CS-537: Final Exam (Fall 2013)

The Worst Case

Please Read All Questions Carefully!

There are ten (10) total numbered pages.

Please put your NAME (mandatory) on THIS page, and this page only.

Name: _____

The Worst Case

"In any moment of decision,
the best thing you can do is the right thing,
the next best thing is the wrong thing,
and the worst thing you can do is nothing."
Theodore Roosevelt

Systems are designed to behave well in the **common case**. For example, most programs frequently re-use data and instructions; thus, hardware caches store frequently used data and instructions, and, in the common case, programs run faster as a result.

However, sometimes systems designed for the common case run up against something unpleasant, which we call **the worst case**. When the worst case occurs, all of the optimizations made for the common case don't work.

In this exam, we'll be studying worst-case behavior. Here is a simple example:

1. What is your worst-case score on this exam?

The answer here is zero. Perhaps we should provide a better example:

1. What is the worst-case number of memory accesses required to do a page-table lookup on a TLB miss on a system with a (two-level) multi-level page table?

The answer here is two: one for the page directory, and one for the page table entry itself.

The rest of the exam is like that, but probably harder. Good luck! And thanks for a fun semester – I enjoyed it; hope you did too.

1. Hard-disk Drives

Assume you have a hard-disk drive with these characteristics: 200 MB/s transfer rate, 9 millisecond (ms) **maximum** seek time, and 10,000 RPM speed of rotation.

(a) When reading a 4KB block, what is the worst-case time for this hard drive?

(b) What would you expect the **average** time to read a 4KB block to be?

(c) When reading a 2MB block, what is the worst-case time for this hard drive?

(d) What would you expect the **average** time to read a 2MB block to be?

(e) Now assume you have a series of **ten** 4KB reads to perform on the disk. What series of ten reads would you send to the drive to get the worst case performance? (Assume the disk scheduler is FIFO)

(f) How long will these ten requests take to complete?

(g) Now assume you have a disk scheduler that performs shortest-access-time-first (SATF) scheduling. What ten requests could you send to the drive to get the worst case performance, given that the disk will re-order requests as per SATF?

(h) How long will these ten requests take to complete?

2. RAIDs

Assume you have a RAID system with 4 disk drives; each disk drive has a peak transfer rate of **100 MB/s**, a worst-case seek time of **9 millisecond (ms)**, and a rotation rate of **15,000 RPM**. In this question, we'll explore some of the worst-case performance scenarios such a system could encounter.

Assume for all questions that the chunk size for the RAID is 4KB.

(a) Assume the RAID is configured to use RAID-0 (striping). Given a set of 4 4KB read requests, how would you pick the locations (e.g., [disk,offset] pairs) of the reads so as to get **worst-case** performance from this RAID?

(b) How long would it take to complete these 4 read requests?

(c) What is the **best case** time to complete 4 reads to a RAID-0 configured in this manner? (write down any assumptions you make)

(d) Now assume you have a mirrored (RAID-1) system. What is the worst-case time a single 4KB write takes?

(e) How long will 16 4KB writes take to complete on this 4-disk RAID-1? Assume that these writes are spread randomly across the RAID.

(f) We now have a RAID-4 (parity) based disk array, with one of the four disks used for parity. When a single-block write takes place, how many I/Os take place on the RAID in the worst case?

(g) How long will such a "small write" take, assuming the disk parameters above, and assuming a worst-case seek?

3. File and Directories

Files and directories provide a basic abstraction of persistent data to users. Here we explore (abstractly) how basic file systems work, focusing on **links**. Sometimes links lead to pretty odd performance problems.

(a) Assume we have a regular file that is referred to by the pathname `/a/b/c/orig.txt` – how many directories will we access when opening this file?

(b) Now assume we create a **hard link** to this file, as follows: `ln /a/b/c/orig.txt /hard.txt`. How many directories will we access when opening `/hard.txt`?

(c) Is the file `hard.txt` a special type of file? (explain)

(d) What happens to `hard.txt` when we delete `orig.txt`? Can we still access it? (explain)

(e) Now assume we create a **symbolic (soft) link** to this file, as follows: `ln -s /a/b/c/orig.txt /a/b/c/soft.txt`. How many directories will we access when opening `/a/b/c/soft.txt`?

(f) Is the file `/a/b/c/soft.txt` a special kind of file? (explain)

(g) What happens to `soft.txt` when we delete `orig.txt`? Can we still access it?

(h) Let's say we create a symbolic link to a parent directory, as follows: `ln -s /a/b/c /a/b/c/loop`. How many different pathnames can we use to refer to the file `orig.txt` in the directory `/a/b/c` ?

4. A Very Simple File System

The very simple file system (VSFS) uses basic data structures such as a super block, bitmaps, inodes, and data blocks to store all relevant file system information. Assume further a multi-level index within each inode, with 12 direct pointers, one indirect pointer, and one double-indirect pointer.

(a) You need to read a 4KB (one block) file, given its inode number. Assume no relevant structures are in memory in the file cache. How many disk accesses will the read take? (describe)

(b) If this same file is read frequently, how many disk accesses will typically occur? (with caching)

(c) Assume you need to append a block of size 4KB to a (small) file that already exists. Assume no relevant structures are in the file cache. How many disk accesses will the append take? (describe)

(d) If this small file is appended to frequently, how many disk accesses will typically occur? (with caching)

(e) Assume you need to append a block of size 4KB to a (large) file that already exists. Assume no relevant structures are in the file cache. How many disk accesses will the append take? (describe)

(f) Assume you **create** a file in the root directory (e.g., `/foo.txt`). How many disk writes will take place (in the worst case)?

(g) Assume you **delete** a file in the root directory (e.g., `/foo.txt`). How many disk writes will take place (in the worst case)?

5. The Fast File System

This question is about the Fast File System (FFS). The basic idea in FFS (the Fast File System) is to group related items near one another on disk, and similarly to spread unrelated items apart.

(a) FFS is divided into groups. What structures are found in each group?

(b) Given a disk of size N sectors, and a group size of G sectors (where G is much smaller than N), what is the worst-case distance (in sectors) between a file's inode and its **first** data block in FFS? (assume that no group is full)

(c) Again assume a disk of size N, and groups of size G. What is the worst-case distance (in sectors) between a file's inode and its **last** data block?

(d) Assume two files are in the **same** directory. What is the shortest distance possible between the first data block of one file and the first data block of the other?

(e) Assume two files are in the **different** directories. What is the shortest distance possible between the first data block of one file and the first data block of the other? (assume no group is full, and that the files are small)

(f) Assume a new, zero-sized file is created in an FFS-based file system. How many writes go to the disk to update FFS structures?

(g) How many of these writes are non-contiguous with the others (i.e., will require at least a short seek and rotation)?

6. Journaling File Systems

Journaling is used to aid in crash recovery. In this question, we'll figure out some of its worst-case behaviors.

(a) Assume we are using data journaling, where all updated blocks are first written to the journal. Describe the basic protocol a journaling file system uses to update file-system state.

(b) What happens if a crash occurs during this process? Are there cases where an update will be lost?

(c) Assume a workload that repeatedly picks a random file and appends a block to it; the files are scattered across the disk. Describe the **worst-case location** for the journal given such a workload.

(d) Now describe the **best-case location** for the journal given such a workload.

(e) Assuming a typical disk (100 MB/s transfer, 10 ms average seek, 10k RPM), estimate the performance difference between your best-case and worst-case journal locations.

(f) How much different would performance be if using metadata-only journaling for this workload?

7. The Network File System (NFS)

The Network File System (NFS) is an early example of a distributed file system. In this example, we'll explore some of its worst-case behaviors. Assume for all questions that in-client-memory block reads or writes take M time units, that checking something with the server takes C time units (e.g., a GETATTR request), and that reading or writing a block to a remote server takes R time units.

(a) A program creates a new file, writes to it, closes it, and re-opens the file and reads it. How long will the read take, in terms of M, C, and R?

(b) A program creates a new file, writes to it, closes it, waits a (long) time, and re-opens the file and reads it. How long will the read take, in terms of M, C, and R?

(c) A program creates a new file, writes to it, closes it, and deletes it. Why is this behavior inefficient in NFS?

(d) A program reads a very large file (larger than client memory) which consists of K blocks, and then re-reads the same file. How long does the re-read take, in terms of M, C, and R?

(e) AFS, the other distributed file system we studied, uses local disk to cache files. Describe a workload that brings out a worst-case performance scenario for AFS but does well on NFS.

8. Random Stuff

This is a random, exam-ending question. Enjoy!

(a) When using a **virtual machine monitor** with a software-managed TLB, what happens on a TLB miss?

(b) To speed up this worst-case TLB miss, some VMMs cache TLB entries. How can TLB entry caching within a VMM speed up TLB miss service time?

(c) Assume you are trying to figure out whether to poll a particular device or to use interrupts to get notified of device completion. When is one approach better than the other? (describe)

(d) Could there be a situation where **both** polling and interrupts should be used? (describe)

(e) What was the worst part of this exam? Most original answer gets an extra point.

(f) What was the best part? Funniest answer gets an extra point.