CMU 18–746 **Storage Systems** **23 April 2008**

Spring 2008 **Exam 2**

Name: Solution

Instructions

There are four 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!

Problem 1 : Short answer. [56 points]

(a) Some application programs rely on temporary files to hold intermediate state during their computations. To avoid the need to "clean up" after a program crash, some programmers will use a sequence of steps like: (1) tmpfile = create(/tmp/tmpfile); (2) unlink(tmpfile); (3) compute(); (4) close(tmpfile). If /tmp is a UNIX local file system, these steps provide the program with a file to use during compute() that will automatically be reclaimed by the system if the program crashes (or exits). Explain what problem such a program will encounter if /tmp is an NFS file system.

An NFS server will actually delete the file, when unlink() is called, causing any attempted use of it during compute() to fail. Since NFS is stateless, it does not know that a client application has the file open (like a local file system does).

As a note of general interest: some NFS clients try to address this issue by renaming an open file, when unlink() is called, and then unlink()ing the invented name when the file is closed. This approach creates problems with cleaning up leftover files, if the system crashes without doing the unlink(), and with user confusion if they notice the renamed file in their directory listings.

(b) Most storage networks are said to use asymmetric protocols. For example, a SCSI target can issue a DISCONNECT or a RECONNECT against a SCSI initiator, while the reverse is not true. Why are asymmetric protocols used for storage networks?

Asymmetric protocols allow one side of a communication (usually the device, in storage networks) to be simpler than the other, disconnecting and reconnecting as is convenient for it.

(c) RAID level 5 fixed a bottleneck in RAID level 4. Why might the performance of a log-structured file system with a 100% read cache hit rate be the same for a disk array configured to use RAID-4 as for the same disk array configured to use RAID-5?

Because every request may be using all disks equally. There are no reads, according to the problem, and writes can be of full log segments. If log segments have sizes that are a multiple of the full stripe size (i.e., one stripe unit per device associated with one parity unit), and they are aligned to stripe boundaries, then every write updates the same number of stripe units per disk. The parity can be computed and written at the same time.

(d) Imagine a file system that computes and stores a checksum of each file in a separate "checksum database". Suppose that you read a file from a disk array, compute its checksum, and find that it does not match the value in the checksum database. Can use of RAID-5 guarantee the ability to correct whatever error has occurred? Explain.

No, the parity used in RAID-5 can be used to recompute a lost stripe unit, but it cannot correct arbitrary errors, especially when multiple stripe units within a stripe are corrupted.

(e) Once upon a time, Greg wrote a simulator for his research. Sometimes, the simulator would take many hours to complete. To help him monitor progress, he modified it to open a file (cleverly named "progress") when it starts and to write a line of text describing its progress every hour. When a friend offered to share his powerful compute server, Greg ran the simulator on the compute server, using a distributed file system to provide input files and a place to store the "progress" file. But, when he checked the "progress" file from his desktop computer three hours later, he found it empty even though the simulator appeared to be running. Later, when the program completed, it produced the proper output and the "progress" file had all of the expected output. What distributed file system was being used? Explain.

AFS. The data from one "open() session" is not made visible to other open()s until the file is close()d.

(f) CompanyX has created a file system that supports efficient snapshots based on copy-on-write and exposed in directories named /.snapshot-YYYY-MM-DD-HH-MM-SS. With their file system, they provide a backup tool that runs as an application using standard POSIX interfaces. CompanyX's lead engineer believes that the incremental backup feature of the tool could be made much more efficient (fewer disk reads and less data stored into the backup) by allowing it to access internal structures of the file system implementation. Explain how the engineer might achieve their claim.

By examining the copy-on-write structures, one can identify which blocks have been modified since the last snapshot. Using POSIX interfaces, one can only identify files that have changed. So, a file in which only one block changes would be backed up in its entirity by the current tool, but just the one new block could be backed up by examining the copy-on-write structures.

(g) Fred has designed a direct-access storage system in which clients are provided with cryptographic capabilities that are checked by data servers before any client read or write request is serviced. One requirement given by the marketing team is that it must be possible to revoke a client's access to a given file's data immediately upon a change to that file's permissions. Fred's first thought is to implement this requirement by using a callback-like approach of having the file manager send a "throw away capability X" message to any affected client, when necessary. Identify and explain a problem with this approach.

If a client machine does not receive, or chooses to ignore, the callback message, then the client can just keep using its capability. (This is why such revocation is done by telling the data servers that certain capabilities are no longer valid.)

Problem 2 : Disk Arrays. [19 points]

You are given a disk array with 4 disks. Assume that each disk has an average 7ms seek time, a 6ms full revolution time (i.e., 6ms to rotate 360 degrees), and a transfer rate of 80MB/s. The array is configured to use RAID-5 with a stripe unit size of 64KB.

(a) Assume a workload consisting of random aligned 4KB requests. Compute the average request service time and the maximum array throughput for each of these cases:

- 100% reads.

  Each read requires a single disk access. Assuming that each 4KB block is on a single track in a zone providing 80MB/s media transfers, the average time to do the disk read is:
  *7ms (seek) + 3ms (rotational latency) + 0.05 (transfer) = 10.05ms Since each read only need one disk: total throughput: 4 * 1 /10.05ms = 398 req/sec.*

- 100% writes.

  Assuming the RMW approach, each write involves reading and then writing a data block on one disk and a corresponding parity block on another disk. For each, the write is to the same block as the read, so it involves a full rotation but no seek. Assuming the above plus identical service times for the requests on the two disks, the average time time to complete the disk write is:
  *7ms (seek) + 3ms (rotlat) + 0.05 (xfer) + 6ms (rotate+xfer) = 16.05ms Since each write requires two disks: total throughput: 2*1/16.05ms = 124.6 req/sec*

(b) Using mirroring would provide higher throughput for writes. Give one reason why an administrator might choose to use RAID-5 anyway.

There are several possible answers, including:

- *Capacity might be a bigger concern than performance.*
- *The workload may have very few writes.*
- *The writes may be very large and aligned, avoiding the RMW overhead.*

Problem 3 : Distributed storage systems. [25 points]

(a) Direct-access SAN file systems come in two types: block-based and object-based. Which type involves more metadata being sent from the central file manager to clients? Explain.

Block-based. Block maps are bigger than object IDs, for anything other than very small files.

(b) The PVFS "Trove" Storage layer maintains two different types of storage for files: bstreams and keyvals. Identify what PVFS uses each to store and one implementation difference between them.

Bstreams: store file content. They are stored on disk as flat UNIX files and accessed through file-like read/write calls and/or list-io.

keyvals: store file metadata. They are stored on disk in Berkeley DB databases and accessed through key/value dictionary like lookup.

(c) Thin provisioning is a marketing term for a "virtual" storage system in which real storage is not allocated to an LBN range in the storage space unless data is actually written to that range. Reads to LBNs that have not been written return zeros without going to disk. The marketing folks tell customers that they can configure a file system to believe it has a single huge disk (e.g., 100TB or more) even if they buy only a few TBs. But, if an old-style file system written for plain old SCSI disks uses such virtual storage, it can run out of space after awhile even when there is less user data in the file system than physical storage capacity. (For example, a scan of the file system might reveal only 1 TB of file data on a system with 4 1TB disks, but the virtual storage system pretending to have 100TB reports that it is out of space.) Why can this problem occur? (Hint: Petal has a special operation to fix this problem.)

A traditional ("old-style") file system does not report file deletion to the underlying storage devices. So, once a LBN is written once, it will consume physical storage thereafter, even if the FS views it as "unallocated".

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

(a) Which guest speaker shared an office with Greg during graduate school?

Bruce Worthington

(b) Which guest speaker earned their Ph.D. at the same school as Garth?

Two possible answers: Kim Keeton, Brent Welch

(c) Which guest speaker worked at the same school Brandon is attending for graduate school?

Marc Unangst

(d) Given 10 slides of material, which instructor would you expect to talk longest? How long?

Noone knows, but a man could grow an impressive beard while sitting through the contest between Garth and Greg.