Disk Hardware

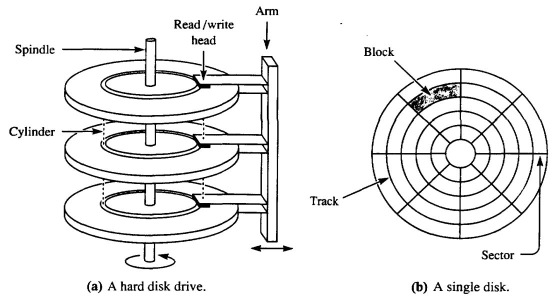


Diagram from *Computer Science*, Volume 2, J. Stanley Warford, Heath, 1991.

- Arm can move in and out
  - Read / write head can access a ring of data as the disk rotates
- Disk consists of one or more *platters*
  - Each platter is divided into rings of data, called *tracks*, and each track is divided into *sectors*
  - One particular platter, track, and sector is called a *block*

Disk Hardware (cont.)

- Typical disk a few years ago (Compaq 40GB Ultra ATA 100 7200RPM = $369):
  - 16383 cylinders, 16 heads, 63 sectors/track
  - 16 platters * 16383 tracks/platter * 63 sectors/track * 4048 bytes/sector * 1/1024^3 GB/byte = 63GB unformatted
  - 7200 rpm spindle speed, 8 ms average seek time, 100 MBps data transfer rate
- Trends in disk technology
  - Disks get smaller, for similar capacity
    - Faster data transfer, lighter weight
  - Disk are storing data more densely
    - Faster data transfer
    - Density improving faster than mechanical limitations (seek time, rotational delay)
  - Disks are getting cheaper (factor of 2 per year since 1991)

Improving Disk Performance

- Keep some structures in memory
  - Active inodes, file tables
- Efficient free space management
  - Bitmaps
- Careful allocation of disk blocks
  - Contiguous allocation where possible
  - Direct / indirect blocks
  - Good choice of block size
  - Cylinder groups
  - Keep some disk space in reserve
- Disk management
  - Cache of disk blocks
  - Disk scheduling

Improving Performance Using a Disk Cache

- Have OS (not hardware) manage a *disk block cache* to improve performance
  - Use part of main memory as a cache
  - When OS reads a file from disk, it copies those blocks into the cache
  - Before OS reads a file from disk, it first checks the cache to see if any of the blocks are there (if so, uses cached copy)
- Replacement policies for the blocks:
  - Same options as paging
    - FIFO, LRU using clock / second chance
  - Easy to implement exact LRU
    - OS just records time along with everything else it has to update when a block is read
  - But — sequential access degrades LRU
    - Solution: free-behind policy for large sequentially-accessed files — as block is read, remove previous one from cache

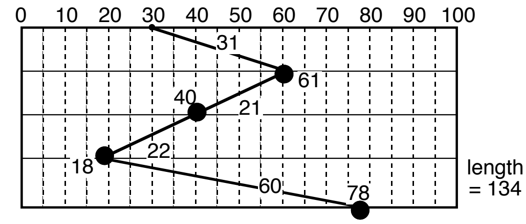
Improving Performance with Disk Head Scheduling

- Permute the order of the disk requests
  - From the order that they arrive in
  - Into an order that reduces the *distance* of seeks
- Examples:
  - Head just moved from lower-numbered track to get to track 30
  - Request queue: 61, 40, 18, 78
- Algorithms:
  - First-come first-served (FCFS)
  - Shortest Seek Time First (SSTF)
  - SCAN (0 to 100, 100 to 0, …)
  - C-SCAN (0 to 100, 0 to 100, …)

Disk Head Scheduling (cont.)

- FCFS
  - Handle in order of arrival

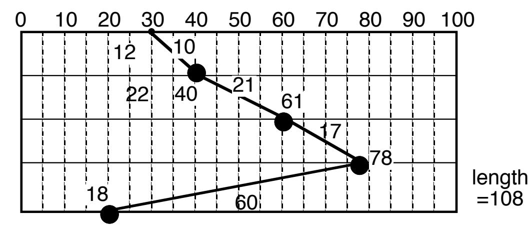
FCFS



- SSTF
  - Select request that requires the smallest seek from current track

SSTF

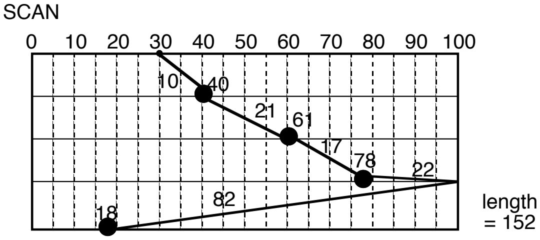
C



Disk Head Scheduling (cont.)

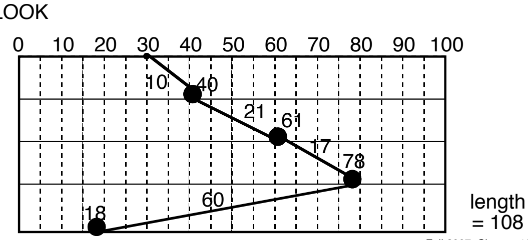
- SCAN (Elevator algorithm)
  - Move the head 0 to 100, 100 to 0, picking up requests as it goes

SCAN



- LOOK (variation on SCAN)
  - Don’t go to end unless necessary

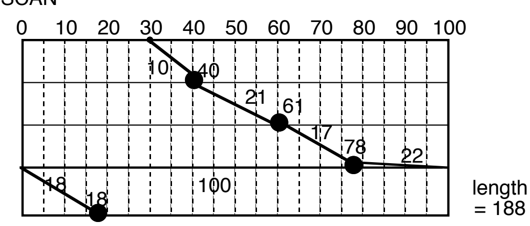
LOOK



Disk Head Scheduling (cont.)

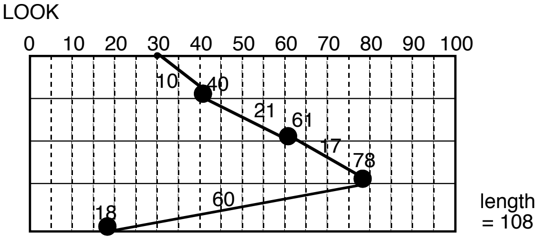
- C-SCAN (Circular SCAN)
  - Move the head 0 to 100, picking up requests as it goes, then big seek to 0

CSCAN



- C-LOOK (variation on C-SCAN)
  - Don’t go to end unless necessary

LOOK



Comparison of Disk Head Scheduling Methods

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced as necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm

Disk Management

- Disk formatting
  - Physical formatting — dividing disk into sectors: header, data area, trailer
  - Most disks are preformatted, although special utilities can reformat them
  - After formatting, must partition the disk, then write the data structures for the file system (logical formatting)
- *Boot block* contains the "bootstrap" program for the computer
  - System also contains a ROM with a bootstrap loader that loads this program
- Disk system should ignore bad blocks
  - When disk is formatted, a scan detects bad blocks and tells disk system not to assign those blocks to files
  - Disk may also do this as disk is used

Disk Management (cont.)

- Swap space management
  - Swap space in normal file system
  - Swap space in separate partition
    - One big file — don't need whole file system, directories, etc.
    - Only need manager to allocate/deallocate blocks (optimized for speed)
- Disk reliability
  - Data normally assumed to be persistent
  - Disk striping — data broken into blocks, successive blocks stored on separate drives
  - Mirroring — keep a "shadow" or "mirror" copy of the entire disk
  - Stable storage — data is never lost during an update — maintain two physical blocks for each logical block, and both must be same for a write to be successful