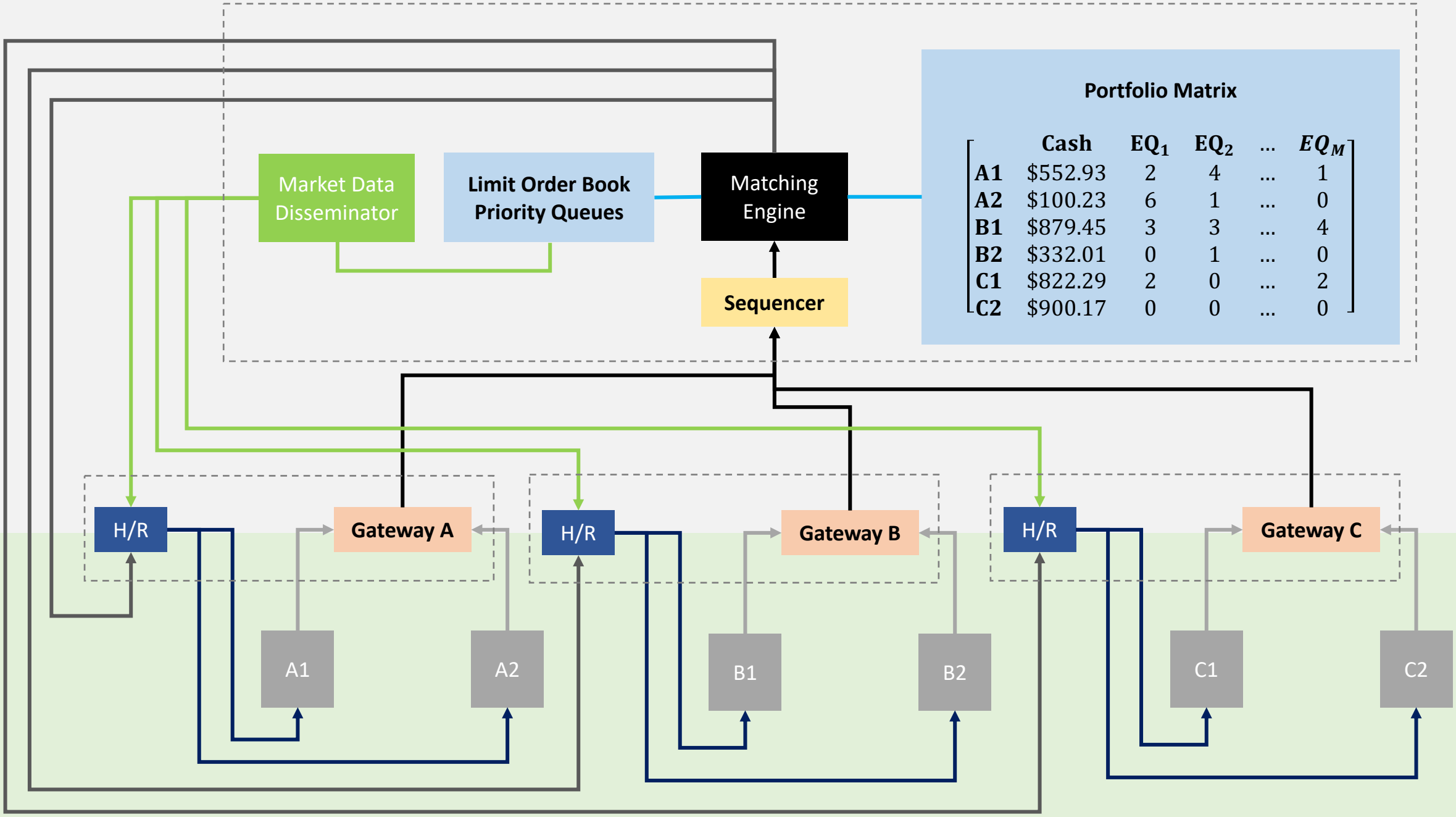
CloudEx: Implementation of a Financial Exchange in the Cloud

Vinay Sriram, Jinkun Geng, Vig Sachidananda, Ahmad Ghalayini, Balaji Prabhakar, Mendel Rosenblum

Exchange Overview


Market Data Disseminator
Limit Order Book Priority Queues
Matching Engine
Sequencer
Portfolio Matrix
Cash
EQ_1
EQ_2
...
EQ_M
A1 $552.93 2 4 ... 1
A2 $100.23 6 1 ... 0
B1 $879.45 3 3 ... 4
B2 $332.01 0 1 ... 0
C1 $822.29 2 0 ... 2
C2 $900.17 0 0 ... 0
H/R
Gateway A
H/R
Gateway B
H/R
Gateway C
A1
A2
B1
B2
C1
C2

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

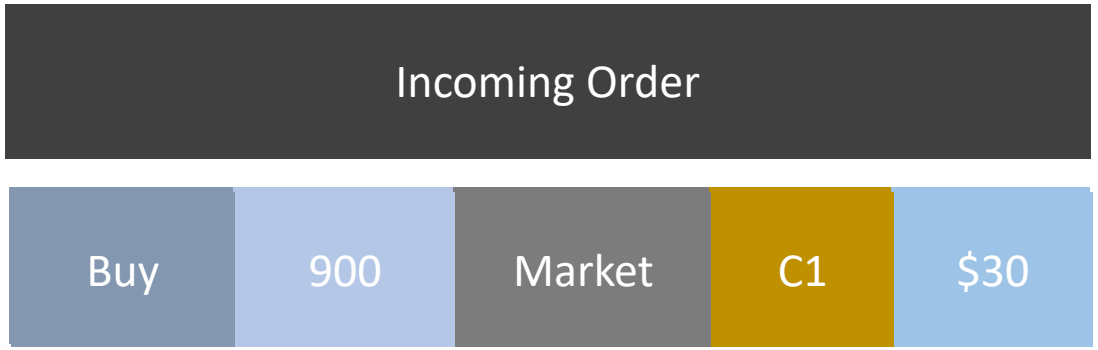
When an order is received by the matching engine, it validates the order and then processes it according to the order type using the following rules.

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



Disseminator and Hold / Release Protocols

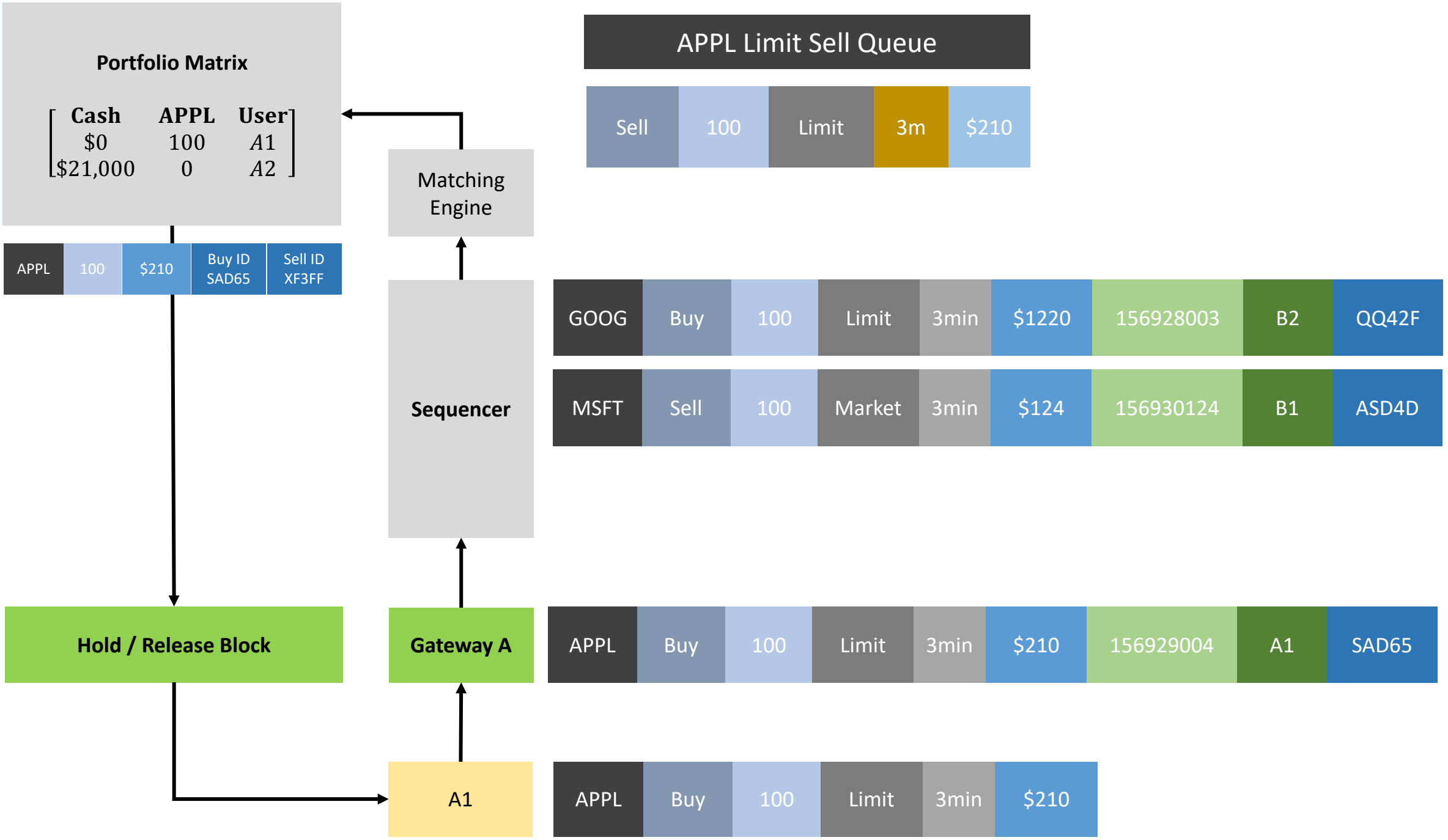
1. **Disseminator:** Every T milliseconds, we take a snapshot of the state of the buy and sell limit order books for each symbol.

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.

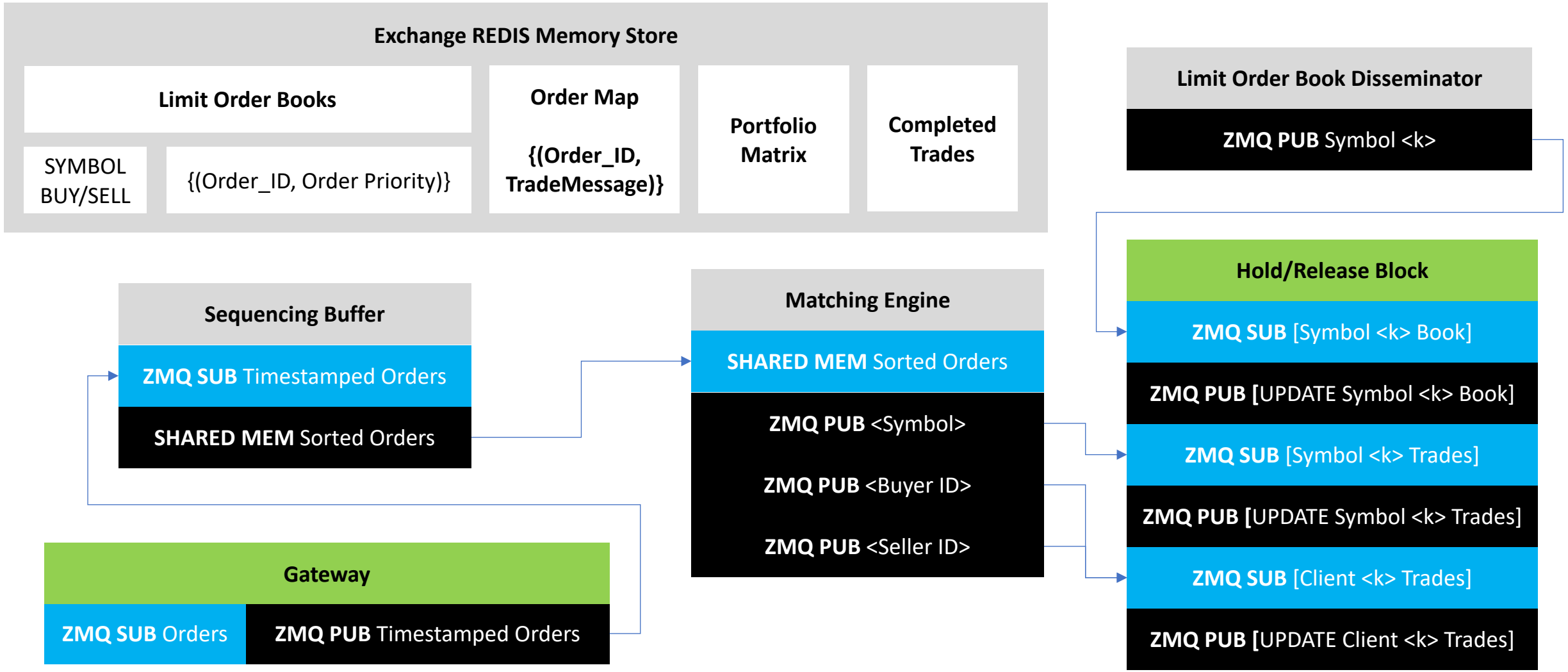
Robinhood

High-Level Animation: Life Cycle of an Order

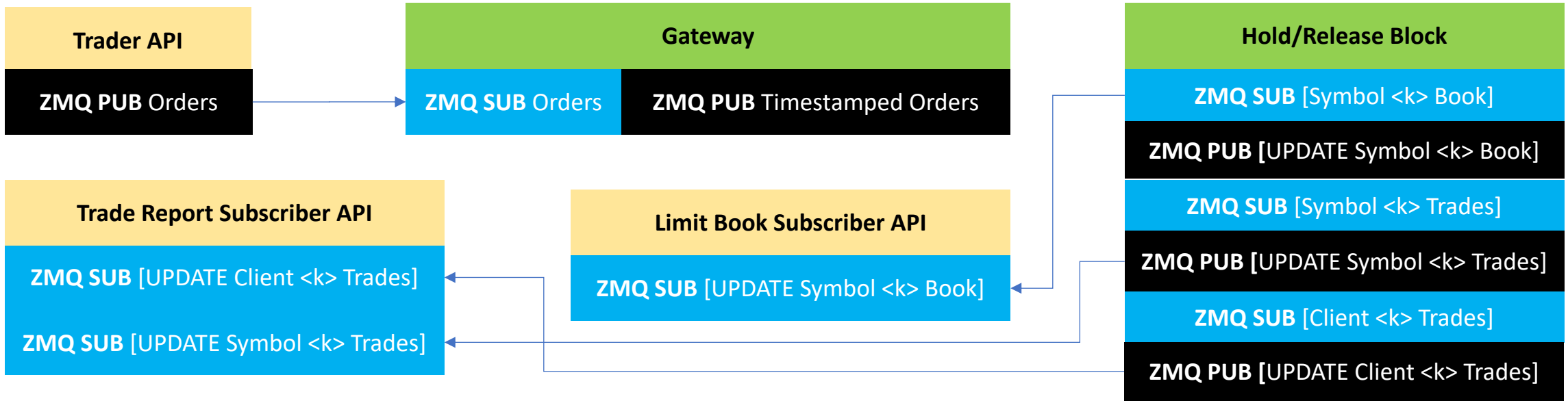

Portfolio Matrix
Cash APPL User
$0 100 A1
$21,000 0 A2
APPL 100 $210 Buy ID SAD65 Sell ID XF3FF
Hold / Release Block
A1
Gateway A
Sequencer
Matching Engine
APPL Limit Sell Queue
Sell 100 Limit 3m $210
GOOG Buy 100 Limit 3min $1220 156928003 B2 QQ42F
MSFT Sell 100 Market 3min $124 156930124 B1 ASD4D
APPL Buy 100 Limit 3min $210 156929004 A1 SAD65
APPL Buy 100 Limit 3min $210

System Architecture and Data Structures

Upstream of the Fairness Perimeter



Downstream of the Fairness Perimeter



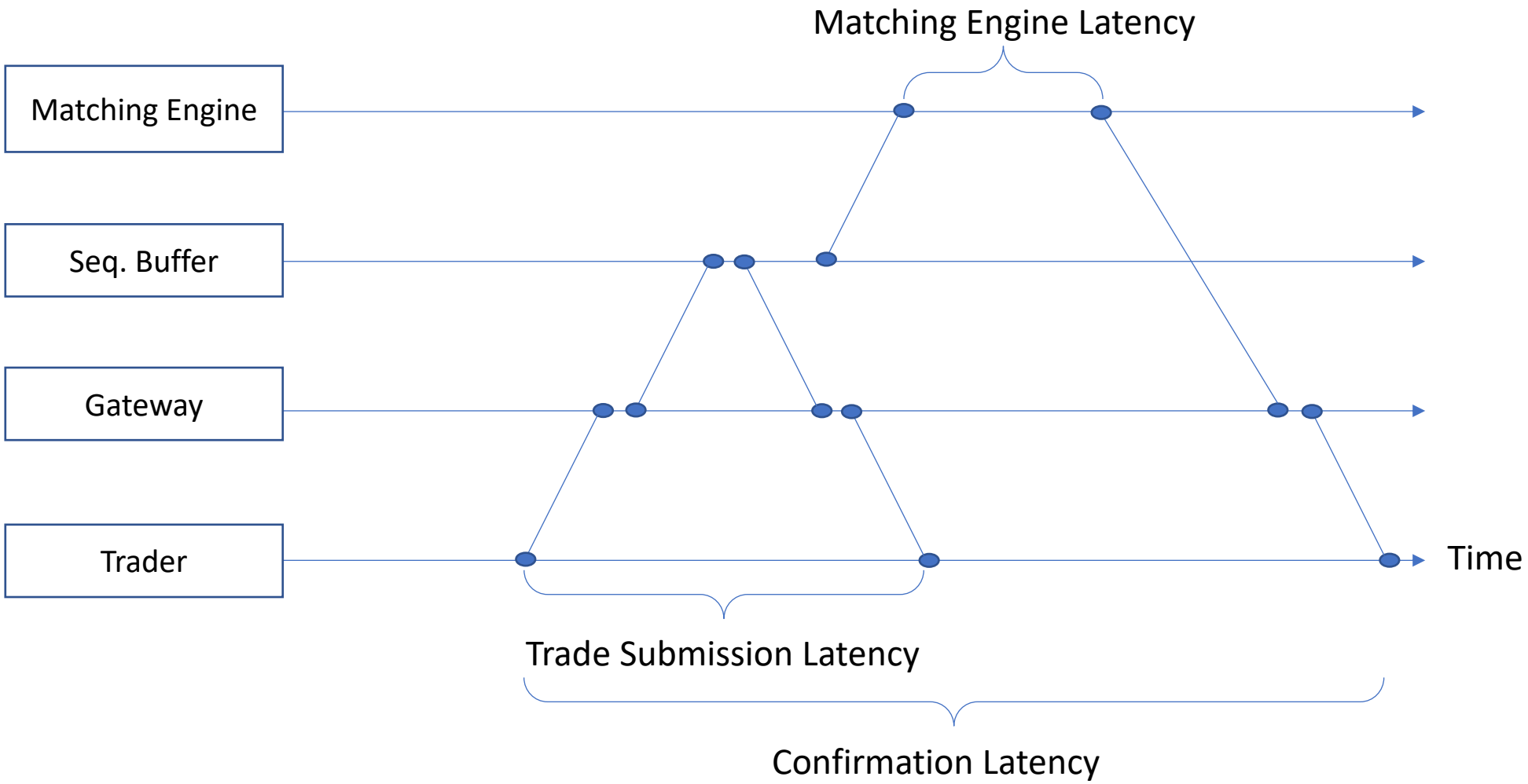
Market Participant Memory Store

Subscribed Limit Order Books		Order Map	Portfolio Vector	Completed Trades	Incomplete Orders
SYMBOL BUY/SELL	{(Order_ID, Order Priority)}	{(Order_ID, TradeMessage)}			

Example State Management Data Structure

Performance Measurements

Measurement Definitions



Experiments

Measurements:

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

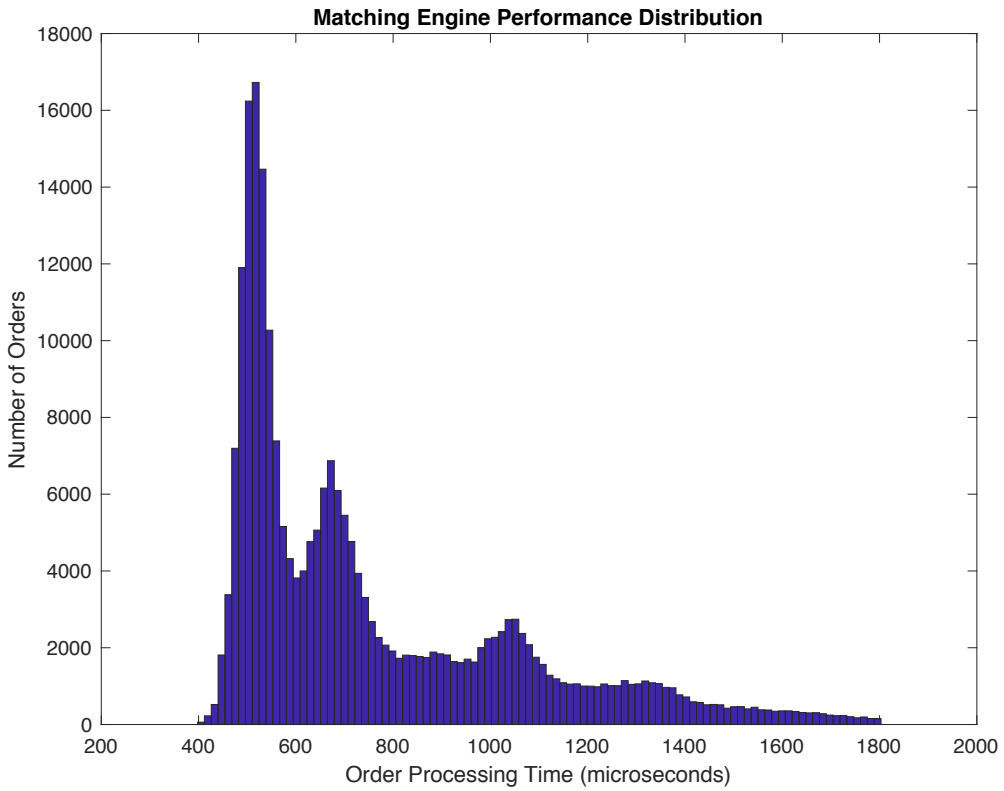
System Configurations:

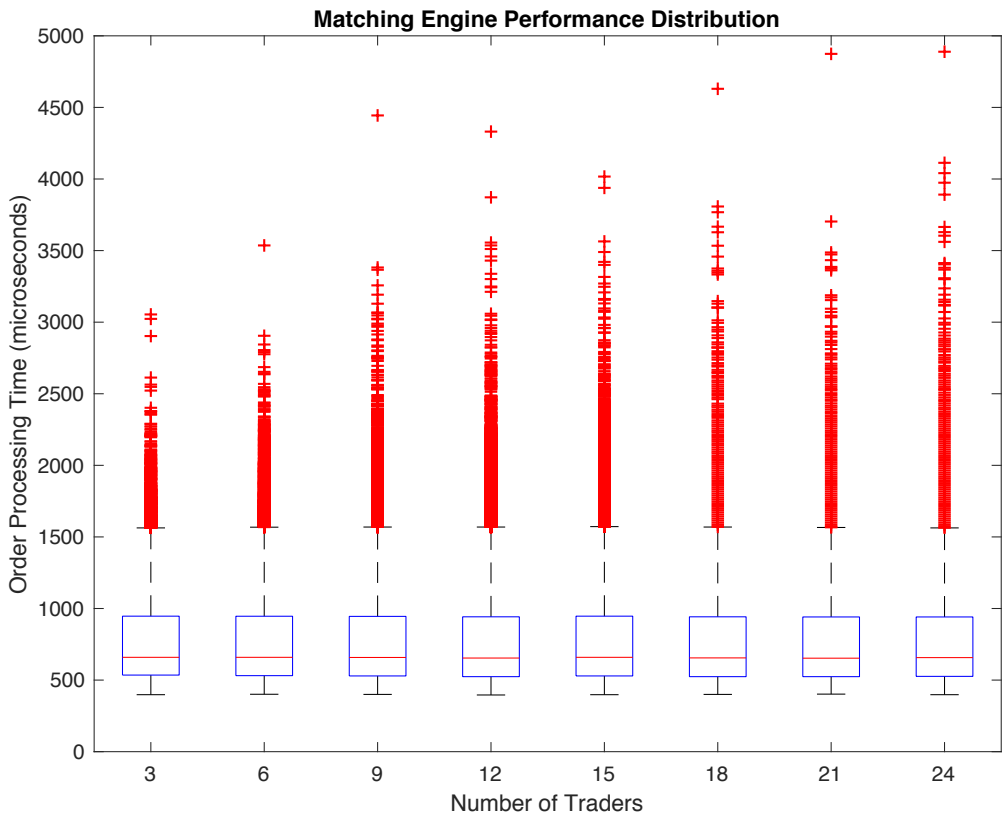
- {3 Traders, 1 Gateway}
- {6 Traders, 2 Gateways}

⋮

- {24 Traders, 8 Gateways}

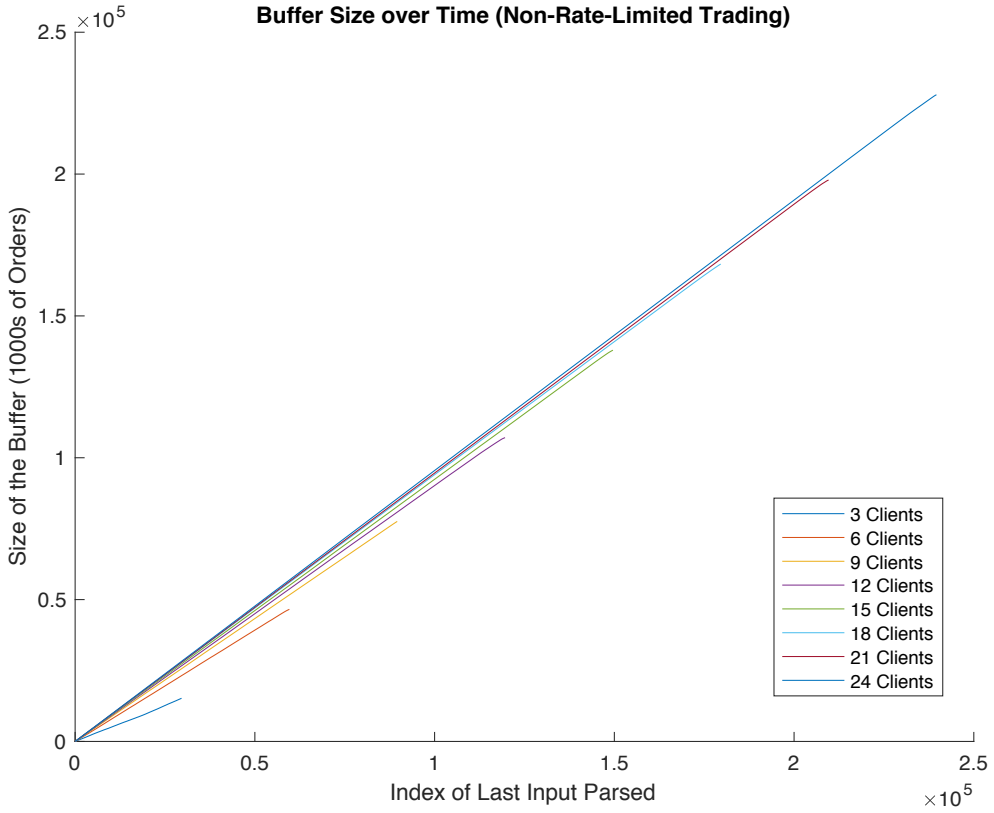
Matching Engine Latency Results

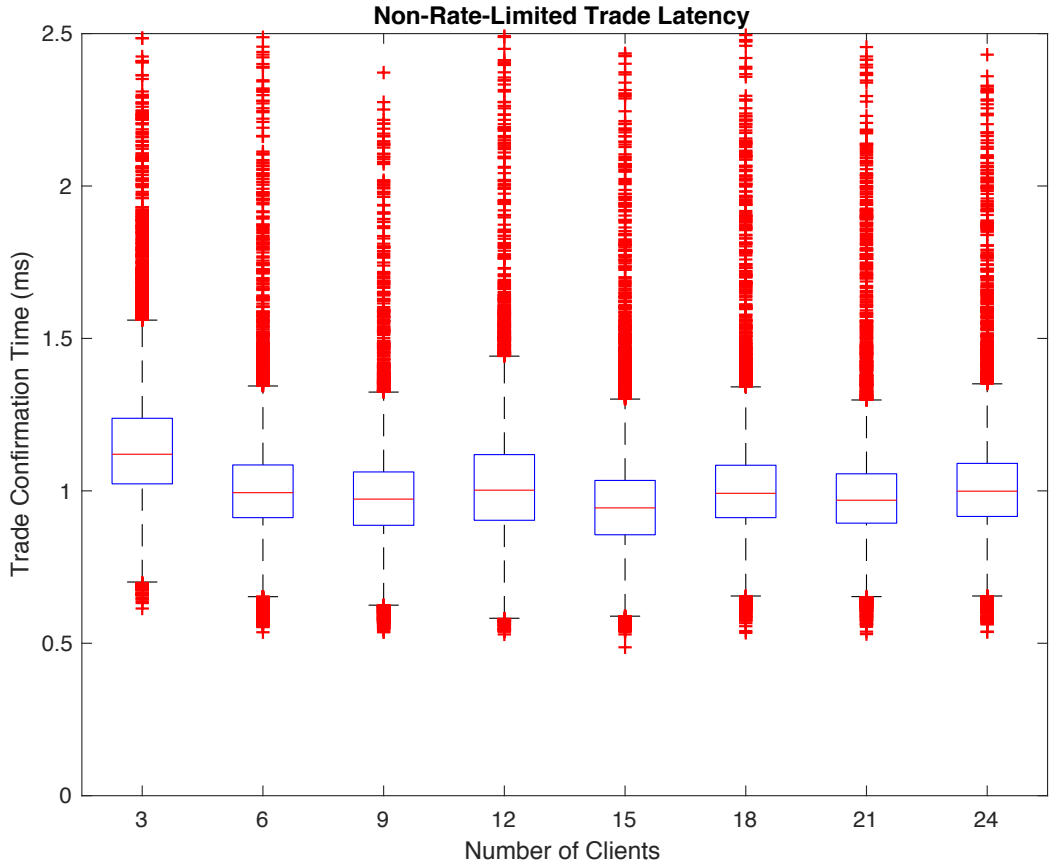




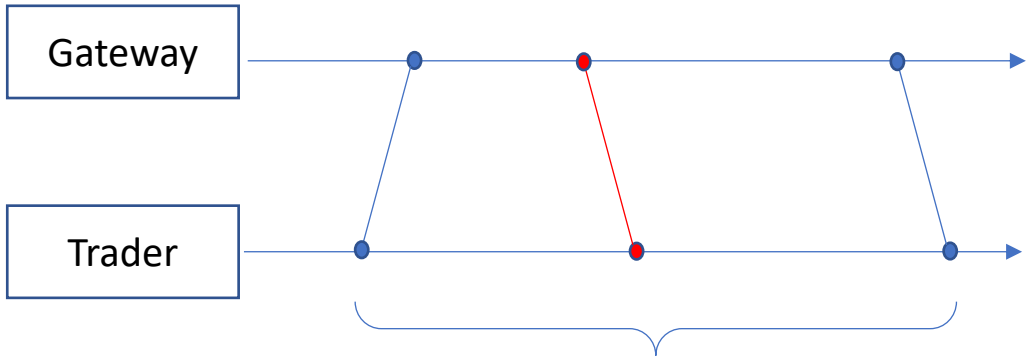
Median: **680 Microseconds** (First-Cut Implementation – Will be optimized)
99th Percentile: **1.8 Milliseconds**

Non-Rate-Limited Trader (Unstable)



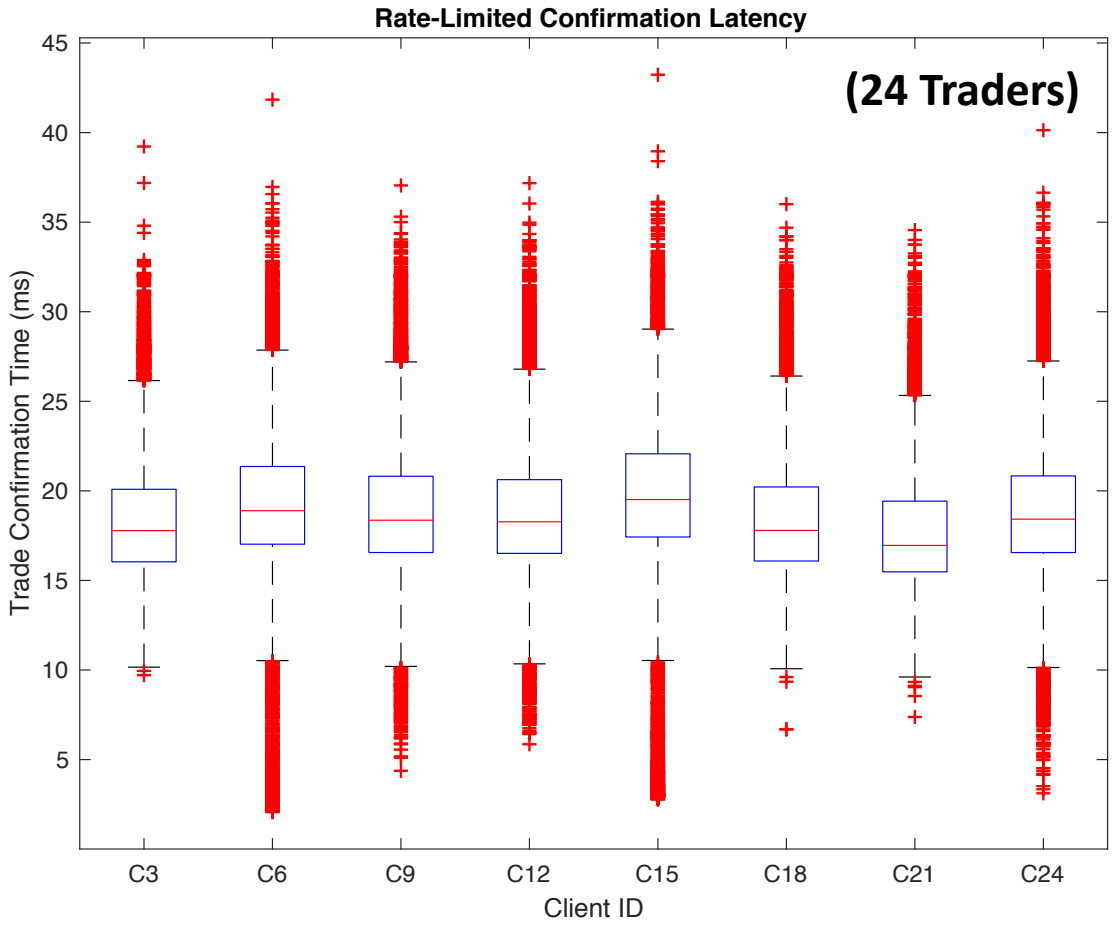


Rate-Limited Trader (Stable)

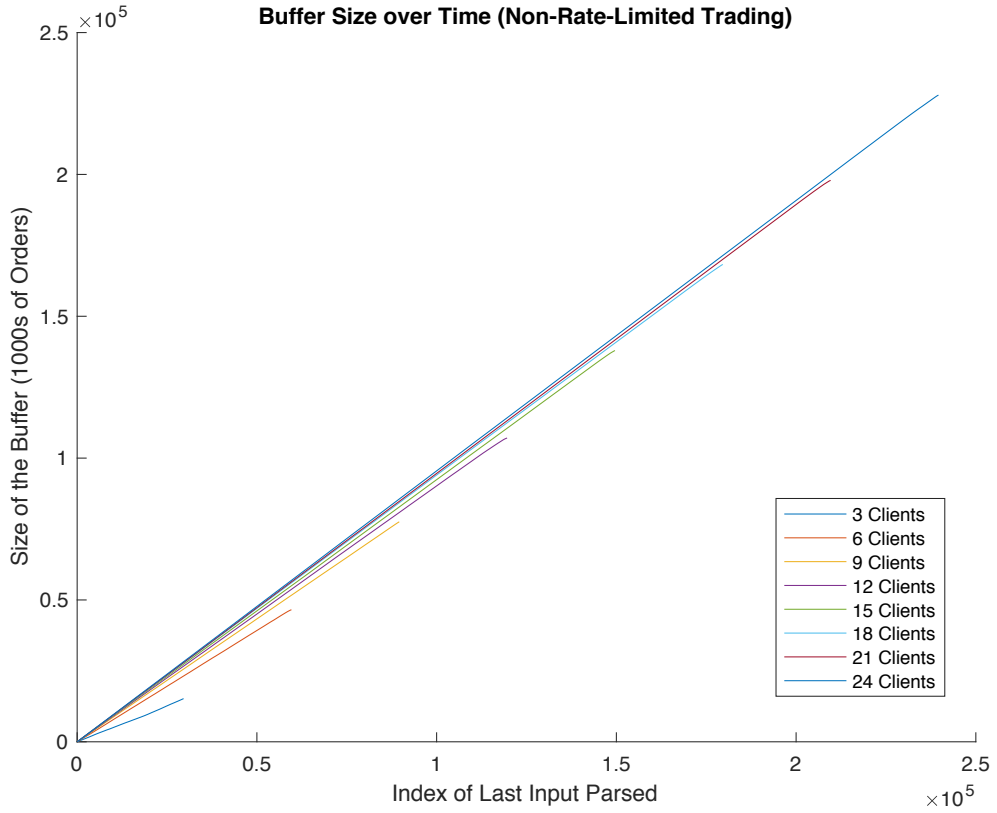


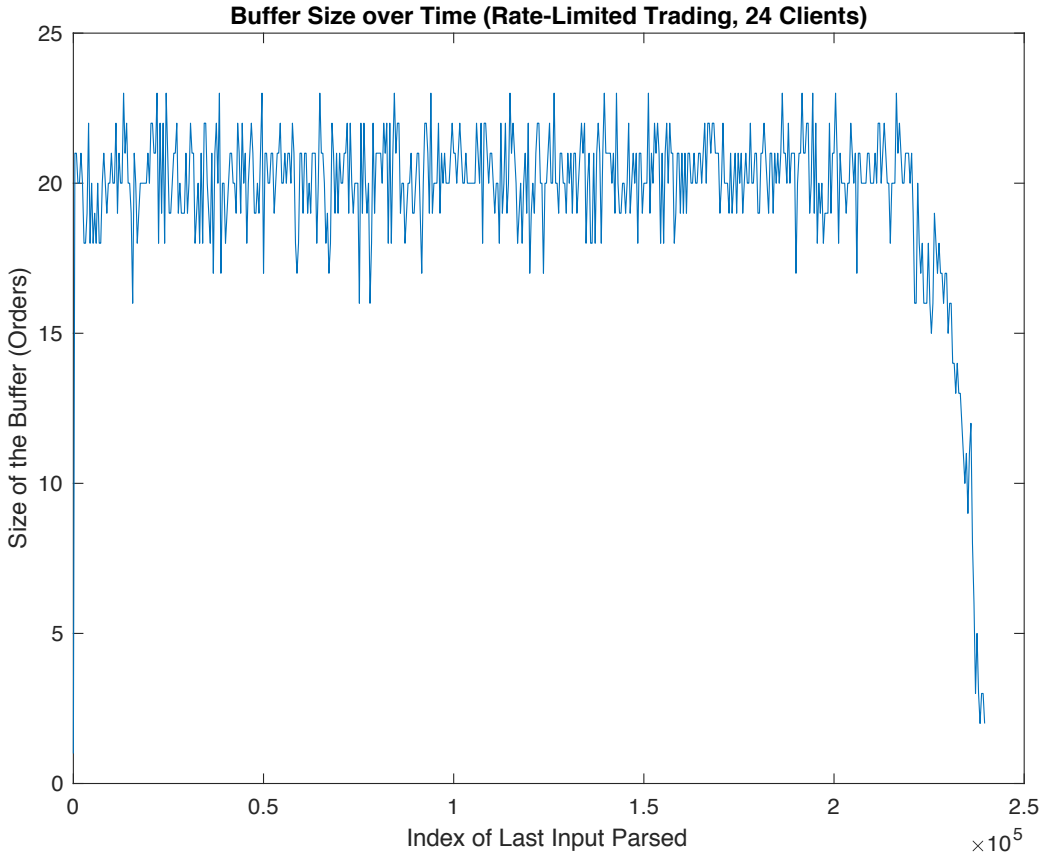
Trade Blocks until the Confirmation is Received

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





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