Eventual Broker

Software Architecture

April 4, 2024

Brae Webb & Richard Thomas

1 Brief

Now that we are over halfway through software architecture, we should have sufficient knowledge to start making a profit. Instead of applying for positions as a software architect, we have decided to use our knowledge to design Eventual Broker, a financial trading system that monitors market data and triggers trades based on certain conditions, such as price movements, news events, or other market data.

Eventual Broker aims to be a highly reliable and scalable system capable of processing a large volume of market data and trades in real-time. Some trading strategies (e.g. arbitrage[1] or high-frequency trading[2]) require decisions to be made, and trades placed, in very short time periods. Eventual Broker's decision making logic and trade processing should be highly performant. You want to design a system that is extensible, as you are able to consume more data sources, you want your core trading logic to be able to process the new data streams.

2 Requirements

The system should:

1. Be able to receive market data from multiple sources in real-time.
2. Be able to process the market data and trigger trades based on certain conditions, as specified by a trading logic unit(s).
3. Cope with large volumes of data and trades.
4. Be scalable to handle an increasing number and variety of data sources.
5. Be observable to appropriately monitor trading and the performance of historical trades.

3 Outline

Introduction (5 minutes)

Introduction to the brief, what do our input data streams look like, what are the requirements, where to start.

Planning (10 minutes)

In small groups, attempt to enumerate potential sources of input data. Of course, we will need feeds informing us of changes in stock values. What other sources of data might be relevant to give you an edge? Discuss your existing *domain knowledge* with your peers. What types of financial instruments will you trade (e.g. stocks, bonds, options, futures, etc.)?

[1] https://online.hbs.edu/blog/post/what-is-arbitrage
[2] https://www.investopedia.com/terms/h/high-frequency-trading.asp

Design (25 minutes)

In your group, design the components that your system requires. Each component should have a brief description of the functionality that it provides the system. You will also need to design the flow of information through your system. This flow of information results in trades being performed.

Your final design can include more sources of information than originally planned. Make sure that you are considering risk management strategies and observability. How are the system's decisions being reported? How are the results of these decisions evaluated?

Presentation (10 minutes)

In the remaining time, each group should present their proposed design. This is an opportunity for discussion amoungst the class to point out limitations of the proposed system designs.

4 Design Challenges

Challenge 1: Duplicate Data

The market data from different sources may overlap, leading to duplicate data being processed by the system. For example if you are monitoring the stock price of Apple, you may receive data from multiple sources, each of which may have a different delay. How would you modify your design to prevent this?

Challenge 2: Regulation Compliance

How can you design the system to comply with financial regulations, such as those related to insider trading, market manipulation, or other requirements? Can you use auditing, logging, or other techniques to track and monitor trading activities?

Challenge 3: Trading Manipulation

Now that our system is not performing market manipulation, we need to consider preventing others from manipulating our trading system [1]. For example, if you are monitoring social networking sites, how could you prevent one, or a few, users manipulating positive tweets of a company to trigger buying their shares?

References

[1] C. Yagemann, S. P. Chung, E. Uzun, S. Ragam, B. Saltaformaggio, and W. Lee, "On the feasibility of automating stock market manipulation," in *Annual Computer Security Applications Conference*, ACSAC '20, (New York, NY, USA), p. 277–290, Association for Computing Machinery, 2020.