CMU 18-746	**Storage Systems**	**Assigned: 1 Feb 2010**
Spring 2010	**Homework 2**	**Due: 22 Feb 2010**

Solutions are due at the beginning of class on the due date and must be **typed** and neatly organized. Late homeworks will not be accepted. You are permitted to discuss these problems with your classmates; however, your work and answers must be your own. All questions should be directed to the teaching assistant.

Problem 1 : Write-based Skippy.

Before answering the following two problems, please read the following paper: *Microbenchmark-based Extraction of Local and Global Disk Characteristics* by Nisha Talagala, Remzi H. Arpaci-Dusseau, and David Patterson (Berkeley TR CSD-99-1063). It is available from the course website.

(a) First, describe in words the functionality of `lseek`. Specifically, what happens when lseek(fd, 1048576, SEEK_CUR) is issued?

(b) Refer to Figure 1, which shows the result of simulating the Skippy experiment on a disk simulator. Label this graph with the following values: rotational latency, MTM, sectors per track, number of heads, head switch time, and cylinder switch time. Also label the head and cylinder switches. Make sure you mark the parts of the figure that indicate these values.

(c) How would Figure 1 change, if the disk were replaced with one that has a higher rotation speed, but is the same in every other respect?

(d) If the WRITEs used in the Skippy experiment were replaced with READs, do you think the observed value for MTM would change? Explain your answer.

Problem 2 : Skippy variant.

Assume the skippy algorithm is replaced with the following:

```
fd = open(``raw disk device'');
for (i = 0; i < measurements; i++) {
    lseek(fd, 0, SEEK_SET);
    write(fd, buffer, SINGLE_SECTOR);

    //time the following write and output <i, time>
    //i is the hop size
    lseek(fd, i * SINGLE_SECTOR, SEEK_SET);
    write(fd, buffer, SINGLE_SECTOR);
}
close(fd);
```

Figure 2 shows the results of running this experiment (it is a plot of time vs. hop size) on a different disk than that used for problem 1. Note that these results were obtained experimentally and not via simulation, so the figure may contain a few anomalous datapoints.

(a) Explain why the above algorithm generates the results shown in the figure. Be sure to explain any minima, sudden jumps, etc. Include words such as MTM, track/cylinder skew, rotational latency, and head/cylinder switch time in your description. You do not need to disambiguate head and cylinder switches.

(b) Label the figure with the rotational latency and MTM. Make sure you mark the parts of the figure that indicate these values.

(c) Label the figure with the head/cylinder switch time. Do not worry about distinguishing head switches from cylinder switches — just mark the parts of the figure where head or cylinder switches occur and approximate their average time.

Problem 3 : Block mapping.

The Linux ext2 file system inode structure uses 12 direct blocks as well as one single indirect, one double indirect, and one triple indirect block. Assume blocks are 4 KB in size, and block pointers are unsigned 32-bit integer values. What is the maximum file size accessible:

(a) Using only the direct blocks?

(b) Using the direct and single indirect blocks?

(c) Using the direct, single, and double indirect blocks?

(d) Using the direct, single, double, and triple indirect blocks?

(e) Repeat (d), assuming blocks are only 1 KB in size. (Block size is chosen when a partition is formatted.)

(f) Imagine adding a "quadruple indirect" block. What is the maximum file size (for 4 KB blocks) using the direct, single, double, triple, and quadruple indirect blocks?

(g) Why are quadruple-indirect blocks irrelevant given the inode structure, block size, and block pointer size assumed in this problem?

An interesting artifact of the original ISO/ANSI C specification of the `fseek()` and `ftell()` system calls is that compliant operating systems were unable to address anywhere near the file size of (d). This is because these functions use the *long* type (a signed 32 bit integer) to specify the file position indicator (in bytes) for a stream.

(h) Because of the use of the *long* type, what is the maximum file size that can be handled by the ISO/ANSI C `fseek()` and `ftell()` system calls (and, therefore, the maximum file size that can be handled by compliant operating systems)?

Problem 4 : Extents.

Another technique to map data blocks from the inode is to use extent lists. An extent list is similar to the existing block pointers (see previous problem) except each entry is a vector:

- Pointer to the first block of the extent;
- Length of the extent (number of blocks following the first block).

Imagine replacing the 12 direct blocks in the ext2 inode with 6 extent descriptors, each 64 bits long (32 bits for the block pointer and 32 bits for the extent length). Assume a 512-byte block size.

(a) What is the largest file addressible using these 6 extent descriptors?

(b) What is the smallest file possible using all 6 descriptors?

(c) Are indirect blocks still necessary when extent lists are used? Why or why not?

Problem 5 : Those that do not learn from the mistakes of history are doomed to repeat them.

Back in the days when Bill Gates was only moderately rich and "high-end" hard disk capacities were perhaps tens of megabytes, the INT 13 interface (BIOS software interrupt 0x13) was used by operating systems to send commands to hard disks. The INT 13 interface uses 24 bits to specify 512-byte sector locations based on physical disk geometry:

- 10 bits for cylinder number (up to 1024 cylinders),
- 8 bits for head number (up to 256 heads/cylinder),
- 6 bits for sector number (up to 63 sectors/track: sectors are numbered 1–63, omitting zero).

(a) What is the largest disk capacity (in sectors and GB) addressable through the INT 13 interface? Note: use 1 GB = 2^{30} B.

In a similar manner, the ATA/ATAPI-5 (what used to be IDE) interface defines 28 bits for addressing sectors based on the physical disk geometry:

- 16 bits for cylinder number (up to 65,536 cylinders),
- 4 bits for head number (up to 16 heads/cylinder),
- 8 bits for sector number (up to 255 sectors/track; zero is not used).

Notice that these bit fields vary significantly in size from the INT 13 specification. This was caused by poor communication and cooperation among the standards bodies, and resulted (around 1994) in the first of many embarassing limitations on disk addressing. When combined, each of the cylinder/head/sector addresses are truncated:

Standard	Bits for cylinder number	Bits for head number	Bits for sector number	Total bits for address
INT 13	**10**	8	**6**	24
ATA	16	**4**	8	28
Combined	**10**	**4**	**6**	20

(b) What is the largest disk capacity addressable through the INT 13 interface using ATA hard disks? (Recall that sector zero is unused in both interfaces.)

A short-term fix for these problems is logical block addressing (LBA) on the ATA bus combined with the "INT 13 extensions" described below. With LBA addressing, the BIOS no longer sends a physical cylinder/head/sector address. Instead, all available address bits are combined and used to send a logical address, with the logical-to-physical conversion done by the disk firmware.

(c) What is the largest disk capacity addressable using LBA over ATA?

(d) Maxtor Corporation released the DiamondMax 80 in August 2000. This ATA disk contains 160,086,528 sectors (76.335 GB). Assuming disk capacity growth continues present trends (doubling every 12 months), in what month of what year will ATA disks no longer be addressable using LBA over ATA?

Now isn't that interesting? In 1998 the ATA standards committee (NCITS Technical Committee T13) began evaluating options for increasing the LBA address to a 48- or 64-bit value. 48-bit LBA extensions were officially added to the developing ATA-6 standard in October 2000; however, the ATA-6 standard has not yet been standardized.

(e) What is the largest disk capacity addressable using 48-bit LBA over ATA-6?

(f) Assuming disk capacity growth continues present trends as in (d), in what year will 48-bit LBA no longer be sufficient?

To overcome the BIOS limitation of (a), the INT 13 interface was changed to use the "INT 13 extensions." Instead of the OS passing a cylinder/head/sector address through the INT 13 call, the OS passes a physical memory address pointing to a 64-bit logical block address.

(g) What is the largest disk capacity addressable using the INT 13 extensions?

(h) Assuming disk capacity growth continues present trends as in (d), in what year will the INT 13 extensions no longer be sufficient?

Problem 6 : I/O system design.

You have been given a number of individual components with which you have been asked to build a large storage system. However, there are some tradeoffs between the price and performance of the components and your company wants to achieve the greatest performance possible at the lowest cost.

Given the following components with their performance information:

- 3 GHz CPU
- 16 byte wide memory with a 50 ns cycle time
- 33 MHz/32-bit PCI bus with up to 8 SCSI controllers
- SCSI controller adds 0.2 ms overhead per I/O, 1 bus/controller, cost $350
- SCSI bus 7 disks/bus, bus @ 320 MB/s
- OS with 10,000 CPU cycles per disk I/O
- large disk: 146 GB
- small disk: 36 GB
- both disks are 10,000 RPM, 6 ms avg seek, stream at 25 MB/s, and cost $4/GB
- avg I/O size of 8 KB

Make the following assumptions when answering the questions below:

- Every disk I/O requires an average seek and average rotational delay (of half a rotation).

- All devices are used at 100% capacity and that the workload is evenly distributed between all the disks.

(a) Give the breakdown of the max I/O operations per second (IOPS) for each component (CPU, memory, I/O bus, a SCSI controller, a disk).

(b) In a fully configured system (a system with the max number of disks controllers) which components are under-utilized? Which components will be a bottleneck?

(c) For a 2 TB database what is the optimal configuration of the system? Find highest performance first and then lowest cost for that performance level (in terms of disks and controllers).

(d) Given the configuration in Part (c), you are told you can upgrade one type of component. Given the following choices which is the best component to upgrade:

  - new disks (same sizes) but 15,000 RPM, 4 ms avg seek, stream at 30 MB/s
  - new PCI I/O backplane 66 MHz/64-bit
  - new 1000 MIPS processor

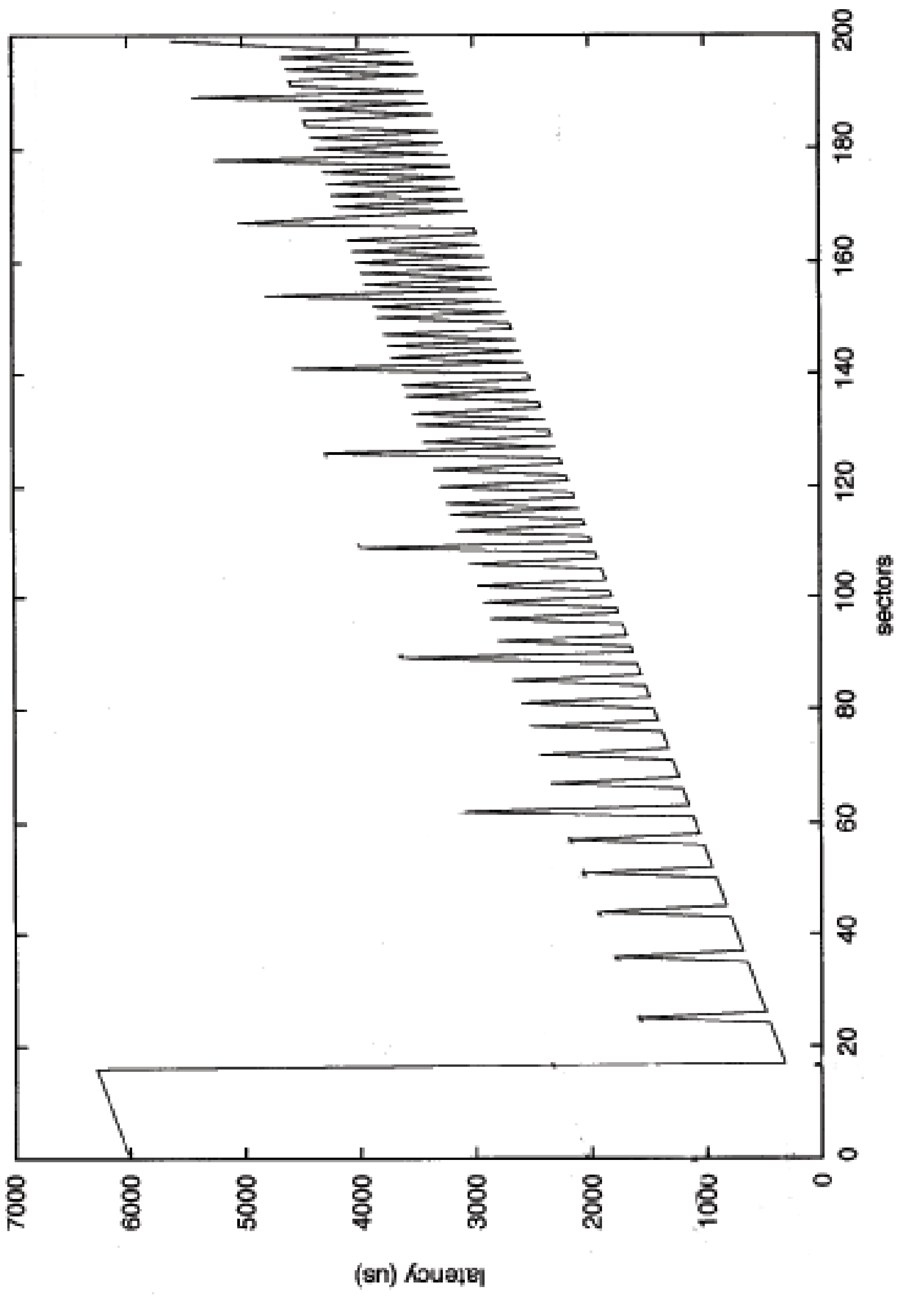

Figure 1: *Skippy results for problem 1*

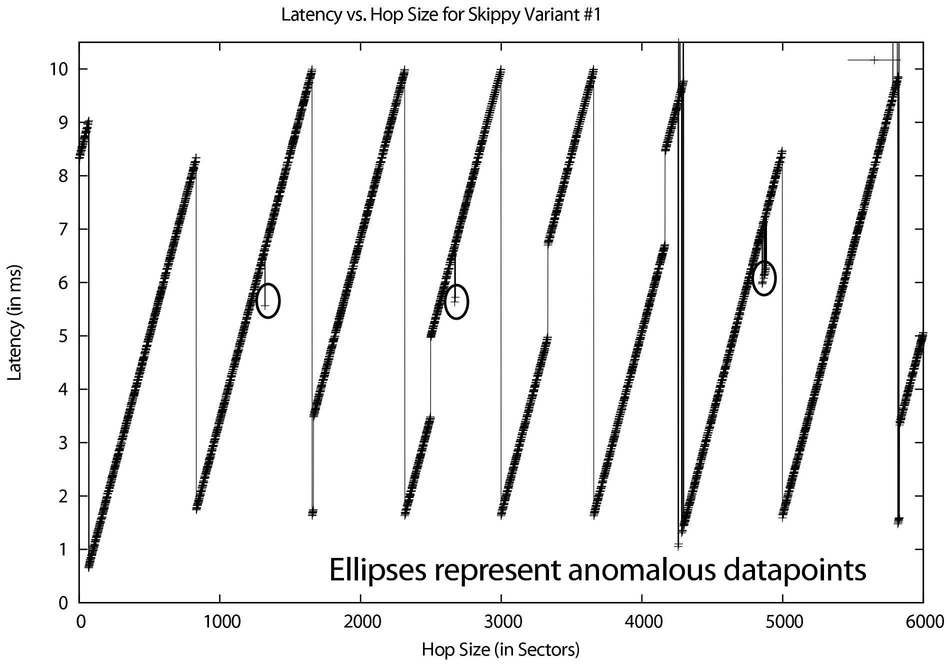

Figure 2: *Skippy variant results for problem 2*