Stanford University
Computer Science Department
CS 240 Quiz 2
Winter 2004

March 7, 2004

This is an open-book exam. You have 50 minutes to answer eight questions. Write all of your answers directly on the paper. Make your answers as concise as possible. Sentence fragments ok.

NOTE: We will take off points if a correct answer also includes incorrect or irrelevant information. (I.e., don't put in everything you know in hopes of saying the correct buzzword.)

Question	Score
1 - 2	
3 - 4	
5 - 6	
7 - 8	
total	

Stanford University Honor Code

In accordance with both the letter and the spirit of the Honor Code, I did not cheat on this exam nor will I assist someone else cheating.

Name and Stanford ID:

Signature:

Answer the following eight questions and, in a sentence or two, say *why* your answer holds (5 points each).

1. From the FLASH paper: what are Figures 9 and 10 really trying to measure? Briefly describe a better experiment to do so.

   They are trying to show the relative performance of the webservers as a function of file cache miss rate. The problem with the experiment as it stands is that you have no idea what the real miss rate is, so it's hard to draw conclusions. A better experiment would be to precisely control it. For example, have two files: one in core, the other out of core (set `O_DIRECT` *as in capriccio) and for a miss rate of N% do N references to the out of core and 100-N references to the one in core.*

2. Give three examples from the papers covered by this quiz (but not in the end-to-end paper) that make use of some version of the end-to-end argument either explicitly or implicitly.

   (a) *GoogleFS: validity and elimination of duplicate record appends is not builtin to the file system, but is synthesized by clients.*

   (b) *LBFS: does not guarantee that the database is consistent but uses an end-to-end check to tell when its invalid and then regenerates it.*

   (c) *GoogleFS: system does not take request and guarantee it gets out there, client has to sit and retry if it fails.*

3. What is the reasoning behind the GoogleFS paper's claim that

   (a) applying mutations to chunk in same order on all its replicas AND

   (b) using chunk version numbers to detect staleness

   guarantees a mutated file region is defined and contains the last mutation? Is this claim correct?

   A defined state: all clients see the same valid record. The key feature is that all replicas must have the same state so that it does not matter where you do a read from. If all replicates play mutations in the same order and all the replicas you can read from are not stale, then you (and your friends) will always see the same value. It's not really correct, since clients cache replicas and so can read from a stale one before it gets shut-down.

4. The livelock paper states you can use one of two approaches in solving livelock: "(1) do (almost) everything at high IPL, or do (almost) nothing at high IPL." As they describe these approaches, in what sense are they actually the same?

   Both will shut off work generation when processing packets. The first by setting a flag and disabling interrupts, the second by doing everything in the interrupt handler, also preventing further interrupts from occuring until it is finished.

5. The NFS paper says that because (1) NFS servers are stateless then (2) the server must write data to disk before replying to the client. Give an intuition and example for why (2) is implied by (1). Note, the paper does a poor job in explaining this, so you will have to be more concrete and insightful than they are.

   If the server does not write its state to disk, then if the server crashes while a client program is doing a sequence of operations the file system state will (for example) go back in time by losing some updates. They mention that the client would otherwise have to detect server crashes and reply its modifications.

6. LBFS lazily checks that the SHA-1 hash in its database matches the actual data contents. At which steps in Figures 2 and 3 does this validation most likely occur and why?

   For read: before sending the hash values back. Otherwise the client would use bogus values. The client similarly checks after it gets the gethash. For writes: the client checks before sending hashes, and the server rechecks before telling the client which ones it has. In both cases if they did not check, could get bad values.

7. For the workloads where LFS does worse than FFS: when is it plausible that the relative difference between the systems decreases on a RAID system?

   Worse workload: random writes, followed by sequential reads. It should do relatively better on a RAID since two disks could be seeking in parallel. You could also state that it wouldn't do better since a well-organized RAID could stream large files.

8. How could Jeremy alter VMware to prevent the forwarding system in the livelock paper from livelocking *without* reducing MLFRR much?

   For the forwarding system (not screend!) he knows that one packet in should = one packet out. Thus, livelock = many packets that come in do not make it out (occassionally you may lose things even not under livelock). So the key is to figure out roughly what the MLFRR is and then never exceed that packet input rate. A simple approach is to gradually increase the packet delivery rate until he sees a non-negligible loss rate. A more clever approach would be to do something like a binary search.