CMPSCI 677 Operating Systems **Spring 2015**

Lecture 17: March 31

Lecturer: Prashant Shenoy *Scribe: Nick Braga*

17.1 Recap

- Consistency protocols
  - Consistency semantics / guarantees: *Strict consistency*, several weaker forms of consistency, and *eventual consistency* (the system does not try to give any guarantees to the user about when data propogates to all replicas)

17.2 Epidemic protocols

- Based on the theory of how epidemics, diseases spread (through pair-wise contact)
- An epidemic protocol implements *eventual consistency* through pair-wise 'contact' of weakly-connected (i.e., mobile) peers
- *Infective stores* and *Susceptible stores*
- Types:
  1. *Anti-entropy*: (1) Push only (2) Pull only (3) Push/Pull wrt servers P and Q; initially, push-based is superior (only a few machines have seen the update); later, the chances of a server Q having seen the update already grows higher so non-updated servers should instead pull; since this is a randomized approach, we can never guarantee a complete update, but hybrid approach helps
  2. *Rumor mongering ("gossiping")*: Push-based; P pushes update to Q; if Q had already received it, then P stops spreading with probability $1/k$, where k is the back-off probability and can be arbitrarily set; non-zero (but small) chance of never seeing a given update
- Impossible to distinguish between a deleted copy and no copy (ever existing); no state information! *Death certificates* treat deletes as updates, spreading a certificate as an update of the delete operation and later clean-up
- Problematic Case: Two users modify data concurrently (write-write conflict); user intervention required

17.3 Implementation Issues

The following are general solutions for implementing consistency protocols:

- Primary-based protocols: Master-slave scenario

  - Local-Write protocol: Changes can be made safely at any copy, then local replica sends to all other peers; Primary is in essence "continually changing"; Timing is dependant on consistency guarantees made
  - Remote-Write protocol: Local changes should always be pushed to master, which will then propogate
- Replicated write protocols: No primary can be assumed
  - Gifford's Quorum-based protocol: Voting-based approach to determine the latest version of a file
  - Ensure both *safety* and *liveness* properties with properly set quora
    * Read-based quorum: The number of servers (possibly from a larger set) that you are to ask and must agree (requirement: Read Q. + Write Q. >Num. of Servers)
    * Write-based quorum: The number of servers (possibly from a larger set) that you must send a write to (and process) before task is finished (requirement: more than half the number of servers – if less, can lead to write-write conflicts!)
  - In general, the concept of *agreement* will be further encountered with a number of key fault tolerance techniques
- Replica Management:
  - *How many replicas? How to place them?* How can we systematically make such decisions?
  - Geographic distribution of replicas might be necessary to prevent certain types of failures (network/power loss, natural disaster) (*Degree of replication*)
- Real-world problems require a choice of consistency semantics that factor in their respective trade-offs (costs, benefits)

17.4 Fault Tolerance

- Distinct from consistency (which deals with synchronization and associated issues), fault tolerance is concerned with using replication when faced with machine failures
- In distributed systems, failures can be automatically limited to partial failures; this is particularly important in a distributed setting (as opposed to single-machine *"all or nothing"*) since the probability of a node failing increases with the number of nodes in a system
- This requires that any node should be able to take over the functionality of its peers
- Pervasive computing motivates fault tolerance, where most users are not computer scientists; these users require *dependable* systems
  - Availability: Maximizing system use up to *five nines* (99.999 percent available) in practice
  - Reliability: Related to Availability; The ability of a system to move past a failure
  - Safety: Also related to Availability and Reliability; a latent failure should not cause a long-term failure
  - Maintability: Property of a system that is easiest to repair in the face of failure
- While individual hardware/software components are not reliable on their own, fault tolerance seeks to provide resilience to individual failures for the system; as discussed by example in class, redundancy of computations allows us to *agree* on the correct (or, majority's) result
- Failure to replicate all failure-prone components will not provide the desired degree of fault tolerance