**Assignment-4**

|  |
| --- |
|  |

**Q.1) Briefly Explain the architecture of Pig.**

* **The Architecture of Apache Pig**

1. **Parser:**Any pig scripts or commands in the grunt shell are handled by the parser. Parse will perform checks on the scripts like the syntax of the scripts, do type checking and perform various other checks. These checks will give output in a Directed Acyclic Graph (DAG) form, which has a pig Latin statements and logical operators. The DAG will have nodes that are connected to different edges, here our logical operator of the scripts are nodes and data flows are edges.
2. **Optimizer:**As soon as parsing is completed and DAG is generated, It is then passed to the logical optimizer to perform logical optimization like projection and pushdown. Projection and pushdown are done to improve query performance by omitting unnecessary columns or data and prune the loader to only load the necessary column.
3. **Compiler:**The optimized logical plan generated above is compiled by the compiler and generates a series of Map-Reduce jobs. Basically compiler will convert pig job automatically into MapReduce jobs and exploit optimizations opportunities in scripts, due this programmer doesn’t have to tune the program manually. As pig is a data-flow language its compiler can reorder the execution sequence to optimize performance if the execution plan remains the same as the original program.
4. **Execution Engine:**Finally, all the MapReduce jobs generated via compiler are submitted to Hadoop in sorted order. In the end, MapReduce’s job is executed on Hadoop to produce the desired output.
5. **Execution Mode:**Pig works in two types of execution modes depend on where the script is running and data availability :

* **Local Mode:**Local mode is best suited for small data sets. Pig is implemented here on single JVM as all files are installed and run on localhost due to this parallel mapper execution is not possible. Also while loading data pig will always look into the local file system.
* **MapReduce Mode (MR Mode):**In MapReduce, the mode programmer needs access and setup of the Hadoop cluster and HDFS installation. In this mode data on which processing is done is exists in the HDFS system. After execution of pig script in MR mode, pig Latin statement is converted into Map Reduce jobs in the back-end to perform the operations on the data. By default pig uses Map Reduce mode, hence we don’t need to specify it using the -x flag.

**Q.2) List out the benefits of Pig?**

* **Pig provides several benefits that make it a popular choice for analyzing large datasets in Apache Hadoop. Here are some of the key benefits of Pig:**

1. **Ease of Use:** Pig offers a high-level scripting language called Pig Latin, which is designed to be simple and easy to understand. It provides a familiar SQL-like syntax along with procedural programming constructs, making it accessible to users with different levels of programming expertise.
2. **Data Transformation:** Pig simplifies the process of performing complex data transformations on large datasets. It provides a rich set of built-in operators and functions for filtering, joining, grouping, aggregating, and manipulating data. These transformations can be expressed using a few lines of Pig Latin code, reducing the need for writing complex MapReduce jobs.
3. **Abstraction of MapReduce:** Pig abstracts away the complexities of writing MapReduce programs. Users can focus on expressing the data transformations they need, and Pig automatically translates the Pig Latin scripts into optimized MapReduce jobs. This abstraction simplifies the development process and reduces the learning curve for working with Hadoop.
4. **Optimization**: Pig incorporates a query optimizer that analyzes Pig Latin scripts and optimizes the execution plan. It performs optimizations such as operator fusion, projection pushdown, and filter pushdown to minimize the amount of data processed and improve overall performance. This optimization leads to faster execution times and more efficient resource utilization.
5. **Extensibility:** Pig is designed to be extensible, allowing users to create custom functions and load data from various sources. It provides a rich ecosystem of user-defined functions (UDFs) that can be written in Java, Python, or other languages, enabling developers to incorporate their own logic and algorithms into Pig scripts.
6. **Integration with Hadoop Ecosystem:** Pig seamlessly integrates with the Apache Hadoop ecosystem. It can read and write data from and to HDFS, as well as interact with other components of the Hadoop stack such as Hive, HBase, and Spark. This integration enables users to leverage the capabilities of these systems and combine them with Pig for more advanced data processing and analytics tasks.
7. **Scalability:** Pig leverages the distributed processing capabilities of Hadoop, allowing it to scale horizontally across a cluster of machines. It can efficiently process large datasets by dividing the workload into smaller tasks that run in parallel across multiple nodes. This scalability makes Pig suitable for handling big data workloads.

**Q.3) Explain working of Hive with proper steps and diagram?**

|  |
| --- |
| How Hive Works |

|  |  |
| --- | --- |
| **Step No.** | **Operation** |
| 1 | **Execute Query**: The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute. |
| 2 | **Get Plan**: The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query. |
| 3 | **Get Metadata**: The compiler sends metadata request to Metastore (any database). |
| 4 | **Send Metadata**: Metastore sends metadata as a response to the compiler. |
| 5 | **Send Plan**: The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete. |
| 6 | **Execute Plan** :The driver sends the execute plan to the execution engine. |
| 7 | **Execute Job**: Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job. |
| 7.1 | **Metadata Ops**: Meanwhile in execution, the execution engine can execute metadata operations with Metastore. |
| 8 | **Fetch Result**: The execution engine receives the results from Data nodes. |
| 9 | **Send Results**: The execution engine sends those resultant values to the driver. |
| 10 | **Send Results**: The driver sends the results to Hive Interfaces. |

**Q.4) What do you mean by HiveQL data definition language explain any three Hive QL DDL command with its syntax and example?**

* HiveQL (Hive Query Language) is a SQL-like language used in Apache Hive for querying and managing data stored in Hadoop. It is specifically designed to work with structured and semi-structured data and provides a familiar SQL-like syntax for data processing tasks. HiveQL includes several types of SQL commands, including Data Definition Language (DDL) commands. DDL commands are used for defining and managing the structure and properties of database objects such as tables, databases, partitions, views, and functions. These commands allow users to create, modify, and delete database objects in Hive. Here are three commonly used HiveQL DDL commands along with their syntax and examples:

1. **CREATE TABLE:** The CREATE TABLE command is used to create a new table in Hive. It defines the table's structure, column names, data types, and other properties.

|  |
| --- |
| **Here's the syntax:**  CREATE TABLE [IF NOT EXISTS] table\_name  (  column1\_name data\_type,  column2\_name data\_type,  ...  )  [PARTITIONED BY (partition\_column1 data\_type, partition\_column2 data\_type, ...)]  [ROW FORMAT row\_format]  [STORED AS file\_format] |
| **Example:**  CREATE TABLE employees (  id INT,  name STRING,  age INT,  department STRING  )  ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  STORED AS TEXTFILE; |

This example creates a table named "employees" with four columns: "id" (INT), "name" (STRING), "age" (INT), and "department" (STRING). The table uses a comma-separated values (CSV) file format.

1. **ALTER TABLE:** The ALTER TABLE command is used to modify the structure or properties of an existing table in Hive. It can be used to add or drop columns, change column names or data types, add or drop partitions, and more.

|  |
| --- |
| **Here's the syntax:**  ALTER TABLE table\_name  [ADD | DROP] COLUMN column\_name data\_type  [RENAME TO new\_table\_name]  [PARTITIONED BY (partition\_column1 data\_type, partition\_column2 data\_type, ...)] |
| **Example:.**  ALTER TABLE employees  ADD COLUMN salary DECIMAL(10,2),  DROP COLUMN age,  RENAME TO staff; |

In this example, the ALTER TABLE command is used to add a new column "salary" with a decimal data type, drop the column "age", and rename the table from "employees" to "staff".

1. **DROP TABLE:** The DROP TABLE command is used to delete a table and its data from Hive. It permanently removes the table and all associated metadata.

|  |
| --- |
| **Here's the syntax:**  DROP TABLE [IF EXISTS] table\_name; |
| **Example:**  DROP TABLE employees; |

This example drops the "employees" table from Hive. If the "IF EXISTS" clause is included, it prevents an error from being thrown if the table doesn't exist. These are just three examples of HiveQL DDL commands. HiveQL provides more DDL commands for managing databases, partitions, views, and other database objects. These commands enable users to define and manipulate the structure and properties of their data stored in Hive.

**Q.5) Write a Short Note?**

**A) Explain meta store in hive:** Metastore is the central repository of Apache Hive metadata. It stores metadata for Hive tables (like their schema and location) and**partitions**in a relational database. It provides client access to this information by using metastore service API.

Hive metastore consists of two fundamental units:

1. A service that provides metastore access to other Apache Hive services.
2. Disk storage for the Hive metadata which is separate from **HDFS** storage.

## Hive Metastore Modes: There are three modes for Hive Metastore deployment

|  |
| --- |
| Embedded Deployment mode for Hive Metastore |

1. **Embedded Metastore:** In Hive by default, metastore service runs in the same JVM as the Hive service. It uses embedded derby database stored on the local file system in this mode. Thus both metastore service and hive service runs in the same JVM by using embedded Derby Database. But, this mode also has limitation that, as only one embedded Derby database can access the database files on disk at any one time, so only one Hive session could be open at a time.If we try to start the second session it produces an error when it attempts to open a connection to the metastore. So, to allow many services to connect the Metastore, it configures Derby as a network server. This mode is good for unit testing. But it is not good for the practical solutions.

|  |
| --- |
| Local Deployment mode for Hive Metastore |

1. **Local Metastore:** Hive is the data-warehousing framework, so hive does not prefer single session. To overcome this limitation of Embedded Metastore, for **Local Metastore** was introduced. This mode allows us to have many Hive sessions i.e. many users can use the metastore at the same time. We can achieve by using any JDBC compliant like MySQL which runs in a separate JVM or different machines than that of the Hive service and metastore service which are running in the same JVM. This configuration is called as local metastore because metastore service still runs in the same process as the Hive. But it connects to a database running in a separate process, either on the same machine or on a remote machine.

|  |
| --- |
| Remote deployment mode for Hive Metastore |

1. **Remote Metastore:** In this mode, metastore runs on its own separate JVM, not in the Hive service JVM. If other processes want to communicate with the metastore server they can communicate using Thrift Network APIs. We can also have one more metastore servers in this case to provide more availability. This also brings better manageability/security because the database tier can be completely firewalled off. And the clients no longer need share database credentials with each Hiver user to access the metastore database.

**B) Explain storage mechanisms:** In Apache Hive, the storage mechanism refers to how data is stored and organized within Hive tables. Hive provides flexibility in choosing different storage formats and mechanisms based on the specific requirements of the data and the desired performance characteristics. Here are some common storage mechanisms used in Hive:

1. **Text File Storage:** Text File storage is the default storage mechanism in Hive. It stores data in plain text files, typically in delimited formats such as CSV (comma-separated values) or TSV (tab-separated values). Text File storage is widely supported and allows for easy data interchange with other systems. However, it may not provide optimal performance for large-scale data processing and querying due to the lack of built-in indexing and compression.
2. **SequenceFile Storage:** SequenceFile storage in Hive stores data in a binary format known as SequenceFile. This format is optimized for high-performance reading and writing in Hadoop. SequenceFiles can be compressed and support block-level compression, which reduces storage space and improves I/O performance. SequenceFile storage is suitable for scenarios where fast data access and efficient compression are important.
3. **ORC (Optimized Row Columnar) File Storage:** ORC File storage is a columnar storage format specifically designed for Hive. It organizes data in a columnar fashion, allowing for highly efficient compression, predicate pushdown, and column-level statistics. ORC files provide improved query performance by reading only the relevant columns and skipping unnecessary data. ORC File storage is widely used in Hive for data warehousing and analytics workloads.
4. **Parquet File Storage:** Parquet File storage is another columnar storage format supported by Hive. It offers similar benefits to ORC files, such as efficient compression, predicate pushdown, and column-level statistics. Parquet files are designed to be highly optimized for query performance and are compatible with various big data processing frameworks. Parquet File storage is commonly used in Hive for analytical workloads that require high performance and efficient columnar storage.
5. **Avro File Storage:** Avro File storage in Hive uses the Avro data serialization system. Avro files store data in a compact binary format, providing efficient storage and serialization/deserialization performance. Avro supports schema evolution, allowing for changes to the table schema without requiring modification of the existing data. Avro File storage is suitable for scenarios where schema flexibility and compatibility are important.

**Q.6) Explain in short and Hive shell, Hive metastore, HBase.**

* **Hive Shell:** Hive Shell is an interactive command-line interface provided by Hive. It allows users to interact with Hive by executing HiveQL commands and managing Hive-related tasks.
* The Hive Shell provides a convenient way to submit queries, create and manage tables, load data, run administrative tasks, and retrieve query results. It accepts commands written in HiveQL, which is a SQL-like language specifically designed for querying and analyzing structured data stored in Hadoop.
* The Hive Shell provides a prompt where users can enter commands and view the results. It supports features such as command history, tab completion, and script execution, making it easy to work with Hive. Users can write HiveQL queries directly in the shell and execute them, or they can save queries in script files and run the files within the shell.
* **Hive Metastore:** The Hive Metastore is a critical component of the Hive ecosystem. It acts as a central metadata repository that stores and manages metadata information about tables, columns, partitions, and other related objects in Hive. The metastore decouples the metadata from the compute layer, allowing multiple instances of Hive to share the same metadata.
* **The Hive Metastore provides several important functions:**
* **Metadata Storage**: It stores the schema and structural information of tables, such as table names, column names, data types, file locations, partitioning information, and storage properties. This metadata allows Hive to understand the structure of data and optimize query execution.
* **Metadata Retrieval:** When a query is executed, Hive relies on the metastore to retrieve the relevant metadata. This includes information about table locations, column statistics, partition details, and other metadata required for query planning and optimization.
* **Metadata Management:** The metastore provides APIs and interfaces to create, alter, and drop tables, partitions, and other database objects. It ensures consistency and integrity of metadata across different Hive sessions and operations.
* **Integration with External Tools:** The metastore supports integration with external tools and frameworks. For example, it can integrate with Apache Ranger for access control and Apache Atlas for metadata governance and lineage tracking.
* **HBase:** HBase is a distributed, scalable, and non-relational database that provides random, real-time read/write access to large datasets. It is designed to handle massive amounts of structured and semi-structured data with high scalability and fault tolerance. HBase is built on top of the Hadoop ecosystem and tightly integrated with HDFS, making it suitable for big data processing.
* **Here are key features and concepts related to HBase:**
* **Data Model:** HBase follows a columnar NoSQL data model, where data is organized in tables consisting of rows and columns. Each row has a unique row key, and columns are grouped into column families. HBase provides fast random read/write access to individual records based on their row key.
* **Scalability:** HBase is designed to scale horizontally by distributing data across a cluster of machines. It can handle massive datasets and provides automatic sharding and load balancing.
* **Data Consistency:** HBase guarantees strong consistency at the row level, allowing atomic read and write operations within a row. It provides mechanisms like row locks and atomic compare-and-set operations to maintain data consistency.
* **High Availability:** HBase ensures high availability through data replication. It replicates data across multiple nodes in the cluster, providing fault tolerance and eliminating single points of failure.
* **HBase and HDFS:** HBase uses HDFS for storing its data. It leverages HDFS's fault tolerance and data locality features to efficiently manage and access data.
* **Integration:** HBase integrates with other Hadoop ecosystem components, such as Hive, Pig, and MapReduce, allowing seamless data integration and processing.

**Q.7) Illustrate Hive with help of diagram and what is Hive QL.**

* **Hive:**

|  |
| --- |
| Hive Architecture |

|  |  |
| --- | --- |
| **Unit Name** | **Operation** |
| **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. |

* **Hive QL:** HiveQL (Hive Query Language) is a SQL-like language used in Apache Hive for querying and managing data stored in Hadoop. It is specifically designed to work with structured and semi-structured data and provides a familiar SQL-like syntax for data processing tasks. HiveQL includes several types of SQL commands, including Data Definition Language (DDL) commands. DDL commands are used for defining and managing the structure and properties of database objects such as tables, databases, partitions, views, and functions. These commands allow users to create, modify, and delete database objects in Hive.

**Q.8) Differentiate between HBase and RDBMS.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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. |

**Q.9) Describe HBase With examples.**

* HBase is an open-source, distributed, and scalable NoSQL database built on top of the Hadoop Distributed File System (HDFS). It provides real-time read and write access to large amounts of structured and semi-structured data. Here's an overview of HBase with some examples:
* **Column-Oriented Storage:** HBase organizes data in a column-oriented fashion, which means that data within a table is stored by columns rather than rows. Each column can have multiple versions, allowing efficient data retrieval based on timestamps. This storage model is well-suited for scenarios where fast random access to specific columns of data is required.
* **Data Model:** HBase follows a schema-less data model, allowing flexibility in storing and retrieving data. It uses tables to store data, where each table consists of rows and columns. The primary key of a row, called the row key, is used to uniquely identify and access the data. HBase supports nested data structures, making it suitable for handling complex and hierarchical data.
* **Example:** Consider a social media application where users can post messages. In HBase, you can create a table called "posts" with columns like "user\_id", "post\_id", "message", "timestamp", etc. Each row represents a post, and the row key could be a combination of "user\_id" and "post\_id". The column-oriented storage allows efficient retrieval of individual columns, such as retrieving all posts by a specific user or fetching the timestamp of a post.
* **Distributed and Scalable:** HBase is designed to scale horizontally by distributing data across a cluster of machines. It leverages Hadoop's distributed infrastructure and data locality to achieve high scalability and fault-tolerance. HBase automatically partitions and distributes data across regions, allowing for parallel processing and seamless scalability.
* **Example:** Suppose you have a large dataset of customer information that you want to store and analyze in HBase. You can create an HBase table called "customers" and distribute the data across multiple regions. Each region is hosted on a separate node in the cluster, enabling parallel processing of queries and handling large datasets.
* **Strong Consistency and ACID:** HBase provides strong consistency for read and write operations within a row. It supports Atomicity, Consistency, Isolation, and Durability (ACID) properties within a single row. This means that when updating or reading data within a row, the changes are immediately visible to subsequent read operations.
* **Example:** Suppose you have an HBase table for tracking customer orders. Each row represents an order, and columns within the row store order details like order ID, customer ID, product ID, quantity, etc. When multiple users concurrently update their orders, HBase ensures that the updates are applied consistently within each order, maintaining data integrity.
* **Integration with Hadoop Ecosystem:** HBase seamlessly integrates with other components of the Hadoop ecosystem. It can be used alongside tools like Apache Spark, Hive, and MapReduce to perform advanced analytics, batch processing, and complex queries on HBase data. HBase can also interact with Apache Kafka for real-time data streaming.
* **Example:** You can use HBase as a data source for Apache Spark to perform distributed data processing and analytics on large datasets. Spark can leverage HBase's distributed storage and processing capabilities to run complex computations and gain insights from the data stored in HBase.

**Q.10) What are the Hive Services & Explain in short BigSQL.**

* **Hive Services:**

1. **Hive Metastore:** The Hive Metastore is a central component of Apache Hive that stores and manages metadata information about Hive tables, partitions, columns, databases, and other related objects. It acts as a catalog or registry for Hive, allowing users to query and manipulate the metadata associated with the data stored in Hive. The metastore enables Hive to provide a structured view of the data, enforce data integrity, and optimize query execution.
2. **Hive Server:** The Hive Server provides a Thrift and JDBC interface to interact with Hive. It acts as a gateway for clients to submit queries and commands to Hive and retrieve results. The Hive Server facilitates client-server communication, query processing, and resource management. It supports multiple clients concurrently, allowing users to connect to Hive from various applications and tools.
3. **Hive Driver:** The Hive Driver is responsible for executing HiveQL queries and managing the interaction between the Hive Server and the Hive execution engine. It parses and validates queries, optimizes query execution plans, and coordinates the execution of tasks across the cluster.

* **BigSQL:** BigSQL is an open-source SQL-on-Hadoop solution developed by IBM. It is built on top of the Apache Hive and Apache Hadoop ecosystem, providing enhanced SQL capabilities for data processing and analytics. BigSQL aims to simplify data access and analysis by providing a familiar SQL interface while leveraging the power and scalability of Hadoop.
* **Features of BigSQL include:**

1. **ANSI SQL Compliance:** BigSQL supports ANSI SQL standards, making it easier for users familiar with SQL to interact with big data stored in Hadoop. It allows users to write SQL queries that can be executed against large-scale datasets.
2. **Advanced Query Optimization:** BigSQL includes optimizations for query execution, such as predicate pushdown, join optimizations, and data pruning. These optimizations improve query performance and reduce the amount of data processed, resulting in faster query response times.
3. **Data Virtualization:** BigSQL enables virtualization of data stored in various formats and locations within the Hadoop ecosystem. It provides a unified view of data, allowing users to query and analyze data from multiple sources, including Hive tables, HBase, Avro files, and more, using a single SQL interface.
4. **Data Federation:** BigSQL allows users to query data across multiple Hadoop clusters, including both Hive and HBase tables, as well as external data sources. It provides a mechanism for accessing and integrating data distributed across different clusters, enabling cross-cluster analytics and reporting.
5. **Security and Administration:** BigSQL offers security features such as authentication, authorization, and auditing. It integrates with Hadoop's security framework, allowing users to secure their data and control access to various resources. BigSQL also provides tools for monitoring and managing the SQL environment.

**(PPT Give you more understanding than PDF)** The material for the PDF has been compiled from various sources such as books, tutorials (offline and online), lecture notes, several resources available on Internet. The information contained in this PDF is for general information and education purpose only. While we endeavor to keep the information up to date and correct, **we make no representation of any kind about the completeness and accuracy of the material.** The information shared through this PDF material should be used for educational purpose only.

**Thank you…**