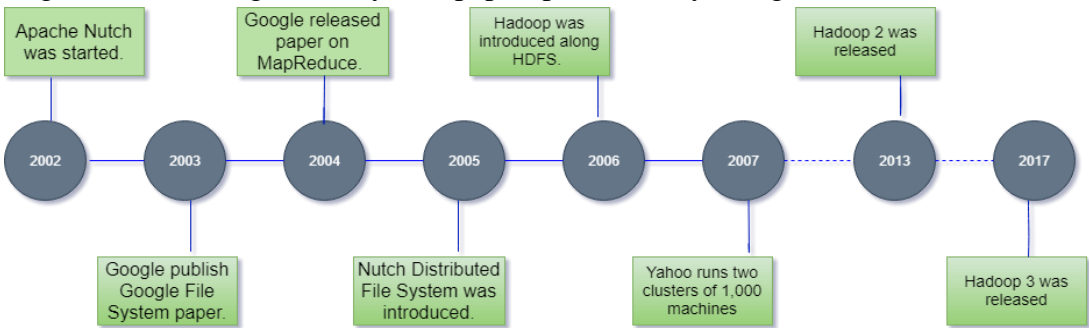
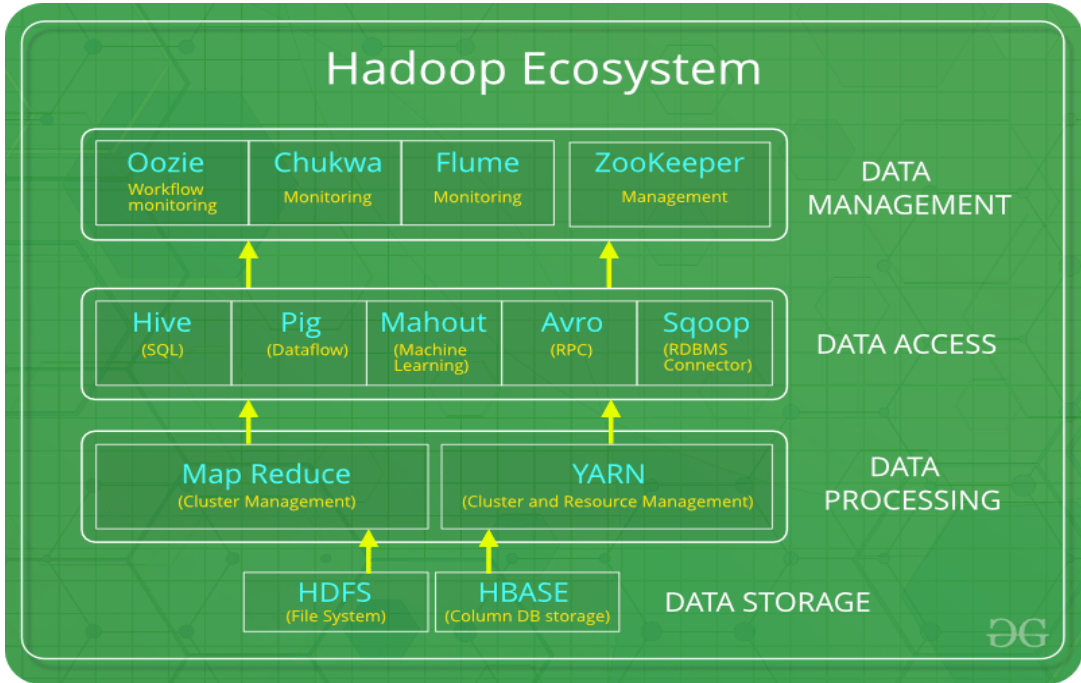


UNIT –I

INTRODUCTION TO BIG DATA AND HADOOP

1	Discuss in detail about Apache Hadoop and History of Hadoop?	[L2][CO1]	[12M]
	<p><u>Apache Hadoop</u></p> <p>The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.</p> <p>The Hadoop was started by Doug Cutting and Mike Cafarella in 2002. Its origin was the Google File System paper, published by Google.</p>  <pre> graph LR 2002((2002)) --- 2003((2003)) 2003 --- 2004((2004)) 2004 --- 2005((2005)) 2005 --- 2006((2006)) 2006 --- 2007((2007)) 2007 -.- 2013((2013)) 2013 -.- 2017((2017)) 2002 --- N1[Apache Nutch was started.] 2003 --- N2[Google publish Google File System paper.] 2004 --- N3[Google released paper on MapReduce.] 2005 --- N4[Nutch Distributed File System was introduced.] 2006 --- N5[Hadoop was introduced along HDFS.] 2007 --- N6[Yahoo runs two clusters of 1,000 machines] 2013 --- N7[Hadoop 2 was released] 2017 --- N8[Hadoop 3 was released] </pre> <ul style="list-style-type: none"> ○ In 2002, Doug Cutting and Mike Cafarella started to work on a project, Apache Nutch. It is an open source web crawler software project. ○ While working on Apache Nutch, they were dealing with big data. To store that data they have to spend a lot of costs which becomes the consequence of that project. This problem becomes one of the important reason for the emergence of Hadoop. ○ In 2003, Google introduced a file system known as GFS (Google file system). It is a proprietary distributed file system developed to provide efficient access to data. ○ In 2004, Google released a white paper on Map Reduce. This technique simplifies the data processing on large clusters. ○ In 2005, Doug Cutting and Mike Cafarella introduced a new file system known as NDFS (Nutch Distributed File System). This file system also includes Map reduce. ○ In 2006, Doug Cutting quit Google and joined Yahoo. On the basis of the Nutch project, Dough Cutting introduces a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System). Hadoop first version 0.1.0 released in this year. ○ Doug Cutting gave named his project Hadoop after his son's toy elephant. ○ In 2007, Yahoo runs two clusters of 1000 machines. ○ In 2008, Hadoop became the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds. ○ In 2013, Hadoop 2.2 was released. ○ In 2017, Hadoop 3.0 was released. ○ In 2018, Hadoop 3.1 was released 		

2a)	Examine the different types of digital data with examples?	[L4][CO1]	[6M]
	<p>1. Structured Data : Structured data is created using a fixed schema and is maintained in tabular format. The elements in structured data are addressable for effective analysis. It contains all the data which can be stored in the <u>SQL database</u> in a tabular format. Examples –Relational data, Geo-location, credit card numbers, addresses, etc.</p> <p>2. Unstructured Data : It is defined as the data in which is not follow a pre-defined standard or you can say that any does not follow any organized format. This kind of data is also not fit for the relational database because in the relational database you will see a pre-defined manner. Unstructured data is also very important for the big data domain and To manage and store Unstructured data there are many platforms to handle it like No-SQL Database. Examples –Word, PDF, text, media <u>logs</u>, etc.</p> <p>3. Semi-Structured Data : Semi-structured data is information that does not reside in a relational database but that have some organizational properties that make it easier to analyze. With some process, you can store them in a relational database but is very hard for some kind of semi-structured data, but semi-structured exist to ease space. Example –<u>XML data</u>.</p>		
b)	Summarize Big Data in terms of three dimensions volume, variety and velocity.	[L2][CO1]	[6M]
	<p>1. Volume: The name ‘Big Data’ itself is related to a size which is enormous.</p> <ul style="list-style-type: none"> • Volume is a huge amount of data. • To determine the value of data, size of data plays a very crucial role. If the volume of data is very large, then it is actually considered as a ‘Big Data’. This means whether a particular data can actually be considered as a Big Data or not, is dependent upon the volume of data. • Hence while dealing with Big Data it is necessary to consider a characteristic ‘Volume’. <p>2. Velocity: Velocity refers to the high speed of accumulation of data.</p> <ul style="list-style-type: none"> • In Big Data velocity data flows in from sources like machines, networks, social media, mobile phones etc. • There is a massive and continuous flow of data. This determines the potential of data that how fast the data is generated and processed to meet the demands. • Sampling data can help in dealing with the issue like ‘velocity’. • <i>Example:</i> There are more than 3.5 billion searches per day are made on Google. Also, Facebook users are increasing by 22%(Approx.) year by year. <p>3. Variety: It refers to nature of data that is structured, semi-structured and unstructured data.</p> <ul style="list-style-type: none"> • It also refers to heterogeneous sources. • Variety is basically the arrival of data from new sources that are both inside and outside of an enterprise. It can be structured, semi-structured and unstructured. • Structured data: This data is basically an organized data. It generally refers to data that has defined the length and format of data. • Semi- Structured data: This data is basically a semi-organised data. It is generally a form of data that do not conform to the formal structure of data. Log files are the examples of this type of data. 		

	<ul style="list-style-type: none"> • Unstructured data: This data basically refers to unorganized data. It generally refers to data that doesn't fit neatly into the traditional row and column structure of the relational database. Texts, pictures, videos etc. are the examples of unstructured data which can't be stored in the form of rows and columns. 		
3	Establish the evolution of Hadoop ecosystem with neat diagram.	[L3][CO1]	[12M]
	<p><i>Hadoop Ecosystem</i> 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 <i>four major elements of Hadoop</i> i.e. HDFS, MapReduce, YARN, and Hadoop Common.</p>  <p>HDFS:</p> <p>HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.</p> <p>HDFS consists of two core components i.e.</p> <ol style="list-style-type: none"> 1. Name node 2. Data Node <p>YARN:</p> <p>Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.</p> <p>Consists of three major components i.e.</p> <ol style="list-style-type: none"> 1. Resource Manager 2. Nodes Manager 3. Application Manager <p>MapReduce:</p> <p>By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.</p> <p>PIG:</p> <p>Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.</p> <ul style="list-style-type: none"> • It is a platform for structuring the data flow, processing and analyzing huge data sets. 		

- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.

HIVE:

With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).

- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: JDBC Drivers and HIVE Command Line.

Mahout:

Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or on the basis of algorithms.

- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning.

Apache Spark:

It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.

- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

Apache HBase:

It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.

Zookeeper: There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.

Oozie: Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

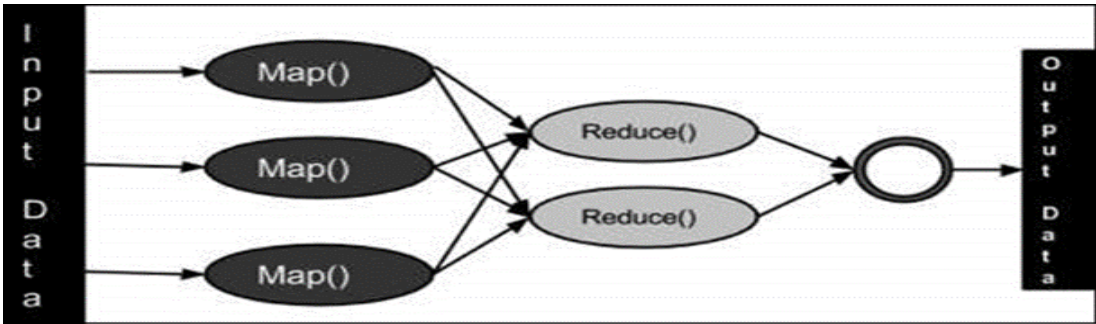
Flume is a distributed and reliable system designed for efficiently collecting, aggregating, and transporting large amounts of streaming data. It is commonly used in big data environments to ingest log files, social media data, clickstreams, and other high-volume data sources.

Sqoop is a tool used to transfer bulk data between Hadoop and external datastores, such as relational databases (MS SQL Server, MySQL). To process data using Hadoop, the data first needs to be loaded into Hadoop clusters from several sources.

Chukwa is an open source large-scale log collection system for monitoring large distributed systems. It is one of the common big data terms related to Hadoop. It is built on the top of Hadoop Distributed File System (HDFS) and Map/Reduce framework. It inherits Hadoop's robustness and scalability.

4	Distinguish between structure, unstructured and semi-structure data with an examples.				[L4][CO1]	[12M]
	Properties	Structured data	Semi-structured data	Unstructured data		
	Technology	It is based on Relational database table	It is based on XML/RDF(Resource Description Framework).	It is based on character and binary data		
	Transaction management	Matured transaction and various concurrency techniques	Transaction is adapted from DBMS not matured	No transaction management and no 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		
5a)	List out the challenges faced by big data.				[L1][CO1]	[6M]
<p>1. Sharing and Accessing Data:</p> <ul style="list-style-type: none">Perhaps the most frequent challenge in big data efforts is the inaccessibility of data sets from external sources.Sharing data can cause substantial challenges.It include the need for inter and intra- institutional legal documents.Accessing data from public repositories leads to multiple difficulties. <p>2. Privacy and Security:</p> <ul style="list-style-type: none">It is another most important challenge with Big Data. This challenge includes sensitive, conceptual, technical as well as legal significance.Most of the organizations are unable to maintain regular checks due to large amounts of data generation.There is some information of a person which when combined with external large data may lead to some facts of a person which may be secretive and he might not						

	<p>want the owner to know this information about that person.</p> <p>3. Analytical Challenges:</p> <ul style="list-style-type: none"> • There are some huge analytical challenges in big data which arise some main challenges questions like how to deal with a problem if data volume gets too large. • These large amount of data on which these type of analysis is to be done can be structured (organized data), semi-structured (Semi-organized data) or unstructured (unorganized data). • Either incorporate massive data volumes in the analysis or determine upfront which Big data is relevant. <p>4. Technical challenges:</p> <p>Quality of data:</p> <ul style="list-style-type: none"> • When there is a collection of a large amount of data and storage of this data, it comes at a cost. Big companies, business leaders and IT leaders always want large data storage. • For better results and conclusions, Big data rather than having irrelevant data, focuses on quality data storage. <p>Fault tolerance:</p> <ul style="list-style-type: none"> • Fault tolerance is another technical challenge and fault tolerance computing is extremely hard, involving intricate algorithms. <p>Scalability:</p> <ul style="list-style-type: none"> • Big data projects can grow and evolve rapidly. The scalability issue of Big Data has lead towards cloud computing. • It leads to various challenges like how to run and execute various jobs so that goal of each workload can be achieved cost-effectively. 		
b)	Examine the Significance of big data analytics.	[L3][CO1]	[6M]
	<p>Big data analytics helps organisations harness their data and use it to identify new opportunities. That, in turn, leads to smarter business moves, more efficient operations, higher profits and happier customers. Businesses that use big data with advanced analytics gain value in many ways, such as:</p> <p>Reducing cost. Big data technologies like cloud-based analytics can significantly reduce costs when it comes to storing large amounts of data (for example, a data lake). Plus, big data analytics helps organisations find more efficient ways of doing business.</p> <p>Making faster, better decisions. The speed of in-memory analytics – combined with the ability to analyse new sources of data, such as streaming data from IoT – helps businesses analyse information immediately and make fast, informed decisions.</p> <p>Developing and marketing new products and services. Being able to gauge customer needs and customer satisfaction through analytics empowers businesses to give customers what they want, when they want it. With big data analytics, more companies have an opportunity to develop innovative new products to meet customers' changing needs.</p> <p>The importance of big data analytics</p> <p>Big data analytics is important because it helps companies leverage their data to identify opportunities for improvement and optimisation. Across different business segments, increasing efficiency leads to overall more intelligent operations, higher profits, and satisfied customers. Big data analytics helps companies reduce costs and develop better, customer-centric products and services.</p> <p>Data analytics helps provide insights that improve the way our society functions. In health care, big data analytics not only keeps track of and analyses individual records but it plays a critical role in measuring outcomes on a global scale.</p>		

	During the COVID-19 pandemic, big data-informed health ministries within each nation's government on how to proceed with vaccinations and devised solutions for mitigating pandemic outbreaks in the future.		
6	<p>Discuss about the Analysis of data through Unix tools and Hadoop</p> <p><u>Analyzing data with UNIX</u></p> <p>To understand how to work with Unix, data — Weather Dataset is used. Weather sensors gather information consistently in numerous areas over the globe and assemble an enormous volume of log information, which is a decent possibility for investigation with MapReduce in light of the fact that is required to process every one of the information, and the information is record-oriented and semi-organized.</p> <p>The information utilized is from the <i>National Climatic Data Center</i> or NCDC. The data is in line-oriented ASCII format & the line is a record. Data files are organized by date and weather station. The organization underpins a rich arrangement of meteorological components, huge numbers of which are discretionary or with variable information lengths. For straightforwardness, centre around the fundamental components, for example, temperature, which is constantly present and are of fixed width.</p> <p><u>Sample code</u></p> <pre>#!/user / bin / env bash for year in all/* do # using each year data file echo -ne `basename \$year .gz`"\t" # substring to search in gunzip -c \$year \ awk '{ temp = substr(\$0, 88, 5) + 0; q = substr(\$0, 93, 1); if (temp != 9999 && q ~ /[01459]/ && temp > max) max = temp } END { print max }'</pre> <p>done</p> <p><u>Analysing Data with Hadoop</u></p> <p>To take advantage of the parallel processing that Hadoop provides, we need to express our query as a MapReduce job. After some local, small-scale testing, we will be able to run it on a cluster of machines.</p> <p>MapReduce works by breaking the processing into two phases: the map phase and the reduce phase.</p>  <p>Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function. The input to our map phase is the raw NCDC data. We choose a text input format that gives us each line in the dataset as a text value. The key is the offset of the beginning of the line from the beginning of the file, but as we have no need for this, we ignore it.</p> <p>These lines are presented to the map function as the key-value pairs: (0, 0067011990999991950051507004...9999999N9+00001+99999999999...) (106,</p>	[L2][CO1]	[12M]

0043011990999999**1950**051512004...9999999N9+**0022**1+99999999999...) (212,
 0043011990999999**1950**051518004...9999999N9-**0011**1+99999999999...) (318,
 0043012650999999**1949**032412004...0500001N9+**0111**1+99999999999...) (424,
 0043012650999999**1949**032418004...0500001N9+**0078**1+99999999999...)

The keys are the line offsets within the file, which we ignore in our map function. The map function merely extracts the year and the air temperature (indicated in bold text), and emits them as its output (the temperature values have been interpreted as integers):

(1950, 0)

(1950, 22)

(1950, -11)

(1949, 111)

(1949, 78)

The output from the map function is processed by the MapReduce framework before being sent to the reduce function. This processing sorts and groups the key-value pairs by key. So, continuing the example, our reduce function sees the following input:

(1949, [111, 78])

(1950, [0, 22, -11])

Each year appears with a list of all its air temperature readings. All the reduce function has to do now is iterate through the list and pick up the maximum reading:

(1949, 111)

(1950, 22)

This is the final output: the maximum global temperature recorded in each year.

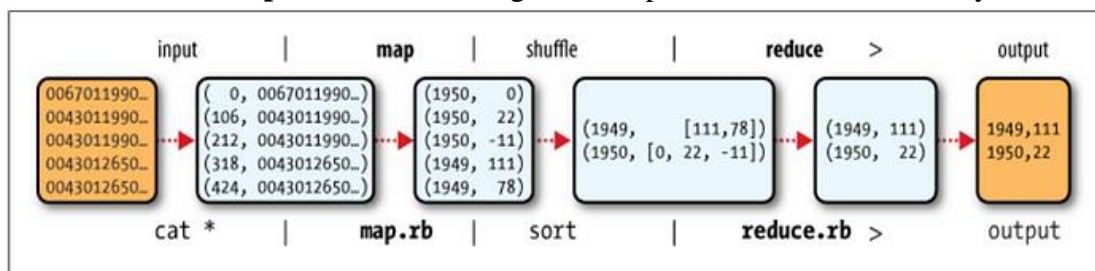


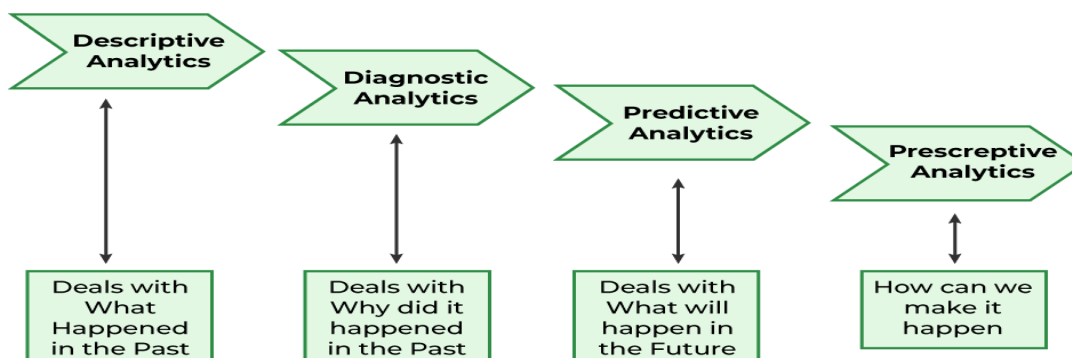
Figure 2-1. MapReduce logical data flow

7a) What is big data analytics? Identify the Classification of Analytics.

[L3][CO1]

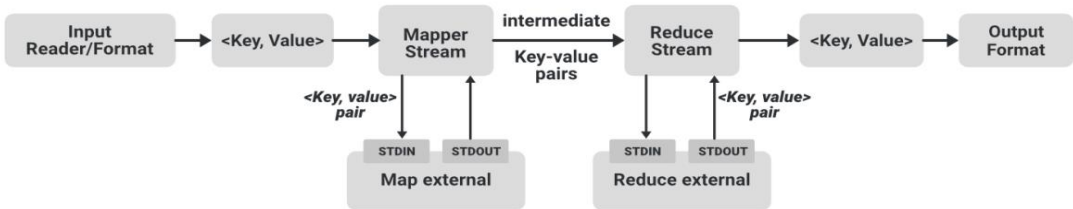
[6M]

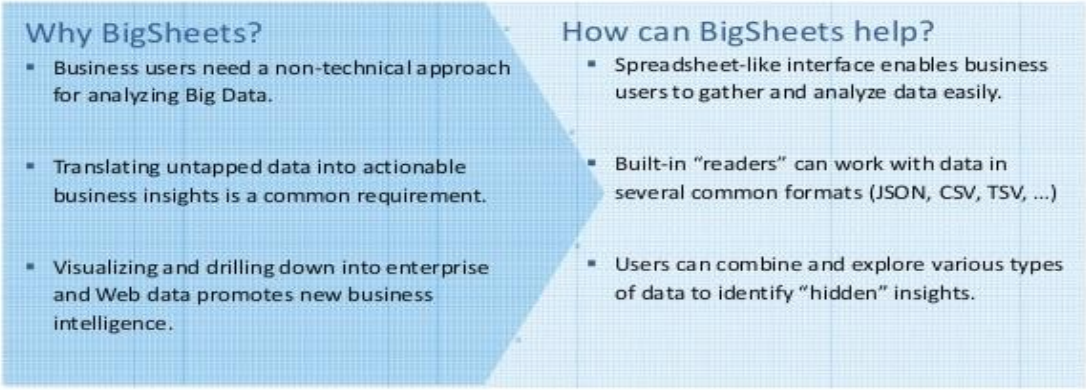
Big data analytics refers to the methods, tools, and applications used to collect, process, and derive insights from varied, high-volume, high-velocity data sets. These data sets may come from a variety of sources, such as web, mobile, email, social media, and networked smart devices.



Predictive Analytics

Predictive analytics turn the data into valuable, actionable information. predictive analytics uses data to determine the probable outcome of an event or a likelihood of a situation occurring. Predictive analytics holds a variety of statistical techniques from modeling, machine learning, data mining, and game theory that analyze current and historical facts to make predictions about a future event.

	<p>Descriptive Analytics</p> <p>Descriptive analytics looks at data and analyze past event for insight as to how to approach future events. It looks at past performance and understands the performance by mining historical data to understand the cause of success or failure in the past. Almost all management reporting such as sales, marketing, operations, and finance uses this type of analysis.</p> <p>Prescriptive Analytics</p> <p>Prescriptive Analytics automatically synthesize big data, mathematical science, business rule, and machine learning to make a prediction and then suggests a decision option to take advantage of the prediction.</p> <p>Prescriptive analytics goes beyond predicting future outcomes by also suggesting action benefits from the predictions and showing the decision maker the implication of each decision option.</p> <p>Diagnostic Analytics</p> <p>In this analysis, we generally use historical data over other data to answer any question or for the solution of any problem. We try to find any dependency and pattern in the historical data of the particular problem.</p>		
b)	<p>Illustrate in detail about Hadoop streaming</p>	[L2][CO1]	[6M]
	<p>Hadoop streaming is a utility or feature that comes with a Hadoop distribution that allows developers or programmers to write the Map-Reduce program using different programming languages like Ruby, Perl, Python, C++, etc. We can use any language that can read from the standard input(STDIN) like keyboard input and all and write using standard output(STDOUT). We all know the Hadoop Framework is completely written in java but programs for Hadoop are not necessarily need to code in Java programming language. feature of Hadoop Streaming is available since Hadoop version 0.14.1.</p> <p>How Hadoop Streaming Works</p> <h3 style="text-align: center;">Hadoop Streaming</h3>  <pre> graph LR A[Input Reader/Format] -- "<Key, Value>" --> B[Mapper Stream] B -- "intermediate Key-value pairs" --> C[Reduce Stream] C -- "<Key, Value>" --> D[Output Format] E[Map external: STDIN, STDOUT] -- "<Key, value> pair" --> B B -- "<Key, value> pair" --> E F[Reduce external: STDIN, STDOUT] -- "<Key, value> pair" --> C C -- "<Key, value> pair" --> F </pre> <p>In that, we have an Input Reader which is responsible for reading the input data and produces the list of key-value pairs. We can read data in .csv format, in delimiter format, from a database table, image data(.jpg, .png), audio data etc. The only requirement to read all these types of data is that we have to create a particular input format for that data with these input readers.</p> <p>These external map processes are not part of the basic MapReduce flow. This external mapper will take input from STDIN and produce output to STDOUT. As the key-value pairs are passed to the internal mapper the internal mapper process will send these key-value pairs to the external mapper.</p> <p>Similarly, Reducer does the same thing. Once the intermediate key-value pairs are processed through the shuffle and sorting process they are fed to the internal reducer which will send these pairs to external reducer process that are working separately through the help of STDIN and gathers the output generated by external reducers with help of STDOUT.</p>		

8a)	What is Big Sheets? What can be done with big sheets?	[L1][CO6]	[6M]
	<p>BigSheets is a browser-based analytic tool included in the InfoSphere® BigInsights™ Console that you use to break large amounts of unstructured data into consumable, situation-specific business contexts.</p>  <p>What you can do with big sheets</p> <ul style="list-style-type: none"> ▪ Model “big data” collected from various sources in spreadsheet-like structures ▪ Filter and enrich content with built-in functions ▪ Combine data in different workbooks ▪ Visualize results through spreadsheets, charts ▪ Export data into common formats (if desired) <p>Big sheets is a user Interface program developed for non- technical business users to enable data gathering and analysis. It provides new insights by comparing various data from different resources. It allows the users to explore the risk factor.</p> <p>Big sheets generates actionable workflow by Gathering the information, Extracting or Analyzing the information and Exploring or visualizing the data in a creative way which enable the users to work. Big sheets can translate untapped data in to actionable business insights is one of the common requirements.</p> <p>Big sheets can translate untapped data in to actionable business insights is one of the common requirements. Data visualization and mining into commodity purpose promotes web business.</p>		
b)	Explain in detail about Infosphere Big Insights.	[L2][CO1]	[6M]
	<p>IBM InfoSphere BigInsights is a platform for managing and analyzing large amounts of structured and unstructured data in a reliable, fault-tolerant manner.</p> <p>Introduction to InfoSphere :</p> <ul style="list-style-type: none"> • InfoSphere Information Server provides a single platform for data integration and governance. • The components in the suite combine to create a unified foundation for enterprise information architectures, capable of scaling to meet any information volume requirements. • You can use the suite to deliver business results faster while maintaining data quality and integrity throughout your information landscape. • InfoSphere Information Server helps your business and IT personnel collaborate to 		

	<p>understand the meaning, structure, and content of information across a wide variety of sources.</p> <ul style="list-style-type: none"> • By using InfoSphere Information Server, your business can access and use information in new ways to drive innovation, increase operational efficiency, and lower risk. <p>Need for a solution like BigInsights</p> <ul style="list-style-type: none"> • Build sophisticated predictive models from the combination of existing information and big data information flows, providing a level of depth that only analytics applied at a large scale can offer. • Broadly and automatically perform consumer sentiment and brand perception analysis on data gathered from across the Internet, at a scale previously impossible using partially or fully manual methods. <p>Highlights of InfoSphere BigInsights</p> <p>BigInsights allows organizations to cost-effectively analyze a wide variety and large volume of data to gain insights that were not previously possible. BigInsights is focused on providing enterprises with the capabilities they need to meet critical business requirements while maintaining compatibility with the Hadoop project.</p> <p>BigInsights includes a variety of IBM technologies that enhance and extend the value of open-source Hadoop software to facilitate faster time-to-value, including application accelerators, analytical facilities, development tools, platform improvements and enterprise software integration.</p>		
9a)	Discriminate the Big Data in Healthcare, Transportation & Medicine.	[L5][CO1]	[6M]
	<p><u>Transportation:</u> Data analytics increasingly power mobility information and insights – transforming transportation planning by making it easier, faster, cheaper, and safer to collect and understand critical information.</p> <ul style="list-style-type: none"> • Prioritize projects accurately to guide effective resource investment and make the biggest impact. • Make informed decisions based on recent, accurate data, not on guesses or input from a few vocal stakeholders. • Accurately and quickly measure results of transportation initiatives, enabling timely adjustment and optimization. <p><u>Health care</u></p> <p>Healthcare as a big-data repository. Healthcare is a multi-dimensional system established with the sole aim for the prevention, diagnosis, and treatment of health-related issues or impairments in human beings.</p> <p>Electronic health records</p> <p>It is important to note that the National Institutes of Health (NIH) recently announced the “All of Us” initiative (https://allofus.nih.gov/) that aims to collect one million or more patients’ data such as EHR, including medical imaging, socio-behavioral, and environmental data over the next few years.</p> <p>Digitization of healthcare and big data</p> <p>Similar to EHR, an electronic medical record (EMR) stores the standard medical and clinical data gathered from the patients. EHRs, EMRs, personal health record (PHR), medical practice management software (MPM), and many other healthcare data components collectively have the potential to improve the quality, service efficiency, and costs of healthcare along with the reduction of medical errors.</p> <p><u>Medicine</u></p> <p>Examples of patient data used for analysis include blood sugar level, temperature, and blood test results. Analysing these with a system can help professionals better</p>		

	<p>understand complex healthcare environments, develop systematic approaches to improve patient outcomes, and continuously enhance healthcare processes.</p> <p>The diagram illustrates the integration of various data sources for healthcare analytics. On the left, a 'Multi Omics Profile' is shown as a stack of boxes: Genomics, Transcriptomics, Proteomics, Metabolomics, Glycomics, Fluxomics, Epigenomics, and Phenomics. An arrow points from this profile to a central 'Data Integration (Computational and systems biology)' box. Above this, a dashed box labeled 'Health Care Analytics' contains three stacked boxes: 'Electronic health record', 'Clinical information', and 'Patient health information'. An arrow points from this box to the central 'Data Integration' box. Below the 'Data Integration' box, an arrow points to a group of stylized human figures, with the text 'Personalized treatment' above them. A large curved arrow on the right side of the diagram points from the 'Personalized treatment' area back up to the 'Health Care Analytics' box, indicating a feedback loop.</p>		
b)	<p>Why organizations using big data for competitive advantage?</p>	[L4][CO1]	[6M]
	<p>Analysing the datasets a business gathers is just one part of the big data process. Big data experts also need to understand what the company wants to gain out of the exercise and how they intend to use the information to their benefit.</p> <p>Confident decision making – Analytics aims to improve decision making and big data continues to support this. With so much information available big data can help business speed up their decision making process while still being confident in the choice they have made.</p> <p>Asset optimisation – Big data means that companies can monitor assets at an individual level. This means they can better optimise assets based on the data sourced, improving productivity, extending the lifespan of assets and reducing the downtime some assets may need. This provides a competitive advantage by ensuring the business are getting the most out of its assets and links with cutting costs.</p> <p>Cost reduction – Big data can help businesses cut down their outgoings. From analysing energy usage to evaluating the effectiveness of staff working patterns, data collected by companies can help them identify where they can make cost savings without having a negative impact on business operations.</p> <p>Improve customer engagement – When browsing online customers make certain decisions indicating their preferences, habits and tendencies that can then be used to improve and tailor customer dialogue, which could then translate into increased business. Understanding what each customer is looking for through the data stored on them not only means you can target them with certain products but it also gives service a personal touch that many consumers today have come expect.</p> <p>Identify new revenue streams – Big data analytics can also help businesses identify new revenue streams and expand into other areas. Understanding customer trends and decisions allows firms to make decisions about the direction they should go. The data businesses collect can also potentially be sold, adding a further revenue stream and the potential to build partnerships with other businesses.</p> <p>Protecting against fraud – By collecting and monitoring your transactional data, you'll be able to detect problematic trends or be alerted to possible fraud right when it happens, which reduces the risk that your company will lose money.</p>		
10a)	<p>How to implement IBM Big Data Strategy?</p>	[L2][CO1]	[6M]
	<p>IBM Big Data Strategy :</p> <p>IBM, a US-based computer hardware and software manufacturer, had implemented a Big Data strategy.</p> <p>Where the company offered solutions to store, manage, and analyze the huge amounts</p>		

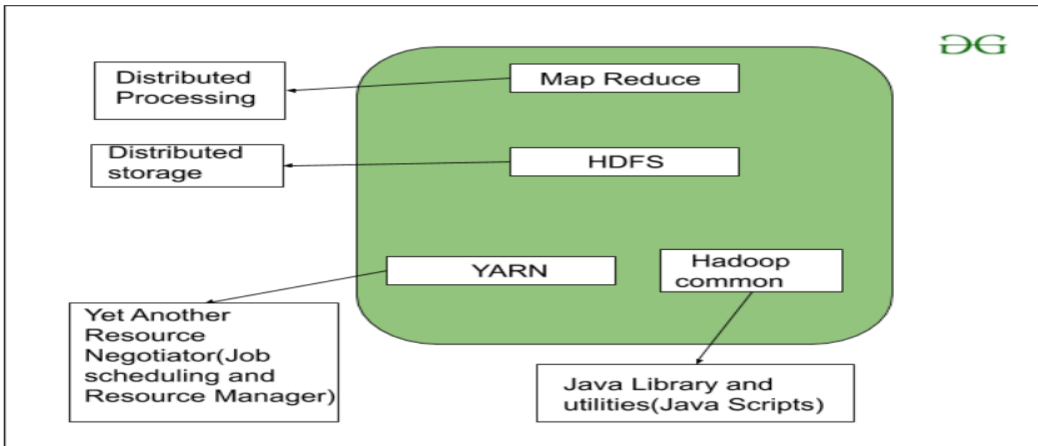
	<p>of data generated daily and equipped large and small companies to make informed business decisions.</p> <p>The company believed that its Big Data and analytics products and services would help its clients become more competitive and drive growth.</p> <p><u>Issues :</u></p> <ul style="list-style-type: none"> • Understand the concept of Big Data and its importance to large, medium, and small companies in the current industry scenario. • Understand the need for implementing a Big Data strategy and the various issues and challenges associated with this. • Analyze the Big Data strategy of IBM. • Explore ways in which IBM's Big Data strategy could be improved further. <p><u>DEVELOP YOUR STRATEGY</u></p> <p><u>1. Understand your business objectives</u></p> <p>Identify the most compelling use cases</p> <p>If you had better access to high quality data, where in your organization could you solve problems.</p> <p>Know the tools in your toolkit</p> <p>Work hand in hand with IT to take your data strategy to the next level by leveraging existing infrastructure and technologies, as well as new and leading-edge technologies.</p> <p><u>2. Assess your current state</u></p> <p>Unpack pain points to reveal blockers and gaps</p> <p>Organizational silos often underlie challenges with data integration, data management and workflows. In fact, 82% of enterprises are inhibited by data silos.</p> <p>Prioritize critical data elements for governance</p> <p>Keeping a handle on critical and regulated data elements—such as names, addresses, social security numbers and more—is essential to running various business systems.</p> <p><u>3.Measure progress toward your goals</u></p> <p>Use data for the win to contribute directly to the growth of the company.</p> <p>Capture your data strategy highlights—and share them^{[1][SEP]}</p> <p>A look at the big picture—where you are and what's ahead—gives you strategic context to make actionable plans for delivery and scale.</p> <p><u>4. Establish controls</u></p> <p>Communicate results for maximum visibility</p> <p>Let people know how much your efforts are paying off. “Build credibility with business process and data connection, and by telling a compelling story with your data.</p> <p>Build strong partnerships across the organization</p> <p>On the most basic level, your job as a data leader is to help your organization make the wisest decisions about data collection, management and use.</p>		
b)	Generalize the list of tools related to Hadoop.	[L6][CO1]	[6M]
	<p>1. Apache Spark</p> <p>Apache spark in an open-source processing engine that is designed for ease of analytics operations. It is a cluster computing platform that is designed to be fast and made for general purpose uses.</p> <p>2. Map Reduce</p> <p>MapReduce is just like an Algorithm or a data structure that is based on the YARN framework. The primary feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster, which Makes Hadoop working so fast.</p> <p>3. Apache Hive</p> <p>Apache Hive is a Data warehousing tool that is built on top of the Hadoop, and Data</p>		

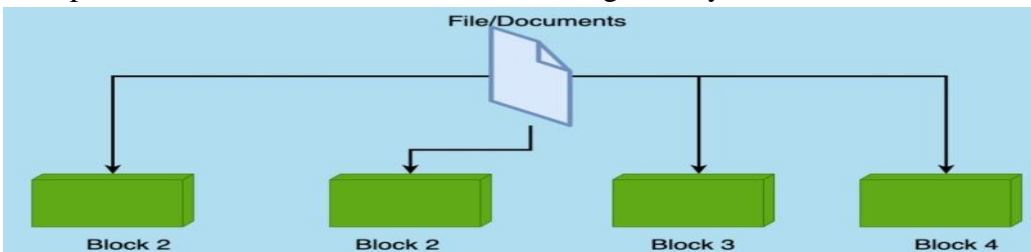
<p>Warehousing is nothing but storing the data at a fixed location generated from various sources. Hive is one of the best tools used for data analysis on Hadoop. The query language of hive is known as HQL or HIVEQL.</p> <p>4. Apache Impala</p> <p>Apache Impala is an open-source SQL engine designed for Hadoop. Impala overcomes the speed-related issue in Apache Hive with its faster-processing speed. Apache Impala uses similar kinds of SQL syntax, ODBC driver, and user interface as that of Apache Hive.</p> <p>5. Apache Mahout</p> <p>The name <i>Mahout</i> is taken from the Hindi word Mahavat which means the elephant rider. Apache Mahout runs the algorithm on the top of Hadoop, so it is named Mahout. Mahout is mainly used for implementing various Machine Learning algorithms on our Hadoop.</p> <p>6. Apache Pig</p> <p>This Pig was Initially developed by Yahoo to get ease in programming. Apache Pig has the capability to process an extensive dataset as it works on top of the Hadoop. Apache pig is used for analyzing more massive datasets by representing them as dataflow.</p> <p>7. HBase</p> <p>HBase is nothing but a non-relational, NoSQL distributed, and column-oriented database. HBase consists of various tables where each table has multiple numbers of data rows. These rows will have multiple numbers of column family's, and this column family will have columns that contain key-value pairs.</p> <p>8. Apache Sqoop</p> <p>Sqoop is a command-line tool that is developed by Apache. The primary purpose of Apache Sqoop is to import structured data i.e., RDBMS like MySQL, SQL Server, Oracle to our HDFS(Hadoop Distributed File System).</p> <p>9. Tableau</p> <p>Tableau is a data visualization software that can be used for data analytics and business intelligence. It provides a variety of interactive visualization to showcase the insights of the data and can translate the queries to visualization and can also import all ranges and sizes of data.</p> <p>10. Apache Storm</p> <p>Apache Storm is a free open source distributed real-time computation system build using Programming languages like Clojure and java. It can be used with many programming languages. Apache Storm is used for the Streaming process, which is very faster.</p>	
---	--

UNIT –II

HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

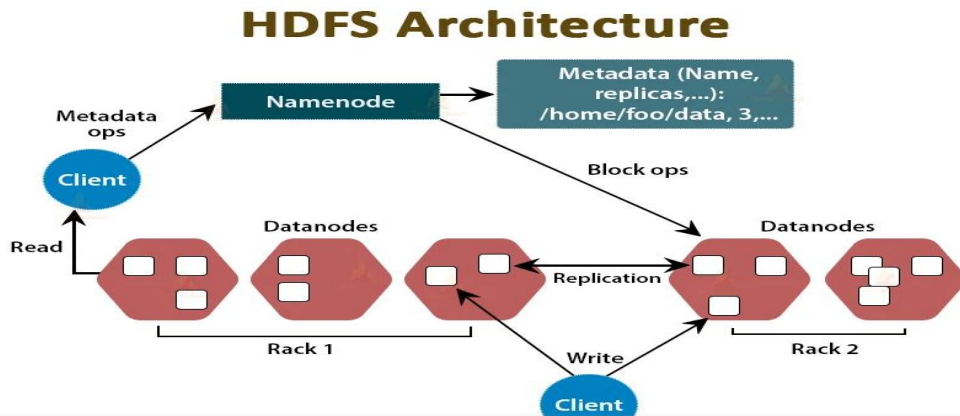
1	Illustrate the concepts of HDFS.	[L3][CO2]	[12M]
	<p>What is HDFS</p> <p>Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.</p> <p>It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.</p> <p>Where to use HDFS</p> <ul style="list-style-type: none"> ○ Very Large Files: Files should be of hundreds of megabytes, gigabytes or more. ○ Streaming Data Access: The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern. ○ Commodity Hardware :It works on low cost hardware. <p>Where not to use HDFS</p> <ul style="list-style-type: none"> ○ Low Latency data access: Applications that require very less time to access the first data should not use HDFS as it is giving importance to whole data rather than time to fetch the first record. ○ Lots Of Small Files:The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible. ○ Multiple Writes:It should not be used when we have to write multiple times. <div data-bbox="172 1220 1236 1668" data-label="Diagram"> <p>The diagram shows a '100 TB file' in a blue box. An arrow points down to a 'NameNode (masterNode)' represented by a red cylinder. From the NameNode, three arrows point to three colored boxes (yellow, blue, and pink). These boxes represent the file divided into blocks. Arrows from these blocks point to a grid of 'DataNodes (slaveNodes)', which are represented by red cylinders. Some cylinders have colored segments (yellow, blue, pink) indicating they store specific blocks. A label 'D1' is on one cylinder. Text at the bottom says 'File divided into blocks of 10TB each' and 'The blocks are then replicated among data nodes'.</p> </div> <p>HDFS Concepts:</p> <ol style="list-style-type: none"> 1. Blocks: A Block is the minimum amount of data that it can read or write. HDFS blocks are 128 MB by default and this is configurable. Files in HDFS are broken into block-sized chunks, which are stored as independent units. Unlike a file system, if the file in HDFS is smaller than block size, then it does not occupy full block's size, i.e. 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only. The HDFS block size is large just to minimize the cost of seek. 2. Name Node: HDFS works in master-worker pattern where the name node acts as master. Name Node is controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS; the metadata information being file permission, names and location of each block. The metadata are small, so it is stored in the memory of name node, allowing faster access to data. Moreover the 		

	<p>HDFS cluster is accessed by multiple clients concurrently, so all this information is handled by a single machine. The file system operations like opening, closing, renaming etc. are executed by it.</p> <p>3. Data Node: They store and retrieve blocks when they are told to; by client or name node. They report back to name node periodically, with list of blocks that they are storing. The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name node.</p>		
2	Explain Hadoop Architecture and its Components with neat diagram, What are the advantages of Hadoop?	[L1][CO2]	[12M]
	<p>Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google.</p> <ul style="list-style-type: none"> • MapReduce • HDFS(Hadoop Distributed File System) • YARN(Yet Another Resource Negotiator) • Common Utilities  <p>The diagram illustrates the Hadoop Architecture. A central green rounded rectangle contains three main components: 'Map Reduce' at the top, 'HDFS' in the middle, and 'YARN' at the bottom. To the left of this central box, three separate boxes are connected to the central components by arrows: 'Distributed Processing' points to 'Map Reduce', 'Distributed storage' points to 'HDFS', and 'Yet Another Resource Negotiator(Job scheduling and Resource Manager)' points to 'YARN'. To the right of the central box, a box labeled 'Hadoop common' is connected to the central box. Below the central box, a box labeled 'Java Library and utilities(Java Scripts)' is connected to the central box. The Hadoop logo is visible in the top right corner of the diagram.</p> <p>Components of Hadoop</p> <p>1.MapReduce: A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google. The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.</p> <p>2.HDFS: Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application. It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.</p> <p>3.YARN: 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</p> <p>Hadoop common Utilities: Hadoop common or Common utilities are nothing but our java library and java files</p>		

	<p>or we can say the java scripts that we need for all the other components present in a Hadoop cluster. these utilities are used by HDFS, YARN, and MapReduce for running the cluster.</p> <p><u>Advantages of Hadoop:</u></p> <p>1. Cost Hadoop is open-source and uses cost-effective commodity hardware which provides a cost-efficient model, unlike traditional Relational databases that require expensive hardware and high-end processors to deal with Big Data.</p> <p>2. Scalability Hadoop is a highly scalable model. A large amount of data is divided into multiple inexpensive machines in a cluster which is processed parallelly.</p> <p>3. Flexibility Hadoop is designed in such a way that it can deal with any kind of dataset like structured(MySql Data), Semi-Structured(XML, JSON), Un-structured (Images and Videos) very efficiently.</p> <p>4. Speed In DFS(Distributed File System) a large size file is broken into small size file blocks then distributed among the Nodes available in a Hadoop cluster, as this massive number of file blocks are processed parallelly which makes Hadoop faster.</p> <p>5.Fault Tolerance Hadoop uses commodity hardware(inexpensive systems) which can be crashed at any moment. In Hadoop data is replicated on various DataNodes in a Hadoop cluster which ensures the availability of data if somehow any of your systems got crashed.</p> <p>6.High Throughput Hadoop works on Distributed file System where various jobs are assigned to various Data node in a cluster, the bar of this data is processed parallelly in the Hadoop cluster which produces high throughput.</p> <p>7.Minimum Network Traffic In Hadoop, each task is divided into various small sub-task which is then assigned to each data node available in the Hadoop cluster. Each data node process a small amount of data which leads to low traffic in a Hadoop cluster.</p>		
3.	Explain the block, name node and data node in Hadoop file system	[L2][CO2]	[12M]
	<p>What is Block: The smallest quantity of data it can read or write is called a block. The default size of HDFS blocks is 128 MB, although this can be changed. HDFS files are divided into block-sized portions and stored as separate units. Unlike a file system, if a file in HDFS is less than the block size, it does not take up the entire block size; for example, a 5 MB file saved in HDFS with a block size of 128 MB takes up just 5 MB of space. The HDFS block size is big solely to reduce search costs.</p>  <p>Replication Method for HDFS Blocks The block size and replication factor in HDFS may be customised per file. The number of replicas can be set up programmatically by an application. It can be</p>		

supplied at the time of file creation and altered later if necessary. Name Node is in charge of determining the block size. HDFS makes two copies of the data in the same rack and the third copy in a separate rack using a rack-aware replica placement strategy.

The default block size in HDFS is 64 MB for Hadoop 1.1x and 128 MB for Hadoop 2.x and above. Depending on the cluster's size, this block size can be changed. HDFS blocks are larger than disc blocks, primarily to reduce seek costs. The default replication size in an older version of Hadoop is three, which implies that each block is duplicated three times and stored on various nodes.



NameNode

NameNode can be regarded as the system's master. It keeps track of the file system tree and metadata for all of the system's files and folders. Metadata information is stored in two files: 'Namespace image' and 'edit log.' Namenode is aware of all data nodes carrying data blocks for a particular file, but it does not keep track of block positions. When the system starts, this information is rebuilt from data nodes each time.

DataNode:

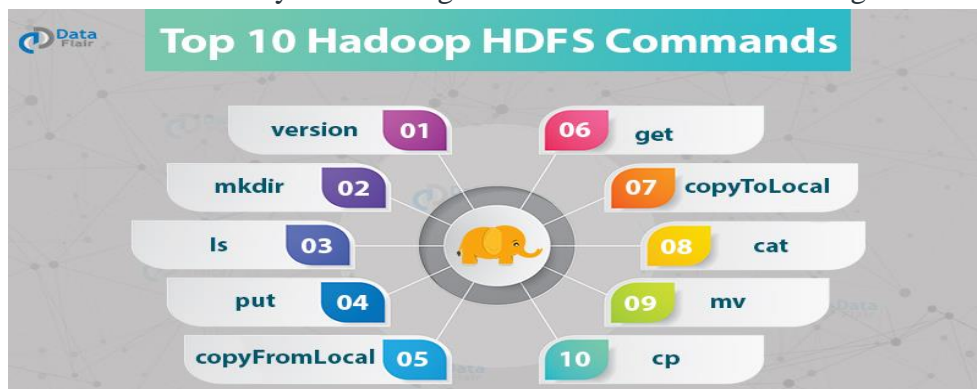
The data node is a commodity computer with the GNU/Linux operating system and data node software installed. In a cluster, there will be a data node for each node (common hardware/system). These nodes are in charge of the system's data storage. Data nodes respond to client requests by performing read-write operations on file systems. They also carry out actions such as block creation, deletion, and replication in accordance with the name node's instructions.

4 Determine the basic commands in Hadoop command line interface

[L3][CO2]

[12M]

HDFS is the primary or major component of the Hadoop ecosystem which is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.



Commands:

1.ls: This command is used to list all the files. Use *lsr* for recursive approach. It is useful when we want a hierarchy of a folder.

	<p>Syntax:</p> <pre>bin/hdfs dfs -ls <path></pre> <p>2.mkdir: To create a directory. In Hadoop <i>dfs</i> there is no home directory by default.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -mkdir <folder name></pre> <p>creating home directory:</p> <pre>hdfs/bin -mkdir /user</pre> <pre>hdfs/bin -mkdir /user/username -> write the username of your computer</pre> <p>3.copyFromLocal (or) put: To copy files/folders from local file system to hdfs store. This is the most important command. Local filesystem means the files present on the OS.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -copyFromLocal <local file path> <dest(present on hdfs)></pre> <p>4.cat: To print file contents.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -cat <path></pre> <p>5.moveFromLocal: This command will move file from local to hdfs.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -moveFromLocal <local src> <dest(on hdfs)></pre> <p>6.cp: This command is used to copy files within hdfs. Lets copy folder <i>geeks</i> to <i>geeks_copied</i>.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -cp <src(on hdfs)> <dest(on hdfs)></pre> <p>7.mv: This command is used to move files within hdfs. Lets cut-paste a file <i>myfile.txt</i> from <i>geeks</i> folder to <i>geeks_copied</i>.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -mv <src(on hdfs)> <src(on hdfs)></pre> <p>8.rmr: This command deletes a file from HDFS <i>recursively</i>. It is very useful command when you want to delete a <i>non-empty directory</i>.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -rmr <filename/directoryName></pre> <p>9.du: It will give the size of each file in directory.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -du <dirName></pre> <p>10.dus: This command will give the total size of directory/file.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -dus <dirName></pre> <p>11. Hadoop HDFS version Command Usage:</p> <pre>hadoop version</pre> <p>12. touchz: It creates an empty file.</p> <p>Syntax:</p> <pre>bin/hdfs dfs -touchz <file_path></pre>		
5a)	<p>What is an interface? Establish the Hadoop system interfaces</p> <p>Hadoop is an open-source software framework written in <u>Java</u> along with some shell scripting and <u>C</u> code for performing computation over very large data. Hadoop is utilized for batch/offline processing over the network of so many machines forming a physical cluster.</p> <p>The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications. HDFS employs a NameNode and DataNode</p>	[L3][CO2]	[6M]

architecture to implement a distributed file system that provides high-performance access to data across highly scalable Hadoop clusters.

Hadoop is capable of running various file systems and HDFS is just one single implementation that out of all those file systems. The Hadoop has a variety of file systems that can be implemented concretely.

The Java abstract class *org.apache.hadoop.fs.FileSystem* represents a file system in Hadoop.

Table 3-1. Hadoop filesystems

Filesystem	URI scheme	Java implementation (all under org.apache.hadoop)	Description
Local	<i>file</i>	fs.LocalFileSystem	A filesystem for a locally connected disk with client-side checksums. Use RawLocalFileSystem for a local filesystem with no checksums. See “ LocalFileSystem ” on page 84.
HDFS	<i>hdfs</i>	hdfs.DistributedFileSystem	Hadoop’s distributed filesystem. HDFS is designed to work efficiently in conjunction with MapReduce.
HFTP	<i>hftp</i>	hdfs.HftpFileSystem	A filesystem providing read-only access to HDFS over HTTP. (Despite its name, HFTP has no connection with FTP.) Often used with <i>distcp</i> (see “ Parallel Copying with distcp ” on page 76) to copy data between HDFS clusters running different versions.
HSFTP	<i>hsftp</i>	hdfs.HsftpFileSystem	A filesystem providing read-only access to HDFS over HTTPS. (Again, this has no connection with FTP.)
WebHDFS	<i>webhdfs</i>	hdfs.web.WebHdfsFileSystem	A filesystem providing secure read-write access to HDFS over HTTP. WebHDFS is intended as a replacement for HFTP and HSFTP.

b) Generalize about Hadoop Archives and its Limitations

[L2][CO2]

[6M]

Hadoop archive is a facility which packs up small files into one compact HDFS block to avoid memory wastage of name node. Name node stores the metadata information of the the HDFS data. So, say 1GB file is broken in 1000 pieces then Namenode will have to store metadata about all those 1000 small files. In that manner, Namenode memory will be wasted in storing and managing a lot of data.

HAR is created from a collection of files and the archiving tool will run a MapReduce job. These MapReduce jobs process the input files in parallel to create an archive file.

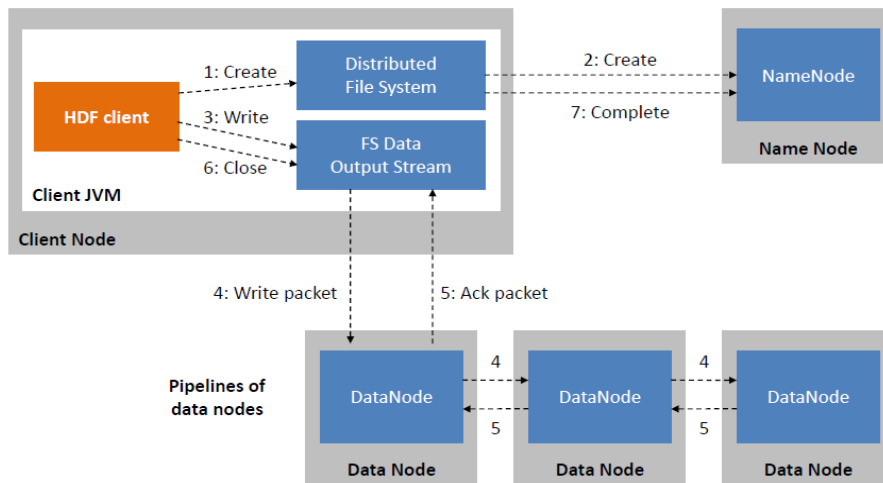
The problem with small files

- Hadoop works best with big files and small files are handled inefficiently in HDFS.
- As we know, Namenode holds the metadata information in memory for all the files stored in HDFS.
- Let’s say we have a file in HDFS which is 1 GB in size and the Namenode will store metadata information of the file – like file name, creator, created time stamp, blocks, permissions etc.
- Now assume we decide to split this 1 GB file into 1000 pieces and store all 1000 “small” files in HDFS.
- Now Namenode has to store metadata information of 1000 small files in memory.
- This is not efficient – first it takes up a lot of memory and second soon Namenode will become a bottleneck as it is trying to manage a lot of data.

HAR command

```
hadoop archive -archiveName myhar.har /input/location /output/location
```

	Limitations of HAR Files: <ol style="list-style-type: none"> 1) Creation of HAR files will create a copy of the original files. So, we need as much disk space as size of original files which we are archiving. We can delete the original files after creation of archive to release some disk space. 2) Once an archive is created, to add or remove files from/to archive we need to re-create the archive. 3) HAR file will require lots of map tasks which are inefficient. 		
6	Infer File read and File write operations in HDFS	[L2][CO2]	[12M]
	<p>Big data is nothing but a collection of data sets that are large, complex, and which are difficult to store and process using available data management tools or traditional data processing applications. Hadoop is a framework (open source) for writing, running, storing, and processing large datasets in a parallel and distributed manner. It is a solution that is used to overcome the challenges faced by big data.</p> <p>File Read in HDFS</p> <p>Let's get an idea of how data flows between the client interacting with HDFS, the name node, and the data nodes with the help of a diagram.</p> <pre> graph LR subgraph Client_Node [Client Node] HDF_client[HDF client] subgraph Client_JVM [Client JVM] DFS[Distributed File System] FSDIS[FS Data Input Stream] end end subgraph Name_Node [Name Node] NameNode[NameNode] end subgraph Data_Nodes [Data Nodes] DN1[DataNode] DN2[DataNode] DN3[DataNode] end HDF_client -.-> 1: Open DFS DFS -.-> 2: Get block locations NameNode NameNode -.-> 3: Read FSDIS FSDIS -.-> 4: Read DN1 FSDIS -.-> 5: Read DN2 FSDIS -.-> 6: Close HDF_client </pre> <p>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).</p> <p>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 FSDataInputStream to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the data node and name node I/O.</p> <p>Step 3: The client then calls read() on the stream. DFSInputStream, 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.</p> <p>Step 4: Data is streamed from the data node back to the client, which calls read() repeatedly on the stream.</p> <p>Step 5: When the end of the block is reached, DFSInputStream 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 DFSInputStream 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.</p> <p>Step 6: When the client has finished reading the file, a function is called, close() on the FSDataInputStream.</p>		

File Write in HDFS:

HDFS follows the Write once Read many times model. In HDFS we cannot edit the files which are already stored in HDFS, but we can append data by reopening the files.

Step 1: The client creates the file by calling `create()` on `DistributedFileSystem(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. `IOException`. The DFS returns an `FSDataOutputStream` for the client to start out writing data to.

Step 3: Because the client writes data, the `DFSOutputStream` splits it into packets, which it writes to an indoor queue called the info queue. The data queue is consumed by the `DataStreamer`, 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 `DataStreamer` 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 `DFSOutputStream` 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.

7a) Discuss about the data ingest operation using Sqoop and Flume

[L2][CO2]

[6M]

Data ingestion is the process of importing large, assorted data files from multiple sources into a single, cloud-based storage medium a data warehouse, data mart or database where it can be accessed and analyzed.

Sqoop

Apache Sqoop(which is a portmanteau for "sql-to-hadoop") is an open source tool that allows users to extract data from a structured data store into Hadoop for further processing.Sqoop can automatically create Hive tables from imported data from a RDBMS (Relational Database Management System) table.

How Sqoop works

Sqoop is an abstraction for MapReduce, meaning it takes a command, such as a request to import a table from an RDBMS into HDFS, and implements this using a MapReduce processing routine. Specifically, Sqoop implements a **Map-only** MapReduce process.

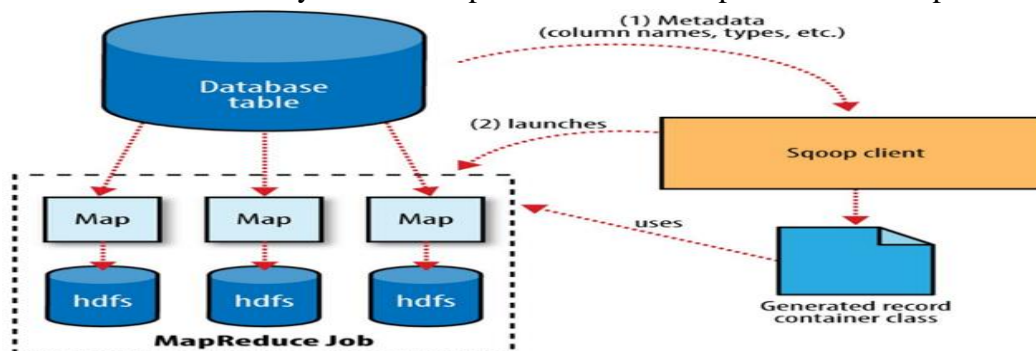
Sqoop performs the following steps to complete an import operation:

1. Connect to the database system using JDBC or a customer connector.

Examine the table to be imported.

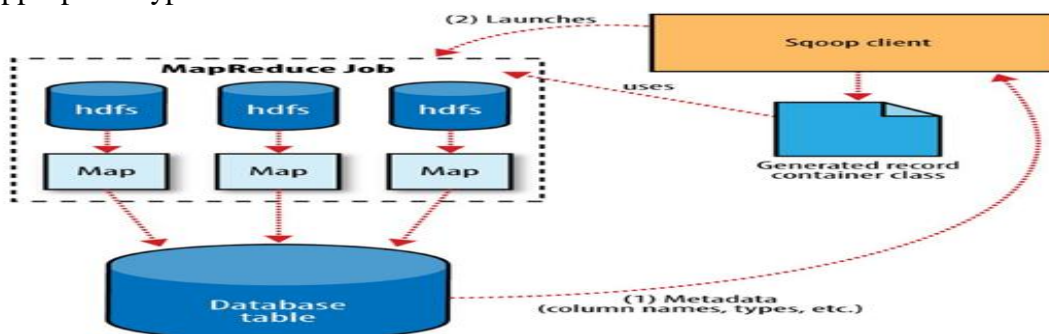
Create a Java class to represent the structure (schema) for the specified table. This class can then be reused for future import operations.

Execute a Map-only MapReduce job with a specified number of tasks (mappers) to connect to the database system and import data from the specified table in parallel.



The Sqoop performs exports is very similar in nature to how Sqoop performs imports. Before performing the export, Sqoop picks a strategy based on the database connect string.

Sqoop then generates a Java class based on the target table definition. This generated class has the ability to parse records from text files and insert values of the appropriate types into a table



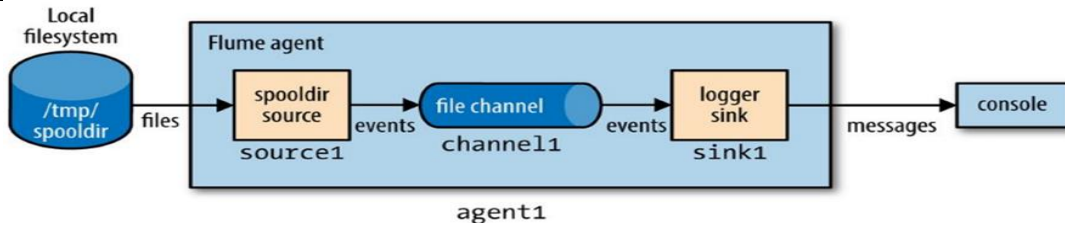
Flume:

Apache Flume is a Hadoop ecosystem project originally developed by Cloudera designed to capture, transform, and ingest data into HDFS using one or more agents. Apache Flume is an ideal fit for streams of data that we would like to aggregate, store, and analyze using Hadoop.

Flume is designed for high-volume ingestion into Hadoop of event-based data.

The initial use case was based upon capturing log files, or web logs, from a source system like a web server, and routing these files and their messages into HDFS as they are generated.

Flume implements a transactional architecture for added reliability. This enables rollback and retry operations if required.



b) Differentiate the compression and serialization operation in Hadoop I/O.

[L4][CO2]

[6M]

Compression In the Hadoop framework, where large data sets are stored and processed, you will need storage for large files. These files are divided into blocks and those blocks are stored in different nodes across the cluster so lots of I/O and network data transfer is also involved. Using data compression in Hadoop you can compress files at various steps, at all of these steps it will help to reduce storage and quantity of data transferred.

File compression brings two major benefits:

1. It reduces the space needed to store files, and it speeds up data transfer across the network.
2. There are many different compression formats, tools and algorithms, each with different characteristics.

Table 4-1. A summary of compression formats

Compression format	Tool	Algorithm	Filename extension	Splittable
DEFLATE ^a	N/A	DEFLATE	.deflate	No
gzip	gzip	DEFLATE	.gz	No
bzip2	bzip2	bzip2	.bz2	Yes
LZO	lzop	LZO	.lzo	No ^b
Snappy	N/A	Snappy	.snappy	No

- The different tools have very different compression characteristics.
- Gzip is a general purpose compressor, and sits in the middle of the space/time trade-off.
- Bzip2 compresses more effectively than Gzip, but is slower.
- Bzip2's decompression speed is faster than its compression speed, but it is still slower than the other formats.
- LZO and Snappy, on the other hand, both optimize for speed and are around an order of magnitude faster than gzip, but compress less effectively.
- Snappy is also significantly faster than LZO for decompression.

Serialization

- *Serialization* is the process of turning structured objects into a byte stream for transmission over a network.
- *Deserialization* is the reverse process of turning a byte stream back into a series of structured objects.

There are two ways to serialize Hadoop data: One is provided by Hadoop's native library, that is the Writable classes, we can use it. And, the other one is Sequence Files which store the data in binary format, we can also use this one.

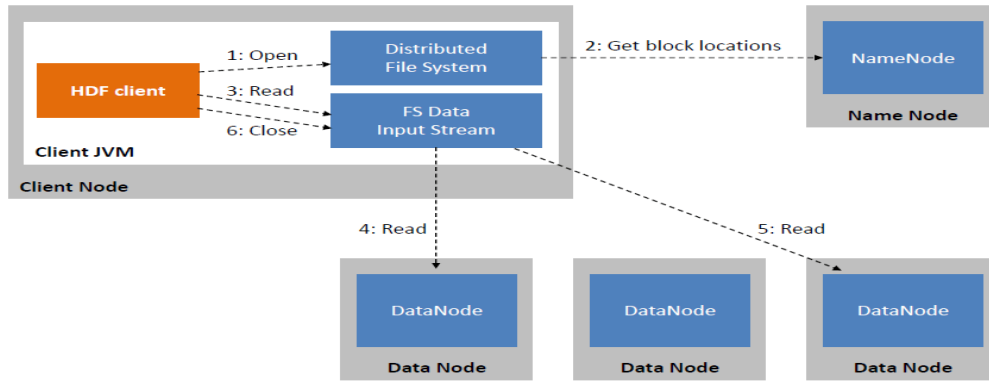
- Hadoop uses its own serialization format, *Writable*s

The Writable Interface

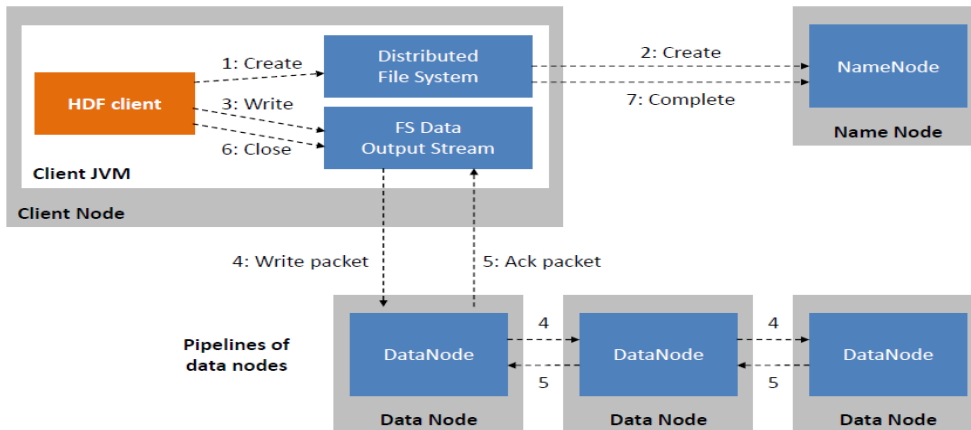
The Writable interface defines two methods:

	<p>one for writing its state to a DataOutput binary stream, and one for reading its state from a DataInput binary stream.</p> <pre> package org.apache.hadoop.io; import java.io.DataOutput; import java.io.DataInput; import java.io.IOException; public interface Writable { void write(DataOutput out) throws IOException; void readFields(DataInput in) throws IOException; } </pre>		
8	<p>Elaborate the AVRO file format with diagram.</p>	[L6][CO2]	[12M]
	<p>What is Avro?</p> <p>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 write operations. Avro serializes the data which has a built-in schema. Avro uses JSON format to declare the data structures. Presently, it supports languages such as Java, C, C++, C#, Python, and Ruby.</p> <p>Avro Schemas</p> <p>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.</p> <p>Avro schemas are defined with JSON that simplifies its implementation in languages with JSON libraries.</p> <p>Like Avro, there are other serialization mechanisms in Hadoop such as Sequence Files, Protocol Buffers, and Thrift.</p> <p>File format:</p> <p>A file format is a standard way that information is encoded for storage in a computer file. File format can have some significant benefits:</p> <ol style="list-style-type: none"> 1. Faster read times 2. Faster write times 3. Splittable files 4. Schema evolution support 5. Advanced compression support <div data-bbox="156 1765 1212 2123"> <p>AVRO File:</p> <p>Header: 4 bytes: ASCII 'O','b','j', followed by File Metadata*: Includes avro.schema and avro.codec*, 16-byte sync marker</p> <p>Block 1: Count of objects in block, Size (bytes) of the serialized objects in block, Serialized objects compressed by specified codec, 16-byte sync marker</p> <p>* File metadata follows: {"type": "map", "values": "bytes"}</p> </div>		

	<p><u>Features of Avro</u></p> <ul style="list-style-type: none"> • Avro is a language-neutral data serialization system. • It can be processed by many languages (currently C, C++, C#, Java, Python, and Ruby). • Avro creates binary structured format that is both compressible and splittable. Hence it can be efficiently used as the input to Hadoop MapReduce jobs. • 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. • Avro schemas defined in JSON, facilitate implementation in the languages that already have JSON libraries. • Avro creates a self-describing file named <i>Avro Data File</i>, in which it stores data along with its schema in the metadata section. • Avro is also used in Remote Procedure Calls (RPCs). During RPC, client and server exchange schemas in the connection handshake. <p><u>General Working of Avro</u></p> <p>To use Avro, you need to follow the given workflow –</p> <ul style="list-style-type: none"> • Step 1 – Create schemas. Here you need to design Avro schema according to your data. • Step 2 – Read the schemas into your program. It is done in two ways – <ul style="list-style-type: none"> ○ By Generating a Class Corresponding to Schema – Compile the schema using Avro. This generates a class file corresponding to the schema ○ By Using Parsers Library – You can directly read the schema using parsers library. • Step 3 – Serialize the data using the serialization API provided for Avro, which is found in the package org.apache.avro.specific. • Step 4 – Deserialize the data using deserialization API provided for Avro, which is found in the package org.apache.avro.specific 		
9a)	Describe the dataflow process in Hadoop Distributed file System	[L3][CO2]	[6M]
	<p>A basic data flow of the Hadoop system can be divided into four phases:</p> <ol style="list-style-type: none"> 1. Capture Big Data : The sources can be extensive lists that are structured, semi-structured, and unstructured, some streaming, real-time data sources, sensors, devices, machine-captured data, and many other sources. For data capturing and storage, we have different data integrators such as, Flume, Sqoop, Storm, and so on in the Hadoop ecosystem, depending on the type of data. 2. Process and Structure: We will be cleansing, filtering, and transforming the data by using a MapReduce-based framework or some other frameworks which can perform distributed programming in the Hadoop ecosystem. The frameworks available currently are MapReduce, Hive, Pig, Spark and so on. 3. Distribute Results: The processed data can be used by the BI and analytics system or the big data analytics system for performing analysis or visualization. 4. Feedback and Retain: The data analyzed can be fed back to Hadoop and used for improvements. <p><u>File Read in HDFS</u></p> <p>Let's get an idea of how data flows between the client interacting with HDFS, the name node, and the data nodes with the help of a diagram.</p>		



File Write in HDFS



b) Analyze the features of Apache Hadoop

[L4][CO2]

[6M]

1) Distributed Processing & Storage

The framework itself provides great flexibility and manages the distributed processing and distributed storage by itself leaving only the custom logic to be built for data processing by users. That made Apache Hadoop different from other distributed systems and become highly popular so quickly.

2) Highly Available & Fault Tolerant

Hadoop is highly available and provides fault tolerance both in terms of data availability and distributed processing. The data is stored in HDFS where data automatically gets replicated at two other locations. So, even if one or two of the systems collapse, the file is still available on the third system at least. This brings a high level of fault tolerance. The highly distributed MapReduce batch processing engine provides high availability in terms of processing failure due to hardware / machine failure.

3) Highly & Easily Scalable

Both vertical and horizontal scaling is possible. The differentiator however is the horizontal scaling where new nodes can be easily added in the system on the fly as and when data volume of processing needs grow without altering anything in the existing systems or programs.

4) Data Reliability

The data is stored reliably due to data replication in cluster where multiple copies are maintained on different nodes. The framework itself provides mechanisms to ensure data reliability by Block Scanner & Volume Scanner, Directory Scanner and Disk Checker. In case of data corruption and hardware failures the data integrity and availability it maintained.

5) Robust Ecosystem

Hadoop has a very robust ecosystem that is well suited to meet the analytical needs

	<p>of developers and small to large organizations. Hadoop Ecosystem comes with a suite of tools and technologies making it a very much suitable to deliver to a variety of data processing needs. Just to name a few, Hadoop ecosystem comes with projects such as MapReduce, Yarn, Hive, HBase, Zookeeper, Pig, Flume, Avro etc. and many new tools and technologies are being added to the ecosystem as the market grows.</p> <p>6) Very Cost effective</p> <p>Hadoop generates cost benefits by bringing massively parallel computing to commodity servers, resulting in a substantial reduction in the cost per terabyte of storage, which in turn makes it reasonable to model all your data.</p> <p>7) Open Source</p> <p>No worries for licensing cost with very strong open source community support. Can be easily accommodated and build for custom requirement.</p>		
10a)	Justify File Based Data structures.	[L5][CO2]	[6M]
	<p><u>1.Sequence file:</u></p> <ol style="list-style-type: none"> 1. sequence file files are <key, value>flat files (Flat file) designed by Hadoop to store binary forms of pairs. 2.All the files packaged into the Sequence file class can be efficiently stored and processed small files . 3. The key and value in Sequence file can be any type writable or a custom writable type. <p>Benefits of the Sequence file format:</p> <ul style="list-style-type: none"> ● Supports data compression based on records (record) or blocks (block). ● Simple to modify <p>Disadvantages of the Sequence file format:</p> <ul style="list-style-type: none"> ● The downside is the need for a merge file, and the merged file will be inconvenient to view, because it is a binary file. <p><u>2.Map File:</u></p> <p>Mapfile is the sequence file of the sorted index and can be found based on key. Unlike Sequence file, map file key must implement Writable comparable interface, that is, the key value is comparable, and value is the writable type. You can use the Map file.fix () method to reconstruct the index and convert the Sequence file to map file. It has two static member variables:</p> <ul style="list-style-type: none"> ● static final String INDEX_FILE_NAME ● static final String DATA_FILE_NAME <p>Map file does not record all records into an index, which by default stores an index map for every 128 records. The retrieval efficiency of map file is efficient relative to sequence file, and the disadvantage is that it consumes a portion of memory to store index data.</p> <p><u>3.Text files:</u></p> <p>A text file is the most basic and a human-readable file. It can be read or written in any programming language and is mostly delimited by comma or tab. The text file format consumes more space when a numeric value needs to be stored as a string. It is also difficult to represent binary data such as an image.</p> <p><u>4.Parquet File Format:</u></p> <p>Parquet is a columnar format developed by Cloudera and Twitter. It is supported in Spark, MapReduce, Hive, Pig, Impala, Crunch, and so on. Like Avro, schema metadata is embedded in the file.Parquet file format uses advanced optimizations described in Google's Dremel paper.</p>		

b)	Differentiate Sqoop and Flume?			[L2][CO2]	[6M]
	Parameter	Sqoop	Flume		
	Basic nature	It is basically designed to work with different types of RDBMS, which have JDBC connectivity.	It is basically designed for transferring streaming data such as log files from different sources to the Hadoop ecosystem.		
	Driven Events	Sqoop data load is not driven by events.	Flume is completely event-driven.		
	Data Flow	Sqoop is used for transferring data parallelly from relational databases to Hadoop.	It is used for collecting and aggregating data from different sources because of its distributed nature.		
	Architecture	Sqoop follows connector-based architecture.	Flume follows agent-based architecture.		
	Performance	Apache Sqoop reduces the processing loads and excessive storage by transferring them to the other systems.	Apache Flume is highly robust, fault-tolerant, and has a tunable reliability mechanism.		
	Release History	The first version of Sqoop was released in March 2012.	The first version of Flume was released in June 2012.		
	Link to HDFS	Hadoop Distributed File System is the destination while importing data using Sqoop.	In Flume, data flows to Hadoop Distributed File System through multiple channels.		
	Companies	Companies like Mozilla, Capillary technologies, etc use Apache Flume.	Companies like Mozilla, Capillary technologies, etc use Apache Flume.		

UNIT –III

MAP REDUCE

1	Examine the Anatomy of a MapReduce Job Run.	[L4][CO3]	[12M]
	<p>Anatomy of a MapReduce Job Run:</p> <p>You can run a MapReduce job with a single method call: <code>submit()</code> on a Job object (you can also call <code>waitForCompletion()</code>, which submits the job if it hasn't been submitted already, then waits for it to finish).[51] This method call conceals a great deal of processing behind the scenes. This section uncovers the steps Hadoop takes to run a job.</p> <ul style="list-style-type: none"> • The client, which submits the MapReduce job. • The YARN resource manager, which coordinates the allocation of compute resources on the cluster. • The YARN node managers, which launch and monitor the compute containers on machines in the cluster. • The MapReduce application master, which coordinates the tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager and managed by the node managers. • The distributed filesystem (normally HDFS, covered in Chapter 3), which is used for sharing job files between the other entities. <p>To create an internal JobSubmitter instance, use the submit() which further calls submitJobInternal() on it. Having submitted the job, waitForCompletion() polls the job's progress after submitting the job once per second. If the reports have changed since the last report, it further reports the progress to the console. The job counters are displayed when the job</p>		

	<p>completes successfully. Else the error (that caused the job to fail) is logged to the console.</p> <p>Processes implemented by JobSubmitter for submitting the Job :</p> <ul style="list-style-type: none"> • The resource manager asks for a new application ID that is used for MapReduce Job ID. • Output specification of the job is checked. For e.g. an error is thrown to the MapReduce program or the job is not submitted or the output directory already exists or it has not been specified. • If the splits cannot be computed, it computes the input splits for the job. This can be due to the job is not submitted and an error is thrown to the MapReduce program. • Resources needed to run the job are copied – it includes the job JAR file, and the computed input splits, to the shared filesystem in a directory named after the job ID and the configuration file. • It copies job JAR with a high replication factor, which is controlled by mapreduce.client.submit.file.replication property. AS there are a number of copies across the cluster for the node managers to access. • By calling submitApplication(), submits the job to the resource manager. 		
2	Sketch neatly and Explain MapReduce Architecture in detail.	[L3][CO3]	[12M]
	<p>MapReduce is a part of the Apache Hadoop ecosystem. It is a software framework that processes huge amounts of data across computer clusters. It does parallel processing and stores data in distributed form.</p> <p>How MapReduce Works?</p> <p>The entire process is divided into four steps of execution which are splitting, mapping, shuffling and reducing.</p> <div data-bbox="266 1180 1136 1547" data-label="Diagram"> <p style="text-align: center;">Map Reduce Architecture</p> <pre> graph TD Client[Client] --> Job[Job] Job --> Master[hadoop MapReduce Master] Master --> JobParts1[Job parts] Master --> JobParts2[Job parts] JobParts1 --> Map1([Map]) JobParts2 --> Map2([Map]) JobParts1 --> Reduce1([Reduce]) JobParts2 --> Reduce2([Reduce]) Map1 --> Reduce1 Map1 --> Reduce2 Map2 --> Reduce1 Map2 --> Reduce2 ID[Input Data] --> Map1 ID --> Map2 Map1 --> OD[Output Data] Map2 --> OD Map1 --> Reduce1 Map1 --> Reduce2 Map2 --> Reduce1 Map2 --> Reduce2 </pre> </div> <p>Components of MapReduce Architecture</p> <p>Client: The MapReduce client is the one who brings the Job to the MapReduce for processing. There can be multiple clients available that continuously send jobs for processing to the Hadoop MapReduce Manager.</p> <p>Job: The MapReduce Job is the actual work that the client wanted to do which is comprised of so many smaller tasks that the client wants to process or execute.</p> <p>Hadoop MapReduce Master: It divides the particular job into subsequent job-parts.</p> <p>Job-Parts: The task or sub-jobs that are obtained after dividing the main job. The result of all the job-parts combined to produce the final output.</p> <p>Input Data: The data set that is fed to the MapReduce for processing.</p> <p>Output Data: The final result is obtained after the processing.</p> <p>The MapReduce task is mainly divided into 2 phases i.e. Map phase and</p>		

	<p>Reduce phase.</p> <p>Map: As the name suggests its main use is to map the input data in key-value pairs. The input to the map may be a key-value pair where the key can be the id of some kind of address and value is the actual value that it keeps. The Map() function will be executed in its memory repository on each of these input key-value pairs and generates the intermediate key-value pair which works as input for the Reducer or Reduce() function.</p> <p>Reduce: The intermediate key-value pairs that work as input for Reducer are shuffled and sort and send to the Reduce() function. Reducer aggregate or group the data based on its key-value pair as per the reducer algorithm written by the developer.</p> <p>How Job tracker and the task tracker deal with MapReduce:</p> <p>Job Tracker: The work of Job tracker is to manage all the resources and all the jobs across the cluster and also to schedule each map on the Task Tracker running on the same data node since there can be hundreds of data nodes available in the cluster.</p> <p>Task Tracker: The Task Tracker can be considered as the actual slaves that are working on the instruction given by the Job Tracker. This Task Tracker is deployed on each of the nodes available in the cluster that executes the Map and Reduce task as instructed by Job Tracker.</p> <p>Benefits of MapReduce:</p> <ul style="list-style-type: none"> • Faster Processing: MapReduce processes huge unstructured data in a short span of time as it has faster processing. • Scalable: MapReduce allows us to run applications on many different data nodes. • Reliable: In case of task failure, the job tracker in MapReduce reschedules the task to a different task tracker. This makes MapReduce more reliable and fault tolerant. • Parallel Processing: MapReduce does parallel processing by processing multiple job-parts of the same datasets parallelly. 		
3	Explain in detail about Hadoop YARN Architecture	[L2][CO3]	[12M]
	<p>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.</p> <p><u>YARN Features:</u></p> <p>Scalability: The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.</p> <p>Compatibility: YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well.</p> <p>Cluster Utilization: Since YARN supports Dynamic utilization of cluster in Hadoop, which enables optimized Cluster Utilization.</p> <p>Multi-tenancy: It allows multiple engine access thus giving organizations a benefit of multi-tenancy.</p>		



The main components of YARN architecture include:

Client: It submits map-reduce jobs.

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.

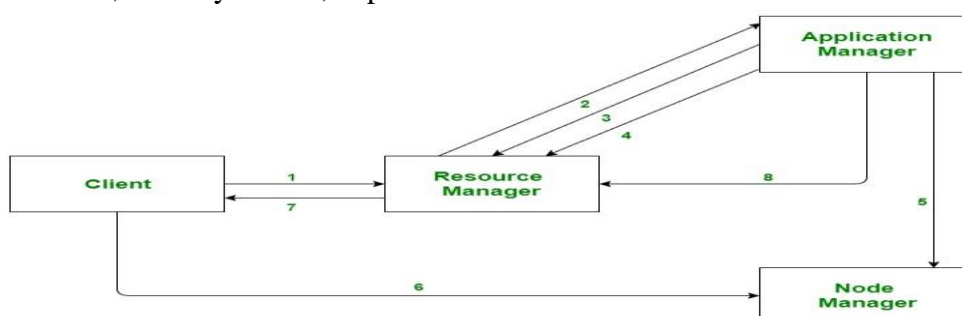
Scheduler: It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails.

Application manager: It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.

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 Resource Manager.


Application Master: An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application

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.



1. Client submits an application
2. The Resource Manager allocates a container to start the Application Manager
3. The Application Manager registers itself with the Resource Manager
4. The Application Manager negotiates containers from the Resource Manager
5. The Application Manager notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application's status
8. Once the processing is complete, the Application Manager un-registers with the Resource Manager

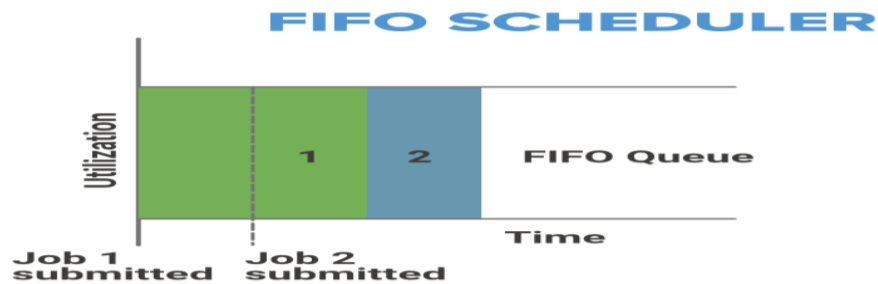
4a)	Discuss different types of failures in Classic MapReduce.	[L2][CO3]	[6M]
	<p>Failures in Classic MapReduce</p> <p>In the MapReduce 1 runtime there are three failure modes to consider: failure of the running task, failure of the tastracker, and failure of the jobtracker. Let's look at each in turn.</p> <p><u>1.Task Failure:</u></p> <p>In Hadoop, task failure is similar to an employee making a mistake while doing a task. Consider you are working on a large project that has been broken down into smaller jobs and assigned to different employees in your team. If one of the team members fails to do their task correctly, the entire project may be compromised.</p> <p>How to Overcome Task Failure</p> <ul style="list-style-type: none"> • Increase memory allocation: Assign extra memory to jobs to ensure they have the resources to process the data. • Implement fault tolerance mechanisms: Using data replication and checkpointing techniques to defend against disc failures and retrieve lost data. • Regularly update software and hardware: Keep the Hadoop framework and supporting hardware up to date to fix bugs, errors. <p><u>2.TaskTracker Failure</u></p> <p>A TaskTracker in Hadoop is similar to an employee responsible for executing certain tasks in a large project. If a TaskTracker fails, it signifies a problem occurred while an employee worked on their assignment</p> <p>How to Overcome TaskTracker Failure</p> <ul style="list-style-type: none"> • Update software and hardware on a regular basis: Keep the Hadoop framework and associated hardware up to date to correct bugs, errors, and performance issues that might lead to task failures. • Upgrade or replace hardware: If TaskTracker's hardware is outdated or insufficiently powerful, try upgrading or replacing it with more powerful components. • Restart or reinstall the program: If the TaskTracker software is causing problems, a simple restart or reinstall may be all that is required. <p><u>3.JobTracker Failure:</u></p> <p>A JobTracker in Hadoop is similar to a supervisor or manager that oversees the entire project and assigns tasks to TaskTrackers (employees). If a JobTracker fails, it signifies the supervisor is experiencing a problem or has stopped working properly.</p> <p>How to Overcome JobTracker Failure</p> <ul style="list-style-type: none"> • Avoiding Database Connectivity: To avoid database connectivity failures in the JobTracker, ensure optimized database configuration, robust network connections • To overcome security-related problems: implement strong authentication and authorization, enable SSL/TLS for secure communication, keep software updated with security patches, standards. 		
4b)	List out the different types of failures in Yet Another Resource Negotiator.	[L1][CO3]	[6M]
	<p>Task Failure</p> <p>1. The most common occurrence of this failure is when user code in the map or reduce task throws a runtime exception.</p>		

	<p>2. For Streaming tasks, if the Streaming process exits with a nonzero exit code, it is marked as failed.</p> <p>3. Sudden exit of the task JVM—perhaps there is a JVM bug that causes the JVM to exit for a particular set of circumstances exposed by the Map Reduce user code.</p> <p>Application Master Failures</p> <ul style="list-style-type: none"> • An application master sends periodic heartbeats to the resource manager, • In the event of application master failure, then resource manager will detect the failure and start a new instance of the master running in a new container (managed by a node manager). • In the case of the MapReduce application master, it will use the job history to recover the state of the tasks that were already run by the (failed) application so they don't have to be rerun. <p>Node Manager failure</p> <ul style="list-style-type: none"> • If a node manager fails by crashing or running very slowly, it will stop sending heartbeats to the resource manager (or send them very infrequently). • The resource manager will notice a node manager that has stopped sending heartbeats if it hasn't received one for 10 minutes. • Node managers may be <i>blacklisted</i> if the number of failures for the application is high, even if the node manager itself has not failed. <p>Resource Manager failure</p> <ul style="list-style-type: none"> • Failure of the resource manager is serious, because without it, neither jobs nor task containers can be launched. • The resource manager is a single point of failure, all running jobs fail can't be recovered. • When the new resource manager starts, it reads the application information from the state store, then restarts the application masters for all the applications running on the cluster. 		
5a)	Examine the different types of Job Scheduling process in Map Reduce.	[L3][CO3]	[6M]
	<p>There are mainly 3 types of Schedulers in Hadoop:</p> <ol style="list-style-type: none"> 1. FIFO (First In First Out) Scheduler. 2. Capacity Scheduler. 3. Fair Scheduler. <p>A Job queue is nothing but the collection of various tasks that we have received from our various clients. The tasks are available in the queue and we need to schedule this task on the basis of our requirements.</p> <p style="text-align: center;">JOB QUEUE</p>  <p>1. FIFO Scheduler</p> <p>As the name suggests FIFO i.e. First In First Out, so the tasks or application that comes first will be served first. This is the default Scheduler we use in Hadoop. The tasks are placed in a queue and the tasks are performed in their submission order. In this method, once the job is scheduled, no intervention is allowed.</p> <p>Advantage:</p> <ul style="list-style-type: none"> • No need for configuration • First Come First Serve 		

- simple to execute

Disadvantage:

- Priority of task doesn't matter, so high priority jobs need to wait
- Not suitable for shared cluster

**2. Capacity Scheduler**

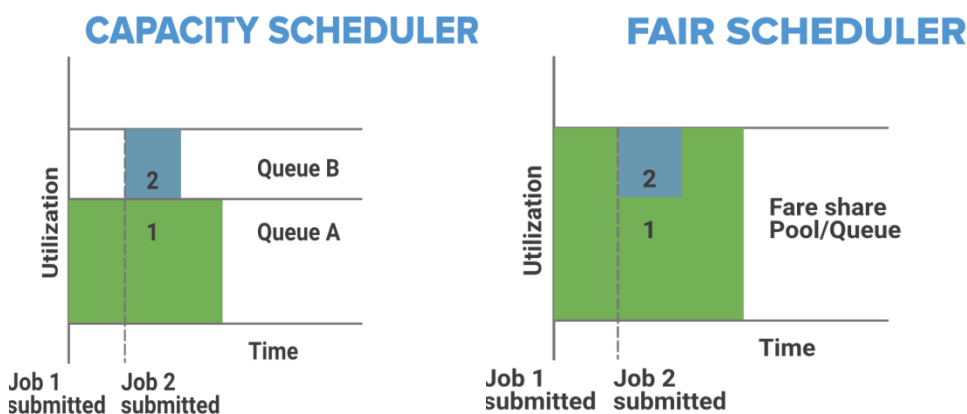
In Capacity Scheduler we have multiple job queues for scheduling our tasks. The Capacity Scheduler allows multiple occupants to share a large size Hadoop cluster. In Capacity Scheduler corresponding for each job queue, we provide some slots or cluster resources for performing job operation. Each job queue has its own slots to perform its task.

Advantage:

- Best for working with Multiple clients or priority jobs in a Hadoop cluster
- Maximizes throughput in the Hadoop cluster

Disadvantage:

- More complex
- Not easy to configure for everyone

**3. Fair Scheduler**

The Fair Scheduler is very much similar to that of the capacity scheduler. The priority of the job is kept in consideration. With the help of Fair Scheduler, the YARN applications can share the resources in the large Hadoop Cluster and these resources are maintained dynamically so no need for prior capacity. The resources are distributed in such a manner that all applications within a cluster get an equal amount of time. Fair Scheduler takes Scheduling decisions on the basis of memory.

Advantages:

- Resources assigned to each application depend upon its priority.
- it can limit the concurrent running task in a particular pool or queue.

Disadvantages: The configuration is required.

5b) Identify the Significance of YARN over MapReduce.

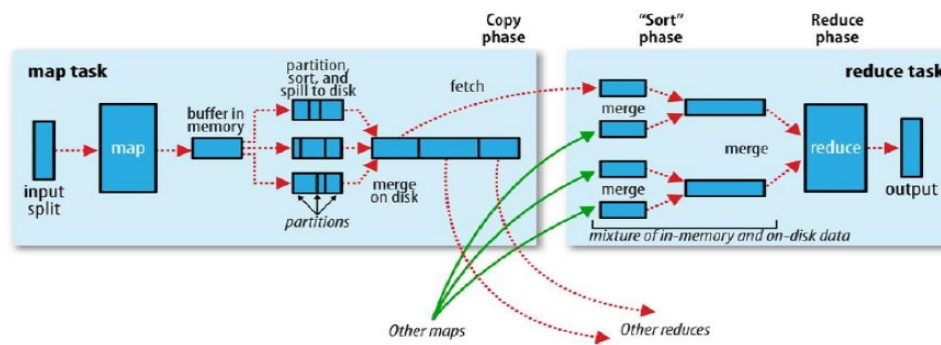
[L3][CO3]

[6M]

Features of YARN

Better utilization of resources – The YARN framework does not have any fixed slots for tasks. It provides a central resource manager which allows you to

	<p>share multiple applications through a common resource.</p> <p>Running non-MapReduce applications – In YARN, the scheduling and resource management capabilities are separated from the data processing component. This allows Hadoop to run varied types of applications which do not conform to the programming of the Hadoop framework. Hadoop clusters are now capable of running independent interactive queries and performing better real-time analysis.</p> <p>Backward compatibility – YARN comes as a <u>backward-compatible</u> framework, which means any existing job of MapReduce can be executed in Hadoop 2.0.</p> <p>JobTracker no longer exists – The two major roles of the JobTracker were resource management and job scheduling. With the introduction of the YARN framework these are now segregated into two separate components, namely: NodeManager ResourceManager</p> <p>Features of MapReduce</p> <ul style="list-style-type: none"> • Flexibility: YARN offers flexibility to run various types of distributed processing systems such as Apache Spark, Apache Flink, Apache Storm, and others. It allows multiple processing engines to run simultaneously on a single Hadoop cluster. • Resource Management: YARN provides an efficient way of managing resources in the Hadoop cluster. It allows administrators to allocate and monitor the resources required by each application in a cluster, such as CPU, memory, and disk space. • Scalability: YARN is designed to be highly scalable and can handle thousands of nodes in a cluster. It can scale up or down based on the requirements of the applications running on the cluster. • Improved Performance: YARN offers better performance by providing a centralized resource management system. It ensures that the resources are optimally utilized, and applications are efficiently scheduled on the available resources. • Security: YARN provides robust security features such as Kerberos authentication, Secure Shell (SSH) access, and secure data transmission. It ensures that the data stored and processed on the Hadoop cluster is secure. 		
6	Illustrate Shuffle and Sort operations in MapReduce.	[L5][CO4]	[12M]
	<p>Shuffle phase in Hadoop transfers the map output from Mapper to a Reducer in MapReduce. Sort phase in MapReduce covers the merging and sorting of map outputs. Data from the mapper are grouped by the key, split among reducers and sorted by the key. Every reducer obtains all values associated with the same key.</p> <p>In this lesson, we will learn completely about MapReduce Shuffling and Sorting. Here we will offer you a detailed description of the Hadoop Shuffling and Sorting phase. Initially, we will discuss what is MapReduce Shuffling, next with MapReduce Sorting, then we will discuss MapReduce the secondary sorting phase in detail.</p>		



What is MapReduce Shuffling and Sorting?

Shuffling is the process by which it transfers the mapper's intermediate output to the reducer. Reducer gets one or more keys and associated values based on reducers. The intermediated key – value generated by the mapper is sorted automatically by key. In Sort phase merging and sorting of the map, the output takes place.

Shuffling and Sorting in Hadoop occur simultaneously.

Shuffling in MapReduce The process of moving data from the mappers to reducers is shuffling. Shuffling is also the process by which the system performs the sort. Then it moves the map output to the reducer as input. This is the reason the shuffle phase is required for the reducers. Else, they would not have any input (or input from every mapper). Meanwhile, shuffling can begin even before the map phase has finished. Therefore this saves some time and completes the tasks in lesser time.

Sorting in MapReduce

MapReduce Framework automatically sorts the keys generated by the mapper. Therefore, before starting of reducer, all intermediate key-value pairs get sorted by key and not by value. It does not sort values transferred to each reducer. They can be in any order.

Sorting in a MapReduce job helps reducer to easily differentiate when a new reduce task should start. This saves time for the reducer. Reducer in MapReduce begins a new reduce task when the next key in the sorted input data is different from the earlier. Each reduce task takes key-value pairs as input and creates a key-value pair as output.

The crucial thing to note is that shuffling and sorting in Hadoop MapReduce is will not take place at all if you specify zero reducers (setNumReduceTasks(0)). If the reducer is zero, then the MapReduce job stops at the map phase. And the map phase does not comprise any kind of sorting (even the map phase is faster).

Secondary Sorting in MapReduce

If we need to sort reducer values, then we use a secondary sorting technique. This technique allows us to sort the values (in ascending or descending order) transferred to each reducer.

7a)	Summarize Task Execution Environment Properties.	[L2][CO4]	[6M]
	<p>Task Execution & Environment. The TaskTracker executes the Mapper/Reducer task as a child process in a separate jvm. The child-task inherits the environment of the parent TaskTracker. The user can specify additional options to the child-jvm via the mapred.</p> <p>To write a Flink program, you need an execution environment. You can use an existing environment or create a new environment. Based on your requirements, Flink allows you to use an existing Flink environment, create a local environment, or create a remote environment.</p> <p>Use the getExecutionEnvironment() command to accomplish different tasks based on your requirement:</p> <ul style="list-style-type: none"> To execute on a local environment in an IDE, it starts a local execution environment To execute a JAR, the Flink cluster manager executes the program in a distributed manner To create your own local or remote environment, you can use methods such as createLocalEnvironment() and createRemoteEnvironment (string host, int port, string, and .jar files) 		
7b)	Discuss about Speculative Execution and its Properties.	[L2][CO4]	[6M]
	<p>In Hadoop, MapReduce breaks jobs into tasks and these tasks run parallel rather than sequential, thus reduces overall execution time. This model of execution is sensitive to slow tasks (even if they are few in numbers) as they slow down the overall execution of a job.</p> <p>In Hadoop, MapReduce breaks jobs into tasks and these tasks run parallel rather than sequential, thus reduces overall execution time. This model of execution is sensitive to slow tasks (even if they are few in numbers) as they slow down the overall execution of a job.</p> <p>There may be various reasons for the slowdown of tasks, including hardware degradation or software misconfiguration, but it may be difficult to detect causes since the tasks still complete successfully, although more time is taken than the expected time. Hadoop doesn't try to diagnose and fix slow running tasks, instead, it tries to detect them and runs backup tasks for them. This is called speculative execution in Hadoop. These backup tasks are called Speculative tasks in Hadoop.</p>		
8)	Categorize different types of MapReduce input formats.	[L4][CO4]	[12M]
	<p>TextInputFormat – TextInputFormat is the default InputFormat. Each record is a line of input. The key, a LongWritable, is the byte offset within the file of the beginning of the line. The value is the contents of the line, excluding any</p>		

line terminators.

KeyValueTextInputFormat – TextInputFormat's keys, being simply the offsets within the file, are not normally very useful. It is common for each line in a file to be a key-value pair, separated by a delimiter such as a tab character.

For example, this is the kind of output produced by TextOutputFormat, Hadoop's default OutputFormat. To interpret such files correctly, KeyValueTextInputFormat is appropriate. You can specify the separator via the `mapreduce.input.keyvaluelinerecorder.key.value.separator` property. It is a tab character by default.

NLineInputFormat – With TextInputFormat and KeyValueTextInputFormat, each mapper receives a variable number of lines of input. The number depends on the size of the split and the length of the lines. If you want your mappers to receive a fixed number of lines of input, then NLineInputFormat is the InputFormat to use. Like with TextInputFormat, the keys are the byte offsets within the file and the values are the lines themselves. N refers to the number of lines of input that each mapper receives. With N set to 1 (the default), each mapper receives exactly one line of input. The `mapreduce.input.lineinputformat.linespermap` property controls the value of N.

StreamInputFormat – Hadoop comes with a InputFormat for streaming which can be used outside streaming and can be used for processing XML documents. You can use it by setting your input format to StreamInputFormat and setting the `stream.recordreader.class` property to `org.apache.hadoop.streaming.mapreduce.StreamXmlRecordReader`. The reader is configured by setting job configuration properties to tell it the patterns for the start and end tags.

XmlInputFormat courtesy of the Lucene sub-project: Mahout is been recommended in many blogging sites to be working for xml parsing and can be considered for xml reading. Link here [XmlInputFormat.java](#)

SequenceFileInputFormat – Hadoop MapReduce is not restricted to processing textual data. It has support for binary formats, too. Hadoop's sequence file format stores sequences of binary key-value pairs. Sequence files are well suited as a format for MapReduce data because they are splittable (they have sync points so that readers can synchronize with record boundaries from an arbitrary point in the file, such as the start of a split), they support compression as a part of the format, and they can store arbitrary types using a variety of serialization frameworks.

SequenceFileAsTextInputFormat – Sequence File AsTextInputFormat is a variant of SequenceFileInputFormat that converts the sequence file's keys and values to Text objects. The conversion is performed by calling `toString()` on the keys and values. This format makes sequence files suitable input for Streaming

SequenceFileAsBinaryInputFormat – Its is a variant of SequenceFileInputFormat that retrieves the sequence file's keys and values as opaque binary objects. They are encapsulated as BytesWritable objects, and the

	<p>application is free to interpret the underlying byte array as it pleases.</p> <p>FixedLengthInputFormat – FixedLengthInputFormat is for reading fixed-width binary records from a file, when the records are not separated by delimiters. The record size must be set via <code>fixedlengthinputformat.record.length</code>.</p>		
9)	Justify types of output formats in MapReduce.	[L5][CO4]	[12M]
	<p>1. Record Writer in Hadoop MapReduce</p> <p>As we know, Reducer takes Mappers intermediate output as input. Then it runs a reducer function on them to generate output that is again zero or more key-value pairs.</p> <p>So, RecordWriter in MapReduce job execution writes these output key-value pairs from the Reducer phase to output files.</p> <p>Follow TechVidvan on Google & Stay updated with latest technology trends</p> <p>2. Hadoop Output Format</p> <p>From above it is clear that RecordWriter takes output data from Reducer. Then it writes this data to output files. OutputFormat determines the way these output key-value pairs are written in output files by RecordWriter.</p> <p>The OutputFormat and InputFormat functions are similar. OutputFormat instances are used to write to files on the local disk or in HDFS. In MapReduce job execution on the basis of output specification;</p> <ul style="list-style-type: none"> • Hadoop MapReduce job checks that the output directory does not already present. • OutputFormat in MapReduce job provides the RecordWriter implementation to be used to write the output files of the job. Then the output files are stored in a FileSystem. <p>The framework uses FileOutputFormat.setOutputPath() method to set the output directory.</p> <p>Types of OutputFormat in MapReduce</p> <p>There are various types of OutputFormat which are as follows:</p> <p>1. TextOutputFormat</p> <p>The default OutputFormat is TextOutputFormat. It writes (key, value) pairs on individual lines of text files. Its keys and values can be of any type. The reason behind is that TextOutputFormat turns them to string by calling toString() on them.</p> <p>It separates key-value pair by a tab character. By using MapReduce.output.textoutputformat.separator property we can also change it. KeyValueTextOutputFormat is also used for reading these output text files.</p> <p>3. SequenceFileOutputFormat</p> <p>This OutputFormat writes sequences files for its output. SequenceFileInputFormat is also intermediate format use between MapReduce jobs. It serializes arbitrary data types to the file. And the corresponding SequenceFileInputFormat will deserialize the file into the same types. It</p>		

	<p>presents the data to the next mapper in the same manner as it was emitted by the previous reducer. Static methods also control the compression.</p> <p>3. SequenceFileAsBinaryOutputFormat</p> <p>It is another variant of SequenceFileInputFormat. It also writes keys and values to sequence file in binary format.</p> <p>4. MapFileOutputFormat It is another form of FileOutputFormat. It also writes output as map files. The framework adds a key in a MapFile in order. So we need to ensure that reducer emits keys in sorted order.</p> <p>5. MultipleOutputs This format allows writing data to files whose names are derived from the output keys and values.</p> <p>6. LazyOutputFormat In MapReduce job execution, FileOutputFormat sometimes create output files, even if they are empty. LazyOutputFormat is also a wrapper OutputFormat.</p> <p>7. DBOutputFormat It is the OutputFormat for writing to relational databases and HBase. This format also sends the reduce output to a SQL table. It also accepts key-value pairs. In this, the key has a type extending DBWritable.</p>		
10)	<p>Discriminate the below features in MapReduce.</p> <p>a) Counters b) Sorting c) Joins</p>	[L4][CO4]	[12M]
	<p>a) Counters</p> <p>Counters in Hadoop are a useful channel for gathering statistics about the MapReduce job. Like for quality control or for application-level. Counters are also useful for problem diagnosis.</p> <p>A Counter represents Apache Hadoop global counters, defined either by the MapReduce framework. Each counter in MapReduce is named by an “Enum”. It also has a long for the value.</p> <p>Hadoop Counters validate that:</p> <ul style="list-style-type: none"> • It reads and written correct number of bytes. • It has launched and successfully run correct number of tasks or not. • Counters also validate that the amount of CPU and memory consumed is appropriate for our job and cluster nodes or not. <p>b) Sorting</p> <p>The keys generated by the mapper are automatically sorted by MapReduce Framework, i.e. Before starting of reducer, all intermediate key-value pairs in MapReduce that are generated by mapper get sorted by key and not by value. Values passed to each reducer are not sorted; they can be in any order.</p> <p>Sorting in Hadoop helps reducer to easily distinguish when a new reduce task should start. This saves time for the reducer. Reducer starts a new reduce task when the next key in the sorted input data is different than the previous. Each reduce task takes key-value pairs as input and generates key-value pair as output.</p> <p>Note that shuffling and sorting in Hadoop MapReduce is not performed at all if you specify zero reducers (setNumReduceTasks(0)). Then, the MapReduce job stops at the map phase, and the map phase does not include any kind of sorting (so even the map phase is faster).</p>		

	c) Joins MapReduce Joins There are two types of join operations in MapReduce: Map Side Join: As the name implies, the join operation is performed in the map phase itself. Therefore, in the map side join, the mapper performs the join and it is mandatory that the input to each map is partitioned and sorted according to the keys.		
--	--	--	--



**SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY:: PUTTUR
(AUTONOMOUS)**

Siddharth Nagar, Narayanavanam Road – 517583

QUESTION BANK (DESCRIPTIVE)

Subject with Code: BIG DATA (20CS1214)

Course & Branch: B.Tech – CCC

Regulation: R20

Year & Sem: IV-B.Tech & I-Sem

HADOOP ECO SYSTEM-PIG

1	a)	Illustrate the concept of grunt	[L3][CO2]	[5M]
		<p>After invoking the Grunt shell, you can run your Pig scripts in the shell. In addition to that, there are certain useful shell and utility commands provided by the Grunt shell. This chapter explains the shell and utility commands provided by the Grunt shell</p> <p>Shell Commands</p> <p>The Grunt shell of Apache Pig is mainly used to write Pig Latin scripts. Prior to that, we can invoke any shell commands using sh and fs.</p> <p>sh Command</p> <p>Using sh command, we can invoke any shell commands from the Grunt shell. Using sh command from the Grunt shell, we cannot execute the commands that are a part of the shell environment (ex – cd).</p> <p>Syntax</p> <p>Given below is the syntax of sh command.</p> <p>Example</p> <p>We can invoke the ls command of Linux shell from the Grunt shell using the sh option as shown below. In this example, it lists out the files in the /pig/bin/ directory.</p> <pre>grunt> sh ls</pre> <pre>pig pig_1444799121955.log pig.cmd pig.py</pre> <p>fs Command</p> <p>Using the fs command, we can invoke any FsShell commands from the Grunt shell.</p>		

	<p>Syntax</p> <p>Given below is the syntax of fs command.</p> <p>grunt> sh File System command parameters</p> <p>Example</p> <p>We can invoke the ls command of HDFS from the Grunt shell using fs command. In the following example, it lists the files in the HDFS root directory.</p> <p>grunt> fs -ls</p> <p>Found 3 items</p> <pre>drwxrwxrwx - Hadoop supergroup 0 2015-09-08 14:13 Hbase drwxr-xr-x - Hadoop supergroup 0 2015-09-09 14:52 seqgen_data drwxr-xr-x - Hadoop supergroup 0 2015-09-08 11:30 twitter_data</pre> <p>In the same way, we can invoke all the other file system shell commands from the Grunt shell using the fs command.</p>		
	<p>b) Why Do We Need Apache Pig? Identify the features of PIG.</p>	[L4][CO2]	[7M]
	<p>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.</p> <p>Features of Apache Pig</p> <p>Let's see the various uses of Pig technology.</p> <p>1) Ease of programming</p> <p>Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.</p> <p>2) Optimization opportunities</p> <p>It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.</p> <p>3) Extensibility</p> <p>A user-defined function is written in which the user can write their logic to execute over the data set.</p> <p>4) Flexible</p> <p>It can easily handle structured as well as unstructured data.</p>		

	<p>5) In-built operators</p> <p>It contains various type of operators such as sort, filter and joins.</p> <p>Differences between Apache MapReduce and PIG</p> <table><thead><tr><th>Apache MapReduce</th><th>Apache PIG</th></tr></thead><tbody><tr><td>It is a low-level data processing tool.</td><td>It is a high-level data flow tool.</td></tr><tr><td>Here, it is required to develop complex programs using Java or Python.</td><td>It is not required to develop complex programs using Java or Python.</td></tr><tr><td>It is difficult to perform data operations in MapReduce.</td><td>It provides built-in operators for union, sorting and ordering.</td></tr><tr><td>It doesn't allow nested data types.</td><td>It provides nested data types like list, map, etc.</td></tr></tbody></table>	Apache MapReduce	Apache PIG	It is a low-level data processing tool.	It is a high-level data flow tool.	Here, it is required to develop complex programs using Java or Python.	It is not required to develop complex programs using Java or Python.	It is difficult to perform data operations in MapReduce.	It provides built-in operators for union, sorting and ordering.	It doesn't allow nested data types.	It provides nested data types like list, map, etc.		
Apache MapReduce	Apache PIG												
It is a low-level data processing tool.	It is a high-level data flow tool.												
Here, it is required to develop complex programs using Java or Python.	It is not required to develop complex programs using Java or Python.												
It is difficult to perform data operations in MapReduce.	It provides built-in operators for union, sorting and ordering.												
It doesn't allow nested data types.	It provides nested data types like list, map, etc.												
2	<p>What is Pig? How to Install and execute PIG on Hadoop Cluster</p>	[L2][CO5]	[12M]										
	<p>Pig Represents Big Data as data flows. Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as Pig Latin which is used to develop the data analysis codes.</p> <p>Step-by-step installation of Apache Pig on Hadoop cluster on Ubuntu</p> <p>Pre-requisite:</p> <ul style="list-style-type: none">· Ubuntu 16.04 or higher version running (I have installed Ubuntu on Oracle VM (Virtual Machine) VirtualBox),· Run Hadoop on ubuntu (I have installed Hadoop 3.2.1 on Ubuntu 16.04). You may refer to my blog “How to install Hadoop installation” click here for Hadoop installation). <p>Pig installation steps</p> <p>Step 1: Login into Ubuntu</p>												

```

hadoop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
hadoop@hadoop-VirtualBox: ~
hadoop@hadoop-VirtualBox:~$ $ wget https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
$: command not found
hadoop@hadoop-VirtualBox:~$ wget https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
--2022-06-21 11:57:52-- https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42:
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... co
d.
HTTP request sent, awaiting response... 200 OK
Length: 177279333 (169M) [application/x-gzip]
Saving to: 'pig-0.16.0.tar.gz.1'

pig-0.16.0.tar.gz.1  94%[=====] 158.94M  5.19MB/s   eta 2s

```

Step 2: Go to <https://pig.apache.org/releases.html> and copy the path of the latest version of pig that you want to install. Run the following command to download

Apache Pig in Ubuntu:

```
$ wget https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz
```

```

hadoop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
hadoop@hadoop-VirtualBox: ~
hadoop@hadoop-VirtualBox:~$ 

```

Step 3: To untar pig-0.16.0.tar.gz file run the following command:

```
$ tar xvfz pig-0.16.0.tar.gz
```

Step 4: To create a pig folder and move pig-0.16.0 to the pig folder, execute the following command:

```
$ sudo mv /home/hadoop/pig-0.16.0 /home/hadoop/pig
```

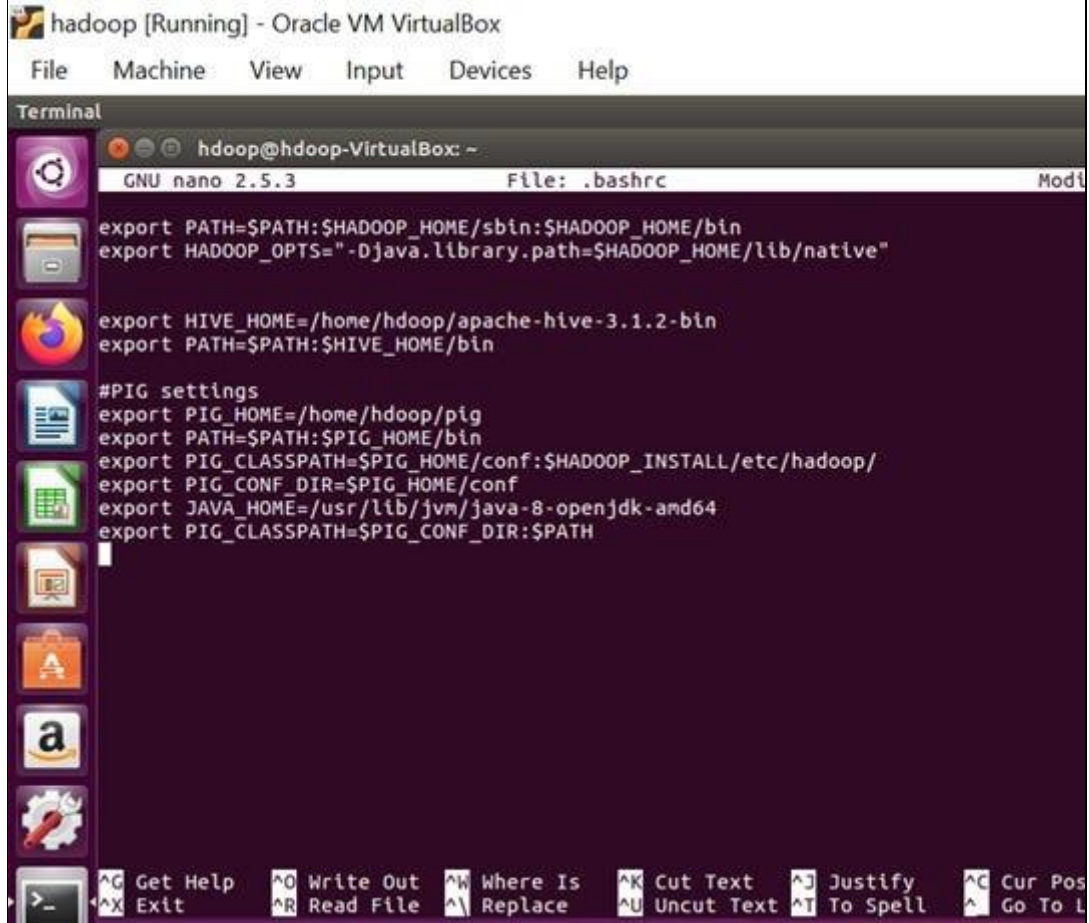
Step 5: Now open the .bashrc file to edit the path and variables/settings for pig.

Run the following command:

```
$ sudo nano .bashrc
```

Add the below given to .bashrc file at the end and save the file.

```
#PIG settings
export PIG_HOME=/home/hadoop/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop/
export PIG_CONF_DIR=$PIG_HOME/conf
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PIG_CLASSPATH=$PIG_CONF_DIR:$PATH
#PIG setting ends
```



Step 6: Run the following command to make the changes effective in the .bashrc file:

```
$ source .bashrc
```

Step 7: To start all Hadoop daemons, navigate to the hadoop-3.2.1/sbin folder and run the following commands:

```
$ ./start-dfs.sh $ ./start-yarn$ jps
```



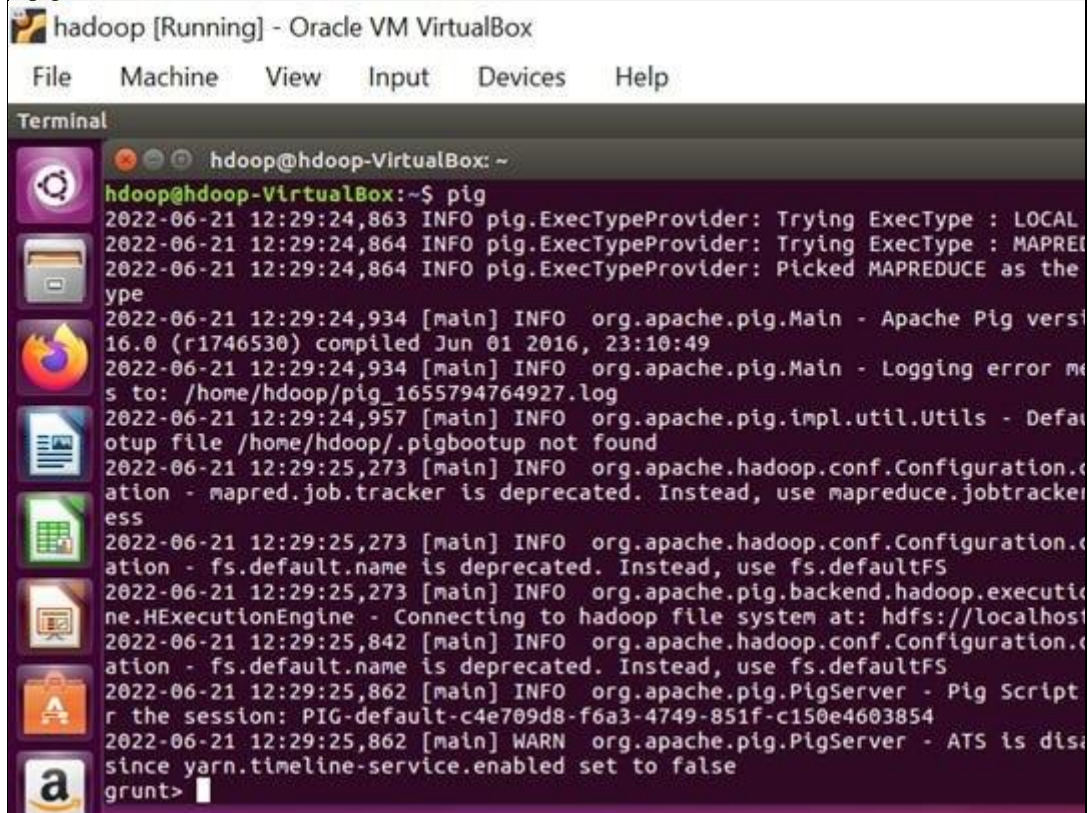
```

hadoop@hadoop-VirtualBox:~$ cd hadoop-3.2.1/sbin
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [hadoop-VirtualBox]
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$ jps
4817 DataNode
5298 ResourceManager
5000 SecondaryNameNode
5450 NodeManager
4683 NameNode
5982 Jps
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$

```

Step 8: Now you can launch pig by executing the following command:

```
$ pig
```



Step 9: Now you are in pig and can perform your desired tasks on pig. You can come out of the pig by the quit command:

```
> quit;
```

3	a) Compare the PIG with Databases with an Example	[L5][CO3]	[6M]
	Hive vs Pig vs SQL		
	Nature of Language	Uses Procedural language called Pig Latin	Uses Declarative language called HiveQL

		Definition	An open source and high-level data flow language with a Multi-query approach	An open source built with an analytical focus used for Analytical queries		General purpose language for transactional																																	
		Suitable for	Suitable for Complex as well as Nested data structure	Ideal for Batch Processing – OLAP (Online Analytical Processing)		Ideal for more straightforward demands for analysis																																	
		Operational for	Semi-structured and structured data	Used only for Structured Data		A domain-specific for a relational management																																	
		Compatibility	Pig works on top of MapReduce	Hive works on top of MapReduce		Not compatible with MapReduce																																	
		Use of Schema	No concept of a schema to store data	Supports Schema for data insertion in tables		Strict use of schema in case of storing																																	
	b)	Evaluate the Expressions and types in Pig Latin.				[L4][CO4]	[6M]																																
		Pig Latin – Data types Given below table describes the Pig Latin data types. <table><tr><th>S.N.</th><th>Data Type</th><th>Description & Example</th></tr><tr><td>1</td><td>int</td><td>Represents a signed 32-bit integer. Example : 8</td></tr><tr><td>2</td><td>long</td><td>Represents a signed 64-bit integer. Example : 5L</td></tr><tr><td>3</td><td>float</td><td>Represents a signed 32-bit floating point. Example : 5.5F</td></tr><tr><td>4</td><td>double</td><td>Represents a 64-bit floating point. Example : 10.5</td></tr><tr><td>5</td><td>chararray</td><td>Represents a character array (string) in Unicode UTF-8 format. Example : ‘tutorials point’</td></tr><tr><td>6</td><td>Bytearray</td><td>Represents a Byte array (blob).</td></tr><tr><td>7</td><td>Boolean</td><td>Represents a Boolean value. Example : true/ false.</td></tr><tr><td>8</td><td>Datetime</td><td>Represents a date-time. Example : 1970-01-01T00:00:00.000+00:00</td></tr><tr><td>9</td><td>Biginteger</td><td>Represents a Java BigInteger. Example : 60708090709</td></tr><tr><td>10</td><td>Bigdecimal</td><td>Represents a Java BigDecimal Example : 185.98376256272893883</td></tr></table>			S.N.	Data Type	Description & Example	1	int	Represents a signed 32-bit integer. Example : 8	2	long	Represents a signed 64-bit integer. Example : 5L	3	float	Represents a signed 32-bit floating point. Example : 5.5F	4	double	Represents a 64-bit floating point. Example : 10.5	5	chararray	Represents a character array (string) in Unicode UTF-8 format. Example : ‘tutorials point’	6	Bytearray	Represents a Byte array (blob).	7	Boolean	Represents a Boolean value. Example : true/ false.	8	Datetime	Represents a date-time. Example : 1970-01-01T00:00:00.000+00:00	9	Biginteger	Represents a Java BigInteger. Example : 60708090709	10	Bigdecimal	Represents a Java BigDecimal Example : 185.98376256272893883		
S.N.	Data Type	Description & Example																																					
1	int	Represents a signed 32-bit integer. Example : 8																																					
2	long	Represents a signed 64-bit integer. Example : 5L																																					
3	float	Represents a signed 32-bit floating point. Example : 5.5F																																					
4	double	Represents a 64-bit floating point. Example : 10.5																																					
5	chararray	Represents a character array (string) in Unicode UTF-8 format. Example : ‘tutorials point’																																					
6	Bytearray	Represents a Byte array (blob).																																					
7	Boolean	Represents a Boolean value. Example : true/ false.																																					
8	Datetime	Represents a date-time. Example : 1970-01-01T00:00:00.000+00:00																																					
9	Biginteger	Represents a Java BigInteger. Example : 60708090709																																					
10	Bigdecimal	Represents a Java BigDecimal Example : 185.98376256272893883																																					

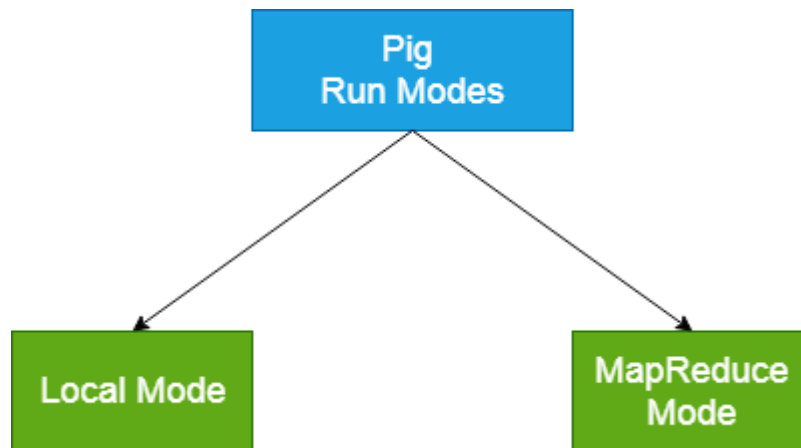
Complex Types

- | | | |
|----|-------|--|
| 11 | Tuple | A tuple is an ordered set of fields.
Example : (raja, 30) |
| 12 | Bag | A bag is a collection of tuples.
Example : {(raju,30),(Mohhammad,45)} |
| 13 | Map | A Map is a set of key-value pairs.
Example : ['name'#:Raju', 'age'#:30] |

4**Examine the different execution modes available in Pig****[L3][CO4]****[12M]**

Apache Pig Run Modes

Apache Pig executes in two modes: Local Mode and MapReduce Mode.



Local Mode

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

The command for local mode grunt shell:

\$ pig-x local

MapReduce Mode

- The MapReduce mode is also known as Hadoop Mode.
- It is the default mode.
- In this Pig renders Pig Latin into MapReduce jobs and executes them on the cluster.

	<ul style="list-style-type: none"> ○ It can be executed against semi-distributed or fully distributed Hadoop installation. ○ Here, the input and output data are present on HDFS. <p>The command for Map reduce mode:</p> <p>\$ pig</p> <p>Or,</p> <p>\$ pig -x mapreduce</p> <p>Ways to execute Pig Program</p> <p>These are the following ways of executing a Pig program on local and MapReduce mode: -</p> <ul style="list-style-type: none"> ○ Interactive Mode - In this mode, the Pig is executed in the Grunt shell. To invoke Grunt shell, run the pig command. Once the Grunt mode executes, we can provide Pig Latin statements and command interactively at the command line. ○ Batch Mode - In this mode, we can run a script file having a .pig extension. These files contain Pig Latin commands. ○ Embedded Mode - In this mode, we can define our own functions. These functions can be called as UDF (User Defined Functions). Here, we use programming languages like Java and Python. 		
5	Construct User Define Functions in Pig Latin.	[L6][CO5]	[12M]
	<p>After writing the UDF and generating the Jar file, follow the steps given below –</p> <p>Step 1: Registering the Jar file</p> <p>After writing UDF (in Java) we have to register the Jar file that contain the UDF using the Register operator. By registering the Jar file, users can intimate the location of the UDF to Apache Pig.</p> <p>Syntax</p> <p>Given below is the syntax of the Register operator.</p> <p>REGISTER path;</p> <p>Example</p> <p>As an example let us register the sample_udf.jar created earlier in this chapter.</p>		

Start Apache Pig in local mode and register the jar file sample_udf.jar as shown below.

```
$cd PIG_HOME/bin
```

```
./pig -x local
```

```
REGISTER '$PIG_HOME/sample_udf.jar'
```

Note – assume the Jar file in the path – /\$PIG_HOME/sample_udf.jar

Step 2: Defining Alias

After registering the UDF we can define an alias to it using the Define operator.

Syntax

Given below is the syntax of the Define operator.

```
DEFINE alias { function | [ `command` [input] [output] [ship] [cache] [stderr] ] };
```

Example

Define the alias for sample_eval as shown below.

```
DEFINE sample_eval sample_eval();
```

Step 3: Using the UDF

After defining the alias you can use the UDF same as the built-in functions.

Suppose there is a file named emp_data in the HDFS /Pig_Data/ directory with the following content.

```
001,Robin,22,newyork
```

```
002,BOB,23,Kolkata
```

```
003,Maya,23,Tokyo
```

```
004,Sara,25,London
```

```
005,David,23,Bhuwaneshwar
```

```
006,Maggy,22,Chennai
```

```
007,Robert,22,newyork
```

```
008,Syam,23,Kolkata
```

```
009,Mary,25,Tokyo
```

```
010,Saran,25,London
```

```
011,Stacy,25,Bhuwaneshwar
```

```
012,Kelly,22,Chennai
```

And assume we have loaded this file into Pig as shown below.

```
grunt> emp_data = LOAD 'hdfs://localhost:9000/pig_data/emp1.txt' USING  
PigStorage(',')
```

		<div>as (id:int, name:chararray, age:int, city:chararray);</div> <div>Let us now convert the names of the employees in to upper case using the UDF sample_eval.</div> <div>grunt> Upper_case = FOREACH emp_data GENERATE sample_eval(name);</div> <div>Verify the contents of the relation Upper_case as shown below.</div> <div>grunt> Dump Upper_case;</div>																							
6	a)	Explain about Arithmetic Operators in Pig Latin .	[L2][CO3]	[6M]																					
		<div>Arithmetic Operators</div> <div>These pig latin operators are basic mathematical operators.</div> <table><thead><tr><th>Operator</th><th>Description</th><th>Example</th></tr></thead><tbody><tr><td>+</td><td>Addition – It add values on any single side of the operator.</td><td>if a= 10, b= 30, a + b gives 40</td></tr><tr><td>–</td><td>Subtraction – It reduces the value of right hand operand from left hand operand.</td><td>Technology is evolving rapidly! Stay updated with DataFlair on if a= 40, b= 30, a-b gives 10</td></tr><tr><td>*</td><td>Multiplication – This operation multiplies the values on either side of the operator.</td><td>a * b gives you 1200</td></tr><tr><td>/</td><td>Division – This operator divides the left hand operand by right hand operand.</td><td>if a= 40, b= 20, b / a results to 2</td></tr><tr><td>%</td><td>Modulus – It divides the left hand operand by right hand operand with remainder as result.</td><td>if a= 40, b= 30, b%a results to 10</td></tr><tr><td>? :</td><td>Bincond – It evaluates the Boolean operators. Moreover, it has three operands below. variable x = (expression) ? value1 if true : value2 if false.</td><td>b = (a == 1)? 40: 20; if a = 1 the value is 40. if a!=1 the value is 20.</td></tr></tbody></table> <div>b) Find the Grouping and Joining Data in Pig Latin.</div>	Operator	Description	Example	+	Addition – It add values on any single side of the operator.	if a= 10, b= 30, a + b gives 40	–	Subtraction – It reduces the value of right hand operand from left hand operand.	Technology is evolving rapidly! Stay updated with DataFlair on if a= 40, b= 30, a-b gives 10	*	Multiplication – This operation multiplies the values on either side of the operator.	a * b gives you 1200	/	Division – This operator divides the left hand operand by right hand operand.	if a= 40, b= 20, b / a results to 2	%	Modulus – It divides the left hand operand by right hand operand with remainder as result.	if a= 40, b= 30, b%a results to 10	? :	Bincond – It evaluates the Boolean operators. Moreover, it has three operands below. variable x = (expression) ? value1 if true : value2 if false.	b = (a == 1)? 40: 20; if a = 1 the value is 40. if a!=1 the value is 20.	[L3][CO3]	[6M]
Operator	Description	Example																							
+	Addition – It add values on any single side of the operator.	if a= 10, b= 30, a + b gives 40																							
–	Subtraction – It reduces the value of right hand operand from left hand operand.	Technology is evolving rapidly! Stay updated with DataFlair on if a= 40, b= 30, a-b gives 10																							
*	Multiplication – This operation multiplies the values on either side of the operator.	a * b gives you 1200																							
/	Division – This operator divides the left hand operand by right hand operand.	if a= 40, b= 20, b / a results to 2																							
%	Modulus – It divides the left hand operand by right hand operand with remainder as result.	if a= 40, b= 30, b%a results to 10																							
? :	Bincond – It evaluates the Boolean operators. Moreover, it has three operands below. variable x = (expression) ? value1 if true : value2 if false.	b = (a == 1)? 40: 20; if a = 1 the value is 40. if a!=1 the value is 20.																							
		<div>Pig Latin – Grouping and Joining:</div> <div>JOIN:</div> <div>Join concept is similar to Sql joins, here we have many types of joins such as Inner join, outer join and some specialized joins.</div> <div>INNER JOIN:</div>																							

The JOIN operator always performs an inner join. Inner joins ignore null keys, so it makes sense to filter them out before the join.

Note: Both Cogroup and join work in a similar way, just the difference is Cogroup creates a nested set of output records.

```
X = JOIN A BY a1, B BY b1;
```

OUTER JOIN:

Use the OUTER JOIN operator to perform left, right, or full outer joins. Outer joins will only work provided the relations which need to produce nulls (in the case of non-matching keys) have schemas.

```
C = JOIN A BY $0 LEFT OUTER, B BY $0;
```

REPLICATED:

This join is used when we know that one or more relations is small enough to fit into the memory. Then the pig performs this join very fast because here all the hadoop work is done on the map side.

In this large relation is followed by one or more small relation. If the small relation is more than the memory, then the process fails and we get an error. We can use this join using both inner and outer join.

```
C = JOIN A BY $0 LEFT, B BY $0 USING 'replicated';
```

There are also Skewed and Merge joins, which have certain limitations and conditions. Here is an example

```
C = JOIN A BY name FULL, B BY name USING 'skewed';
```

```
C = JOIN A BY a1, B BY b1 USING 'merge';
```

GROUP:

Similar to Group in Sql, here we use group for one relation and Cogroup of more number of relations. Both GROUP and COGROUP are similar to each other.

```
B = GROUP A BY age;
```

```
X = GROUP A BY f2*f3;
```

```
X = COGROUP A BY owner INNER, B BY friend2 INNER;
```

```
B = GROUP A BY (tcid, tpid);
```

CROSS:

CROSS operator to use to compute the cross product (Cartesian product) of two or more relations.

```
X = CROSS A, B;
```

	<p>Relational Operators:</p> <p>Relational operators are the main tools Pig Latin provides to operate on the data. It allows you to transform the data by sorting, grouping, joining, projecting and filtering. This section covers the basic relational operators.</p> <p>LOAD:</p> <p>LOAD operator is used to load data from the file system or HDFS storage into a Pig relation.</p> <p>In this example, the Load operator loads data from file 'first' to form relation 'loading1'. The field names are user, url, id.</p> <p>FOREACH:</p> <p>This operator generates data transformations based on columns of data. It is used to add or remove fields from a relation. Use FOREACH-GENERATE operation to work with columns of data.</p> <p>FILTER:</p> <p>This operator selects tuples from a relation based on a condition.</p> <p>In this example, we are filtering the record from 'loading1' when the condition 'id' is greater than 8.</p> <p>DISTINCT:</p> <p>Distinct removes duplicate tuples in a relation. Lets take an input file as below, which has amr,crap,8 and amr,myblog,10 twice in the file. When we apply distinct on the data in this file, duplicate entries are removed.</p>		
8	Develop Pig Latin Schemas and Functions.	[L3][CO5]	[12M]
	<p>Schemas</p> <p>Schemas enable you to assign names to fields and declare types for fields. Schemas are optional but we encourage you to use them whenever possible; type declarations result in better parse-time error checking and more efficient code execution.</p> <p>Schemas for simple types and complex types can be used anywhere a schema definition is appropriate.</p> <p>Schemas are defined with the LOAD, STREAM, and FOREACH operators using the AS clause. If you define a schema using the LOAD operator, then it is the load function that enforces the schema (see LOAD and User Defined Functions for more information).</p> <p>Known Schema Handling</p> <p>Note the following:</p> <p>You can define a schema that includes both the field name and field type.</p> <p>You can define a schema that includes the field name only; in this case, the field type defaults to bytearray.</p> <p>You can choose not to define a schema; in this case, the field is un-named and the field type defaults to bytearray.</p>		

If you assign a name to a field, you can refer to that field using the name or by positional notation. If you don't assign a name to a field (the field is un-named) you can only refer to the field using positional notation.

If you assign a type to a field, you can subsequently change the type using the cast operators. If you don't assign a type to a field, the field defaults to bytearray; you can change the default type using the cast operators.

Unknown Schema Handling

Note the following:

When you JOIN/COGROUP/CROSS multiple relations, if any relation has an unknown schema (or no defined schema, also referred to as a null schema), the schema for the resulting relation is null.

If you FLATTEN a bag with empty inner schema, the schema for the resulting relation is null.

If you UNION two relations with incompatible schema, the schema for resulting relation is null.

If the schema is null, Pig treats all fields as bytearray (in the backend, Pig will determine the real type for the fields dynamically)

See the examples below. If a field's data type is not specified, Pig will use bytearray to denote an unknown type. If the number of fields is not known, Pig will derive an unknown schema.

List of Apache Pig Built in Functions

Let's discuss various Apache Pig Built in Functions namely eval, load, store, math, string, bag, and tuple, one by one in depth.

a. Eval Functions

Eval Functions is the first types of Pig Built in Functions. Here are the Pig Eval functions, offered by Apache Pig.

i. AVG()

- AVG Syntax

AVG(expression)

We use AVG(), to compute the average of the numerical values within a bag.

- AVG Example

In this example, the average GPA for each Employee is computed

A = LOAD 'Employee.txt' AS (name:chararray, term:chararray, gpa:float);

DUMP A;

(johny,fl,3.9F)

(johny,wt,3.7F)

(johny,sp,4.0F)

(johny,sm,3.8F)

(Mariya,fl,3.8F)

(Mariya,wt,3.9F)

(Mariya,sp,4.0F)

(Mariya,sm,4.0F)

B = GROUP A BY name;

DUMP B;

(johny,{(johny,fl,3.9F),(johny,wt,3.7F),(johny,sp,4.0F),(johny,sm,3.8F)})

(Mariya,{(Mariya,fl,3.8F),(Mariya,wt,3.9F),(Mariya,sp,4.0F),(Mariya,sm,4.0F)})

C = FOREACH B GENERATE A.name, AVG(A.gpa);

DUMP C;

	<p>((johny),(johny),(johny),(johny}),3.850000023841858) ((Mariya),(Mariya),(Mariya),(Mariya}),3.925000011920929)</p> <p>ii. BagToString() This function is used to concatenate the elements of a bag into a string. We can place a delimiter between these values (optional) while concatenating.</p> <p>iii. CONCAT() • The syntax of CONCAT()</p> <p>CONCAT (expression, expression) We use this Pig Function to concatenate two or more expressions of the same type. • Example of CONCAT() In this example, fields f1, an underscore string literal, f2 and f3 are concatenated. X = LOAD 'data' as (f1:chararray, f2:chararray, f3:chararray); DUMP X; (apache,open,source) (hadoop,map,reduce) (pig,pig,latin) Y = FOREACH X GENERATE CONCAT(f1, '_', f2,f3); DUMP Y; (apache_opensource) (hadoop_mapreduce) (pig_piglatin)</p> <p>iv. COUNT() • The syntax of COUNT() COUNT(expression) While counting the number of tuples in a bag, we use it to get the number of elements in a bag.</p>																										
9	a) Explain about the data types in Pig Latin.	[L2][CO2]	[6M]																								
	<p>Pig Latin – Data types</p> <p>Given below table describes the Pig Latin data types.</p> <table><tr><th>S.N.</th><th>Data Type</th><th>Description & Example</th></tr><tr><td>1</td><td>int</td><td>Represents a signed 32-bit integer. Example : 8</td></tr><tr><td>2</td><td>long</td><td>Represents a signed 64-bit integer. Example : 5L</td></tr><tr><td>3</td><td>float</td><td>Represents a signed 32-bit floating point. Example : 5.5F</td></tr><tr><td>4</td><td>double</td><td>Represents a 64-bit floating point. Example : 10.5</td></tr><tr><td>5</td><td>chararray</td><td>Represents a character array (string) in Unicode UTF-8 format. Example : 'tutorials point'</td></tr><tr><td>6</td><td>Bytearray</td><td>Represents a Byte array (blob).</td></tr><tr><td>7</td><td>Boolean</td><td>Represents a Boolean value. Example : true/ false.</td></tr></table>	S.N.	Data Type	Description & Example	1	int	Represents a signed 32-bit integer. Example : 8	2	long	Represents a signed 64-bit integer. Example : 5L	3	float	Represents a signed 32-bit floating point. Example : 5.5F	4	double	Represents a 64-bit floating point. Example : 10.5	5	chararray	Represents a character array (string) in Unicode UTF-8 format. Example : 'tutorials point'	6	Bytearray	Represents a Byte array (blob).	7	Boolean	Represents a Boolean value. Example : true/ false.		
S.N.	Data Type	Description & Example																									
1	int	Represents a signed 32-bit integer. Example : 8																									
2	long	Represents a signed 64-bit integer. Example : 5L																									
3	float	Represents a signed 32-bit floating point. Example : 5.5F																									
4	double	Represents a 64-bit floating point. Example : 10.5																									
5	chararray	Represents a character array (string) in Unicode UTF-8 format. Example : 'tutorials point'																									
6	Bytearray	Represents a Byte array (blob).																									
7	Boolean	Represents a Boolean value. Example : true/ false.																									

		<p>8 Datetime Represents a date-time. Example : 1970-01-01T00:00:00.000+00:00</p> <p>9 BigInteger Represents a Java BigInteger. Example : 60708090709</p> <p>10 BigDecimal Represents a Java BigDecimal Example : 185.98376256272893883</p> <p style="text-align: center;">Complex Types</p> <p>11 Tuple A tuple is an ordered set of fields. Example : (raja, 30)</p> <p>12 Bag A bag is a collection of tuples. Example : {(raju,30),(Mohhammad,45)}</p> <p>13 Map A Map is a set of key-value pairs. Example : ['name' #'Raju', 'age' #30]</p> <p>b) Develop a program to calculate the maximum recorded temperature by year for the weather dataset in Pig Latin.</p>	[L6][CO5]	[6M]
		<p>MapReduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. Our previous traversal has given an introduction about MapReduce. This traversal explains how to design a MapReduce program. The aim of the program is to find the Maximum temperature recorded for each year of NCDC data. The input for our program is weather data files for each year. This weather data is collected by National Climatic Data Center – NCDC from weather sensors at all over the world. You can find weather data for each year from ftp://ftp.ncdc.noaa.gov/pub/data/noaa. MapReduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. Our previous traversal has given an</p> <p>MapReduce is based on set of key value pairs. So first we have to decide on the types for the key/value pairs for the input. Map Phase: The input for Map phase is set of weather data files as shown in snap shot. The types of input key value pairs are LongWritable and Text and the types of output key value pairs are Text and IntWritable. Each Map task extracts the temperature data from the given year file. The output of the map phase is set of key value pairs. Set of keys are the years. Values are the temperature of each year.</p> <p>Reduce Phase: Reduce phase takes all the values associated with a particular key. That is all the temperature values belong to a particular year is fed to a same reducer. Then each reducer finds the highest recorded temperature for each year. The types of output key value pairs in Map phase</p> <p>is same for the types of input key value pairs in reduce phase (Text and IntWritable). The types of</p> <p>output key value pairs in reduce phase is too Text and IntWritable.</p> <p>So, in this example we write three java classes:</p> <ul style="list-style-type: none"> <input type="checkbox"/> HighestMapper.java <input type="checkbox"/> HighestReducer.java <input type="checkbox"/> HighestDriver.java <p>Program: HighestMapper.java</p> <pre>import java.io.IOException; import org.apache.hadoop.io.*; import org.apache.hadoop.mapred.*; public class HighestMapper extends MapReduceBase implements</pre>		

```
Mapper<LongWritable, Text,
Text, IntWritable>
{
    public static final int MISSING = 9999;
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
    output,
    Reporter reporter) throws IOException
    {
        String line = value.toString();
        String year = line.substring(15,19);
        int temperature;
        if (line.charAt(87)=='+')
            temperature = Integer.parseInt(line.substring(88, 92));
        else
            temperature = Integer.parseInt(line.substring(87, 92));
        String quality = line.substring(92, 93);
        if(temperature != MISSING && quality.matches("[01459]"))
            output.collect(new Text(year),new IntWritable(temperature));
    }
}

HighestReducer.java
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class HighestReducer extends MapReduceBase implements Reducer<Text,
IntWritable,
Text, IntWritable>
{
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable>
    output, Reporter reporter) throws IOException
    {
        int max_temp = 0;
        ;
        while (values.hasNext())
        {
            BIG DATA ANALYTICS LABORATORY (19A05602P)
            16
            Department of CSE
            int current=values.next().get();
            if ( max_temp < current)
                max_temp = current;
        }
        output.collect(key, new IntWritable(max_temp/10));
    }
}

HighestDriver.java
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
```



```

import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class HighestDriver extends Configured implements Tool{
public int run(String[] args) throws Exception
{
JobConf conf = new JobConf(getConf(), HighestDriver.class);
conf.setJobName("HighestDriver");
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(HighestMapper.class);
conf.setReducerClass(HighestReducer.class);
Path inp = new Path(args[0]);
Path out = new Path(args[1]);
FileInputFormat.addInputPath(conf, inp);
FileOutputFormat.setOutputPath(conf, out);
JobClient.runJob(conf);
return 0;
}
public static void main(String[] args) throws Exception
{
int res = ToolRunner.run(new Configuration(), new HighestDriver(),args);
System.exit(res);
}
}

```

10	a)	Discriminate the Structures, Statements in Pig Latin	[L4][CO1]	[6M]
----	----	---	-----------	------

Components of Pig

There are two major components of the Pig:

- Pig Latin script language
- A runtime engine

Pig Latin script language

The Pig Latin script is a procedural data flow language. It contains syntax and commands that can be applied to implement business logic. Examples of Pig Latin are LOAD and STORE.

A runtime engine

The runtime engine is a compiler that produces sequences of MapReduce programs. It uses HDFS to store and retrieve data. It is also used to interact with the Hadoop system (HDFS and MapReduce).

The runtime engine parses, validates, and compiles the script operations into a sequence of MapReduce jobs.

	<p>How Pig Works and Stages of Pig Operations</p> <p>Pig operations can be explained in the following three stages:</p> <p>Stage 1: Load data and write Pig script</p> <hr/> <pre> A = LOAD 'myfile' AS (x, y, z); B = FILTER A by x > 0; C = GROUP B BY x; D = FOREACH A GENERATE x, COUNT(B); STORE D INTO 'output'; </pre> <hr/> <p>In this stage, data is loaded and Pig script is written.</p> <p>Stage 2: Pig Operations</p> <p>In the second stage, the Pig execution engine Parses and checks the script. If it passes the script optimized and a logical and physical plan is generated for execution.</p> <p>The job is submitted to Hadoop as a job defined as a MapReduce Task. Pig Monitors the status of the job using Hadoop API and reports to the client.</p> <p>Stage 3: Execution of the plan</p> <p>In the final stage, results are dumped on the section or stored in HDFS depending on the user command.</p> <p>Let us now understand a few salient features of Pig</p>		
	b) Evaluate Data Processing Operators in Pig Latin.	[L5][CO4]	[6M]
	<p>Here's an example of using Pig Storage to store tuples as plain-text values separated by a colon character:</p> <pre> grunt> STORE A INTO 'out' USING PigStorage(':'); </pre>		

```
grunt> cat out
```

```
Joe:cherry:2
```

```
Ali:apple:3
```

```
Joe:banana:2
```

```
Eve:apple:7
```

Filtering Data

Once you have some data loaded into a relation, often the next step is to filter it to remove the data that you are not interested in. By filtering early in the processing pipeline, you minimize the amount of data flowing through the system, which can improve efficiency

FOREACH...GENERATE

We have already seen how to remove rows from a relation using the FILTER operator with simple expressions and a UDF. The FOREACH...GENERATE operator is used to act on every row in a relation. It can be used to remove fields or to generate new ones.

In this example, we do both:

```
grunt> DUMP A;
```

```
(Joe,cherry,2)
```

```
(Ali,apple,3)
```

```
(Joe,banana,2)
```

```
(Eve,apple,7)
```

```
grunt> B = FOREACH A GENERATE $0, $2+1, 'Constant';
```

```
grunt> DUMP B;
```

```
(Joe,3,Constant)
```

```
(Ali,4,Constant)
```

```
(Joe,3,Constant)
```

```
(Eve,8,Constant)
```

Here we have created a new relation B with three fields. Its first field is a projection of the first field (\$0) of A. B's second field is the third field of A (\$2) with one added to it. B's third field is a constant field (every row in B has the same third field) with the chararray value Constant.

STREAM

The STREAM operator allows you to transform data in a relation using an external program or script. It is named by analogy with Hadoop Streaming, which provides a similar capability for MapReduce

STREAM can use built-in commands with arguments. Here is an example that uses the Unix cut command to extract the second field of each tuple in A. Note that the command and its arguments are enclosed in backticks:

```
grunt> C = STREAM A THROUGH `cut -f 2`;
```

```
grunt> DUMP C;
```

```
(cherry)
```

```
(apple)
```

```
(banana)
```

```
(apple)
```

The STREAM operator uses PigStorage to serialize and deserialize relations to and from the program's standard input and output streams. Tuples in A are converted to tabdelimited lines that are passed to the script. The output of the script is read one line at a time and split on tabs to create new tuples for the output relation C. You can provide a custom serializer and deserializer, which implement PigToStream and StreamToPig, respectively (both in the org.apache.pig package), using the DEFINE operator.

Grouping and Joining Data

Joining datasets in MapReduce takes some work on the part of the programmer whereas Pig has very good built-in support for join operations, making it much more approachable. Since the large datasets that are suitable for analysis by Pig (and MapReduce in general) are not normalized, joins are used more infrequently in Pig than they are in SQL.

JOIN

Let's look at an example of an inner join. Consider the relations A and B:

```
grunt> DUMP A;
```

```
(2,Tie)
```

```
(4,Coat)
```

```
(3,Hat)
```

```
(1,Scarf)
```

```
grunt> DUMP B;
```

```
(Joe,2)
```

```
(Hank,4)
```

```
(Ali,0)
```

```
(Eve,3)
```

```
(Hank,2)
```

We can join the two relations on the numerical (identity) field in each:

```
grunt> C = JOIN A BY $0, B BY $1;
```

```
grunt> DUMP C;
```

```
(2,Tie,Joe,2)
```

```
(2,Tie,Hank,2)
```

```
(3,Hat,Eve,3)
```

```
(4,Coat,Hank,4)
```

This is a classic inner join, where each match between the two relations corresponds to a row in the result. (It's actually an equijoin because the join predicate is equality.) The result's fields are made up of all the fields of all the input relations.

COGROUP

JOIN always gives a flat structure: a set of tuples. The COGROUP statement is similar to JOIN, but instead creates a nested set of output tuples. This can be useful if you want to exploit the structure in subsequent statements:

```
grunt> D = COGROUP A BY $0, B BY $1;
```

```
grunt> DUMP D;
```

```
(0,{},{(Ali,0)})
```

```
(1,{(1,Scarf)},{})
```

```
(2,{(2,Tie)},{(Joe,2),(Hank,2)})
```

```
(3,{(3,Hat)},{(Eve,3)})
```

```
(4,{(4,Coat)},{(Hank,4)})
```

COGROUP generates a tuple for each unique grouping key. The first field of each tuple is the key, and the remaining fields are bags of tuples from the relations with a matching key. The first bag contains the matching tuples from relation A with the same key.

CROSS

Pig Latin includes the cross-product operator (also known as the cartesian product), which joins every tuple in a relation with every tuple in a second relation (and with every tuple in further relations if supplied). The size of the output is the product of the size of the inputs, potentially making the output very large:

GROUP

Although COGROUP groups the data in two or more relations, the GROUP statement groups the data in a single relation. GROUP supports grouping by more than equality of keys: you can use an expression or user-defined function as the group key.

Sorting Data

Relations are unordered in Pig. Consider a relation A:

```
grunt> DUMP A;
```

```
(2,3)
```

```
(1,2)
```

```
(2,4)
```

There is no guarantee which order the rows will be processed in. In particular, when retrieving the contents of A using DUMP or STORE, the rows may be written in any order. If you want to impose an order on the output, you can use the ORDER operator to sort a relation by one or more fields. The default sort order compares fields of the same type using the natural ordering, and different types are given an arbitrary, but deterministic, ordering (a tuple is always “less than” a bag, for example).

The following example sorts A by the first field in ascending order and by the second field in descending order:

```
grunt> B = ORDER A BY $0, $1 DESC;
```

```
grunt> DUMP B;
```

```
(1,2)
```

```
(2,4)
```

```
(2,3)
```

Any further processing on a sorted relation is not guaranteed to retain its order. For example:

```
grunt> C = FOREACH B GENERATE *;
```

Even though relation C has the same contents as relation B, its tuples may be emitted in any order by a DUMP or a STORE. It is for this reason that it is usual to perform the ORDER operation just before retrieving the output.

Combining and Splitting Data

Sometimes you have several relations that you would like to combine into one. For this, the UNION statement is used. For example:

```
grunt> DUMP A;
```

```
(2,3)
```

```
(1,2)
```

```
(2,4)
```

```
grunt> DUMP B;
```

```
(z,x,8)
```

```
(w,y,1)
```

```
grunt> C = UNION A, B;
```

```
grunt> DUMP C;
```

```
(2,3)
```

```
(1,2)
```

```
(2,4)
```

```
(z,x,8)
```

```
(w,y,1)
```

C is the union of relations A and B, and because relations are unordered, the order of the tuples in C is undefined. Also, it's possible to form the union of two relations with different schemas or with different numbers of fields, as we have done here. Pig attempts to merge the schemas from the relations that UNION is operating on. In this case, they are incompatible, so C has no schema:



**SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY:: PUTTUR
(AUTONOMOUS)**

Siddharth Nagar, Narayanavanam Road – 517583

QUESTION BANK (DESCRIPTIVE)

Subject with Code: BIG DATA (20CS0538)

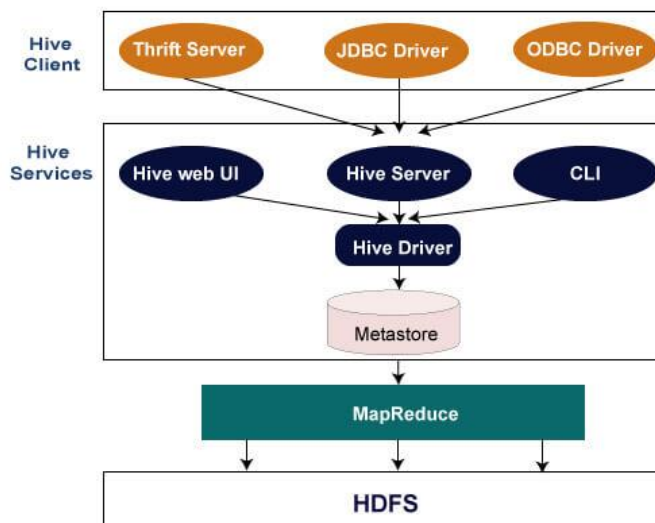
Course & Branch: B.Tech – CSE, CSM

Regulation: R20

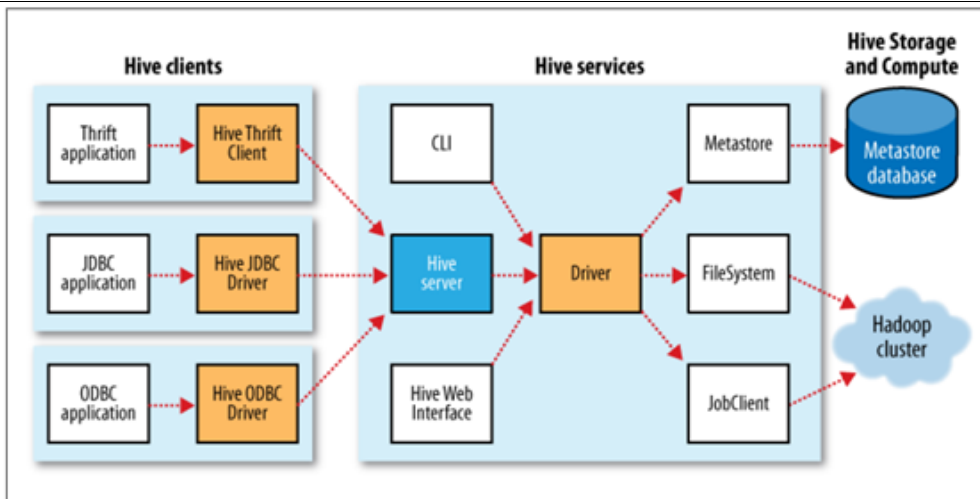
Year & Sem: IV-B.Tech & I-Sem

**UNIT V
HIVE, HBASE, BIG SQL**

1	Illustrate Hive table with example.	[L3][CO5]	[12M]															
	<p>Tables in the hive are analogous to tables in a relational database management system. Each table belongs to a directory in HDFS. By default, it is /user/hive/warehouse directory. For instance, a table named students will be located at /user/hive/warehouse/students.</p> <p>Create Table is a statement used to create a table in Hive. The syntax and example are as follows:</p> <p>Syntax</p> <pre>CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name [(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment] [ROW FORMAT row_format] [STORED AS file_format]</pre> <p>Example</p> <p>Let us assume you need to create a table named employee using CREATE TABLE statement. The following table lists the fields and their data types in employee table:</p> <table><tr><th>Sr.No</th><th>Field Name</th><th>Data Type</th></tr><tr><td>1</td><td>Eid</td><td>int</td></tr><tr><td>2</td><td>Name</td><td>String</td></tr><tr><td>3</td><td>Salary</td><td>Float</td></tr><tr><td>4</td><td>Designation</td><td>string</td></tr></table> <p>The following data is a Comment, Row formatted fields such as Field terminator, Lines terminator, and Stored File type.</p>	Sr.No	Field Name	Data Type	1	Eid	int	2	Name	String	3	Salary	Float	4	Designation	string		
Sr.No	Field Name	Data Type																
1	Eid	int																
2	Name	String																
3	Salary	Float																
4	Designation	string																

	COMMENT ‘Employee details’ FIELDS TERMINATED BY ‘\t’ LINES TERMINATED BY ‘\n’ STORED IN TEXT FILE																										
2	Discuss about Hive shell command line interface.	[L2][CO5]	[12M]																								
	<p>Hive CLI is a legacy tool which had two main use cases. The first is that it served as a thick client for SQL on Hadoop and the second is that it served as a command line tool for Hive Server (the original Hive server, now often referred to as “HiveServer1”).</p> <p>Hive Command Line Options</p> <p>To get help, run “hive -H” or “hive –help”. Usage (as it is in Hive 0.9.0)</p> <p>usage: hive</p> <table><thead><tr><th>Option</th><th>Explanation</th></tr></thead><tbody><tr><td>-d,–define <key=value></td><td>Variable substitution to apply to hive commands. e.g</td></tr><tr><td>-e <quoted-query-string></td><td>SQL from command line</td></tr><tr><td>-f <filename></td><td>SQL from files</td></tr><tr><td>-H,–help</td><td>Print help information</td></tr><tr><td>-h <hostname></td><td>Connecting to Hive Server on remote host</td></tr><tr><td>–hiveconf <property=value></td><td>Use value for given property</td></tr><tr><td>–hivevar <key=value></td><td>Variable substitution to apply to hive commands. e.g</td></tr><tr><td>-i <filename></td><td>Initialization SQL file</td></tr><tr><td>-p <port></td><td>Connecting to Hive Server on port number</td></tr><tr><td>-S,–silent</td><td>Silent mode in interactive shell</td></tr><tr><td>-v,–verbose</td><td>Verbose mode (echo executed SQL to the console)</td></tr></tbody></table>	Option	Explanation	-d,–define <key=value>	Variable substitution to apply to hive commands. e.g	-e <quoted-query-string>	SQL from command line	-f <filename>	SQL from files	-H,–help	Print help information	-h <hostname>	Connecting to Hive Server on remote host	–hiveconf <property=value>	Use value for given property	–hivevar <key=value>	Variable substitution to apply to hive commands. e.g	-i <filename>	Initialization SQL file	-p <port>	Connecting to Hive Server on port number	-S,–silent	Silent mode in interactive shell	-v,–verbose	Verbose mode (echo executed SQL to the console)		
Option	Explanation																										
-d,–define <key=value>	Variable substitution to apply to hive commands. e.g																										
-e <quoted-query-string>	SQL from command line																										
-f <filename>	SQL from files																										
-H,–help	Print help information																										
-h <hostname>	Connecting to Hive Server on remote host																										
–hiveconf <property=value>	Use value for given property																										
–hivevar <key=value>	Variable substitution to apply to hive commands. e.g																										
-i <filename>	Initialization SQL file																										
-p <port>	Connecting to Hive Server on port number																										
-S,–silent	Silent mode in interactive shell																										
-v,–verbose	Verbose mode (echo executed SQL to the console)																										
3	a) Draw a neat sketch of Hive architecture.	[L3][CO2]	[4M]																								
	<p>he following architecture explains the flow of submission of query into Hive.</p>  <pre>graph TD subgraph Hive_Client [Hive Client] TS([Thrift Server]) JD([JDBC Driver]) OD([ODBC Driver]) end subgraph Hive_Services [Hive Services] HUI([Hive web UI]) HS([Hive Server]) CLI([CLI]) HD([Hive Driver]) MS[(Metastore)] end MR[MapReduce] HDFS[HDFS] TS --> HS JD --> HS OD --> HS HUI --> HS CLI --> HS HS --> HD HD --> MS MS --> MR MR --> HDFS</pre>																										

	b) Explain about components of Hive architecture.	[L2][CO2]	[8M]
	<p>Hive Client</p> <p>Hive allows writing applications in various languages, including Java, Python, and C++. It supports different types of clients such as:-</p> <ul style="list-style-type: none"> ○ Thrift Server - It is a cross-language service provider platform that serves the request from all those programming languages that supports Thrift. ○ JDBC Driver - It is used to establish a connection between hive and Java applications. The JDBC Driver is present in the class <code>org.apache.hadoop.hive.jdbc.HiveDriver</code>. ○ ODBC Driver - It allows the applications that support the ODBC protocol to connect to Hive. <p>Hive Services</p> <p>The following are the services provided by Hive:-</p> <ul style="list-style-type: none"> ○ Hive CLI - The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands. ○ Hive Web User Interface - The Hive Web UI is just an alternative of Hive CLI. It provides a web-based GUI for executing Hive queries and commands. ○ Hive MetaStore - It is a central repository that stores all the structure information of various tables and partitions in the warehouse. It also includes metadata of column and its type information, the serializers and deserializers which is used to read and write data and the corresponding HDFS files where the data is stored. ○ Hive Server - It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver. ○ Hive Driver - It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver. It transfers the queries to the compiler. ○ Hive Compiler - The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions. It converts HiveQL statements into MapReduce jobs. ○ Hive Execution Engine - Optimizer generates the logical plan in the form of DAG of map-reduce tasks and HDFS tasks. In the end, the execution engine executes the incoming tasks in the order of their dependencies. 		



4 a) Deduce the various services offered by Hive.

[L4][CO4]

[6M]

Hive Services

The following are the services provided by Hive:-

- Hive CLI - The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands.
- Hive Web User Interface - The Hive Web UI is just an alternative of Hive CLI. It provides a web-based GUI for executing Hive queries and commands.
- Hive MetaStore - It is a central repository that stores all the structure information of various tables and partitions in the warehouse. It also includes metadata of column and its type information, the serializers and deserializers which is used to read and write data and the corresponding HDFS files where the data is stored.
- Hive Server - It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver.
- Hive Driver - It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver. It transfers the queries to the compiler.
- Hive Compiler - The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions. It converts HiveQL statements into MapReduce jobs.
- Hive Execution Engine - Optimizer generates the logical plan in the form of DAG of map-reduce tasks and HDFS tasks. In the end, the execution engine executes the incoming tasks in the order of their dependencies.

b) Examine the Characteristics of HBase

[L4][CO1]

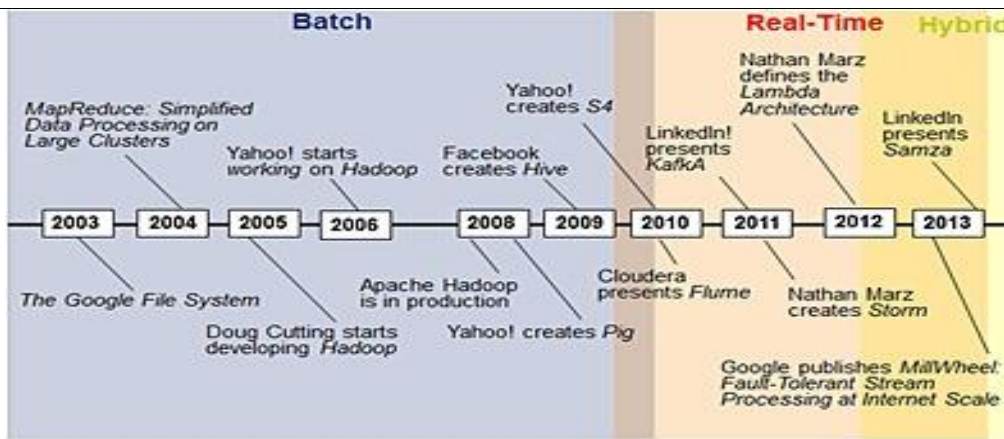
[6M]

HBase, which has the following characteristics:

No real indexes

Rows are stored sequentially, as are the columns within each row. Therefore, no

		<p>issues with index bloat, and insert performance is independent of table size.</p> <p>Automatic partitioning As your tables grow, they will automatically be split into regions and distributed across all available nodes.</p> <p>Scale linearly and automatically with new nodes Add a node, point it to the existing cluster, and run the regionserver. Regions will automatically rebalance, and load will spread evenly.</p> <p>Commodity hardware Clusters are built on \$1,000–\$5,000 nodes rather than \$50,000 nodes. RDBMSs are I/O hungry, requiring more costly hardware.</p> <p>Fault tolerance Lots of nodes means each is relatively insignificant. No need to worry about individual node downtime.</p> <p>Batch processing MapReduce integration allows fully parallel, distributed jobs against your data with locality awareness.</p> <p>If you stay up at night worrying about your database (uptime, scale, or speed), you should seriously consider making a jump from the RDBMS world to HBase. Utilize a solution that was intended to scale rather than a solution based on stripping down and throwing money at what used to work. With HBase, the software is free, the hardware is cheap, and the distribution is intrinsic.</p> <ul style="list-style-type: none"> • HBase is linearly scalable. • It has automatic failure support. • It provides consistent read and writes. • It integrates with Hadoop, both as a source and a destination. • It has easy java API for client. • It provides data replication across clusters. 		
5	a)	Infer the advantages of Hive over traditional databases?	[L2][CO5]	[6M]
		<p>Hive in Big Data is an easy-to-use software program that allows one to analyze large amounts of data using a collection process. An efficient system, using standard software that uses HiveQL, a very similar language to the SQL-programmed query language used to communicate with databases.</p> <p>Comparing Hive with Traditional Databases</p> <p>1. Schema Flexibility</p> <p>Hive: Hive provides schema-on-read, allowing users to define the structure of data during query execution. This flexibility is advantageous when dealing with unstructured or semi-structured data.</p> <p>2. Query Language</p> <p>Hive: Hive uses HQL, which closely resembles SQL. However, complex queries might perform slower due to the underlying MapReduce processing.</p> <p>3. Performance</p> <p>Hive: While Hive is scalable and suitable for large datasets, it may not match the real-time performance of traditional databases for ad-hoc queries.</p> <p>4. Data Processing Paradigm</p>		



Hive: Hive is well-suited for batch processing and data warehousing scenarios. Its performance shines in complex data transformations and analytics tasks.

5. Use Cases

Hive: Hive is ideal for scenarios involving large-scale data processing, log analysis, social media analytics, and other Big Data use cases.

b) What are the operators and functions in HIVE?

[L1][CO2]

[6M]

Arithmetic Operators in Hive

In Hive, the arithmetic operator accepts any numeric type. The commonly used arithmetic operators are: -

Backward Skip 10s Play Video Forward Skip 10s

Operators	Description
$A + B$	This is used to add A and B.
$A - B$	This is used to subtract B from A.
$A * B$	This is used to multiply A and B.
A / B	This is used to divide A and B and returns the quotient of the division.
$A \% B$	This returns the remainder of A / B .
$A B$	This is used to determine the bitwise OR of A and B.
$A \& B$	This is used to determine the bitwise AND of A and B.
$A \wedge B$	This is used to determine the bitwise XOR of A and B.
$\sim A$	This is used to determine the bitwise NOT of A.

6 a) Appraise about Hive query language?

[L4][CO5]

[6M]

Hive Query Language (HQL), often referred to as HiveQL, is a query language used in Apache Hive, a data warehouse infrastructure built on top of Hadoop. HQL enables users to write SQL-like queries to interact with and analyze large datasets stored in Hadoop's distributed file system.

The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore. This chapter explains how to use the SELECT statement with WHERE clause.

SELECT statement is used to retrieve the data from a table. WHERE clause works similar to a condition. It filters the data using the condition and gives you a finite result. The built-in operators and functions generate an expression, which fulfils the condition.

Syntax

Given below is the syntax of the SELECT query:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]
[LIMIT number];
```

Example

Let us take an example for SELECT...WHERE clause. Assume we have the employee table as given below, with fields named Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 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> SELECT * FROM employee WHERE salary>30000;
```

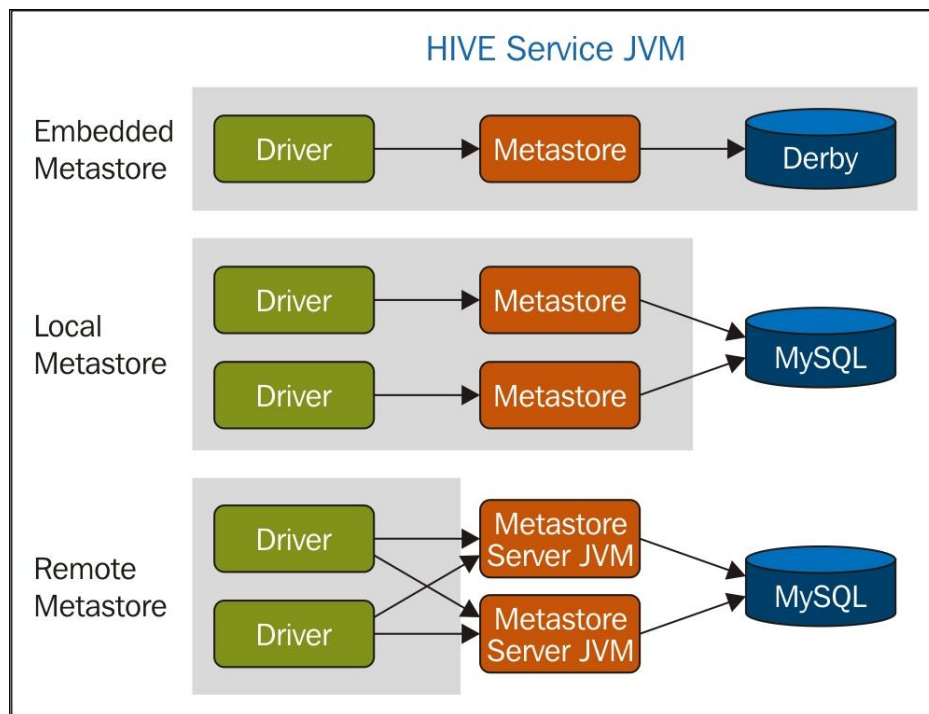
b) Review Metastore in Hive?

[L2][CO5]

[6M]

The *metastore* is the central repository of Hive metadata. The metastore is divided into two pieces: a service and the backing store for the data. By default, the metastore service runs in the same JVM as the Hive service and contains an embedded Derby

database instance backed by the local disk. This is called the *embedded metastore* configuration



machine. Any JDBC-compliant database may be used by setting the `javax.jdo.option`.

MySQL is a popular choice for the standalone metastore. In this case, the `javax.jdo.option.ConnectionURL` is set to `jdbc:mysql://host/dbname?createDatabaseIfNotExist=true`, and the `javax.jdo.option.ConnectionDriverName` is set to `com.mysql.jdbc.Driver`. (The username and password should be set, too, of course.) The JDBC driver JAR file for MySQL (Connector/J) must be on Hive's classpath, which is simply achieved by placing it in Hive's *lib* directory.

Going a step further, there's another metastore configuration called a *remote metastore*, where one or more metastore servers run in separate processes to the Hive service. This brings better manageability and security because the database tier can be completely firewalled off, and the clients no longer need the database credentials.

A Hive service is configured to use a remote metastore by setting `hive.meta store.local` to false and `hive.metastore.uris` to the metastore server URIs, separated by commas if there is more than one. Metastore server URIs are of the form `thrift://host:port`, where the port corresponds to the one set by `METASTORE_PORT` when starting the metastore server.

7 Differentiate Hbase over RDBMS.

[L4][CO1] [12M]

Difference between RDBMS and HBase:

S. No.	Parameters	RDBMS	HBase
1.	SQL	It requires SQL (Structured Query Language).	SQL is not required in HBase.

	2.	Schema	It has a fixed schema.	It does not have a fixed schema and allows for the addition of columns on the fly.	
	3.	Database Type	It is a row-oriented database	It is a column-oriented database.	
	4.	Scalability	RDBMS allows for scaling up. That implies, that rather than adding new servers, we should upgrade the current server to a more capable server whenever there is a requirement for more memory, processing power, and disc space.	Scale-out is possible using HBase. It means that, while we require extra memory and disc space, we must add new servers to the cluster rather than upgrade the existing ones.	
	5.	Nature	It is static in nature	Dynamic in nature	
	6.	Data retrieval	In RDBMS, slower retrieval of data.	In HBase, faster retrieval of data.	
	7.	Rule	It follows the ACID (Atomicity, Consistency, Isolation, and Durability) property.	It follows CAP (Consistency, Availability, Partition-tolerance) theorem.	
	8.	Type of data	It can handle structured data.	It can handle structured, unstructured as well as semi-structured data.	
	9.	Sparse data	It cannot handle sparse data.	It can handle sparse data.	
	10.	Volume of data	The amount of data in RDBMS is determined by the server's configuration.	In HBase, the amount of data depends on the number of machines deployed rather than on a single machine.	
	11.	Transaction Integrity	In RDBMS, mostly there is a guarantee	In HBase, there is no such guarantee	

			associated with transaction integrity.	associated with the transaction integrity.		
12.	Referential Integrity	Referential integrity is supported by RDBMS.	When it comes to referential integrity, no built-in support is available.			
13.	Normalize	In RDBMS, you can normalize the data.	The data in HBase is not normalized, which means there is no logical relationship or connection between distinct tables of data.			
14.	Table size	It is designed to accommodate small tables.. Scaling is difficult.	It is designed to accommodate large tables. HBase may scale horizontally.			

8 Explain with a neat diagram the architecture of Hbase.

[L2][CO2] [12M]

HBase architecture has 3 main components: HMaster, Region Server, Zookeeper.

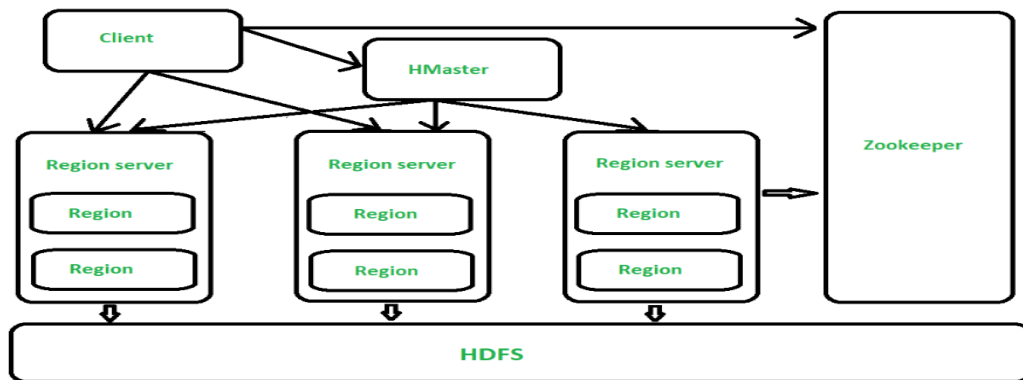


Figure – Architecture of HBase

All the 3 components are described below:

1. HMaster –

The implementation of Master Server in HBase is HMaster. It is a process in which regions are assigned to region server as well as DDL (create, delete table) operations. It monitor all Region Server instances present in the cluster. In a distributed environment, Master runs several background threads. HMaster has many features like controlling load balancing, failover etc.

2. Region Server –

HBase Tables are divided horizontally by row key range into Regions. Regions are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families. Region Server runs on HDFS DataNode which is present in Hadoop cluster. Regions of Region Server are responsible for several things, like handling, managing, executing as well as reads and writes HBase operations on that set of regions.

		<p>The default size of a region is 256 MB.</p> <p>3. Zookeeper – It is like a coordinator in HBase. It provides services like maintaining configuration information, naming, providing distributed synchronization, server failure notification etc. Clients communicate with region servers via zookeeper.</p> <p>Advantages of HBase –</p> <ol style="list-style-type: none"> 1. Can store large data sets 2. Database can be shared 3. Cost-effective from gigabytes to petabytes 4. High availability through failover and replication <p>Disadvantages of HBase –</p> <ol style="list-style-type: none"> 1. No support SQL structure 2. No transaction support 3. Sorted only on key 4. Memory issues on the cluster 		
9	a)	Categorize the joins in HiveQL	[L4][CO5]	[6M]
		<p>Type of Joins in Hive</p> <ol style="list-style-type: none"> 1. Inner join in Hive. 2. Left Outer Join in Hive. 3. Right Outer Join in Hive. 4. Full Outer Join in Hive. <p>JOIN</p> <p>JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.</p> <p>The following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:</p> <pre>hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);</pre> <p>On successful execution of the query, you get to see the following response:</p> <pre>+---+-----+---+-----+ ID NAME AGE AMOUNT </pre>		

3	kaushik	23	3000	
3	kaushik	23	1500	
2	Khilan	25	1560	
4	Chaitali	25	2060	

LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

The following query demonstrates LEFT OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
LEFT OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

The following query demonstrates RIGHT OUTER JOIN between the CUSTOMER and ORDER tables.

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM
CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON (c.ID =
o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME | AMOUNT | DATE |
+-----+-----+-----+-----+
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
+-----+-----+-----+-----+
```

FULL OUTER JOIN

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

The following query demonstrates FULL OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
FULL OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME | AMOUNT | DATE |
+-----+-----+-----+-----+
| 1 | Ramesh | NULL | NULL |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
| 5 | Hardik | NULL | NULL |
| 6 | Komal | NULL | NULL |
+-----+-----+-----+-----+
```

```

| 7 | Muffy | NULL | NULL |
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
+-----+-----+-----+-----+

```

b) Report the Implementation of queries on sorting and aggregation of data in Hive

[L6][CO3]

[6M]

By using HiveQL ORDER BY and SORT BY clause, we can apply sort on the column. It returns the result set either in ascending or descending order. Here, we are going to execute these clauses on the records of the below table:

emp

Id	Name	Salary	Department
1	Gaurav	30000	Developer
2	Aryan	20000	Manager
3	Vishal	40000	Manager
4	John	10000	Trainer
5	Henry	25000	Developer
6	William	9000	Developer
7	Lisa	25000	Manager
8	Ronit	20000	Trainer

HiveQL - ORDER BY Clause

In HiveQL, ORDER BY clause performs a complete ordering of the query result set. Hence, the complete data is passed through a single reducer. This may take much time in the execution of large datasets. However, we can use LIMIT to minimize the sorting time.

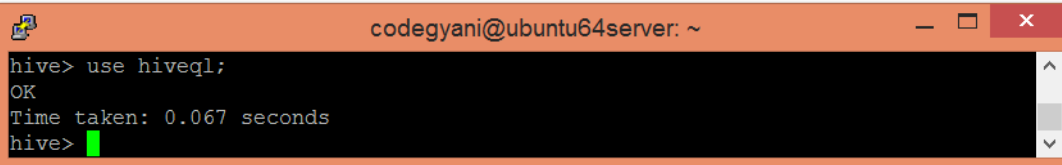
Example of ORDER BY Clause in Hive

Let's see an example to arrange the data in the sorted order by using ORDER BY clause.

ADVERTISEMENT
ADVERTISEMENT

- Select the database in which we want to create a table.

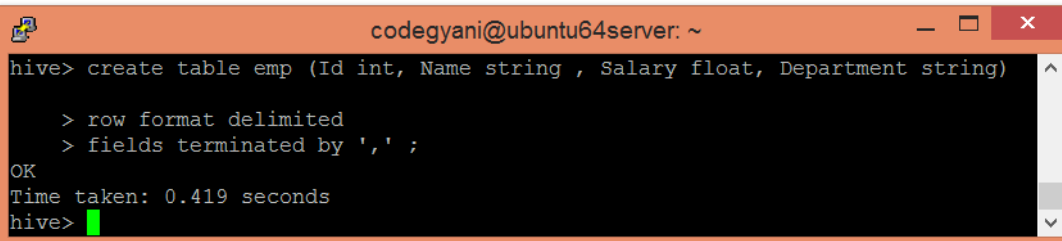
1. hive> use hiveql;



```
codegyani@ubuntu64server: ~
hive> use hiveql;
OK
Time taken: 0.067 seconds
hive>
```

- Now, create a table by using the following command:

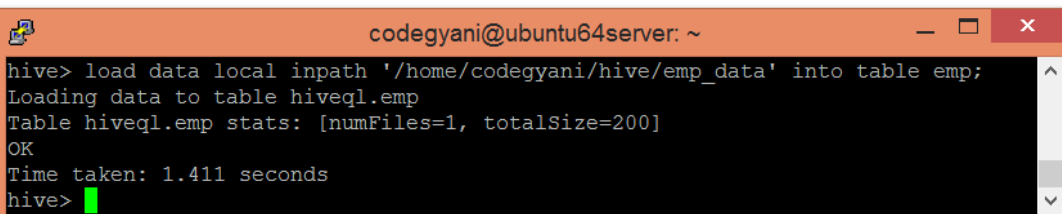
- hive> create table emp (Id int, Name string , Salary float, Department string)
- row format delimited
- fields terminated by ',' ;



```
codegyani@ubuntu64server: ~
hive> create table emp (Id int, Name string , Salary float, Department string)
> row format delimited
> fields terminated by ',' ;
OK
Time taken: 0.419 seconds
hive>
```

- Load the data into the table.

- hive> load data local inpath '/home/codegyani/hive/emp_data' into table emp ;



```
codegyani@ubuntu64server: ~
hive> load data local inpath '/home/codegyani/hive/emp_data' into table emp;
Loading data to table hiveql.emp
Table hiveql.emp stats: [numFiles=1, totalSize=200]
OK
Time taken: 1.411 seconds
hive>
```

- Now, fetch the data in the descending order by using the following command:

- hive> select * from emp order by salary desc;

10 a) Explain about IBM Big SQL?

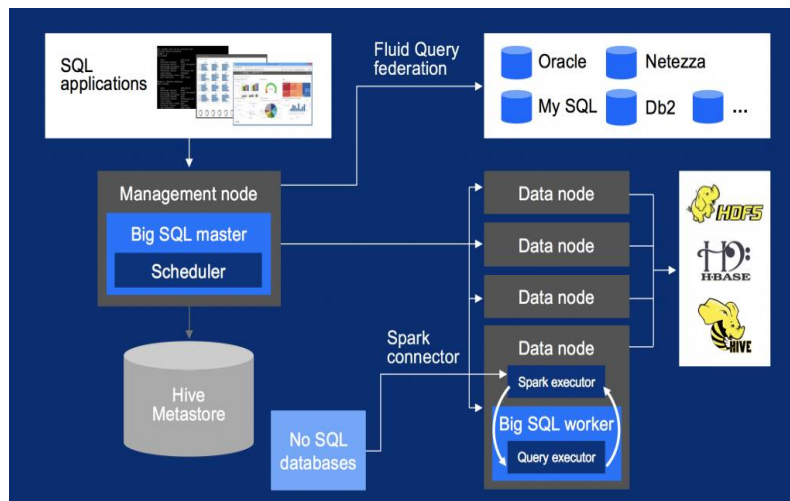
[L2][CO6]

[6M]

With Big SQL, your organization can derive significant value from your enterprise data. IBM Big SQL is a high performance massively parallel processing (MPP) SQL engine for Hadoop that makes querying enterprise data from across the organization an easy and secure experience. A Big SQL query can quickly access a variety of data sources including HDFS, RDBMS, NoSQL databases, object stores, and WebHDFS by using a single database connection or single query for best-in-class analytic capabilities.

It looks like to provide tools to help you manage your system and your databases, and you can use popular analytic tools to visualize your data.

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



Big SQL provides world-class scalability and performance, and supports the following key use cases:

Enterprise Data Warehouse (EDW) offloading

Big SQL understands commonly-used SQL syntax from other vendors and producers. You can offload and consolidate old data more quickly and easily from existing Oracle, IBM® Db2®, and IBM Netezza® enterprise data warehouses or data marts while preserving most of the SQL from those platforms.

Federated access to relational data

For data that can't be moved to Hadoop, Big SQL provides federated access to many relational database management system (RDBMS) sources outside of Hadoop with IBM Fluid Query technology and NoSQL databases with the use of Spark connectors. You can use a single database connection to access data across Hadoop and dozens of relational/NoSQL database types, whether they are on the cloud, on local systems, or both. Wherever the data resides, Big SQL offers data virtualization and enables querying disparate sources in a single query.

Big SQL also boasts:

- Elastic boost technology to support more granular resource usage and increase performance without increasing memory or CPU
- High-performance scans, inserts, updates, and deletes
- Deeper integration with Spark 2.1 than other SQL-on-Hadoop technologies
- Machine learning or graph analytics with Spark with a single security model
- Open Data Platform initiative (ODPi) compliance
- Advanced, ANSI-compliant SQL queries

b) Assess how HBase is implemented at Streamy.com

[L4][CO6]

[6M]

Streamy.com is a real-time news aggregator and social sharing platform. With a broad feature set, we started out with a complex implementation on top of PostgreSQL. It's a terrific product with a great community and a beautiful code base. We tried every trick in the book to keep things fast as we scaled, going so far as to modify the PostgreSQL code directly to suit our needs. Originally taking advantage of all RDBMS goodies, we found that eventually, one by one, we had to let them all go. Along the way, our entire team became the DBA.

We did manage to solve many of the issues that we ran into, but there were two that eventually led to the decision to find another solution from outside the world of RDBMS. Streamy crawls thousands of RSS feeds and aggregates hundreds of millions of items from them. In addition to having to store these items, one of our more complex queries reads a time-ordered list of all items from a set of sources. At the high end, this can run to several thousand sources and all of their items all in a single query.

Very large items tables At first, this was a single items table, but the high number of secondary indexes made inserts and updates very slow. We started to divide items up into several one-to-one link tables to store other information, separating static fields from dynamic ones, grouping fields based on how they were queried, and denormalizing everything along the way. Even with these changes, single updates required rewriting the entire record, so tracking statistics on items was difficult to scale. The rewriting of records and having to update indexes along the way are intrinsic properties of the RDBMS we were using. They could not be decoupled. We partitioned our tables, which was not too difficult because of the natural partition of time, but the complexity got out of hand fast. We needed another solution!

Very large sort merges

Performing sorted merges of time-ordered lists is common in many Web 2.0 applications. An example SQL query might look like this:

```
SELECT id, stamp, type FROM streams
WHERE type IN ('type1','type2','type3','type4',..., 'typeN')
ORDER BY stamp DESC LIMIT 10 OFFSET 0;
```

Assuming id is a primary key on streams, and that stamp and type have secondary indexes, an RDBMS query planner treats this query as follows:

```
MERGE (
SELECT id, stamp, type FROM streams
WHERE type = 'type1' ORDER BY stamp DESC,
...,
SELECT id, stamp, type FROM streams
WHERE type = 'typeN' ORDER BY stamp DESC
) ORDER BY stamp DESC LIMIT 10 OFFSET 0;
```

The problem here is that we are after only the top 10 IDs, but the query planner actually materializes an entire merge and then limits at the end. A simple heap sort across each of the types would allow you to “early out” once you have the top 10. In our case, each type could have tens of thousands of IDs in it, so materializing the entire list and sorting it was extremely slow and unnecessary. We actually went so far as to write a custom PL/Python script that performed a heap sort using a series of queries like the following:

```
SELECT id, stamp, type FROM streams
WHERE type = 'typeN'
ORDER BY stamp DESC LIMIT 1 OFFSET 0;
```

If we ended up taking from typeN (it was the next most recent in the heap), we would run another query:

```
SELECT id, stamp, type FROM streams
WHERE type = 'typeN'
```

ORDER BY stamp DESC LIMIT 1 OFFSET 1;

In nearly all cases, this outperformed the native SQL implementation and the query planner's strategy. In the worst cases for SQL, we were more than an order of magnitude faster using the Python procedure. We found ourselves continually trying to outsmart the query planner.

Again, at this point, we really needed another solution.

Life with HBase

Our RDBMS-based system was always capable of correctly implementing our requirements; the issue was scaling. When you start to focus on scale and performance rather than correctness, you end up shortcutting and optimizing for your domain-specific use cases everywhere possible. Once you start implementing your own solutions to your data problems, the overhead and complexity of an RDBMS gets in your way. The abstraction from the storage layer and ACID requirements are an enormous barrier and luxury that you cannot always afford when building for scale.

HBase is a distributed, column-oriented, sorted map store and not much else. The only major part that is abstracted from the user is the distribution, and that's exactly what we don't want to deal with. Business logic, on the other hand, is very specialized and optimized. With HBase not trying to solve all of our problems, we've been able to solve them better ourselves and rely on HBase for scaling our storage, not our logic. It was an extremely liberating experience to be able to focus on our applications and logic rather than the scaling of the data itself.

We currently have tables with hundreds of millions of rows and tens of thousands of columns; the thought of storing billions of rows and millions of columns is exciting, not scary.

Prepared by:Dr.J.Sridhar,Mrs.R.Surekha