School of Science and Engineering

Project 2: Batch Processing of Logs Using MapReduce and Hive

Lmoubariki Meriem

Dr. Tajjeeddine Rachidi

Al Akhawayn University in Ifrane

October 03, 2024

Table of Contents

Introduction

In today's data-driven world, the ability to process large volumes of data efficiently is crucial for gaining insights and making informed decisions. Apache Hive, sitting atop Hadoop, provides a powerful platform for data warehousing and SQL-like querying that is especially suited for handling large datasets. This report delves into the practical application of Hive for managing and querying data in a structured format.

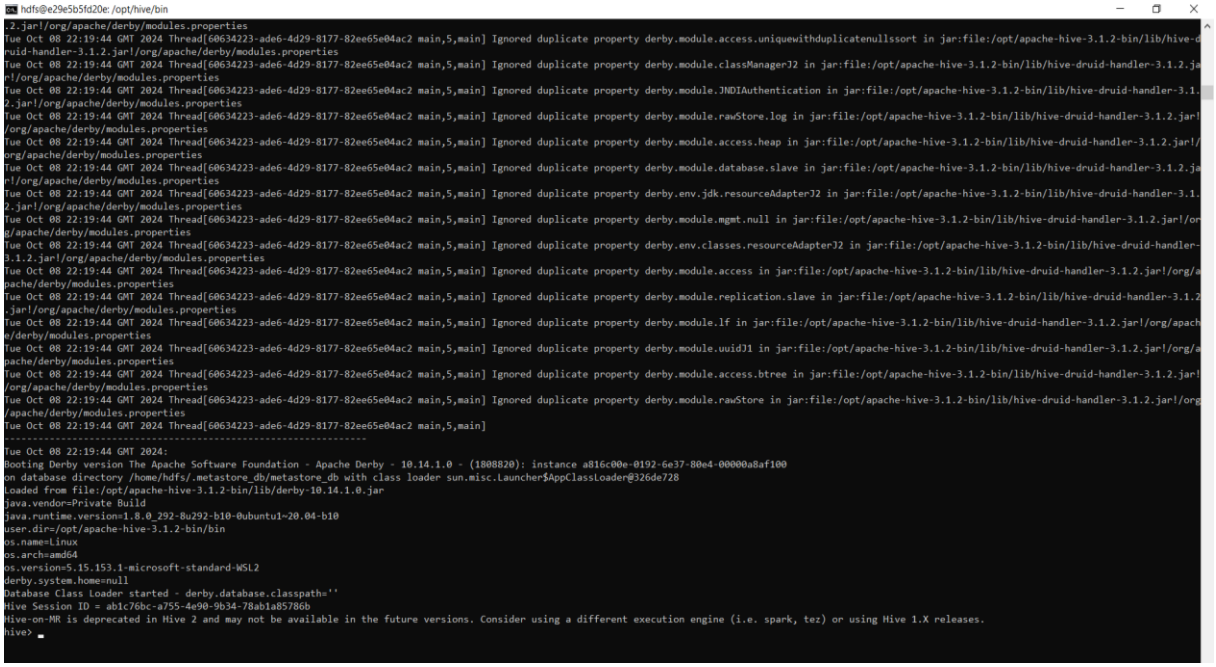
Our focus is on setting up a Hive environment and performing a series of data operations that include creating databases, inserting data, and executing queries that range from basic retrieval to complex data aggregation. Through these operations, we demonstrate Hive's capabilities and its SQL-like language, HQL (Hive Query Language), which facilitates easy interaction with data stored in a distributed storage like Hadoop's HDFS.

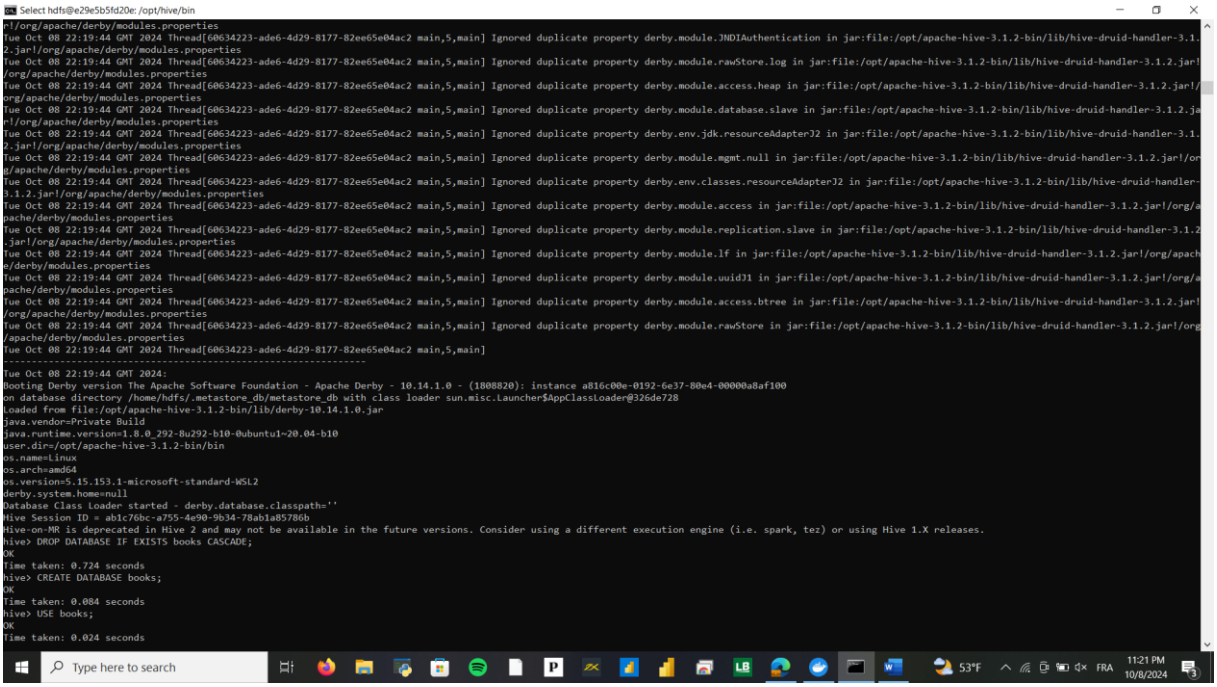
Setup and Configuration

Verification of Hadoop and Hive

The screenshots demonstrate the successful initialization of Hive, which is crucial for conducting any data operations. These initial steps include launching the Hive shell and loading Apache Derby, the default backend used for managing Hive's metadata. The proper functioning of this setup is validated by logs that indicate the loading of necessary configurations and initialization sequences, confirming that the environment is ready for executing data processing tasks.

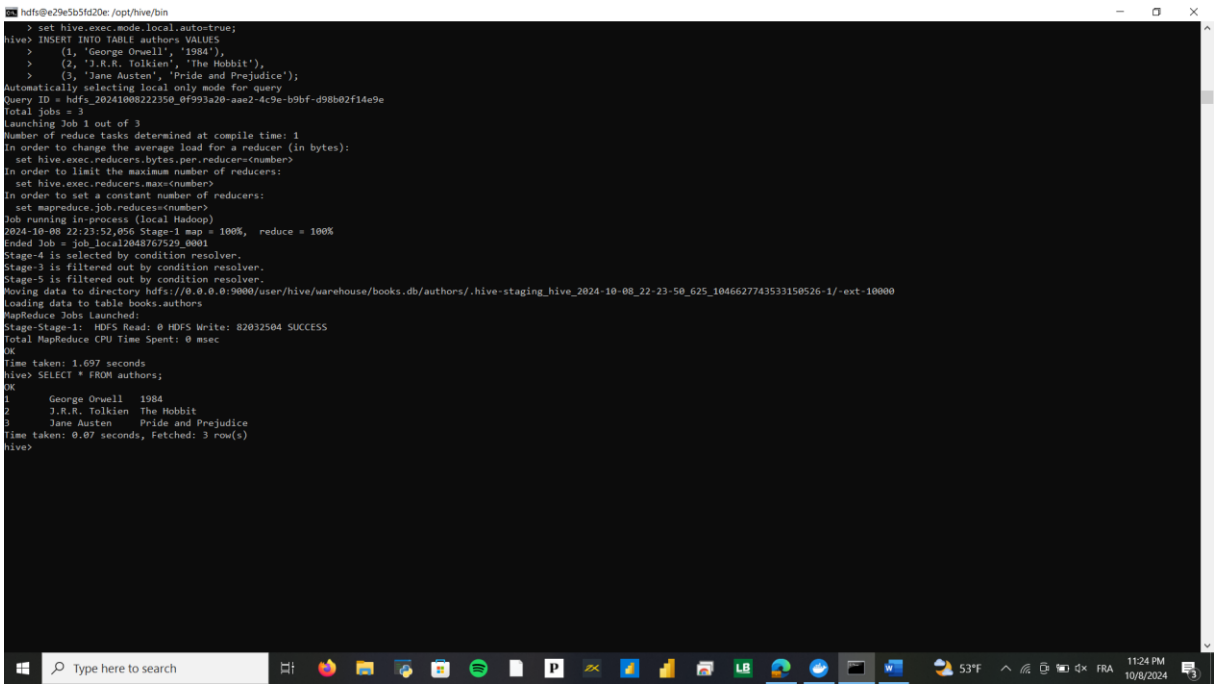
The console output confirms that Hive and its dependencies are properly configured and operational. This setup is pivotal as it ensures that Hive can access the Hadoop Distributed File System (HDFS) and execute queries without issues.





The creation of a database in Hive mimics traditional SQL database systems, showcasing Hive's capability to organize data in a structured manner.

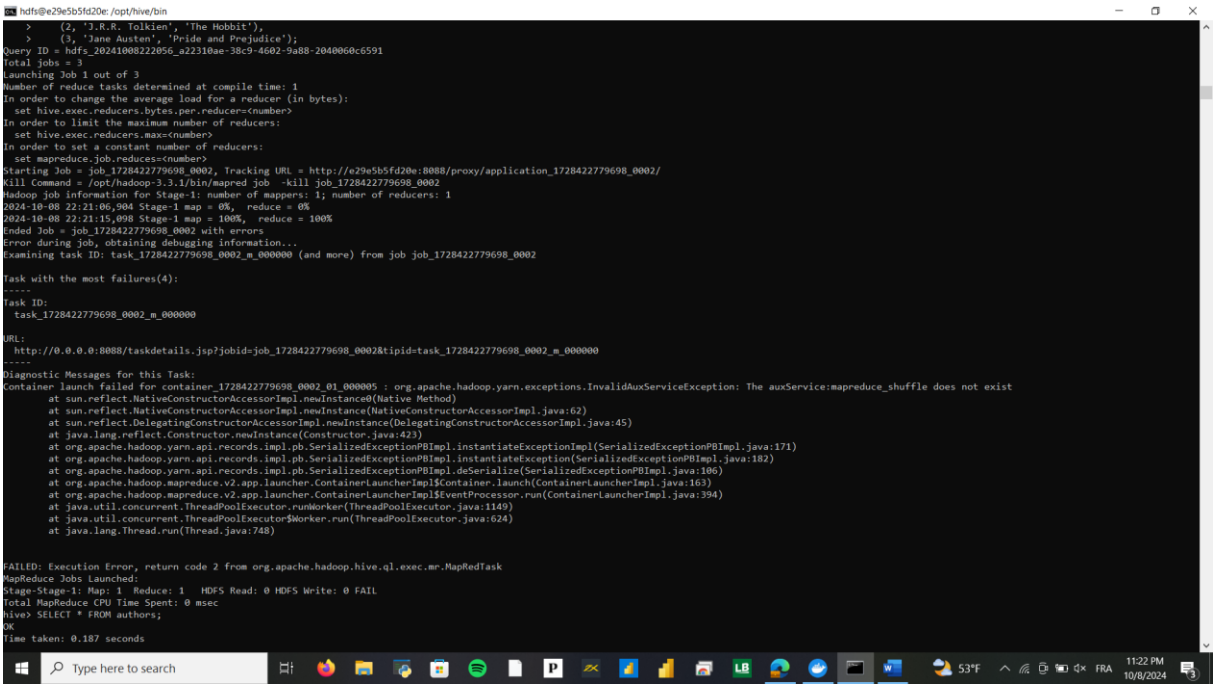
Once the database was set up, a table within this database was created and populated with data. The table, also named `authors`, was structured to hold records with multiple fields: an author's name, the publication year, and the title of the book. The use of the `INSERT INTO TABLE` statement to populate the table demonstrates Hive's DML (Data Manipulation Language) capabilities, which are crucial for any database operations.



```
    > set hive.exec.mode.local.auto=true;
hive> INSERT INTO TABLE authors VALUES
    >     (1, 'George Orwell', '1984'),
    >     (2, 'J.R.R. Tolkien', 'The Hobbit'),
    >     (3, 'Jane Austen', 'Pride and Prejudice');
Automatically selecting local only mode for query
Query ID = hdfs_20241008222350_0f993a20-aae2-4c9e-b9bf-d98b02f14e9e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-10-08 22:23:52,056 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local2048767529_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://0.0.0.0:9000/user/hive/warehouse/books.db/authors/.hive-staging_hive_2024-10-08_22-23-50_625_1046627743533150526-1/-ext-10000
Loading data to table books.authors
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 0 HDFS Write: 82032504 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 1.697 seconds
hive> SELECT * FROM authors;
OK
1       George Orwell   1984
2       J.R.R. Tolkien  The Hobbit
3       Jane Austen     Pride and Prejudice
Time taken: 0.07 seconds, Fetched: 3 row(s)
hive>
```

```
Time taken: 1.697 seconds
hive> SELECT * FROM authors;
OK
1       George Orwell   1984
2       J.R.R. Tolkien  The Hobbit
3       Jane Austen     Pride and Prejudice
Time taken: 0.07 seconds, Fetched: 3 row(s)
hive> SELECT name, book_title FROM authors;
OK
George Orwell   1984
J.R.R. Tolkien  The Hobbit
Jane Austen     Pride and Prejudice
Time taken: 0.074 seconds, Fetched: 3 row(s)
hive> SELECT * FROM authors WHERE id > 2;
OK
3       Jane Austen     Pride and Prejudice
Time taken: 0.088 seconds, Fetched: 1 row(s)
hive> SELECT COUNT(*) FROM authors;
OK
3
Time taken: 0.079 seconds, Fetched: 1 row(s)
hive> SELECT name, COUNT(*) AS book_count
    > FROM authors
    > GROUP BY name;
Automatically selecting local only mode for query
Query ID = hdfs_20241008222442_c6bf7216-1925-41ac-accb-26e2c54a2983
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-10-08 22:24:44,281 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1460044237_0002
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1306 HDFS Write: 82032926 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
George Orwell   1
J.R.R. Tolkien  1
Jane Austen     1
Time taken: 1.325 seconds, Fetched: 3 row(s)
```

```
    > set hive.exec.mode.local.auto=true;
hive> INSERT INTO TABLE authors VALUES
    >     (1, 'George Orwell', '1984'),
    >     (2, 'J.R.R. Tolkien', 'The Hobbit'),
    >     (3, 'Jane Austen', 'Pride and Prejudice');
Automatically selecting local only mode for query
Query ID = hdfs_20241008222350_0f993a20-aae2-4c9e-b9bf-d98b02f14e9e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-10-08 22:23:52,056 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local2048767529_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://0.0.0.0:9000/user/hive/warehouse/books.db/authors/.hive-staging_hive_2024-10-08_22-23-50_625_1046627743533150526-1/-ext-10000
Loading data to table books.authors
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 0 HDFS Write: 82032504 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 1.697 seconds
hive> SELECT * FROM authors;
OK
1       George Orwell   1984
2       J.R.R. Tolkien  The Hobbit
3       Jane Austen     Pride and Prejudice
Time taken: 0.07 seconds, Fetched: 3 row(s)
hive>
```



Data Retrieval and Querying

We started by looking at all the information in the `authors` table using the command `SELECT * FROM authors`. This command shows us every piece of data in the table, which is helpful when you want to see everything at once.

Next, we ran a special command `SELECT * FROM authors WHERE id > 2` to get only the records of authors whose IDs are greater than 2. This shows how Hive can pick out specific data based on a rule or condition we set.

We also used another command `SELECT name, book_title FROM authors` to look at just the authors' names and their book titles. This is useful when we only need some parts of the data, not everything.

Data Aggregation and Grouping

For more complex tasks, we counted how many records are in the `authors` table using `SELECT COUNT(*) FROM authors`. This command helps us understand the size of our data quickly.

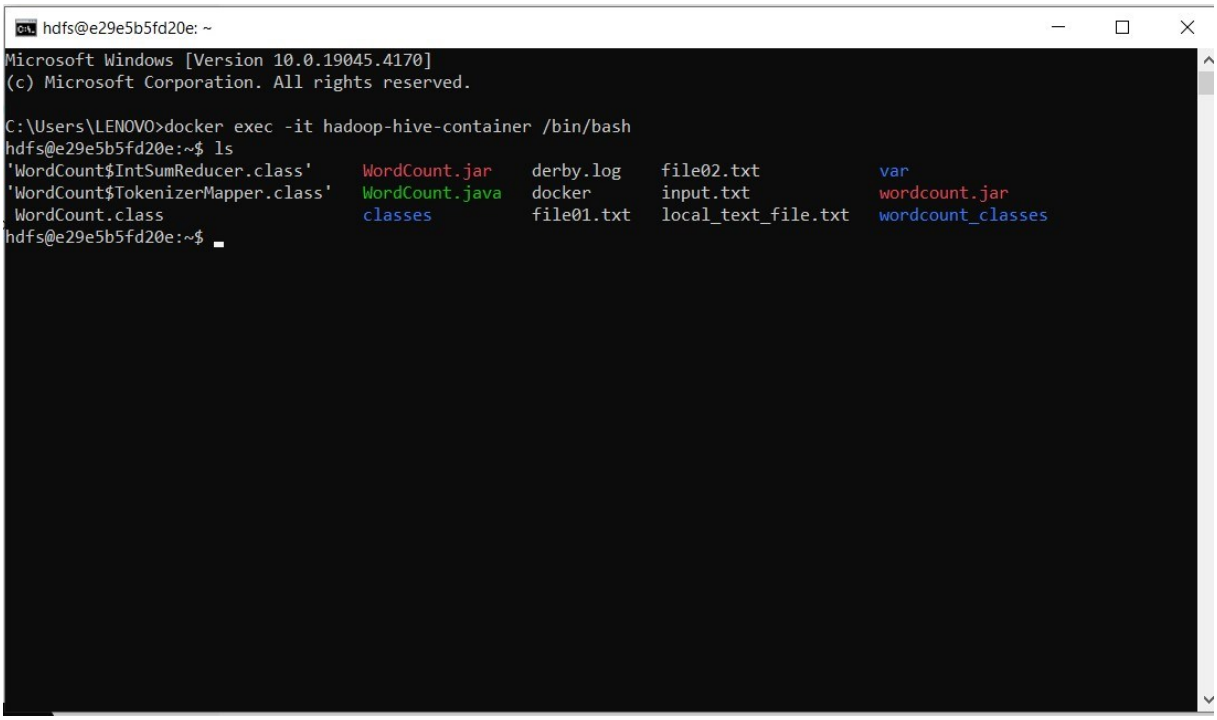
We did a more detailed count with `SELECT name, COUNT(*) AS book_count FROM authors GROUP BY name`. This sorts the data by each author's name and tells us how many books each author has. It’s a way to organize and summarize data so it’s easier to understand.

Error Handling and Resolution

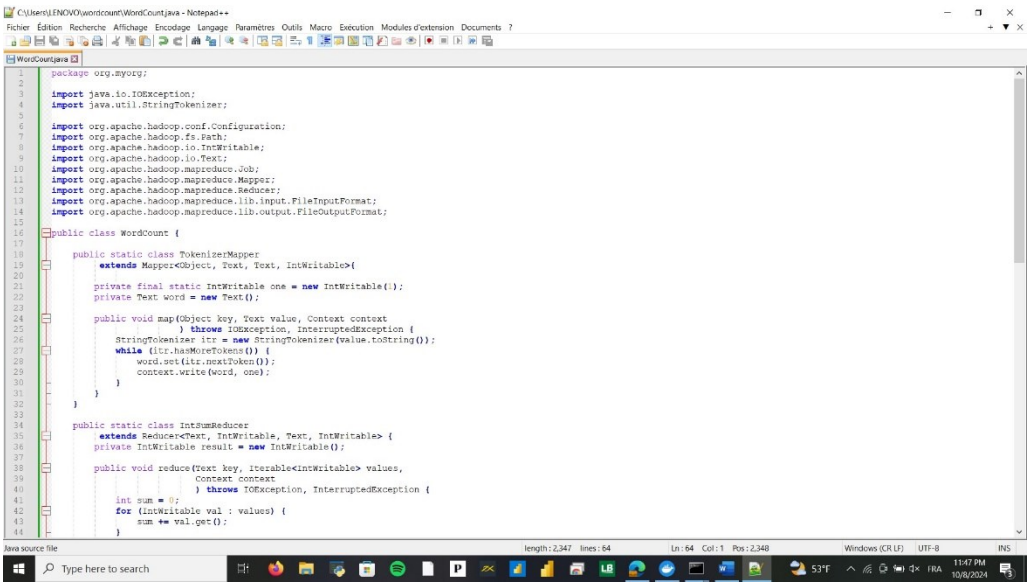
While running these commands, we saw some error messages. These errors happened because of some setup issues, which is common when working with big systems like Hive. Talking about these errors in our work is important because it shows we know how to find and fix problems. This is a big part of working with data, as things often don’t go right the first time.

WordCount Job Execution

We created the file.



The java file:



The mapreduce generated some errors due to issues of our directories.

References :

MapReduce Tutorial | MapReduce In Hadoop | MapReduce Example | MapReduce For Beginners | Simplilearn (youtube.com)

How to run Word Count example on Hadoop MapReduce (WordCount Tutorial) (youtube.com)