CMU 18-746/15-746 **Storage Systems** **25 Feb 2009**

Spring 2010 **Exam 1**

Name: ______________________

Instructions

There are four (4) questions on the exam. You may find questions that could have several answers and require an explanation or a justification. As we've said, many answers in storage systems are "It depends!". In these cases, we are more interested in your justification, so make sure you're clear. Good luck!

If you have several calculations leading to a single answer, please place a $\boxed{\text{box around your answer}}$.

Problem 1 : Short answer. [48 points]

(a) Joe used FUSE to implement the client-side functionality of a simple distributed file system. On Joe's clients, the kernel-level FUSE module redirects each intercepted file system call to a user-level process that Joe wrote, called Joes_FS_client, using the libfuse library. Joes_FS_client sends each such call to a program running on the server, waits for a response, and provides that response to the FUSE module. If no response is received from the server within 100ms, Joes_FS_client resends the request. After extensively testing with file reads and writes, Joe extends his tests to include file creates and deletes, and is surprised to uncover rare error cases where the create fails with "file exists" or the delete fails with "file does not exist". Assuming that Joe's code works as described (i.e., no unspecified behavior), what is the most likely cause of the errors?

Reads and Writes are idempotent, so the retries will always succeed, even after the original request is serviced. Creates and deletes are not idempotent, so retries for them will fail if the server has already serviced the original request.

(b) Moe argues that the file system cache, currently kept in RAM, should instead be kept in Flash-based storage because Flash is cheaper per byte than RAM. What is the biggest performance concern you would raise with this design choice? (That is, why might it result in lower performance than using RAM for the file system cache.)

We accepted a wide variety of answers for this question. Some are listed below. Note that to receive full credit for this question, your answer had to list a performance problem, not one related to durability or correctness.

- *Flash is still slower than RAM, so using flash would result in slower accesses*

- *Small writes are slow in Flash due to the write amplification problem. Since flash is only block addressable, small writes such writes require the entire block to be erased and re-written.*

(c) Roe is planning to use 100 5-disk RAID-6 arrays in his disk array system. Assuming his system does not support rebuild, what will be the mean time to data loss for Joe's final disk array system (as a function of $MTBF_{disk}$)?

Data loss will occur when one of the 100 disk arrays loses data.

$$MTBF_{set} = \frac{MTBF_{disk}}{5} + \frac{MTBF_{disk}}{4} + \frac{MTBF_{disk}}{3} \tag{1}$$

$$MTBF_{set} = \frac{47 \times MTBF_{disk}}{60} \tag{2}$$

$$MTBF_{array} = \frac{MTBF_{set}}{100} \tag{3}$$

$$MTBF_{array} = \frac{47 \times MTBF_{disk}}{6000} \tag{4}$$

(d) Toe is shocked that BigTable always uses a 3-level B-tree, since it puts an upper bound on size and uses more internal index nodes than necessary for smaller tables. What is the biggest benefit that the BigTable creators would claim in favor of their design choice?

The biggest benefit is simplicity for code development, testing and maintenance. Many students gave performance related answers. A 3-level B-tree might improve performance in some cases, however, it is not the reason behind this design decision. We didn't give any credit for performance related answers.

(e) Poe relies on a distributed log entry collection application. It consists of a process running on each node that periodically checks the local log for new records, opens the shared log file on a file server, appends the new records to the shared log file, and closes it. Explain the most likely reason he could observe some records being lost when using a bug-free NFS server to store the shared log file.

NFS offers weak consistency when multiple clients write to the same file (within the same 30 sec *interval). The latest write wins in this case. It is most likely that some of Poe's records have being overwritten by other clients.*

(f) Doe's organization has 1TB of user data and wants a daily backup. Each day, 100GB of user data is updated, and all user data (and modifications) are unique. How much data (maximum) is stored by the back-up system for each of the following backup strategies:

- keep one week of daily full backups:

  Total: 7TB

- keep one weekly full backup and six daily incremental backups:

  1TB for the weekly full back-up and 600GB for 6 incremental backups (Total: 1.6TB).

- keep one week of daily full backups with perfect de-duplication:

  Since all user data are unique, we will need a full back up and 6 incremental backups (Total: 1.6TB).

Problem 2 : More short answer. [24 points]

(a) Identify (for Koe) a workload (i.e., access pattern and file characteristics) for which performance will be better with AFS than with NFS.

There were many potential answers to this question, the most obvious having to do with write aggregation. An example is listed below.

Koe opens up a non-shared file and keeps modifying/overwriting most of the file before eventually closing it. E.g. overwrites the contents of the file 20 times. This is because AFS won't need to send intermediate data to the server, only the final version will be sent.

Note that "concurrent write-workloads" was not a valid answer to this problem, as AFS's performance could suffer in comparison to NFS for them. For each concurrent write, assuming that the file is closed after each write, the server will have to break callbacks to every other client, forcing them to refetch the entire file's data.

(b) Loe and Woe use separate computers and each run a program for displaying a file's contents that open a file, keep it open, and re-read its contents every second. Loe updates the file (via a different program) and tells Woe, but Woe does not see the new version even after a few seconds. Which of these three distributed file systems could **not** be the one they are using? Sprite, AFS, or NFS. Explain your answer.

Sprite — disables caching for shared files and thus it ensures that every read sees the most recent write.

(c) Many parallel file systems stripe data across multiple servers in order to improve bandwidth. Foe is convinced that PLFS should be irrelevant, given such striping. Describe a workload for which he is incorrect and explain why the workload creates a performance problem (without PLFS)

Example: Workload in which 8 clients each update the Nth 512-byte region of each 4KB block. The processes will conflict on every block, needing to read it and then write it back, bottlenecking forward progress and being very inefficient. PLFS on the other hand will create an index file, provide each process with a different file that it can write to and convert the N-1 writing pattern to a N-N one.

Problem 3 : Solid State Disks. [28 points]

(a) Solid state disks (SSD) are built out of NAND flash technology. These parts are comparable in price to magnetic disks but store much less data. Why is the entire storage industry trying to figure out how to use SSDs in their storage systems?

100X more IO/sec is the primary answer although lower power consumption and an expectation of a more robust component.

(b) Flash-n-disk (your project 2) stores the contents of a file system split between two devices, a Seagate Barracuda magnetic disk and an Intel SLC SSD. Suppose Lin asked you to test another student's Flash-n-disk by creating 1 million files in the file system and then running "ls -lR" on it. The 'ls' takes about an hour to finish. Lin says she thinks it took too long and asks you two questions:

- approximately how long should the 'ls' have taken?
- what is the most likely mistake made in the student's Flash-n-disk implementation? Explain why it is wrong.

Backup scans should read only attributes if the files are unchanged, at 10,000 small reads per second, taking perhaps 200 seconds to scan 1M files in each of the target and backup file system. We suspect that the student who wrote this did not replicate the attributes of the large files in the flash, so that the backup scan has to get the files' timestamps from the disk, requiring lots of disk seeks at 5 sec per seek, or 200 files/second. Since an hour is 3600 seconds, this run was scanning 2M files / 3600 sec = 2K / 3.6 sec = 6K / 10 sec = 600 files/sec, which suggests that many of the accesses were going to disk.

(c) Flash-n-disk uses one magnetic disk for every SSD, but SSDs are much more expensive per byte than a magnetic disk. The MBA student in the back says that we should amortize the SSD cost over more magnetic disks—that is, split the file system over one SSD and five magnetic disks. Then, to make this new Flash-n-disk even better, we decide that really big files, say $>$ 1 MB, will be striped over all the magnetic disks. This introduces new challenges for your implementation, because big files are stored with parts in each of five separate magnetic disks. Identify an implementation problem that this would create in the basic Flash-n-disk design and what would you do about it?

A single symbolic link will not allow reading or writing of a big file with one open and redirected *reads or writes. There will need to be a different type of dangling link with five pointers. And the one file opened in Flash-n-disk by applications will have to cause five files to be opened in magnetic disks, with each read or write (maybe split and) redirected with a recomputed offset to a specific disk.*

Problem 4 : Instructor trivia. [up to 2 bonus points]

(a) List one paper you have read so far for which Garth is an author.

RAID, PLFS, etc.

(b) What is this semester's record for the longest time with a single slide displayed (while lecturing) by an instructor in this class? Which instructor?

Both Greg and Garth are still arguing as to which one of them deserves to be listed as the answer. Our favourite answer from those submitted by students was: "Hugo Patterson because he gave a talk with 0 slides, so technically his time per slide was infinity."

(c) How does Greg suggest implementing functionality in a decentralized system, whenever feasible?

As centralized as possible. This keeps thing simple.

(d) Which instructor had more pie thrown at him this semester

Greg by a good mile

(e) Which instructor or TA is most in need of a vacation? What should they do?

Students provided a lot of good answers to this question. We accepted all of them :)