CMPSCI 677 Operating Systems **Spring 2013**

Lecture 17: March 27

Lecturer: Prashant Shenoy *Scribe: Srikar Damaraju*

17.1 Epidemic Protocols

Epidemic protocols implement eventual consistency. These are analogous to disease spreading, where disease spreads from one person to another person through contact.

17.1.1 Spreading of an Epidemic

17.1.2 Anti-Entropy

A server picks another server randomly and exchange all the updates with that server. Updates are made either by Push-Only or Pull-Only, or both approaches. In Push-Only, server pushes or sends the data to the receiver server but does not receive any data, whereas in Pull-Only receiver server extracts data from sender server, but does not send any.

A pure push-only approach will not spread updates quickly, because it is a randomized protocol, and there can be instances where few machines will never see the update. This can be solved by used by implementing both Pull-only or initial push with Pull-only approaches.

17.1.3 Rumor Mongering (gossip)

It is a randomized protocol. A server, as soon as it sees an update, randomly picks a machine and pushes the update. If that receiving machine has already seen this update, server will stop spreading the updates with a probability of 1/k. Rumor mongering protocol will not guarantee that all replicas will receive updates. Chances of a replica not receiving the update is exponential, which is s= e $^{(k+1)(1-s)}$, where "s" is the probability of not receiving the update.

17.2 Removing Data

If a copy of data has been deleted at a replica, machines should be able to distinguish between deleted copy and no copy. This can be solved by marking the copy to be deleted as "deleted", and not actually deleting it, which is analogously called as "issuing death certificate". But there should be some kind of clean up that actually deletes the copy.

17.3 Implementation Issues

17.3.1 Primary-based Protocols

Primary-based protocols assume that there is a coordinator/primary replica for each data item, who takes care of all writes.

17.3.2 Replicated-Write Protocols

Here, there is no notion of coordinator/primary replica, and a write can take place at any replica.

17.4 Remote-Write Protocols

This is a primary-based protocol. There is a primary replica through which all writes are propagated. Any write that comes in to any machine has to be notified to the primary replica, which then notifies about the update to all other replicas.

17.5 Local-Write Protocols

This protocol also has a primary replica, but the difference is that the primary/master is not static, but dynamically changes as the activity of machines changes. It is essentially a performance optimization of Remote-Write Protocols. This protocol will ensure sequential consistency in the system.

17.6 Replicated-Write protocol

Replicated-write protocol does not have any master/primary replica. Any replica is allowed to update.

17.6.1 Quorum-Based Protocol

Quorum-based protocol says that if there are N replicated servers in the system, and one server receives a write, it should ensure that at least (N/2+1) servers would get that update.

There are two quorums, namely Nr which says that Nr number of servers have to contacted to perform a read, Nw, which says that Nw servers have to be contacted to perform a write.

Quorums should be chosen in such a manner that the following rules are followed

1. (Nr + Nw) >N This rule ensures that a data item is not read and written by two transactions concurrently.
2. Nw >N/2 This rule ensures that two write operations from two transactions cannot occur concurrently on the same data item. These two rules ensure that one-copy serializability is maintained.

17.6.2 Giffrord's Quorum-Based Protocol

This protocol exactly implements the logic mentioned in the above section, i.e Quorum-based protocol. Voting takes place for 'read' and 'write' operations as per the rules mentioned in the protocol.

17.7 Replica Management

Issues such as Geo-locations of the replicas, number of replicas ext. have to be taken care of. This will be discussed in further lectures.

17.8 Fault Tolerance

There are many reasons for a system or service to crash. In a distributed system if a machine fails, other machines in the system can share the responsibility of the crashed machine. Fault tolerance deals with the semantics and the mechanisms of dealing with failed machines.

Fault tolerance system should provide services despite faults, such as transient faults, intermittent faults, and permanent faults.