Overview

• Disk trends
• Memory Hierarchy
• Performance
• Scheduling
• SSDs
  – Read
  – Write
  – Performance
  – Cost
Secondary storage

- Secondary storage:
  - anything outside “primary memory”
  - direct execution of instructions/data retrieval via machine load/store
    - not permitted
- Characteristics:
  - it’s large: 250-2000GB and more
  - it’s cheap: $0.05/GB for hard drives
- Persistent: data survives power loss
  - it’s slow: milliseconds to access
- It does fail, if rarely
- Big failures
  - drive dies; Mean Time Between Failure ~3 years
  - 100K drives and MTBF is 3 years,
    - that’s 1 “big failure” every 15 minutes!
- Little failures (read/write errors, one byte in $10^{13}$)
The First Commercial Disk Drive

1956 IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second
In the past

IBM 2314
About the size of 6 refrigerators
8 x 29MB
Required similar-sized air cond.

.01% the capacity of this $100 100x150x25mm item
Disk trends

• Disk capacity, 1975-1989
  – doubled every 3+ years
  – 25% improvement each year
  – factor of 10 every decade
  – Still exponential, but far less rapid than processor performance

• Disk capacity, 1990-recently
  – doubling every 12 months
  – 100% improvement each year
  – factor of 1000 every decade
  – Capacity growth 10x as fast as processor performance!
Disk cost

• Only a few years ago, disks purchased by the megabyte

• Today, 1 GB (a billion bytes) costs $0.05 from Dell
  – (except you have to buy in increments of 1000 GB)
  – => 1 TB costs $50, 1 PB costs $50K

• Performance analogy
  – Flying an aircraft at 600 mph 6” above the ground
  – Reading/writing a strip of postage stamps
Memory hierarchy

- Each level acts as a cache of lower levels
Disks and the OS

• Disks are difficult devices
  – errors, bad blocks, missed seeks, etc.

• OS abstracts this for higher-level software
  – low-level device drivers (initiate a disk read, etc.)
  – higher-level abstractions (files, databases, etc.)
  – disk hardware increasingly helps with this

• OS provide different levels of disk access to different clients
  – physical disk block (surface, cylinder, sector)
  – disk logical block (disk block #)
  – file logical (filename, block or record or byte #)
Physical disk structure

- Disk components
  - platters
  - surfaces
  - tracks
  - sectors
  - cylinders
  - arm
  - heads
Disk Structure

- Disk drives are addressed as
  - large 1-dimensional arrays of **logical blocks**,  
  - the logical block is the smallest unit of transfer  
  - Low-level formatting creates **logical blocks** on physical media

- The 1-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
  - Mapping proceeds in order through that track,  
  - Then the rest of the tracks in that cylinder,  
  - Then through the rest of the cylinders from outermost to innermost

- Logical to physical address should be easy
  - Except for bad sectors  
  - Non-constant # of sectors per track via constant angular velocity
Disk performance

- Performance depends on a number of steps
- **Seek**: moving the disk arm to the correct cylinder
  - depends on how fast disk arm can move
    - not diminishing quickly due to physics
- **Rotation (latency)**: waiting for the sector to rotate under head
  - depends on rotation rate of disk
    - rates are slowly increasing,
- **Transfer**: transferring data from surface to disk controller,
  - then sending it back to host
  - depends on density of bytes on disk
    - increasing, relatively quickly
- When the OS uses the disk, it tries to minimize the cost of all of these steps
  - particularly seeks and rotation
Performance

• OS may increase file block size
  – in order to reduce seeking
• OS may seek to co-locate “related” items
  – in order to reduce seeking
    • blocks of the same file
    • data and metadata for a file
• Keep data or metadata in memory to reduce physical disk access
  – Waste valuable physical memory?
• If file access is sequential,
  – fetch blocks into memory before requested
Performance via disk scheduling

- Seeks are very expensive, so the OS attempts to schedule disk requests that are queued waiting for the disk
  - FCFS (do nothing)
    - reasonable when load is low
    - long waiting time for long request queues
  - SSTF (shortest seek time first)
    - minimize arm movement (seek time), maximize request rate
    - unfairly favors middle blocks
  - SCAN (elevator algorithm)
    - service requests in one direction until done, then reverse
    - skews wait times non-uniformly
  - C-SCAN
    - like scan, but only go in one direction (typewriter)
    - uniform wait times
  - C-LOOK
    - Similar to C-SCAN
    - The arm goes only as far as the final request in each direction
FCFS

Illustration shows total head movement of 640 cylinders

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
SSTF

Illustration shows total head movement of *236 cylinders*

*-may cause starvation*

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53
The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.

SCAN algorithm Sometimes called the elevator algorithm

But note
- that if requests are uniformly dense,
- largest density at other end of disk
- and those wait the longest
Illustration shows total head movement

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
C-SCAN

• Provides a more uniform wait time than SCAN
• The head moves from one end of the disk to the other, servicing requests as it goes
  – When it reaches the other end, however,
    • it immediately returns to the beginning of the disk
    • without servicing any requests on the return trip
• Treats the cylinders as a circular list
  – that wraps around from the last cylinder to the first one
Illustration shows total head movement of 382 cylinders

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
C-LOOK

• LOOK a version of SCAN, C-LOOK a version of C-SCAN
• Arm only goes as far
  – as the last request in each direction,
  – then reverses direction immediately,
  – without first going all the way to the end of the disk
C-LOOK

Illustration shows total head movement of 322 cylinders

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
  - And metadata layout
- The disk-scheduling algorithm should be
  - written as a separate module of the operating system,
  - allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm
- What about rotational latency?
  - Difficult for OS to calculate
Interacting with disks

• Previously
  – OS would specify cylinder #, sector #, surface #, transfer size
    • i.e., OS needs to know all of the disk parameters
• Modern disks more complex
  – not all sectors are the same size, sectors are remapped, …
• Disk provides a higher-level interface, e.g., SCSI
  • exports data as a logical array of blocks [0 … N]
  • maps logical blocks to cylinder/surface/sector
• OS only names logical block #,
  – disk maps this to cylinder/surface/sector
  – on-board cache
  – as a result, physical parameters are hidden from OS
Seagate Barracuda 9cm disk drive

- 1 Terabyte of storage (1000 GB)
- $100
- 4 platters, 8 disk heads
- 63 sectors (512 bytes) per track
- 16,383 cylinders (tracks)
- 164 Gbits / inch-squared (!)
- 7200 RPM
- 300 MB/second transfer
- 9 ms avg. seek, 4.5 ms avg. rotational latency
- 1 ms track-to-track seek
- 32 MB cache
Solid state drives: ongoing disruption

- Hard drives are based on spinning magnetic platters
  - *mechanics* of drives determine performance characteristics
    - sector addressable, not byte addressable
    - capacity improving exponentially
    - sequential bandwidth improving reasonably
    - random access latency improving very slowly

- Cost dictated by
  - massive economies of scale,
  - and many decades of commercial development and optimization
SSD

• Solid state drives are based on NAND flash memory
  – no moving parts; performance characteristics driven by electronics and physics – more like RAM than spinning disk
  – relative technological newcomer, so costs are still quite high in comparison to hard drives, but dropping fast
SSD performance: reads

- **Reads**
  - unit of read is a *page*, typically 4KB large
- **Today's SSD can typically handle**
  - 10,000 – 100,000 reads/s
- **0.01 – 0.1 ms read latency**
  - 50-1000x better than disk seeks
- **40-400 MB/s read throughput**
  - 1-3x better than disk seq. throughput
SSD performance: writes

- **Writes**
  - flash media must be *erased* before it can be written to
  - unit of erase is a block, typically 64-256 pages long
    - usually takes 1-2ms to erase a block
    - blocks can only be erased a certain number of times before they become unusable – typically 10,000 – 1,000,000 times
  - unit of write is a page
    - writing a page can be 2-10x slower than reading a page
- **Writing to an SSD is complicated**
  - random write to existing block: read block, erase block, write back modified block
    - leads to hard-drive like performance (300 random writes / s)
  - sequential writes to erased blocks: fast!
    - SSD-read like performance (100-200 MB/s)
SSDs: dealing with erases, writes

- Lots of higher-level strategies can help hide the warts of an SSD
- Many of these work by
  - exposing logical pages, not physical pages
- Wear-levelling:
  - when writing,
  - try to spread erases out evenly across physical blocks of the SSD
    - Intel promises 100GB/day x 5 years for its SSD drives
- Log-structured filesystems:
  - convert random writes within a filesystem
  - to log appends on the SSD
SSD cost

• Capacity
  – today, flash SSD costs ~$2.50/GB
  • 1TB drive costs around $2500
    – 1TB hard drive costs around $50
  – Data on cost trends is volatile/preliminary

• Energy
  – SSD is typically more energy efficient than a hard drive
    • 1-2 watts to power an SSD
    • ~10 watts to power a high performance hard drive
    – (can also buy a 1 watt lower-performance drive)
Summary

• Disk trends
• Memory Hierarchy
• Performance
• Scheduling
• SSDs
  – Read
  – Write
  – Performance
  – Cost
• Next lecture: Revision tutorial