Secondary-Storage Structure

In this chapter we describe the internal data structures and algorithms used by the operating system to implement the file system. We also discuss the lowest level of the file system, the secondary storage structure. We first describe disk-head-scheduling algorithms. Next we discuss disk formatting and management of boot blocks, damaged blocks, and swap space. We end with coverage of disk reliability and stable storage.

The basic implementation of disk scheduling should be fairly clear: requests, queues, servicing; so the main new consideration is the actual algorithms: FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK. Simulation may be the best way to involve the student with the algorithms. Exercise 12.17 provides a question amenable to a small but open-ended simulation study.

The paper by Worthington et al. [1994] gives a good presentation of the disk-scheduling algorithms and their evaluation. Be suspicious of the results of the disk-scheduling papers from the 1970s, such as Teory and Pinkerton [1972], because they generally assume that the seek time function is linear, rather than a square root. The paper by Lynch [1972b] shows the importance of keeping the overall system context in mind when choosing scheduling algorithms. Unfortunately, it is fairly difficult to find.

Chapter 2 introduced the concept of primary, secondary, and tertiary storage. In this chapter, we discuss tertiary storage in more detail. First, we describe the types of storage devices used for tertiary storage. Next, we discuss the issues that arise when an operating system uses tertiary storage. Finally, we consider some performance aspects of tertiary storage systems.

12.1 a. Disks have fast random-access times, so they give good performance for interleaved access streams. By contrast, tapes have high positioning times. Consequently, if two users attempt to share a tape drive for reading, the drive will spend most of its time switching tapes and positioning to the desired data, and relatively little time performing data transfers. This performance problem is similar to the thrashing of a virtual memory system that has insufficient physical memory.

b. Tape cartridges are removable. The owner of the data may wish to store the cartridge off-site (far away from the computer) to keep a copy of the data safe from a fire at the location of the computer.

c. Tape cartridges are often used to send large volumes of data from a producer of data to the consumer. Such a tape cartridge is reserved for that particular data transfer and cannot be used for general-purpose shared storage space.

To support shared file-system access to a tape jukebox, the operating system would need to perform the usual file-system duties, including

a. Manage a file-system name space over the collection of tapes

b. Perform space allocation

c. Schedule the I/O operations

The applications that access a tape-resident file system would need to be tolerant of lengthy delays. For improved performance, it would be desirable for the applications to be able to disclose a large number of I/O operations so that the tape-scheduling algorithms could generate efficient schedules.

12.2 Tapes are easily removable, so they are useful for off-site backups and for bulk transfer of data (by sending cartridges). As a rule, a magnetic hard disk is not a removable medium.

12.3 Truly stable storage would never lose data. The fundamental technique for stable storage is to maintain multiple copies of the data, so that if one copy is destroyed, some other copy is still available for use. But for any scheme, we can imagine a large enough disaster that all copies are destroyed.

12.4 The area of a 5.25-inch disk is about 19.625 square inches. If we suppose that the diameter of the spindle hub is 1.5 inches, the hub occupies an area of about 1.77 square inches, leaving 17.86 square inches for data storage. Therefore, we estimate the storage capacity of the optical disk to be 2.2 gigabytes.

The surface area of the tape is 10,800 square inches, so its storage capacity is about 26 gigabytes.

If the 10,800 square inches of tape had a storage density of 1 gigabit per square inch, the capacity of the tape would be about 1,350 gigabytes, or 1.3 terabytes. If we charge the same price per gigabyte for the optical tape as for magnetic tape, the optical tape cartridge would cost about 50 times more than the magnetic tape; that is, $1,250.

12.5 First let's consider the pure disk system. One terabyte is 1024 GB. To be correct, we need 103 disks at 10 GB each. But since this question is about approximations, we will simplify the arithmetic by rounding off the numbers. The pure disk system will have 100 drives. The cost of the disk drives would be $100,000, plus about 20% for cables, power supplies, and enclosures; that is, around $120,000. The aggregate data rate would be 100×5 MB/s, or 500 MB/s. The average waiting time depends on the workload. Suppose that the requests are for transfers

of size 8 KB, and suppose that the requests are randomly distributed over the disk drives. If the system is lightly loaded, a typical request will arrive at an idle disk, so the response time will be 15 ms access time plus about 2 ms transfer time. If the system is heavily loaded, the delay will increase, roughly in proportion to the queue length.

Now let's consider the hierarchical storage system. The total disk space required is 5% of 1 TB, which is 50 GB. Consequently, we need 5 disks, so the cost of the disk storage is $5,000 (plus 20%; that is, $6,000). The cost of the 950 GB tape library is $9500. Thus the total storage cost is $15,500. The maximum total data rate depends on the number of drives in the tape library. We suppose there is only one drive. Then the aggregate data rate is 6×10 MB/s; that is, 60 MB/s. For a lightly loaded system, 95% of the requests will be satisfied by the disks with a delay of about 17 ms. The other 5% of the requests will be satisfied by the tape library, with a delay of slightly more than 20 seconds. Thus the average delay will be $(95 \times 0.017 + 5 \times 20)/100$, or about 1 second. Even with an empty request queue at the tape library, the latency of the tape drive is responsible for almost all of the system's response latency, because 1/20th of the workload is sent to a device that has a 20-second latency. If the system is more heavily loaded, the average delay will increase in proportion to the length of the queue of requests waiting for service from the tape drive.

The hierarchical system is much cheaper. For the 95% of the requests that are served by the disks, the performance is as good as a pure-disk system. But the maximum data rate of the hierarchical system is much worse than for the pure-disk system, as is the average response time.

12.6 Truly stable storage would never lose data. The fundamental technique for stable storage is to maintain multiple copies of the data, so that if one copy is destroyed, some other copy is still available for use. But for any scheme, we can imagine a large enough disaster that all copies are destroyed.

12.7 RAID Level 1 organization can perform writes by simply issuing the writes to mirrored data concurrently. RAID Level 5, on the other hand, would require the old contents of the parity block to be read before it is updated based on the new contents of the target block. This results in more overhead for the write operations on a RAID Level 5 system.

12.8

a. New requests for the track over which the head currently resides can theoretically arrive as quickly as these requests are being serviced.

b. All requests older than some predetermined age could be "forced" to the top of the queue, and an associated bit for each could be set to indicate that no new request could be moved ahead of these requests. For SSTF, the rest of the queue would have to be reorganized with respect to the last of these "old" requests.

c. To prevent unusually long response times.

d. Paging and swapping should take priority over user requests. It may be desirable for other kernel-initiated I/O, such as the writing

of file system metadata, to take precedence over user I/O. If the kernel supports real-time process priorities, the I/O requests of those processes should be favored.

12.9 a. The FCFS schedule is 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. The total seek distance is 7081.

b. The SSTF schedule is 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774. The total seek distance is 1745.

c. The SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 130, 86. The total seek distance is 9769.

d. The LOOK schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 130, 86. The total seek distance is 3319.

e. The C-SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 86, 130. The total seek distance is 9813.

f. (Bonus.) The C-LOOK schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 86, 130. The total seek distance is 3363.

12.10 a. 7200 rpm gives 120 rotations per second. Thus, a full rotation takes 8.33 ms, and the average rotational latency (a half rotation) takes 4.167 ms.

b. Solving $t = 0.7561 + 0.2439\sqrt{L}$ for $t = 4.167$ gives $L = 195.58$, so we can seek over 195 tracks (about 4% of the disk) during an average rotational latency.

12.11 a. Solving $d = \frac{1}{2}at^2$ for t gives $t = \sqrt{(2d/a)}$.

b. Solve the simultaneous equations $t = x + y\sqrt{L}$ that result from $(t = 1, L = 1)$ and $(t = 18, L = 4999)$ to obtain $t = 0.7561 + 0.2439\sqrt{L}$.

c. The total seek times are: FCFS 65.20; SSTF 31.52; SCAN 62.02; LOOK 40.29; C-SCAN 62.10 (and C-LOOK 40.42). Thus, SSTF is fastest here.

d. $(65.20 - 31.52)/65.20 = 0.52$. The percentage speedup of SSTF over FCFS is 52%, with respect to the seek time. If we include the overhead of rotational latency and data transfer, the percentage speedup will be less.

12.12 First, calculate performance of the device. 6000×6000 bits per millisecond = 4394 KB per millisecond = 4291 MB/sec(!). Clearly this is orders of magnitude greater than current hard disk technology, as the best production hard drives do less than 40 MB/sec. The following answers assume that the device cannot store data in smaller chunks than 4 MB.

a. This device would find great demand in the storage of images, audio files, and other digital media.

b. Assuming that interconnection speed to this device would equal its throughput ability (that is, the other components of the system could keep it fed), large-scale digital load and store performance would be greatly enchanced. Manipulation time of the digital

object would stay the same, of course. The result would be greatly enhanced overall performance.

c. Currently, objects of that size are stored on optical media, tape media, and disk media. Presumably, demand for those would decrease as the holographic storage became available. There are likely to be uses for all of those media even in the presence of holographic storage, so it is unlikely that any would become obsolete. Hard disks would still be used for random access to smaller items (such as user files). Tapes would still be used for off-site archiving, and disaster-recovery uses, and optical disks (CDRW for instance) for easy interchange with other computers, and low-cost bulk storage.

Depending on the size of the holographic device, and its power requirements, it would also find use in replacing solid-state memory for digital cameras, MP3 players, and hand-held computers.

12.13 Please refer to the supporting Web site for source code solution.

12.14 The center of the disk is the location having the smallest average distance to all other tracks. Thus the disk head tends to move away from the edges of the disk. Here is another way to think of it. The current location of the head divides the cylinders into two groups. If the head is not in the center of the disk and a new request arrives, the new request is more likely to be in the group that includes the center of the disk; thus, the head is more likely to move in that direction.

12.15 For 8 KB random I/Os on a lightly loaded disk, where the random access time is calculated to be about 13 ms, the effective transfer rate is about 615 MB/s. In this case, 15 disks would have an aggregate transfer rate of less than 10 MB/s, which should not saturate the bus. For 64 KB random reads to a lightly loaded disk, the transfer rate is about 3.4 MB/s, so five or fewer disk drives would saturate the bus. For 8 KB reads with a large enough queue to reduce the average seek to 3 ms, the transfer rate is about 1 MB/s, so the bus bandwidth may be adequate to accommodate 15 disks.

12.16 a. 750,000 drive-hours per failure divided by 1000 drives gives 750 hours per failure—about 31 days or once per month.

b. The human-hours per failure is 8760 (hours in a year) divided by 0.001 failure, giving a value of 8,760,000 "hours" for the MTBF. 8760,000 hours equals 1000 years. This tells us nothing about the expected lifetime of a person of age 20.

c. The MTBF tells nothing about the expected lifetime. Hard disk drives are generally designed to have a lifetime of five years. If such a drive truly has a million-hour MTBF, it is very unlikely that the drive will fail during its expected lifetime.

12.17 There is no simple analytical argument to answer the first part of this question. It would make a good small simulation experiment for the students. The answer can be found in Figure 2 of Worthington et al. [1994]. (Worthington et al. studied the LOOK algorithm, but similar

results obtain for SCAN.) Figure 2 in Worthington et al. shows that C-LOOK has an average response time just a few percent higher than LOOK but that C-LOOK has a significantly lower variance in response time for medium and heavy workloads. The intuitive reason for the difference in variance is that LOOK (and SCAN) tend to favor requests near the middle cylinders, whereas the C-versions do not have this imbalance. The intuitive reason for the slower response time of C-LOOK is the "circular" seek from one end of the disk to the farthest request at the other end. This seek satisfies no requests. It causes only a small performance degradation because the square-root dependency of seek time on distance implies that a long seek isn't terribly expensive by comparison with moderate-length seeks.

For the second part of the question, we observe that these algorithms do not schedule to improve rotational latency; therefore, as seek times decrease relative to rotational latency, the performance differences between the algorithms will decrease.

12.18 a. Assume that a disk drive holds 10 GB. Then 100 disks hold 1 TB, 100,000 disks hold 1 PB, and 100,000,000 disks hold 1 EB. To store 4 EB would require about 400 million disks. If a magnetic tape holds 40 GB, only 100 million tapes would be required. If an optical tape holds 50 times more data than a magnetic tape, 2 million optical tapes would suffice. If a holographic cartridge can store 180 GB, about 22.2 million cartridges would be required.

b. A 3.5″ disk drive is about 1″ high, 4″ wide, and 6″ deep. In feet, this is 1/12 by 1/3 by 1/2, or 1/72 cubic feet. Packed densely, the 400 million disks would occupy 5.6 million cubic feet. If we allow a factor of two for air space and space for power supplies, the required capacity is about 11 million cubic feet.

c. A 1/2″ tape cartridge is about 1″ high and 4.5″ square. The volume is about 1/85 cubic feet. For 100 million magnetic tapes packed densely, the volume is about 1.2 million cubic feet. For 2 million optical tapes, the volume is 23,400 cubic feet.

d. A CD-ROM is 4.75″ in diameter and about 1/16″ thick. If we assume that a holostore disk is stored in a library slot that is 5″ square and 1/8″ wide, we calculate the volume of 22.2 million disks to be about 40,000 cubic feet.

12.19 a) A write of one block of data requires the following: read of the parity block, read of the old data stored in the target block, computation of the new parity based on the differences between the new and old contents of the target block, and write of the parity block and the target block. b) Assume that the seven contiguous blocks begin at a four-block boundary. A write of seven contiguous blocks of data could be performed by writing the seven contiguous blocks, writing the parity block of the first four blocks, reading the eight block, computing the parity for the next set of four blocks and writing the corresponding parity block onto disk.

12.20 While allocating blocks for a file, the operating system could allocate blocks that are geometrically close by on the disk if it had more information regarding the physical location of the blocks on the disk. In particular, it could allocate a block of data and then allocate the second block of data in the same cylinder but on a different surface at a rotationally optimal place so that the access to the next block could be made with minimal cost.

12.21 Most disks do not export their rotational position information to the host. Even if they did, the time for this information to reach the scheduler would be subject to imprecision and the time consumed by the scheduler is variable, so the rotational position information would become incorrect. Further, the disk requests are usually given in terms of logical block numbers, and the mapping between logical blocks and physical locations is very complex.

12.22 To achieve the same areal density as a magnetic disk, the areal density of a tape would need to improve by two orders of magnitude. This would cause tape storage to be much cheaper than disk storage. The storage capacity of a tape would increase to more than 1 terabyte, which could enable a single tape to replace a jukebox of tapes in today's technology, further reducing the cost. The areal density has no direct bearing on the data transfer rate, but the higher capacity per tape might reduce the overhead of tape switching.

12.23 Frequently updated data need to be stored on RAID Level 1 disks while data that is more frequently read as opposed to being written should be stored in RAID Level 5 disks.

12.24 A system can perform only at the speed of its slowest bottleneck. Disks or disk controllers are frequently the bottleneck in modern systems as their individual performance cannot keep up with that of the CPU and system bus. By balancing I/O among disks and controllers, neither an individual disk nor a controller is overwhelmed, so that bottleneck is avoided.

12.25

a. SSTF would take greatest advantage of the situation. FCFS could cause unnecessary head movement if references to the "high-demand" cylinders were interspersed with references to cylinders far away.

b. Here are some ideas. Place the hot data near the middle of the disk. Modify SSTF to prevent starvation. Add the policy that if the disk becomes idle for more than, say, 50 ms, the operating system generates an *anticipatory seek* to the hot region, since the next request is more likely to be there.

c. Cache the metadata in primary memory, and locate a file's data and metadata in close physical proximity on the disk. (UNIX accomplishes the latter goal by allocating data and metadata in regions called *cylinder groups*.)

12.26 Sector sparing can cause an extra track switch and rotational latency, causing an unlucky request to require an extra 8 ms of time. Sector slipping has less impact during future reading, but at sector remapping time it can require the reading and writing of an entire track's worth of data to slip the sectors past the bad spot.

12.27 Since the disk holds 22,400,000 sectors, the probability of requesting one of the 100 remapped sectors is very small. An example of a worst-case event is that we attempt to read, say, an 8 KB page, but one sector from the middle is defective and has been remapped to the worst possible location on another track in that cylinder. In this case, the time for the retrieval could be 8 ms to seek, plus two track switches and two full rotational latencies. It is likely that a modern controller would read all the requested good sectors from the original track before switching to the spare track to retrieve the remapped sector and thus would incur only one track switch and rotational latency. So the time would be 8 ms seek + 4.17 ms average rotational latency + 0.05 ms track switch + 8.3 ms rotational latency + 0.83 ms read time (8 KBis 16 sectors, 1/10 of a track rotation). Thus, the time to service this request would be 21.8 ms, giving an I/O rate of 45.9 requests per second and an effective bandwidth of 367 KB/s. For a severely time-constrained application this might matter, but the overall impact in the weighted average of 100 remapped sectors and 22.4 million good sectors is nil.

12.28 a) The amount of throughput depends on the number of disks in the RAID system. A RAID Level 5 comprising of a parity block for every set of four blocks spread over five disks can support four to five operations simultaneously. A RAID Level 1 comprising of two disks can support two simultaneous operations. Of course, there is greater flexibility in RAID Level 1 as to which copy of a block could be accessed and that could provide performance benefits by taking into account position of disk head. b) RAID Level 5 organization achieves greater bandwidth for accesses to multiple contiguous blocks since the adjacent blocks could be simultaneously accessed. Such bandwidth improvements are not possible in RAID Level 1.

12.29 Yes, a RAID Level 1 organization could achieve better performance for read requests. When a read operation is performed, a RAID Level 1 system can decide which of the two copies of the block should be accessed to satisfy the request. This choice could be based on the current location of the disk head and could therefore result in performance optimizations by choosing a disk head that is closer to the target data.