Distributed Systems Project

Developing a Real-Time Financial Analysis & Trading System

Introduction

In financial markets, making swift, well-informed decisions is paramount. This project enables students to develop a system to analyze simulated financial data in real-time to generate actionable trading signals, utilizing distributed computing principles, microservices architecture, and stream processing.

Objective

Construct a scalable, distributed system that processes and analyzes simulated financial data to create actionable trading signals. The system will consist of components for data ingestion, stream processing, signal generation, and data visualization, offering real-time financial insights.

Scope

You'll work with simulated financial data, analyzing several fields such as stock_symbol, opening_price, etc. The system will compute and visualize mandatory trading indicators on a dashboard, allowing users to interpret stock performance easily.

Data

You'll receive a script, `generator.py`, to simulate data producers, serving as a starting point that can be modified. The `manager.sh` is a bash script that helps to setup, start, and stop the generator. Please read the script for more information.

Team Structure

Teams should consist of no more than two students.

Mandatory Trading Indicators

The following trading indicators must be calculated and delivered as part of this project:

1. **Moving Average:** Helps identify the direction of the trend by averaging the closing prices of stocks over a specified number of periods.
2. **Exponential Moving Average:** Reacts more quickly to price changes than the simple moving average by giving more weight to the most recent prices.
3. **Relative Strength Index (RSI):** Measures the speed and change of price movements to identify overbought or oversold conditions in the market.

Project Overview

> Developing an architecture based on **microservices** and incorporating **stream processing** is **mandatory**, as is using **websockets** to present real-time data.

1. **Data Generator Script:** Generates and modifies simulated financial data from multiple sources.
2. **Data Ingestion Service:** Receives, validates, and forwards the simulated data.
3. **Stream Processing Service:** Processes the data in real-time, analyzing it and calculating the mandatory trading indicators.
4. **Trading Signal Service:** Produces buy/sell signals based on the analyzed data.
5. **Notification Service:** Informs users instantly when a trading signal is generated.
6. **Visualization Service:** Represents processed data and signals on a user-friendly dashboard.
7. **Load Balancer:** Manages incoming traffic across multiple servers.
8. **Aggregator Service:** Summarizes each stock's performance.
9. **User Interface:** Allows users to interact with the system, view visualized data, and receive notifications.

Example Workflow

1. **Data Generation:** Simulated financial data is created by the Data Generator Script.
2. **Data Ingestion:** Data is validated and sent to the Stream Processing Service.
3. **Stream Processing:** Data is processed, and mandatory trading indicators are calculated.
4. **Signal Generation:** Buy/sell signals are generated and sent to the Notification Service.
5. **Notification:** Users receive instant notifications of the generated signals.
6. **Data Aggregation:** Data is aggregated and summarized.
7. **Visualization:** Data and signals are displayed on the dashboard in real-time.
8. **User Interaction:** Users can view detailed information and receive real-time trading signals, facilitating informed trading decisions.

Getting Started

The `generator.py` is provided for getting started and can be modified as per your needs. Implement your desired interface, generate multiple types of data, but keep the request fields intact. The rest can be modified as per your requirements.

Technology Stack

You may use your preferred programming languages and frameworks. However, you cannot remove the assumptions (e.g., stream processing) described directly in the project description.

Working

Create a repository on GitHub and share it with mohammad.siavashi@outlook.com so that your work/commits are trackable. The repository should demonstrate the collaboration of both team members. Projects that do not show progress over time will not be scored.

Assumptions

Before making any assumption, contact your TA for clarification.

Academic Integrity

Maintain the highest standards of honesty and integrity throughout the project. Any form of plagiarism or dishonesty will result in severe academic penalties.

Contact

For any questions or clarifications, contact mohammad.siavashi@outlook.com anytime.

Evaluation Criteria

- Software Architecture and Compliance with the Designed Architecture
- Microservice Design
- Usability
- Scalability
- Accuracy of Signals
- Performance (Demonstrate the throughput of your architecture at presentation day)

Deliverables

- Source Code (Well-commented)
- Project Report (including project architecture, challenges faced, and how they were addressed)
- Presentation (Summary of approach, architecture, and results. Scalability of stream processing services using Kubernetes should be demonstratable during the presentation.)

Bonuses

1. **Testing on Real-World Dataset:**
    - Implementing and testing the system on a real-world dataset will earn you bonus points.
    - For this, you need to implement a new service that reads from your dataset and integrates with the existing system.
2. **Distributed Caching:**
    - Integrating a caching solution like Redis or Memcached will be considered a bonus.

- This should be aimed at reducing database load and improving overall system performance.