and manages the file system namespace, that is information about blocks of files like block locations, permissions, etc. The slave nodes store data blocks of files.



(Figure : HDFS architecture)

The Master node is the Name Node and Data Nod are the slave nodes.

Name Node : Name Node is the centerpiece of the Hado Distributed File System. It maintains and manages the f system namespace and provides the right acce permission to the clients.

The Name Node stores information about bloc locations, permissions, etc. on the local disk in the form two files :

(1) **Fsimage** : Fsimage stands for File System image. contains the complete namespace of the Hadoop f system since the Name Node creation.

(2) **Edit Log** : It contains all the recent chang performed to the file system namespace to the mo recent Fsimage.

Functions of Name Node

(1) It executes the file system namespace operation like opening, renaming, and closing files an directories.

(2) Name Node manages and maintains the Data Nodes.

(3) It determines the mapping of blocks of a file to Dat Nodes.

(4) Name Node records each change made to the fil system namespace.

(5) It keeps the locations of each block of a file.

(6) Name Node takes care of the replication factor of al the blocks.

(7) Name Node receives heartbeat and block reports fro all Data Nodes that ensure Data Node is alive.

(8) If the Data Node fails, the Name Node chooses ne Data Nodes for new replicas.

Before Hadoop2, Name Node was the single point o failure. The High Availability Hadoop cluster architectur introduced in Hadoop 2, allows for two or more Nam Nodes running in the cluster in a hot standb configuration.

Data Node : Data Nodes are the slave nodes in Hadoo HDFS. Data Nodes are inexpensive commodity hardware. They store blocks of a file.

Functions of Data Node

(1) Data Node is responsible for serving the client read/write requests.

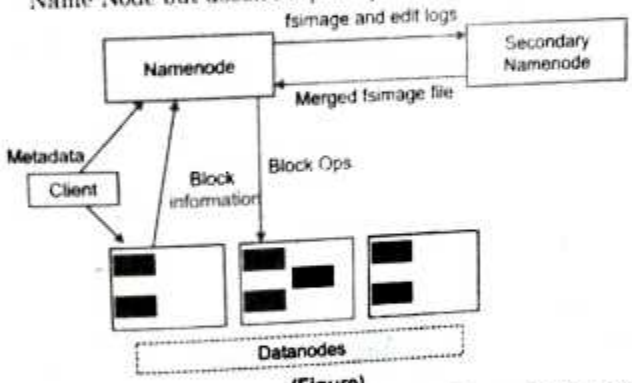
(2) Based on the instruction from the Name Node, Data Nodes performs block creation, replication, and deletion.

(3) Data Nodes send a heartbeat to Name Node to report the health of HDFS.

(4) Data Nodes also sends block reports to Name Node to report the list of blocks it contains.

Secondary Name Node

Apart from Data Node and Name Node, there is another daemon called the secondary Name Node. Secondary Name Node works as a helper node to primary Name Node but doesn't replace primary Name Node.



(Figure)

When the Name Node starts, the Name Node merges the Fsimage and edit logs file to restore the current file system namespace. Since the Name Node runs continuously for a long time without any restart, the size of edit logs becomes too large. This will result in a long restart time for Name Node. Secondary Name Node solves this issue. Secondary Name Node downloads the Fsimage file and edit logs file from Name Node. It periodically applies edit logs to Fsimage and refreshes the edit logs. The updated Fsimage is then sent to the Name Node so that Name Node doesn't have to re-apply the edit log records during its restart. This keeps the edit log size small and reduces the Name Node restart time.

If the Name Node fails, the last save Fsimage on th secondary Name Node can be used to recover file syste metadata. The secondary Name Node performs regula checkpoints in HDFS.

2. *What is Rack Awareness in HDFS Architecture?*

Rack is the collection of around 40-50 machine (Data Nodes) connected using the same network switch. I the network goes down, the whole rack will be unavailable

Rack Awareness is the concept of choosing the closest node based on the rack information.

To ensure that all the replicas of a block are not stored on the same rack or a single rack, Name Node follows a rack awareness algorithm to store replicas and provide latency and fault tolerance.

Suppose if the replication factor is 3, then according to the rack awareness algorithm :

(1) The first replica will get stored on the local rack.

(2) The second replica will get stored on the other Data Node in the same rack.

(3) The third replica will get stored on a different rack.

3. *What is Backup Node?*

Backup node provides the same check pointing functionality as the Checkpoint node. In Hadoop, Backup node keeps an in-memory, up-to-date copy of the file system namespace. It is always synchronized with the active Name Node state.

It is not required for the backup node in HDFS architecture to download Fsimage and edits files from the active Name Node to create a checkpoint. It already has an up-to-date state of the namespace state in memory.

The Backup node checkpoint process is more efficient as it only needs to save the namespace into the local Fsimage file and reset edits. Name Node supports one Backup node at a time.

4. *What is the purpose of bloom filter?* (2016-17)

A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. For example, checking availability of username is set membership problem, where the set is the list of all registered username. The price we pay for efficiency is that it is probabilistic in nature that means, there might be some False Positive results. False positive means, it might tell that given username is already taken but actually it's not.

Transaction bloom filtering is a method used in Bitcoin that allows lightweight clients to limit the amount of transaction data they receive from full nodes to only those transactions that affect their wallet (plus a configurable amount of additional transactions to generate plausible deniability about which transactions belong to the client).

Interesting Properties of Bloom Filters :

(1) Unlike a standard hash table, a Bloom filter of a fixed size can represent a set with an arbitrarily large number of elements.

(2) Adding an element never fails. However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, at which point all queries yield a positive result.

(3) Bloom filters never generate false negative result, i.e., telling you that a username doesn't exist when it actually exists.

(4) Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements. Example – if we delete "geeks" (in given example below) by clearing bit at 1, 4 and 7, we might end up deleting "nerd" also Because bit at index 4 becomes 0 and bloom filter claims that "nerd" is not present.

5. *Mention the uses of Grunt.* (2016-17)

Grunt is a command-line JavaScript task runner utilizing the Node.js platform. A tool used to automatically perform frequent tasks, therefore, automating these tasks means a massive increase in productivity.

If you are developing websites, then Grant wi almost always be an enhancement for you. Grunts are al a good option for things like websites.

Advantages of Grunt :

(1) Multiple file size decrease on the folder then aft page load fast on browser

(2) Grunt plugin using, your perform minificatio compilation, and files easily testing.

(3) Easily work with a new codebases use Grunt plug because it contains less infrastructure.

(4) It speeds up the development workflow and enhance the performance of projects.

Disadvantages of Grunt :

(1) Every time npm packages are updated, you need t wait until the writer of the Grunt plugin update it.

(2) All task is designed to do specified work. If you wan to extend a specified task, then you need to use som tricks to get the work done.

(3) Grunt plugins cover large data of configuratio parameters for separate plugins, normally, Grun plugin configuration files are longer.

6. *Illustrate on how cloud and Big data related to each other?* **(2017-18)**

"Big Data" refers to the large sets of data collected while "Cloud Computing" refers to the mechanism tha remotely takes this data in and performs any operation specified on that data.

Cloud Computing providers often utilize a "softwar as a service" model to allow customers to easily proces data. Typically, a console that can take in specialize commands and parameters is available, but everything can also be done from the site's user interface. Some product that are usually part of this package include database management systems, cloud-based virtual machines and containers, identity management systems, machine learning capabilities, and more. In turn, Big Data is often generated by large, network-based systems. It can be in either a standard or non-standard format. If the data is in a non-standard format, artificial intelligence from the

Cloud Computing provider may be used in addition to machine learning to standardize the data.

From there, the data can be harnessed through the Cloud Computing platform and utilized in a variety of ways. For example, it can be searched, edited, and used for future insights. This cloud infrastructure allows for real-time processing of Big Data. It can take huge "blasts" of data from intensive systems and interpret it in real-time. Another common relationship between Big Data and Cloud Computing is that the power of the cloud allows Big Data analytics to occur in a fraction of the time it used to.

However, Cloud Computing allows us to use state-of-the-art infrastructure and only pay for the time and power that we use! Cloud application development is also fueled by Big Data. Without Big Data, there would be far fewer cloud-based applications, since there wouldn't be any real necessity for them. Remember, Big Data is often collected by cloud-based applications, as well!

In short, Cloud Computing services largely exist because of Big Data. Likewise, the only reason that we collect Big Data is because we have services that are capable of taking it in and deciphering it, often in a matter of seconds. The two are a perfect match, since neither would exist without the other.

7. *What is a 'block' in HDFS? Explain the benefits of block transfer?*

A 'block' is the minimum amount of data that can be read or written. In HDFS, the default block size is 64 MB as contrast to the block size of 8192 bytes in Unix/Linux. Files in HDFS are broken down into block-sized chunks, which are stored as independent units. HDFS blocks are large as compared to disk blocks, particularly to minimize the cost of seeks. If a particular file is 50 mb, will the HDFS block still consume 64 mb as the default size? No, not at all! 64 mb is just a unit where the data will be stored. In this particular situation, only 50 mb will be consumed by an HDFS block and 14 mb will be free to store something else. It is the Master Node that does data allocation in an efficient manner.

A file can be larger than any single disk in the network. There's nothing that requires the blocks from a file to be stored on the same disk, so they can take advantage of any of the disks in the cluster. Making the unit of abstraction a block rather than a file simplifies the storage subsystem. Blocks provide fault tolerance and availability. To insure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three). If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client.

8. *What are Problems with small files and HDFS?*

HDFS is not good at handling large number of small files. Because every file, directory and block in HDFS is represented as an object in the name node's memory, each of which occupies approx 150 bytes. So 10 million files, each using a block, would use about 3 gigabytes of memory. When we go for a billion files the memory requirement in name node cannot be met.

9. *Why is Check pointing Important in Hadoop?*

As more and more files are added the name node creates large edit logs. Which can substantially delay Name Node startup as the Name Node reapplies all the edits. Check pointing is a process that takes an fsimage and edit log and compacts them into a new fsimage. This way, instead of replaying a potentially unbounded edit log, the Name Node can load the final in-memory state directly from the fsimage. This is a far more efficient operation and reduces Name Node startup time.

10. *How data Organization done in HDFS? Explain in brief.*

Data Organization

Data organization is an important task of HDFS. HDFS perform it by splitting data into chunks as it is designed to support very large files. Following steps are required for data organization :

(1) **Data Blocks** : Applications that are compatible with HDFS are those that deal with large data sets. These applications write their data only once but they read it one or more times and require these reads to be satisfied at streaming speeds. HDFS supports write-once-read-many semantics on files. A typical block size used by HDFS is 64 MB. Thus, an HDFS file is chopped up into 64 MB chunks, and if possible, each chunk will reside on a different Data Node.

(2) **Staging** : A client request to create a file does not reach the Name Node immediately. In fact, initially the HDFS client caches the file data into a temporary local file. Application writes are transparently redirected to this temporary local file. When the local file accumulates data worth over one HDFS block size, the client contacts the Name Node. The Name Node inserts the file name into the file system hierarchy and allocates a data block for it. The Name Node responds to the client request with the identity of the Data Node and the destination data block. Then the client flushes the block of data from the local temporary file to the specified Data Node. When a file is closed, the remaining un-flushed data in the temporary local file is transferred to the Data Node. The client then tells the Name Node that the file is closed. At this point, the Name Node commits the file creation operation into a persistent store. If the Name Node dies before the file is closed, the file is lost.

The above approach has been adopted after careful consideration of target applications that run on HDFS. These applications need streaming writes to files. If a client writes to a remote file directly without any client side buffering, the network speed and the congestion in the network impacts throughput considerably. Thus approach is not without precedent. Earlier distributed file systems, e.g. AFS, have used client side caching to improve performance. A POSIX requirement has been relaxed to achieve higher performance of data uploads.

(3) **Replication Pipelining** : When a client is writing data to an HDFS file, its data is first written to a local file as explained in the previous section.

Suppose the HDFS file has a replication factor of three. When the local file accumulates a full block of user data, the client retrieves a list of Data Nodes from the Name Node. This list contains the Data Nodes that will host a replica of that block. The client then flushes the data block to the first Data Node. The first Data Node starts receiving the data in small portions (4 KB), writes each portion to its local repository and transfers that portion to the second Data Node in the list. The second Data Node, in turn starts receiving each portion of the data block, writes that portion to its repository and then flushes that portion to the third Data Node. Finally, the third Data Node writes the data to its local repository. Thus, a Data Node can be receiving data from the previous one in the pipeline and at the same time forwarding data to the next one in the pipeline. Thus, the data is pipelined from one Data Node to the next.

11. *How Robustness is ensuring in HDFS? Explain in detail.*

Robustness

The main objective of Hadoop is to store data reliably even in the event of failures. Various kind of failure is Name Node failure, Data Node failure, and network partition. Data Node periodically sends a heartbeat signal to Name Node. In-network partition, a set of Data Nodes gets disconnected with the Name Node. Thus Name Node does not receive any heartbeat from these Data Nodes. It marks these Data Nodes as dead. Also, Name node does not forward any I/O requests to them. The replication factor of the blocks stored in these Data Nodes falls below their specified value. As a result, Name Node initiates the replication of these blocks. In this way, Name Node recovers from the failure.

Data Disk Failure, Heartbeats and Re-Replication :

Each Data Node sends a Heartbeat message to the Name Node periodically. A network partition can cause a subset of Data Nodes to lose connectivity with the Name Node. The Name Node detects this condition by the absence of a Heartbeat message. The Name Node marks

Data Nodes without recent Heartbeats as dead and does not forward any new IO requests to them. Any data that was registered to a dead Data Node is not available to HDFS anymore. Data Node death may cause the replication factor of some blocks to fall below their specified value. The Name Node constantly tracks which blocks need to be replicated and initiates replication whenever necessary. The necessity for re-replication may arise due to many reasons: a Data Node may become unavailable, a replica may become corrupted, a hard disk on a Data Node may fail, or the replication factor of a file may be increased.

Cluster Rebalancing : The HDFS architecture is compatible with data rebalancing schemes. A scheme might automatically move data from one Data Node to another if the free space on a Data Node falls below a certain threshold. In the event of a sudden high demand for a particular file, a scheme might dynamically create additional replicas and rebalance other data in the cluster. These types of data rebalancing schemes are not yet implemented.

Data Integrity : It is possible that a block of data fetched from a Data Node arrives corrupted. This corruption can occur because of faults in a storage device, network faults, or buggy software. The HDFS client software implements checksum checking on the contents of HDFS files. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace. When a client retrieves file contents it verifies that the data it received from each Data Node matches the checksum stored in the associated checksum file. If not, then the client can opt to retrieve that block from another Data Node that has a replica of that block.

Metadata Disk Failure : The FsImage and the Edit Log are central data structures of HDFS. A corruption of these files can cause the HDFS instance to be non-functional. For this reason, the Name Node can be configured to support maintaining multiple copies of the FsImage and EditLog. Any update to either the FsImage or Edit Log causes each of the FsImages and Edit Logs to get updated synchronously. This synchronous updating of multiple

copies of the FsImage and Edit Log may degrade the ra of namespace transactions per second that a Name No can support. However, this degradation is acceptab because even though HDFS applications are very da intensive in nature, they are not metadata intensive. Wh a Name Node restarts, it selects the latest consiste FsImage and Edit Log to use.

The Name Node machine is a single point of failu for an HDFS cluster. If the Name Node machine fai manual intervention is necessary. Currently, automat restart and failover of the Name Node software to anoth machine is not supported.

Snapshots : Snapshot is nothing but storing a copy of da at a particular instance of time. One of the usages of th snapshot is to rollback a failed HDFS instance to a goo point in time. We can take Snapshots of the sub-tree of th file system or entire file system. Some of the uses s snapshots are disaster recovery, data backup, an protection against user error. We can take snapshots o any directory. Only the specific directory should be set as Snapshot table. The administrators can set any directory as a snapshot table. We cannot rename or delete a snapshot table directory if there are snapshots in it. After removing all the snapshots from the directory, we can rename or delete it.

12. *Explain Communication Protocols of HDFS in detail.*

The HDFS communication protocol works on the top of the TCP/IP protocol. The client establishes a connection with Name Node using configurable TCP port. Hadoop cluster creates a connection to the client using client protocol. Data Node talks to Name Node using the Data Node Protocol. A Remote Procedure Call (RPC) abstraction wraps both Client protocol and Data Node protocol. Name Node does not initiate any RPC instead it responds to RPC from the Data Node.

Communication Among HDFS Elements

This section describes the communication among the HDFS elements. The application process has two elements for which we will discuss the communication the

application code and the client library. We will also discuss the communication between the various processes. Each of the three elements of HDFS is described in a different section, see Name Node Decomposition, Client decomposition, and Data Node Block Management.

Communication Between Application Code And Client : HDFS provides a Java API for applications to use. Fundamentally, the application uses the standard java.io interface. A C language wrapper for this Java API is also available.

The client and the application code are bound into the same address space.

Communication between Client and Name Node : The connection between Name Node and the client is governed by the Client Protocol documented in ...\hdfs\protocol\ClientProtocol.java. A client establishes a connection to a configurable TCP port on the Name Node machine. This is an RPC connection. The communication between the Client and the Name Node uses the Client Protocol.

The Major Functions of the Client Protocol are :

(1) **Create :** Create a new file in the name space

(2) **Append :** After a file has been created and closed, it is still possible to add data to the end of the file. This is the purpose of the append function

(3) **Add Block :** Add a new block to an existing file

(4) **Complete : (close) :** The client has finished writing to this file.

(5) **Read :** A client wishes to read from the file

(6) **Error Reporting :** Report bad blocks detected by the client

(7) **Lease Management :** Leases are created automatically on writing and renewed explicitly by the client.

(8) **Book Keeping :** Affect the state of the Name Node, check on Data Nodes, get a list of files in a particular directory

(9) **Directory Management :** rename, delete, copy

Communication between Client and Data Node : A client communicates with a Data Node directly to transfer (send/receive) data using the Data Transfer Protocol, defined in DataTransferProtocol.java. For performance

15. *How will you measure HDFS space consumed?*

The two popular utilities or commands to measu HDFS space consumed are hdfs dfs –du and hdfs dfs adm –report. HDFS provides reliable storage by copying data multiple nodes. The number of copies it creates is usual referred to as the replication factor which is greater tha one.

(1) **hdfs dfs du** : This command shows the spa consumed by data without replications.

(2) **hdfs dfs admin : Report** : This command shows th real disk usage by considering data replicatio Therefore, the output of hdfs dfsadmin –report wi always be greater than the output of hdfs dfs –d command.

16. *Replication causes data redundancy then why is it still preferred in HDFS?*

As we know that Hadoop works on commodit hardware, so there is an increased probability of gettin crashed. Thus to make the entire Hadoop system high tolerant, replication factor is preferred even though creates multiple copies of the same data at differen locations. Data on HDFS is stored in at least 3 differen locations. Whenever one copy of the data is corrupted an the other copy of the data is not available due to som technical glitches then the data can be accessed from th third location without any data loss.

17. *Where are the two types of metadata that Name Node server stores?*

The two types of metadata that Name Node serve stores are in Disk and RAM.

Metadata is linked to two files which are :

(1) **EditLogs** : It contains all the latest changes in th file system regarding the last FsImage.

(2) **FsImage** : It contains the whole state of th namespace of the file system from the origination o the Name Node.

Once the file is deleted from HDFS, the Name Node will immediately store this in the Edit Log. All the file systems and metadata which are present in the Name node's Ram are read by the Secondary Name Node continuously and later get recorded into the file system or hard disk. Edit Logs is combined with FsImage in the Name Node. Periodically, Secondary Name Node downloads the Edit Logs from the Name Node, and then it is implemented to FsImage. The new FsImage is then copied back into the Name Node and used only after the Name Node has started the subsequent time.

18. *What does the HDFS error "File could only be replicated to 0 nodes, instead of 1" mean?*

This exception occurs when the Data Node is not available to the Name Node (i.e. the client is not able to communicate with the Data Node) due to one of the following reasons :

(1) In hdfs-site.xml file, if the block size is a negative value.

(2) If there are any network fluctuations between the Data Node and Name Node, as a result of which the primary Data Node goes down whilst write is in progress.

(3) Disk of Data Node is full.

(4) Data Node is eventful and occupied with block reporting and scanning.

19. *What do you understand by Safe Mode in Hadoop? How will you manually enter and leave Safe Mode in Hadoop?*

The state in which Name Node does not perform replication or deletion of blocks is referred to as Safe Mode in Hadoop. In safe mode, Name Node only collects block reports information from the Data Nodes.

Below command is used to enter Safe Mode manually :

```
$ Hdfs dfsadmin -safe mode enter
```

Once the safe mode is entered manually, it should be removed manually.

Below command is used to leave Safe Mode manually :

$ hdfs dfsadmin -safe mode leave

20. *How will you reduce the size of large cluster by removing a few nodes in HDFS?*

A set of existing nodes can be removed using the decommissioning feature to reduce the size of a large cluster. The nodes that have to be removed should be added to the exclude file. The name of the exclude file should be stated as a config parameter dfs.hosts.exclude. By editing the exclude files or the configuration file, the decommissioning process can be ended.

21. *What do you understand by Active and Passive Name Nodes? How will you balance the disk space usage on a HDFS cluster?*

The Name Node that works and runs in the Hadoop cluster is often referred to as the Active Name Node. Passive Name Node also known as Standby Name Node is the similar to an active Name Node but it comes into action only when the active Name Node fails. Whenever the active Name Node fails, the passive Name Node or the standby Name Node replaces the active Name Node, to ensure that the Hadoop cluster is never without a Name Node.

Balancer tool helps achieve this by taking a threshold value as input parameter which is always a fraction between 0 and 1. The HDFS cluster is said to be balanced, if, for every Data Node, the ratio of used space at the node to total capacity of the node differs from the ratio of used space in the cluster to total capacity of the cluster - is not greater than the threshold value.

22. *Replication causes data redundancy then why is it still preferred in HDFS?*

As we know that Hadoop works on commodity hardware, so there is an increased probability of getting crashed. Thus to make the entire Hadoop system highly

tolerant, replication factor is preferred even though it creates multiple copies of the same data at different locations. Data on HDFS is stored in at least 3 different locations. Whenever one copy of the data is corrupted and the other copy of the data is not available due to some technical glitches then the data can be accessed from the third location without any data loss.

23. *Data is replicated at least thrice on HDFS. Does it imply that any alterations or calculations done on one copy of the data will be reflected in the other two copies also?*

Calculations or any transformations are performed on the original data and do not get reflected to all the copies of data. Master node identifies where the original data is located and performs the calculations. Only if the node is not responding or data is corrupted then it will perform the desired calculations on the second replica.

24. *What are the Limitations of Hadoop 1.0?*

There are many limitations of Hadoop 1.0 some of them are listed below :

(1) Only one Name Node is possible to configure.
(2) Secondary Name Node was to take hourly backup of Meta Data from Name Node.
(3) It is only suitable for Batch Processing of a vast amount of Data, which is already in the Hadoop System.
(4) It is not ideal for Real-time Data Processing.
(5) It supports up to 4000 Nodes per Cluster.
(6) It has a single component: Job Tracker to perform many activities like Resource Management, Job Scheduling, Job Monitoring, Re-scheduling Jobs etc.
(7) Job Tracker is the single point of failure.
(8) It supports only one Name No and One Namespace per Cluster.
(9) It does not help the Horizontal Scalability of NameNode.
(10) It runs only Map/Reduce jobs.

25. *Explain Different Features of Hadoop Distributed File System in brief*

Features of HDFS

Hadoop helps to analyze vast amounts of data parallelly and more swiftly. Apache Hadoop was acquainted with the public in 2012 by The Apache Software Foundation (ASF). Hadoop is economical to use as data is stored on affordable commodity Servers that run as clusters. Some of the key features of Hadoop are discussed below :

(1) **Fault Tolerance :** Hadoop framework divides data into blocks and creates various copies of blocks on several machines in the cluster. So, when any device in the cluster fails, clients can still access their data from the other machine containing the exact copy of data blocks.

(2) **High Availability :** In the HDFS environment, the data is duplicated by generating a copy of the blocks. So, whenever a user wants to obtain this data, or in case of an unfortunate situation, users can simply access their data from the other nodes because duplicate images of blocks are already present in the other nodes of the HDFS cluster.

(3) **High Reliability :** HDFS splits the data into blocks, these blocks are stored by the Hadoop framework on nodes existing in the cluster. It saves data by generating a duplicate of every block current in the cluster. Hence presents a fault tolerance facility. By default, it creates 3 duplicates of each block containing information present in the nodes. Therefore, the data is promptly obtainable to the users. Hence the user does not face the difficulty of data loss. Therefore, HDFS is very reliable.

(4) **Replication :** Replication resolves the problem of data loss in adverse conditions like device failure, crashing of nodes, etc. It manages the process of replication at frequent intervals of time. Thus, there is a low probability of a loss of user data.

(5) **Scalability :** HDFS stocks the data on multiple nodes. So, in case of an increase in demand, it can scale the cluster.

26. *Explain the distributed Cache in Map Reduce framework in Hadoop.*

Distributed Cache is a significant feature provided by the Map Reduce Framework, practiced when you want to share the files across all nodes in a Hadoop cluster. These files can be jar files or simple properties files.

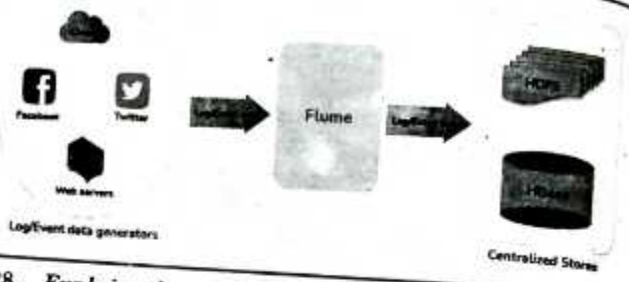
Hadoop's Map Reduce framework allows the facility to cache small to moderate read-only files such as text files, zip files, jar files, etc., and distribute them to all the Data nodes (worker-nodes) Map Reduce jobs are running. All Data node gets a copy of the file (local-copy), which is sent by Distributed Cache.

(Figure : Distributed Cache in Hadoop)

27. *What is Apache Flume in Hadoop?*

Apache Flume is a tool/service/data ingestion mechanism for assembling, aggregating, and carrying huge amounts of streaming data such as record files, events from various references to a centralized data store.

Flume is a very stable, distributed, and configurable tool. It is generally designed to copy streaming data (log data) from various web servers to HDFS.



28. *Explain the architecture of Flume with suitable diagram.*

In general Apache Flume architecture is composed of the following components :

(1) Flume Source
(2) Flume Channel
(3) Flume Sink
(4) Flume Agent
(5) Flume Event

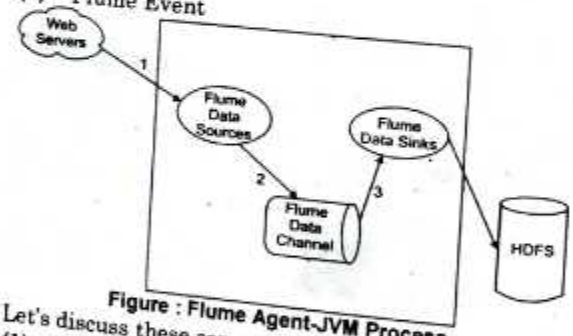


Figure : Flume Agent-JVM Process

Let's discuss these components in detail one by one :

(1) **Flume Source** : Flume Source is available on various networking platforms like Facebook or Instagram. It is a Data generator that collects data from the generator, and then the data is transferred to a Flume Channel in the form of a Flume.

(2) **Flume Channel** : The data from the flume source is sent to an Intermediate Store which buffers the events till they get transferred into Sink. The Intermediate Store is called Flume Channel. Channel is an intermediate source. It is a bridge between

Source and a Sink Flume channel. It supports both the Memory channel and File channel. The file channel is non-volatile which means once the data is entered into the channel, the data will never be lost unless you delete it. In contrast, in the Memory channel, events are stored in memory, so it's volatile, meaning data may be lost, but Memory Channel is very fast in nature.

(3) **Flume Sink** : Data repositories like HDFS have Flume Sink. Which takes Flume events from the Flume Channel and stores them into the Destination specified like HDFS? It is done in such a way where it should deliver the events to the Store or another agent. Various sinks like Hive Sink, Thrift Sink, etc are supported by the Flume.

(4) **Flume Agent** : A Java process that works on Source, Channel, Sink combination is called Flume Agent. One or more than one agent is possible in Flume. Connected Flume agents which are distributed in nature can also be collectively called Flume.

(5) **Flume Event** : An Event is the unit of data transported in Flume. The general representation of the Data Object in Flume is called Event. The event is made up of a payload of a byte array with optional headers.

29. *What are the various consequences of Distributed Applications? Explain in detail.*

(1) **Heterogeneity** : The design of applications should allow the users to access services and run applications over a heterogeneous collection of computers and networks taking into consideration Hardware devices, OS, networks, Programming languages.

(2) **Transparency** : Distributed system Designers must hide the complexity of the system as much as they can. Some Terms of transparency are location, access, migration, Relocation, and so on.

(3) **Openness** : It is a characteristic that determines whether the system can be extended and reimplemented in various ways.

(4) **Security :** Distributed system Designers must take care of confidentiality, integrity, and availability.

(5) **Scalability :** A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance.

30. *Compare the main differences between HDFS (Hadoop Distributed File System) and Network Attached Storage(NAS) ?*

HDFS	NAS
HDFS is a Distributed File system that is mainly used to store data by commodity hardware.	NAS is a file-level computer data storage server connected to a computer network that provides network access to a heterogeneous group of clients.
HDFS is programmed to work with the Map Reduce paradigm.	NAS is not suitable to work with a Map Reduce paradigm.
HDFS is Cost-effective.	NAS is a high-end storage device that is highly expensive.

31. *Explain what is Speculative Execution?*

In Hadoop during Speculative Execution, a certain number of duplicate tasks are launched. On a different slave node, multiple copies of the same map or reduce task can be executed using Speculative Execution. In simple words, if a particular drive is taking a long time to complete a task, Hadoop will create a duplicate task on another disk. A disk that finishes the task first is retained and disks that do not finish first are killed.

32. *Mention what is the difference between an RDBMS and Hadoop?*

Difference between an RDBMS and Hadoop

RDBMS	Hadoop
RDBMS is a relational database management system	Hadoop is a node based flat structure
It used for OLTP processing whereas Hadoop	It is currently used for analytical and for BIG DATA processing
In RDBMS, the database cluster uses the same data files stored in a shared storage	In Hadoop, the storage data can be stored independently in each processing node
You need to preprocess data before storing it	you don't need to preprocess data before storing it

33. *For a Hadoop job, how will you write a custom partitioner?*

You write a custom partitioner for a Hadoop job, you follow the following path :

(1) Create a new class that extends Partitioner Class

(2) Override method getPartition

(3) In the wrapper that runs the MapReduce

(4) Add the custom partitioner to the job by using method set Partitioner Class or – add the custom partitioner to the job as a config file

34. *Why do we need Apache Hadoop?*

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

Many organizations use Hadoop for handling Big Data mainly because of it's ability to store, manage and analyze vast amounts of structured, semi-structured and unstructured data quickly, reliably, flexibly and at a very low-cost.

There are many reasons to believe that Hadoop is the best software for Big Data analytics. Some of the important benefits are :

Ability to store and process huge amounts of any kind of data, quickly. With data volumes and varieties constantly increasing, especially from social media (Facebook, Twitter, Walmart, etc), online video streaming platform (YouTube, Daily Motion, etc), gadgets (smartphones, smart watches, etc) and many more varieties, is a key consideration in Hadoop to process large volumes of data in a short period of time.

High computing Power : Hadoop's distributed computing model processes big data fast. The more

computing nodes we use in the Hadoop cluster, the more processing power we will have.

Fault Tolerance : Data and application processing is protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes in the cluster to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically (Data replication across multiple data blocks).

Flexibility : Unlike traditional relational databases, you don't have to pre-process data before storing it. We can store as much data as we want and decide how to use it at a later stage. That includes unstructured data like text, images and videos.

Low Cost : Since Hadoop is an open-source framework it is free and uses commodity hardware to store large quantities of data, which is very cost effective

Scalability : We can easily grow our system to handle more data simply by adding nodes. Very minimal administration is required.

Hadoop consists of three key components :

HDFS : HDFS is the storage layer of Hadoop.

Map Reduce : The data processing layer of Hadoop.

YARN : The resource management layer of Hadoop.

All the above three key components are highly responsible for Hadoop's key benefits to function. Many large web companies such as Google, Yahoo, Amazon and Facebook, etc., have used Hadoop over huge data sets creating innovative data products such as online advertising systems and recommendation engines.

Hadoop provides a platform with which enterprise IT can apply Big Data analysis and overcome various business problems such as product recommendation, fraud detection, sentiment analysis and many more such use cases, thus making Hadoop the most important software to be used for Big Data analytics. □□

(HADOOP ECO SYSTEM AND YARN, NoSQL, SPARK AND SCALA)

1. *Describe about graph database and schema less databases.* *(2016-17)*

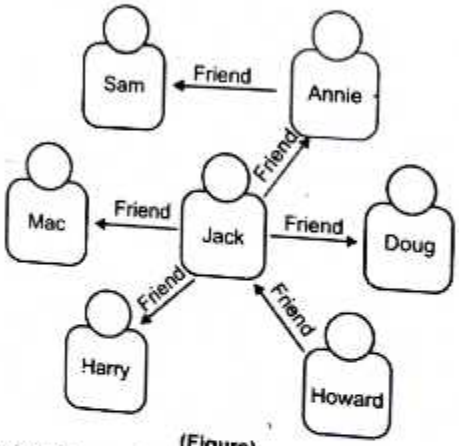
Graph Databases, as the name suggests, organize data in the form of a graph, based on the mathematical principle of graph theory. Fundamentally, we can consider a graph as a collection of nodes and edges. Nodes typically represent entities, edges are used to represent the relationships between those entities. What makes this useful to us in terms of databases is that nodes can hold data, describing the entity, but edges can also hold data, describing the nature and detail of that relationship. That data may be as simple as the type of relationship, or much more detailed.

Let's take a look at an example graph, and see what that looks like.

Need of Graph Databases : We live in a connected world! There are no isolated pieces of information, but rich, connected domains all around us. Only a database that natively embraces relationships is able to store, process, and query connections efficiently. While other databases compute relationships at query time through expensive JOIN operations, a graph database stores connections alongside the data in the model. Accessing nodes and relationships in a native graph database is an efficient, constant-time operation and allows you to quickly traverse millions of connections per second per core. Independent of the total size of your dataset, graph databases excel at managing highly-connected data and complex queries. With only a pattern and a set of starting points, graph databases explore the neighboring data around those initial starting points collecting and aggregating information from millions of nodes and relationships

and leaving any data outside the search perimete untouched.

The following graph shows an example of a social network graph. Given the people (nodes) and the relationships (edges), you can find out who the "friends of friends" of a particular person are for example, the friend of Howard's friends.



(Figure)

Advantages : Graph databases are capable of sophisticated fraud prevention. With graph databases, you can use relationships to process financial and purchase transactions in near-real time. With fast graph queries, you are able to detect that, for example, a potential purchaser is using the same email address and credit card as included in a known fraud case. Graph databases can also help you easily detect relationship patterns such as multiple people associated with a personal email address, or multiple people sharing the same IP address but residing in different physical addresses.

Examples : Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds latency. Neptune is highly available with read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across Availability Zones. Neptune is secure with support for encryption at rest.

Neptune is fully-managed, so you no longer need to worry about database management tasks such as hardware provisioning, software patching, setup, configuration, or backups.Neo4j is most widely used graph data base now a days.

Schema Less Database : Traditional relational databases are well-defined, using a schema to describe every functional element, including tables, rows views, indexes, and relationships. By exerting a high degree of control, the database administrator can improve performance and prevent capture of low-quality, incomplete, or malformed data. In a SQL database, the schema is enforced by the Relational Database Management System (RDBMS) whenever data is written to disk. But in order to work, data needs to be heavily formatted and shaped to fit into the table structure. This means sacrificing any undefined details during the save, or storing valuable information outside the database entirely.

A schema less database, like MongoDB, does not have these up-front constraints, mapping to a more 'natural' database. Even when sitting on top of a data lake, each document is created with a partial schema to aid retrieval. Any formal schema is applied in the code of your applications; this layer of abstraction protects the raw data in the NoSQL database and allows for rapid transformation as your needs change.

Any data, formatted or not, can be stored in a non-tabular NoSQL type of database. At the same time, using the right tools in the form of a schema less database can unlock the value of all of your structured and unstructured data types.

Advantages of Schema Less Database :

(1) **Greater Flexibility Over Data Types :** By operating without a schema, schema less databases can store, retrieve, and query any data type - perfect for big data analytics and similar operations that are powered by unstructured data. Relational databases apply rigid schema rules to data, limiting what can be stored.

(2) **No Pre-Defined Database Schemas :** The lack of schema means that your NoSQL database can accept any data type including those that you do not yet

use. These future-proofs your databases, allowing ... to grow and change as your data-driven operation... changes and mature.

(3) **No Data Truncation** : A schema less databa... makes almost no changes to your data: each item ... saved in its own document with a partial schema ... leaving the raw information untouched. This mean... that every detail is always available and nothing ... stripped to match the current schema. This ... particularly valuable if your analytics needs t... change at some point in the future.

(4) **Suitable for Real-time Analytics Functions** : With the ability to process unstructured data, applications built on NoSQL databases are better able to process real-time data, such as readings and measurements from IoT sensors. Schema ... databases is also ideal for use with machin... and artificial intelligence operations, ... accelerate automated actions in your busin...

(5) **Enhanced Scalability and Flexibility** : With NoSQL, you can use whichever data model is best suited to the job. Graph databases allow you to view relationships between data points, or you can use traditional wide table views with an exceptionally large number of columns. You can query, report, and model information however you choose. And as your requirements grow, you can keep adding nodes to increase capacity and power.

When a record is saved to a relational database, anything (particularly metadata) that does not match the schema is truncated or removed. Deleted at write, these details cannot be recovered at a later point in time.

Example : NoSQL Database is a non-relational Data Management System that does not require a fixed schema.For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day. NoSQL database stands for "Not Only SQL" or "Not SQL." Mongo DB is very popular schema less database in now a days.

2. *Discuss Hadoop YARN in detail with Failures in classic Map Reduce.* **(2018-19)**

In MapReduce framework. MapReduce job (MapReduce application) is divided between number of tasks called mappers and reducers. Each task runs on one of the machine (DataNode) of the cluster, and each machine has a limited number of predefined slots (map slot, reduce slot) for running tasks concurrently.

(1) Here, JobTracker is responsible for both managing the cluster's resources and driving the execution of the MapReduce job. It reserves and schedules slots for all tasks, configures, runs and monitors each task, and if a task fails, it allocates a new slot and reattempts the task. After a task finishes, the job tracker cleans up temporary resources and releases the task's slot to make it available for other jobs.

Problems with this approach in Hadoop 1.0:

(a) **It limits Scalability** : JobTracker runs on single machine doing several task like
  (i) Resource management
  (ii) Job and task scheduling and
  (iii) Monitoring

(b) Although there are so many machines (DataNode) available; they are not getting used. This limits scalability.

(c) **Availability Issue** : In Hadoop 1.0, JobTracker is single Point of availability. This means if JobTracker fails, all jobs must restart.

(d) **Problem with Resource Utilization:** In Hadoop 1.0, there is concept of predefined number of map slots and reduce slots for each TaskTrackers. Resource Utilization issues occur because maps slots might be 'full' while reduce slots is empty (and vice-versa). Here the compute resources (DataNode) could sit idle which are reserved for Reduce slots even when there is immediate need for those resources to be used as Mapper slots.

Limitation in Running non-MapReduce Application

In Hadoop 1.0, Job tracker was tightly integrated with MapReduce and only supporting application that obeys MapReduce programming framework can run on Hadoop.

Let's try to understand point 4 in more detail. Hadoop distributed file system (HDFS) makes it cheap to store large amounts of data, and its scalable MapReduce analysis engine makes it possible to extract insights from that data. MapReduce works on batch-driven data analysis, where the input data is partitioned into small batches that can be processed in parallel across many machines in the Hadoop cluster. But MapReduce, while powerful enough to express many data analysis algorithms, is not always the optimal choice of programming paradigm. It's often desirable to run other computation paradigms in the Hadoop cluster – here are some examples.

(1) **Problem in Performing Real-time Analysis** : MapReduce is batch driven. What if I want to perform real time analysis instead of batch processing (where results is available after several hours).There are many applications which need results in real time like fraud detection algorithms. There are real time engines like Apache Storm which can work better in this case. But in Hadoop 1.0, due to tight coupling these engines cannot run independently.

(2) **Problem in Running Message-Passing Approach** : It is a stateful process that runs on each node of a distributed network. The processes communicate with each other by sending messages and alter their state based on the messages they receive. This is not possible in MapReduce.

(3) **Problem in Running Ad-hoc Query** : Many users like to query their big data using SQL. Apache Hive can execute a SQL query as a series of MapReduce jobs, but it has shortcomings in terms of performance. Recently, some new approaches such as Apache Tajo, Facebook's Presto and Cloudera's Impala drastically improve the performance, but they require to run services in other form than MapReduce form. It is not possible to run all such non Map Reduce jobs on Hadoop Cluster. Such jobs have to "disguise" themselves as mappers and reducers in order to be able to run on Hadoop 1.0.

(4) YARN took over the task of cluster management from MapReduce and MapReduce is streamlined to perform Data Processing only in which it is best. YARN has central resource manager component which manages resources and allocates the resources to the application. Multiple applications can run on Hadoop via YARN and all application could share common resource management.

Advantage of YARN :

(1) **YARN Does Efficient Utilization of Resource** : There are no more fixed map-reduce slots. YARN provides central resource manager. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource.

(2) **Yarn can Even Run Application that do Not follow MapReduce Model** : YARN decouples MapReduce's resource management and scheduling capabilities from the data processing component, enabling Hadoop to support more varied processing approaches and a broader array of applications. For example, Hadoop clusters can now run interactive querying and streaming data applications simultaneously with MapReduce batch jobs. This also streamlines MapReduce to do what it does best – process data.

(3) **YARN is Backward Compatible** : This means that existing MapReduce job can run on Hadoop 2.0 without any change.

(4) **No more Job Tracker and Task Tracker Needed in Hadoop 2.0** : JobTracker and Task Tracker has totally disappeared. YARN splits the two major functionalities of the Job Tracker i.e. resource management and job scheduling/monitoring into 2 separate daemons (components).
   (a) Resource Manager
   (b) Node Manager (node specific)

(5) Central Resource Manager and node specific Node Manager together constitutes YARN.

3. *Explain Avro File base data sturctures in detail.* (2018-19)

Avro

Apache Avro is a language-neutral data serialization system. It was developed by Doug Cutting, the father of Hadoop. Since Hadoop writable classes lack language portability, Avro becomes quite helpful, as it deals with data formats that can be processed by multiple languages. Avro is a preferred tool to serialize data in Hadoop.

Avro has a schema-based system. A language-independent schema is associated with its read and writes operations. Avro serializes the data which has a built-in schema. Avro serializes the data into a compact binary format, which can be deserialized by any application.

Avro uses JSON format to declare the data structures. Presently, it supports languages such as Java, C, C++, C#, Python, and Ruby.

Avro Schemas : Avro depends heavily on its schema. It allows every data to be written with no prior knowledge of the schema. It serializes fast and the resulting serialized data is lesser in size. Schema is stored along with the Avro data in a file for any further processing.

In RPC, the client and the server exchange schemas during the connection. This exchange helps in the communication between same named fields, missing fields, extra fields, etc.

Avro schemas are defined with JSON that simplifies its implementation in languages with JSON libraries.

Like Avro, there are other serialization mechanisms in Hadoop such as Sequence Files,

Protocol Buffers, and Thrift :

Features of Avro : Listed below are some of the prominent features of Avro :

(1) Avro is a language-neutral data serialization system.

(2) It can be processed by many languages (currently C, C++, C#, Java, Python, and Ruby).

(3) Avro creates binary structured format that is both compressible and splittable. Hence it can be efficiently used as the input to Hadoop MapReduce jobs.

(4) Avro provides rich data structures. For example, you can create a record that contains an array, an enumerated type, and a sub record. These datatypes can be created in any language, can be processed in Hadoop, and the results can be fed to a third language.

(5) Avro schemas defined in JSON, facilitate implementation in the languages that already have JSON libraries.

(6) Avro creates a self-describing file named *Avro Data File*, in which it stores data along with its schema in the metadata section.

(7) Avro is also used in Remote Procedure Calls (RPCs). During RPC, client and server exchange schemas in the connection handshake.

4. *Elaborate on graph mapping schemas. What do you mean by lower bound replication rate?*

Schema mapping defines how data is converted between the schemas of an external data source and the Integrate session schema (stored by the cache). The mapping translates relational database tables and columns into classes and attributes in the session schema.

You select the tables and columns to import and optionally override the names of the corresponding classes and attributes in Integrate. You can also add new classes and attributes for internal use.

The schema used by the session is created from all data stores from which data has been opened.

Each data store has two schema mappings: one for input and one for output to a different location.

(1) ***Input mapping*** is used to read the data and in reverse when committing the data back to the source.

(2) ***Output mapping*** is used only for a copy-to task when the data is not being written back to the original source location.

We shall use the convention that q is the maximum number of inputs that can be sent to any one reducer. A mapping schema for a given problem, with a given value of q, is an assignment of a set of inputs to each reducer, subject to the constraints that: 1. No reducer is assigned more than q inputs. For every output, there is (at least) one reducer that is assigned all of the inputs for that output. We say such a reducer covers the output. Thus reducer

need not be unique, and it is, of course, permitted that these same inputs are assigned also to other reducers. The figure of merit for a mapping schema with a given reducer size q is the replication rate, which we defined to be the average number of reducers to which an input is mapped by that schema. Suppose that for a certain algorithm, the ith reducer is assigned $qi \leq q$ inputs, and let I be the number of different inputs. Then the replication rate r for this algorithm is

$$r = \sum_{i=1}^{p} qi/I$$

The replication rate of any map-reduce algorithm is the average number of key-value pairs that the mappers create from each input.

While upper bounds on r for all problems are derived using constructive algorithms, there is a generic technique for deriving lower bounds. Before proceeding to concrete lower bounds, we outline in this section the recipe that we use to derive all the lower bounds used in this paper.

(1) **Deriving g(q) :** First, find an upper bound, *g(q)*, on the number of outputs a reducer can cover if q is the number of inputs it is given.

(2) **Number of Inputs and Outputs :** Count the total numbers of inputs $|I|$ and outputs $|O|$.

(3) **The Inequality :** Assume there are p reducers, each receiving $qi \leq q$ inputs and covering *g(qi)* outputs. Together they cover all the outputs. That is: $Xp\ i = 1 g(qi) \geq |O|(1)$

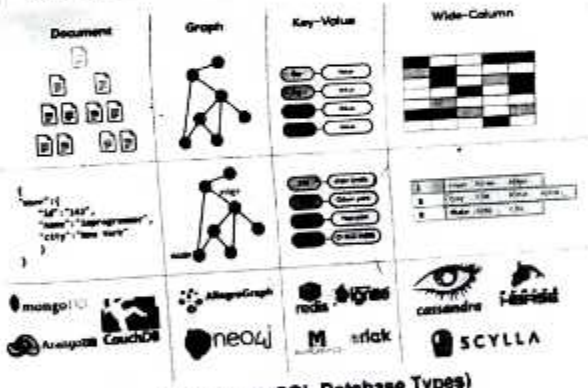
(4) **Replication Rate :** Manipulate the inequality from Equation 1 to get a lower bound on the replication rate, which is $P_p i = 1qi/|I|$.

5. *Write short note on NOSQL databases. List the differences between No SQL and relational databases.* (2016-17)

NoSQL databases (additionally called Not Only SQL Databases) are non-relational database systems used for storing and retrieving data. In today's world, we should not store all the data in table format only which has not

predefined fixed schemas (fix no of columns). Like User-generated data, GEO location data, IoT generated data; social graphs are examples of real-world data which has been increasing exponentially. These huge amounts of data required lots of processing also. Here, the NoSQL database comes into the picture. Using NoSQL database we can store and retire document, key-value, graph-based data easily & faster. We can easily avoid complex SQL joins operations. Easy to scale horizontally for real-world problems (web and enterprise business applications) using NoSQL DBs. Carlo Strozzi came with NoSQL term in the 1998 year. The motivation of using NoSQL – the simplicity of design, horizontal scaling to clusters of machines which is difficult to achieve in RDMS databases.

The Following List Describes Popular SQL and RDBMS Databases :



(Figure : NoSQL Database Types)

RDBMS Databases :

(1) **Oracle :** An object-relational database management system (DBMS) written in the C++ language.

(2) **IBM DB2 :** A family of database server products from IBM.

(3) **SAP ASE :** A business relational database server product for primarily Unix operating systems.

(4) **Microsoft SQL Server :** An RDBMS for enterprise-level databases that supports both SQL and NoSQL architectures.

(5) **Maria DB** : An enhanced, drop-in version of MySQL.

(6) **PostgreSQL** : An enterprise-level, object-relational DBMS that uses procedural languages, such as Perl and Python, in addition to SQL-level code.

Popular NoSQL Databases : The following list describes popular NoSQL databases :

(1) **MongoDB** : The most popular open-source NoSQL system. MongoDB is a document-oriented database that stores JSON-like documents in dynamic schemas.

(2) **Apache CouchDB** : An open-source, web-oriented database developed by Apache. CouchDB uses the JSON data exchange format to store its documents; JavaScript for indexing, combining, and transforming documents; and HTTP for its API.

(3) **Apache HBase** : An open-source Apache project developed as a part of Hadoop. HBase is a column store database written in Java with capabilities similar to those that Google BigTable provides.

(4) **Oracle NoSQL Database** : A proprietary database that supports JSON table and key-value datatypes running on-premise or as a cloud service.

(5) **Apache Cassandra DB** : A distributed database that excels at handling extremely large amounts of structured data. Cassandra DB is also highly scalable. Facebook created Cassandra DB.

(6) **Riak** : An open-source, key-value store database written in Erlang. Riak has built-in fault-tolerance replication and automatic data distribution that enable it to offer excellent performance.

(7) **Objectivity InfiniteGraph** : A highly specialized graph database that focuses on graph data structures. InfiniteGraph, implemented in Java, is useful for finding hidden relationships in big data.

Differeneces between NoSQL and Relational database

NoSQL Database	Relational Database
NoSQL Database supports a very simple query language	Relational Database supports a powerful query language.
NoSQL Database has no fixed schema.	Relational Database has a fixed schema.

NoSQL Database is only eventually consistent.	Relational Database follows acid properties. (Atomicity, Consistency, Isolation, and Durability)
NoSQL databases don't support transactions (support only simple transactions).	Relational Database supports transactions (also complex transactions with joins)
NoSQL Database is used to handle data coming in high velocity.	Relational Database is used to handle data coming in low velocity.
The NoSQL?s data arrive from many locations.	Data in relational database arrive from one or few locations.
NoSQL database can manage structured, unstructured and semi-structured data.	Relational database manages only structured data.
NoSQL databases have no single point of failure.	Relational databases have a single point of failure with failover
NoSQL databases can handle big data or data in a very high volume.	NoSQL databases are used to handle moderate volume of data.
NoSQL has decentralized structure.	Relational database has centralized structure.
NoSQL database gives both read and write scalability.	Relational database gives read scalability only.
NoSQL database is deployed in horizontal fashion.	

6. ◆ ***List down the entities of YARN in YARN architecture.*** **(2016-17)**

◆ ***Explain the YARN Architecture and workflow in Hadoop YARN with suitable diagarm***

YARN stands for "***Yet Another Resource Negotiator***". It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN was described as a "*Redesigned Resource Manager*" at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing. YARN architecture basically separates resource management layer from the processing layer. In Hadoop 1.0 version, the responsibility of Job tracker is split between the resource manager and application manager. YARN also allows different data processing engines like graph

processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS (Hadoop Distributed File System) thus making the system much more efficient. Through its various components, it can dynamically allocate various resources and schedule the application processing. For large volume data processing, it is quite necessary to manage the available resources properly so that every application can leverage them.

The Main Entities of YARN Architecture Include :

(1) **Client** : It submits map-reduce jobs.

(2) **Resource Manager** : It is the master daemon of YARN and is responsible for resource assignment and management among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components.

(3) **Scheduler** : It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.

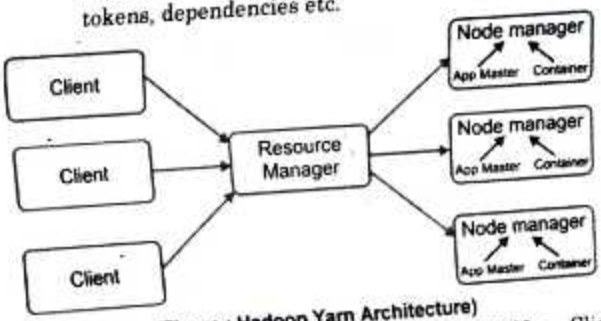
(4) **Application manager** : It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Manager container if a task fails.

(5) **Node Manager** : It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Node Manager. It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.

(6) **Application Master** : An application is a single job submitted to a framework. The application manager

is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application. The application master requests the container from the node manager by sending a Container Launch Context (CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.

(7) **Container** : It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context (CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.



(Figure : Hadoop Yarn Architecture)

Application workflow in Hadoop YARN : Client submits an application

(1) The Resource Manager allocates a container to start the Application Manager

(2) The Application Manager registers itself with the Resource Manager

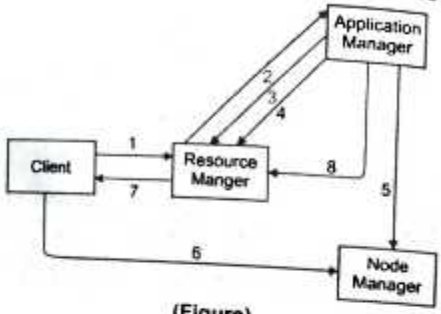
(3) The Application Manager negotiates containers from the Resource Manager

(4) The Application Manager notifies the Node Manager to launch containers

(5) Application code is executed in the container

(6) Client contacts Resource Manager/Application Manager to monitor application's status

(7) Once the processing is complete, the Application Manager un-registers with the Resource Manager



(Figure)

7. *What is the advantage of using Scala over other functional programming languages? Also Explain the advantages of Companion Objects in Scala.*

Scala supports both functional programming and object-oriented programming making it a very scalable language.Scala is a pure object-oriented language, but it's also fully functional, giving developers the best of both worlds.

(1) As the name itself indicates Scala meaning Scalable Language, its high scalable, maintainability, productivity and testability features make it advantageous to use Scala.

(2) Singleton and Companion Objects in Scala provide a cleaner solution unlike static in other JVM languages like Java.

(3) It eliminates the need for having a ternary operator as if blocks', 'for-yield loops', and 'code' in braces return a value in Scala.

Advantages of Companion Objects in Scala :

(1) Companion objects are beneficial for encapsulating things and they act as a bridge for writing functional and object oriented programming code.

(2) Using companion objects, the Scala programming code can be kept more concise as the static keyword need not be added to each and every attribute.

(3) Companion objects provide a clear separation between static and non-static methods in a class because everything that is located inside a companion object is not a part of the class's runtime objects but is available from a static context and vice versa.

8. *What is a Scala Map? Explain in brief.*

Scala Map

Scala Map is a collection of key value pairs wherein the value in a map can be retrieved using the key. Values in a Scala Map are not unique but the keys are unique. Scala supports two kinds of maps- mutable and immutable. By default, Scala supports immutable map and to make use of the mutable map, programmers have to import the scala.collection.mutable.Map class explicitly. When programmers want to use mutable and immutable map together in the same program then the mutable map can be accessed as mutable.map and the immutable map can just be accessed with the name of the map.

Basic Operations on MAP : All operations on maps can be expressed in terms of the following three methods.

Sr.No	Methods & Description
1.	**Keys** : This method returns an iterable containing each key in the map.
2.	**Values** : This method returns an iterable containing each value in the map.
3.	**isEmpty** : This method returns true if the map is empty otherwise false.

9. *Explain the features of Scala in detail.*

Scala, is a hybrid functional programming language. It was created by Martin Odersky. Scala smoothly integrates the features of object-oriented and functional languages. Scala is compiled to run on the Java Virtual Machine. Many existing companies, who depend on Java for business critical applications, are turning to Scala to boost their development productivity, applications scalability and overall reliability.

Here we have presented a few points that ... Scala the first choice of application developers.

(1) **Scala is Object-oriented** : Scala is a pure ... oriented language in the sense that every value ... object. Types and behavior of objects are describe... classes and traits which will be explained ... subsequent chapters.

Classes are extended by subclassing ... flexible mixin-based composition mechanism ... clean replacement for multiple inheritance.

(2) **Scala is Functional** : Scala is also a functio... language in the sense that every function is a ... and every value is an object so ultimately ev... function is an object.

(3) Scala provides a lightweight syntax for defin... anonymous functions, it supports higher-or... functions, it allows functions to be nested, ... supports currying. These concepts will be explained in subsequent chapters.

(4) **Scala is Statically Typed** : Scala, unlike some ... the other statically typed languages (C, Pascal, Rust, etc.), does not expect you to provide redundant type information. You don't have to specify a type in most cases, and you certainly don't have to repeat it.

(5) **Scala Runs on the JVM** : Scala are compiled into Java Byte Code which is executed by the Java Virtual Machine (JVM). This means that Scala and Java have a common runtime platform. You can easily move from Java to Scala.

(6) The Scala compiler compiles your Scala code into Java Byte Code, which can then be executed by the '**scala**' command. The '**scala**' command is similar to the **java** command, in that it executes your compiled Scala code.

(7) **Scala can Execute Java Code** : Scala enables you to use all the classes of the Java SDK and also your own custom Java classes, or your favorite Java open source projects.

(8) **Scala can do Concurrent & Synchronize Processing** : Scala allows you to express general programming patterns in an effective way. It reduces the number of lines and helps the programmer to code in a type-safe way. It allows you to write codes in an immutable manner, which makes it easy to apply concurrency and parallelism (Synchronize).

10. *Explain various components of Spark with suitable diagram.*

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

Components of Spark : The following illustration depicts the different components of Spark.

(1) **Apache Spark Core :** Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

(2) **Spark SQL :** Spark SQL is a component on top of Spark Core that introduces a new data abstraction

called Schema RDD, which provides support for structured and semi-structured data.

(3) **Spark Streaming** : Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

(4) **MLlib (Machine Learning Library)** : MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

(5) **GraphX** : GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

11. *Write short note on Resilient Distributed Datasets.*

Resilient Distributed Datasets

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs – **parallelizing** an existing collection in your driver program, or **referencing**

a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations. Let us first discuss how MapReduce operations take place and why they are not so efficient.

12. *How Spark can be built with Hadoop components. Explain in brief.*

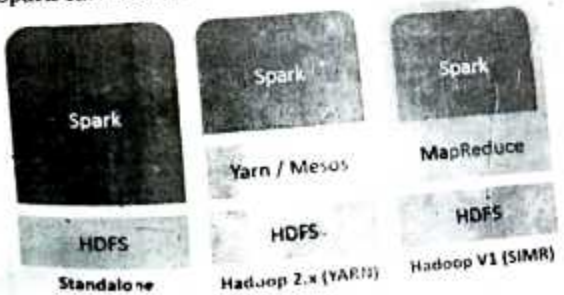
There are three ways of Spark deployment as explained below.

(1) **Standalone** : Spark Standalone deployment means Spark occupies the place on top of HDFS (Hadoop Distributed File System) and space is allocated for HDFS, Spark and MapReduce will run side by side to cover all spark jobs on cluster.

(2) **Hadoop Yarn** : Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.

(3) **Spark in MapReduce (SIMR)** : Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

The following diagram shows three ways of how Spark can be built with Hadoop components.



13. *Explain feature of inheritance in Scala in brief.*

Inheritance is an important pillar of (O (Object Oriented Programming). It is the mechanism Scala by which one class is allowed to inherit the featur (fields and methods) of another class. Inheritan is utilized the concept of reusability of code. You ca achieve inheritance by using extends keyword. T achieve inheritance a class must extend to oth class. A class which is extended called super or paren class. a class which extends class is called derived or ba class.

Syntax :

```
(1)  Class SubClassName extends SuperClassName(){
(2)  /* Write your code
(3)  * methods and fields etc
(4)  */
(5)  }
```

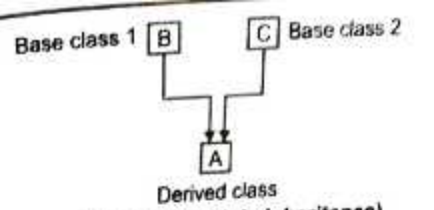
Important Terminology :

(1) **Super Class** : The class whose features ar inherited is known as super class (or a base class o a parent class).

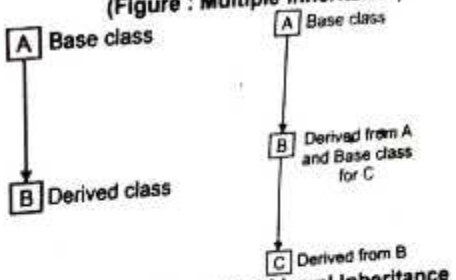
(2) **Sub Class** : The class that inherits the other class i known as subclass (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

(3) **Reusability** : Inheritance supports the concept o "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this we are reusing the fields and methods of the existing class.

Types of Inheritance in Scala : Scala supports various types of inheritance including single, multilevel, multiple, and hybrid. You can use single, multilevel, and hierarchal in your class. Multiple and hybrid can only be achieved by using traits. Here, we are representing all types of inheritance by using pictorial form.

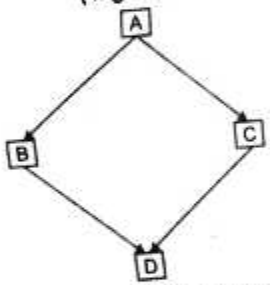


(Figure : Multiple Inheritance)

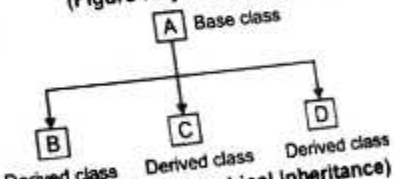


Single Inheritence | Multi Level Inheritance

(Figure)



(Figure : Hybrid Inheritance)



(Figure : Hierarchical Inheritance)

14. *Write down the advantages of Hadoop ecosystem over traditional system.*

Problem with Traditional Systems

By traditional systems, I mean systems like Relational Databases and Data Warehouses. Organizations

have been using them for the last 40 years to store and analyze their data. But the data being generated today can't be handled by these databases for the following reasons :

(1) Most of the data generated today are semi-structured or unstructured. But traditional systems have been designed to handle only structured data that has well-designed rows and columns

(2) Relations Databases are vertically scalable which means you need to add more processing, memory, storage to the same system. This can turn out to be very expensive

(3) Data stored today are in different silos. Bringing them together and analyzing them for patterns can be a very difficult task.

Most of the services available in the ecosystem are to supplement the main four core components of Hadoop which include HDFS, YARN, MapReduce and Hadoop Common. Hadoop eco system overcome the problems associated with traditional system in following ways :

(Figure : Apache Hadoop Ecosystem)

(1) Hadoop is highly scalable because it handles data in distributed manner

(2) Compared to vertical scaling in RDBMS, Hadoop offers horizontal scaling

(3) It creates and saves replicas of data making it fault tolerant

(4) It is economical as all the nodes in the cluster are commodity hardware which is nothing but inexpensive machines

(5) Hadoop utilizes the data locality concept to process the data on the nodes on which they are stored rather than moving the data over the network thereby reducing traffic

(6) It can handle any type of data: structured, semi-structured, and unstructured. This is extremely important in today's time because most of our data (emails, Instagram, Twitter, IoT devices, etc.) has no defined format

15. *Explain mahout in brief.*

It performs collaborative filtering, clustering and classification. Some people also consider frequent item set missing as Mahout's function. Let us understand them individually:

(1) **Collaborative Filtering:** Mahout mines user behaviors, their patterns and their characteristics and based on that it predicts and make recommendations to the users. The typical use case is E-commerce website.

(2) **Clustering :** It organizes a similar group of data together like articles can contain blogs, news, research papers etc.

(3) **Classification :** It means classifying and categorizing data into various sub departments like articles can be categorized into blogs, news, essay, research papers and other categories.

(4) **Frequent Item Set Missing :** Here Mahout checks, which objects are likely to be appearing together and make suggestions, if they are missing. For example, cell phone and cover are brought together in general. So, if you search for a cell phone, it will also recommend you the cover and cases. Mahout provides a command line to invoke various algorithms.

It has a predefined set of library which already contains different inbuilt algorithms for different use cases.

16. *What is OOzie in Apache Hadoop Ecosystem? Explain in brief.*

Apache Oozie

Apache Oozie is a workflow scheduler for Hadoop. It is a system which runs the workflow of dependent jobs. Here, users are permitted to create Directed Acyclic

Graphs of workflows, which can be run in parallel and sequentially in Hadoop.

It Consists of Two Parts :

(1) **Workflow Engine :** Responsibility of a workflow engine is to store and run workflows composed of Hadoop jobs e.g., MapReduce, Pig, Hive.

(2) **Coordinator Engine :** It runs workflow jobs based on predefined schedules and availability of data.

Oozie is scalable and can manage the timely execution of thousands of workflows (each consisting of dozens of jobs) in a Hadoop cluster. Oozie is very much flexible, as well. One can easily start, stop, suspend and rerun jobs. Oozie makes it very easy to rerun failed workflows. One can easily understand how difficult it can be to catch up missed or failed jobs due to downtime or failure. It is even possible to skip a specific failed node.

How Does OOZIE Work : Oozie runs as a service in the cluster and clients submit workflow definitions for immediate or later processing. Oozie workflow consists of **action nodes** and **control-flow nodes.**

An **action node** represents a workflow task, e.g., moving files into HDFS, running a MapReduce, Pig or Hive jobs, importing data using Sqoop or running a shell script of a program written in Java.

A control-flow node controls the workflow execution between actions by allowing constructs like conditional logic wherein different branches may be followed depending on the result of earlier action node.

Start Node, End Node, and Error Node fall under this category of nodes.

Start Node, designates the start of the workflow job.

End Node, signals end of the job.

Error Node designates the occurrence of an error and corresponding error message to be printed.

At the end of execution of a workflow, HTTP callback is used by Oozie to update the client with the workflow status. Entry-to or exit from an action node may also trigger the callback.

17. *OOzie is a important component of Hadoop Eco System. Justify the statement.*

The main purpose of using Oozie is to manage different type of jobs being processed in Hadoop system.

Dependencies between jobs are specified by a user in the form of Directed Acyclic Graphs. Oozie consumes this information and takes care of their execution in the correct order as specified in a workflow. That way user's time to manage complete workflow is saved. In addition, Oozie has a provision to specify the frequency of execution of a particular job. There are many features of OOzie which makes it a important part of Hadoop Eco System. Some of them are listed below.

Features of Oozie :

(1) Oozie has client API and command line interface which can be used to launch, control and monitor job from Java application.

(2) Using its Web Service APIs one can control jobs from anywhere.

(3) Oozie has provision to execute jobs which are scheduled to run periodically.

(4) Oozie has provision to send email notifications upon completion of jobs.

18. *Explain R Connectors in brief?*

Oracle R Connector for Hadoop is an R package that provides an interface between the local R environment and Hadoop. You install and load this package the same as you would for any other R package. Using simple R functions, you can copy data between R memory, the local file system, and HDFS. You can schedule R programs to execute as Hadoop MapReduce jobs and return the results to any of those locations.

Oracle R Connector for Hadoop APIs

Oracle R Connector for Hadoop provides API access from a local R client to Hadoop, using these APIs:

(1) **Hadoop** : Provides an interface to Hadoop MapReduce.

(2) **Hdfs** : Provides an interface to HDFS.

(3) **Orhc** : Provides an interface between the local 2 instance and Oracle Database.

All of these functions are included in the ORHC library. The functions are listed in this chapter in alphabetical order.

19. *Discuss the different ways of constructing version stamps. What are their pros and cons?* (2016-17)

Version Stamps allow tracking of a variable number of replicas and do not resort to counters. This mechanism can depict scalability problems in some settings, but can be replaced by Interval Tree Clocks. Interval Tree Clocks generalize version vectors and vector clocks and allows dynamic numbers of replicas/processes.

Version stamps help you detect concurrency conflicts. When you read data, then update it, you can check the version stamp to ensure nobody updated the data between your read and write.

(1) Version stamps can be implemented using counters, GUIDs, content hashes, timestamps, or a combination of these.

(2) With distributed systems, a vector of version stamps allows you to detect when different nodes have conflicting updates.

Different Ways of Constructing Version Stamps

There are various ways you can construct your version stamps.

Counter Method : You can use a counter, always incrementing it when you update the resource. Counters are useful since they make it easy to tell if one version is more recent than another. On the other hand, they require the server to generate the counter value, and also need a single master to ensure the counters aren't duplicated.

GUID Method : Another approach is to create a GUID, a large random number that's guaranteed to be unique. These use some combination of dates, hardware information, and whatever other sources of randomness they can pick up. The nice thing about GUIDs is that they can be generated by anyone and you'll never get a

duplicate; a disadvantage is that they are large and can't be compared directly for recentness.

Hashing Method : A third approach is to make a hash of the contents of the resource. With a big enough hash key size, a content hash can be globally unique like a GUID and can also be generated by anyone; the advantage is that they are deterministic any node will generate the same content hash for same resource data. However, like GUIDs they can't be directly compared for recentness, and they can be lengthy.

Time Stamp Method : A fourth approach is to use the timestamp of the last update. Like counters, they are reasonably short and can be directly compared for recentness, yet have the advantage of not needing a single master. Multiple machines can generate timestamps but to work properly, their clocks have to be kept in sync. One node with a bad clock can cause all sorts of data corruptions. There's also a danger that if the timestamp is too granular you can get duplicates it's no good using timestamps of a millisecond precision if you get many updates per millisecond.

Vector Stamp : set of version stamps for all nodes in a distributed system . Allows detection of conflicting updates on different nodes

Pros of Version Stamp

(1) You can blend the advantages of these different version stamp schemes by using more than one of them to create a composite stamp. For example, CouchDB uses a combination of counter and content hash. Most of the time this allows version stamps to be compared for recentness, even when you use peer-to-peer replication.

(2) Two peers update at the same time, the combination of the same count and different content hashes makes it easy to spot the conflict.

(3) As well as helping to avoid update conflicts, version stamps are also useful for providing session consistency.

(4) Provide a means of detecting concurrency conflicts.

(5) Version stamps are also used to associate object versions with database

(6) The database version approach offers a very powerful tool for managing multi version databases, because of version stamp semantics.

(7) The main advantage of the database version approach is its orthogonality to the object model, object addressing, concurrency control, access authorization and other object management problems.

(8) Version stamps are easy to implement and economical with respect to space. In comparison with other approaches.

Cons of Version Stamp :

(1) Version management is a common problem associated with version stamp.

(2) Problem of stack overflow is sometimes reported while creating version stamp.

❑❑

(HADOOP ECO SYSTEM FRAMEWORK, PIG , HIVE AND HBASE)

1. *Explain Hadoop Eco system framework in brief?*

Hadoop Ecosystem

It is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are four major elements of Hadoop i.e. **HDFS, MapReduce, YARN, and Hadoop Common :** Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem :

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

(2) **YARN :** Yet Another Resource Negotiator

(3) **MapReduce :** Programming based Data Processing

(4) **Spark :** In-Memory data processing

(5) **PIG, HIVE :** Query based processing of data services

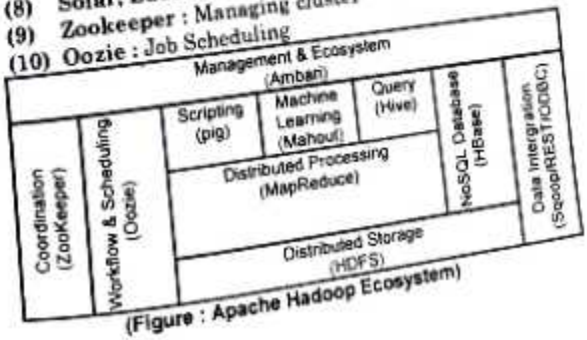
(6) **HBase :** NoSQL Database

(7) **Mahout, Spark MLLib :** Machine Learning algorithm libraries

(8) **Solar, Lucene :** Searching and Indexing

(9) **Zookeeper :** Managing cluster

(10) **Oozie :** Job Scheduling



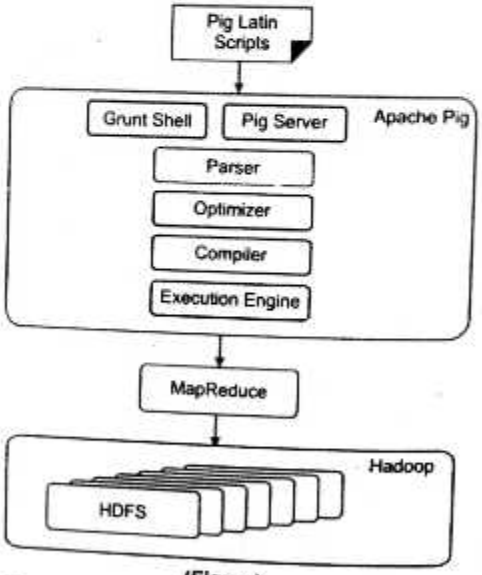
(Figure : Apache Hadoop Ecosystem)

2. *What are the components of Pig execution Environment?*

Apache Pig Components

Hadoop stores raw data coming from various sources like IOT, websites, mobile phones, etc. and preprocessing is done in Map-reduce. Pig framework converts any pig job into Map-reduce hence we can use the pig to do the ETL (Extract Transform and Load) process on the raw data. Apache pig can handle large data stored in Hadoop to perform data analysis and its support file formats like text, CSV, Excel, RC, etc.

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



(Figure)

(1) **Parser** : Any pig scripts or commands in the grunt shell are handled by the parser. Parse will perform checks on the scripts like the syntax of the scripts, do type checking and perform various other checks. These checks will give output in a Directed Acyclic Graph (DAG) form, which has a pig Latin statements and logical operators. The DAG will have nodes that are connected to different edges, here our logical operator of the scripts is nodes and data flows are edges.

(2) **Optimizer** : As soon as parsing is completed and DAG is generated, It is then passed to the logical optimizer to perform logical optimization like projection and pushdown. Projection and pushdown are done to improve query performance by omitting unnecessary columns or data and prune the loader to only load the necessary column.

(3) **Compiler** : The optimized logical plan generated above is compiled by the compiler and generates a series of Map-Reduce jobs. Basically compiler will convert pig job automatically into MapReduce jobs and exploit optimizations opportunities in scripts, due this programmer doesn't have to tune the program manually. As pig is a data-flow language its compiler can reorder the execution sequence to optimize performance if the execution plan remains the same as the original program.

(4) **Execution Engine** : Finally, all the MapReduce jobs generated via compiler are submitted to Hadoop in sorted order. In the end, MapReduce's job is executed on Hadoop to produce the desired output.

(5) **Execution Mode** : Pig works in two types of execution modes depend on where the script is running and data availability :

Local Mode : Local mode is best suited for small data sets. Pig is implemented here on single JVM as all files are installed and run on local host due to this parallel mapper execution is not possible. Also while loading data pig will always look into the local file system.

Command to Invoke Grunt Shell in Local Mode :

pig -x local

To run pig in tez local modes (Internally invoke tez runtime) use below :

pig -x tez_local

MapReduce Mode (MR Mode) : In MapReduce, the mode programmer needs access and setup of the Hadoop cluster and HDFS installation. In this mode data on which processing is done is exists in the HDFS system. After execution of pig script in MR mode, pig Latin statement is converted into Map Reduce jobs in the back-end to perform the operations on the data. By default pig uses Map Reduce mode, hence we don't need to specify it using the -x flag.

Command to Invoke Grunt Shell in MR Mode :

pig -x map reduce OR pig

Apart from execution mode there three different ways of execution mechanism in Apache pig:

(1) **Interactive Mode Or Grunt Shell :** Enter the pig statement and get the output in the command shell.

(2) **Batch Mode or Script Mode :** This is the non-interactive way of executing where programmer write a pig Latin script with '.pig' extension and execute the same.

(3) **Embedded Mode :** Use programming languages like java in our script with the help of UDFs.

3. *Explain storage mechanism in HBase. Write a query to create a table in HBase.*

Storage Mechanism in HBase

HBase is a column-oriented database and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase :

(1) Table is a collection of rows.
(2) Row is a collection of column families.
(3) Column family is a collection of columns.
(4) Column is a collection of key value pairs.

Given below is an example schema of table in Hbase

Row Id	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12
1												
2												
3												

Column Oriented and Row Oriented : Column-oriented databases are those that store data tables as sections of columns of data, rather than as rows of data. Shortly, they will have column families.

Row-Oriented Database	Column-Oriented Database
It is suitable for Qnline Transaction Process (OLTP)	It is suitable for Online Analytical Processing (OLAP)
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

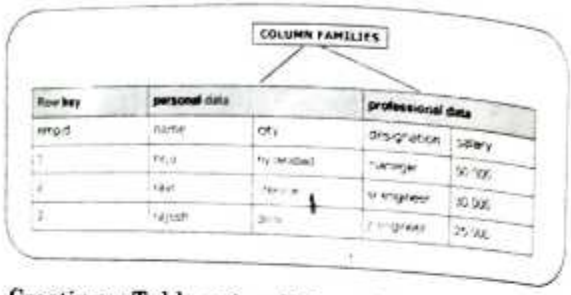
Row-Oriented Data Stores :

(1) Data is stored and retrieved one row at a time and hence could read unnecessary data if only some of the data in a row is required.
(2) Easy to read and write records
(3) Well suited for OLTP systems
(4) Not efficient in performing operations applicable to the entire dataset and hence aggregation is an expensive operation
(5) Typical compression mechanisms provide less effective results than those on column-oriented data stores

Column-Oriented Data Stores :

(1) Data is stored and retrieved in columns and hence can read only relevant data if only some data is required
(2) Read and Write are typically slower operations
(3) Well suited for OLAP systems
(4) Can efficiently perform operations applicable to the entire dataset and hence enables aggregation over many rows and columns
(5) Permits high compression rates due to few distinct values in columns

The following image shows column families in a column-oriented database :



Creating a Table using HBase Shell : You can create a table using the create command, here you must specify the table name and the Column Family name. The syntax to create a table in HBase shell is shown below.

create '<table name>','<column family>'

Example : Given below is a sample schema of a table named emp. It has two column families: "personal data" and "professional data".

Row Key	Personal Data	Professional Data

You can create this table in HBase shell as shown below.

hbase(main):002:0> create 'emp', 'personal data', 'professional data'

And it will give you the following output.

0 row(s) in 1.1300 seconds

=> Hbase::Table - emp

4. *What is Zookeeper? How does Zookeeper work? Why we need it?* **(2018-19)**

Zookeeper

Zookeeper is top-level software developed by Apache that acts as a centralized service and is used to maintain naming and configuration data and to provide flexible and robust synchronization within distributed systems. Zookeeper keeps track of status of the Kafka cluster nodes and it also keeps track of Kafka topics, partitions etc.

Zookeeper itself is allowing multiple clients to perform simultaneous reads and writes and acts as a shared configuration service within the system. The

Zookeeper atomic broadcast (ZAB) protocol is the brains of the whole system, making it possible for Zookeeper to act as an atomic broadcast system and issue orderly updates.

The data within Zookeeper is divided across multiple collections of nodes and this is how it achieves its high availability and consistency. In case a node fails, Zookeeper can perform instant failover migration; for example if a leader node fails, a new one is selected in real-time by polling within an ensemble. A client connecting to the server can query a different node if the first one fails to respond. Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc. Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers. In addition to availability, the nodes are also used to track server failures or network partitions. Clients communicate with region servers via zookeeper. In pseudo and standalone modes, HBase itself will take care of zookeeper.

Working of Zookeeper

The data within Zookeeper is divided across multiple collection of nodes and this is how it achieves its high availability and consistency. In case a node fails, Zookeeper can perform instant failover migration; e.g. if a leader node fails, a new one is selected in real-time by polling within an ensemble. A client connecting to the server can query a different node if the first one fails to respond.

Need of Zookeeper

Controller Election : The controller is one of the most important broking entity in a Kafka ecosystem, and it also has the responsibility to maintain the leader-follower relationship across all the partitions. If a node by some reason is shutting down, it's the controller's responsibility to tell all the replicas to act as partition leaders in order to fulfill the duties of the partition leaders on the node that is about to fail. So, whenever a node shuts down, a new controller can be elected and it can also be made sure that at any given time, there is only one controller and all the follower nodes have agreed on that.

Configuration of Topics : The configuration regarding

all the topics including the list of existing topics, the number of partitions for each topic, the location of all the replicas, list of configuration overrides for all topics and which node is the preferred leader, etc.

Access Control lists : Access control lists or ACLs for all the topics are also maintained within Zookeeper.

Membership of the Cluster : Zookeeper also maintains a list of all the brokers that are functioning at any given moment and are a part of the cluster.

5. *What is Pig Latin? Explain the necessity, features and applications of Pig Latin?* **(2018-19)**

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Apache Pig. To write data analysis programs, Pig provides a high-level language known as Pig Latin. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using Apache Pig, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

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.

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

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

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

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

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

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

(2) **Ease of Programming :** Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.

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

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

(5) **UDF's :** Pig provides the facility to create **User-defined Functions** in other programming languages such as Java and invoke or embed them in Pig Scripts.

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

Applications of Apache Pig : Apache Pig are generally used by data scientists for performing tasks involving ad-hoc processing and quick prototyping. Apache Pig is used:

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

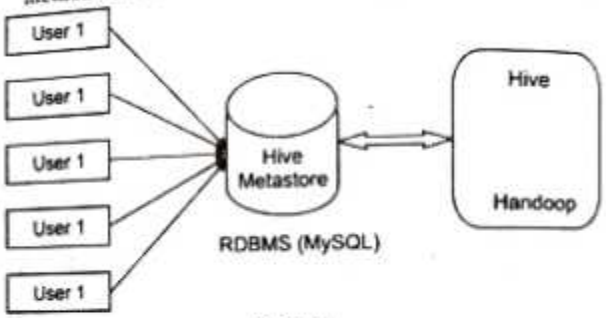
(2) To perform data processing for search platforms.

(3) To process time sensitive data loads.

6. *Explain megastores in HIVE?* **(2018-19)**

Hive metastore (HMS) is a service that stores metadata related to Apache Hive and other services, in a backend RDBMS, such as MySQL or PostgreSQL. When new data is saved to object storage, we register it into Hive Metastore by calling the metastore API from the code of any data application or orchestration tool. This declarative

phase maps a set of objects in the object store to a table exposed by Hive. Part of registration includes specifying the schema of the table held in the file, with some metadata describing the columns.



(Figure)

Using Hive Metastore in this way provides four main benefits related to:

(1) Virtualization
(2) Discoverability
(3) Schema Evolution
(4) Performance

(1) **Virtualization :** Data analysts using SQL usually aren't interested in the details of object storage and its access patterns. They'd simply like to have their tables, please. This dynamic is the driving force that makes Hive Metastore irreplaceable while other Hadoop components were replaced. Every new technology that was introduced made sure to support Hive Metastore to avoid breaking critical analytic workflows dependent upon the table objects defined in Hive.

(2) **Discoverability :** Hive Metastore naturally becomes a catalog of all the collections held in object storage when exposing new data is accompanied by updating it. If well maintained, this allows for the discovery of data sets available to query. Additionally, supplemental can be saved in the metastore to provide helpful information about the data like its update frequency, who owns it, etc.

(3) **Schema Evolution :** One of the challenges of managing data sets over time is their mutability. Records may change over time with respect to the existing columns describing their attributes. Or the set of attributes itself changes over time, resulting in a change to the schema of the table. The registration process described above provides a record of the schema for each additional data file that belongs to the table. This means that if the schema has changed at some point in time, it will be recorded within the Hive Metastore. When accessing the data, it can be accessed with the appropriate schema. This also provides a good basis to validate a schema if it should not have changed, and alert on it. Hive holds the information to create such a test.

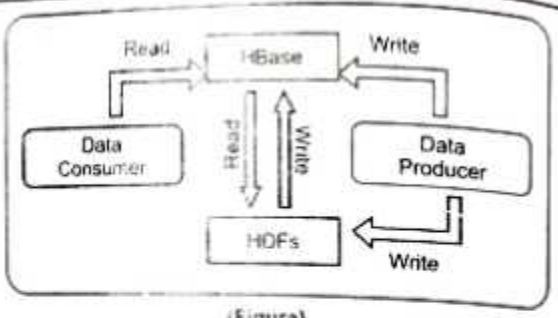
(4) **Performance :** Since Hive Metastore maps the table to the underlying object, it allows the representation of partitions according to the primary key supported by the object storage. The granularity of the partitions can be set by the user, and if partitions are balanced and their number is reasonable, this mapping allows improvement in query performance.

7. *Explain HBase and and their Data model and implementation.* **(2015-16)**

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

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS). It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

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



(Figure)

Features of HBase :

(1) HBase is linearly scalable.
(2) It has automatic failure support.
(3) It provides consistent read and writes.
(4) It integrates with Hadoop, both as a source and a destination.
(5) It has easy java API for client.
(6) It provides data replication across clusters.

Where to Use HBase :

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

Applications of HBase :

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

HBase Data Model : The Data Model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns. Additionally, the layout of the data model makes it easier to partition the data and distribute it across the cluster. The Data Model in

HBase is made of different logical components such as Tables, Rows, Column Families, Columns, Cells and Versions.

Tables : The HBase Tables are more like logical collection of rows stored in separate partitions called Regions. As shown above, every Region is then served by exactly one Region Server. The figure above shows a representation of a Table. Each table must have an element defined as Primary Key.

Rows : A row is one instance of data in a table and is identified by a row key. Row keys are unique in a Table and are always treated as a byte [].Row key acts as a Primary key in HBase.

Column Families : Data in a row are grouped together as Column Families. Each Column Family has one more Columns and these Columns in a family are stored together in a low level storage file known as HFile. Column Families form the basic unit of physical storage to which certain HBase features like compression are applied. Hence it's important that proper care be taken when designing Column Families in table.

The table above shows Customer and Sales Column Families. The Customer Column Family is made up 2 columns – Name and City, whereas the Sales Column Families is made up to 2 columns – Product and Amount.

Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburgh, PA	Desk	$500.00
104	Jack Black	St, Louis, MO	Bed	$1600.00

Column Families

(Figure)

Columns : A Column Family is made of one or more columns. A Column is identified by a Column Qualifier that consists of the Column Family name concatenated

with the Column name using a colon – example: columnfamily:columnname.

There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns. Each column present in HBase denotes attribute corresponding to object

Cell : A Cell stores data and is essentially a unique combination of rowkey, Column Family and the Column (Column Qualifier). The data stored in a Cell is called its value and the data type is always treated as byte[].

Version : The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data retained in a column family is configurable and this value by default is 3.

8. *Explain Cassandra Data model with an example.*

Cassandra Data Model

Cassandra is a NoSQL database, which is a key-value store. Some of the features of Cassandra data model are as follows :

(1) Data in Cassandra is stored as a set of rows that are organized into tables.
(2) Tables are also called column families.
(3) Each Row is identified by a primary key value.
(4) Data is partitioned by the primary key.
(5) You can get the entire data or some data based on the primary key.

Cassandra Data Model Components : Cassandra data model provides a mechanism for data storage. The components of Cassandra data model are keyspaces, tables, and columns.

Keyspaces : Cassandra data model consists of keyspaces at the highest level. Keyspaces are the containers of data, similar to the schema or database in a relational database. Typically, keyspaces contain many tables.

Tables : Within the keyspaces, the tables are defined. Tables are also referred to as Column Families in the earlier versions of Cassandra. Tables contain a set of columns and a primary key, and they store data in a set of rows.

Columns : Columns define the structure of data in a table. Each column has an associated type, such as integer, text, double, and Boolean. These Cassandra data model components will be discussed in detail in this lesson.

The given query shows the statement for creating a table named employee with four columns empid, empfirstname, emplastname, and empsalary.

```
Create table Department (
deptID int primary key,
numEmployees counter);
Update Department set numEmployees = numEmployees + 1
    where deptID = 1000;
```

In this query, Empid is of the integer type, empfirstname and emplastname are the text type, and empsalary is defined as a double. Note that empid is defined as the primary key for the table. In database terms, primary key refers to a column that has unique values for each row in a table. For example, in an organization, two employees may have the same first and the last names. However, each employee is assigned a unique employee ID. Therefore, if you create a table to hold employee data, the employee ID can be used as the primary-key, as it is unique for all the rows of the employee table. On the other hand, employees' first or last name cannot be used as a primary key, as it cannot uniquely identify a row in the table.

9. ◆ *Explain in detail about Hive data manipulation, queries, data definition and data types.*
◆ ***Discuss the queries involved in HIVE data definition.*** (2016-17)

Hive Data Manipulation

Hive QL data manipulation data doesn't offer any row-level insert, update or delete operation. Therefore, data can be inserted into hive tables using either "bulk" load operations or writing the files into correct directories by other methods. Some of them are discussed below.

Hive QL Insert Data into Hive Tables from Queries :

```
Hive> INSERT OVERWRITE TABLE Employee
Partition (country= 'IN',state='KA')
SELECT * FROM emp_stage ese
WHERE ese.country='IN' AND ese.state='KA';
```

Create table and load them from Hive Queries

```
Hive> CREATE TABLE Employees

AS SELECT eno,ename,sal,address

FROM emp

WHERE country='IN';
```

Exporting Data out of Hive : If LOCAL keyword is used, Hive will write the data to local directory

```
Hive>INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/data'

SELECT name, age

FROM aliens

WHERE date_sighted >'2014-09-15'
```

Hive Data Definition : Hive Data Definition Language (DDL) is a subset of Hive SQL statements that describe the data structure in Hive by creating, deleting, or altering schema objects such as databases, tables, views, partitions, and buckets. Most Hive DDL statements start with the keywords CREATE, DROP, or ALTER.

DDL Command	Use With
CREATE	Database, Table
SHOW	Databases, Tables, Table Properties, Partitions, Functions, Index
DESCRIBE	Database, Table, view
USE	Database
DROP	Database, Table
ALTER	Database, Table
TRUNCATE	Table

Before moving forward, note that the Hive commands are case-insensitive.

(1) **CREATE DATABASE in Hive :** The CREATE DATABASE statement is used to create a database in the Hive. The DATABASE and SCHEMA are interchangeable. We can use either DATABASE or SCHEMA.

Syntax :

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS]
database_name
[COMMENT database_comment]
[LOCATION hdfs_path]
[WITH DBPROPERTIES (property_name=property_value ,)];
```

(2) **SHOW DATABASE in Hive :** The SHOW DATABASES statement lists all the databases present in the Hive.

Syntax :

```
SHOW (DATABASES|SCHEMAS);
```

(3) **DESCRIBE DATABASE in Hive :** The DESCRIBE DATABASE statement in Hive shows the name of Database in Hive, its comment (if set), and its location on the file system.

The EXTENDED can be used to get the database properties.

Syntax :

```
DESCRIBE DATABASE/SCHEMA [EXTENDED] db_name;
```

(4) **USE DATABASE in Hive :** The USE statement in Hive is used to select the specific database for a session on which all subsequent HiveQL statements would be executed.

Syntax :

```
USE database_name;
```

(5) DROP DATABASE in Hive : The DROP DATABASE statement in Hive is used to Drop (delete) the database.

The default behavior is RESTRICT which means that the database is dropped only when it is empty. To drop the database with tables, we can use CASCADE.

Syntax :

```
DROP (DATABASE|SCHEMA) [IF EXISTS]
database_name [RESTRICT|CASCADE];
```

(6) **ALTER DATABASE in Hive :** The ALTER DATABASE statement in Hive is used to change the metadata associated with the database in Hive.

Syntax for Changing Database Properties :

ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES (property_name=property_value,);

HIVE Data Types-Hive data types are categorized in numeric types, string types, misc types, and complex types. A list of Hive data types is given below.

Integer Types :

Type	Size	Range
TINYINT	1-byte signed integer	-128 to 127
SMALLINT	2-byte signed integer	32,768 to 32,767
INT	4-byte signed integer	2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Decimal Type :

Type	Size	Range
FLOAT	4-byte	Single precision floating point number
DOUBLE	8-byte	Double precision floating point number

Date/Time Types :

TIMESTAMP :

(1) It supports traditional UNIX timestamp with optional nanosecond precision.

(2) As Integer numeric type, it is interpreted as UNIX timestamp in seconds.

(3) As Floating point numeric type, it is interpreted as UNIX timestamp in seconds with decimal precision.

(4) As string, it follows java.sql.Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" (9 decimal place precision)

DATES : The Date value is used to specify a particular year, month and day, in the form YYYY--MM--DD. However, it didn't provide the time of the day. The range of Date type lies between 0000--01--01 to 9999--12--31.

String Types

STRING : The string is a sequence of characters. It values can be enclosed within single quotes (') or double quotes (").

Varchar : The varchar is a variable length type whose range lies between 1 and 65535, which specifies that the maximum number of characters allowed in the character string.

CHAR : The char is a fixed-length type whose maximum length is fixed at 255.

Complex Type :

Type	Size	Range
Struct	It is similar to C struct or an object where fields are accessed using the "dot" notation.	struct('James','Roy')
Map	It contains the key-value tuples where the fields are accessed using array notation.	map('first','James','last','Roy')
Array	It is a collection of similar type of values those indexable using zero-based integers.	

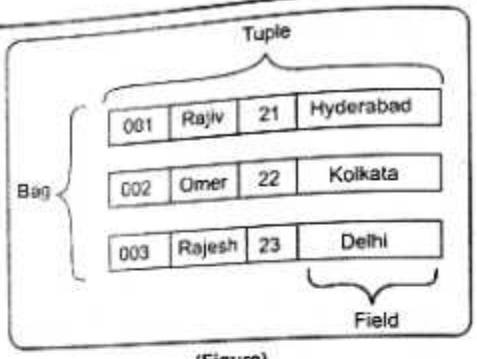
10. *Write in detail about Hive data model and Pig data model in detail.* **(2016-17)**

Hive Data Modeling

Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. Hive structures data into well-understood database concepts such as tables, rows, columns and partitions. It supports primitive types like Integers, Floats, Doubles, and Strings. Hive also supports Associative Arrays, Lists, Structs, and Serialize and Deserialized API is used to move data in and out of tables. The main components of Hive data modeling are tables, partitions, and buckets :

(1) **Tables :** Tables in Hive are created the same way it is done in RDBMS

(2) **Partitions :** Here, tables are organized into partitions for grouping similar types of data based on the partition key. Partition means dividing a table into coarse grained parts based on the value of a partition column such as 'data'. This makes it faster to do queries on slices of data



(Figure)

Atom : Any single value in Pig Latin, irrespective of their data, type is known as an Atom. It is stored as string and can be used as string and number. int, long, float, double, char array, and byte array are the atomic values of Pig. A piece of data or a simple atomic value is known as a field.

Example : 'raja' or '30'

Tuple : A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.

Example : (Raja, 30)

Bag : A bag is an unordered set of tuples. In other words, a collection of tuples (non-unique) is known as a bag. Each tuple can have any number of fields (flexible schema). A bag is represented by '{}'. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

Example : {(Raja, 30), (Mohammad, 45)}

A bag can be a field in a relation; in that context, it is known as **inner bag**.

Example : {Raja, 30, {9848022338, raja@gmail.com,}}

Map-A map (or data map) is a set of key-value pairs. The key needs to be of type char array and should be unique. The value might be of any type. It is represented by '[]'

Example : [name#Raja, age#30]

Relation : A relation is a bag of tuples. The relations in Pig Latin are unordered (there is no guarantee that tuples are processed in any particular order).

11. *Write down the difference between Pig and Hive in detail.*

Pig

Pig is used for the analysis of a large amount of data. It is abstract over MapReduce. Pig is used to perform all kinds of data manipulation operations in Hadoop. It provides the Pig-Latin language to write the code that contains many inbuilt functions like join, filter, etc. The two parts of the Apache Pig are Pig-Latin and Pig-Engine. Pig Engine is used to convert all these scripts into a specific map and reduce tasks. Pig abstraction is at a higher level. It contains less line of code as compared to MapReduce.

Hive

Hive is built on the top of Hadoop and is used to process structured data in Hadoop. Hive was developed by Facebook. It provides various types of querying language which is frequently known as Hive Query Language. Apache Hive is a data warehouse and which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop.

Difference between Pig and Hive :

S.No.	Pig	Hive
1.	Pig operates on the client side of a cluster.	Hive operates on the server side of a cluster.
2.	Pig uses pig-latin language.	Hive uses HiveQL language.
3.	Pig is a Procedural Data Flow Language.	Hive is a Declarative SQLish Language.
4.	It was developed by Yahoo.	It was developed by Facebook.
5.	It is used by Researchers and Programmers.	It is mainly used by Data Analysts.
6.	It is used to handle structured and semi-structured data.	It is mainly used to handle structured data.
7.	It is used for programming.	It is used for creating reports.

8.	Pig scripts end with .pig extension.	In Hive, all extensions are supported.
9.	It does not support partitioning.	It supports partitioning.
10.	It loads data quickly.	It loads data slowly.
11.	It does not support JDBC.	It supports JDBC.
12.	It does not support ODBC.	It supports ODBC.
13.	Pig does not have a dedicated metadata database.	Hive makes use of the exact variation of dedicated SQL-DDL language by defining tables beforehand.
14.	It supports Avro file format.	It does not support Avro file format.
15.	Pig is suitable for complex and nested data structures.	Hive is suitable for batch-processing OLAP systems.
16.	Pig does not support schema to store data.	Hive supports schema for data insertion in tables.

12. *What are views in HIVE? What is the difference between internal and external tables in HIVE?*

Views are generated based on user requirements. You can save any result set data as a view. The usage of view in Hive is same as that of the view in SQL. It is a standard RDBMS concept. We can execute all DML operations on a view.

Creating a View : You can create a view at the time of executing a SELECT statement. The syntax is as follows :

```
CREATE VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment] ...) ]
[COMMENT table_comment]
AS SELECT...
```

Example : Let us take an example for view. Assume employee table as given below, with the fields Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than ₹ 30000. We store the result in a view named emp_30000.


```
+------+----------+--------+--------------------+------+
| ID   | Name     | Salary | Designation        | Dept |
+------+----------+--------+--------------------+------+
|1201  | Gopal    | 45000  | Technical manager  | TP   |
|1202  | Manisha  | 45000  | Proofreader        | PR   |
|1203  | Masthanvali | 40000 | Technical writer | TP   |
|1204  | Krian    | 40000  | Hr Admin           | HR   |
|1205  | Kranthi  | 30000  | Op Admin           | Admin|
+------+----------+--------+--------------------+------+
```

The following query retrieves the employee details using the above scenario :

```
hive> CREATE VIEW emp_30000 AS
SELECT * FROM employee
WHERE salary>30000;
```

Dropping a View : Use the following syntax to drop a view :

```
DROP VIEW view_name
```

The following query drops a view named as emp_30000

```
hive> DROP VIEW emp_30000;
```

Difference between Internal and External Tables : Fundamentally, Hive knows two different types of tables : **Internal table** and the **External table**. The Internal table is also known as the managed table.

We can identify the internal or External tables using the DESCRIBE FORMATTED table_name statement in the Hive, which will display either MANAGED_TABLE or EXTERNAL_TABLE depending on the table type.

(1) Hive Internal Table : Hive owns the data for the internal tables. It is the default table in Hive. When the user creates a table in Hive without specifying it as external, then by default, an internal table gets created in a specific location in HDFS.By default, an internal table will be created in a folder path similar to /user/hive/warehouse directory of HDFS. We can override the default location by the location property during table creation.

If we drop the managed table or partition, the table data and the metadata associated with that table will be deleted from the HDFS.

(2) Hive External Table : Hive does not manage the data of the External table. We create an external table for external use as when we want to use the data outside the Hive. External tables are stored outside the warehouse directory. They can access

data stored in sources such as remote HDFS locations or Azure Storage Volumes. Whenever we drop the external table, then only the metadata associated with the table will get deleted, the table data remains untouched by Hive.

We can create the external table by specifying the **EXTERNAL** keyword in the Hive create table statement.

Difference between Hive Internal and External Table :

Sr. No.	Internal Table	External Table
1.	Hive moves table data to warehouse directory	Hive does not moves table data to warehouse directory
2.	Dropping deletes table data and metadata	Dropping deletes table's metadata
3.	Support TRUNCATE	No TRUNCATE support
4.	Support ACID transaction	No ACID transaction support
5.	Querry result Caching works	Querry result Caching does not works

□□

MCA
(SEM. III) MODEL PAPER
KCA – 022 : BIG DATA

Time : Three Hours *Maximum Marks : 70*

Note : Attempt *all* Sections. If require any missing data; then choose suitably.

SECTION – A

1. Attempt *all* questions in brief. (2 ×7 = 14)

(a) List the characteristics of big data.
(b) How to calculate risk in marketing?
(c) Why would you use inferential statistics in big data?
(d) What do you mean by shrading?
(e) State the usage of Hadoop pipes.
(f) Compare Master-Slave and peer to peer architecture in NoSql.
(g) What is the purpose of bloom filter?
(h) Compare the classic Map Reduce with YARN.
(i) Mention the usage of Grunt.
(j) How Date and Time data types are used in Hive?

SECTION – B

2. Attempt *any three* of the following : (3 ×7 = 21)

(a) Differentiate "Scale up and Scale out" Explain with an example How Hadoop uses Scale out feature to improve the Performance.
(b) Provide overview of HBase data model.
(c) Discuss in detail about the basic building blocks of Hadoop with a neat sketch.
(d) Explain in detail about Map-reduce Workflows.
(e) Relate crowd sourcing and big data. Justify the relationship with an example.

SECTION – C

3. Attempt *any one* part of the following : (1 ×7 = 7)

(a) (i) Discuss the different ways of constructing version stamps. What are their pros and cons?
(ii) Write in detail about the three dimensions of big data.
(b) (i) Explain with a neat sketch about the processing of a job in hadoop.
(ii) List the various operational modes of hadoop cluster configuration and explain in detail about configuring/installing the hadoop in local/standalone mode

4. **Attempt *any one* part of the following :** (1 ×7 =

(a) Draw and explain HDFS Architecture. Explain the functio of Name Node and Data Node. What is Secondary Nam Node? Is it a substitute to the Name node?

(b) What are views in HIVE? What is the difference betwee internal and external tables in HIVE?

5. **Attempt *any one* part of the following :** (1 ×7 =

(a) What are the benefits of Big Data? Discuss challenge under Big Data .How Big Date analytic can useful in th development of smart cities?

(b) What are the components of Pig execution environment?

6. **Attempt *any one* part of the following :** (1 ×7 =

(a) What are structured, unstructured and semi-structure data? Explain with examples.

(b) What are the different modes in which Hadoop can b installed and what is the use of each mode from applicatio and developer point of view?

7. **Attempt *any one* part of the following :** (1 ×7 =

(a) Define the role of combiner and partitioner in map reduce Application.

(b) Explain Master slave and peer-peer replication in detail