CHAPTER:1

Introduction to Data Science and Data Analytics: Data Science and Data Analytics are two closely related fields that deal with extracting valuable insights and knowledge from data to inform decision-making and solve complex problems. These disciplines have gained immense popularity in recent years due to the exponential growth of data and the increasing importance of data-driven decision-making in various industries.

Data Science is an interdisciplinary field that combines techniques from mathematics, statistics, computer science, and domain expertise to extract knowledge and insights from data. The primary goal of data science is to discover meaningful patterns, relationships, and trends in large and complex datasets. Data scientists use various tools and techniques to analyze and interpret data, build predictive models, and create data-driven solutions.

The typical data science workflow includes:
Data Collection: Gathering relevant data from various sources, including databases, APIs, websites, and sensors.
Data Cleaning: Preprocessing and cleaning the data to remove errors, inconsistencies, and missing values.
Data Exploration: Exploring the data to gain an understanding of its characteristics and identify potential patterns.
Data Analysis: Applying statistical and machine learning techniques to extract insights and draw conclusions from the data.
Model Building: Developing predictive or descriptive models to solve specific problems or make forecasts.
Model Evaluation: Assessing the performance of the models to ensure their accuracy and effectiveness.
Visualization: Communicating the findings through visualizations and reports to make the insights more understandable and actionable.

Data Analytics is a subset of data science that focuses on the exploration, interpretation, and communication of data patterns to guide business decisions and improve performance. It involves the use of various analytical techniques to gain insights from data, turning raw data into actionable information. Data analytics plays a vital role in aiding businesses in understanding customer behavior, optimizing marketing strategies, improving operational efficiency, and making informed decisions.

Data analytics can be broadly categorized into three types:
Descriptive Analytics: Describes past events and provides an overview of historical data to understand what happened and why.
Predictive Analytics: Uses historical data to build models and make predictions about future outcomes.
Prescriptive Analytics: Suggests actions to optimize outcomes based on the insights obtained from descriptive and predictive analytics

Data Science - Key Components:
Data science is a field that involves working with data to gain insights and solve problems. It consists of several key components that are essential for its functioning:
1. Data Collection: This is the first step where data scientists gather information from various sources, like databases, sensors, or surveys.
2. Data Cleaning and Preprocessing: Raw data often has errors or missing values. Data scientists clean and prepare the data to make it suitable for analysis.
3. Data Exploration and Visualization: Data scientists explore the data to find patterns and trends. They use charts and graphs to visualize the data for better understanding.
4. Statistical Analysis: Data science uses statistical methods to draw meaningful conclusions from the data.
5. Machine Learning: It is a technique where computer programs learn from data and make predictions or decisions without explicit programming.
6. Data Modeling: Data scientists create models to represent relationships within the data and make predictions.
7. Feature Engineering: This involves selecting and transforming relevant data features to improve model performance.
8. Model Evaluation and Validation: Data scientists assess model accuracy and reliability using various techniques.
9. Deployment and Integration: Implementing data science solutions into real-world applications or systems.
10. Ethics and Privacy: Data scientists must consider ethical implications and privacy concerns when dealing with data.

Big Data and its importance: https://youtu.be/I_ku0D4uQzQ

Why is big data analytics important?

In today's world, Big Data analytics is fueling everything we do online—in every industry.

Take the music streaming platform Spotify for example. The company has nearly 96 million users that generate a tremendous amount of data every day. Through this information, the cloud-based platform automatically generates suggested songs—through a smart recommendation engine—based on likes, shares, search history, and more. What enables this is the techniques, tools, and frameworks that are a result of Big Data analytics.

Importance of Big data

Big Data importance doesn't revolve around the amount of data a company has. Its importance lies in the fact that how the company utilizes the gathered data.

Every company uses its collected data in its own way. More effectively the company uses its data, more rapidly it grows.

The companies in the present market need to collect it and analyze it because:

1. Cost Savings: Big Data tools like Apache Hadoop, Spark, etc. bring cost-saving benefits to businesses when they have to store large amounts of data. These tools help organizations in identifying more effective ways of doing business.

2. Time-Saving: Real-time in-memory analytics helps companies to collect data from various sources. Tools like Hadoop help them to analyze data immediately thus helping in making quick decisions based on the learnings.

3. Understand the market conditions: Big Data analysis helps businesses to get a better understanding of market situations.

For example, analysis of customer purchasing behavior helps companies to identify the products sold most and thus produces those products accordingly. This helps companies to get ahead of their competitors.

4. Social Media Listening: Companies can perform sentiment analysis using Big Data tools. These enable them to get feedback about their company, that is, who is saying what about the company.

Companies can use Big data tools to improve their online presence.

5. Boost Customer Acquisition and Retention: Customers are a vital asset on which any business depends on. No single business can achieve its success without building a robust customer base. But even with a solid customer base, the companies can't ignore the competition in the market.

If we don't know what our customers want then it will degrade companies' success. It will result in the loss of clientele which creates an adverse effect on business growth.

Big data analytics helps businesses to identify customer related trends and patterns. Customer behavior analysis leads to a profitable business.

6. Solve Advertisers Problem and Offer Marketing Insights: Big data analytics shapes all business operations. It enables companies to fulfill customer expectations. Big data analytics helps in changing the company's product line. It ensures powerful marketing campaigns.

7. The driver of Innovations and Product Development: Big data makes companies capable to innovate and redevelop their products.

Four Vs: https://youtu.be/ioBHS7A9IxY

1) Volume: Amount of data generated per second, It refers to the size of Big Data. Data can be considered Big Data or not is based on the volume. The rapidly increasing volume data is due to cloud-computing traffic, IoT, mobile traffic etc.

2) Velocity: It refers to the speed at which the data is being transferred. This is mainly due to IoTs, mobile data, social media etc.In the year 2000, Google was receiving 32.8 million searches per day. As for 2018, Google was receiving 5.6 billion searches per day!

3) Variety: It refers to Structured, Semi-structured and Unstructured data due to different sources of data generated either by humans or by machines.

4) Veracity: Refers to Trustworthiness of data. It refers to the assurance of quality/integrity/credibility/accuracy of the data. Since the data is collected from multiple sources, we need to check the data for accuracy before using it for business insights.

5) Value:Just because we collected lots of Data, it's of no value unless we garner some insights out of it. Value refers to how useful the data is in decision making. We need to extract the value of the Big Data using proper analytics.

Types of data under big data: https://youtu.be/IAU-bXjjvyI

Structured data: It's the traditional data which is organized and conforms to the formal structure of data. This data can be stored in a relational database. Example: Bank statement containing date, time, amount etc.

Semi-structured data: It's semi-organized data. It doesn't conform to the formal structure of data. Example: Log files, JSON files, Sensor data, csv files etc. It's more flexible than structured data.

Unstructured data: It's not an organized data and doesn't fit into rows and columns structure of a relational database. Example: Text files, Emails, images, videos, voicemails, audio files etc. Unstructured data has no predefined structure, making it the most challenging to work with.

Diff between Structured , Semi Structured and Unstructured data :

https://byjus.com/gate/difference-between-structured-semi-structured-and-unstructured-data/

Drivers for Big data:

Big data has become increasingly important in today's world due to several key drivers that push for its adoption:

1. Data Explosion: The rise of the internet, social media, and digital devices has led to an enormous amount of data being generated every day. This data overload requires advanced technologies to manage and analyze it effectively.
2. Better Decision Making: Organizations want to make smarter decisions based on data insights rather than relying on intuition alone. Big data analytics provides the tools to extract valuable information from vast datasets, helping businesses make more informed choices.
3. Personalized Experiences: Customers expect personalized experiences from companies. Big data allows businesses to understand individual preferences and tailor products and services accordingly, leading to improved customer satisfaction.
4. Staying Ahead of Competitors: Companies that embrace big data gain a competitive edge. By analyzing market trends and consumer behavior, businesses can adapt quickly to changing demands and outperform their rivals.
5. Cost Savings and Efficiency: Big data analytics can identify inefficiencies and optimize processes, leading to cost savings and improved operational efficiency.
6. Real-Time Insights: Quick access to real-time data enables organizations to respond swiftly to market changes and make timely decisions.
7. Advancing Healthcare and Research: Big data plays a significant role in medical research, disease prediction, and personalized treatment, leading to advancements in healthcare.
8. IoT Impact: The Internet of Things (IoT) generates vast amounts of data from interconnected devices. Big data technologies are necessary to make sense of this data and derive valuable insights.
9. Social Media Influence: Big data analytics helps businesses understand public sentiment and opinions on social media platforms, influencing marketing strategies and brand perception.
10. Fraud Detection and Security: In the financial sector, big data analytics is essential for detecting and preventing fraudulent activities, ensuring security and safeguarding financial interests.

Big Data Applications:

Big data finds numerous applications across various industries, enabling businesses and organizations to make data-driven decisions and derive valuable insights. Some of the key big data applications are as follows:

1. E-commerce and Retail: Big data helps retailers analyze customer behavior, preferences, and purchase patterns to offer personalized product recommendations and targeted marketing campaigns, enhancing the overall shopping experience.
2. Healthcare: In healthcare, big data facilitates medical research, disease prediction, and personalized treatment plans by analyzing vast amounts of patient data and genomic information.
3. Financial Services: Big data is used for fraud detection, risk assessment, and real-time transaction monitoring, ensuring the security and integrity of financial transactions.
4. Marketing and Advertising: Big data analytics allows marketers to understand consumer trends, sentiment, and social media interactions, enabling them to design effective marketing strategies and measure campaign success.
5. Transportation and Logistics: Big data optimizes logistics operations, route planning, and vehicle maintenance by analyzing real-time data from sensors and GPS devices.
6. Manufacturing: Big data aids manufacturers in optimizing production processes, predicting maintenance needs, and reducing downtime through predictive analytics.
7. Telecommunications: Telecommunication companies use big data to analyze network performance, customer behavior, and call data to improve network efficiency and customer service.

8. Energy and Utilities: Big data helps energy companies optimize energy consumption, predict equipment failures, and manage renewable energy resources effectively.
9. Government and Public Services: Big data is utilized in areas like urban planning, disaster response, and healthcare management to improve public services and policy-making.
10. Social Media and Online Platforms: Social media platforms leverage big data to analyze user interactions, preferences, and trends to enhance user experiences and provide personalized content.

Traditional vs Big Data Systems https://youtu.be/UUMGfThF-D0

Traditional Data	Big Data
Traditional data is generated in enterprise level.	Big data is generated outside the enterprise level.
Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
Traditional database system deals with structured data.	Big data system deals with structured, semi-structured,database, and unstructured data.
Traditional data is generated per hour or per day or more.	But big data is generated more frequently mainly per seconds.
Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.
Data integration is very easy.	Data integration is very difficult.
Normal system configuration is capable to process traditional data.	High system configuration is required to process big data.
The size of the data is very small.	The size is more than the traditional data size.
Traditional data base tools are required to perform any data base operation.	Special kind of data base tools are required to perform any databaseschema-based operation.
Normal functions can manipulate data.	Special kind of functions can manipulate data.
Its data model is strict schema based and it is static.	Its data model is a flat schema based and it is dynamic.
Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.
Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.
It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
Its data sources includes ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

CHAPTER:2

https://drive.google.com/file/d/1cyoFcPJ0QOPhi4y9c4pbm--nmOmOfyZM/view?usp=sharing

STATISTICAL INFERENCE: It refers to the process of drawing conclusions, making predictions, or making decisions about a population based on a sample of data. In other words, it involves using sample data to make inferences or generalizations about a larger population from which the sample is drawn.

Probability Distributions: a probability distribution refers to a mathematical function or model that describes the likelihood of various outcomes or events occurring in a specific data or random variable. Probability distributions are fundamental tools used to analyze, understand, and make predictions about data and uncertainty.

Event Space: In data science and probability theory, the term "event space" refers to the set of all possible outcomes or events that can occur in a particular experiment or random process. It is a fundamental concept used to describe and analyze uncertainty and randomness. The event space is often denoted as Ω (the Greek letter omega)

CHAPTER:3 DATA PRE-PROCESSING AND DATA VISUALIZATION

Dataset: A dataset is a structured collection of data that is organized and stored in a way that allows for efficient retrieval, analysis, and management. Datasets can take various forms and can consist of many types of data, such as text, numbers, images, videos, and more

Types of Dataset:

Numerical Datasets: The numerical data set is a data set, where the data are expressed in numbers rather than natural language. The numerical data is sometimes called quantitative data.

Examples include the number of pages in a book, heights and weights

Bivariate Datasets: A data set that has two variables is called a Bivariate data set. It deals with the relationship between the two variables. Bivariate dataset usually contains two types of related data.
For example, a business might track the amount of time a customer spends in the store and their average purchase amount. By gathering data on multiple clients regarding these two variables, they can create a bivariate data set and analyze the relationship between the amount of time clients' spent in the store and the amount of goods they purchased.
Multivariate Datasets: A data set with multiple variables. When the dataset contains three or more than three data types (variables), then the data set is called a multivariate dataset.
For example, a home renovation company measures the dimensions of multiple rooms in a house and creates a multivariate data set sharing the height, width, and length of each room. They use this data set to estimate material costs or offer a quote to a client.
Categorical Datasets: Categorical data sets divide information into distinct groups. Categorical data sets represent features or characteristics of a person or an object. The categorical dataset consists of a categorical variable also called the qualitative variable, that can take exactly two values.
Example: A person's gender (male or female)
Correlation Datasets: The set of values that demonstrate some relationship with each other indicates correlation data sets. Here the values are found to be dependent on each other.
For example, a data set containing the height and weight of multiple people is often a correlation data set, as someone's weight is likely to increase according to their height.

Importance of Pre-processing the Data:

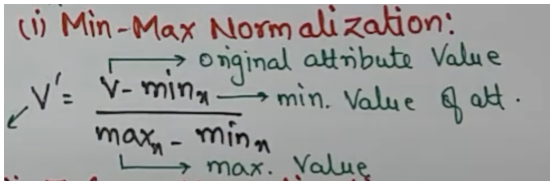
It improves accuracy and reliability: Preprocessing data removes missing or inconsistent data values resulting from human or computer error, which can improve the accuracy and quality of a dataset, making it more reliable.
It makes data consistent: When collecting data, it's possible to have data duplicates, and discarding them during preprocessing can ensure the data values for analysis are consistent, which helps produce accurate results.
It increases the data's algorithm readability: Preprocessing enhances the data's quality and makes it easier for machine learning algorithms to read, use, and interpret it.

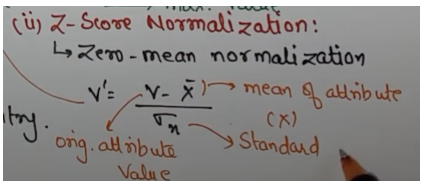
Data Cleaning: https://youtu.be/m9aHlwGYi7A?si=lFomlz4FCv7gKjxW
https://youtu.be/Rt0bmS7crNU?si=a1WzW3GZMeJ1dWqp
Steps of Data cleaning:https://www.tableau.com/learn/articles/what-is-data-cleaning
Data cleansing, also referred to as data cleaning or data scrubbing, is the process of fixing incorrect, incomplete, duplicate or otherwise erroneous data in a data set. It involves identifying data errors and then changing, updating or removing data to correct them.
Steps: Step 1: Remove duplicate or irrelevant observations
Step 2: Fix structural errors
Step 3: Filter unwanted outliers
Step 4: Handle missing data
Step 5: Validate and QA
Advantages of Data Cleaning: Improved decision making , Boost results and revenue, Save money and reduce waste, Save time and increase productivity

Data Integration and Transformation:

Data Integration: https://youtu.be/UKUq7hZdZUw?si=th6NdfzFWVmeub-f
There are mainly 2 major approaches for data integration:
1. Tight Coupling: This approach involves creating a centralized repository or data warehouse to store the integrated data. The data is extracted from various sources, transformed and loaded into a data warehouse. Data is integrated in a tightly coupled manner, meaning that the data is integrated at a high level, such as at the level of the entire dataset or schema. This approach is also known as data warehousing, and it enables data consistency and integrity, but it can be inflexible and difficult to change or update.
2. Loose Coupling: This approach involves integrating data at the lowest level, such as at the level of individual data elements or records. Data is integrated in a loosely coupled manner, meaning that the data is integrated at a low level, and it allows data to be integrated without having to create a central repository or data warehouse. This approach is also known as data federation, and it enables data flexibility and easy updates, but it can be difficult to maintain consistency and integrity across multiple data sources.

Data Transformation: https://youtu.be/RQ0I1u-q8N8?si=Wdk7dECslWj_yriK ←***(SUM Of Z-Score and Min-Max)***





$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

'u' = mean

Data Normalization one more Technique:

Decimal Scaling: https://youtu.be/OLaotS4mY50?si=IQHEICUDin5AxXr0

$$v_i' = \frac{v_i}{10^j}$$

Example: Let the input data is: -10, 201, 301, -401, 501, 601, 701 To normalize the above data, Step 1: Maximum absolute value in given data(m): 701 Step 2: Divide the given data by 1000 (i.e j=3) Result: The normalized data is: -0.01, 0.201, 0.301, -0.401, 0.501, 0.601, 0.701

Advantages of Data Transformation: Improves Data Quality, Facilitates Data Integration, Improves Data Analysis, Increases Data Security, Enhances Data Mining Algorithm Performance

Disadvantages: Time-consuming, High Cost, Sometimes during data transformation might be data loss happens, Data transformation can be a complex process

Data Reduction: https://youtu.be/0l3tzQI_ApM?si=MlC8DOofU6kBBjQA
https://youtu.be/rrfyq5IhcB0?si=Fg-QuDX_Gex0F6mt
https://www.geeksforgeeks.org/data-reduction-in-data-mining/

Data reduction is a technique used in data mining to reduce the size of a dataset while still preserving the most important information. This can be beneficial in situations where the dataset is too large to be processed efficiently, or where the dataset contains a large amount of irrelevant or redundant information.

There are several different data reduction techniques that can be used in data mining, including:

Data Sampling: This technique involves selecting a subset of the data to work with, rather than using the entire dataset. This can be useful for reducing the size of a dataset while still preserving the overall trends and patterns in the data.

Dimensionality Reduction: This technique involves reducing the number of features in the dataset, either by removing features that are not relevant or by combining multiple features into a single feature.

Data Compression: This technique involves using techniques such as lossy or lossless compression to reduce the size of a dataset.

Data Discretization: This technique involves converting continuous data into discrete data by partitioning the range of possible values into intervals or bins.

Feature Selection: This technique involves selecting a subset of features from the dataset that are most relevant to the task at hand.

It's important to note that data reduction can have a trade-off between the accuracy and the size of the data. The more data is reduced, the less accurate the model will be and the less generalizable it will be.

Methods of data reduction:

1. Data Cube Aggregation: This technique is used to aggregate data in a simpler form. For example, imagine the information you gathered for your analysis for the years 2012 to 2014, that data includes the revenue of your company every three months. They involve you in the annual sales, rather than the quarterly average, So we can summarize the data in such a way that the resulting data summarizes the total sales per year instead of per quarter. It summarizes the data.

2. Dimension reduction: Whenever we come across any data which is weakly important, then we use the attribute required for our analysis. It reduces data size as it eliminates outdated or redundant features.

Step-wise Forward Selection – The selection begins with an empty set of attributes later on we decide the best of the original attributes on the set based on their relevance to other attributes. We know it as a p-value in statistics.

Step-wise Backward Selection – This selection starts with a set of complete attributes in the original data and at each point, it eliminates the worst remaining attribute in the set.

3. Data Compression: The data compression technique reduces the size of the files using different encoding mechanisms (Huffman Encoding & run-length Encoding). We can divide it into two types based on their compression techniques.

Lossless Compression – Encoding techniques (Run Length Encoding) allow a simple and minimal data size reduction. Lossless data compression uses algorithms to restore the precise original data from the compressed data.

Lossy Compression – Methods such as the Discrete Wavelet transform technique, PCA (principal component analysis) are examples of this compression. For e.g., the JPEG image format is a lossy compression, but we can find the meaning equivalent to the original image. In lossy-data compression, the decompressed data may differ from the original data but are useful enough to retrieve information from them.
4. Numerosity Reduction: In this reduction technique, the actual data is replaced with mathematical models or smaller representations of the data instead of actual data, it is important to only store the model parameter. Or non-parametric methods such as clustering, histogram, and sampling.
5. Discretization & Concept Hierarchy Operation: Techniques of data discretization are used to divide the attributes of the continuous nature into data with intervals. We replace many constant values of the attributes by labels of small intervals. This means that mining results are shown in a concise, and easily understandable way.
Top-down discretization – If you first consider one or a couple of points (so-called breakpoints or split points) to divide the whole set of attributes and repeat this method up to the end, then the process is known as top-down discretization also known as splitting.
Bottom-up discretization – If you first consider all the constant values as split points, some are discarded through a combination of the neighborhood values in the interval, that process is called bottom-up discretization.
Concept Hierarchies: It reduces the data size by collecting and then replacing the low-level concepts (such as 43 for age) with high-level concepts (categorical variables such as middle age or Senior).
For numeric data following techniques can be followed:
Binning – Binning is the process of changing numerical variables into categorical counterparts. The number of categorical counterparts depends on the number of bins specified by the user.
Histogram analysis – Like the process of binning, the histogram is used to partition the value for the attribute X, into disjoint ranges called brackets. There are several partitioning rules:
Equal Frequency partitioning: Partitioning the values based on their number of occurrences in the data set.
Equal Width Partitioning: Partitioning the values in a fixed gap based on the number of bins i.e. a set of values ranging from 0-20.
Clustering: Grouping similar data together.

Data Discretization and Concept Hierarchy Generation: https://youtu.be/KFE6vLE1Xtg?si=Z5vacP9fSDGW9RyZ
Data discretization: Data discretization is the process of converting continuous or numerical data into discrete intervals or categories. It simplifies data analysis and makes it suitable for certain types of algorithms. In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss.
There are two forms of data discretization:
Supervised Discretization - This discretization process uses class information.
Unsupervised Discretization - This discretization process does not use class information.

Discretization techniques can be categorized based on which direction it proceeds as follows:
Top-Down Discretization: Starts from the top (the whole dataset) and divides it into smaller parts. Typically begins with one big interval and recursively splits it into smaller intervals based on certain criteria or rules. Commonly used with decision tree-based methods. Example: Start with the age range of 0-100 and split it into child (0-18) and adult (19-100). Then further split adults into young adults (19-35) and middle-aged (36-100).
Bottom-Up Discretization: Begins from the bottom (individual data points) and groups them into larger intervals.
Starts with individual data points and clusters them together to form intervals based on certain criteria or similarity measures. Can be used with clustering algorithms. Example: Start with individual data points for ages and group them into clusters such as "children," "young adults," and "seniors" based on similarity in age.
In summary, top-down discretization starts with the entire dataset and breaks it down into smaller intervals, while bottom-up discretization begins with individual data points and groups them into larger intervals based on similarity. The choice between these methods depends on the specific data and analysis requirements.

Concept Hierarchies: Concept hierarchy generation is the process of organizing categorical data into a hierarchy of concepts or levels of abstraction. It helps in understanding the relationships between different categories and their attributes.
Concept hierarchies can be used to reduce the data by collecting and replacing low-level concepts with higher-level concepts.Data mining on a reduced data set means fewer input and output operations and is more efficient than mining on a larger data set.Because of these benefits, discretization techniques and concept hierarchies are typically applied before data mining, rather than during mining.

Typical Methods of Discretization and Concept Hierarchy Generation for Numerical Data:

1. Binning : Binning is a top-down splitting technique based on a specified number of bins. Binning is an unsupervised discretization technique because it does not use class information. In this, The sorted values are distributed into several buckets or bins and then replaced with each bin value by the bin mean or median. It is further classified into
Equal-width (distance) partitioning , Equal-depth (frequency) partitioning
2. Histogram Analysis: It is an unsupervised discretization technique because histogram analysis does not use class information.In this technique, a histogram is constructed to represent the data distribution. Peaks in the histogram can be considered as bin boundaries.It is also further classified into Equal-width histogram, Equal frequency histogram
3. Cluster Analysis: Cluster analysis methods like k-means clustering can be used to group similar data points into clusters.Each cluster can then be treated as a discrete category.This technique is particularly useful when the data distribution is not uniform.

Data visualization techniques: https://youtu.be/leXt9btuyEY?si=8MxwYyPtogzlb6hS
https://youtu.be/gFY61CH8lnQ?si=oxH03fRI1Loi4LPJ
https://youtu.be/XJ6-xKGhzWE?si=-gz3YQqukeNkfRaW
https://youtu.be/b_Pkbwes-So?si=ZVdBqY3EvDb1D_bY

CHAPTER:4 INTRODUCTION TO MAP-REDUCE AND HADOOP ARCHITECTURE

Big Data Apache Hadoop & Hadoop EcoSystem, Moving Data in and out of Hadoop Understanding inputs and outputs of MapReduce: https://youtube.com/playlist?list=PLxCzCOWd7aiHUUi6ZlansKbDw_cXut0El *(Lecture 2 to 4)*
Features of Hadoop: https://youtu.be/IDTGrlaYYEU
Hadoop Architecture: https://youtu.be/7bJTDRhJp3I

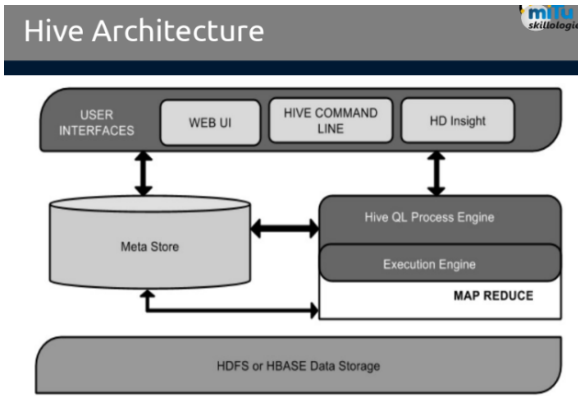
Data serialization in Hadoop refers to the process of converting complex data structures, such as objects or records, into a format that can be easily stored, transmitted, and reconstructed when needed
When data is emitted from a MapReduce Mapper or Reducer, it needs to be converted into a serialized format before being stored or sent to the next stage of processing. Avro allows you to define a schema for the data (in JSON format), which specifies the structure and types of the data elements. Then, the data is serialized into a compact binary format based on the defined schema.
There are various data serialization formats used in Hadoop, including:
Avro: Avro is a data serialization framework that provides a compact, fast, and schema-based data serialization format. It allows data with complex structures to be serialized and deserialized efficiently.
Serialization in Hadoop is crucial for efficient data storage and processing, as it reduces data size, improves performance, and allows for seamless data interchange between different components of the Hadoop ecosystem

CHAPTER:5 HDFS, HIVE AND HIVEQL, HBASE

HDFS-Overview and Architecture: https://youtu.be/VavRifvVwIo or https://youtu.be/oWefH_4ftUY
Hadoop File System was developed using distributed file system design. It is suitable for the distributed storage and processing.
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines.
HDFS also makes applications available to parallel processing. Hadoop provides a command interface to interact with HDFS.
Core Component of HDFS:
1. Name Node: It is the master daemon that maintains and manages the DataNodes (slave nodes).
- It records the metadata of all the blocks stored in the cluster, e.g. location of blocks stored, size of the files, permissions, hierarchy, etc.
- It records each and every change that takes place to the file system metadata. If a file is deleted in HDFS, the NameNode will immediately record this in the EditLog
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are alive
- It keeps a record of all the blocks in the HDFS and DataNode in which they are stored
2. Data Node: The actual data is stored on DataNodes.
- It is responsible for serving read and write requests from the clients.
- It sends heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.
3.Secondary Node: The Secondary NameNode works concurrently with the primary NameNode as a helper daemon/process.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

Shell commands: https://www.geeksforgeeks.org/hdfs-commands/
Hive Architecture and Installation: https://youtu.be/taTfW2kXSoE
- It is highly scalable because it supports real-time processing and batch processing.Initially Hive was developed by Facebook.
- In addition, Hive supports all SQL data types, making query processing easier.
- Similar to the query processing framework, HIVE also has two components: JDBC driver and HIVE Command-Line.
- JDBC is used with the ODBC driver to set up connection and data storage permissions whereas HIVE command line facilitates query processing.
- its query language is called as HQL (Hive Query Language). Hive+SQL=HQL.

Architecture:



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

Comparison with Traditional Database:
RDBMS vs Hadoop

S.No.	RDBMS	Hadoop
1.	Traditional row-column based databases, basically used for data storage, manipulation and retrieval.	An open-source software used for storing data and running applications or processes concurrently.
2.	In this structured data is mostly processed.	In this both structured and unstructured data is processed.
3.	It is best suited for OLTP environment.	It is best suited for BIG data.
4.	It is less scalable than Hadoop.	It is highly scalable.
5.	Data normalization is required in RDBMS.	Data normalization is not required in Hadoop.
6.	It stores transformed and aggregated data.	It stores huge volume of data.
7.	It has no latency in response.	It has some latency in response.
8.	The data schema of RDBMS is static type.	The data schema of Hadoop is dynamic type.
9.	High data integrity available.	Low data integrity available than RDBMS.
10.	Cost is applicable for licensed software.	Free of cost, as it is an open source software.

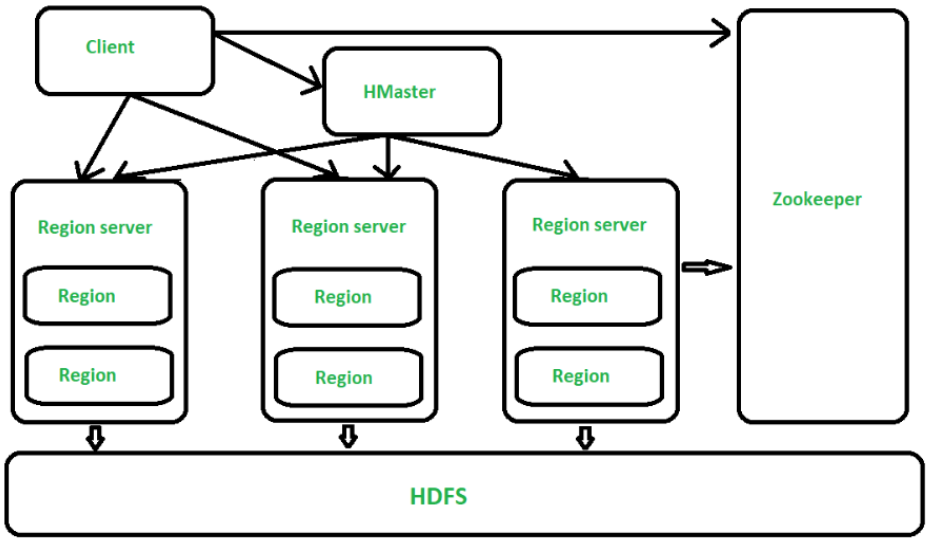
RDBMS vs HIVE:

RDBMS	Hive
It is used to maintain database.	It is used to maintain data warehouse.
It uses SQL (Structured Query Language).	It uses HQL (Hive Query Language).
Schema is fixed in RDBMS.	Schema varies in it.
Normalized data is stored.	Normalized and de-normalized both type of data is stored.
Tables in rdms are sparse.	Table in hive are dense.
It doesn't support partitioning.	It supports automation partition.
No partition method is used.	Sharding method is used for partition.

HBase concepts: https://youtu.be/ViuLZjC1B0c

- HBase is an open source non-relational distributed database. In other words, it is a NoSQL database.
- HBase is a distributed column-oriented database built on top of the Hadoop file system.
- It supports all types of data, which is why it can handle anything in the Hadoop ecosystem.
- It is based on Google's Big-Table, which is a distributed storage system designed to handle large data sets.
- It provides us with a fault-tolerant way of storing sparse data, which is common in most big data use cases.
- HBase is written in Java, and HBase applications can be written in REST, Avro, and Thrift API.

HBase Architecture:



Main 3 Components:

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. The default size of a region is 256 MB.
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.

Difference:

HDFS vs HBASE

HDFS	HBase
HDFS is a java based **file distribution system**	Hbase is **hadoop database** that runs on top of HDFS
HDFS is **highly fault-tolerant** and **cost-effective**	HBase is **partially tolerant** and **highly consistent**
HDFS Provides only **sequential read/write operation**	**Random access** is possible due to hash table
HDFS is based on **write once read many times**	HBase supports **random read and write**operation into filesystem
HDFS has a **rigid architecture**	HBase support **dynamic changes**
HDFS is prefereable for **offline batch processing**	HBase is preferable for **real time processing**
HDFS provides **high latency** for access operations.	HBase provides **low latency** access to small amount of data

RDBMS vs 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 associated with transaction integrity.	In HBase, there is no such guarantee 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.

Pig: https://youtu.be/mnOERgCKOWc?si=bzrn6RhENMDzdsoP

- Pig is basically developed by Yahoo, it uses the Pig Latin language, which is a query-based language similar to SQL.
- It is a platform for structuring the data flows, processing and analyzing massive data sets.
- Pig is responsible for executing commands and processing all MapReduce activities in the background. After processing, pig stores the result in HDFS.
- Pig Latin language is specially designed for this framework running in Pig Runtime. Like the way Java runs on the JVM.
- Pig helps simplify programming and optimization and is therefore an important part of the Hadoop ecosystem.
- Pig working like first the load command, loads the data. Then we perform various functions on it like grouping, filtering, joining, sorting, etc.
- At last, either you can dump the data on the screen, or you can store the result back in HDFS.
- It can easily handle structured as well as unstructured data.

Advantages of Apache Pig:

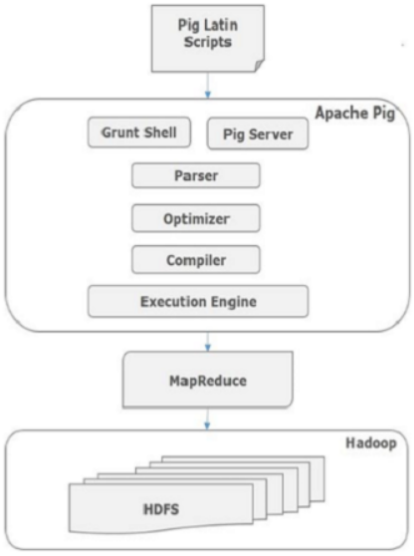
Less code - The Pig consumes less line of code to perform any operation.

Reusability - The Pig code is flexible enough to reuse again.

Difference between Pig and MapReduce

Apache Pig	MapReduce
It is a scripting language.	It is a compiled programming language.
Abstraction is at higher level.	Abstraction is at lower level.
It have less line of code as compared to MapReduce.	Lines of code is more.
Less effort is needed for Apache Pig.	More development efforts are required for MapReduce.
Code efficiency is less as compared to MapReduce.	As compared to Pig efficiency of code is higher.
Pig provides built in functions for ordering, sorting and union.	Hard to perform data operations.
It allows nested data types like map, tuple and bag	It does not allow nested data types

Architecture of Pig:



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

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

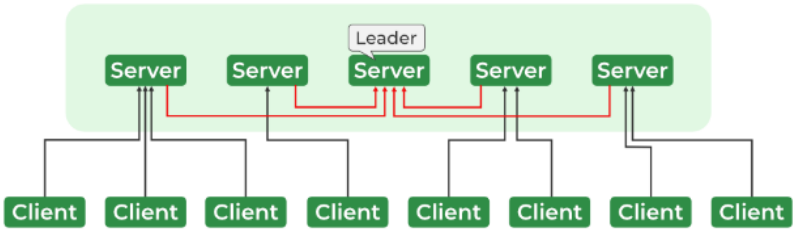
Compiler: The compiler compiles the optimized logical plan into a series of MapReduce jobs.

Execution Engine: Eventually, all the MapReduce jobs are submitted to Hadoop in a sorted order. Ultimately, it produces the desired results while these MapReduce jobs are executed on Hadoop.

Zookeeper:

- There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency.
- Before Zookeeper, it was very difficult and time consuming to coordinate between different services in Hadoop Ecosystem.
- Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- Zookeeper is implemented in Java and is widely used in distributed systems, particularly in the Hadoop ecosystem
- ZooKeeper provides a simple and reliable way to manage complex distributed systems, making it easier to build and maintain Hadoop applications.
- Leader Election: ZooKeeper can elect a leader in a distributed system, ensuring that only one node is making critical decisions.
- ZooKeeper provides distributed locks and semaphores that Hadoop applications can use to coordinate access to shared resources, preventing conflicts and ensuring data consistency.

Architecture of Zookeeper:



The ZooKeeper architecture consists of a hierarchy of nodes called znodes, organized in a tree-like structure. Each znode can store data and has a set of permissions that control access to the znode. The znodes are organized in a hierarchical namespace, similar to a file system. At the root of the hierarchy is the root znode, and all other znodes are children of the root znode.

How Zookeeper helps in monitoring a cluster:

ZooKeeper itself is not primarily a monitoring tool; it is a distributed coordination service. However, it can play a role in helping monitor a cluster by providing information and coordination for monitoring systems and applications.

1. Configuration Management: ZooKeeper can store configuration information for various components of a cluster. Monitoring tools can access this configuration data to ensure that they are collecting the right metrics and monitoring the correct resources.
2. Health Checks: Applications and services within a cluster can periodically update their health status in ZooKeeper. Monitoring tools can read this information to assess the health of individual nodes or services. If a node becomes unhealthy or a service fails, ZooKeeper can help propagate this information to the monitoring tools
3. Leader Election: ZooKeeper can elect a leader in a distributed system, ensuring that only one node is making critical decisions.
4. ZooKeeper provides distributed locks and semaphores that Hadoop applications can use to coordinate access to shared resources, preventing conflicts and ensuring data consistency.
5. Failover and Recovery: If a monitoring tool or node goes down, ZooKeeper can help facilitate failover and recovery processes, ensuring continuous monitoring without manual intervention.
6. Watchers for Real-Time Alerts: ZooKeeper allows clients to set watches on specific znodes. Monitoring tools can use these watches to receive real-time notifications when changes occur. For example, if a critical configuration parameter changes in ZooKeeper, the monitoring system can immediately adjust its monitoring strategy.

How HBASE uses Zookeeper:

HBase, a distributed NoSQL database that runs on top of the Hadoop Distributed File System (HDFS), indeed uses Apache ZooKeeper for coordination and management. ZooKeeper plays a crucial role in HBase for tasks such as leader election, cluster coordination, and monitoring.

HBase uses ZooKeeper primarily for coordination and synchronization among its distributed components. Here are some key aspects of HBase's use of ZooKeeper:

1. Cluster Coordination: ZooKeeper helps HBase maintain information about the active region servers, table schema, and other cluster-related metadata.
2. Master Failover: ZooKeeper facilitates leader election for the HBase Master, ensuring high availability in the event of a Master failure.
3. Root and Meta Tables: HBase uses ZooKeeper to locate the ROOT and META tables, which are essential for discovering the locations of user tables.
4. Schema Changes: ZooKeeper assists in tracking and distributing schema changes across the cluster.

CHAPTER:6 APACHE SPARK

Introduction to Data Analysis with Spark:

- Apache Spark is a lightning-fast cluster computing designed for fast computation.
- It was built on top of Hadoop MapReduce and it extends the MapReduce model to efficiently use more types of computations which includes Interactive Queries and Stream Processing.
- One of the main features Spark offers for speed is the ability to run computations in memory, but the system is also more efficient than MapReduce for complex applications running on disk.
- Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming.

- Spark is designed to be highly accessible, offering simple APIs in Python, Java, Scala, and SQL, and rich built-in libraries. It also integrates closely with other Big Data tools.
- Spark can run in Hadoop clusters and access any Hadoop data source, including Cassandra.

Apache Spark has following features:
1. Speed : Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing the number of read/write operations to disk.
2. Supports multiple languages: Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages.
3. Advanced Analytics: Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Spark Architecture:
- The Spark follows the master-slave architecture. Its cluster consists of a single master and multiple slaves.
- The Spark architecture depends upon two abstractions:
  Resilient Distributed Dataset (RDD) & Directed Acyclic Graph (DAG)

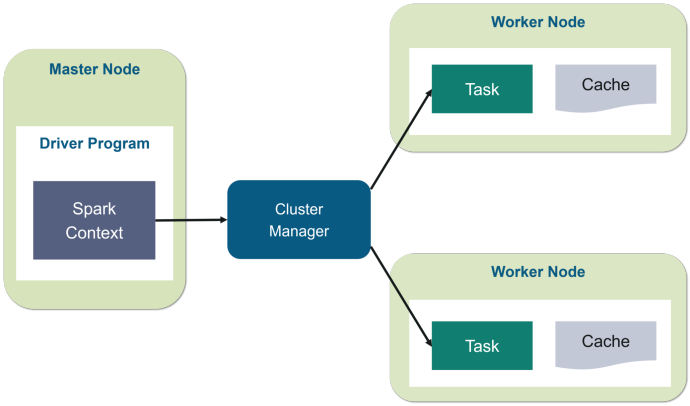
RDD(Resilient Distributed Datasets): The Resilient Distributed Datasets are the group of data items that can be stored in-memory on worker nodes. Here,
RDD stands for,
Resilient: Restore the data on failure.
Distributed: Data is distributed among different nodes.
Dataset: Group of data.

Key characteristics of RDDs in Spark:
1. Immutable: RDDs are read-only and cannot be modified once created. To apply transformations or perform actions on RDDs, you create new RDDs with the desired modifications.
2. Distributed: RDDs are divided into partitions, and these partitions are distributed across worker nodes in the Spark cluster. Each partition can be processed in parallel on different nodes.
3. Resilient: RDDs are fault-tolerant, meaning they can recover from node failures.
4. In-Memory: RDDs are stored in memory, which allows for fast and efficient processing.
5. Transformation and Action: RDDs support two types of operations: transformations and actions. Transformations create new RDDs from existing ones (e.g., map, filter, reduceByKey), while actions perform computations on RDDs and return results (e.g., count, collect, saveAsTextFile).
6. Caching: You can persist (cache) RDDs in memory to speed up repetitive computations.
7. Data Sources: RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, Apache HBase, Apache Cassandra, and more.
8. Lazy Evaluation: RDD transformations are lazily evaluated, which means that they are not computed immediately when defined. Instead, they are evaluated when an action is called, allowing Spark to optimize the execution plan.

- There are two ways to create RDDs − parallelizing an existing collection in your driver program, or by referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, etc.

Directed Acyclic Graph (DAG):
In Apache Spark, a DAG (Directed Acyclic Graph) is a fundamental concept that represents the logical execution plan of a Spark application. The DAG defines the sequence of transformations and actions to be applied to RDDs or DataFrames in order to compute the desired result.
When an action is called, Spark initiates the execution of the DAG from the root nodes (e.g., input data) to compute the final result. Intermediate results are computed and cached when necessary.
Spark stores the DAG and lineage information in memory, allowing it to recover lost data partitions in case of node failures. This lineage information helps achieve fault tolerance.

Working of Spark Architecture:



Master Node: The Master Node is the central coordinator of a Spark cluster.
- It is responsible for managing the overall execution of Spark applications and coordinating tasks across the Worker Nodes.
- *Key components of the Master Node include:*
1. Driver Program: The Driver Program is the entry point for Spark applications. It runs on the Master Node and controls the application's execution.
2. Job Scheduler: The Master Node schedules tasks and stages for execution. It breaks down the application into smaller units of work called stages, which are distributed to Worker Nodes.
3. Task Scheduler: The Task Scheduler assigns tasks to Worker Nodes based on data locality, resource availability, and other factors. It tracks the progress of tasks and retries failed tasks if necessary.
4. Cluster Manager: The Master Node interacts with a cluster manager, such as Apache Mesos, Hadoop YARN, or Spark's built-in standalone cluster manager. The cluster manager allocates resources to the application and manages Worker Nodes.

Worker Nodes: Worker Nodes are the computing nodes in the Spark cluster where the actual data processing occurs.
- They run tasks assigned to them by the Master Node and store data in memory for efficient processing.
- *Key components of Worker Nodes include:*
1. Caches: Worker Nodes can store data in memory using the built-in in-memory data storage called RDD (Resilient Distributed Dataset) cache. This enables fast data access for iterative algorithms.
2. Task Execution: Worker Nodes execute tasks concurrently, and the results are returned to the Driver Program on the Master Node.
3. Executor: Each Worker Node runs one or more Executors, which are separate JVM processes responsible for executing tasks. Executors manage the allocation of CPU and memory resources.

Spark Components:
1. Spark Core: The Spark Core is the heart of Spark and performs the core functionality.
It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.
2. Spark SQL: The Spark SQL is built on the top of Spark Core. It provides support for structured data.
- It supports JDBC and ODBC connections that establish a relation between Java objects and existing databases, data warehouses and business intelligence tools.
- It also supports various sources of data like Hive tables, Parquet, and JSON.
- It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL called the HQL (Hive Query Language).
3. Spark Streaming: Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data.
- It uses Spark Core's fast scheduling capability to perform streaming analytics.
- It accepts data in mini-batches and performs RDD transformations on that data.
4. MLlib: The MLlib is a Machine Learning library that contains various machine learning algorithms.
- These include correlations and hypothesis testing, classification and regression, clustering, and principal component analysis.
- It is nine times faster than the disk-based implementation used by Apache Mahout.
5. GraphX: The GraphX is a library that is used to manipulate graphs and perform graph-parallel computations.
- It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge.
- To manipulate graph, it supports various fundamental operators like subgraph, join Vertices, and aggregate Messages.

CHAPTER:7 NoSQL

What is NoSQL?:

- NoSQL stands for Not Only SQL. These are non-relational, open source, distributed databases.
- NoSQL (commonly known as "Not Just SQL") represents a completely different database framework that can achieve high-performance and agile processing of large-scale information.
- In other words, it is a database infrastructure, very suitable for the huge needs of big data.
- They provide flexible schemas and scale easily with large amounts of data and high user loads.
- They are schema free.
- Most of the NoSQL are open sources and has capability of horizontal scaling.
- can handle huge amount of data.
- NoSQL databases are distributed: They are distributed meaning the data is distributed across several nodes in a cluster constituted of low-cost commodity hardware.

Where it is used?

NoSQL databases are widely used in real-time web applications and big data, because their main advantages are high scalability and high availability.

Use Cases:

1. Big Data and Analytics: NoSQL databases are often used in big data applications for storing and processing vast amounts of data generated by sensors, social media, e-commerce, and more
2. IoT (Internet of Things): IoT applications generate a massive amount of data from sensors and devices. NoSQL databases can handle the high volume and variety of IoT data.
3. Document Stores: NoSQL document stores are used for managing and querying semi-structured or unstructured data, such as JSON or XML documents.
4. Social Media and User Profile Management: NoSQL databases are employed by social media platforms to store user profiles, social connections, comments, and other user-generated data.
5. Real-Time Applications: NoSQL databases are used in real-time applications like online gaming, chat applications, and financial trading systems, where low-latency data access and real-time updates are critical.

Apart from that is also used in E-commerce and recommendation engines etc. so when we have to deal with large volumes of data, specifically unstructured data we use NoSQL.

Types of NoSQL Database:

1. Key-Value Pair Oriented: Key-value Stores are the simplest type of NoSQL database. Data is stored in key/value pairs.
- In Key-value store databases, the key can only be string, whereas the value can store string, JSON, XML, Blob, etc. Due to its behavior, it is capable of handling massive data and loads.
- The use case of key-value stores mainly stores user preferences, user profiles, shopping carts, etc.

Key	Value
First Name	Rahul
Last Name	Mehta

- DynamoDB, Riak, Redis are a few famous examples of Key-value store NoSQL databases.

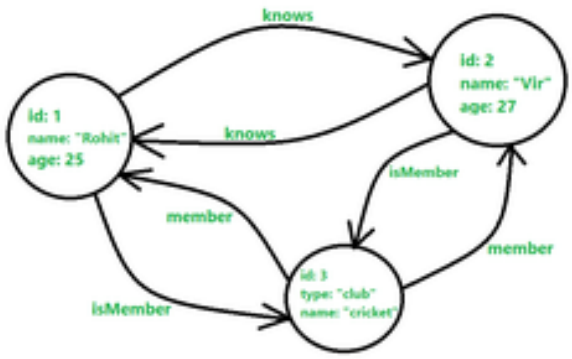
2. Document Oriented: Document Databases use key-value pairs to store and retrieve data from the documents. A document is stored in the form of XML and JSON.
- Data is stored as a value. Its associated key is the unique identifier for that value.
- The difference is that, in a document database, the value contains structured or semi-structured data.
- This structured/semi-structured value is referred to as a document and can be in XML, JSON or BSON format.
- Examples of Document databases are – MongoDB, OrientDB, Apache CouchDB, IBM Cloudant, CrateDB, BaseX, and many more.
- Use cases: E-commerce platforms, Content management systems , Analytics platforms , Blogging platforms

Example:

```
{
"Book Name": "Fundamentals of Business Analytics",
"Publisher":
"Wiley India",
"Year of Publication": "2011"
}
```

3. Column-Oriented: Column based database store data together as columns instead of rows and are optimized for queries over large datasets.
- It works on columns and are based on BigTable paper by Google.
- Every column is treated separately. Values of single column databases are stored contiguously.
- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column-based database.
- Use cases: Content management systems, Blogging platforms, Systems that maintain counters, Systems that require heavy write requests (like log aggregators)

4. Graph Oriented: Graph databases form and store the relationship of the data.
- Each element/data is stored in a node, and that node is linked to another data/element.
- A typical example for Graph database use cases is Facebook. It holds the relationship between each user and their further connections.
- Graph databases help search the connections between data elements and link one
part to various parts directly or indirectly.
- The Graph database can be used in social media, fraud detection, and knowledge graphs. Examples of Graph Databases are – Neo4J, Infinite Graph, OrientDB, FlockDB, etc.



Advantages of NoSQL:
- Flexible Schema: NoSQL databases do not have a fixed schema and can accommodate changing data structures without the need for migrations or schema alterations.
- Horizontal Scaling: NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well-suited for handling large amounts of data and high levels of traffic.
- High Availability
- Easy insert and read operations
- Caching mechanism
- Store unstructured, semi-structured, or structured data
- Handle large volumes of data at high speed with a scale-out architecture
- Enable easy updates to schemas and fields
- NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.

- Cost-effectiveness: NoSQL databases are often more cost-effective than traditional relational databases, as they are typically less complex and do not require expensive hardware or software.
- Ideal for agile development.

Why NoSQL: In real life use case data is becoming more unstructured and NoSQL database allows to store huge amount of unstructured data and gives a lot of flexibility.
- Nowadays storage is not a problem, redundancy is allowed but data retrieval must be fast, that's what nosql does.

NewSQL: NewSQL is a category of relational database management systems (RDBMS) that aims to combine the best aspects of traditional relational databases with the advantages of NoSQL databases.
- The goal of the newsql database is to expand and maintain consistency.
- NewSQL is a unique database system that combines ACID compliance with horizontal scaling.
Features: 1. SQL Support: NewSQL databases use SQL as their query language
2. In-memory storage and data processing supply fast query results.
3. High availability due to the database replication mechanism.
4. ACID properties preserve the features of RDBMS.
Examples of NewSQL databases include Google Spanner, CockroachDB, NuoDB, and TiDB.

CAP Theorem:https://youtu.be/EIl02n-FxTg?si=nfnqhgNcdJ61i3-M

BASE Properties: https://youtu.be/qrwhdNf51aU?si=iAJY1DGlahyMHP-p
Basically Available: NoSQL databases prioritize high availability and fault tolerance. This means that the system remains operational even in the presence of node failures. It may sacrifice consistency temporarily to achieve availability.
Soft state: Soft state implies that the data in a NoSQL database can change over time without any input. In other words, the state of the system may change due to eventual consistency, where replicas of the data may take some time to converge to a consistent state.
Eventually Consistent: NoSQL databases often provide eventual consistency. This means that, given enough time and after all updates have ceased, all replicas will converge to a consistent state. During this convergence period, the data may appear inconsistent.

CHAPTER:8 DATABASE FOR MODERN WEB:

https://drive.google.com/file/d/1u3IvR1ql98OvUdlyqpXDQ3Kk-3LfGIqo/view
MongoDB key features:
No-SQL database,
Schema-less Design: It has dynamic Schema,
Follows CAP Theorem : Consistency, Availability, and Partition tolerance,
Highly Scalable: Horizontal scaling for big data,Faster than RDBMS
High Availability: Automatic failover with replica sets.
Flexible Queries: Rich query capabilities.
Indexing: Supports various index types.
Aggregation Framework: Complex data transformations.
Auto-Sharding: Distributes data across servers.
JSON/BSON Format: Familiar data format.
Geospatial Support: Location-based data handling.
Replica Sets: Data redundancy and fault tolerance.
Community and Enterprise Editions: Options for support.
Rich Ecosystem: Tools and libraries for development.

Core Server tools: MongoDB provides core server tools such as mongod for running the MongoDB server, and mongo for accessing the MongoDB shell to interact with the server.

UNIT TEST 1 PAPER:

https://drive.google.com/file/d/12f68fDY6Q2Yyef5hqaVmA2KDp22x1a2P/view?usp=sharing

QUESTION-BANK REMAINING QUESTIONS:

Q. List various application of big data. How it can be used to improve business for a superstore.

Applications of Big Data:

Recommendation: By tracking customer spending habit, shopping behavior, Big retails store provide a recommendation to the customer. E-commerce site like Amazon, Walmart, Flipkart does product recommendation.

Healthcare: Disease Surveillance: Analyzing medical data to track and predict disease outbreaks.

Personalized Medicine: Tailoring treatment plans to individual patients based on their genetic and medical history.

Smart Traffic System: Data about the condition of the traffic of different road, collected through camera kept beside the road, at entry and exit point of the city, GPS device placed in the vehicle (Ola, Uber cab, etc.). All such data are analyzed and jam-free or less jam way, less time taking ways are recommended. Such a way smart traffic system can be built in the city by Big data analysis.

Finance: Fraud Detection: Identifying fraudulent transactions in real-time.

Algorithmic Trading: Making investment decisions based on real-time market data.

Secure Air Traffic System: At various places of flight (like propeller etc) sensors present. These sensors capture data like the speed of flight, moisture, temperature, other environmental condition. Based on such data analysis, an environmental parameter within flight are set up and varied.

Auto Driving Car: Big data analysis helps drive a car without human interpretation. In the various spot of car camera, a sensor placed, that gather data like the size of the surrounding car, obstacle, distance from those, etc. These data are being analyzed, then various calculation like how many angles to rotate, what should be speed, when to stop,

Virtual Personal Assistant Tool: Big data analysis helps virtual personal assistant tool (like Siri in Apple Device, Cortana in Windows, Google Assistant in Android) to provide the answer of the various question asked by users. This tool tracks the location of the user, their local time, season, other data related to question asked, etc. Analyzing all such data, it provides an answer.

Manufacturing: Predictive Maintenance: Monitoring equipment to schedule maintenance before failures occur.

Quality Control: Ensuring product quality by analyzing production data.

Supply Chain Optimization: Improving the efficiency of the supply chain.

Media and Entertainment Sector: Media and entertainment service providing company like Netflix, Amazon Prime, Spotify do analysis on data collected from their users. Data like what type of video, music users are watching, listening most, how long users are spending on site, etc are collected and analyzed to set the next business strategy.

IoT: Manufacturing company install IOT sensor into machines to collect operational data. Analyzing such data, it can be predicted how long machine will work without any problem when it requires repairing so that company can take action before the situation

How it can be used to improve business for a superstore:-

Big data can be a valuable asset for improving business operations in a superstore. Like

Customer Insights: Analyze purchase history, demographics, and behavior to group customers Use data to offer personalized product recommendations, discounts, and marketing messages to individual customers.

Pricing Optimization: Adjust prices in real-time based on market conditions, competitor pricing, and customer demand.

Customer Feedback and Sentiment Analysis: Monitor customer reviews and social media feedback to understand customer sentiment and make improvements accordingly.

Competitor Analysis: Analyze market data and competitors' pricing, promotions, and customer strategies to make informed decisions and stay competitive.

Predictive Maintenance: Monitor and maintain store infrastructure, such as refrigeration, and lighting, to prevent equipment failures and reduce maintenance costs.

To effectively implement big data solutions, a superstore should invest in data collection tools, data storage infrastructure, data analysis platforms, and data science expertise. By leveraging big data insights, a superstore can enhance its decision-making processes, improve customer experiences,

Explain “Map Phase” and “Combiner Phase” in Map-Reduce. Explain working of reduce phase of Map-Reduce with an example. Explain “Shuffle & Sort” phase and “Reducer Phase” in Map-Reduce.
https://www.tutorialspoint.com/map_reduce/map_reduce_algorithm.htm

List various configuration files used in Hadoop Installation. What is use of mapred-site.xml?
Configuration Files are the files which are located in the extracted tar.gz file in the etc/hadoop/ directory.
1) HADOOP-ENV.sh:It specifies the environment variables that affect the JDK used by Hadoop Daemon (bin/hadoop). We know that Hadoop framework is written in Java and uses JRE so one of the environment variable in Hadoop Daemons is $Java_Home in Hadoop-env.sh.
2) CORE-SITE.XML:It is one of the important configuration files which is required for runtime environment settings of a Hadoop cluster. It configures the core settings for Hadoop, including the default file system, Hadoop runtime directory, and various I/O settings. It informs Hadoop daemons where the NAMENODE runs in the cluster. It also informs the Name Node as to which IP and ports it should bind.
3) HDFS-SITE.XML:It is one of the important configuration files which is required for runtime environment settings of a Hadoop. It contains the configuration settings for NAMENODE, DATANODE, SECONDARYNODE. It is used to specify default block replication. The actual number of replications can also be specified when the file is created.
This file is specific to HDFS (Hadoop Distributed File System) and contains configurations related to data storage and replication, such as block size and replication factor
4) MAPRED-SITE.XML->>It contains the configuration settings for MapReduce .The mapred-site.xml file in Hadoop is used to configure settings specific to the MapReduce framework, which is a core component of Hadoop responsible for processing large datasets. This configuration file allows you to define parameters related to the MapReduce infrastructure, including the configuration of JobTracker and TaskTracker services, task scheduling, and other MapReduce-specific settings
In this file, we specify a framework name for MapReduce, by setting the MapReduce.framework.name.
5) Masters:It is used to determine the master Nodes in Hadoop cluster. It will inform about the location of SECONDARY NAMENODE to Hadoop Daemon.
6) Slave:It is used to determine the slave Nodes in Hadoop cluster. The Slave file at Master Node contains a list of hosts, one per line. The Slave file at Slave server contains IP address of Slave nodes.

How to create collection in MongoDB? Explain with its syntax. Explain MongoDB sharding process. Which terms are used for table, row, column and table-join in MongoDB?
- You can create a collection using the createCollection() database method.
Syntax: db.createCollection(“Collection_Name”)
- Sharding is a method for allocating data across multiple machines. MongoDB used sharding to help deployment with very big data sets and large throughput the operation.
What is Apache Pig and why do we need it? OR
Discuss how will Pig data model help in effective data flow.
Apache Pig is a high-level platform and scripting language built on top of Hadoop, designed for processing and analyzing large datasets. It was developed to simplify the task of writing complex data processing tasks on Hadoop clusters.
Why we Need it:
- Apache Pig provides an abstraction layer on top of Hadoop, allowing users to express data processing operations using a high-level scripting language, called Pig Latin.
- Like Hadoop, Pig is designed to work on large-scale distributed data processing tasks. It can handle massive datasets and leverage the scalability of Hadoop clusters.
- Pig Latin is more user-friendly than writing complex MapReduce jobs in Java.
- Pig works seamlessly with other Hadoop ecosystem tools, such as Hive, HBase, and Flume. This integration allows users to build end-to-end data processing pipelines.
- Pig includes optimization techniques that automatically optimize the execution plan, improving the performance of data processing jobs.
- Pig can handle large datasets and is well-suited for distributed data processing, ensuring that data flows effectively even as the volume of data grows.

How can you minimize data transfers when working with Spark?

Several ways for minimizing data transfers while working with Apache Spark are as follows.

Using Accumulators

Using Broadcast Variables

Avoiding- ByKey operations, repartitions responsible for triggering shuffles.

https://www.edureka.co/community/21159/how-can-i-minimize-data-transfers-when-working-with-spark

Explain Avro data serialization technique in Map-Reduce.

Avro is a data serialization framework that is commonly used in Hadoop and MapReduce. It provides a compact, fast, and efficient way to serialize data in a binary format.

- Avro uses a schema to define the structure of the data. The schema is used for both serialization and deserialization, ensuring that the data's structure is consistent, making it easier to read and write data during MapReduce tasks.
- Avro's binary format is highly compact, which means it reduces the size of the data during serialization. This is particularly important in the context of MapReduce, where reducing data size can lead to more efficient data transfer and storage.
- Avro's binary format is designed for speed. In MapReduce, where large amounts of data need to be serialized and deserialized quickly, Avro's efficiency is a significant advantage.
- Avro supports schema evolution, which means you can change the schema of your data without breaking compatibility with existing data.

In a MapReduce job, Avro can be used to serialize data before it's written to the Hadoop Distributed File System (HDFS) and deserialize data when it's read during processing.

How Big Data Analytics can be useful in the development of smart cities

Big Data Analytics plays a significant role in the development of smart cities by enabling data-driven decision-making, optimizing resources.

- Traffic Management: Big Data Analytics can process data from various sources, such as sensors, GPS, and traffic cameras, to monitor traffic conditions in real-time. This data can be used to optimize traffic flow, reduce congestion, and improve road safety.
- Waste Management: Big Data can optimize waste collection routes and schedules based on real-time data, ensuring that garbage collection is efficient and reducing costs.
- Healthcare: Health data from wearables and health facilities can be analyzed to track disease outbreaks, monitor public health, and improve healthcare services.
- Urban Planning: Big Data helps in making informed decisions regarding city development, including where to build new infrastructure, parks, and public spaces based on data on population density and demographics.
- Infrastructure Maintenance: By analyzing data from sensors embedded in infrastructure like bridges and buildings, cities can predict maintenance needs, reducing downtime and ensuring safety.

Compare Raw oriented and Column Oriented database structures.

Characteristic	Raw-Oriented Database	Column-Oriented Database
Data Storage	Stores data in a row-wise format where each row contains all the attributes of a single record.	Stores data in a column-wise format where all the values for a single column are stored together.
Data Retrieval Efficiency	Efficient for retrieving a single record or a subset of records because all attributes are stored together.	Efficient for aggregations, analytics, and queries that involve a subset of columns but may require scanning the entire column.
Compression Efficiency	Inefficient for compression because different data types and values are intermixed in rows.	Highly efficient for compression because similar data types and values are stored together, enabling better compression ratios.
Insert and Update Efficiency	Inefficient for insert and update operations since updating a single attribute may require rewriting the entire row.	More efficient for insert and update operations as it only requires updating the relevant column values, reducing I/O overhead.
Schema Flexibility	Less flexible because changing the schema often requires altering the entire table structure.	More flexible as adding or removing columns is relatively straightforward, making it adaptable to changing data requirements.
Analytical Processing	Less efficient for analytical queries and reporting because it may involve scanning through irrelevant columns.	Highly efficient for analytical processing as it can skip irrelevant columns, reducing I/O and processing overhead.
Aggregation and Summarization	May require scanning through all records and columns, making aggregations and summarization slower.	Well-suited for aggregations and summarization due to the efficient storage and retrieval of column data.
Usage in Data Warehousing	Less common in data warehousing environments.	Commonly used in data warehousing due to its efficiency in analytical processing.

What is Compute and Storage nodes in Hadoop?

Storage Nodes: Storage nodes, often referred to as DataNodes, are responsible for storing and managing the actual data in a Hadoop cluster. They store data in the Hadoop Distributed File System (HDFS). DataNodes keep data blocks and replicate them to ensure fault tolerance and because of these replicas if one datanode fails, data can still be retrieved from other nodes. They respond to read and write requests from Compute nodes (TaskTrackers).

Compute Nodes: Compute nodes, often referred to as TaskTrackers, are responsible for processing data in a Hadoop cluster. They are responsible for executing tasks such as MapReduce jobs. TaskTrackers communicate with the DataNodes to read data for processing and write intermediate results. Compute nodes execute tasks in a distributed and parallel manner. Compute nodes retrieve data from Storage nodes, process it, and then write the results back to the HDFS.

Explain the core methods of the Reducer?

The 3 core methods of a reducer are –

1) setup () - This method of the reducer is used for configuring various parameters like the input data size, distributed cache, heap size, etc. Basically It allows you to perform any setup or configuration tasks that are needed before processing the data.

Function Definition- public void setup (context)

2) reduce () - it is heart of the reducer which is called once per key with the associated reduce task. This is the main method of the Reducer class, and it is where the actual data processing and aggregation occur. Inside the reduce method, you can implement custom aggregation and computation logic based on the key and associated values.

Function Definition -public void reduce (Key,Value,context)

3) cleanup () - This method is called only once at the end of reduce task for clearing all the temporary files.

Function Definition -public void cleanup (context)

What are primary and secondary replica sets in MongoDB?

Primary Replica Set: The primary replica set member is the main read/write node in a MongoDB replica set. It is the only member that can accept write operations, such as insert, update, and delete operations. All write operations go to the primary node by default.

- If the primary node becomes unavailable due to hardware failure or other issues, the replica set can automatically trigger an election process to select a new primary node from the available secondary nodes. This process helps maintain the high availability of the database.
- Any data changes made to the primary node are asynchronously replicated to the secondary nodes in the replica set. This replication ensures data redundancy and fault tolerance.

Secondary Replica Set: Secondary nodes in a replica set are read-only copies of the primary data. They mirror the data stored on the primary node, allowing for read-heavy workloads and providing data redundancy.
- Secondary nodes replicate data from the primary node, ensuring that they have the same dataset. Data replication can be asynchronous, meaning there may be a slight delay between changes on the primary node and their appearance on the secondary nodes.
- In the event of the primary node's unavailability, a secondary node can be elected as the new primary node through an automated process, ensuring uninterrupted write operations.
- Secondary nodes are used for read operations like queries and data retrieval. They help distribute read traffic, improving read scalability and performance.

What is the role of a profiler in MongoDB? Where does the writes all the data?

In MongoDB, the profiler is a diagnostic tool that allows you to capture information about the operations performed on the database. It helps you understand and analyze the performance of queries, updates, and other operations. The profiler collects detailed data on the execution of various database operations and logs this information for analysis.

Basically The MongoDB profiler is a built-in feature that helps in monitoring and profiling database operations.

By default, the profiler does not write data to the database itself. Instead, it stores the profile information in a system collection named '*system.profile*' within the MongoDB database. You can query this collection to retrieve information about recorded database operations, and you can analyze the data to gain insights into the database's behavior and performance.

How is big data analysis helpful in increasing business revenue?

Big data analysis is helpful in increasing business revenue in easy terms because it provides valuable insights and opportunities for making better decisions.

Customer Insights: Big data analysis helps businesses understand their customers better. By analyzing customer data, such as purchase history, demographics, and online behavior, companies can tailor products, services, and marketing to individual preferences.

Market Trends: Big data analysis allows businesses to monitor market trends and changes in real-time. By identifying emerging trends and adapting their strategies accordingly, companies can seize new opportunities and stay ahead of competitors.

Product Development: Analyzing customer feedback and market data can guide product development efforts. Companies can create products that better meet customer needs and preferences, leading to higher sales and market share.

Price Optimization: Big data analysis enables dynamic pricing strategies. Companies can adjust prices in real-time based on demand, competitor pricing, and other factors, maximizing revenue and profitability.

Targeted Marketing: Data analysis helps in targeted marketing. Businesses can reach the right audience with relevant messages and offers, leading to higher conversion rates and revenue.

One well-known example of a business successfully using big data analytics to improve the customer experience is Netflix. It uses this technology to personalize its recommendations and content offerings to each customer based on their viewing history and preferences. This has helped Netflix save a whopping $1 billion annually simply through customer retention that it made possible using big data.

Explain the steps to be followed to deploy a Big Data solution.

There are three steps to deploy a Big Data

Data Ingestion: The first step for deploying a big data solution is the data ingestion i.e. extraction of data from various sources.
- The data source may be a CRM like Salesforce, Enterprise Resource Planning System like SAP, RDBMS like MySQL or any other log files, documents, social media feeds etc.
- The data can be ingested either through batch jobs or real-time streaming. The extracted data is then stored in HDFS.

Data Storage: After the data ingestion, the next step is to store the extracted data. The data either be stored in HDFS or NoSQL database (i.e. HBase).
- The HDFS storage works well for sequential access whereas HBase for random read/write access.

Data Processing: The final step in deploying a big data solution is the data processing.

The data is processed through one of the processing frameworks like Spark, MapReduce, Pig, etc.

Define HDFS and YARN, and talk about their respective components.

The Hadoop Distributed File System (HDFS) is the storage unit that's responsible for storing different types of data blocks in a distributed environment.

The two main components of HDFS are-
1.NameNode – A master node that processes metadata information for data blocks contained in the HDFS
2.DataNode – Nodes which act as slave nodes and simply store the data, for use and processing by the NameNode
The Yet Another Resource Negotiator (YARN) is the processing component of Apache Hadoop and is responsible for managing resources and providing an execution environment for said processes.
The two main components of YARN are-
1.ResourceManager– Receives processing requests and allocates its parts to respective NodeManagers based on processing needs.
2.NodeManager– Executes tasks on every single Data Node

Explain the term 'Commodity Hardware.'

Commodity hardware, in the context of big data, refers to standard hardware components that are widely available in the market and are relatively inexpensive compared to specialized or high-end hardware and easily interchangeable with other commodity hardware. Commodity hardware is commonly used in big data environments, including Hadoop clusters and distributed computing platforms. Commodity hardware comprises commonly available hardware components, including processors (CPUs), memory (RAM), hard drives or solid-state drives (HDDs or SSDs), and network interfaces.

Define and describe the term FSCK.

Fsck stands for File System Check. This command is used by HDFS and also used to check inconsistencies and if there is any problem in the file.
- For example, if there are any missing blocks for a file, HDFS gets notified through this command.
- There are two modes used in fsck command interactive and non-interactive.

Interactive mode prompts the user each time it finds an error to determine whether or not the user wants to fix the error or continue without fixing.

Non-interactive mode automatically corrects errors without prompting the user.

What is the purpose of the JPS command in Hadoop?

We use the JPS (Java Virtual Machine Process Status Tool) command to check if all the Hadoop daemons like NameNode, DataNode, ResourceManager, NodeManager etc. are properly running. This is a basic check to see if all the required Hadoop services are running or not before going forward with any Hadoop operations or any additional setups.

Name the different commands for starting up and shutting down Hadoop Daemons.

start-dfs.sh - Starts the HDFS daemons, the namenode and datanodes.
stop-dfs.sh - Stops the HDFS daemons.
start-mapred.sh - Starts the Hadoop Map/Reduce daemons, the jobtracker and tasktrackers.
stop-mapred.sh - Stops the Hadoop Map/Reduce daemons.
start-all.sh - Starts all Hadoop daemons, the namenode, datanodes, the jobtracker and tasktrackers
stop-all.sh - Stops all Hadoop daemons.
start-yarn.sh, stop-yarn.sh - To start and stop YARN daemons, the resource manager

Explain working of reduce phase of MapReduce with an example.

Map Phase:
- Input Split: The input data is divided into smaller chunks called input splits, with each split processed by a separate map task. Input splits are usually based on the underlying Hadoop file system's block size.
- Mapping Function: The map phase begins with the execution of a user-defined mapping function on each input split. This function takes input data and processes it to generate a set of intermediate key-value pairs.
- Intermediate Key-Value Pairs: The mapping function produces a collection of intermediate key-value pairs. These pairs are not the final output but serve as an intermediate step in the data processing.
- Shuffling and Sorting: After mapping, Hadoop groups and sorts the intermediate key-value pairs based on keys. This step is essential to prepare data for the reduce phase. All values associated with the same key are grouped together.
- Partitioning: The data is partitioned into multiple partitions based on the number of reduce tasks specified. Each partition is sent to a corresponding reduce task for processing.

Reduce Phase:
- Reduce Task: In the reduce phase, a separate reduce task is assigned to process each partition of data generated in the map phase. These tasks execute in parallel on different nodes in the cluster.
- Reducing Function: A user-defined reducing function is applied to the grouped and sorted key-value pairs. This function typically performs some aggregation or computation on the values associated with each key.

- Final Output: The reducing function produces the final output, which consists of a set of key-value pairs. These key-value pairs represent the results of the data processing job and are typically stored in the Hadoop Distributed File System (HDFS).
- Data Writing: The final output data is written to the specified output directory in HDFS or another storage system, making it available for further analysis or use.

Example: Explain this word count example: https://youtu.be/pcTFiU7wKkQ?si=9S5-F4UNb_IgUMv4&t=413

What is Big Data? Explain how big data processing differs from distributed processing.

Big Data refers to extremely large and complex datasets that cannot be effectively processed or analyzed using traditional data processing tools.

let's simplify the difference between big data processing and distributed processing:

Characteristic	Big Data Processing	Distributed Processing
Data Type	Handles very large, complex, and diverse datasets, including structured, semi-structured, and unstructured data.	Can handle a range of data sizes and types, including structured data, but not necessarily very large or diverse datasets.
Scale	Specifically deals with the challenges of extremely large volumes of data (terabytes to petabytes or more).	Can encompass data of various sizes, which may not necessarily be "big data."
Tools and Frameworks	Relies on specialized tools and frameworks designed for big data, like Hadoop, Spark, and NoSQL databases.	Utilizes a wide range of technologies, including traditional databases and distributed computing libraries like MPI.
Parallelism	Emphasizes distributing data and computation across a large number of nodes in a clustered environment to manage the volume of data.	Employs parallelism to improve computational efficiency and reliability but may not require as extensive distribution.
Data Challenges	Focuses on addressing the unique challenges of handling the "Three Vs" of big data: Volume, Velocity, and Variety.	May focus primarily on structured data or specific data types without the complexities of big data.

Talk about the different tombstone markers used for deletion purposes in HBase.

In HBase, a normal deletion process results in a tombstone marker. The deleted cells become invisible, but the data represented by them is actually removed during compaction.

HBase has three types of tombstone markers:

Version delete marker: It marks a single version of a column for deletion

Column delete marker: It marks all versions of a column

Family delete marker: It sets up all columns of a column family for deletion

Define Term Frequency and Inverse Document Frequency

Term Frequency (TF): Term Frequency measures how frequently a term (word or phrase) appears within a specific document. It quantifies the importance of a term within the document.

Inverse Document Frequency (IDF): Inverse Document Frequency quantifies the rarity or uniqueness of a term across a collection of documents. It measures how much information a term provides based on its distribution in the entire corpus.

Explain following commands with syntax and at least one example of each. (1) copyFromLocal (2) showing the content of outputfile.

copyFromLocal: The copyFromLocal command is used to copy files or directories from the local file system (on your machine) to the Hadoop Distributed File System (HDFS).

Syntax: hdfs dfs -copyFromLocal <local-source> <hdfs-destination>

Example: Let's say you have a file named local-data.txt on your local machine, and you want to copy it to an HDFS directory called /user/hadoop/input:

hdfs dfs -copyFromLocal local-data.txt /user/hadoop/input/

cat: The cat command is used to display the contents of a file stored in HDFS.

Syntax: hdfs dfs -cat <hdfs-source>

Example: Let's say you want to view the contents of a file named sample-data.txt that is stored in HDFS at the path /user/hadoop/input/sample-data.txt:

hdfs dfs -cat /user/hadoop/input/sample-data.txt

Write Map Reduce steps for counting occurrences of specific numbers in the input text file(s). Also write the commands to compile and run the code

Explain following in brief with respect to Mongo DB : 1) Collections and documents 2) Indexing and retrieval

Collections: In MongoDB, a collection is a container for organizing related data documents.. Each collection can hold multiple documents, and each document within a collection can have its own schema or structure. Collections are schema-less, meaning that documents within a collection do not need to have the same fields or structure.

Documents: Documents in MongoDB are individual records or data entries. They are JSON-like objects, which means they consist of key-value pairs. Each document in a collection represents a single data entity. Documents can have varying structures, and you can store different types of data in the same collection. This flexibility makes MongoDB suitable for storing semi-structured or unstructured data.

Indexing: Indexes in MongoDB are data structures that improve the speed of data retrieval operations. They work similarly to indexes in traditional databases and allow MongoDB to quickly locate and access specific documents within a collection

Retrieval: Retrieval in MongoDB involves querying the database to fetch specific documents or data. MongoDB provides a powerful query language for retrieving data. You can use various query operators to filter, sort, and project data as needed.

Write difference between MongoDB and Hadoop.

Features	Hadoop	MongoDB	Cassandra
Built	Hadoop is written in Java.	MongoDB is written in C++	Cassandra is written in Java.
Performance	Hadoop has high latency due to disk access	MongoDB has low latency due to in-memory data access	Cassandra has low latency due to in-memory data access
Database type	Distributed File System	Document-oriented	Column-oriented
Data Management	Batch Processing and Real-time	Real-time data analysis	Real-time
Data Format	Structured and unstructured data	BSON and JSON data	Structured data
Use Cases	ideal for batch processing and is commonly used for big data analytics, ETL (Extract, Transform, Load), and data warehousing.	commonly used for content management, IoT applications, and mobile apps.	Used for time series data, data analytics, and fraud detection.

Explain scaling in MongoDB

Scaling in MongoDB refers to the process of expanding a MongoDB database system to accommodate growing data and user demands. MongoDB offers two primary methods for scaling: horizontal scaling (sharding) and vertical scaling (replication and increasing hardware resources).

Horizontal Scaling (Sharding): Sharding: Sharding is a method of distributing data across multiple servers or clusters. In MongoDB, data is partitioned into smaller chunks, called shards, based on a specified shard key. Each shard is stored on a separate server or replica set, allowing data to be distributed across multiple nodes.

Key Benefits:

Load Balancing: MongoDB's sharding system automatically balances data distribution across shards, preventing uneven workloads on specific servers.

High Availability: Sharded clusters can include replica sets, providing fault tolerance and data redundancy.

Improved Write and Read Scalability: Sharding allows for write and read operations to be distributed across multiple servers, improving overall system performance.
Vertical Scaling: Vertical scaling, also known as scaling up, involves increasing the computing and storage resources of a single MongoDB server. This can be achieved by adding more CPU cores, RAM, or storage capacity to the server.
Key Benefits:
Improved Performance: Adding more resources to a single server can enhance query performance and throughput.

Explain CRUD operations in MongoDB.

https://www.scaler.com/topics/crud-operations-in-mongodb/

Why is RDD better than MapReduce.

- RDD avoids all of the reading/writing to HDFS. By significantly reducing I/O operations, RDD offers a much faster way to retrieve and process data in a Hadoop cluster. In fact, it's estimated that Hadoop MapReduce apps spend more than 90% of their time performing reads/writes to HDFS.
- RDDs are designed for in-memory data processing, which means they can cache and persist data in memory. This leads to significantly faster data processing compared to the disk-based processing used by MapReduce.
- While MapReduce is primarily associated with Java, Spark supports multiple programming languages, including Java, Scala, Python, and R, making it accessible to a broader range of developers.

Which terms are used for table, row, column and table-join in MongoDB?

SQL Terms	MongoDB Terms
database	Database
table	Collection
row	document or BSON document
column	field
index	index
table joins	$lookup, embedded document
primary key	primary key
In SQL, we can specify any unique column or column combination as the primary key.	In MongoDB, we don't need to set the primary key. The _id field is automatically set to the primary key.
aggregation	aggregation pipeline
SELECT INTO NEW_TABLE	$out
MERGE INTO TABLE	$merge
transactions	transactions

What are the common input formats in Hadoop?

TextInputFormat: This is the default input format for Hadoop MapReduce. It reads text files where each line is treated as a separate record. Often used for processing unstructured or semi-structured textual data.
KeyValueTextInputFormat: Similar to TextInputFormat, but it interprets each line as a key-value pair separated by a delimiter. This is useful for processing structured data where data elements are separated by a delimiter. Commonly used for data in formats like CSV or TSV (tab-separated values)
FileInputFormat: It is the base class for all file-based InputFormat. It specifies input directory where data files are present. FileInputFormat also read all files. And, then divides these files into one or more InputSplits.
MultipleInputs: While not a standard input format, the MultipleInputs technique allows you to process data from multiple input formats in a single MapReduce job. This is valuable when dealing with datasets in different formats.

Explain the different modes in which Hadoop run.

Hadoop can run in 3 different modes.
1) Local (Standalone) Mode: In this mode, Hadoop runs as a single Java process on a single machine. It doesn't use the Hadoop Distributed File System (HDFS) and is mainly used for debugging and development. This mode is useful for writing and testing MapReduce jobs on a small dataset without the complexities of a multi-node cluster. It useslocal file system for input and output
- Standalone mode is usually the fastest Hadoop modes as it uses the local file system for all the input and output.
- Here no need to change any configuration files.

2) Pseudo-Distributed Mode: The pseudo-distribute mode is also known as a single-node cluster where both NameNode and DataNode will reside on the same machine. This mode is often used for development and testing purposes.
- Here Changes in configuration files will be required for all the three files- mapred-site.xml, core-site.xml, hdfs-site.xml
- Here HDFS is utilized for input and ouput.

3) Fully-Distributed Mode (Multi-Node Cluster): This is the production mode of Hadoop where multiple nodes will be running. Here data will be distributed across several nodes and processing will be done on each node.
- In this mode typically one machine in the cluster is designated as NameNode and another as Resource Manager exclusively. These are masters. All other nodes act as Data Node and Node Manager. These are the slaves.
- This mode is used for production environments to process large datasets across a cluster of machines.
- This mode offers fully distributed computing capability,, reliability , fault tolerance and scalability.

How can you achieve security in Hadoop?

Security in Hadoop is a critical aspect, especially when dealing with sensitive data and large-scale distributed systems

Kerberos Authentication: Hadoop uses Kerberos for strong authentication. Users and services are required to authenticate themselves before accessing the Hadoop cluster. This ensures that only authorized entities can interact with the cluster.

HDFS and HBase Access Controls: Hadoop components like HDFS and HBase provide access controls that allow administrators to define fine-grained permissions for files, directories, and tables. This ensures that only authorized users can read or write data.

Firewall Configuration: Implement a firewall to restrict access to Hadoop cluster nodes. Limiting access to only trusted IP addresses can help prevent unauthorized access.

Data Encryption: Hadoop supports data encryption at rest and in transit. You can use tools like HDFS Transparent Data Encryption (TDE) for data at rest and SSL/TLS for encrypting data in transit between nodes.

Audit Logs: Enable auditing and maintain comprehensive logs of activities and access to Hadoop cluster resources. Audit logs help in monitoring and identifying security incidents.

What are the differences between Hadoop 2 and Hadoop 3?

S.No.	Feature	Hadoop 2.x	Hadoop 3.x
1	License	Apache 2.0 is used for licensing which is open-source.	Apache 2.0 is used for licensing which is open-source.
2	Minimum supported Java version	JAVA 7 is the minimum compatible version.	JAVA 8 is the minimum compatible version.
3	Fault Tolerance	Replication is the only way to handle fault tolerance which is not space optimized.	Erasure coding is used for handling fault tolerance.
4	Data Balancing	HDFS balancer is used for Data Balancing.	Intra-data node balancer is used which is called via HDFS disk-balancer command-line interface.
5	Storage Scheme	3x Replication Scheme is used.	uses eraser encoding in HDFS.
6	Storage Overhead	200% of HDFS is consumed in Hadoop 2.x	50% used in Hadoop 3.x means we have more space to work.
7	YARN Timeline Service	Uses timeline service with scalability issue.	Improve the time line service along with improving scalability and reliability of this service.
8	Scalability	Limited Scalability, can have upto 10000 nodes in a cluster.	Scalability is improved, can have more then 10000 nodes in a cluster.
9	Default Port Range (32768-61000)	Linux ephemeral port range is used as default, which is failed to bind at startup time.	Ports used are out of this ephemeral port range.
10	Compatible File System.	HDFS(default), FTP, Amazon S3 and Windows Azure Storage Blobs (WASB) file system.	All file systems including Microsoft Azure Data Lake filesystem.
11	Name Node recovery	Manual intervention is needed for the namenode recovery.	No need of Manual intervention for name node recovery.

How is NFS different from HDFS?

Characteristic	NFS (Network File System)	HDFS (Hadoop Distributed File System)
Data Storage	Typically used for centralized, shared storage on a traditional file system.	Designed for distributed storage, optimized for big data, and distributed processing.
Data Distribution	Primarily operates on a single server or network-attached storage (NAS).	Distributes data across a cluster of commodity hardware.
Scalability	Limited scalability and performance with single-point-of-failure concerns.	Scales horizontally by adding more nodes, ensuring high availability and fault tolerance.
Data Replication	Replication is not an inherent feature; relies on external solutions for data redundancy.	Provides built-in data replication for fault tolerance and data durability.
Data Blocks	Organizes data into traditional fixed-size blocks, typically 4 KB to 64 KB.	Uses large fixed-size blocks (typically 128 MB or 256 MB) to optimize data processing.
File Size Limitations	Limited by the capabilities of the underlying file system.	Supports very large files, making it suitable for big data workloads.
Metadata Management	NFS has centralized metadata management, which can be a bottleneck for metadata-intensive workloads.	HDFS utilizes a distributed and scalable metadata architecture, allowing for efficient management of metadata.
Data Processing Paradigm	Designed for general-purpose file access and not specifically optimized for big data processing.	Tailored for batch processing, real-time processing, and various big data applications through MapReduce, Spark, etc.
Data Consistency	Strong consistency model with immediate data consistency.	Provides tunable consistency based on the requirements of the application.
Latency	Low-latency file access for traditional use cases.	Higher-latency access but optimized for batch processing and data locality.
Usage	Suitable for general-purpose file sharing and storage across a network.	Primarily used for big data storage and processing in distributed computing environments.
Security and Access Control	Provides access control at the file and directory level.	Offers role-based access control, fine-grained authorization, and integration with Kerberos for robust security.

Explain the process that overwrites the replication factors in HDFS.

Answer: There are two methods to overwrite the replication factors in HDFS -
Method 1: On File Basis
In this method, the replication factor is changed on the basis of file using Hadoop FS shell. The command used for this is:
$hadoop fs - setrep -w2/my/test_file
Here, test_file is the filename that's replication factor will be set to 2.
Method 2: On Directory Basis
In this method, the replication factor is changed on directory basis i.e. the replication factor for all the files under a given directory is modified.
$hadoop fs -setrep -w5/my/test_dir
Here, test_dir is the name of the directory, the replication factor for the directory and all the files in it will be set to 5.

What will happen with a NameNode that doesn't have any data?

There does not exist any NameNode without data. If it is a NameNode then it should have some sort of data in it.

Explain NameNode recovery process.

- Use the FsImage which is file system metadata replica to start a new NameNode.
- Configure the DataNodes and also the clients to make them acknowledge the newly started NameNode.
- Once the new NameNode completes loading the last checkpoint FsImage which has received enough block reports from the DataNodes, it will start to serve the client.
- In case of large Hadoop clusters, the NameNode recovery process consumes a lot of time which turns out to be a more significant challenge in case of routine maintenance.
- The newly promoted active NameNode ensures that its metadata, including the file system namespace, directory structures, file names, and the mapping of data blocks to DataNodes, is synchronized and up to date.

What are the main differences between NAS (Network-attached storage) and HDFS?

1)NAS stands for Network Attached storage which is a file-level computer data storage server connected to a computer network providing network access to heterogeneous group of clients

HDFS stands for Hadoop distributed file system which is a java based file system that provides scalable and reliable data storage and is designed to span large clusters of commodity hardware.

2)In HDFS data blocks are distributed across the local drives of all machines in a cluster whereas in NAS data is stored on a dedicated server.

3)HDFS includes commodity hardware which will be cost-effective, but NAS is a high-end storage device which is expensive.

4)It includes features like rack-awarenessHDFS, data locality which makes it more scalable and effective then NAS.

What is the Command to format the NameNode?

To format the NameNode in Hadoop's HDFS (Hadoop Distributed File System), you can use the *'hadoop namenode -format'* command. This command initializes the NameNode and the file system's metadata, essentially wiping clean the existing namespace and starting with a fresh, empty file system.

Keep in mind that formatting the NameNode is a critical operation and should be used with caution.

What do you understand by Rack Awareness in Hadoop?

- The Rack is the collection of around 40-50 DataNodes connected using the same network switch. If the network goes down, the whole rack will be unavailable. A large Hadoop cluster is deployed in multiple racks.
- In a large Hadoop cluster, there are multiple racks. Each rack consists of DataNodes. Communication between the DataNodes on the same rack is more efficient as compared to the communication between DataNodes residing on different racks.
- To reduce the network traffic during file read/write, NameNode chooses the closest DataNode for serving the client read/write request. NameNode maintains rack ids of each DataNode to achieve this rack information. This concept of choosing the closest DataNode based on the rack information is known as Rack Awareness.

Why Rack Awareness ?

- To reduce the network traffic while file read/write, which improves the cluster performance.
- To achieve fault tolerance and high availability.

What is the difference between "HDFS Block" and "Input Split"?

HDFS Blockis the physical part of the disk which has the minimum amount of data that can be read/write.

While MapReduce InputSplit is the logical chunk of data created by theInputFormat specified in the MapReduce job configuration.

Logical partition means it will have just the information about blocks address or location.

Input Splits are not fixed in size, and they are typically smaller than HDFS blocks. While HDFS divides large files into fixed-size blocks, typically 128 megabytes (MB) or 256 MB in size.

What are the configuration parameters in a "MapReduce" program?

The main configuration parameters in "MapReduce" framework are:

- Input location of Jobs in the distributed file system
- Output location of Jobs in the distributed file system
- The input format of data
- The output format of data
- The class which contains the map function
- The class which contains the reduce function
- JAR file which contains the mapper, reducer and the driver classes

What is Distributed Cache in a MapReduce Framework?

Distributed Cache in Hadoop is a facility provided by the MapReduce framework. Distributed Cache can cache files when needed by the applications. It can cache read only text files, archives, jar files etc.

Once we have cached a file for our job, Apache Hadoop will make it available on each datanodes where map/reduce tasks are running. Thus, we can access files from all the datanodes in our MapReduce job.

By default, distributed cache size is 10 GB. If we want to adjust the size of distributed cache we can adjust by using local.cache.size.

Advantage:

Single point of failure- As distributed cache run across many nodes. Hence, the failure of a single node does not result in a complete failure of the cache.

What are the Port Numbers for NameNode, Task Tracker, and Job Tracker?

The port number for job tracker is 30, the port number for task tracker is 60, and the port number for Namenode is 70.

What are the different file permissions in HDFS for files or directory levels?

https://www.wikitechy.com/interview-questions/big-data/what-are-the-different-types-of-file-permissions-in-hdfs/

What are the basic parameters of a Mapper?

The four basic parameters of a mapper are LongWritable, text, text and IntWritable. The first two represent input parameters and the second two represent intermediate output parameters.

How Is Hadoop CLASSPATH essential to start or stop Hadoop daemons?

CLASSPATH includes necessary directories that contain jar files to start or stop Hadoop daemons. Hence, setting CLASSPATH is essential to start or stop Hadoop daemons.

However, setting up CLASSPATH every time is not the standard that we follow. Usually CLASSPATH is written inside /etc/hadoop/hadoop-env.sh file. Hence, once we run Hadoop, it will load the CLASSPATH automatically.

Here's how the CLASSPATH is essential:

Class Loading for Hadoop Daemons:Hadoop is a distributed computing framework written in Java, and its daemons (such as NameNode, DataNode, ResourceManager, and NodeManager) are Java processes.

The CLASSPATH is used by these daemons to locate the required Java classes and libraries during their startup.

Starting Hadoop Daemons:When you start Hadoop daemons, the Hadoop scripts use the CLASSPATH to find the necessary Hadoop and HDFS JAR files, as well as any additional libraries that are required for the specific daemons.

The Hadoop scripts set up the environment, including the CLASSPATH, before launching the daemons.

Stopping Hadoop Daemons:When you stop Hadoop daemons, the CLASSPATH is still essential because it helps the shutdown scripts locate the required classes and libraries for proper termination.

Configuration Files:Hadoop configuration files, such as hadoop-env.sh or yarn-env.sh, often include settings related to the CLASSPATH. These files may define additional directories or JAR files to be included in the CLASSPATH.

Custom Libraries and Dependencies:If you have custom libraries or dependencies that your Hadoop application relies on, you need to ensure that they are included in the CLASSPATH to avoid runtime errors.

Why is HDFS only suitable for large data sets and not the correct tool to use for many small files?

Hadoop Distributed File System (HDFS) is optimized for handling large data sets and is not the ideal choice for managing many small files. Here are some reasons why HDFS is not well-suited for scenarios with numerous small files:\

Block Size and Storage Efficiency:HDFS stores data in blocks, typically with a default block size of 128 MB or 256 MB. If a file is smaller than the block size, it still occupies a full block, leading to inefficient storage utilization.

With many small files, the storage efficiency decreases, as each small file may occupy a full block, resulting in wasted storage space.

Metadata Overhead:HDFS relies on a master-slave architecture with a central NameNode that manages metadata such as file names, permissions, and block locations. Each file and block in HDFS contributes to the metadata overhead.

With many small files, the overhead of storing metadata for each file becomes significant. The NameNode's memory is finite, and a large number of small files can exhaust this memory, potentially leading to performance issues.

Increased NameNode Load:Operations involving metadata, such as listing or modifying directories with many small files, can place a heavy load on the NameNode. This increased load can lead to slower response times and reduced overall system performance.

Why do we need Data Locality in Hadoop? Explain.

https://tutorials.freshersnow.com/map-reduce-tutorial/data-locality-in-hadoop-mapreduce/

DFS can handle a large volume of data then why do we need Hadoop framework?

The function of Distributes File System is to partitions the data, store and manage the data across different machines. DFS can handle the large volume of data, but Hadoop framework will help to process the large amount of data.

- Large data is divided into several blocks and stored in different commodity hardware (storing data into distributed way).
- Lets take an example in general:

If we want to process those data, first we will go to commodity hardware and will copy the data to the processing unit and finally, we will do the processing.

This process has some complications, for large data.

- When transferring large data some data may lose and make trouble.
- But if we use HADOOP, we don't need to take the data to the processing unit.
- Instead, we will take our processing unit to the commodity hardware, where our data is stored.
- Processing can be done over there and we can take the output.

This is how Hadoop framework is more beneficial then DFS.

How does HDFS Index Data blocks? Explain.

What are Edge Nodes in Hadoop?

Hadoop Gateway or edge node is a node that connects to the Hadoop cluster, but does not run any of the daemons. The purpose of an edge node is to provide an access point to the cluster and prevent users from a direct connection to critical components such as Namenode or Datanode.Another important reason for its use is the data distribution across the cluster.

- The EdgeNode is the access point for the external applications, tools, and users that need to utilize the Hadoop environment. The EdgeNode sits between the Hadoop cluster and the corporate network to provide access control, policy enforcement, logging, and gateway services to the Hadoop environment. A typical Hadoop environment will have a minimum of one EdgeNode and more based on performance needs. The Edge Node does not have to be part of the cluster. If the edge node is outside the Hadoop cluster

Write the use and syntax of following HDFS commands:

i. put ii. expunge iii. chmod iv. get;

put: The put command is used to copy a local file or directory to the Hadoop Distributed File System (HDFS).

Syntax: hadoop fs -put <localsrc> <dst>

Expunge: The expunge command is used to permanently delete all files in the trash. In HDFS, when you delete a file, it goes to a trash directory by default, and the expunge command can be used to empty this trash.

Syntax: hadoop fs -expunge

Chmod: The chmod command is used to change the permissions of files or directories on HDFS.

Syntax: hadoop fs -chmod [-R] <mode> <path>

Get:The get command is used to copy files from HDFS to the local file system.

Syntax: hadoop fs -get <src> <localdst>

Write down the goals of HDFS.

Fast recovery from hardware failures: Because one HDFS instance may consist of thousands of servers, failure of at least one server is inevitable. HDFS has been built to detect faults and automatically recover quickly.

Hardware efficiency - When large datasets are involved it can reduce the network traffic and increase the processing speed.

Detecting faults - HDFS should have technology in place to scan and detect faults quickly and effectively as it includes a large number of commodity hardware. Failure of components is a common issue.

Manage large datasets - Organizing and storing datasets can be a hard talk to handle. HDFS is used to manage the applications that have to deal with huge datasets. To do this, HDFS should have hundreds of nodes per cluster.

How type of data affects data serialization

The type of data affects data serialization in terms of complexity, size, compatibility, performance, and human readability.

Structured data with a defined schema is well-suited for serialization, while unstructured data presents challenges.

Binary serialization is efficient for large datasets and performance-critical tasks, while textual serialization offers cross-platform compatibility and human readability for debugging.

Justify: "SPARK is faster than MapReduce".

In-Memory Processing:

Spark: Spark performs in-memory processing, keeping intermediate data in memory rather than writing it to disk after each stage of computation. This reduces the overhead of reading and writing to disk, resulting in faster data processing.

MapReduce: MapReduce, in contrast, writes intermediate results to disk between Map and Reduce stages, incurring additional I/O operations, which can be a performance bottleneck.

Ease of Use and High-Level APIs:

Spark: Spark provides high-level APIs in Java, Scala, Python, and R, making it more accessible to a broader audience. The use of expressive APIs can lead to more concise and readable code.

MapReduce: MapReduce requires developers to write more low-level and verbose code in Java, which can be more error-prone and time-consuming.

DAG : The concept of DAG being introduced in spark has its own benefits. Recompilation of failed stages being back tracked with the help of DAG has reduced the burden of starting the computation again from the beginning.

Spark query optimizer: Especially while working with spark data frames we come across an internal query optimizer which optimizes the queries run across the data while processing it. Spark comes up with various query plans and choses the most optimal one for its execution. We can check the query plan execution as well to get a detailed view of how the spark tries to optimize the queries before the execution.

Write a brief short note on: Spark Unified Stack

The Spark Unified Stack refers to the comprehensive set of libraries and components provided by Apache Spark, which collectively enable a unified and versatile platform for various data processing and analytics tasks. This unified approach allows users to perform batch processing, interactive queries, machine learning, and stream processing within a single ecosystem.

Key components of Spark unified Stack is : Spark core, Spark SQL, Spark Streaming, MLib, GraphX etc.

Explain various data insertion techniques in HIVE with example.

https://myitlearnings.com/hive-queries-inserting-data-into-hive-table/

Compare Distributed File System (DFS), Google File System (GFS) with Hadoop Distributed File System (HDFS)?

Elaborate the working of Map-Reduce Algorithm.

https://www.tutorialspoint.com/map_reduce/map_reduce_algorithm.htm

Explain "Combiner Phase" in MapReduce.

https://www.codingninjas.com/studio/library/mapreduce-combiner

Explain MongoDB sharding process

https://www.geeksforgeeks.org/mongodb-replication-and-sharding/

Explain Collections and documents, Indexing and retrieval wrt to MongoDB.

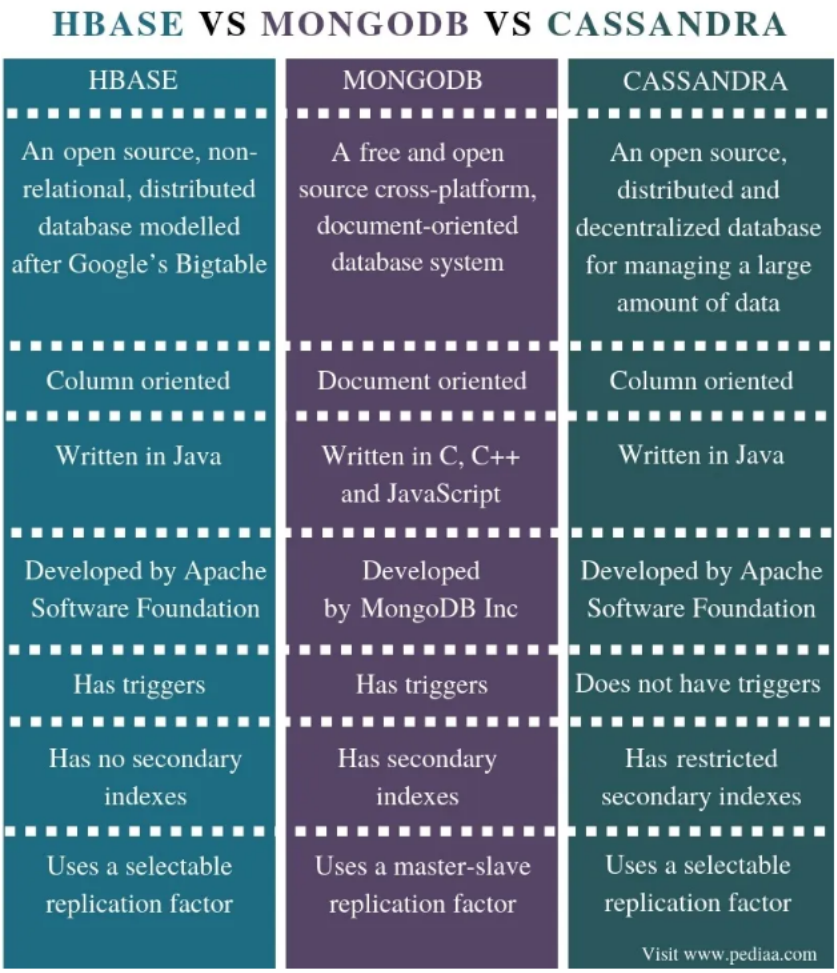
Collections: A collection in MongoDB is a group of MongoDB documents. Collections are schema-less, meaning each document in a collection can have a different structure. This flexibility allows for the storage of diverse types of data within the same collection.

Documents: A document in MongoDB is a JSON-like BSON (Binary JSON) object, where BSON is a binary-encoded serialization of JSON-like documents. Documents are the basic unit of data in MongoDB and are stored in collections. They consist of key-value pairs and can include nested documents or arrays.

Indexing: Indexes in MongoDB are similar to indexes in relational databases. They provide a way to efficiently access and retrieve data from collections. MongoDB automatically creates an index on the _id field for every collection, which is the primary key. Developers can create custom indexes on specific fields to improve query performance. Indexes are stored in a B-tree data structure, making data retrieval more efficient.

Retrieval: Retrieval in MongoDB involves fetching and querying documents from collections based on specified criteria. MongoDB provides a flexible and powerful querying system. Like there is find method in mongoDB is there which is used for Retrieval of data.

Compare Cassandra with HBase and MongoDB.



Compare RDBMS with Cassandra

S.No.	CASSANDRA	RDBMS
1.	Cassandra is a high performance and highly scalable distributed NoSQL database management system.	RDBMS is a Database management system or software which is designed for relational databases.
2.	Cassandra is a NoSQL database.	RDBMS uses SQL for querying and maintaining the database.
3.	It deals with unstructured data.	It deals with structured data.
4.	It has a flexible schema.	It has fixed schema.
5.	Cassandra has peer-to-peer architecture with no single point of failure.	RDBMS has master-slave core architecture means a single point of failure.
6.	Cassandra handles high volume incoming data velocity.	RDBMS handles moderate incoming data velocity.
7.	In RDBMS there is limited data source means data come from many locations.	In Cassandra there are various data source means data come from one/few location.
8.	It supports simple transactions.	It supports complex and nested transactions.
9.	In Cassandra the outermost container is Keyspace.	In RDBMS the outermost container is database.
10.	Cassandra follows decentralized deployments.	RDBMS follows centralized deployments.
11.	In Cassandra data written in many locations.	In RDBMS mainly data are written in one location.
12.	In Cassandra row represents a unit of replication.	In RDBMS row represents a single record.
13.	In Cassandra column represents a unit of storage.	In RDBMS column represents an attribute.
14.	In Cassandra, relationships are represented using collections.	In RDBMS relationships are represented using keys and join etc.

Compare RDBMS with Neo4j

Neo4j is example of Graph database

Feature	RDBMS	Graph Database
Data Model	Tables with rows and columns	Graph (nodes and relationships)
Schema	Structured schema with predefined tables	Dynamic or schema-free
Query Language	SQL (Structured Query Language)	Cypher Query Language (or other graph-specific query languages)
Relationships	Modeled using foreign keys and joins	Relationships are first-class citizens, no need for joins
Flexibility	Fixed schema, less flexible	Dynamic schema, more flexible
Scalability	Vertical scalability (scaling up)	Horizontal scalability (scaling out)
Performance	Excellent for complex queries with joins	Excellent for graph traversal and pattern matching
ACID Compliance	Strong ACID compliance	ACID compliant, but may be tunable for performance
Use Cases	Structured data, complex queries	Highly connected data, complex relationships, graph-based queries
Transaction Support	ACID transactions	ACID transactions
Indexes	B-tree indexes on columns	Indexes on nodes and relationships
Normalization	Higher degree of normalization	Denormalization is common in graph modeling
Example Applications	Enterprise applications, reporting	Social networks, recommendation engines, fraud detection
Joins	Common and used for complex queries	Relationships are traversed directly, no need for joins
Community Support	Mature, large community	Growing community

Explain the working Cassandra with proper steps and diagram.

https://www.javatpoint.com/cassandra-architecture

Explain the working Neo4j with proper steps and diagram.

Explain the working MongoDB with proper steps and diagram.

https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/

Which are the features of BigTable and Titan InitGraph have been combined in Cassandra?

Differentiate between Apache Pig and Hive.

Difference between Pig and Hive :

S.No.	Pig	Hive
1.	Pig operates on the client side of a cluster.	Hive operates on the server side of a cluster.
2.	Pig uses pig-latin language.	Hive uses HiveQL language.
3.	Pig is a Procedural Data Flow Language.	Hive is a Declarative SQLish Language.
4.	It was developed by Yahoo.	It was developed by Facebook.
5.	It is used by Researchers and Programmers.	It is mainly used by Data Analysts.
6.	It is used to handle structured and semi-structured data.	It is mainly used to handle structured data.
7.	It is used for programming.	It is used for creating reports.
8.	Pig scripts end with .pig extension.	In Hive, all extensions are supported.
9.	It does not support partitioning.	It supports partitioning.
10.	It loads data quickly.	It loads data slowly.
11.	It does not support JDBC.	It supports JDBC.
12.	It does not support ODBC.	It supports ODBC.
13.	Pig does not have a dedicated metadata database.	Hive makes use of the exact variation of dedicated SQL-DDL language by defining tables beforehand.
14.	It supports Avro file format.	It does not support Avro file format.
15.	Pig is suitable for complex and nested data structures.	Hive is suitable for batch-processing OLAP systems.
16.	Pig does not support schema to store data.	Hive supports schema for data insertion in tables.
17.	It is very easy to write UDFs to calculate matrices.	It does support UDFs but is much hard to debug.

Which are the problems related to Map-Reduce data storage?

Explain principles of schema design in MongoDB.

Explain Job Scheduling in Map-Reduce.

https://www.geeksforgeeks.org/hadoop-schedulers-and-types-of-schedulers/