CS 240 - Final Exam

Stanford University
Computer Science Department

June 5, 2012

!!!!! SKIP 15 POINTS WORTH OF QUESTIONS. !!!!!

This is an open-book (but closed-laptop) exam. You have 75 minutes. Cross out the questions you skip. 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
1-4 (30 points)	
5-7 (25 points)	
8-11 (35 points)	
Total (max 75 points):	

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 number:

SCPD? (Yes / No)

Signature:

1. *(10 points)* You write a file system for a storage device that guarantees bytes are written atomically (i.e., if a crash occurs during a storage write the written byte either has the old value or the new) rather than traditional sector atomicity (where the written sector has either the old or new value). For a non-journaling Unix file system: how do you ensure that meta data (inodes, directory entries) are consistent? For a journaling FS: how would you do this? (Don't forget the journal!) Does your job get simpler?

2. *(5 points)* Consider the following pseudocode:

```
write_to_file_and_sync();
display_message_to_user();
write_to_file_and_sync();
display_message_to_user();
...
write_to_file_and_sync();
display_message_to_user();
```

Would you expect this program to achieve a significant speedup when run on xsyncfs? Why or why not?

3. *(5 points)* X and NOTX are arguing in the hallway about adding journaling to NFS. X claims this will help performance and points to a graph from a cs240 paper to justify this claim. NOTX says journaling will just add complexity and not help since you have to write everything to disk anyway. What graph is X talking about? (Hint: not in the NFS paper.) Who is more correct?

4. *(10 points)* Some kid in the hall tells you the key to happiness in NFS is to store the NFS data on a Venti server and to add the Venti hash of the entire file to the file handle (which now comprises the inode number, the generation number, and the venti hash). Explain how to modify the NFS protocol so that: (1) writes by a single process work, (2) concurrent writes by two processes do something sensible and (3) when you close a file, the contents are exactly what you wrote or that of a later writer, not a mix of the two (so: close-to-open consistency where `close` does an atomic flush of a client's modified data). Note: don't use leases or other cache coherence gimmicks.

5. *(10 points)* Venti left a lot of cycles on the table. Describe a way to increase Venti's performance by:

   (a) Keeping extra state on the client.

   (b) Keeping extra state on the server.

   (c) Exploiting the fact that you can verify the contents of a Venti block.

6. *(10 points)* Modify the `map` and `reduce` code in the paper to implement (1) grep and (2) sort. You don't have to put in every brace, but if the code in the appendix gives a function to use you should use it.

7. *(5 points)* Haystack writes some data synchronously and some data asynchronously. Give the intuition for why either choice is made and the general approach it uses to guarantee correctness for asynchronous writes.

8. *(10 points)* Your cs140 partner wants to sell a FB-B-Gon service that will use denial-of-service attacks to prevent access to embarrassing photos posted to facebook. His plan is to request the photo, determine the machine storing the photo from the URL returned by the haystack directory, and then attack this machine by generating lots of requests for fake pictures to it, which he claims will result in disk access per request. Will this approach work? (Always, sometimes, not really.) Please give your reasoning.

9. *(5 points)* We've discussed some of the features of the x86 that make virtualization difficult. What are some that *help* virtualization? (Don't include recent features like Intel VT or AMD-V that were *designed* to enable virtualization. If you don't know what "VT" or "AMD-V" mean, don't worry about it.)

10. *(10 points)* You know the future. Concisely describe three optimizations you could do (and how) that would clearly give performance wins for papers from this quarter other than predicting which cache entry to evict.

11. *(10 points)*

(a) Some code doesn't care about checking for `malloc` failures. A routine `foo` checks the result of `malloc` against null and then calls another routine `bar` which does not. What belief-argument can you make about flagging an error?

(b) A routine `foo` calls a routine `bar(p)` that dereferences `p`. What belief propagates to `foo`? It also calls `baz(q)` which checks `q` against null. What belief propagates to `foo`? (For your answers, state why or why not a belief is propagated.)

(c) Give three examples of code (not from the paper!) that imply either MAY or MUST beliefs. Explain how to find an error using one or more of these beliefs.