Day 17: CS 1380
Today

Top Hat Question

Replication & Ordering

CAP Theorem

Practical Replication Service
Linearizable = total + FIFO + real time

Sequential = total + FIFO

<table>
<thead>
<tr>
<th>Active</th>
<th>Passive</th>
<th>Causal +</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>Linearizable</td>
<td>Lazy</td>
</tr>
</tbody>
</table>

- FE assign IDs by their ID
- MM order goes by these ID
- the breaking rules to ensure total ordering
**CAP Theorem**

Consistency = reads will return the value of last/recent write according to some ordering constraint

Availability = reads/writes will be processed by the systems no hanging

Partition = during a network partition some nodes in the system can make forward progress in protocol

You can only **HAVE TWO** of the three

**CA** - if there partitions (failures) the protocol hangs

**CP** - during a partition, subset of the notes will be unavailable because they wait to provide consistency

**AP** - during partition, all notes are available & data may get lost
CP = maybe neither FE can read/write until the partition is over

AP = but FEs can keep read/write to their RMS
<table>
<thead>
<tr>
<th></th>
<th>CA</th>
<th>CP</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raft</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lazy Replication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chandy-Lamport</td>
<td>√</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Raft**
- No active leader
- No availability for some clients
- No active leader means no client availability

**Chandy-Lamport**
- Protocol requires Masters if during a partition masters will not be delivered & the protocol will fail to terminate or make forward progress

All writes respond instantaneously:
- No coordination required
Practical Consensus (Practical Passive Replication)

( ZooKeeper )

MapReduce (Primary)
* plugging map or reduce on servers
* detecting server failure & restarting map/reduce tasks
* using progress reports

Controller
* Replicas of the controller
* you need leader election code

Cluster of servers

haift = leader election

Consensus / Coordination Services = primitives to automate leader election & techniques to maintain persistent state (log or file)
Most of ALL calls are reads:
90% are reads & 10% are writes

Under the hood:
Running raft/paxos or raft-like protocols
Read-specific optimizations

1. Caching
2. Have followers respond to read requests
3. Make all data in memory
4. Leader/followers can respond to read requests

Techniques:

- Staleness: When note gets update, it sends notification to cache.
  - Can get stale/bolt information!!!
  - Provide a "sync" command to force followers to synchronize with leader

Memory --> Disk

Computer

Program

Memory

Write

(only leader processes writes)
MapReduce

Google File System

Big Table

Coordination Service
all replication schemes provide "consistency" guarantee with a shard!

These no guarantees for changes across these shards.
This class

1. advanced ordering
2. CAP Theorem
3. Zookeeper (motivation & how it scales to reads)