Agenda

- Problem Statement
- Functional & Non-functional requirements
- API design
- Scale Estimation
- Architecture
- Sharding
- Bonus Material

High Level Design → bird's eye view of the software system

focus on the distributed architecture

facebook → laptop

2B+ users 10B+ msg/day ✗

10,000 computers → distributed across data centres → around the globe

- software crash
- power outage
- h/w fails
- n/w outage
- disk fsn.

communicate over network

- outage
- congested/slow

Staff Engg @ Google. → given a list of strings, sort them in asc order.

50 PB

SDE 2+ → top startup
→ MNC
} atleast 1 HLD round

HLD interviews → conversational.

1. Problem is vaguely defined
2. No one correct solution → focus on trade offs
3. Conversation could go in any direction.

~45 mins

- Problem Statement
- Functional & Non-functional requirements
- API design
- Scale Estimation
- Sharding
- Architecture

Problem Statement

✗ Stock Broker → Zerodha / ICICI direct

✗ Stock analysis → Trading View

✓ Stock Exchange → NSE/BSE NYSE / NASDAQ

Functional Requirements (MVP)

→ minimal set of features so that the product is usable

✓ → trade stocks : buy / sell / cancel / modify
- ↳ various order types
- ↳ matching order

✓ → Real time price data

✓ → Market Reports
- ↳ past orders
- ↳ price history
- ↳ Overall market stats

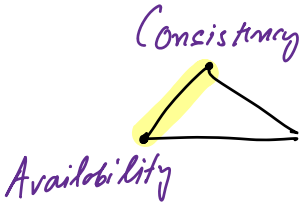
Non-functional Requirements

→ NSE down — Availability !!

→ Orders must be correctly created without errors — Consistency !!

→ multiple servers to distribute load → all servers in one place

CAP theorem

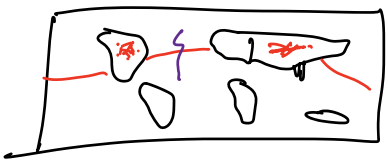


Pick 2

Some n/w issue due to which some machines are unable to comm with other machines

connected via v. high speed LAN (100GBps)

given of partition tolerance



→ Fairness → if UI's order was placed before U2's order & both have same price then UI's order should be fulfilled first.

→ V.V. low latency

→ Process thousands of trades / sec

API design → specify the endpoint name, input, output

↳ how does the external world use our system

brokers, traders — Stock exchange

Order processing

① POST /orders
↳ Create

Request
{ broker-order-id: —
stock: Reliance
price: —
quantity: — | type: timestamp: user-id }

Response
Success / failure

② PATCH /order/id
↳ update

{ type: —
price: —
quantity: — }

③ DELETE /order/id

↳ cancel order

④ GET /order/id

↳ order details

⑤ GET /orders → order book details

Market Data

① GET /stock/<symbol>

↳ current price

Reports

① GET /report/<date>

↳ total order volume & amount transacted on that day.

Scale Estimation

Back of the Envelope calculations

→ # users

total: 70 M

DAU: 14 M

India → 1.4 B people

5% of Indians have a demat account.

20% people who have an account trade on a given day.

→ # orders per day → Pareto principle 80-20 rule

20% people → place trades

80% → monitoring.

3 million people actively trading on a day

↳ each active trader places 5–10 orders

orders per day ≈ 15M – 30M

NSE true data ≈ 20M trades per day

↓ orders

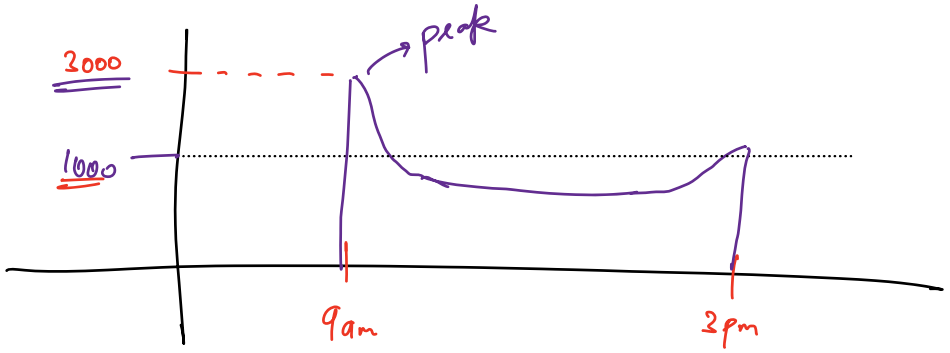
daily volume = 2.6 trillion

20M orders/day

→ # orders/second ?

$$\frac{20M}{60*60*\cancel{24}\ 6} = \frac{20\times10^6}{3600*6 \approx 20{,}000} = \frac{2\times10^7}{2\times10^4}$$

≈ 1000 orders/sec

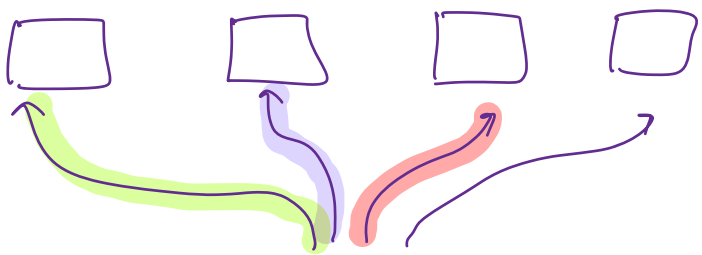


System should be able to handle 5000 orders/sec

Sharding → partition data across servers.

→ multiple computers

→ v.v. high throughput (5000 req/sec)



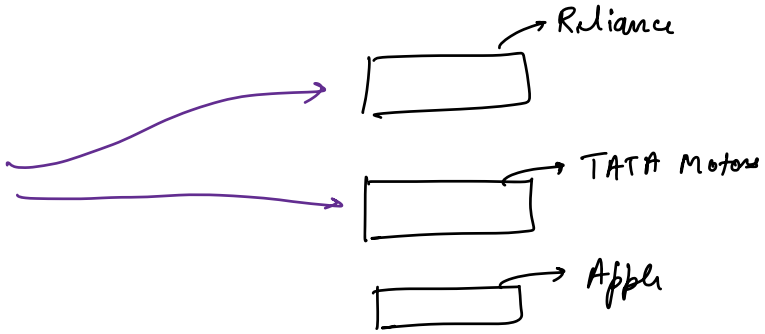
(2 phase Commit)

if we need to talk to multiple shards for 1 req

↳ v. high latency.

goal → everything (all data) that an order needs to be processed must be on a single shard

Shard by stock symbol.



6:31 → 6:40

Architecture

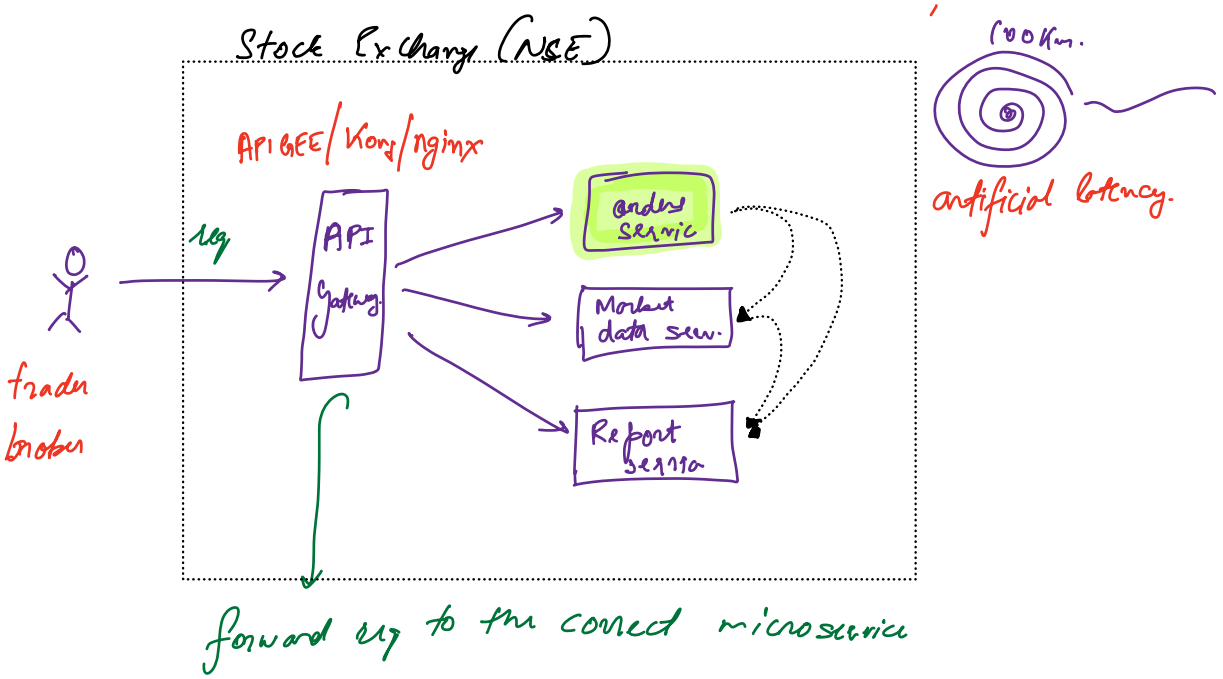
Process Orders, Reports, Live Market data

↓ v. low (< 1 ms) | ↓ high | ↓ decent (every 100 ms)

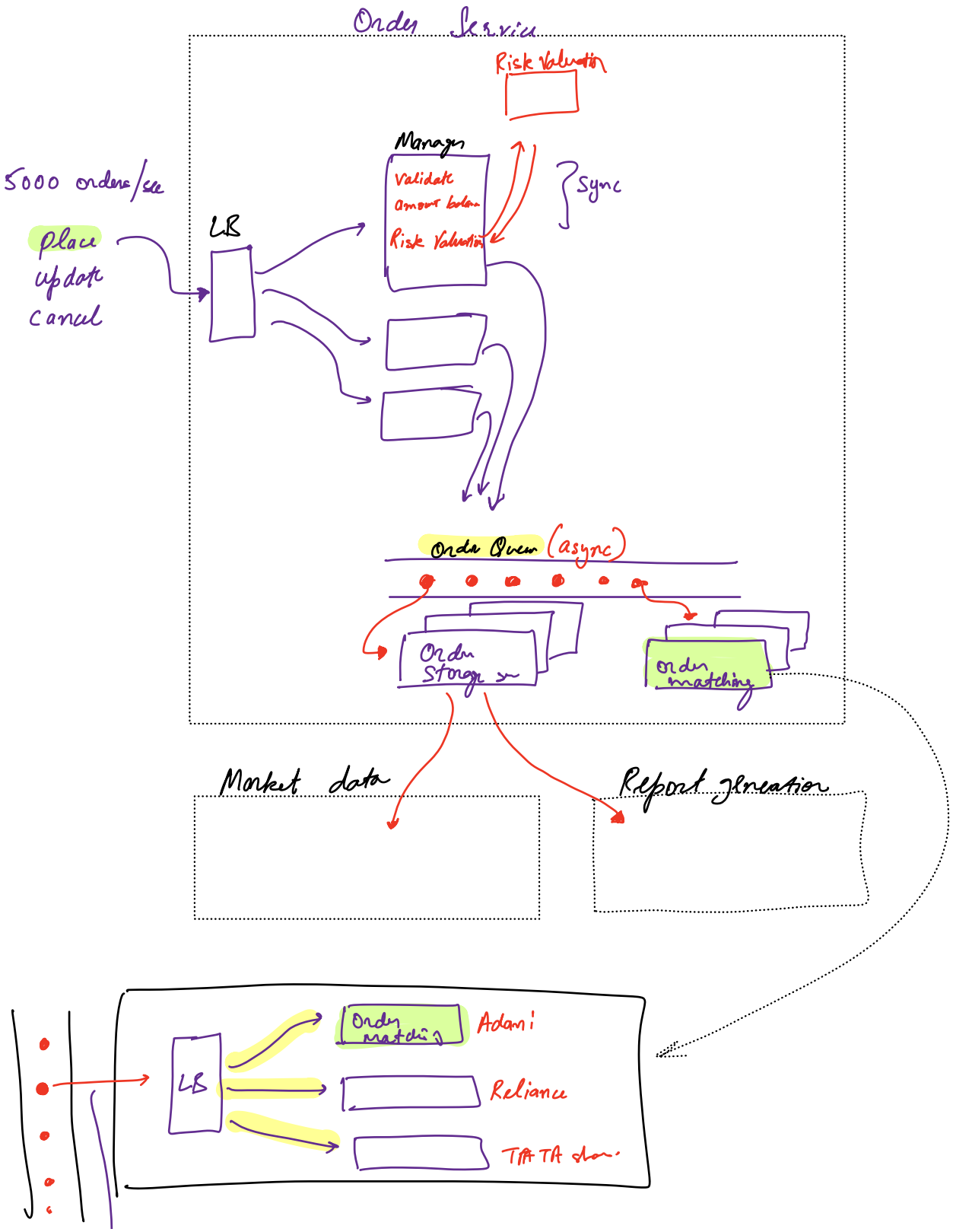
design multiple microservices → 1 for each port

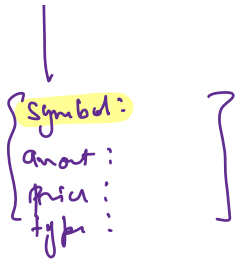
NYSE

100 Km roll of fiber cable



Order Service





designed for comm over n/w.

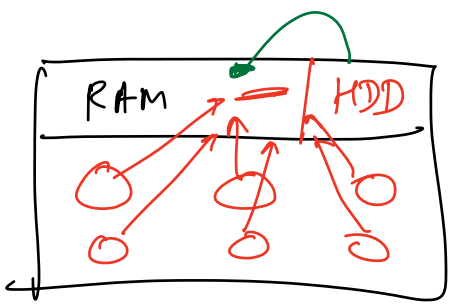
Kafka / RabbitMQ / SQS

optimized for distributed cloud architecture.

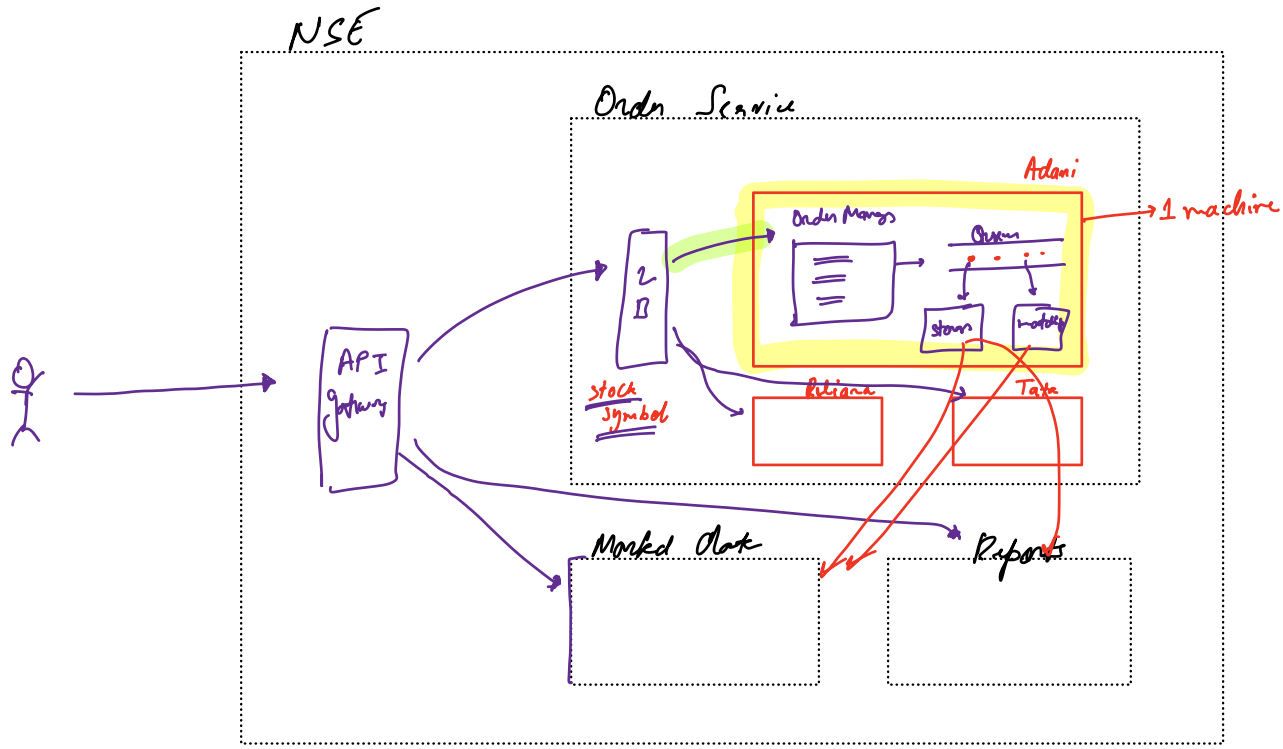
v. high latency from the perspective of Stock Exchange!!

any stock exchange will develop all these Sol^n s in house & deploy on same server

↳ services communicate via IPC (OS provided utilities)



Memory Mapped files



Robustness

Stock exchange 4 9's availability.

99.9999% time

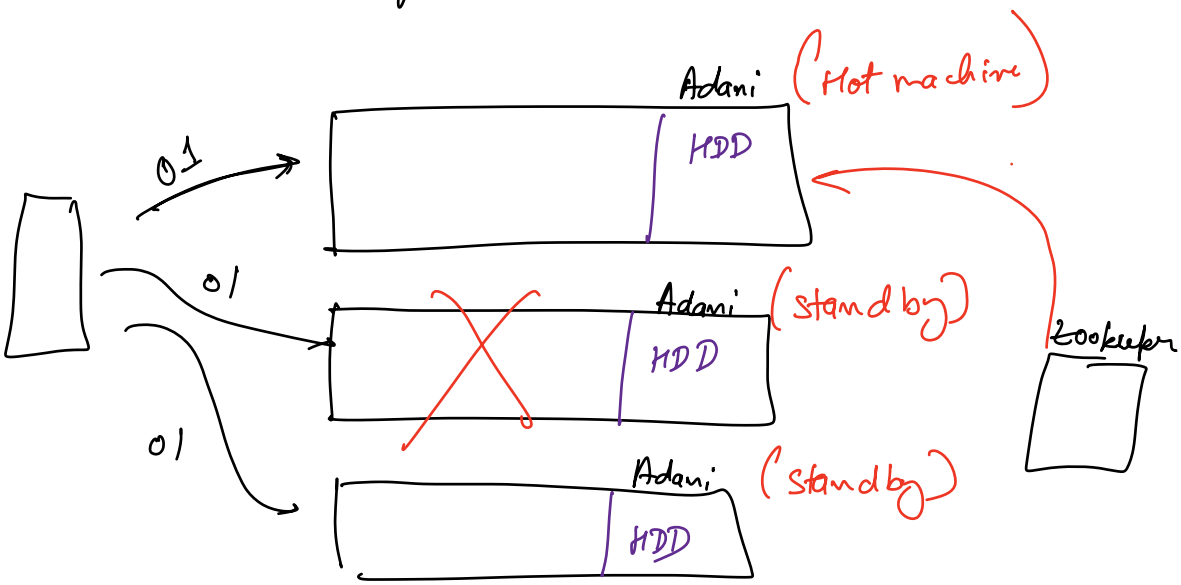
0.0001% time

0.0001% of 24 hours =

$10^{-6} * 3600 * 24$ = 0.08 seconds.

~6 second of downtime per day (NSE)

Hot machine & replicas



Heartbeats

at end of day → persist → in long storage.
→ Cassandra.

Matching Algo → hidden DSA problem

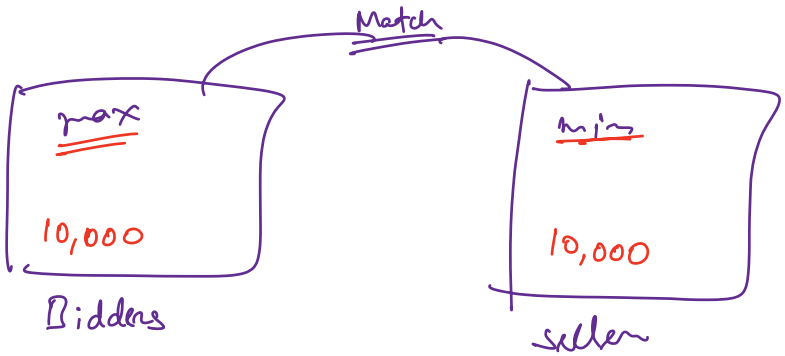
Buy Orders

Sneha	Venkat	Sumit	...
buy 100 Rs 10	buy 98 Rs 50	buy 95 Rs 200	

Sell orders

Vamsi	SK	Ravindar	Prabhat ✓
sell 120 Rs 20	sell 115 Rs 200	sell 110 Rs 50	95 Rs 10

match highest bidder with the lowest seller



linear scan O(n)

O(lgn)

LRU cache Hashmap + DLL O(1)