9 File Systems [10 points]

Consider a disk with a mean seek time of 10 msec, a rotational rate of 12,000 rpm and 200 KB per track. You are designing a file system, and you have chosen the block size to be 4 KB. Suppose file A is created on the disk and its length is 40KB. Your block allocation algorithm is smart, and it has allocated contiguous blocks on the same track to the file.

Part A [2 points]

How long will it take (on average) to read the first block of the file? Show your calculation.

Part B [2 points]

How long will it take (on average) to read the entire file sequentially? Show your calculation.

Question continues ...

Part C [4 points]

Now suppose that the application reads the file in a back-and-forth order as follows: first block, last block, second block, second-last block, third block, third-last block, ..., until it reads all the blocks of the file. How long will it take (on average) to read the entire file? Show your calculation, and if you make any assumptions, state them.

Part D [2 points]

Can you think of a way to improve the performance of your file system for the back-and-forth reading of the file (see Part C)? Hint: a new file-related system call.

5. My App Is Slow [12 marks]

You look up the specifications for the low-end disk drive on your machine. It spins at 7500 RPM, has an average seek time of 6.5 ms and a data transfer rate of 20 MB/s. Each sector on the disk has 512 bytes of user data.

A) [3 marks] On average, how long does it take to read a disk block, consisting of 8 contiguous sectors, when the starting sector is chosen at random? Show your calculations.

1. seek time: 6.5ms
2. rotational delay for half a rotation (on average): 7500 rotations / min = 7500/60 rotations/sec = 125 rotations/sec, 1/2 rotation in 1/250 second = 4ms
3. transfer time for 4KB: 20 MB/s, 4KB/20MB * 1000 ms = 1000/(5 * 1024) ~= 0.2 ms

total time = 10.7 ms

B) [3 marks] Suppose that the OS maintains a 1 MB disk block cache in memory. The data transfer rate from this cache to another location in memory (e.g., to user space) is 160 MB/s. Suppose that the hit rate for this cache is 99%. What is the expected time to read a randomly chosen disk block? Show your calculations.

expected time = 0.99 * 0.2/8 + 0.01 * 10.7 = 0.99 * 0.025 + 0.107 ~= 0.132 ms

C) [3 marks] Your machine has 1 GB of memory. To increase the cache hit ratio, you configure the disk block cache in the OS to use much larger than 1MB of memory. To your surprise, you find that several of your applications run much slower than faster. Give two different example scenarios why this might be the case.

Since the disk block cache is very large, it reduces the amount of memory that can be allocated to the page cache. That will affect 1) applications that have very large code segments, 2) applications that have very large data segments, e.g., access large array. In both cases, there may be additional page eviction and thus additional swapping, making applications slower.

D) [3 marks] You decide to buy a new computer because you would really like to speed up your applications. You contact a trusted friend, H. Geek, who is hardware savvy. He says that your best bet is to buy a computer with a high-end disk that has the same specifications as your current low-end disk but spins at 15000 RPM. The disk costs twice as much as your disk. You get a second opinion from a hardware shop on College St. They suggest buying a new computer with two low-end disks and connecting them together using a RAID-0 software driver. Which option would you choose? Justify your choice, quantitatively if possible.

Option 1:
random reads: 6.5 + 2 + 0.1 = 8.6 ms (improves slightly)
sequential reads: 40 MB/s (assuming that the transfer rate doubles)

Option 2:
random reads: 6.5 + 4 + 0.2 = 10.7 ms (no change from single disk)
sequential reads: 40 MB/s (both disks can be read simultaneously)

Price: same.
In this case, Option 1 is slightly better.

Question 8. Disks [10 MARKS]

Consider a disk with a mean seek time of 10 msec, a rotational rate of 12,000 rpm and 200 KB per track. You are designing a file system, and you have chosen the block size to be 4 KB. Suppose file A is created on the disk and its length is 40KB. Your block allocation algorithm is smart, and it has allocated contiguous blocks on the same track to the file.

Part (a) [2 MARKS] How long will it take (on average) to read the first block of the file? Show your calculation.

1. Average seek time = 10ms
2. Time to read the track = 60 s/12000 = 5 ms

   Time to read half the track (on average) = 2.5 ms
3. 1 track has 200KB/4KB = 50 blocks

   So time to read one block = 5/50 = 0.1 ms
4. So average time to read the first block of the file = 10 + 2.5 + 0.1 ms = 12.6 ms

Part (b) [2 MARKS] How long will it take (on average) to read the entire file sequentially? Show your calculation.

10ms (average seek time) + 2.5 ms (average rotational latency) + 10 * 0.1 ms = 13.5 ms

Part (c) [4 MARKS] Now suppose that the application reads the file in a back-and-forth order as follows: first block, last block, second block, second-last block, third block, third-last block, ..., until it reads all the blocks of the file. How long will it take (on average) to read the entire file? Show your calculation, and if you make any assumptions, state them.

1. Average seek time = 10 ms
2. Average rotational latency = 2.5 ms
3. Time to read block #1 = 0.1 ms
4. Time to read block #10 + get to the beginning of block #2 = 5 ms (rotational latency)

   Time to read block #2 = 0.1 ms

   This step is done 4 times (to read (#2, #10), (#3, #9), (#4, #8), (#5, #9) blocks)
5. Time to read block# 6 = 0.1 ms

Total time = 10 + 2.5 + 0.1 + 5.1 * 4 + 0.1 = 33.1 ms

Part (d) [2 MARKS] Can you think of a way to improve the performance of your file system for the back-and-forth reading of the file (see Part C)? Hint: a new file-related system call.

Assuming that the application knows that it will be reading blocks in the order #1, #10, #2, #9, ..., it can use a system call to inform the OS that is will be reading in this order before it starts reading the first block. The OS can then use the disk scheduling algorithm to order the requests and prefetch them sequentially in 13.5 ms (see Part 2).

Question 7. Disks [12 MARKS]

The shortest-access-time-first (SATF) scheduler is a variant of the shortest-seek-first (SSF) algorithm. It picks the disk IO request from the IO scheduler queue that has the shortest access time from the current head position and services this request. In this question, we'll perform some simple calculations on a highly-simplified disk.

Part (a) [3 MARKS] Assume a simple disk that has only a single track, and a simple FIFO scheduling policy. The rotational delay on this disk is R. There is no seek cost (only one track!), and transfer time is so fast that we just consider it to be free. What is the (approximate) worst case execution time for three (3) requests (to different blocks)?

3R, each request takes roughly R time.

Part (b) [3 MARKS] Now assume that a SATF scheduler is being used by the disk (but it still only has a single track). What is the worst-case time for three requests (to different blocks) now?

R, in one rotation, all the three requests can be served.

Some students wrote that each request could come right after the previous one was served and so the worst case execution time would be 3R. We accepted that answer.

Part (c) [3 MARKS] Now assume the disk has three tracks. The time to seek between two adjacent tracks is S; it takes twice that to seek across two tracks (e.g., from the outer to the inner track). Given a FIFO scheduler, what is the worst-case time for three requests?

3 * (2S + R), each request takes 2S + R time.

Part (d) [3 MARKS] The same question above, but for a SATF scheduler.

3S + 3R. In the worst case, there will be three seeks required and 3 rotations to get to all the requests. The three seeks will occur when the disk head is in the center track, and the head needs to move to out the outer and then the inner track.

We also accepted answers like 4S + R (in one rotation, get to all the blocks by seeking).

We also accepted 3 * (2S + R) if the answer mentioned that each request could come right after the previous one was served.

Question 7. RAID Regained [8 MARKS]

You have a RAID-4 device (parity-based RAID + a single parity disk) with 5 disks and a 4KB chunk size, as shown below:

```
Disk-0    Disk-1    Disk-2    Disk-3    Disk-4
block0    block1    block2    block3    parity(0..3)
block4    block5    block6    block7    parity(4..7)
```

You are not sure that writes to the RAID device are working correctly. You write the following skeleton code for writing to the RAID device. Fill in the rest of the code in the empty boxes to ensure that writes work correctly.

```
/*
 * This write() routine takes a logical block number (block), as shown above,
 * and writes 4KB (data) to the device.
 *
 * It uses the underlying primitives:
 * read(int disk, int block_nr, char *data)
 * write(int disk, int block_nr, char *data)
 * xor(char *d1, char *d2, char *d3) ;; xor d1 and d2, place result in d3
 */

void write(int block, char *data)
{
  char buf[4096];
  char parity[4096];
  int disk = [block % 4];
  int phy_block_nr = [block / 4];

  [read](disk, phy_block_nr, [buf]);
  read(4, phy_block_nr, parity);
  xor([parity], [buf], [parity]);
  xor(parity, data, parity);
  [write](disk, phy_block_nr, [data]);
  write(4, phy_block_nr, parity);
}
```