Reminders

- Feedback given on final project proposals, ask me about any questions you may have
  - Notes if the project goal needs to be increased, lessened, or looks good as is
- HW4 due tonight (reminder of your late days)
- Joy Liu guest lecture next week, attendance will be taken

Scaling a Brokerage

Early 2019

- Robinhood gaining traction/virality
- Covid 19 hits, huge traffic increase across the entire tech industry
- RH signed up ~3M users in just 4 months
- Daily traffic increased **10x** within just a month or so
- No other brokerages were handling scale comparable to RH at the time (perhaps still today)

Why is This Different?

- Instagram feed fails to load? Annoying, but no real harm
- A brokerage has **strict correctness requirements**
  - Money cannot appear or disappear due to a bug
  - Regulations require accurate accounting at all times
- Most web services optimize for **availability** — a brokerage must also optimize for **consistency**

How Most Services Scale

- **Add read replicas** — spread reads across copies of the database
- **Cache aggressively** — serve most requests without hitting the DB at all
- **Eventual consistency** — tolerate slightly stale data for higher throughput
- **Parallelize everything** — process requests independently, merge later
- These work great for Instagram, Twitter, Netflix, etc.
- Why don't they work for a brokerage?

The Serial Constraint

- Most services can process requests **in parallel** — order doesn't matter
  - Two users liking the same photo? Process both at once, no problem
- Brokerage operations on a single user **must be serial**
  - Check balance → lock funds → place order → confirm fill → update balance
  - If two orders race, you could spend money you don't have
- This means: **per-user operations are inherently sequential**

Real-Time + Serial = Hard to Scale

- Markets move in milliseconds — stale data means wrong prices, missed fills
- But you can't just process faster by adding more workers per user
  - Each user's operations form a **queue**, not a pool
- Traditional scaling (more replicas, eventual consistency) breaks correctness
- You need **strong locks per user** while still keeping latency low

The Scaling Paradox

- More users → more load on the database
- Can't use caching tricks (data must be real-time and consistent)
- Can't use eventual consistency (money must always add up)
- Can't parallelize per-user work (operations must be ordered)
- So what *can* you do?

System Overview

Brokerage Service

- Handles order state transitions & accounting
- Strongest guarantees are needed, e.g. full locks on balances when creating order
- Unlike many other larger companies, correctness is paramount
  - If your Instagram feed fails to load, it's not that bad. If money disappears in a brokerage due to a bug, it's *really bad*.
- Traditionally was a monolith, so still handles a ton of APIs (stocks, options, admin, balances, etc)

Problem: Market Open

- This is when the database of the brokerage service experiences peak load
- We cannot send orders to our venues until the market opens
- Overnight, tons of orders are queued up to be submitted
- At market open, orders are all sent to venue and updates need to be made, e.g. filled, cancelled, pending, etc

Problem: Market Open

- The brokerage service DB begins to experience 100% load at market open
- What do you do when you can't throw money at the problem anymore?

Sharding

- Idea: let's shard the brokerage service
- Multiple ways to shard, important to consider your *shard key*. We'll use user sharding
- Split users across multiple databases, each user lives completely within a single database
- Now we'll have brokerage-service-1, brokerage-service-2, etc.

Sharding Problems

- How do we decide where a user goes?
- Traffic needs to be routed accordingly, internally and externally
- Kafka streams need to be routed accordingly
- Internal processes that read from replicas of the brokerage service need to be updated
- Jobs that run on the brokerage service need to be OK to be replicated
- Users need to be migrated off of current shard (scary!)

User Sharding

- We can rely on the authentication service to help us route users properly
- On signup, the user is assigned to a shard
- Then, when authenticated traffic arrives, nginx will tag it with the shard number
- Add a new nginx in front of all brokerage services to respect this shard number
- We can take a similar approach for Kafka streams

Battle Test: GameStop

GameStop short squeeze

22 languages

Article Talk Read View source View history Tools

From Wikipedia, the free encyclopedia

In January 2021, a short squeeze of the stock of the American video game retailer GameStop and other securities took place, causing major financial consequences for certain hedge funds and large losses for short sellers. Approximately 140 percent of GameStop's public float had been sold short, and the rush to buy shares to cover those positions as the price rose caused it to rise even further. The short squeeze was initially and primarily triggered by users of the subreddit r/wallstreetbets, an Internet forum on the social news website Reddit, although a number of hedge funds also participated. At its height, on January 28, the short squeeze caused the retailer's stock price to reach a pre-market value of over US$500 per share ($125 split-adjusted), nearly 30 times the $17.25 valuation at the beginning of the month. The price of many other heavily shorted securities and cryptocurrencies also increased.

On January 28, some brokerages, particularly app-based brokerage services such as Robinhood, halted the buying of GameStop and other securities, citing the next day their inability to post sufficient collateral at clearing houses to execute their clients' orders. This decision attracted criticism and accusations of market manipulation from prominent politicians and businesspeople from across the political spectrum. Dozens of class action lawsuits have been filed against Robinhood in U.S. courts, and the U.S. House Committee on Financial Services held a congressional hearing on the incident.

GameStop

- One of the highest traffic market opens
- In a purely technical sense, our solution worked!
- Funnily, account creation was the service that fell over instead