Stock Exchange

Facilitate matching buyers and sellers.
200 bill shares per day.

EDS:

- stocks only
- placing, cancelling
- normal trading hours
- trade 1 million shares in one day
- users should have sufficient funds

NFR:

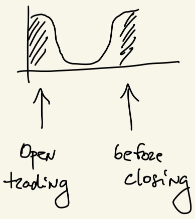
- Availability 99.99 %
- Fault tolerance. Fast recovery.
- Latency. ms-level. 99% latency.
- KYC (know your Client before new acc. opened)

BEE:

- 100 symbols
- 1 mill orders /day
- M-F 9:30 – 4:00 PM EST (6.5 h /day)

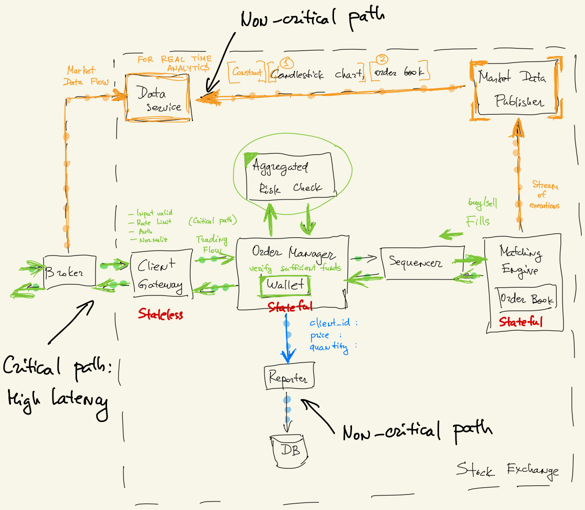
QPS : $\frac{1 \text{ mill}}{6.5 \times 3{,}600 \; (23.400)} \simeq 43.000$

Peak QPS: 5 times × 43 000 = 215.000



HLD:

- Broker – friendly user interface
- Institutional clients – trade large volumes via software.
- Limit order – buy/sell with a fixed price
- Market order – doesn't specify a price.
- Market data levels:
  - L1: best bid price, quantities
  - L2: more than L1 price levels
  - L3: queued quantity at each price level
- Candle stick chart Upper Shadow / Real body / Low Shadow
- Fix protocol 1991.



API :

ORDER

POST /v1/order

Req: Auth requires.

- Symbol
- side
- price
- order Type
- quantity

Resp:

Body:

- id:
- creation Time:
- filled Quantity:
- remain Quantity:
- Status:

Code:

- 200 : Success
- 40x : params error
- 500 : server error

EXECUTION

GET /v1/execution? symbol = {:symbol} & order Id = {:orderId} & start Time = {:startTime} & end Time = {:end Time}

Auth requires

Req. parameters:

- Symbol :
- order Id :
- Start Time:
- end Time:

Resp:

Body:

array with executions

- id:
- order Id:
- symbol:
- side:
- price:
- order Type:

Code: 200, 400, 500.

Historical prices (candle stick charts)

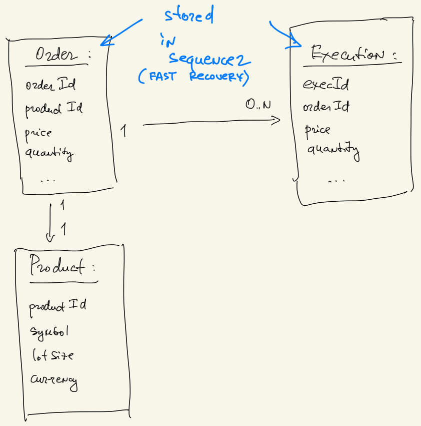
GET /v1/marketdata/candles? symbol
resolution
Start Time
end Time

Queries chart data

Data models:

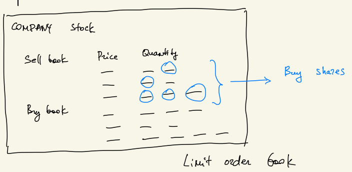
1. Product, order, execution

↳ type, traded symbol, currency, lot size. (highly cachable, Relational DB)
↳ inbound instruction for buy/sell
↳ outbound matched result (fill)



2. Order book (list of buy and sell orders) Data Structure

O(1) lookup time for matching
O(1) add/cancel/execute
Fast update & iterate through price levels



```
class PriceLevel {
    private Price limitPrice;
    private long totalVolume;        Doubly Linked List to
O(N) ✗  private List<Order> orders;      cancel / match O(1)
                                 ← Double Linked List < Order > orders ;
}
  class Book <Side> {
      private Side side;
      private Map<Price, PriceLevel> limitMap;
  }
  class OrderBook {
      private Book<Buy> buyBook;
      private Book<Sell> sellBook;
      ...
  }
```

3. Candlestick chart (processor to produce market data)

Data Structure :

```
class Candlestick {
    private long openPrice;
    private long closePrice;
    private long highPrice;
    private long lowPrice;
    private long volume;
    private long timeStamp;
    private long interval;
}

class CandlestickChart {
    private LinkedList <Candlestick> stick;
}
```

DDD:

99%

$$\left[\text{Latency} = \sum \text{execut. time Along Critical Path} \right]$$

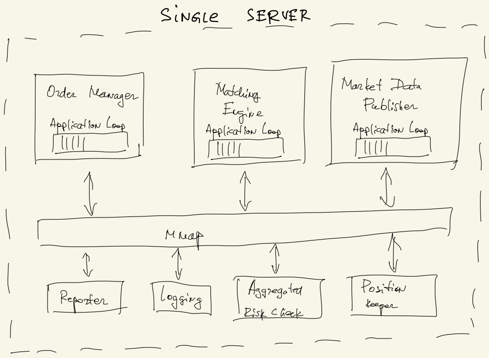
critical path:

gateway → order manager → sequencer → Matching engine
1 2 3 4

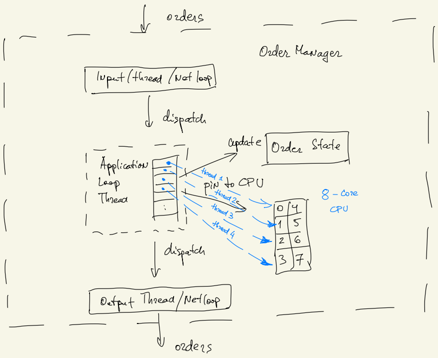
according to network ! [500 MS] host access time and using disk to store date will have tens+ miliseconds.

To get efficiently path use MMap.
Place all the elements on a single server:

To decrease latency-level delay:



Additionally we can pin each of components to run for a fix CPU core. Use multithreading.



MMap ; all the components on one host ;
latency == sub-microseconds ; event sourcing.
contains log of events immutable (source of truth)

High Availability:

4 nines 99.99 %

8.64 sec downtime per day.

- identify SPoF
- decision to failover to the backup instance should be fast.

Use heartbeat to detect potential problem in the primary.

Fault Toolerance:

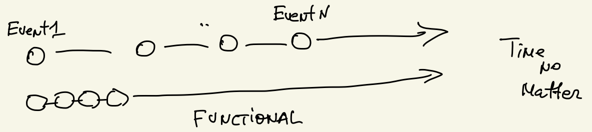
- replicating core data to multiple location data centers

1. if primary goes down → decide failover to the backup?
2. how do we choose a leader among backup instances?
3. $\overset{\text{Time}}{\text{RTO}}$? Time of recovery ?
4. What functionalities need to be recovered $\overset{\text{Point}}{\text{RPO}}$?

[Chaos engineering] ← use to gather all the necessary data to know all about put the system to failover from backup instances.

Latency & Functional Determinism

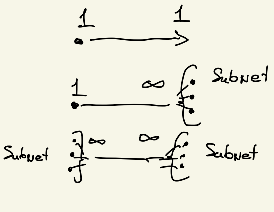
1.



2. In latency time is matter. It is key for the business.

Multicast:

1. Unicast
2. Broadcast
★ 3. Multicast
UDP - Not reliable



DDos issues:

- isolate network to prevent
- good caching won't use/hit DataBase
- harden URL's
  /domain.com/endpoint/ ← !
- network gateway with blacklist network address
- Rate Limiting to defend against DDoS.