Big Data Assignment

1.Difference between map-reduce and Spark?

Map-Reduce and Spark are both distributed computing systems used for processing large amounts of data in parallel across a cluster of computers, but there are several differences between them. Here are a few key differences:

1. Processing Model: Map-Reduce uses a batch processing model where the data is processed in batches, whereas Spark supports both batch and streaming processing. This means that Spark can process data in real-time as it arrives, whereas Map-Reduce cannot.
2. Data Processing Speed: Spark is generally faster than Map-Reduce due to its ability to cache data in memory and process it in-memory rather than reading from and writing to disk. Spark also uses a DAG (Directed Acyclic Graph) execution engine, which optimizes the processing of data across a cluster of computers.
3. API: Spark provides a richer set of APIs and libraries for data processing and analysis than Map-Reduce, including support for SQL, machine learning, and graph processing. This makes it easier to use Spark for a wide range of use cases.
4. Memory Management: Spark has a more efficient memory management system than Map-Reduce, which allows it to cache data in memory and reuse it across multiple tasks. This makes Spark more efficient in handling iterative algorithms that require multiple passes over the data.
5. Fault Tolerance: Both Map-Reduce and Spark are fault-tolerant, but Spark has a more efficient fault-tolerance mechanism, which allows it to recover from node failures more quickly and efficiently than Map-Reduce. This is because Spark stores the lineage information of RDDs (Resilient Distributed Datasets), which helps in reconstructing the lost data in case of a node failure.

In summary, while both Map-Reduce and Spark are used for distributed data processing, Spark offers faster processing, a richer set of APIs, better memory management, and more efficient fault-tolerance mechanisms.

2.Difference between Flume and Scoop?

Flume and Sqoop are both tools used for data ingestion in Hadoop, but they differ in their approach to data ingestion and the types of data sources they support. Here are some key differences between Flume and Sqoop:

1. Data Sources: Flume is designed primarily for streaming data ingestion from sources such as log files, social media feeds, and machine-generated data, whereas Sqoop is designed for bulk data ingestion from structured data sources such as relational databases.
2. Approach to Data Ingestion: Flume uses a push-based approach to data ingestion, where the data source pushes data to the destination (Hadoop cluster). Sqoop, on the other hand, uses a pull-based approach where data is pulled from the source (relational database) and then loaded into Hadoop.
3. Parallelism: Flume is designed to ingest data in parallel across multiple agents and channels, which allows for efficient data ingestion from high-volume data sources. Sqoop, on the other hand, is designed for bulk data transfer in parallel, but the parallelism is limited by the number of mappers configured for a Sqoop job.
4. Data Transformation: Flume allows for data transformation and enrichment in-flight through the use of interceptors, which can modify or add metadata to the data. Sqoop, on the other hand, does not provide built-in support for data transformation, although transformation can be performed using external tools or scripts.
5. Integration with Hadoop Ecosystem: Flume integrates well with other components in the Hadoop ecosystem, such as HDFS, HBase, and Spark Streaming, making it easier to use Flume in various use cases. Sqoop, on the other hand, is primarily designed to work with Hadoop's HDFS and MapReduce components.

In summary, while Flume and Sqoop are both used for data ingestion in Hadoop, they differ in their approach to data ingestion, types of data sources supported, parallelism, data transformation, and integration with other components in the Hadoop ecosystem.

3.For below use case

* You have database of 3 employment website. All resumes are in same template.
* Your task is to make 3 sheet. First one to extract the important data, second one is what transformation you perform, last one Entity relationship model.

Source | Full Name | Address | Phone Number | Email Id | Skills | Experience | Projects Worked

Assessment – Every Sheet will have 10 points.

What technologies you would use to process them.

Collecting the data from different sources like LinkedIn, Naukri and Monster

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Source | Full Name | Address | Phone Number | Email Id | Skills | Experience | Project Worked |
| LinkedIn | Teja Krishna | Kadapa | 99XXXXX | teja@gmail.com | Python, SQL | 0 | None |
| Naukri | Prashant | Rajamundry | 9XXXX | pra@gmail.com | SQL, python | 5 | Fake Account Detention |
| Monster | Jyosthna | Puttaparti | 99XXX | joo@gmail.com | MI, AI, Deep learning | 3 | Model Building |
| LinkedIn | Tameem | Amalapuram | 99XXXX | tams@gmail.com | SQL | 0 | None |
| LinkedIn | Mahendra | Hyderabad | 80XXX | mahi@gmail.com | DS, AI | 1 | ML models |
| LinkedIn | Satyam | Patna | 678XX | sat@gmail.com | SQL, PowerBi | 2 | Finance  Dashboard |
| Monster | Manoj | Bangalore | 567XXX | m@gmail.com | Python, SQL | 0 | None |
| LinkedIn | Teja | Kadapa | 999XX | teja@gmail.com | Python, SQL | 0 | None |
| Naukri | Lokesh | KGF | 888XXX | LCU@gmail.com | Looker, Tableau | 2 | Finance Dashboards |

Extracting the important data from the above raw data,

As the Source and Address columns are not important for the selection criteria we can filter that.

Important Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Full Name | Phone Number | Email Id | Skills | Experience | Project Worked |
| Teja Krishna | 99XXXXX | teja@gmail.com | Python, SQL | 0 | None |
| Prashant | 9XXXX | pra@gmail.com | SQL, python | 5 | Fake Account Detention |
| Jyosthna | 99XXX | joo@gmail.com | MI, AI, Deep learning | 3 | Model Building |
| Tameem | 99XXXX | tams@gmail.com | SQL | 0 | None |
| Mahendra | 80XXX | mahi@gmail.com | DS, AI | 1 | ML models |
| Satyam | 678XX | sat@gmail.com | SQL, PowerBi | 2 | Finance  Dashboard |
| Manoj | 567XXX | m@gmail.com | Python, SQL | 0 | None |
| Teja | 999XX | teja@gmail.com | Python, SQL | 0 | None |
| Lokesh | 888XXX | LCU@gmail.com | Looker, Tableau | 2 | Finance Dashboards |

Transformations Made on Data

* Remove unnecessary columns like Source, Address.
* Filter Out duplicate entries from multiple sources based on Email-Id.
* Delete all entries where skills don’t match keywords.
* If the hiring is for 0-4years experienced candidates, above 4 years of experience will be skipped.
* Filter out candidates who have more than 4 years of experience."

**Categorizing the obtained data**

Persons: (Name, Email (Primary Key), Phone. No, Experience)

Person Skills (Email (foreign Key), Skills)

Person Projects (Email (Foreign Key),Projects)

Persons Table:

|  |  |  |
| --- | --- | --- |
| Full Name | Phone Number | Email Id |
| Teja Krishna | 99XXXXX | teja@gmail.com |
| Jyostana | 99XXX | joo@gmail.com |
| Tameem | 99XXXX | tams@gmail.com |
| Mahendra | 80XXX | mahi@gmail.com |
| Satyam | 678XX | sat@gmail.com |
| Manoj | 567XXX | m@gmail.com |
| Lokesh | 888XXX | LCU@gmail.com |

Person Skills:

|  |  |
| --- | --- |
| Email Id | Skills |
| teja@gmail.com | Python, SQL |
| joo@gmail.com | MI, AI, Deep learning |
| tams@gmail.com | SQL |
| mahi@gmail.com | DS, AI |
| sat@gmail.com | SQL, PowerBi |
| m@gmail.com | Python, SQL |
| LCU@gmail.com | Looker, Tableau |

Person Projects:

|  |  |
| --- | --- |
| Email Id | Project Worked |
| teja@gmail.com | None |
| joo@gmail.com | Model Building |
| tams@gmail.com | None |
| mahi@gmail.com | ML models |
| sat@gmail.com | Finance Dashboard |
| m@gmail.com | None |
| LCU@gmail.com | Finance Dashboards |

Entity Relationship Model:

Diagram

Description automatically generated

Technologies Used:

**Data Extraction/Ingestion Tools:**

* **Flume**
* **Kafka**
* **Sqoop**

**Data Transformation/Processing:**

* **Map-Reduce**
* **Spark**

Flume, Sqoop, and Kafka are three popular technologies used for data ingestion and transfer in big data environments.

1. Apache Flume is an open-source data ingestion tool that is used to collect, aggregate, and transfer large volumes of log data and event data from various sources to a centralized repository, such as Hadoop HDFS. Flume provides a distributed, fault-tolerant architecture, which ensures high availability and reliability of the data transfer process. Flume supports various data sources and sinks, including log files, Twitter, and syslog, and can be extended with custom plugins.
2. Apache Sqoop is another open-source tool used for transferring bulk data between Hadoop and structured data sources, such as relational databases, data warehouses, and NoSQL databases. Sqoop supports various data import and export operations, including incremental data transfers, parallel data transfers, and data compression. Sqoop provides a command-line interface and supports various Hadoop distributions, such as Cloudera, Hortonworks, and MapR.
3. Apache Kafka is a distributed, scalable, and fault-tolerant messaging system that is used to handle large volumes of real-time data streams. Kafka provides a high-throughput, low-latency platform for data ingestion, processing, and analysis. Kafka supports various data sources and sinks, including log files, IoT devices, social media platforms, and messaging systems. Kafka provides a publish-subscribe model, where producers publish data to one or more topics, and consumers subscribe to those topics and consume data as it is produced. Kafka supports horizontal scaling, replication, and fault tolerance, which ensures high availability and reliability of the data transfer process.

Overall, Flume, Sqoop, and Kafka are essential tools for big data environments, where data ingestion, processing, and analysis require scalable, fault-tolerant, and high-throughput solutions. Each tool has its own strengths and weaknesses, and choosing the right tool for the job depends on the specific requirements of the project.

MapReduce and Spark are two popular distributed computing frameworks used for processing large volumes of data in big data environments.

1. MapReduce is a programming model and software framework introduced by Google in 2004. It is used to process large volumes of data in a distributed manner by splitting the data into smaller chunks and processing them in parallel across multiple nodes in a cluster. The MapReduce framework consists of two main phases: the map phase and the reduce phase. The map phase applies a user-defined function to each input data record and generates key-value pairs as output. The reduce phase combines the key-value pairs with the same key into a single output record. MapReduce is widely used in big data environments, and its implementation is available in various tools such as Apache Hadoop, Apache Spark, and Amazon Elastic MapReduce (EMR).
2. Apache Spark is a distributed computing framework that was introduced in 2012 as an open-source project. It is designed to provide a more flexible, efficient, and interactive processing model for big data than MapReduce. Spark is based on the Resilient Distributed Dataset (RDD) abstraction, which provides a fault-tolerant and scalable data structure for distributed computing. Spark supports various programming languages, including Java, Scala, and Python, and provides a wide range of APIs for data processing, including batch processing, stream processing, machine learning, and graph processing. Spark also provides a memory-based caching mechanism, which enables faster processing of frequently accessed data.

Overall, MapReduce and Spark are both powerful distributed computing frameworks that provide scalable and fault-tolerant solutions for processing large volumes of data in big data environments. However, Spark provides a more flexible and efficient processing model than MapReduce, which makes it a popular choice for big data processing applications.