Understanding Systems

CSE542

About Us

- Nisarg Shah
  - Database Software Engineer @ Oracle, AU’2021
  - ASU Grad 2023
  - Previous Intern at Intel Cloud @ File Storage

- Manav Darji
  - Blockchain Engineer @ Polygon Labs, AU’2021
  - Ethereum Protocol Fellow

Agenda

- Intro to systems and types
  - Examples and real-world use cases
  - Problems faced while designing such systems
- Consistency, Availability and Replication
- Fundamentals of decentralised systems and blockchains
- Faults in distributed systems
- Consensus Protocols (BFT, Raft)
- Deep dive into Ethereum
- Open problems in world of blockchain

Centralized vs Distributed vs Decentralized

- Centralized: Central server, data in control
- Distributed: No central server; enables shared ownership of data; control may or may not be with end users
- Decentralized: No central server; multiple central owners; agreement based; control with end users

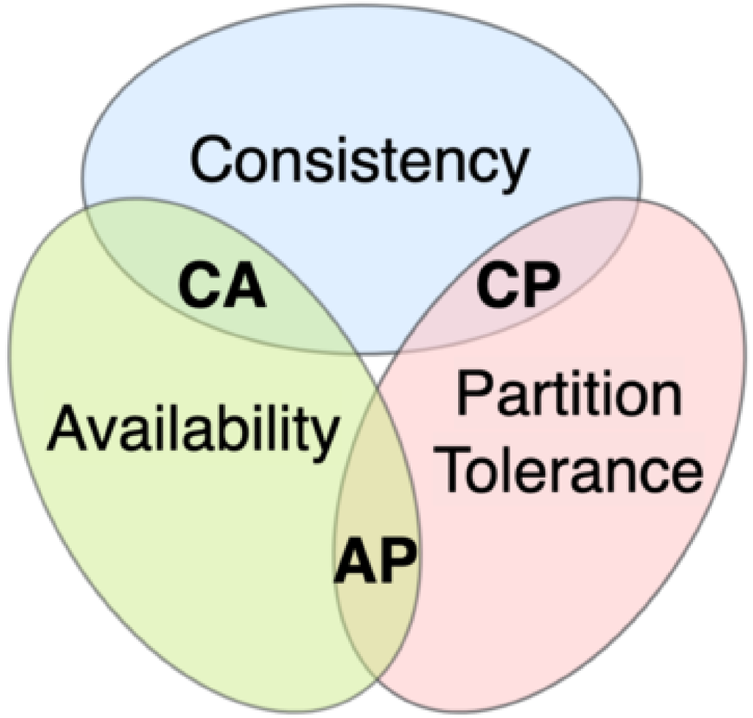
Distributed Systems

- Robustness
- Scalable
- Highly available
- Could be consistent
- Sophisticated

Examples?

Use cases

CAP Theorem: Every system has it’s own purpose



Designing Systems

- Network failures
- Network partitions
- OS failures
- Data corruptions
- Data Availability
- Server lag

Examples

- Email Servers
- Systems from Facebook, Google, Oracle, etc.
- Databases

Failures

- Fail Stop - Node stops working
  - Hardware Failure
  - Memory issues
  - Power outage
  - and many more....
- Recover - Store state somewhere

Failures

- Byzantine Failure - Node is working
  - Corrupted disk
  - Security attacks
  - Malfunctioning of code
  - Conflicting information b/w nodes
  - Message change b/w nodes
- Recover - Very Hard

What can be done?

- Some restrictions
  - Timeouts in message passing
- Don’t trust unreliable sources
- Assertions at regular interval
- Eg: Database block checks

Build solution which solves MOST of problems, not ALL!

Two Generals Problem

- Two Armies A and B
- Attack on C
- One mediator
- A and B can only win if they attach together
- Can they win?

Two Generals Problem

- Assume no byzantine faults
- For lossy network

Byzantine Generals Problem

- The Byzantine Generals Problem, LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE @ ACM 1982

What this paper says?

- How to build byzantine fault tolerant system?
- How many node faults can survive?

Byzantine Generals Problem

- Problem: Get consensus between nodes

Thank you!

Time in Distributed Systems

- Very Important and Toughest Part
- Physical Clocks - Quartz clock, wrist watches
- Examples
  - Fetch updated data
  - Programming environments
  - Transactions

Inspired by Martin Kleppmann Slides: Author of Designing Data Intensive Applications

Physical Clocks

- Time in UTC
- Includes leap seconds to keep UTC roughly in sync
- Computer Timestamps
  - Unix time - no of seconds since 1st Jan 1970
  - ISO 8601 - 2021-11-09T09:50:17+00:00
  - But what about leap seconds?

Leap Seconds

- Ignored in Software
- 30th June 2012 outage

Leap Seconds

- Spread the leap second throughout the day

Clock Synchronization

- NTP servers in OS
  - fetches the current time more accurately

Do we need Physical Clocks?

Clocks

- Physical Clock
  - Count exact seconds
- Logical Clock
  - Count no. of events
  - Can find causal dependencies
  - $(e1 \rightarrow e2) \Rightarrow (T(e1) < T(e2))$

Logical Clocks

- Lamport Clocks
- Vector Clocks

Lamport Clocks

- Each server has it’s own counter
- Attach the current “t” to the message
- Recipients increment the counter to the “t” received in message

Issues

- Duplicate timestamps
  - Attach the server/node number in the timestamp

Issues

- Still can’t find if a -> b

Vector Clocks

- Kind of Global Timestamp
- Stores every piece of event occurence count

Vector Clocks

- N nodes <n1, n2, ...>
- Vector timestamp of event a : V(A) = <t1, t2, ... tn>
- Each node has a vector timestamp
- On an event at Node n2, vector element of 2nd index will be changed
- Timestamp is attached in message

Replication

- Copy of data on multiple servers
- High availablility
- Storage, Databases

Replication

- Copy of data on multiple servers
- High availability
- Storage, Databases
- Centralized system
    - Replicate storage to other disk
- Distributed system
    - Each node is independent

What happens if no ACK?

Facebook Likes Issue

Fixtures?

- Use the request id to check updates
- Retry if not performed commit

Problem of adding and removing element

- First Scenario
  - Client adds value to both servers
  - Removes value, but due to network issue request to S2 lost
- Second Scenario
  - Client tries to add value to both servers, but only able to add in S2.

What can be done?

- Add Timestamp in every request
- Add a boolean while removing from database

After that Reconcile

- The timestamp and false flag will be replicated
- Replication will be done on t1 and t2 of two servers
  - if t1 < t2, it means t2 is latest

What if concurrent users write?

- Data Loss can occur
- Last writer wins
- Store all values
- Not that efficient

Quorum