CSE 452/M552
Problem Set
Due: 9pm, Monday, March 16, 2015

This problem set is to be done individually. Please try to keep your answers short and to the point.

Questions 1-5 are a set of true statements about some of the systems we have discussed in class. For each statement, give a short example illustrating how it can occur.

1. Two events can have the same Lamport clock time value, but can occur an hour apart in real-time.

2. For the same distributed program, a serializable (sequentially consistent) system with write-through caching will sometimes do less communication than (a serializable) one with write-back caching.

3. A distributed system with two-phase commit, but where locks are released before the commit reaches disk, is not serializable. Give an example.

4. Updates in Dynamo are eventually consistent but not serializable. Give an example of how a set of non-serializable updates might occur in Dynamo.

5. In the Google File System, a file may contain two copies of the same record. Assuming the application is non-trivial (e.g., intends to write each record once), give an example sequence of events that would produce duplicate file records.

Short answer:

6. Facebook uses a three tier system for implementing its website. An array of front-end servers interacts with web clients (each client is hashed into exactly one front-end server); these front-end servers gather the information needed to render the client web page from an array of cache servers and a separate array of storage servers. Hashing is used to locate which cache and storage server might have a particular object (e.g., a friend list, or set of postings). The number of front-end servers, cache servers, and storage servers is not identical (the numbers are chosen to balance the workload), so in general, all front-ends talk to all cache servers and all storage servers.

   The cache servers (called memcache servers) are managed as a "lookaside" cache. When rendering an object on a page, the front-end first sends a message to the relevant memcache server; if the data is not available, the front-end (not the cache) then retrieves the data from the relevant storage server. The front-end then stores the fetched data into the memcache server. On update, the front-end invalidates the cached copy (if any) and updates the storage server.

a) What semantics (serializable, eventual, inconsistent) would occur if the front-end first invalidates the cache, and then updates the storage server? Briefly explain.

b) What semantics would occur if the front-end updates the storage server and then invalidates the cache? Briefly explain.

c) What semantics would occur if the front-end invalidates the cache, updates the storage server and then re-invalidates the cache? Briefly explain.

d) An employee at Facebook suggests adding a write-token to the memcache server. When a front-end wants to change a value, it sends a message to memcache to atomically invalidate the entry and set the write-token; subsequent accesses to the server stall. The front-end releases the write-token when the data is updated at the server, allowing stalled accesses to proceed. What semantics would occur in this algorithm? Briefly explain.

7. What is the maximum number of unique values that can be proposed to a group of k Paxos acceptors (for a single instance of the protocol)? Briefly explain.

8. In Paxos, suppose that the acceptors are *A*, *B*, and *C*. *A* and *B* are also proposers, and there is a distinguished learner *L*. According to the Paxos paper, a value is chosen when a majority of acceptors accept it, and only a single value is chosen. How does Paxos ensure that the following sequence of events *cannot* happen? What actually happens, and which value is ultimately chosen?

   a) A proposes sequence number 1, and gets responses from *A*, *B*, and *C*.
   b) A sends accept(1, "foo") messages to *A* and *C* and gets responses from both. Because a majority accepted, *A* tells *L* that "foo" has been chosen. However, *A* crashes before sending an accept to *B*.
   c) B proposes sequence number 2, and gets responses from *B* and *C*.
   d) B sends accept(2, "bar") messages to *B* and *C* and gets responses from both, so *B* tells *L* that "bar" has been chosen.

9. For the Viewstamped Replication algorithm described in the reading list, outline five ways that a byzantine node would be able to cause correctness or liveness to be violated.

10. In Spanner, what would happen to the system performance/correctness if the error bound with true time is infinite?