Real-Time Stock Market Data Processing Pipeline Using Apache Kafka and AWS

Shubham Manisha Naik
Department of Applied Data Science,
San Jose State University,
San Jose, CA, USA
Email: shubhammanisha.naik@sjsu.edu

***Abstract*—This paper presents the implementation of a comprehensive real-time stock market data processing pipeline leveraging Apache Kafka and AWS services. The pipeline encompasses data simulation, real-time streaming, distributed storage, and querying capabilities. The project underscores the practical application of real-time data engineering concepts to handle, process, and analyze dynamic datasets effectively. This paper provides a detailed walkthrough of the system architecture, implementation steps, challenges faced, and solutions devised.**

I. Introduction

Real-time data processing is pivotal for numerous applications, including financial markets, logistics, and e-commerce. Stock trading, in particular, demands high-throughput, low-latency systems to process events as they occur. This project employs Apache Kafka, a distributed event-streaming platform, and Amazon Web Services (AWS) to design a pipeline capable of processing real-time stock market data. The system not only demonstrates real-time streaming but also highlights the integration of cloud-based tools for persistent storage and analytics.

II. System Architecture

The architecture of the pipeline is shown in Fig. 1. It involves the following components:

1) **Data Simulation:** Generates stock market-like events to simulate real-world conditions.
2) **Apache Kafka:** Manages data streaming between producers and consumers.
3) **Amazon S3:** Provides scalable storage for streamed data.
4) **AWS Glue:** Crawls and catalogs the data schema.
5) **Amazon Athena:** Facilitates querying of data using SQL.

III. Implementation

The implementation is divided into five key steps:

A. Data Simulation

Data simulation is the first step to mimic real-time stock market events. Python's Pandas library is used to load and randomly sample data from a pre-existing dataset.

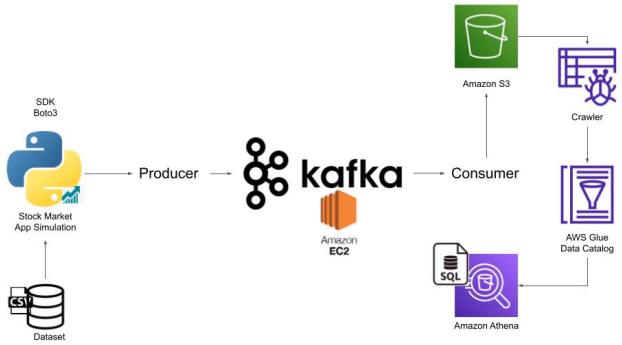

Fig. 1: System Architecture for Real-Time Data Processing.

```python
import pandas as pd
import time

data = pd.read_csv('stock_data.csv')
while True:
    sample =
        data.sample(1).to_dict(orient='records')[0]
    print(sample)  # Replace with Kafka producer
    time.sleep(1)
```

Listing 1: Data Simulation Code

B. Kafka Setup and Configuration

Kafka is installed on an AWS EC2 instance, following these steps:

- Install Kafka and Zookeeper on EC2.
- Configure a Kafka topic to handle stock market events.
- Use Python's Kafka client library (`Kafka-python`) to produce and consume data.

```python
from kafka import KafkaProducer
import json

producer = KafkaProducer(
    bootstrap_servers=['your-ec2-public-ip:9092'],
    value_serializer=lambda v:
        json.dumps(v).encode('utf-8')
)

producer.send('stock_topic', {'stock': 'AAPL',
    'price': 150.25})
```

Listing 2: Kafka Producer Code

C. Data Storage in S3

The consumer reads streamed data and uploads it to an Amazon S3 bucket:

```
import s3fs
import json

s3 = s3fs.S3FileSystem()
data = {"stock": "AAPL", "price": 150.25}

with
    s3.open('s3://your-bucket-name/stock_data.json',
    'w') as f:
    f.write(json.dumps(data))
```

Listing 3: S3 Data Upload Code

D. AWS Glue and Athena

AWS Glue crawls the S3 bucket to build a schema catalog, making the data queryable via Athena:

1) Set up a Glue crawler targeting the S3 bucket.
2) Run the crawler to populate the Glue catalog.
3) Use Athena to query the data with SQL.

IV. Results and Discussion

The pipeline successfully streamed stock market data, stored it in Amazon S3, and queried it in real-time using Athena. Table I shows sample results queried via Athena.

TABLE I: Sample Athena Query Results

Stock	Date	Price	Volume
AAPL	2023-12-28	150.25	1000
TSLA	2023-12-28	720.50	1500

V. Challenges and Solutions

1) **Server Overload:** The Kafka server encountered overload during high-frequency data ingestion. This was mitigated by implementing data throttling with delays.
2) **Authentication Errors:** AWS CLI was configured to ensure secure access to S3.
3) **Data Schema Issues:** AWS Glue automatically inferred schemas for seamless querying in Athena.

VI. Conclusion and Future Work

This project demonstrates the integration of Apache Kafka with AWS services to build a robust real-time data processing pipeline. Future enhancements include integrating machine learning models for stock price prediction and extending the pipeline to support additional data sources such as IoT sensors.