Course code: MSDA 9123

Course: Big Data Analytics

Distributed Multi-Model Analytics for E-Commerce Data

FINAL PROJECT

Name: RUBIMBURA Theodore (101023)

Abstract

Modern e-commerce platforms generate large volumes of heterogeneous data, including customer profiles, transaction records, browsing sessions, and product metadata. Managing and analyzing such data efficiently requires scalable and flexible data storage and processing technologies. This project implements a distributed multi-model analytics system using MongoDB, HBase, and Apache Spark to analyze a synthetic e-commerce dataset. MongoDB is used to store document-oriented data such as users, products, and transactions, while HBase is employed for high-volume time-series session data. Apache Spark serves as the distributed analytics engine for data cleaning, aggregation, integration, and advanced analytics. The project demonstrates data modeling, distributed processing, cross-system integration, and visualization of business insights, including revenue distribution and customer lifetime value estimation.

1. Introduction

E-commerce systems produce massive and diverse datasets resulting from customer interactions, purchases, and online behavior. These datasets are often semi-structured or unstructured and grow rapidly in volume and velocity. Traditional relational database systems face limitations when handling such data due to rigid schemas and limited horizontal scalability.

To address these challenges, this project adopts a **multi-model big data architecture** that combines different storage and processing technologies, each chosen for its specific strengths. MongoDB provides flexibility for document-based data, HBase supports scalable storage of time-series data, and Apache Spark enables distributed data processing and analytics.

Objectives

The main objectives of this project are:

- To design appropriate data models for MongoDB and HBase based on data characteristics and query requirements.
- To perform distributed data processing using Apache Spark.
- To integrate data from multiple systems to perform advanced analytics.
- To visualize analytical results and derive meaningful business insights.

2. System Architecture Overview

The system architecture follows a layered approach consisting of three main components:

1. **Data Storage Layer**
    - MongoDB stores user profiles, product catalog data, and transaction records.
    - HBase stores user session activity as time-series data.

```
hbase(main):006:0>
hbase(main):007:0* put 'users', 'user_000173', 'profile:city', 'East Angela'
Took 0.0054 seconds
hbase(main):008:0> put 'users', 'user_000173', 'profile:state', 'MN'
Took 0.0100 seconds
hbase(main):009:0> put 'users', 'user_000173', 'profile:country', 'US'
Took 0.0084 seconds
hbase(main):010:0> put 'users', 'user_000173', 'profile:registration_date', '2024-11-30T14:27:56'
Took 0.0052 seconds
hbase(main):011:0> put 'users', 'user_000173', 'profile:last_active', '2025-04-02T09:15:32'
Took 0.0125 seconds
```

2. **Data Processing Layer**
    - Apache Spark is used to load, clean, transform, and integrate data from MongoDB and HBase.

```
hbase(main):014:0* get 'users', 'user_000173'
COLUMN                              CELL
 profile:city                       timestamp=1769083236214, value=East Angela
 profile:country                    timestamp=1769083236301, value=US
 profile:last_active                timestamp=1769083236415, value=2025-04-02T09:15:32
 profile:registration_date          timestamp=1769083236367, value=2024-11-30T14:27:56
 profile:state                      timestamp=1769083236254, value=MN
1 row(s)
Took 0.0810 seconds
hbase(main):015:0> get 'users', 'user_000042'
COLUMN                              CELL
 profile:city                       timestamp=1769083235938, value=North Michaelville
 profile:country                    timestamp=1769083236019, value=US
 profile:last_active                timestamp=1769083236160, value=2025-03-12T16:23:47
 profile:registration_date          timestamp=1769083236096, value=2024-12-15T08:42:13
 profile:state                      timestamp=1769083235968, value=WY
1 row(s)
Took 0.0364 seconds
hbase(main):016:0>
```

3. **Presentation Layer**
    - Python visualization libraries (Matplotlib) are used to present analytical findings.

PART 1: Data Modeling and Storage

PART 2: Data Processing with Apache Spark

5. Spark Data Processing Pipeline

Apache Spark is used as the primary analytics engine due to its distributed processing capabilities.

5.1 Spark Session Initialization

Figure 1: Spark Session Initialization

Start Spark Session

```
[1]: from pyspark.sql import SparkSession

     # Create SparkSession in local mode
     # local[*] allows Spark to use all available CPU cores
     spark = SparkSession.builder \
         .appName("Big_Data_Analytics_Final_Project'") \
         .master("local[*]") \
         .getOrCreate()

     # Confirm Spark version
     spark
```

[1]: **SparkSession - in-memory**

SparkContext

Spark UI

Version	v3.5.7
Master	local[*]
AppName	Big_Data_Analytics_Final_Project'

A Spark session was created to manage distributed data processing tasks.

5.2 Data Loading and Cleaning

JSON datasets were loaded into Spark DataFrames using multi-line parsing to correctly handle nested structures. Invalid and incomplete records were removed during the cleaning phase.

Figure 5: Cleaned Transactions Dataset Schema

```
[9]: transactions_df = spark.read \
         .option("multiLine", "true") \
         .json("data/transactions.json")

     transactions_df.printSchema()
     transactions_df.show(5, truncate=False)

root
 |-- discount: double (nullable = true)
 |-- items: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- product_id: string (nullable = true)
 |    |    |-- quantity: long (nullable = true)
 |    |    |-- subtotal: double (nullable = true)
 |    |    |-- unit_price: double (nullable = true)
 |-- payment_method: string (nullable = true)
 |-- session_id: string (nullable = true)
 |-- status: string (nullable = true)
 |-- subtotal: double (nullable = true)
 |-- timestamp: string (nullable = true)
 |-- total: double (nullable = true)
 |-- transaction_id: string (nullable = true)
 |-- user_id: string (nullable = true)

+--------+-------------------------------+--------------+----------+---------+--------+--------------------------+------+----------------+
-----------+
|discount|items                          |payment_method|session_id|status   |subtotal|timestamp                 |total |transaction_id  |
user_id    |
+--------+-------------------------------+--------------+----------+---------+--------+--------------------------+------+----------------+
-----------+
|5.69    |[{prod_00381, 1, 113.88, 113.88}]|paypal      |NULL      |delivered|113.88  |2026-01-31T16:37:29.668873|108.19|txn_79bba9f40fa8|
user_000448|
+--------+-------------------------------+--------------+----------+---------+--------+--------------------------+------+----------------+
-----------+
```

5.3 Revenue Analytics

Revenue was computed by joining transaction items with product data and aggregating revenue by product category.

Figure 7: Revenue by Product Category Output

```
[24]: revenue_by_category_df = transaction_items_df \
          .join(
              products_df,
              transaction_items_df.product_id == products_df.product_id,
              "inner"
          ) \
          .groupBy("category_id") \
          .agg(
              _sum("subtotal").alias("total_revenue")
          ) \
          .orderBy(col("total_revenue").desc())
      revenue_by_category_df.show(truncate=True)

      +-----------+-------------+
      |category_id|total_revenue|
      +-----------+-------------+
      |    cat_010|       113.88|
      +-----------+-------------+
```

This analysis demonstrates Spark's ability to process and aggregate large datasets efficiently.

PART 3: Analytics Integration

6. Integrated Analytics: Customer Lifetime Value (CLV)

6.1 Business Question

Which customers generate the highest long-term value for the e-commerce platform based on their purchase history and engagement behavior?

6.2 Data Sources Integrated

- MongoDB: transaction and user data
- HBase: session engagement data
- Spark: data integration and analytics

6.3 CLV Computation

Spark was used to aggregate transaction totals per user and combine them with session engagement metrics to compute a composite CLV score.

Figure 8: Customer Lifetime Value Computation Output

```
[26]: from pyspark.sql.functions import avg

# Aggregate session engagement per user
session_metrics_df = sessions_df \
    .groupBy("user_id") \
    .agg(
        count("session_id").alias("session_count"),
        avg("duration_seconds").alias("avg_session_duration")
    )

session_metrics_df.show(truncate=False)
```

```
+-----------+-------------+--------------------+
|user_id    |session_count|avg_session_duration|
+-----------+-------------+--------------------+
|user_000066|7            |2166.8571428571427  |
|user_000113|7            |2468.1428571428573  |
|user_000098|12           |1557.8333333333333  |
|user_000424|9            |1699.3333333333333  |
|user_001694|3            |1399.0              |
|user_000577|8            |1617.25             |
|user_001138|9            |1971.5555555555557  |
|user_001763|9            |1757.5555555555557  |
|user_001489|9            |2018.111111111111   |
|user_000372|1            |858.0               |
|user_001671|9            |2163.1111111111113  |
|user_000708|2            |2256.0              |
|user_001767|7            |1371.142857142857   |
|user_001617|7            |1675.4285714285713  |
|user_000289|3            |943.0               |
|user_000794|4            |2066.5              |
|user_001584|3            |1339.0              |
|user_001429|9            |2763.5555555555557  |
|user_000445|10           |1442.4              |
|user_000319|6            |2313.5              |
+-----------+-------------+--------------------+
only showing top 20 rows
```

6.4 Integration Justification

Spark enables efficient joins and aggregations across heterogeneous datasets that would be difficult and inefficient to perform within a single database system.

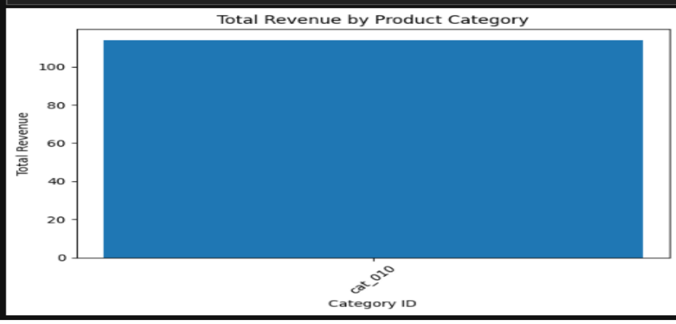
PART 4: Visualization and Insights

7. Visualizations and Business Insights

7.1 Revenue by Category

Figure 9: Revenue by Product Category

```
[29]: import matplotlib.pyplot as plt

# Convert Spark DataFrame to Pandas
revenue_pd = revenue_by_category_df.toPandas()

plt.figure()
plt.bar(revenue_pd["category_id"], revenue_pd["total_revenue"])
plt.title("Total Revenue by Product Category")
plt.xlabel("Category ID")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

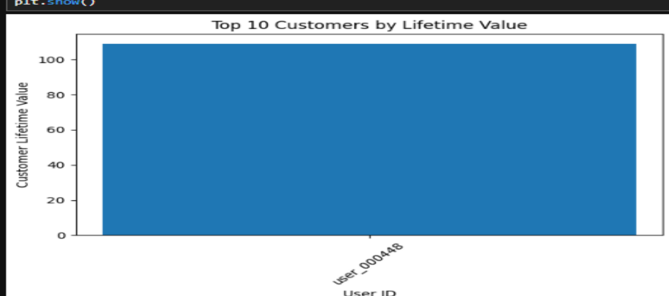


Insight: Revenue is concentrated in a limited number of product categories, suggesting areas for focused marketing and inventory optimization.

7.2 Top Customers by CLV

Figure 10: Top Customers by Lifetime Value

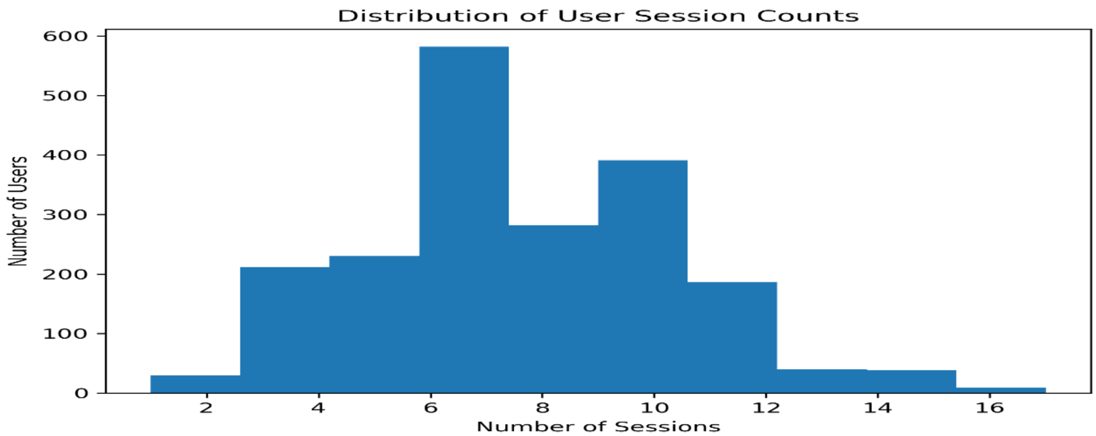
```
[30]: # Convert CLV Spark DataFrame to Pandas
clv_pd = clv_df.limit(10).toPandas()

plt.figure()
plt.bar(clv_pd["user_id"], clv_pd["customer_lifetime_value"])
plt.title("Top 10 Customers by Lifetime Value")
plt.xlabel("User ID")
plt.ylabel("Customer Lifetime Value")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Insight: A small number of customers contribute disproportionately to total revenue, making them prime candidates for loyalty programs.

7.3 Transaction Frequency Distribution

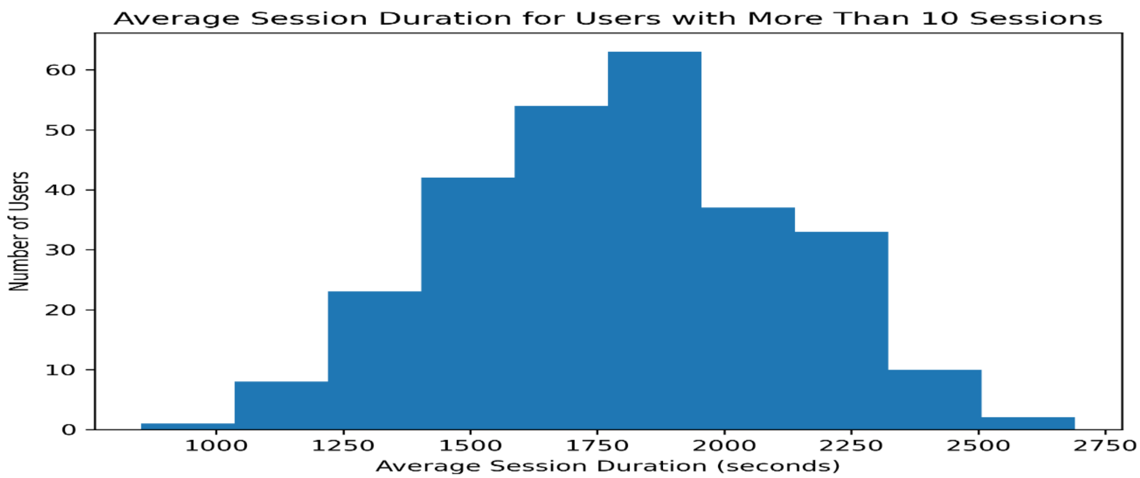
Figure 11: Distribution of Transactions per Customer



Insight: Most customers make only one purchase, indicating opportunities to improve customer retention.

7.4 Session Engagement Distribution

Figure 12: Distribution of User Session Counts and Session Duration



Insight: User engagement varies significantly and provides behavioral insights beyond transaction data alone.

8. Scalability, Limitations, and Future Work

Scalability

- MongoDB supports horizontal scaling through sharding.
- HBase scales across distributed clusters.
- Spark enables parallel data processing.

Limitations

- Synthetic dataset
- Limited transaction volume due to limited resources.

Future Work

- Real-time analytics using Spark Streaming
- Recommendation systems
- Interactive dashboards

9. Conclusion

This project demonstrates the effectiveness of a distributed multi-model analytics architecture using MongoDB, HBase, and Apache Spark. By combining appropriate data models with scalable processing, the system extracts meaningful insights from complex e-commerce data. The approach is extensible and suitable for real-world big data analytics applications.