Cloud-Based Distributed Data Processing Service

Sana AbuNada- 220212445, Farah Al-Turk - 220212936

Faculty of Information Technology
The Islamic University of Gaza

A Requirement for the Course: Cloud and Distributed Systems (SICT 4313)
Instructor: Dr. Rebhi S. Baraka

Abstract

This project focuses on developing a cloud-based data processing program using Apache Spark on the Databricks platform. The program allows users to upload a sales dataset in CSV format, analyze the data, and extract useful insights. Basic data analysis tasks such as counting records, checking missing values, and calculating summary statistics were performed to better understand the dataset.

In addition, several machine learning techniques were applied, including Linear Regression, Decision Tree Regression, and K-Means Clustering. The performance of the system was also evaluated by simulating different numbers of machines (1, 2, 4, and 8) and measuring execution time, speedup, and efficiency. The results showed that parallel processing improves execution time, but efficiency decreases as more machines are used due to overhead. Overall, the project demonstrates practical implementation of cloud computing and distributed data processing concepts.

1. Introduction

In this project, we developed a cloud-based data processing program that allows users to upload a dataset and analyze it using Apache Spark. The main purpose of this project is to apply the concepts of cloud computing and distributed data processing in a practical way, rather than focusing only on theory.

The program allows the user to upload a structured dataset in CSV format. The dataset is stored in cloud storage and then processed using Apache Spark to generate descriptive statistics and to apply several machine learning algorithms. The results are displayed to the user through a simple interface, and they are also saved for later use.

To study the performance of the cloud-based system, the same processing tasks are executed using different numbers of machines. The execution time is recorded for each case, and the results are compared to evaluate the speedup, efficiency, and scalability of the system. This helps in understanding how distributed processing improves performance when working with large datasets.

2. Cloud Program/Service Requirements

The requirements of the cloud-based program are defined based on the required system functions and user interactions. These requirements are presented in the form of user stories and use cases to clearly describe how the user interacts with the system and what the system is expected to do.

2.1 User Stories

- As a user, I want to upload a dataset in CSV format so that I can analyze my data using the cloud-based service.
- As a user, I want the system to validate the uploaded dataset so that it can be processed correctly by Apache Spark.
- As a user, I want the system to store my dataset in cloud storage so that it can be accessed and processed efficiently.
- As a user, I want to generate descriptive statistics such as the number of rows, number of columns, and missing values to better understand my dataset.
- As a user, I want to apply different machine learning algorithms on my data so that I can obtain analytical insights and predictions.
- As a user, I want to view the processing results through a simple interface and download them for later use.

2.2 Use Cases

- **Upload Dataset:** The user uploads a dataset in CSV format through the system interface. The system validates the dataset and stores it in cloud storage.
- **Data Processing:** The system loads the dataset from cloud storage and processes it using Apache Spark to generate descriptive statistics.
- **Machine Learning Analysis:** The system applies multiple machine learning tasks, including regression, classification, and clustering, using Spark MLlib.
- **Performance Evaluation:** The system executes the same processing tasks using different numbers of machines and records execution time to evaluate speedup, efficiency, and scalability.
- **Results Presentation:** The system displays the results to the user and saves them in cloud storage for download and future access.

These user stories and use cases describe the functional requirements of the system at different levels of abstraction and ensure that the developed program meets the project specifications while remaining simple and user-friendly.

3. Architecture and Design

- The Cloud-Based Data Analytics System is a cloud-based data processing application designed to upload, process, and analyze structured datasets. The system focuses on distributed data processing and machine learning using Apache Spark. It follows a simple layered architecture that separates user interaction, data storage, and data processing to improve organization and scalability.

- The user interface is implemented using Databricks notebooks, which provide an interactive and user-friendly environment for uploading datasets, configuring processing options, running Spark jobs, and viewing results.

- The system is designed as a cloud-based data processing application that separates user interaction, data storage, and data processing.

- The user interacts with the system through the notebook-based interface to upload structured datasets in CSV format and view the analysis results.

- After uploading the dataset, the system validates the CSV file to ensure it is suitable for processing with Apache Spark.

- The dataset is stored in cloud storage and is used as input for all processing tasks.

- Apache Spark is used as the main processing engine to perform distributed data analysis and machine learning tasks.

- The system generates descriptive statistics such as the number of rows, number of columns, and missing values.

- Several machine learning algorithms, including regression, classification, and clustering, are applied using Spark MLlib.

- The processing tasks are executed in a distributed manner to improve performance and support scalability.

- The generated results are displayed to the user through the notebook interface and saved in cloud storage for future access.

Figure 1 shows the interaction between the user interface, cloud storage, and Apache Spark processing engine in a distributed environment.

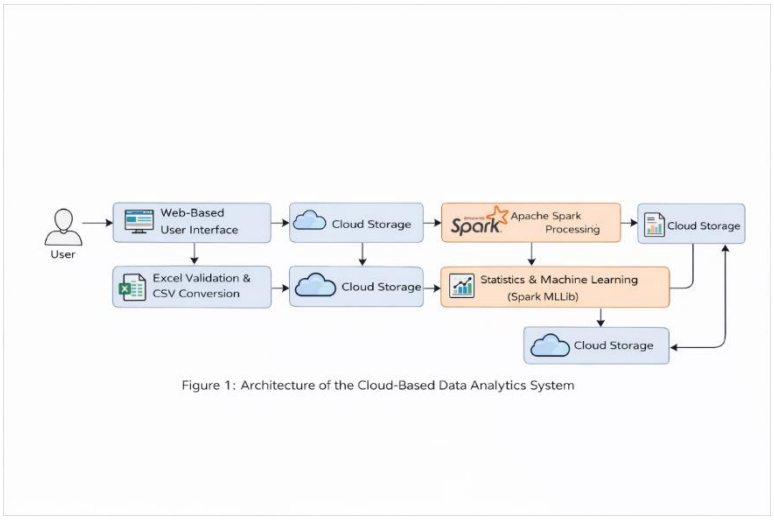

Figure 1: Architecture of the Cloud-Based Data Analytics System

4. Implementation

This section describes the implementation of the cloud-based data analytics system based on the architecture presented in the previous section. The implementation follows a layered design that separates data storage, data processing, and result presentation, which helps improve scalability and organization.

The system was implemented using **Databricks** as the cloud platform and **Apache Spark** as the main distributed data processing engine. All implementation steps were carried out inside a Spark notebook environment using **PySpark**, which allows executing distributed data processing tasks in a simple and interactive way.

4.1 Dataset Upload and Storage

The dataset is uploaded to the Databricks cloud platform and stored in managed cloud storage.
In the Databricks Free Edition, uploaded CSV files are internally managed and accessed as tables, which allows Apache Spark to efficiently load and process the data in a distributed environment.

Once the dataset was uploaded to the Databricks cloud platform, Apache Spark loaded the data into a DataFrame to verify that the dataset was ready for further processing, as illustrated in Figure 1.

Table +

	order_id	product	category	price	quantity	region	total_sales
1	1	Desk	Electronics	245	9	North America	2205
2	2	Chair	Furniture	831	5	Europe	4155
3	3	Desk	Furniture	1059	5	South America	5295
4	4	Phone	Furniture	248	1	Africa	248
5	5	Tablet	Furniture	416	5	North America	2080
6	6	Monitor	Accessories	974	3	Africa	2922
7	7	Desk	Furniture	560	2	Europe	1120
8	8	Chair	Electronics	395	8	South America	3160
9	9	Laptop	Electronics	1440	3	Africa	4320
10	10	Desk	Accessories	461	2	Africa	922
11	11	Tablet	Furniture	938	9	Europe	8442
12	12	Monitor	Accessories	903	4	North America	3612
13	13	Desk	Accessories	1164	8	Asia	9312
14	14	Phone	Accessories	374	5	South America	1870
15	15	Monitor	Electronics	334	7	South America	2338

10,000+ rows | Truncated data | 1.68s runtime — Refreshed 10 minutes ago

Figure 1: *Sales dataset loaded after uploading to the Databricks cloud platform.*

4.2 Data Processing and Analysis

After loading the dataset, several data analysis tasks are performed using Spark DataFrame APIs and Spark SQL functions. First, basic descriptive statistics are generated, including:

- Number of rows in the dataset
- Number of columns
- Column names

To evaluate data quality, the system calculates the number of missing values in each column using Spark aggregation functions. This step ensures that the dataset is suitable for further analysis.

In addition, a general overview of the data is provided by computing minimum and maximum values for numerical attributes such as **price**, **quantity**, and **total sales**. These statistics help in understanding the range and distribution of the data.

As part of the initial data analysis, basic dataset information was extracted, including the number of rows, number of columns, and column names, to provide an overview of the dataset structure.

```
Number of rows: 100000
Number of columns: 7
Column names: ['order_id', 'product', 'category', 'price', 'quantity', 'region', 'total_sales']
```

Figure 2: *Basic dataset information showing the number of rows, number of columns, and column names.*

In addition, summary statistics were calculated to understand the range of numerical attributes in the dataset, including minimum and maximum values for price, quantity, and total sales.

```
+-------+-----+--------+-----------+
|summary|price|quantity|total_sales|
+-------+-----+--------+-----------+
|    min|  100|       1|        100|
|    max| 1499|       9|      13491|
+-------+-----+--------+-----------+
```

Figure 3: *Minimum and maximum values for numerical attributes (price, quantity, and total sales) computed using Spark.*

4.3Aggregations and Insights

To generate meaningful insights, aggregation operations are applied to the dataset. For example, total sales are grouped by product category using Spark's `groupBy` and `sum` functions. This demonstrates how distributed processing can efficiently analyze large datasets and extract useful information.

4.4 Mapping Between Program and Cloud Platform

The dataset is stored in cloud-managed storage provided by the Databricks platform, and Apache Spark is used as the main processing engine to perform distributed data analytics and machine learning tasks.

4.5 Deployment on the Cloud Platform

The system was deployed on the cloud platform by creating a Spark notebook in Databricks. The dataset was uploaded to cloud storage, and all processing and analysis steps were executed directly within the notebook. This deployment approach simplifies development and

allows easy experimentation, execution, and result visualization without the need for complex configuration.

Overall, the implementation demonstrates how cloud computing concepts and distributed data processing can be applied in practice using Apache Spark and Databricks to analyze large datasets efficiently.

4.6 Machine Learning

4.6.1 Linear Regression

As part of the implementation phase, a Linear Regression model was developed using Apache Spark MLlib to predict total sales based on numerical features in the dataset. The input features used for this model were *price* and *quantity*, while *total_sales* was selected as the target variable.

Before training the model, a feature vector was created using the VectorAssembler provided by Spark MLlib. The dataset was then split into training and testing sets to evaluate the model performance.

The Linear Regression model was trained in a distributed manner using Spark. After training, the model coefficients and intercept were extracted to understand the relationship between the input features and the predicted output.

To evaluate the model performance, Root Mean Squared Error (RMSE) and R^2 score were calculated. The results showed a high R^2 value, indicating that the model is able to explain a large portion of the variance in total sales. These results demonstrate the effectiveness of distributed machine learning using Apache Spark in a cloud environment.

The results of the Linear Regression model, including the model coefficients, intercept, RMSE, and R^2 score, are shown in Figure 4.

```
Coefficients: [5.0090144973574064,799.7085988593947]
Intercept: -4006.4243574653883
```

Figure 4: Linear Regression model output showing coefficients, intercept, RMSE, and R^2 score.

4.6.2 KMeans Clustering

As part of the implementation phase, a KMeans clustering model was implemented using Apache Spark MLlib. The main goal of this model is to group similar sales records together based on their characteristics, without using predefined labels.

Before applying the KMeans algorithm, the numerical features **price**, **quantity**, and **total_sales** were combined into a single feature vector using the **VectorAssembler** provided by Spark MLlib. This step prepares the data for the clustering process.

The KMeans model was then trained in a distributed way using Apache Spark, with the number of clusters set to three (k = 3). During training, Spark automatically assigned each data record to the nearest cluster based on similarity.

After training the model, the cluster centers were extracted and analyzed. Each cluster center represents a group of sales records with similar patterns in terms of price, quantity, and total sales. This helps in understanding different types of sales behavior in the dataset.

This experiment shows how unsupervised machine learning algorithms like KMeans can be efficiently applied using Apache Spark in a cloud environment to analyze large datasets.

The resulting cluster centers obtained from the KMeans model are illustrated in Figure 5. These centers summarize the main characteristics of each cluster based on the selected numerical features.

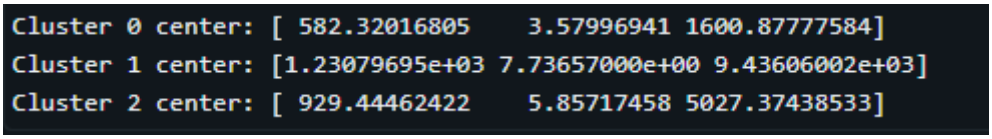

Figure 5: KMeans clustering results showing the cluster centers based on price, quantity, and total sales.

4.6.3 Decision Tree Regression

The Decision Tree Regression model was applied using Apache Spark MLlib to predict total sales based on price and quantity. The model was trained in a distributed environment and evaluated using RMSE. The results show that the Decision Tree model achieved lower error compared to Linear Regression, indicating better prediction accuracy and the ability to capture non-linear patterns in the data.

The model structure allows it to handle non-linear relationships between the input features and the target variable more effectively than linear models.

The performance of the Decision Tree Regression model was evaluated using Root Mean Squared Error (RMSE), and the prediction results are illustrated in Figure 6.

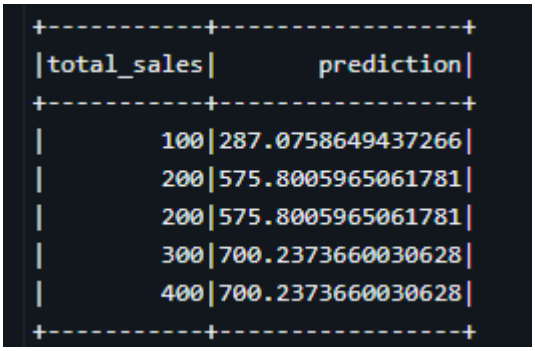

Figure 6: Decision Tree Regression results showing predicted total sales and RMSE evaluation metric.

4.6.4 Classification (Category Prediction)

As part of the implementation phase, a classification model was developed using Apache Spark MLlib to predict the product category based on numerical features in the dataset. The features used for this task were price, quantity, and total_sales, while the category column was selected as the target label.

Since the category attribute is a categorical variable, it was first converted into numerical labels using the StringIndexer technique. After that, a feature vector was created using the VectorAssembler to prepare the data for machine learning.

A Logistic Regression classifier was then trained in a distributed environment using Spark. The trained model was used to predict product categories on the test dataset. To evaluate the model performance, classification accuracy was calculated. The results show that the model is able to perform category prediction, demonstrating the use of distributed classification techniques using Apache Spark in a cloud environment.

The classification results obtained from the Logistic Regression model, including predicted product categories and accuracy, are illustrated in Figure 7.

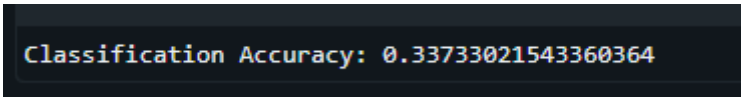

Figure 7: Classification results showing predicted product categories and model accuracy using Logistic Regression.

4.7 Results Integration and Presentation

After completing the data analysis and machine learning tasks, all generated results were unified into a single structured table. The results include descriptive statistics and outputs from regression, classification, and clustering models. To handle heterogeneous outputs, the results were organized into a unified schema consisting of result type, metric, and value. The final results were displayed directly within the Databricks notebook, ensuring that all outputs are stored and accessible in the cloud-based environment.

After integrating the outputs of all data analysis and machine learning tasks, the final results were unified into a single structured table and displayed to the user, as shown in Figure 8.

Table +

	result_type	metric	value
1	basic_statistics	Number of rows	100000
2	basic_statistics	Number of colum...	7
3	missing_values	order_id	0
4	missing_values	product	0
5	missing_values	category	0
6	missing_values	price	0

Figure 8: Unified results table containing descriptive statistics and outputs from regression, classification, and clustering models.

5. Experiments and Evaluation

The experiments were conducted using Apache Spark on the Databricks cloud platform.
To study scalability, the same Spark job was executed multiple times while simulating different numbers of machines (1, 2, 4, and 8) by varying Spark configuration parameters. Execution time, speedup, and efficiency were measured for each configuration.

Note: The number of machines was simulated using Spark configuration parameters in the Databricks environment.

.

Cluster Size	Execution Time (sec)	Speedup	Efficiency
1 Machine	**0.2235**	**1**	**1**
2 Machines	**0.185**	**1.2083**	**0.6041**
4 Machines	**0.1572**	**1.4219**	**0.3555**
8 Machines	**0.1983**	**1.1266**	**0.1408**

Speedup = T_1 / T_n where: T_1 is the execution time using 1 machine, n= 1, 2, 4, and 8.

Efficiency = Speedup / Number of machines

6. User Support

This section explains how users can access and use the cloud-based data processing program.

The program is implemented and executed on the **Databricks cloud platform**. Users can access the program through the Databricks notebook link provided below. No special installation is required, as all processing is performed in the cloud environment.

How to Use the Program

1. The user opens the Databricks notebook using the provided link.
2. The dataset (CSV file) is uploaded to the Databricks cloud platform using the provided data upload interface.
3. The user runs the notebook cells sequentially.
4. The program loads the dataset and performs data analysis.
5. Machine learning models (Linear Regression, Decision Tree, and K-Means) are executed.
6. Performance evaluation results, including execution time, speedup, and efficiency, are displayed directly in the notebook.

No username or password is required other than the user's Databricks account credentials.

7. Conclusion

In this project, a cloud-based data processing system was developed using Apache Spark on the Databricks platform. The system allows users to upload a dataset, perform data analysis, apply machine learning models, and evaluate performance in a cloud environment.

Through the experiments, we studied the effect of using different numbers of machines on execution time, speedup, and efficiency. The results showed that parallel processing helps reduce execution time and improve performance. However, the efficiency decreases when more machines are used due to additional overhead such as task coordination and communication between machines.

Overall, this project helped us understand how cloud computing and distributed data processing work in practice. It demonstrated the benefits and limitations of parallel execution and provided practical experience with Apache Spark and cloud-based systems.

References

1. Apache Spark Documentation.
   https://spark.apache.org/docs/latest/
2. Databricks Documentation.
   https://docs.databricks.com/
3. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010).
   Spark: Cluster Computing with Working Sets.
   Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing.
4. Dean, J., & Ghemawat, S. (2008).
   MapReduce: Simplified Data Processing on Large Clusters.
   Communications of the ACM, 51(1), 107–113.
5. Han, J., Kamber, M., & Pei, J. (2011).
   Data Mining: Concepts and Techniques.
   Morgan Kaufmann.