CMU 18–746 **Storage Systems** **30 Nov 2006**

Fall 2006 **Exam 2**

Name: _____

Instructions

There are three 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. [49 points]

(a) Many in the storage industry view it as simply a matter of time before today's common storage buses (e.g., Serial ATA and parallel SCSI) and networks (e.g., FibreChannel) are replaced with Ethernet. Why are most file system designers not concerned about such a change?

Even if the interconnect technology changes, the SCSI/Serial ATA semantics will still be supported and thus the applications (file systems) don't need to change.

(b) Consider two 8-disk disk arrays using a RAID5 data distribution: one with 50GB disks and one with 500GB disks. Which one is more likely to suffer data loss? Explain. (Assume that the disks in both arrays have the same MTBF and performance characteristics, and that all other aspects of the hardware, software, and administration are identical.)

For a disk array to suffer a data loss, a second failure must occur before a first is repaired. With equal performance, a higher-capacity disk will require more time to reconstruct onto a replacement, resulting in a larger "window of vulnerability" in which a second failure can occur. Thus, data loss is more likely with 500GB disks.

(c) In experimenting with a disk array that uses parity-based redundancy, you observe interesting performance results as you consider different block sizes for your database system. You know that your workload is almost entirely random writes of whatever block size you select; you also know that there is no concurrency (one request at a time from the database system to the disk array controller). In looking at per-request latency, from the database system's perspective, you observe: 20ms for a block size of 8KB, 20ms for 16KB, 20ms for 32KB, 10ms for 64KB, 20ms for 96KB, 10ms for 128KB. Explain why this is happening.

If a request updates an integral number of full stripes, the parity can be computed from the new data without reading any "old" data. Thus, the latency is shorter because of the avoided reads. (The system measured takes 10ms for the reads, if necessary, and 10ms for the writes.)

(d) Stratos wrote data to a file, closed it, and then sent email to tell Greg that it was there. Greg opened the file right away, but didn't see the new data and complained. Stratos patiently asked Greg to try again in a minute, and Greg saw the data when he did. What distributed file system are they using? Explain.

NFS. The client cache on Greg's machine must have had a recent copy of that file's data, and NFS client caches only verify freshness when 30 seconds have passed since the last check.

(e) Many enterprise environments use both hourly snapshots (maintained by each server for the data that it stores) and nightly tape backups. Why do both?

The hourly disk-based snapshots give users easy and fast access to recent versions of their data, but are short-term and get discarded quickly (to make room for new snapshots). The tape backups are more time-consuming to save and to access, but offer long-term archiving.

(f) Block-based storage and file-based storage often get compared in terms of performance by direct measurement. Why does block-based storage often beat file-based storage, in simple data bandwidth measurements, when the block devices are "SCSI over FibreChannel" and the file servers are "NFS/TCP/IP over Ethernet"? (Assume that the two physical networks have the same bandwidth.)

The adapter cards used with block-based storage almost always provide for DMA of read/written blocks directly into specified buffers. This avoids the TCP/IP processing overheads and data copies commonly involved with file server interactions.

(g) We know that deleting data from a file system does not actually remove all traces of its contents from the disks that stored it. Many have proposed using encryption to solve this problem (by having all data be encrypted before being written to disk). Identify two practical challenges with doing so.

(a) Accessing a block has higher overhead during normal operation, because it must be encrypted/decrypted.

(b) If the encryption key is lost, the entire disk becomes unusable.

Problem 2 : Disk Striping. [14 points]

You are given a disk array with 4 disks. Assume that each disk has an average 10ms positioning time (including seek and rotational latency) and a transfer rate of 80MB/s. The array implements simple disk striping, but there is no redundancy.

(a) Assume a workload consisting of random aligned 1MB requests. Compute the average request service time and the maximum array throughput for each of these stripe unit size options:

- 256KB. *An aligned 1MB request is decomposed to 4 256KB requests, one for each disk in the array. Since the 4 requests are executed in parallel, the response time is equal to the response time for a single request (assuming synchronizd spindles):*

$$R_{256KB} = 10ms + 256KB/80 \times 1024KBs = 10ms + 3ms = 13ms. \quad (1)$$

Since every request uses all the disks, there is no parallel execution of requests. The maximum throughput is:

$$X_{256KB} = 1/R_{256KB} = 76.9req/sec \quad (2)$$

- 1MB. *Since the stripe size is equal to the request size, each request involves exactly one disk. Thus the response time is*

$$R_{1MB} = 10ms + 1MB/80MBs = 10ms + 12.5ms = 22.5ms. \quad (3)$$

The maximum possible throughput is achieved when all 4 disks process requests in parallel. Thus:

$$X_{1MB} = 4 \times (1/R_{1MB}) = 4/22.5ms = 177.78req/sec \quad (4)$$

(b) What workload information, beyond that given above, would you need in order to make a good decision between these stripe unit size options? Explain.

A smaller stripe unit size can reduce individual request service times, but limits maximum throughput when each request uses multiple disks. To make a good stripe unit size decision we need to know how much request concurrency there will be.

Problem 3 : Decentralized file system design. [37 points]

(a) Consider a decentralized file system that has a single centralized file/metadata manager and a collection of data storage servers. Each client request in our hypothetical system involves one 1ms access to the metadata server (to acquire data location information) and one 20ms access to one of the data servers.

(a) What is the largest number of data servers that can be in the system before the metadata server will be a throughput bottleneck?

Since the metadata server is 20 times faster than a data server, we can have up to 20 data servers before it becomes a bottleneck.

(b) With 10 data servers, what is the maximum throughput?

The individual demands are

$$D_{metadata} = 1ms/1 = 1ms \tag{5}$$

$$D_{data} = 20ms/10 = 2ms \tag{6}$$

Thus

$$X_{max} = 1/2ms = 500req/sec \tag{7}$$

(c) Assume 10 data servers and 50 clients, each of which performs one request at a time and thinks for 5ms between the completion of one request and the generation of the next. At the maximum throughput from (b), what is the expected response time for a request?

$$E[R] = N/X_{max} - Z = 50/500/s - 5ms = 100ms - 5ms = 95ms \tag{8}$$

(b) Consider a decentralized file system architecture as in 3(a) above again. The designers would like to reduce the metadata server bottleneck, allowing larger collections of data servers. To do so, they propose having clients cache data location information. Identify a problem that this introduces and one solution to solving it.

Caching file metadata information on the client introduces consistency problems when multiple clients update the same file.

Some solutions to the problem include:

(a) Have the metadata server track which clients have cached metadata and send invalidation messages when said metadata is modified.

(b) Have the metadata server track which clients have cached metadata and forward the latest metadata when it is modified.

(c) Have the metadata server track which clients have cached metadata and disallow metadata caching when write sharing occurs.

(c) Consider a decentralized file system architecture as in 3(a) above again. Assume now that data protection is provided by having a single parity server that stores parity for blocks that are striped across the data servers. Describe one approach to protecting the integrity of the parity from client crashes.

The problem is that parity information may be invalid if a client crashes before all the inter-related writes to data servers are completed. One solution would be to use per-client log segments, like Zebra did. Another would be to have the file manager track ongoing writes and repair parity after crashes (like many disk array controllers do).

(d) Consider a decentralized file system architecture as in 3(a) above again. Assume that the data servers know nothing about file structure; they simply support reads and writes to fixed-size blocks (much like disks). Briefly describe an approach that would allow the metadata server to control which reads and writes the data servers will execute. (Any clients can send any access to any data server, but we want a way for the data servers to reject anything other than accesses deemed ”ok” by the metadata server.)

The metadata server authorizes the data server to perform specific operations on specific block ranges and the permissions must be sent to the data servers. One way to implement this communication is to have the metadata server issue cyptographic capabilities to the client that describe the permitted actions. The client then forwards a capability tickets to the data servers with each request. Another option is for the metadata server to send authorizations to the data servers directly.

Problem 4 : Bonus Questions. [Maximum of three points]

(a) Who won The Game (Michigan vs. Ohio State)?
Ohio State.

(b) Was Greg happy about it?
No.

(c) What year will Stratos graduate?
2007 (fingers crossed).

(d) If we get Timmy a PlayStation for the holidays, what game should we include with it?
The top choice (of Timmy): Dance Dance Revolution. (Seriously)
Some other suggestions seen:

(a) NBA 2007
(b) Madden NFL 2007
(c) Call of Duty 3