Athens University of Economics and Business

School of Business

Department of Management Science & Technology

Master of Science in Business Analytics

Program:	Full-time
Quarter:	3rd (Spring Quarter)
Course:	Advanced Topics in Data Engineering
Instructor:	Dr. Giorgos Alexiou
Assignment №:	Drill-Impala Assignment
Students (Registration №):	Souflas Eleftherios - Efthymios (f2822217)

Table of Contents

Drill – Impala Assignment

Introduction

In the world of big data, organizations often deal with vast amounts of data stored in different formats and disparate data sources. To make sense of this data and derive valuable insights, data virtualization becomes essential. In the following tasks, Apache Drill, Hive, and Impala will be compared in terms of data virtualization (Task 1). Additionally, a practical application of Apache Drill will be explored (Task 2), an Impala database will be created (Task 3), and steps to enhance Impala's query performance will be outlined (Task 4).

Task 1

Among Hive, Impala, and Drill, the one that implements data virtualization more precisely is **Apache Drill**.

Data virtualization refers to the ability to query and analyze data from various sources as if it were all stored in a single location. It is a solution that sits on top of multiple, heterogeneous data sources and allows them to be treated as a single database. With a data virtualization tool, data can be retrieved and manipulated without requiring knowledge about the format & physical location of the data (Alexiou, Advanced Topics in Data Engineering: Apache Drill, 2023, pp. 9-11).

Hive and Impala are both SQL-on-Hadoop engines, which means they allow users to run SQL-like queries on data stored in Hadoop. However, they are not limited to querying data stored exclusively in Hadoop:

- Hive is a data warehouse software used to access and manage large, distributed datasets stored in Hadoop Distributed File System (HDFS). When tables are created in Hive, the default location is HDFS. However, Hive can also be configured to work with other storage systems using custom storage handlers. For example, Hive can be used with HBase tables, cloud-based storage systems like Amazon S3, or other external data sources using the Hive External Tables feature.
- Impala was initially designed to work with Hadoop's HDFS as well. However, Impala has evolved to support other storage systems beyond Hadoop. It can also query data from HBase, Kudu (a fast columnar storage engine), and even data stored in Hadoop-compatible cloud storage services like Amazon S3 or Azure Data Lake Storage (ADLS). This allows Impala to provide better performance for certain workloads, especially when low-latency query responses are required.

So, both Hive and Impala can query and analyze data from various sources as if it were all stored in a single location. However, both Hive and Impala follow a relational model (schema-based) and need a table/view to be declared before querying (Alexiou, Advanced Topics in Data Engineering: Impala, 2023, pp. 4-10). So, data cannot be retrieved and manipulated without a-priori knowledge of its format & physical location.

Drill is a low-latency distributed query engine designed to enable data exploration and analytics on both relational and non-relational datastores, scaling to petabytes of data. It can access data stored in a variety of locations, including Hadoop, Hive, HBase, and other NoSQL databases, which makes it easy to query data from different sources without having to move it to a single location. Apache Drill excels also in the aspect of schema-on-read feature and can directly query various data formats, like JSON, Parquet, Avro, CSV, and more, without requiring a predefined schema or data preparation. This flexibility allows users to explore and analyze diverse data sources without needing to load them into a centralized data store (Gaur, 2021). Thus, Apache Drill implements the data virtualization concept better than Hive and Impala, as it can both sit on top of multiple, heterogeneous data sources, allowing them to be treated as a single database, and retrieve and manipulate data without requiring knowledge about its format & physical location.

In summary, Apache Drill is designed from the ground up with data virtualization in mind and excels in this aspect, providing schema-agnostic querying and data analysis implementation from various sources with different formats.

Task 2

For the large bookstore company with data in various formats, stored in different databases, I would suggest using a data virtualization tool like **Apache Drill**, as it would be an excellent solution to simplify their queries and enhance their data analytics capabilities. Below it is described how Apache Drill can address the company's requirements:

1. Single SQL Interface: Apache Drill provides a standard SQL interface that is familiar to most developers and data analysts. This means that users can use the same SQL queries they are accustomed to, regardless of the underlying data source. It eliminates the need for users to learn different query languages for MongoDB, HDFS, and Hive, which saves time and reduces complexity.

2. Schema-on-Read: One of the key features of Apache Drill is its ability to perform schema-on-read. Unlike traditional relational databases that require a predefined schema before data is loaded, Drill can automatically discover and interpret the schema of the data as it is being queried. This flexibility is essential in cases where data formats might change frequently or where new data sources are added over time.

3. Performance: Apache Drill is designed for high-performance query execution. It leverages distributed computing capabilities to parallelize queries across the cluster, ensuring fast response times even when dealing with large datasets. This advantage allows the company's user interface elements to receive query results quickly, improving the overall user experience.

4. No ETL Required: By using Apache Drill for data virtualization, the company can avoid the need for complex and time-consuming Extract, Transform, Load (ETL) processes. ETL typically involves moving and transforming data from various sources into a central data warehouse, which can be resource-intensive and introduce data latency. With Drill, data remains in its original location, and querying happens on-the-fly, eliminating the need for data movement and maintaining data freshness.

5. Support for Various Data Formats: Apache Drill supports querying a wide range of data formats, including JSON, Parquet, Avro, CSV, and more. This is especially beneficial for the large bookstore company, as they have data stored in MongoDB, HDFS, and Hive, each with its respective format. Drill's ability to handle different formats simplifies data access and analysis.

6. Scalability: As the company's data grows, Apache Drill can scale horizontally by adding more nodes to the cluster. This ensures that the system can handle increased query loads and maintain performance levels as data volume and user demands increase.

In conclusion, adopting Apache Drill as a data virtualization tool would provide the large bookstore company with a unified and simplified way to query and analyze data from MongoDB, HDFS, and Hive, using standard SQL. It would eliminate the need for ETL processes, improve query performance, and provide a seamless experience for their user interface elements, ultimately enabling more efficient and effective data-driven decision-making.

Task 3

a) To create the Impala database and the required tables that our client needs, the following Impala commands should be executed:

```
CREATE DATABASE IF NOT EXISTS University;
USE University;
CREATE TABLE IF NOT EXISTS Student (sid INT, name VARCHAR(50));
CREATE TABLE IF NOT EXISTS Course (cid INT, title VARCHAR(50),
                                   description STRING);
CREATE TABLE IF NOT EXISTS Attended (sid INT, cid INT,
                                   ac_year CHAR(9), grade DECIMAL(1,0));
```

Figure 1 - Impala Database & Tables creation

Because the `NUMERIC` and `TEXT` data types do not exist in Impala, the `NUMERIC(1)` and `TEXT` were replaced with the equivalent `DECIMAL(1,0)` and `STRING` data types of Impala database.

b) An example command for inserting an entry to the `Student` table can be the following:

```
INSERT INTO University.Student (sid, name)
VALUES (1, CAST('Lefteris Souflas' AS VARCHAR(50)));
```

Figure 2 - `Student` table entry insertion

Because the name value, i.e. `Lefteris Souflas`, is considered as a value in STRING data type if it is going to be inserted as-is, the `CAST` function must be used to convert it to `VARCHAR(50)`, which is our client's column requirement. If the `CAST` function is not used, then an $AnalysisException$ error will be raised, warning about possible loss of precision for target table `University.Student` suggesting the forementioned value to be cast to the respective column's data type.

c) The statement to retrieve all names of students attending the course "Artificial Intelligence" during the academic year "2021-2022" is the following:

```
SELECT s.name
FROM student s, attended a, course c
WHERE s.sid=a.sid AND a.cid=c.cid AND
    c.title='Artificial Intelligence' AND
    a.ac_year='2021-2022';
```

Figure 3 - The required statement with the JOIN of all schema tables

d) The statement to retrieve the title and average grade of all courses with average grade of students that attended them lower than 6 is the following:

```sql
SELECT c.title, avg(a.grade) AS 'Average Grade'
FROM attended a, course c
WHERE a.cid=c.cid
GROUP BY c.title
HAVING avg(a.grade) < 6;
```

Figure 4 - The required statement with GROUP BY and HAVING clauses

Task 4

If a particular query in the previous Impala database is too slow, the following steps can be taken to investigate and improve its performance:

1. Obtain the EXPLAIN Plan: First, the EXPLAIN statement for the slow query can be run without running the actual query. This report provides an outline of the logical steps that the query will perform and how the work will be distributed among the nodes when it will be executed. It also presents how intermediate results will be combined to produce the final result set (The Apache Software Foundation, n.d.). Overall, this step helps identify potential inefficiencies in the query execution plan.

2. Analyze SUMMARY Report: The SUMMARY command in impala-shell can then be used immediately after executing the query to get a high-level overview of the physical performance characteristics for the query. This report presents which stages took the most time and how the estimates for memory usage and number of rows at each phase compare to actual values (The Apache Software Foundation, n.d.).

3. Use the PROFILE Statement: After running the slow query, the PROFILE statement in impala-shell can be used to obtain detailed low-level performance characteristics about its execution. This includes physical details about memory, CPU, I/O, and network usage (The Apache Software Foundation, n.d.). The profile output will help identify potential bottlenecks and resource usage.

4. Optimize Joins and Predicates: The query should be reviewed to ensure that joins are performed efficiently and that appropriate predicates are used to limit the amount of data processed. A check if any unintended cross-joins exist in the query should be made, as cross-joins can lead to huge result sets and slow down the query significantly.

5. Check for Missing or Outdated Statistics: Impala can do better optimization for complex or multi-table queries when it has access to statistics about the volume of data and how the values are distributed. Impala uses this information to help parallelize and distribute the work for a query. Thus, it must be verified that the involved tables and columns have accurate statistics. Statistics can be updated using the statements `COMPUTE STATS` or the `COMPUTE INCREMENTAL STATS` for partitioned tables (The Apache Software Foundation, n.d.).

6. Check for Skewed Data: Data distribution issues may also cause performance problems. If certain data partitions are significantly larger or smaller than others, it can lead to skewed workloads across nodes. Optimizing partitioning or using techniques, like bucketing, to evenly distribute data should also be considered. The `SHOW TABLE STATS` command can be used to view statistics, including the number of rows and data size, for each partition of the tables that slow down the query execution (The Apache Software Foundation, n.d.). This can help identify partitions with significantly different data sizes.

7. Consider Caching: Impala can use the HDFS caching feature to make more effective use of RAM, so that repeated queries can take advantage of data "pinned" in memory, regardless of how much data is

processed overall. The HDFS caching feature designates a subset of frequently accessed data to be pinned permanently in memory, remaining in the cache across multiple queries and never being evicted. This technique is suitable for tables or partitions that are frequently accessed and are small enough to fit entirely within the HDFS memory cache. For example, several dimension tables can be designated to be pinned in the cache to speed up many different join queries that reference them, or a partition in a partitioned table holding data from the most recent time period can be pinned, because that data will be queried intensively (The Apache Software Foundation, n.d.). In the example below (Figure 5), `hdfs cacheadmin` command is issued to set up a 4-gigabyte cache pool named `four_gig_pool`, owned by `impala` user (same user as the *impalad* daemon) and the `Attended` table of the `University` database is then cached in the forementioned cache pool.

```
terminal>sudo su hdfs -c "hdfs cacheadmin -addPool four_gig_pool
-owner impala -limit 4000000000"

sql-editor>alter table university.attended set cached in
'four_gig_pool';
```

Figure 5 - Caching Pool creation and addition of a table to it

8. <u>Tune Configuration Settings</u>: Limiting the number of resources, such as memory or CPU, used by a single query or group of queries can also be useful in optimizing overall performance. Impala can use several mechanisms that help to smooth out the load during heavy concurrent usage, resulting in faster overall query times and sharing of resources across Impala queries, MapReduce jobs, and other kinds of workloads across a cluster. Using the Impala Admission Control feature, the concurrency limits can be controlled and different limits for different groups of users can be specified to divide cluster resources according to the priorities of different classes of users. The amount of memory Impala reserves during query execution can be restricted by specifying the `MEM_LIMIT` option for the *impalad* daemon (The Apache Software Foundation, n.d.). The MEM_LIMIT query option defines the maximum amount of memory a query can allocate on each node. The total memory that can be used by a query is the MEM_LIMIT times the number of nodes (The Apache Software Foundation, n.d.).

9. <u>Evaluate Hardware</u>: Finally, upgrading hardware for Impala nodes can be considered, if current hardware is not sufficient to handle the query workload.

References

Alexiou, G. (2023, June 12). Advanced Topics in Data Engineering: Apache Drill. Athens, Attica, Greece: AUEB's MSc Business Analytics.

Alexiou, G. (2023, June 13). Advanced Topics in Data Engineering: Impala. Athens, Attica, Greece: AUEB's MSc Business Analytics.

Gaur, C. (2021, January 26). *Apache Drill and it's Architecture to Connect Data Sources*. Retrieved July 25, 2023, from Xenonstack: https://www.xenonstack.com/blog/apache-drill-architecture

The Apache Software Foundation. (n.d.). *Controlling Impala Resource Usage*. Retrieved July 29, 2023, from Apache Impala: https://impala.apache.org/docs/build/html/topics/impala_perf_resources.html#mem_limits

The Apache Software Foundation. (n.d.). *Detecting and Correcting HDFS Block Skew Conditions*. Retrieved July 29, 2023, from Apache Impala: https://impala.apache.org/docs/build/html/topics/impala_perf_skew.html

The Apache Software Foundation. (n.d.). *MEM_LIMIT Query Option*. Retrieved July 29, 2023, from Apache Impala: https://impala.apache.org/docs/build/html/topics/impala_mem_limit.html

The Apache Software Foundation. (n.d.). *Table and Column Statistics*. Retrieved July 28, 2023, from Apache Impala: https://impala.apache.org/docs/build/html/topics/impala_perf_stats.html

The Apache Software Foundation. (n.d.). *Understanding Impala Query Performance - EXPLAIN Plans and Query Profiles*. Retrieved July 28, 2023, from Apache Impala: https://impala.apache.org/docs/build/html/topics/impala_explain_plan.html

The Apache Software Foundation. (n.d.). *Using HDFS Caching with Impala (Impala 2.1 or higher only)*. Retrieved July 29, 2023, from Apache Impala: https://impala.apache.org/docs/build/html/topics/impala_perf_hdfs_caching.html