OPERATING SYSTEMS

Topic – Disk Scheduling

Physical View of Magnetic Disk

Magnetic disks provide the bulk of secondary storage for modern computer systems.

Each disk platter has a flat circular shape, like a CD. Common platter diameters range from 1.8 to 3.5 inches. The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.

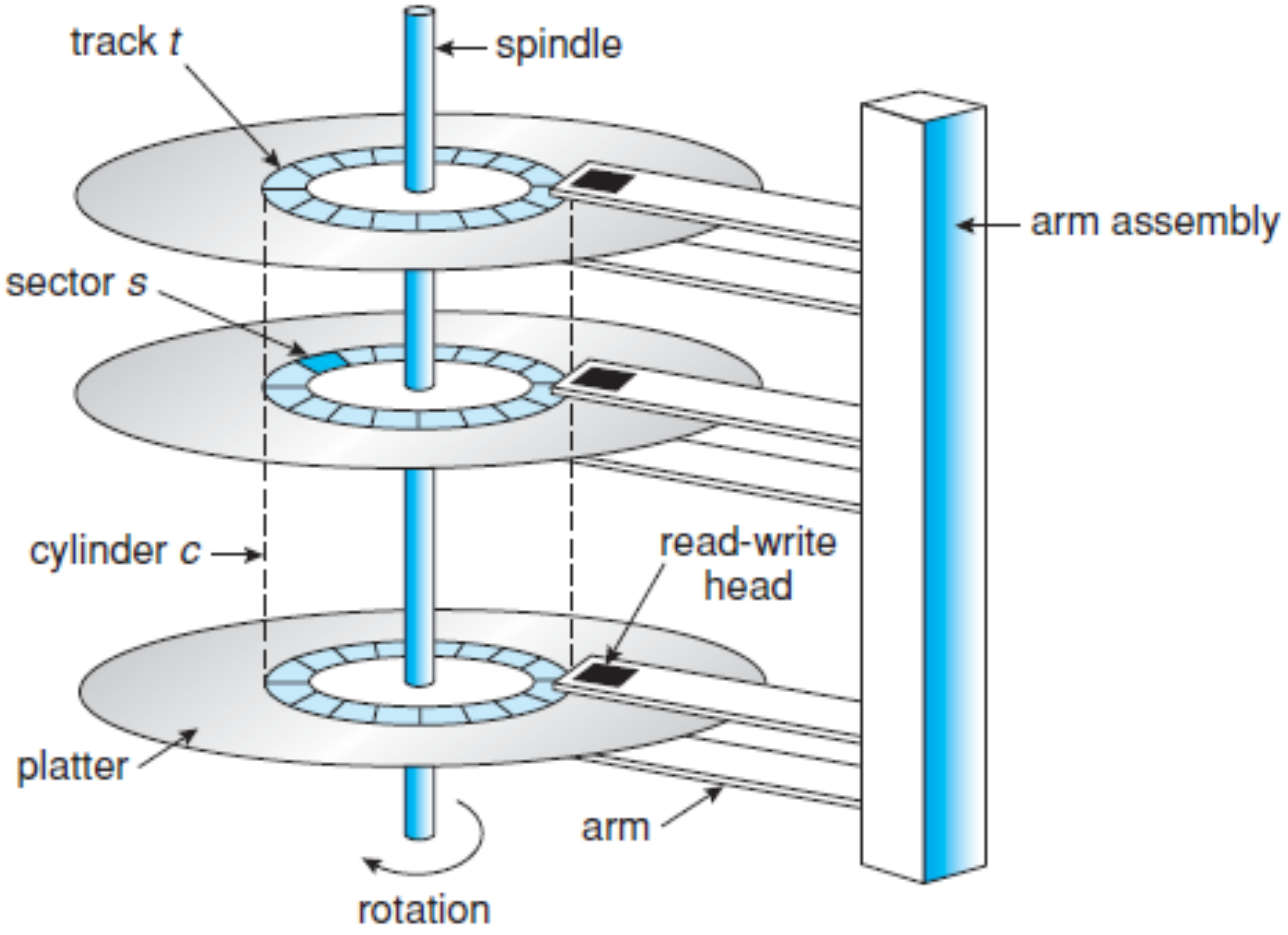
A read–write head “flies” just above each surface of every platter. The heads are attached to a disk arm that moves all the heads as a unit.

The surface of a platter is logically divided into circular tracks, which are subdivided into sectors.

The set of tracks that are at one arm position makes up a cylinder.

There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors. When the disk is in use, a drive motor spins it at high speed. Common drives spin at 5,400, 7,200, 10,000, and 15,000 rotations per minute (RPM).

Magnetic Disk



Logical View of Magnetic Disk

Modern magnetic disk drives are addressed as large one-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer.

- The size of a logical block is usually 512 bytes, although some disks can be low-level formatted to have a different logical block size.

The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially. Sector 0 is the first sector of the first track on the outermost cylinder. The mapping proceeds in order through that track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

.

Logical View of Magnetic Disk (cont.)

Converting a logical block number into a cylinder number, a track number within that cylinder, and a sector number within that track is difficult for two reasons:

- First, most disks have some defective sectors, but the mapping hides this by substituting spare sectors from elsewhere on the disk.
- Second, the number of sectors per track is not a constant on some drives. Tracks in the outermost zone typically hold 40 percent more sectors than tracks in the innermost zone. For convenience cylinders are grouped into three zones (outer, middle and inner zones). The number of sectors per track is constant winthin a particular zone.

Disk Scheduling

The OS must use the disk drive efficiently. The hardware related delays in transfer of data between disk and memory are combination of three factors

- Seek Time the time necessary to move the disk arm to the desired cylinder.
- Rotational Latency: the time necessary for the desired sector to rotate to the disk head.
- The transfer rate is the rate at which data flow between the drive and the computer.

Typical disks can transfer several megabytes of data per second and they have seek times and rotational latencies of several milliseconds.

Disk Scheduling (cont.)

The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Both the access time and the bandwidth can improve by managing the order in which disk I/O requests are serviced.

Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information:

- Whether this operation is input or output
- What the disk address for the transfer is
- What the memory address for the transfer is
- What the number of sectors to be transferred is

Disk-scheduling Algorithms

In a multiprogramming system, many processes request for I/O service and the disk queue may often have several pending requests. When one request is completed, the operating system chooses which pending request to service next. The operating system make this choice by using one of several disk-scheduling algorithms.

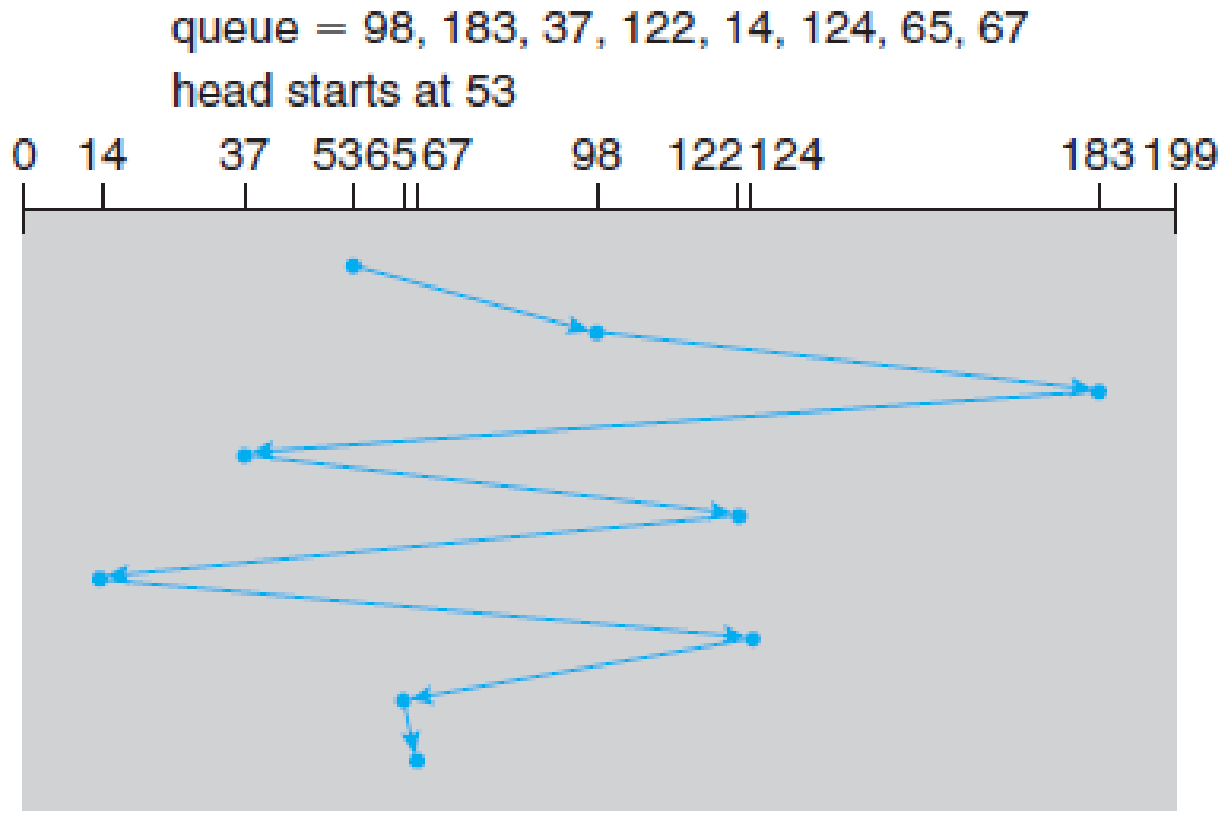
FCFS Scheduling Algorithm

The simplest form of disk scheduling is the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it does not provide the fastest service.

Consider, for example, a disk queue with requests for I/O to blocks on cylinders in the FCFS order of 53, 98, 183, 37, 122, 14, 124, 65, 67

If the disk head is initially at cylinder, it will first move from 53 to 98, then to 183, 37, 122, 14, 124, 65, and finally to 67, for a total head movement of 640 cylinders. The performance can be improved by selecting the nearest cylinder from the current location.

Illustrating the FCFS Algorithm



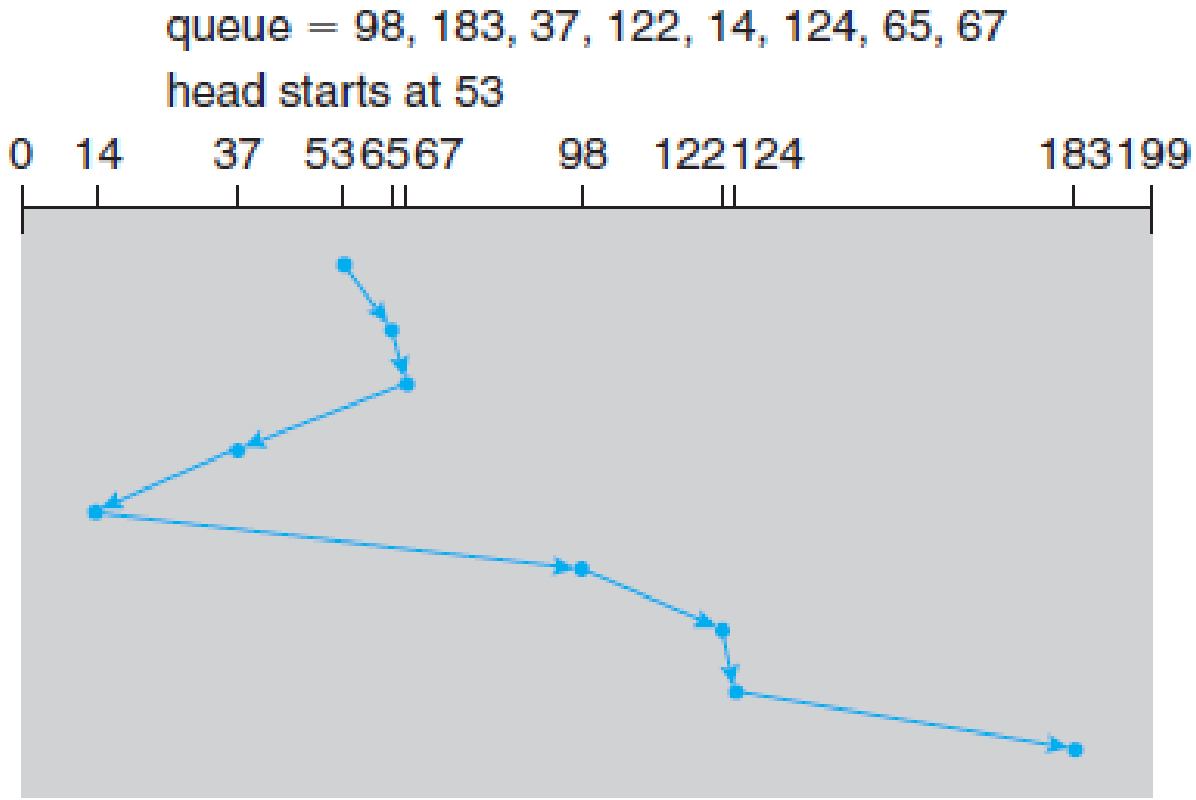
Shortest-seek-time-first Scheduling Algorithm

The basis for the shortest-seek-time-first (SSTF) algorithm is on the principle that servicing all the requests close to the current head position before moving the head far away to service other requests. The SSTF algorithm selects the request with the least seek time from the current head position. In other words, SSTF chooses the pending request closest to the current head position.

For same example request queue, the closest request to the initial head position (53) is at cylinder 65. The next closest request from cylinder 65 is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next. Continuing, the servicing of the request at cylinder 14, then 98, 122, 124, and finally 183 . This scheduling method results in a total head movement of only 236 cylinders—little more than one-third of the distance needed for FCFS scheduling of this request queue.

Clearly, this algorithm gives a substantial improvement in performance.

Illustrating the SSTF Algorithm



SCAN Scheduling Algorithm

In the SCAN algorithm, the read/write disk arm heads starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk.

- The SCAN algorithm is sometimes called the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

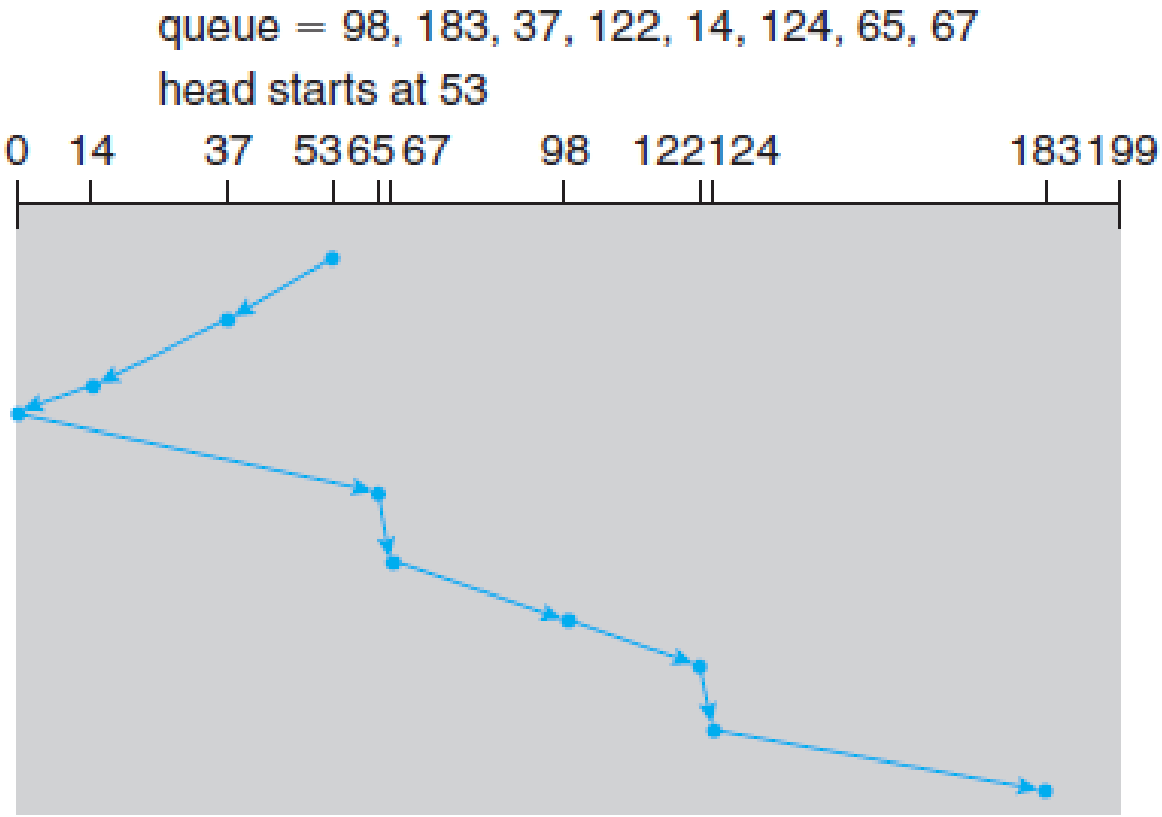
Illustrating the SCAN algorithm

Before applying SCAN to schedule the requests on cylinders 98, 183, 37, 122, 14, 124, 65, and 67, we need to know the direction of head movement in addition to the head's current position and the requests are sorted in ordered by the cylinder number

Assuming that the disk arm is moving toward 0 and that the initial head position is again 53, the head will next service 37 and then 14. At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183.

Drawback: If a request arrives in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

Illustrating the SCAN Algorithm

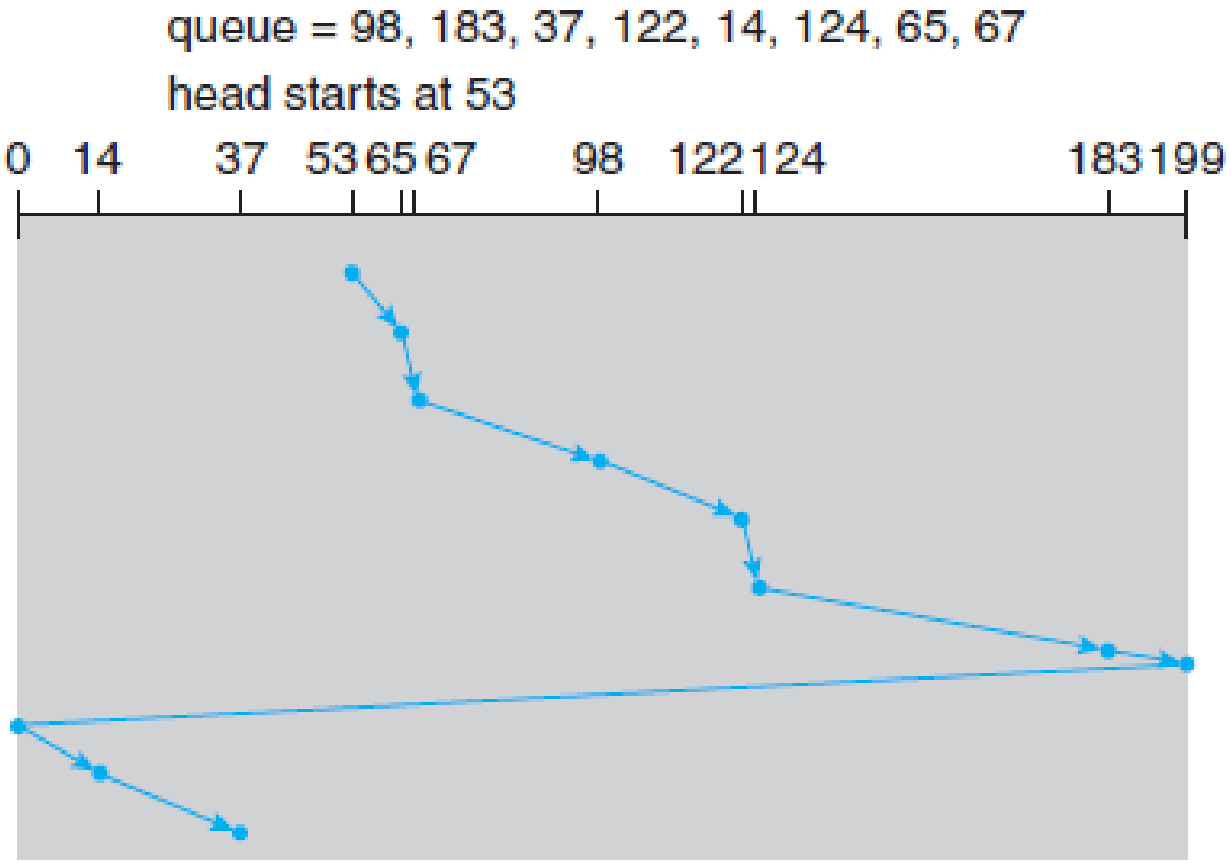


Circular SCAN (C-SCAN) Scheduling Algorithm

Circular SCAN (C-SCAN) scheduling Algorithm is similar to SCAN scheduling Algorithm moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip. This algorithm is designed to provide a more uniform wait time.

The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.

Illustrating the C-SCAN Algorithm



LOOK Scheduling

In both SCAN and C-SCAN move the disk arm across the full width of the disk. In practice, the arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk.

The versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction.

Selection of a Disk-Scheduling Algorithm

SSTF is common and has a natural appeal because it increases performance over FCFS.

SCAN and C-SCAN perform better for systems that place a heavy load on the disk, because they are less likely to cause a starvation problem.

For any particular list of requests, it is possible to define an optimal order of retrieval, but the computation needed to find an optimal schedule.

The performance of a scheduling algorithm depends heavily on the number and types of requests.

I. For instance, suppose that the queue usually has just one outstanding request. Then, all scheduling algorithms behave the same, because they have only one choice of where to move the disk head: they all behave like FCFS scheduling

Selection of a Disk-Scheduling Algorithm (cont.)

II. Requests for disk service can be greatly influenced by the file-allocation method.
   - A program reading a contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement.
   - A linked or indexed file, in contrast, may include blocks that are widely scattered on the disk, resulting in greater head movement.

III. The location of directories and index blocks is also important.
   - Since every file must be opened to be used, and opening a file requires searching the directory structure, the directories will be accessed frequently. Suppose that a directory entry is on the first cylinder and a file's data are on the final cylinder. In this case, the disk head has to move the entire width of the disk.

Selection of a Disk-Scheduling Algorithm (cont.)

Because of these complexities, the disk-scheduling algorithm should be written as a separate module of the operating system, so that it can be replaced with a different algorithm if necessary.