CS 537
Lecture 10
Disks Scheduling

Michael Swift

Disk Access Time

- ***Seek time:*** the time to position disk heads (~8 msec on average)
- ***Rotational latency:*** the time to rotate the target sector to underneath the head
  - Assume 7,200 rotations per minute (RPM)
  - 7,200 / 60 = 120 rotations per second
  - 1/120 = ~8 msec per rotation
  - Average rotational delay is ~4 msec

Disk Access Time

- ***Transfer time:*** the time to transfer bytes
  - Assumptions:
    - 58 Mbytes/sec
    - 4-Kbyte disk blocks
  - Time to transfer a block takes 0.07 msec
- ***Disk access time***
  - Seek time + rotational delay + transfer time

Disk Performance Metrics

- ***Latency***
  - Seek time + rotational delay
- ***Bandwidth***
  - Bytes transferred / disk access time

Examples of Disk Access Times

- If disk blocks are randomly accessed
  - Average disk access time = ~12 msec
  - Assume 4-Kbyte blocks
  - 4 Kbyte / 12 msec = ~340 Kbyte/sec
- If disk blocks of the same cylinder are randomly accessed without disk seeks
  - Average disk access time = ~4 msec
  - 4 Kbyte / 4 msec = ~ 1 Mbyte/sec

Examples of Disk Access Times

- If disk blocks are accessed sequentially
  - Without seeks and rotational delays
  - Bandwidth: 58 Mbytes/sec
- Key to good disk performance
  - Minimize seek time and rotational latency

Disk Tradeoffs

Sector size	Space utilization	Transfer rate
1 byte	8 bits/1008 bits (0.8%)	80 bytes/sec (1 byte / 12 msec)
4 Kbytes	4096 bytes/4221 bytes (97%)	340 Kbytes/sec (4 Kbytes / 12 msec)
1 Mbyte	(~100%)	58 Mbytes/sec (peak bandwidth)

- Disk adds 1000 bits overhead per sector
- Larger sector size → better bandwidth
- Wasteful if only 1 byte out of 1 Mbyte is needed

Disk Calculations

- Example disk:
  - `#surfaces: 4`
  - `#tracks/surface: 64K`
  - `#sectors/track: 1K (assumption??)`
  - `#bytes/sector: 512`
  - `RPM: 7200 = 120 tracks/sec`
  - `Seek cost: 1.3ms - 16ms`
- Questions
  - How many disk heads? How many cylinders?
  - How many sectors/cylinder? Capacity?
  - What is the maximum transfer rate (bandwidth)?
  - Average positioning time for random request?
  - Time and bandwidth for random request of size:
    - 4KB?
    - 128 KB?
    - 1 MB?

Reliability

- Disks fail more often....
  - When continuously powered-on
  - With heavy workloads
  - Under high temperatures
- How do disks fail?
  - Whole disk can stop working (e.g., motor dies)
  - Transient problem (cable disconnected)
  - Individual sectors can fail (e.g., head crash or scratch)
    - Data can be corrupted or block not readable/writable
- Disks can internally fix some sector problems
  - ECC (error correction code): Detect/correct bit flips
  - Retry sector reads and writes: Try 20-30 different offset and timing combinations for heads
  - Remap sectors: Do not use bad sectors in future
    - How does this impact performance contract??

Buffering

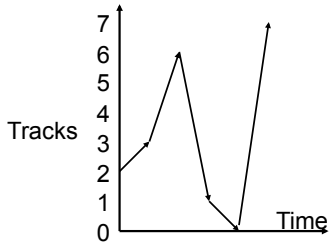
- Disks contain internal memory (2MB-16MB) used as cache
- Read-ahead: "Track buffer"
  - Read contents of entire track into memory during rotational delay
- Write caching with volatile memory
  - Immediate reporting: Claim written to disk when not
  - Data could be lost on power failure
    - Use only for user data, not file system meta-data
- Command queueing
  - Have multiple outstanding requests to the disk
  - Disk can reorder (schedule) requests for better performance

Disk Arm Scheduling Policies

- Goal: Minimize positioning time
  - Performed by both OS and disk itself; Why?
- ***First come, first serve (FCFS):*** requests are served in the order of arrival
  - \+ Fair among requesters
  - \- Poor for accesses to random disk blocks
- ***Shortest seek time first (SSTF):*** picks the request that is closest to the current disk arm position
  - \+ Good at reducing seeks
  - \- May result in starvation

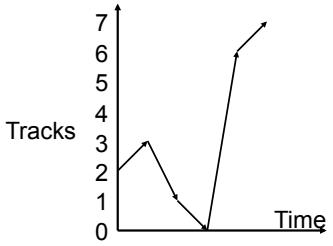
First Come, First Serve

- Request queue: 3, 6, 1, 0, 7
- Head start position: 2
- Total seek distance: 1 + 3 + 5 + 1 + 7 = 17



Shortest Seek Distance First

- Request queue: 3, 6, 1, 0, 7
- Head start position: 2
- Total seek distance: 1 + 2 + 1 + 6 + 1 = 10



Disk Arm Scheduling Policies

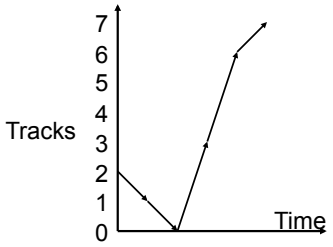
- ***SCAN:*** takes the closest request in the direction of travel (an example of elevator algorithm)

  \+ no starvation

  \- a new request can wait for almost two full scans of the disk

SCAN

- Request queue: 3, 6, 1, 0, 7
- Head start position: 2
- Total seek distance: 1 + 1 + 3 + 3 + 1 = 9

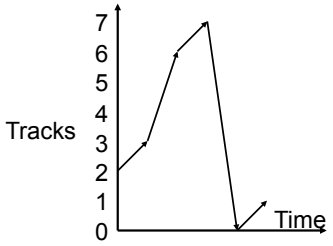


Disk Arm Scheduling Policies

- ***Circular SCAN (C-SCAN):*** disk arm always serves requests by scanning in one direction.
  - Once the arm finishes scanning for one direction
  - Returns to the 0^{th} track for the next round of scanning

C-SCAN

- Request queue: 3, 6, 1, 0, 7
- Head start position: 2
- Total seek distance: 1 + 3 + 1 + 7 + 1 = 13



Look and C-Look

- Similar to SCAN and C-SCAN
  - the arm goes only as far as the final request in each direction, then turns around
    - Look for a request before continuing to move in a given direction.