CSE 451: Operating Systems
Spring 2012

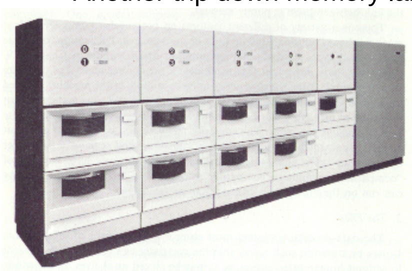
Module 14
Secondary Storage

Ed Lazowska
lazowska@cs.washington.edu
Allen Center 570

Secondary storage

- Secondary storage typically:
  - is anything that is outside of "primary memory"
  - does not permit direct execution of instructions or data retrieval via machine load/store instructions
- Characteristics:
  - it's large: 250-2000GB
  - it's cheap: $0.05/GB for hard drives
  - it's persistent: data survives power loss
  - it's slow: milliseconds to access
    - why is this slow??
  - it *does* fail, if rarely
    - big failures (drive dies; MTBF ~3 years)
      - if you have 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})

Another trip down memory lane …



IBM 2314
About the size of 6 refrigerators
8 x 29MB (M!)
Required similar-sized air condx!

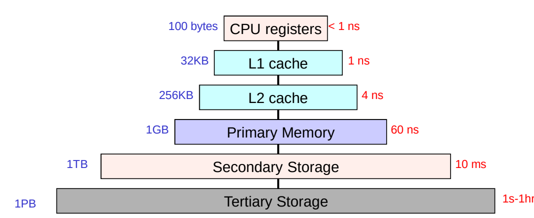
.01% (not 1% – .01%!) the capacity of this $100 4"x6"x1" item

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!

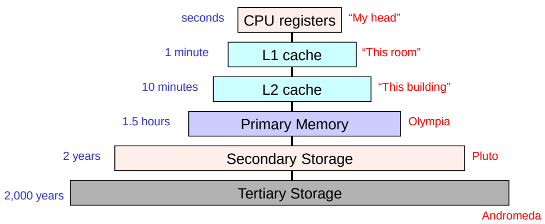
- Only a few years ago, we purchased disks by the megabyte (and it hurt!)
- Today, 1 GB (a billion bytes) costs ~~$1~~ ~~$0.50~~ $0.05 from Dell (except you have to buy in increments of ~~40~~ ~~80~~ ~~250~~ 1000 GB)
  - => 1 TB costs ~~$1K~~ ~~$500~~ $50, 1 PB costs ~~$1M~~ ~~$500K~~ $50K
- Technology is amazing
  - Flying a 747 6" above the ground
  - Reading/writing a strip of postage stamps
- But …
  - Jets do crash …

Memory hierarchy

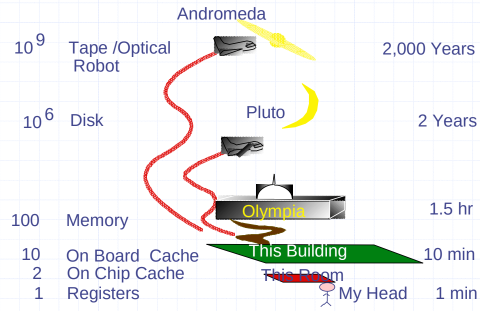


- Each level acts as a cache of lower levels

Memory hierarchy: distance analogy



Storage Latency: How Far Away is the Data?

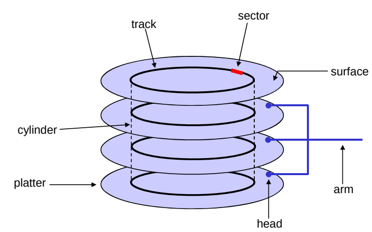


Disks and the OS

- Disks are messy, messy devices
  - errors, bad blocks, missed seeks, etc.
- Job of OS is to hide this mess from higher-level software (disk hardware increasingly helps with this)
  - low-level device drivers (initiate a disk read, etc.)
  - higher-level abstractions (files, databases, etc.)
- OS may 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 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
      - seek times aren't diminishing very quickly (why?)
  - rotation (latency): waiting for the sector to rotate under head
    - depends on rotation rate of disk
      - rates are increasing, but slowly (why?)
  - transfer: transferring data from surface into disk controller, and from there 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 via disk layout

- 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

Performance via caching, pre-fetching

- Keep data or metadata in memory to reduce physical disk access
  - problem?
- 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 (why?)
  - C-SCAN
    - like scan, but only go in one direction (typewriter)
    - uniform wait times

Interacting with disks

- In the old days…
  - OS would have to specify cylinder #, sector #, surface #, transfer size
    - i.e., OS needs to know all of the disk parameters
- Modern disks are even more complicated
  - 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 needs to name logical block #, disk maps this to cylinder/surface/sector
    - on-board cache
    - as a result, physical parameters are hidden from OS
      - both good and bad

Seagate Barracuda 3.5” disk drive

- 1Terabyte 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: imminent 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

- 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. thpt)

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 virtualizing pages and blocks on the drive (i.e., exposing logical pages, not physical pages, to the rest of the computer)
  - wear-leveling: when writing, try to spread erases out evenly across physical blocks of 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 (more later)
  - build drives out of arrays of SSDs, add lots of cache

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 a little sketchy and 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)