Introduction to Resiliency

Introduction

1. Cloud computing uses ideas from distributed computing from several decades ago that were designed for dozens of computers and applies them at scale

Resilience

1. Resilience and replication
    - Key to resilience is having redundant replicas
    - Given failures, how do we know which replicas are up to date?
    - Replica management is fundamental to cloud computing
        - Replicated resource managers
            - Azure Service Fabric, Google's Borg, Apache's Mesos
        - Replicated storage servers
            - Google Bigtable, Oracle NoSQL storage server
2. Types of Failures
    - Byzantine: Upon failure, start sending spurious information to healthy nodes
        - Term invented by Leslie Lamport (some generals are compromised)
    - Fail stop: Upon failure just shut up
3. How much redundancy is needed?
    - Byzantine: To tolerate t failures, we need 2t+1 replicas
    - Fail stop: To tolerate t failures, we need t+1 replicas
4. State maintenance with replicas
    - We want update progress in the presence of failures
        - Allow updates to happen without waiting for all copies to ack
    - Quorum consensus protocols
        - Read: "r" copies
        - Write: "w" copies
        - If N is total number of servers, for correctness
            - Qr+Qw > N ensures read quorum and write quorum overlap
            - Qw+Qw > N ensures that there is no concurrent update to the same data item

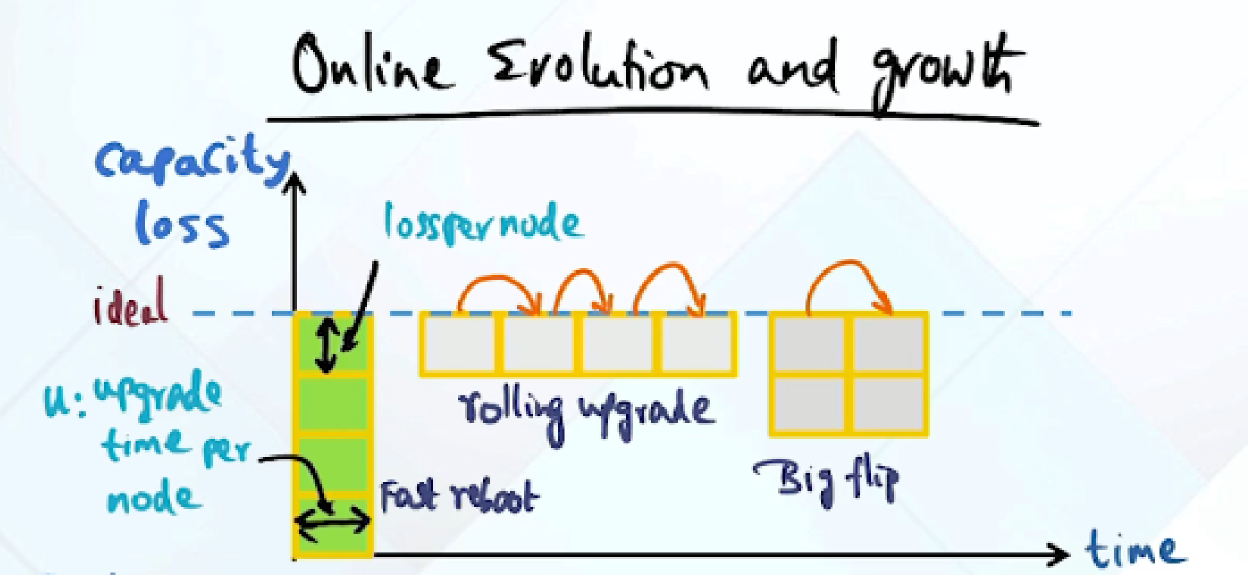
Paxos Algorithm

1. Paxos
    - An elegant formulation of quorum consensus system
        - Roles played by the members of the system
            - Proposer, coordinator, acceptor, learner
    - Proposer: Think of this as an application command to perform an action that is "durable" e.g., write(x, "v")
    - Coordinator (Assume there is only one, works even if there are many)
        - Phase 1: Send "prepare to accept"
            - Receive "ready to accept" ACKS or "not accept" NACKS from acceptors
        - Phase 2: Send "accept" with the actual command
            - Command is performed by the learners who execute the command
        - If insufficient acks, then go back to phase 1
2. Paxos in use
    - First used in DEC SRC Petal storage system
    - With the advent of cloud computing, it experienced wide use
        - Google Borg
        - Azure Service Fabric
        - Facebook Cassandra
        - Amazon Web Services

    - Apache Zookeeper

Upgrades

1. Availability basics
    - Availability: Fraction of time system is up
    - Availability may be subject to contractual obligations
        - e.g., an ISP may contractually promise 99.9% uptime in the SLA for apps
2. Types of online evolution
    - Fast: Take down all servers, upgrade, restart
        - Might be acceptable if it can be scheduled when servers aren't in use
    - Rolling: Take down subsets of all servers at a time
        - Reduction in capacity is less than fast ugprade, but takes more time
    - Big flip: Half the nodes at once, then the other half
    - Loss is same for all strategies: Loss per node * number of nodes * time



Online Evolution and Growth

3. Availability vs Upgrade Conundrum
    - Software ugprades
        - Rolling upgrade usually recommended to avoid service downtimes
            - Applicable only if changes maintain backward compatibility
    - Upgrades in general
        - Affects availability guarantees
        - Needs to be carefully orchestrated
            - Upgrade agility for competitive edge
            - Respecting SLAs for business apps

Elasticity

1. Elasticity: Provisiong and de-provisioning system resources in an automatic manner
    - Dimensions
        - Speed
            - Time to switch from under-provisioned to optimal
            - Time to switch from over-provisioned to optimal

      - Precision
        - Deviation of new allocation from actual demand

2. Implementing Elasticity
   - Proactive cyclic scaling
     - Scaling at fixed intervals (daily, weekly, monthyl, quarterly)
   - Proactive event-based scaling
     - Scaling due to expected surges (e.g., product launch, marketing campaigns)
   - Auto-scaling on demand
     - Monitor key metrics (server utilization, I/O bandwidth, network bandwidth) and trigger scaling
3. Application Tuning for Elasticity
   - Identify app components or layers that can benefit from elastic scaling
   - Design chosen app components for elastic scaling
   - Evaluate the impact of overall architecture due to elasticity

Conclusion

1. Cloud computing applied three decades of distributed computing research to systems of scale
   - Unimaginable at the time that these ideas were proposed
     - Paxos is the bread and butter for resilience and fault tolerance