Mass Storage

CMSC 412, University of Maryland

Guest lecturer: David Hovemeyer

November 15, 2004

The memory hierarchy

Red = Mass Storage

Level	Access time	Capacity	Features
Registers	nanoseconds	100s of bytes	fixed
Cache	nanoseconds	1-2 MB	fixed
RAM	nanoseconds	MBs to GBs	expandable
Disk	milliseconds	100s of GBs	stable, expandable
CD, DVD ROM	10s of milliseconds	MBs to GBs	stable, removeable
Tape	seconds–hours	1000s of GBs	stable, removeable

Disks

How disks are used

- Disks provide *stable storage*
  - Persists when power is turned off
- Stores OS kernel, executable programs, user files
- Processes perform *file I/O*
  - Filesystem translates these into *disk I/O*

Interesting facts

- The basic design of hard disks has not changed since the 1960s
- Storage capacities have increased enormously
- Transfer rates keep increasing (due to increased storage density), but seek times stay relatively constant
- Other technologies on the horizon?
  - MEMs
  - Holographic storage

Partitioning

- Disk can be divided into multiple *partitions* (logical disks)
- Typical disk layout:

Boot block and partition table	Beginning of disk
Partition 1	
Partition 2...	End of disk

Boot block

- Most systems boot the OS from a disk
- The OS installs a *boot block* to initiate loading of the OS
- Boot process:
  - CPU initialized, loads bootstrap program from ROM
  - Bootstrap program loads the *boot block* from disk
  - Boot block usually loads a *second stage loader*
  - Second stage loader loads the OS kernel

A disk read

When the kernel needs to read data from a disk:

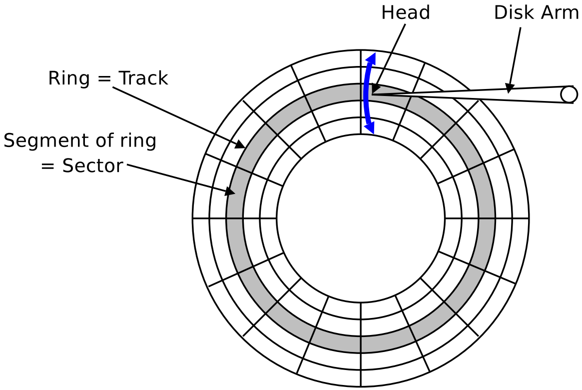
1. Program the disk controller with the address of the data on the disk
2. Wait for read to complete (other tasks may run in the meantime)
3. Controller raises an interrupt when request completes
4. Copy data from controller
   - Controller may do this automatically using DMA
5. Thread or process that issued the request may continue

A disk write

When the kernel wants to write data to the disk:

1. Program disk controller with disk address to be written
2. Copy data to controller
   - Or arrange for DMA

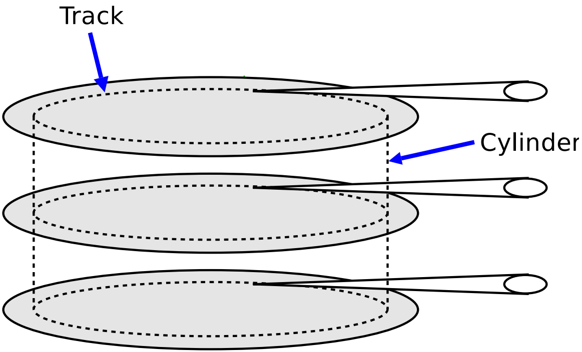
Disk organization



Disk organization

Drives usually contain multiple platters

Disk arms are connected, heads move together



Disk access

- Smallest addressable unit of disk storage: *sector* or *block*
  - Usually 512 bytes
  - Sector size can be changed by a *low-level format*
- Sector addressing
  - CHS: cylinder, head (track), sector
  - LBA: logical block address
    - For disk with N sectors, sectors assigned numbers from 0..N
- All modern disks and OSes use LBA

Low-level formatting

- How is data physically stored?
- Each sector on the disk has extra information (in addition to the sector data), generated by *low-level formatting*
    - Sector number
    - Error-correcting code
- Extra information allows
    - Errors to be detected and corrected
    - Blocks to be moved
- Users generally never need to do low-level format

Performance factors

- Performance factors
  - Seek time: time for heads to move to desired track
  - Rotational latency: time for disk to spin to desired sector
- Typical values
  - 8 milliseconds avg. seek
  - 4 milliseconds avg. latency
- Seeks are slow!
  - 2 GHz CPU executes 1.6 million cycles in 8 milliseconds
- However, transfer rate is high (10's or 100's MB/sec)
  - Once the heads are in the right place

Real disks

- Complicating factors for real disk drives:
    - Outer tracks have more sectors than inner tracks
    - Bad blocks
        - Some sectors will have defects
        - Drive remaps these to good sectors elsewhere
        - Hopefully near original location, but maybe not
- However, logical block address generally reflects physical geometry
    - Sectors with close LBAs should be near each other (generally, in same cylinder)

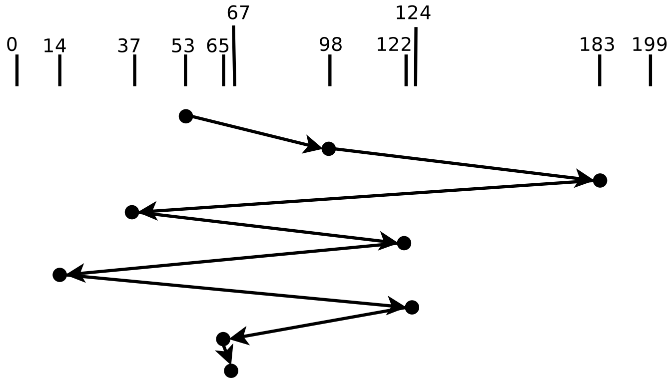
Disk scheduling

- Model: queued requests to read/write disk blocks
- How should the OS schedule the requests so that they complete as quickly as possible?

  $\Longrightarrow$ Minimize seek time (head movement)
- In theory, we could also try to minimize rotational latency
  - Hard to do, because the OS generally doesn't know the real disk geometry

FCFS: First Come, First Served

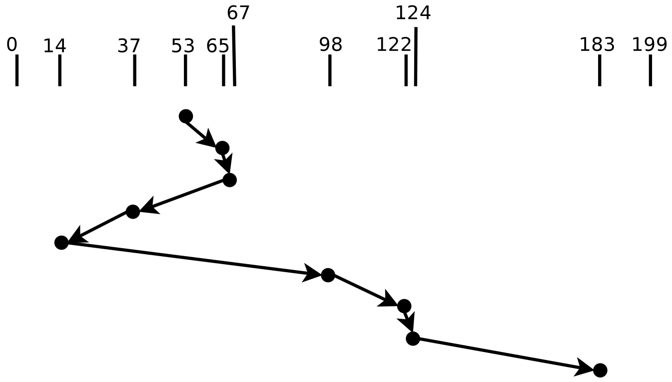
Lots of unnecessary head movement—can do better



Start: 53, Requests: 98, 183, 37, 122, 14, 124, 65, 67

SSTF: Shortest Seek Time First

Better, but requests to far cylinders may be starved

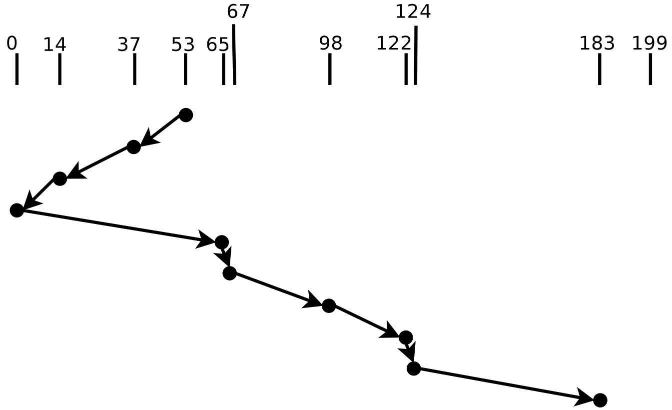


Start: 53, Requests: 98, 183, 37, 122, 14, 124, 65, 67

SCAN (Elevator algorithm)

Move head back and forth across disk

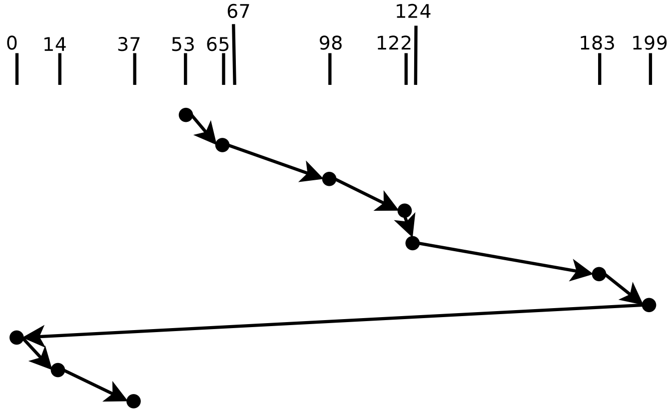
Less prone to starvation, but outer cylinders wait longer



Start: 53, Requests: 98, 183, 37, 122, 14, 124, 65, 67

C-SCAN (“Circular SCAN”)

Only handle requests in one direction, then wrap
All cylinders serviced with uniform frequency



Start: 53, Requests: 98, 183, 37, 122, 14, 124, 65, 67

LOOK, C-LOOK

- SCAN and C-SCAN traverse entire disk, even if there are no further requests in current direction
- LOOK and C-LOOK reverse head motion as soon as possible
- Eliminates some unnecessary head movement with slight reduction in fairness

Tagged queueing

- So far we've assumed only one request can be active
- However, most modern disk drives/controllers can support multiple outstanding requests
- Allows controller and drive to do further scheduling and optimization
  - E.g., if we dispatch multiple requests for sectors in the same cylinder, disk can order them to take rotational latency into account
- Controller tells kernel which requests have completed

RAID

RAID concepts

- Need for more storage capacity that would fit on one disk:
  - Enterprise-wide user directories
  - Large databases
  - Document archives
  - etc.
- RAID—Redundant Array of Independent Disks
  - Create a very large "virtual" disk out of multiple disk drives
- Use multiple disks to increase
  - Throughput
  - Reliability

Disk performance

- Some applications need to read or write faster than the transfer rate of a single disk
  - E.g., satellite data, medical imaging, physics experiments
  - May need 100 GBs/sec of I/O
- Using multiple disks, I/O can proceed in parallel

Disk reliability

- MTBF: Mean Time Between Failures
- For a single disk, MTBF is large (10^6 hours or more)
- For N disks with MTBF of m, mean time between failure of any disk is m/N
  - E.g., for 1000 disks with MTBF of 10^6 hours, MBTF is 1000 hours, about 42 days!
- *Redundancy* is needed
  - Assume a relatively small replacement time (hours) for a failed disk
  - Probability of independent failures of two disks within the replacement time is very small

RAID levels

- There are many ways to organize multiple disks to achieve better throughput and/or reliability
- These are described as *RAID levels*: RAID 0, RAID 1, etc.
- The numbers do not really mean anything

RAID levels 0 and 1

- RAID 0: Also known as *striping*
  - Logical block 0 on disk 0, logical block 1 on disk 1, etc.
  - Multiple blocks can be transferred in parallel: higher transfer rate
  - No redundancy
- RAID 1: Also known as *mirroring*
  - Blocks 0..n on disk 0, n+1..2n on disk 1, etc.
  - Each disk in the array has a mirror with same contents
  - If one disk fails, use mirror
  - Generally, no parallelism for individual requests
    - Since adjacent logical blocks are on the same disk

RAID 2

- Memory-style error correcting organization
- Stripe *bytes* of data across disks
- Dedicate some disks to store error correcting codes for those bytes
  - Automatically detect and correct single bit errors
- This scheme is not generally used in practice

RAID 3

- Bit-interleaved parity organization
- Rely on fact that disk drives can reliably detect errors
- Like RAID 2, stripe bytes across disks
- Use dedicated disks to store *parity* for each byte
- On a single disk error:
  - Compute parity of good bits, compare with known parity bit
  - If parity of good bits matches known parity, damaged bit is 0
  - Otherwise, damaged bit is 1

RAID 4

- Block-interleaved parity organization
- Like RAID 4, but *blocks* are interleaved on data disks
- With array of n disks, each parity block stores parity for $n-1$ data blocks
- Using good blocks and parity block, any damaged data block can be reconstructed
- Read operations for many blocks can proceed in parallel

General critique of RAID 3 and RAID 4

- The problem with RAID 3 and RAID 4 is that parity disks are a bottleneck
  - They must be updated on *every* write of any associated data disk
- Idea: all disks should store both data and parity information

RAID 5

- Block-interleaved distributed parity
- Say we have n disks with S sectors each
- Use S parity blocks
- Parity block i stored on disk i mod n
  - Data blocks stored on other disks
- Why is this good?
  - Like RAID 4, we need to write to two disks for every write of a virtual block
  - But, with interleaving, no single disk is a bottleneck

RAID 6

- P+Q redundancy scheme
- Like RAID 5, but instead of storing parity of data blocks, store error correcting code
  - Allow recovery from multiple disk failures

Issues with parity and error correcting codes

- Maintaining error correcting information imposes performance overhead
  - Can use dedicated hardware for computation
  - Each write still requires multiple steps to complete
- Time required to rebuild redundancy following a failure can be significant (many disks involved)

RAID 0+1, RAID 1+0

- Combine RAID levels 0 and 1
- RAID 0+1: mirrored RAID 0 arrays
- RAID 1+0: stripe blocks across pairs of mirrored disks
- These systems have both high throughput and high reliability
  - Rebuild: just copy a single disk, I/O to other disks can proceed as usual
- However, both impose 100% space overhead compared to RAID 0
  - Useful for small or medium size databases

Tertiary storage

Tertiary storage

- Two problems with disks:
  - They are not *removeable*
  - Relatively high cost per drive
- Cheap, removeable storage:
  - Flash memory ("USB hard drive")
  - Optical disks (CD and DVD ROM)
    - Might be writable once (WORM) or many times
  - Tapes

Optical drives (CD, DVD, etc.)

- For reading, accessed much like fixed hard disks
- Some differences:
  - Variable speed: disk rotates faster when accessing inner tracks
  - Generally slower seek times and transfer rates than hard disks
- Writing:
  - Writable CD and DVD drives don't have fully random access (sector-at-a-time) write support
  - Streaming write model

Tapes

- Transfer rate similar to optical disks
- Differences:
  - Sequential access: random seeks are very slow
  - Larger capacity
  - Cheaper
- Used for data backup and long-term storage

Hierarchical storage management

- A problem with large collections of removeable disks or tapes is that they can't be accessed transparently
- A *robotic jukebox* can solve this problem
  - When data on a removeable disk or tape is requested, robot fetches it from library, inserts in drive
- Hierarchical storage management: use fixed disks as a cache for data in long-term storage
  - Data in disk cache can be accessed very quickly
  - Data not in cache may incur very long access time: minutes or hours
  - However, useful for batch processing

Questions?