CloudEx: Implementation of a Financial Exchange in the Cloud

Vinay Sriram, Jinkun Geng, Vig Sachidananda, Ahmad Ghalayini, Balaji Prabhakar, Mendel Rosenblum
Exchange Overview
Gateway Protocol

- Each gateway serves some subset of the market participants, and processes submitted orders.

When an order is received by the gateway, it:

1. Timestamps the order.
2. Assigns the client ID to the order.
3. Assigns an order ID to the order.
4. Sends the order to the sequencer.
5. Receives an acknowledgement from the sequencer.
6. Returns the order ID to the client.
Matching Engine Protocol

When an order is received by the matching engine, it validates the order and then processes it according to the order type using the following rules.

1. **Market order:**
   - Fill as much of the order as possible until the opposite limit order book is empty, at the best available price.

2. **Limit order:**
   - Fill as much of the order as possible, while the best available price is as good as or better than the specified limit price.

3. **Cancel Order:**
   - Remove the order from the appropriate limit order book if the order still exists (i.e. hasn’t been matched yet).
Matching Engine Protocol

Limit Sell Queue

<table>
<thead>
<tr>
<th>Sell</th>
<th>50</th>
<th>Limit</th>
<th>C3</th>
<th>$120</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sell</td>
<td>750</td>
<td>Limit</td>
<td>C2</td>
<td>$40</td>
</tr>
</tbody>
</table>

Limit Buy Queue

<table>
<thead>
<tr>
<th>Buy</th>
<th>100</th>
<th>Limit</th>
<th>C5</th>
<th>$30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy</td>
<td>50</td>
<td>Limit</td>
<td>C4</td>
<td>$70</td>
</tr>
</tbody>
</table>

Incoming Order

| Buy  | 900| Market | C1 | $30  |
Disseminator and Hold / Release Protocols

1. **Disseminator**: Every $T$ milliseconds, we take a snapshot of the state of the buy and sell limit order books for each symbol.

2. **Hold / Release Block**: Each piece of market data (completed trade or snapshot) is given a release time. We hold all pieces of market data in a buffer until the specified release time.

3. **Anonymized data** is sent from to the participants.
High-Level Animation: Life Cycle of an Order
System Architecture and Data Structures
Upstream of the Fairness Perimeter

Exchange REDIS Memory Store

Limit Order Books
ORDER_ID, Order Priority

Order Map
{Order_ID, TradeMessage}

Portfolio Matrix

Completed Trades

Matching Engine

SHARED MEM Sorted Orders

SEQUENCING BUFFER

Gateway

Hold/Release Block

Sequencing Buffer

ZMQ SUB Timestamped Orders

SHARE MD Sorted Orders

Matching Engine

SHARED MEM Sorted Orders

Limit Order Book Disseminator

ZMQ PUB Symbol <k>

Hold/Release Block

ZMQ PUB Symbol <k> Book

ZMQ PUB [UPDATE Symbol <k> Book]

ZMQ PUB Symbol <k> Trades

ZMQ PUB [UPDATE Symbol <k> Trades]

ZMQ PUB Client <k> Trades

ZMQ PUB [UPDATE Client <k> Trades]
Downstream of the Fairness Perimeter

Trader API
- ZMQ PUB Orders
- ZMQ SUB Orders
- ZMQ PUB Timestamped Orders

Trade Report Subscriber API
- ZMQ SUB [UPDATE Client <k> Trades]
- ZMQ SUB [UPDATE Symbol <k> Trades]

Gateway
- ZMQ SUB Orders
- ZMQ PUB Timestamped Orders

Limit Book Subscriber API
- ZMQ SUB [UPDATE Symbol <k> Book]

Hold/Release Block
- ZMQ SUB [Symbol <k> Book]
- ZMQ PUB [UPDATE Symbol <k> Book]
- ZMQ SUB [Symbol <k> Trades]
- ZMQ PUB [UPDATE Symbol <k> Trades]
- ZMQ SUB [Client <k> Trades]
- ZMQ PUB [UPDATE Client <k> Trades]

Market Participant Memory Store
- Subscribed Limit Order Books
- Order Map
  - {Order_ID, Order Priority}
  - {(Order_ID, TradeMessage)}
- Portfolio Vector
- Completed Trades
- Incomplete Orders

Example State Management Data Structure
Performance Measurements
Measurement Definitions

- Matching Engine
- Seq. Buffer
- Gateway
- Trader

Matching Engine Latency
Trade Submission Latency
Confirmation Latency
Experiments

Measurements:
1. Trade Submission Latency
2. Confirmation Latency
3. Matching Engine Latency
4. Sequencing Buffer Size

System Configurations:
• {3 Traders, 1 Gateway}
• {6 Traders, 2 Gateways}
• {24 Traders, 8 Gateways}
Matching Engine Latency Results

Median: **680 Microseconds** (First-Cut Implementation – Will be optimized)
99th Percentile: **1.8 Milliseconds**
Non-Rate-Limited Trader (Unstable)
Rate-Limited Trader (Stable)

This scheme offers **lower** confirmation latency, but **higher** trading latency. The scheme is simpler in that the client VM does not need to independently manage order confirmations; they are encapsulated in the trading API.
Comparison of Buffer Size over Time

Buffer Size over Time (Non-Rate-Limited Trading)

Buffer Size over Time (Rate-Limited Trading, 24 Clients)
Conclusions

1. This exchange will be used in the CS 349F course in spring quarter 2020.

2. Students trading on this exchange will both learn about financial technologies and high frequency trading and provide valuable trading data for future work.

3. In the future, we can implement and experiment with more complex functionality.
   a) Causal Trade Execution
   b) Matching engine algorithm/auction design
   c) Time-based cancellation policies
   d) New order types