Magnetic Disks

Magnetic disks have dominated secondary storage since 1965. They play two roles in computer systems:

- Long-term, nonvolatile storage for files, even when no programs are running
- Level of memory hierarchy below main memory used for virtual memory during program execution

A magnetic disk is an electro-mechanical device:

- built-in disk controller
- shaft or spindle that rotates disk (e.g., at 5,400 or 15,000 RPM)
- conducting coil (head), mounted on arm, through which data are recorded/retrieved
  - one per surface
  - remains stationary during write/read
  - in contact with surface (of floppy disk), or in fixed position above surface, or floats aerodynamically (Winchester disk)
track
- ring on surface over which head passes as it records/retrieves
  - same width as head

Each surface organised as follows:
- concentric set of tracks
  - typically between 10,000 and 50,000
  - head moves to desired track (seek operation)
- adjacent tracks separated by gaps
  - reduces sensitivity to head alignment and magnetic field interference

cylinder
- refers to tracks at given position on all surfaces

Moving-head disk mechanism

sector
- subdivision of track
  - typically between 100 and 500
  - sectors separated by gaps
- smallest amount of information that can be read or written
  - typically 512 bytes
  - includes sector number and extra bits for error correction
Zone Bit Recording (ZBR)
- Introduced in the 1990s
- Purpose is to increase overall storage capacity of disk
  - Surface divided into zones (e.g. 16)
  - Each zone consists of number of concentric tracks
- Outer zones contain more sectors (and hence bits) per track than inner zones
  - Timing of reads/writes varies between zones
  - Therefore requires more complex disk controller

Disk Performance

Seek time
- Time it takes to perform seek
- Manufacturers report minimum seek time, maximum seek time and average seek time in their manuals
  - Minimum and maximum easy to measure
  - Average calculated as sum of time for all possible seeks divided by their number, e.g., 6 ms
- Actual average seek time depends on application and operating system
  - E.g., only 25% of advertised value because of locality of reference

Rotation latency (Rotational delay)
- Time for requested sector to rotate under head
  - $0.5 / r$, where $r$ is rotation speed
  - E.g., many disks rotate at 5400 RPM, and must on average travel halfway around disk to find sector; then, average latency is $0.5$ revolutions / 5400 RPM = 0.0056 seconds = 5.6 ms

Transfer time
- Time it takes to transfer block of data (i.e., one or more sectors) under head
  - $n / N r$, where $n$ is number of adjacent sectors and $N$ is number of sectors on track
  - E.g., reading 1 sector of 32-sector track at 5400 RPM takes 0.3 ms

Access time
- Sum of seek time, rotation delay and transfer time
N.B. order in which sectors are accessed has tremendous effect on access time
- seeks is required to move to track containing first sector and each time next sector is on different track
- rotational delay is incurred after each seek and each time next sector is not adjacent
- sequential access of sectors much faster than random access

N.B. the effect of disk controller should really be taken into account:
- adds to access time, but
- has a built-in cache that stores sectors as they are passed over
  ➔ higher transfer rate, e.g., 320MB/sec instead of 80MB/sec

Comparing magnetic disk with DRAM:
- 100 times cheaper
- 100,000 slower

At http://www.pcguide.com/ref/hdd/, the PC Guide once again goes into much more detail about hard disk drives than you would ever want to know!

Disk Scheduling
Whenever a process needs to perform disk I/O, it issues system call
- if resource available, request serviced immediately
- if busy, request added to queue for that drive
disk scheduling
- policy or algorithm by which OS selects pending request from queue for service
  – main ones are FIFO, SSTF, SCAN and C-SCAN
- affects number of tracks traversed by head and hence seek time

First-in-first-out (FIFO) or First-come-first-served (FCFS)
- requests served in order of arrival
- simplest and fairest policy
- works well when few processes, each accessing sectors that are clustered together
- poor when many processes compete for access to disk

Shortest-service/seek-time-first (SSTF)
- scheduler needs to know current track position
- chooses request involving nearest track
  – method for tie-breaking also needs to be adopted
- not optimal, but likely to be better than FIFO
- starvation problem
**SCAN or elevator algorithm**

- arm moves in one direction only until it reaches last track (or until no further requests in that direction, also known as LOOK), servicing requests as it goes
- reverses direction after each scan
- no starvation
- upon reversal, tracks with highest density of requests likely to be furthest away!

**Circular SCAN (C-SCAN)**

- arm flies back to beginning instead of reversing direction upon reaching last track or no further requests in that direction (C-LOOK)

---

**Example**

Current position: track 100
Requested tracks, in order received: 55, 58, 39, 18, 90, 160, 150, 38, 184

- **FIFO**: move 45 tracks to track 55; move 3 tracks to track 58; etc.
- **SSTF**: move 10 tracks to track 90; move 32 tracks to track 58; etc.
- **SCAN, C-SCAN** (moving in direction of increasing track number): move 50 tracks to track 150; move 10 tracks to track 160; etc.

---

**RAID**

**Redundant Array of Independent Disks (RAID)**

- set of physical disk drives viewed by OS as single logical drive
- data distributed (striped) across physical drives
  - logically consecutive strips of data are stored on multiple drives
  - way this is done depends on RAID level (0-6)
  - simultaneous access to multiple drives increases I/O performance (and also probability of failure)
- redundancy enables recovery in case of disk failure
  - no redundancy at RAID level 0
  - RAID level 1 duplicates content of each disk (mirroring)
  - higher levels use more sophisticated techniques
Disk Cache

disk cache
- portion of main memory containing copy of some sectors of disk for faster access

Two ways to handle cache hit:
- copy data from disk cache to another part of memory (assigned to user process)
- treat disk cache as shared memory, passing to user process only pointer to location of data

Replacement strategy must be chosen (cf. page replacement) for when new sector brought into disk cache

least recently used (LRU)
- replace block that has been in disk cache longest with no reference to it
- most commonly used algorithm
- cheap to implement
  - maintain linked list of blocks

least frequently used (LFU)
- replace block that has experienced fewest references
- simple scheme uses one counter per block
  - problem is that high count may build up over short interval, after which block is no longer needed
- problem can be overcome by more refined scheme
  - not incrementing block when new, and not making block eligible for replacement until old