CMU 18-746/15-746 **Storage Systems** **29 April 2015**

Spring 2015 **Exam 2**

Name and Andrew ID: _____

Instructions

There are three (3) problems 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 box around your answer.

Problem 1 : Short answer. [48 points]

(a) Company XYZ has a distributed file system in which each data block is replicated on two servers. The new CTO proposes that the design should be modified to improve data reliability by having the client software fetch both copies of a block, on every read, and compare them to ensure that the right data is returned to the application. (That is, the new CTO wants to prevent returning of corrupt data.) Identify a problem with this approach and a better design that provides the intended benefit. Explain your answer.

ANSWER:
It doubles the network bandwidth used for reads, as well as forcing the client to wait for both responses, while doing nothing more than allowing the client to detect a problem. The same benefit can be achieved by storing a checksum with each block, if the checksum computation includes the block ID.

Note that the approach cannot both detect and correct erroneous data, since there would be no way to tell which replica is correct if they do not match (and there is no other evidence).

(b) Todd argues that the only sensible option for where FTL software should run is on a Flash device-embedded co-processor. Describe and explain one situation where it would be better to run FTL software in a host device driver.

ANSWER:
In the host driver, the FTL could run on a faster processor and have access to more memory, allowing more caching, bigger mapping tables, and more logic for compressing/dedpulicating data rapidly. There is also an argument that putting FTL in the host allows the SSD to be a lower-cost, lower-power device.

Note: this response captures multiple possible situations.

(c) When one disk of a mirrored pair fails, reconstruction (a.k.a. rebuild) takes at least as long as the time to completely read the contents from the mirror disk. When a server fails in HDFS or GFS, restoration of full redundancy can be done more quickly than that. Explain why.

ANSWER:
In HDFS and GFS, each data block stored on a given server is replicated on independently selected other nodes. So, re-replication of the lost blocks can be done using many other servers to read different ones of the blocks, in parallel, and writing them to various other servers, again in parallel.

(d) Older file systems (e.g., FFS and ext2fs) used an inode table in which each inode was stored at a predetermined and unchanging LBA. Many modern file systems (e.g., WAFL) instead keep inodes in an inode file in order to not have to always write a given inode to the same LBA. What is **another** benefit of using an inode file instead of a traditional inode table? Explain your answer.

ANSWER:
When using an inode file, a file system does not have a static bound on the number of inodes (just extend the file) and does not have a configuration time parameter that must be set right (i.e, the number of inodes).

(e) In most HDFS and GFS deployments, each machine has multiple disks. Harry points out that write performance could be significantly higher if data were replicated on three disks on the same server instead of being transferred to and stored on three different servers. Give and explain **two** arguments against making this design change.

ANSWER:
There are a number of arguments that work. The failure of a single server node would eliminate all replicas. It is not always true that writing to a local disk is faster than writing to a remote node, because the network can be faster than a disk. Load balancing can be negatively affected, both because the burst of writes on busy clients and because all reads of that newly created data would have to go to the same single server.

(f) Imagine an application that modifies 1 byte of a 100KB file. How much data is transferred to the file server for AFS (version 2)? How much for NFS (version 3)?

ANSWER:
For AFS: the entire file (100KB). For NFS: one NFS block (size not given in question).

Problem 2 : More short answer. [48 points]

(a) The original AFS design called for session semantics wherein modifications by other clients are not exposed to it. To do so, a client made a private local copy of a whole file on open. Modern distributed file systems, including modern AFS implementations, do not perform such whole-file caching. What is an alternate mechanism that many distributed file systems provide in order to allow a client to have a private view of a file during an open file session? Explain your answer.

ANSWER:
File locks (sometimes also known as deny mode), which allow a client to have exclusive access to a file.

This question turned out to be confusing to many. Many students misunderstood this question to be asking for an alternate caching scheme to whole file, discussing block-based caching as the alternative. The question is about having a private view for a session, so a solution that does so for just a part of a file for a part of a session would not help.

Some students described a solution that could be used, such as server-based versioning with background merge (of some sort), rather than describing a mechanism provided in "many distributed file systems", as asked.

(b) Most distributed file systems do not allow client machines to directly modify directory contents, instead having RPCs for each such modification (e.g., create, delete, rename), even though this can result in many more client requests (e.g., when deleting all files in a directory). (Even in dist. FSs that allow read caching of directory contents.) Give and explain one reason for this common design decision.

ANSWER:
There are a number of good reasons, most of which are examples of the primary reason: complexity. Correctly handling distributed metadata updates is much more difficult than doing them on the server. For example, there may be many lock acquisitions required, with potential deadlock difficulties, especially for situations like rename and allocation of scarce resources. Also, any data structure corruption caused by a faulty client would affect the correctness of the server and other clients, creating a need for integrity checking at the server and possibly having errors that cannot be reported for a specific FS operation because they are discovered until later.

(c) When redesigning its file system, Google decided to use BigTable to store file metadata rather than a primary server (with a backup). What was a primary reason for that decision? Explain your answer.

ANSWER:
They wanted a solution that offered greater scale than a single server, and BigTable was already implemented, tested, and supported.

(d) Imagine a client machine that has an NFS file system mounted in its file system namespace. If an application renames a file such that it is moved from a directory in the NFS file systems to a directory on the local disk file system, could the file be lost due to an ill-timed machine crash? Explain your answer.

ANSWER:
Yes, in many systems. The rename is across file systems, meaning that it is copied from NFS to the local file system, then deleted from NFS. The NFS delete is synchronous, by specification, but the local creation may not be, especially for the file data. On the other hand, if the rename implementation ensures that the new name's copy is persistent (e.g., via fsync), then the file could not be lost in this way.

(e) Most scalable distributed file system designs that directly implement software parity-based redundancy, rather than replication, do so by having the client that writes data compute the parity and send the data stripe units and parity to the servers that will store them. Why would they do this work at the client library rather than at a server? Explain your answer.

ANSWER:
One reason is that there are more CPU cycles in the client machines, collectively, than in the server machines. Another is that the full stripe is often already in client memory, meaning that the parity can be computed at the client without moving data between machines; each server, on the other hand, would be storing only a portion of the stripe and those portions would need to be brought together to compute the parity. A third is that the client does not have to wait for all server to response, so long as enough servers have responded such that the data can be reconstructed, reducing tail latency.

(f) Joe's file server supports online snapshots, and he wants to use a strategy of hourly snapshots to ensure that no more than one hour's worth of work would be lost to application or user error. Jill suggests a strategy of keeping the four most recent hourly snapshots, one daily snapshot (most recent midnight), and one weekly snapshot (most recent Sunday midnight). But, Joe is concerned because the company cannot afford to purchase six times as much server storage capacity that it uses without the snapshots. What technical explanation could Jill provide to Joe in order to convince him that it will not require six times the capacity? Explain your answer.

ANSWER:
In practical implementations, a snapshot only requires keeping the data and metadata that has changed since the previous snapshot, plus a small amount of book-keeping metadata. Since most data does not change frequently, the total capacity for the six snapshots should be far less than 6X the non-snapshot capacity.

Problem 3 : Instructor trivia. [up to 2 bonus points]

(a) What does the acronym TLA stand for?

Three Letter Acronym.

(b) Which guest lecturer used the blackboard instead of projected slides?

Ram Kesavan.

(c) How many parts did lab 2 have?

Four, counting the writeup.

(d) Which was your favorite lecture and why?

The cancelled one earlier today (day of exam) ?

(e) If the 746 staff could do a post-semester retreat, to recover from the efforts of making new labs and exams, where should they go? (Be kind ;))

Lots of fun answers... might be nice if they did something, though.