**CMU 18–746** **Storage Systems** **4 May 2004**

**Fall 2005** **Exam 2**

---

## Name: _____

### Instructions

There are five (5) 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!

### Problem 1 : Short answer. [35 points]

(a) Most "disaster recovery" solutions address site failures by trickling buffered updates out to remote replicas. Why are updates buffered (rather than sent immediately) and what trade-off is inherent in the amount of buffering employed??

*Buffering allows writes to be grouped together. This coalesces writes and prevents short-lived data from ever having to be written. The downside to buffering is that it opens up a window during which data could be lost. The longer data is buffered, the more data which could be lost.*

(b) Most "object-based storage" designs call for a centralized file manager to own the file system namespace, control concurrent updates, etc. Why not simply do this on the clients and allow them direct access to the metadata? (recall that they already have direct access to the data)

*Centralization makes a number of distributed systems problems much easier to deal with. A central server is able to ensure that data stays consistent as multiple people access a file. It allows for validation of metadata writes. Access to metadata may also be used as a method of access control. There are also issues trusting a client to control metadata.*

(c) Unlike client caches in most distributed file systems, no support exists in most WWW browser cache implementations for ensuring consistency.

(a) Why? *We accepted a number of answers for this. Too many clients for a web server to keep track of who has what cached. Most pages on the web are static. HTTP wasn't designed with dynamic pages in mind. No bad consequences for web browser being slightly out of sync.*

(b) Explain one approach to improving the situation.

*There are a number of ways to improve this. One way is to have the cache send a small message checking consistency of the cache. Another is to give expiration dates to cache data.*

(d) Greg has designed a super-duper deletion approach, which overwrites the data on magnetic disks such that it is truly impossible to get the data back from them. Explain why, in most environments, this would not be enough to completely eliminate all traces of the files holding his last two years of email.

*In most environments, mail data was likely backed up. Another possible answer would be that writing to blocks on a hard drive which are marked bad will lead to remapped sectors being written to (and the old data still remaining on the bad blocks).*

**Problem 2 : A little system analysis. [14 points]**

Consider an e-commerce system running atop an "out-of-band" distributed storage service. The system is designed for 24/7 operation, but its front-end throttles input requests to 100 at a time, and the operators no that there are breaks between one request's completion and the next request's arrival. Each request requires five 1 ms interactions with the file manager and accesses to 40 file blocks. The cache hit rate is 75%, and file data is striped across five data servers such that each request has an equal probability of going to any of the disks. Each individual disk request takes 10 ms.

(a) What is the throughput of the e-commerce system?

*Only 25% of the blocks read require a disk access and the disks are configured as an array. Each job therefore required 10 block accesses, or 2 array access (5 blocks per stripe). The demand on the file manager and disk array is as follows:*

$Demand_{FM} = 5\ visits/job \times 1\ ms/visit = 5\ ms/job$

*and*

$Demand_{Array} = 2\ visits/job \times 10\ ms/visit = 20\ ms/job$

*The file manager and disk array run in parallel, so the throughput is limited to the slower of the two.*

$Throughput = 1/Demand_{Array} = 50 jobs/sec.$

(b) Which portion (file manager or data server) is the bottleneck?

*The bottleneck is the disk array (the server with the greatest demand).*

**Problem 3 : A few disk array system design questions. [28 points]**

(a) Most high-end disk arrays include large amounts of battery-backed RAM. Explain two potential performance benefits of having such RAM.

*There are a wide range of* ***performance*** *benefits which can come just from adding RAM to a disk array, and even more from battery-backed RAM. One gain of battery-backed is the ability to do write-back caching; it becomes safe to report back to a system that data is written as soon as it makes it to NVRAM rather than all of the way to disk. Write-back caching introduces other benefits such as coalescing writes. Other benefits of having RAM include better read performance, and possible avoidance of the small write problem (see part d) but there are many others.*

(b) Departing briefly from disk arrays, imagine that instead of an array controller with a set of disks, clients talk directly to a set of storage servers that each have battery-backed RAM. Will the two performance benefits you listed above still work? State whether or not they will work as well and, if not, why not.

*We accepted a wide variety of answers to this question. If you focused on write-back benifits, the advantages to the system likely wouldn't change. If this is treated a distributed system, one might not be able to assume that NVRAM is enough to ensure consistancy. The extra work for consistancy could hamper performance.*

(c) Back to disk array systems. The battery-backed RAM is also useful for integrity in disk arrays that use redundancy (e.g., mirroring or parity protection). Explain how.

*For mirroring or parity to work properly, writes need to occur to all devices they are going to. A write can be written to NVRAM, and then finished if the array comes back up and finds the write incomplete.*

(d) Assume, for this one part, that the RAM is **not** battery-backed. How could the RAM cache improve performance for small writes in a parity-protected disk array?

*The small writes problem occurs due to the overhead of doing two reads and two writes for a single small write. In order to do a write, it is necessary to read both the data being replaced and the parity. If this is cached in RAM from previous operations, only writing the new parity and data will be necessary.*

**Problem 4 : File system (ext2) format and recovery. [24 points]**

(a) Which of the following are *not* contained in the file system inode? Check all that apply.

- size
- access times
- √ inode number of parent directory
- √ name
- reference count

(b) Can the name of a file be determined by looking at its inode? Explain why or why not.

*No. Inodes are nameless.*

(c) Can the *number* of names a file has be determined by looking at its inode? Explain why or why not.

*Yes. This is the reference/link count.*

(d) Under which scenario will an inode be placed in lost+found? Check all that apply.

- One or more directories contain an entry for the inode, but the inode's reference count is zero.
- √ The inode's reference count is non-zero and no directories contain an entry for it.
- A file was written, but there was a power failure before the file data was written to disk.
- The inode bitmap does not show the inode as being used.

(e) Which of the following ext2 data or metadata structures are located at fixed positions on disk, that is, after formatting they never get moved. Check all that apply.

- √ superblock
- √ group descriptors
- √ inode bitmap
- √ free block bitmap
- data for a given inode (file or directory)

**Problem 5 : Bonus questions. [a few bonus points]**

*There was of course no way to answer these wrong, except for not answering them. Each was worth an extra point*

(a) After our discussions of it in class, did you enable encryption on your AFS accesses? (We're just curious.)

(b) Which would you rather be required to duplicate: William's in-class calisthenics or Tim's in-class word creations? Explain.