**IS 651, Distributed Systems**

**Homework #3**

**Due Mar 20, 2020 Answers will be posted online after the deadline**

**100pts**

**Q1**. (30pts) Google file system.

Q1-1. (10pts) What are the benefits of having multiple data copies in google file system?

**For fault tolerance. To provide high availability of data, i.e., if one copy is lost, there are other two available data. To have three copies, it guarantees data consistency as well.**

Q1-2. (20pts) Why does the client only need to talk to one chunkserver to read data? Is it possible that the client gets outdated data by connecting to one chunkserver?

**Since all nodes have the same copy of data, the client only needs to talk to one chunkserver.**

**No it’s not possible. This is because the write process guarantees that all the three chunkservers are consistent. (the leading chunkserver finishes writing after it gets confirmation from the other two backup chunkservers).**

**Q2.** (20pts) Explain the problems with the #4 Majority Votes approach for mutual exclusion. Draw a figure or write a scenario to explain your answers.

1. It’s not fair. Some node who sends the request later may get the lock first (due to network delay) – a minor point
2. It may have deadlock (major problem). For instance, consider a scenario of 5 nodes. It’s possible that node 1 gets ‘yes’ from node 1 and 3, node 2 gets ‘yes’ from 2 and 4, node 5 gets ‘yes’ from itself. No node is able to get majority votes, a deadlock.

**Q3**. (50pt) Are the following statements true or false? Briefly explain your answers.

Q3.A. Sequential consistency guarantees causal ordering.

**T. We can think it’s true in most cases since all writes must be seen in the same order by all processes.**

-- This is true because sequential consistency guarantees that all nodes see exactly the same sequence of the write values, e.g., all nodes see either a before b or b before a (if a and b are concurrent). Causal consistency allows different nodes to see different sequence but they can also see the same sequence. Since all nodes already see the same sequence in sequential consistency, they will for sure satisfy causal consistency. Causal consistency just relaxes the requirement so it is easier to guarantee causal than sequential.

Q3.B. Sequential consistency does not admit a scalable implementation because it requires a total ordering of all operations in the system.

**T. mostly because other processes might need to wait for the update to be completed.**

Q3.C. Obeying all causal orderings is sufficient to guarantee convergence of state maintained by multiple replicas.

**T**

-- It is true since causal ordering can capture all concurrent events. As we discussed last time (also in the slides), if we know events are concurrent we can break the tie (e.g., using id) to determine a 'global order'. Examples such as Dynamo db uses it to make sure that concurrent writes are merged. This is also known as convergence.

Q3.D. Causal consistency is sufficient to guarantee that an airline never overbooks its seats when multiple clients reserve seats on the same airplane.

**False. Two clients may (write) concurrently and they do not have to be detected by other people.**

Q3.E. Sequential consistency is sufficient to guarantee that an airline never overbooks its seats when multiple clients reserve seats on the same airplane.

**T. All the writes are reflected in the system.**