University of Kragujevac

Artificial Intelligence

Project for an Processing Large Volumes of Data (VI-23)

Cryptocurrency Data Processing and Analysis

Student:

Babaev Bogdan 0866511

Kragujevac, may 2025

Contents

1. The task

Objective

This project was developed as part of the course *Processing Large Volumes of Data* (Обрада великих количина податока, ВИ-23) to create a scalable Big Data application for collecting, processing, and analyzing historical cryptocurrency price data for Bitcoin (BTC) and Ethereum (ETH). The main objectives align with the course's focus on handling large datasets efficiently:

- Collect large-scale data from an external API.
- Perform data aggregation using Apache Spark, leveraging RDDs and SparkSQL.
- Optimize processing through parallel computing techniques.
- Ensure scalability and data security for real-world applications.

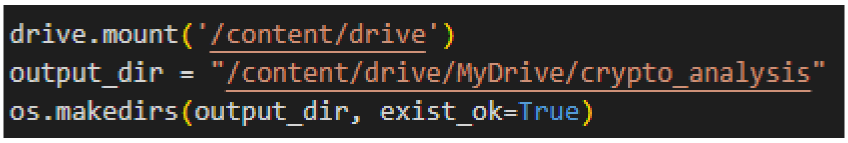
Solution Overview

The application retrieves price and trading volume data for BTC and ETH from January 1, 2023, to January 1, 2025, via the CryptoCompare API. It processes the data using Apache Spark's structured streaming with 5-minute window aggregations, stores results in a SQLite database, and encrypts outputs for security. The solution showcases Big Data technologies applied to financial data analysis, emphasizing scalability and performance.

2. Environment Setup

The project was executed in Google Colab, utilizing Google Drive for persistent storage. Required libraries (pandas, pyspark, requests, pycryptodome) were installed to support data processing, API interaction, and encryption. Data was stored in the directory /content/drive/MyDrive/crypto_analysis.

Setup Code:



```
drive.mount('/content/drive')
output_dir = "/content/drive/MyDrive/crypto_analysis"
os.makedirs(output_dir, exist_ok=True)
```

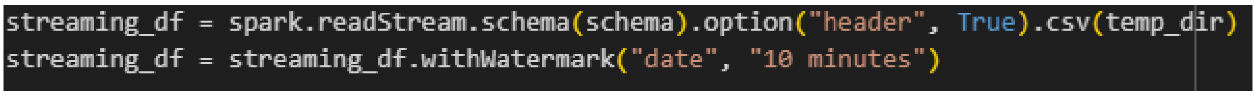
2.2. Data Collection

Historical price and volume data for BTC and ETH were collected via the CryptoCompare API. The average price was calculated as (high + low) / 2. Data was saved as a CSV file (crypto_raw_data.csv) for further processing.

2.3. Streaming and Aggregation

Apache Spark was used for structured streaming to process data in near real-time. Data was aggregated over 5-minute windows, calculating average prices and volumes. Results were saved as CSV files, demonstrating Spark's ability to handle large-scale data efficiently.

Aggregation Code:



```
streaming_df = spark.readStream.schema(schema).option("header", True).csv(temp_dir)
streaming_df = streaming_df.withWatermark("date", "10 minutes")
```

2.4. Data Storage

Aggregated data was stored in a SQLite database (crypto.db) with tables for Cryptocurrency, Date, and PriceData. Bulk insertion was used to optimize database performance, aligning with Big Data principles of efficient data handling.

2.6. Optimization

The project incorporated several optimization techniques to enhance performance, critical for processing large datasets:

- **Parallel Processing**: Spark's distributed computing model parallelized data processing across clusters.
- **Watermarking**: A 10-minute watermark was applied to handle late data in streaming.
- **Early Filtering**: Invalid data was filtered early to reduce processing overhead.
- **Bulk Inserts**: Mass insertion into SQLite minimized database write operations.

3. Technologies Used

- **Apache Spark**: For distributed processing, streaming, and aggregation (RDDs, SparkSQL).
- **Python**: For scripting, API interaction, and data manipulation.
- **Pandas**: For preprocessing and data structuring.
- **SQLite**: For relational storage of aggregated data.
- **CryptoCompare API**: For retrieving cryptocurrency data.
- **PyCryptoDome**: For AES encryption of outputs.
- **Google Colab and Google Drive**: For cloud-based execution and storage.

4. Results

- Developed a scalable Big Data application for processing cryptocurrency data.
- Performed 5-minute window aggregations for BTC and ETH, leveraging Spark's streaming capabilities.
- Stored results in a SQLite database and secured them with AES encryption.
- Demonstrated proficiency in Big Data frameworks, parallel processing, and optimization, meeting the course requirements.

5. Conclusion

The project successfully addresses the challenges of processing large volumes of data, applying key Big Data concepts such as distributed computing, streaming, and optimization. The solution is scalable and can be extended to real-time processing with tools like Kafka. The source code, output files, and database are available for review.

https://github.com/b0gdaan/VI-23