CSE 451: Operating Systems
Winter 2013

Secondary Storage

Gary Kimura

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: 50-1000GB (or more!)
  - it’s cheap: $0.25/GB … er… $0.10/Gb
  - it’s persistent: data survives power loss
  - it’s slow: milliseconds to access
    - why is this slow?
  - it *does* fail, if rarely
    - Big failures (disk dies)
    - Little failures (read/write errors, one byte in 10^13)

Two common storage devices

- Magnetic disks
  - Storage that rarely becomes corrupted (but when it does…)
  - Large capacity at low cost
  - Block level random access
  - Slow performance for random access
  - Better performance for streaming access
- Flash memory
  - Storage that rarely becomes corrupted (but when it does…)
  - Capacity at intermediate cost (50x disk)
  - Block level random access
  - Good performance for read; worse for random writes

Another trip down memory lane …

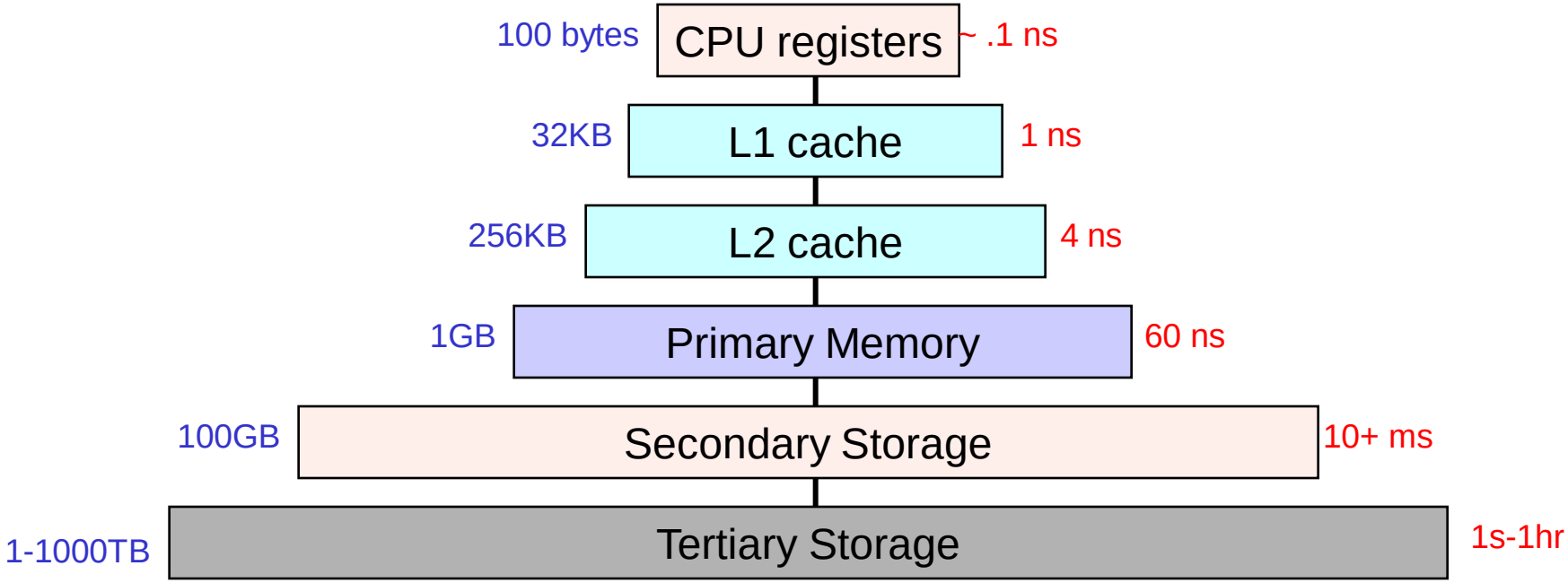
IBM 2314
About the size of
6 refrigerators
8 x 29MB (M!)
Required similar sized
air conditioning

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 since 1990
  - doubling every 12 months
  - 100% improvement each year
  - factor of 1000 every decade
  - Capacity growth10x as fast as processor performance!
- Speed has NOT grown similarly

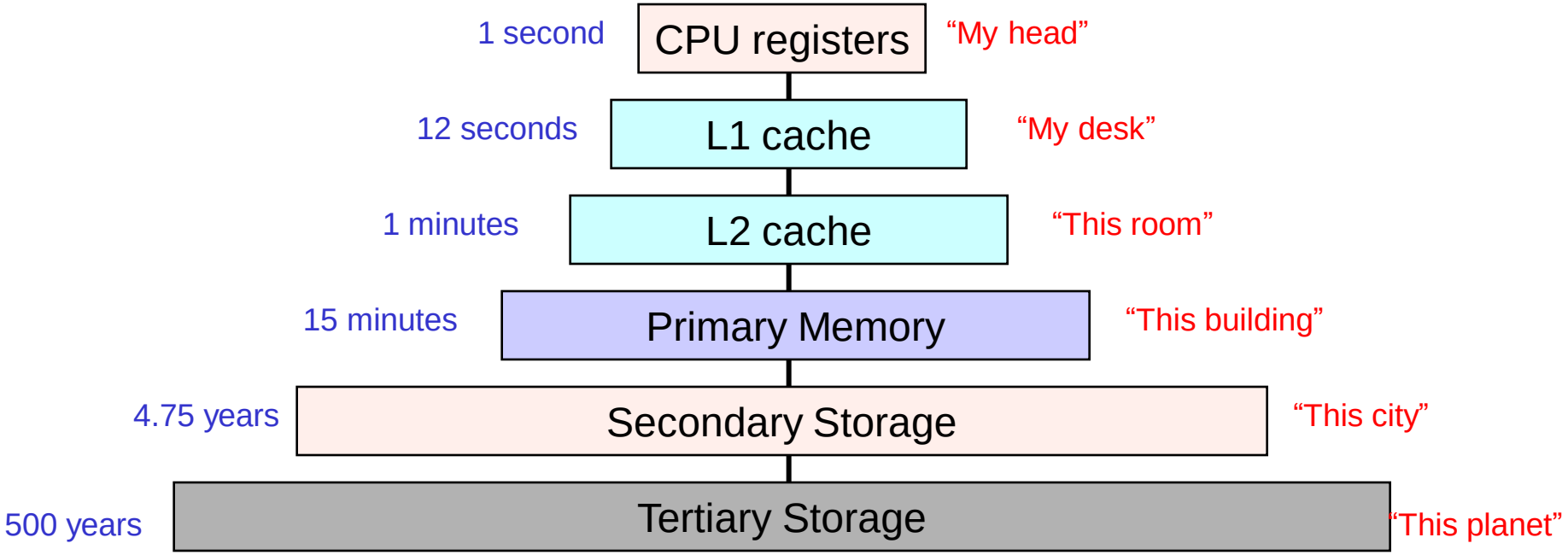
- Only a few years ago, we purchased disks by the megabyte (and it hurt!)
- Today, 1 GB (a billion bytes) costs $1 $0.50 $0.25 $0.10 from Dell (except you have to buy in increments of 40 80 250 500 GB)
  - => 1 TB costs $1K $500 $250 $100, 1 PB costs $1M $500K $250K $10000
- Technology is amazing
  - Flying 747 six inches above the ground at 600mph
  - Reading/writing a strip of postage stamps
  - 5-20 *molecules of gas separating platter from head*
- But…
  - Jet bumps down
  - Bits are so close that cosmic rays/quantum effects change them

Memory hierarchy



- Each level acts as a cache of lower levels

Memory hierarchy: distance analogy



Big Picture

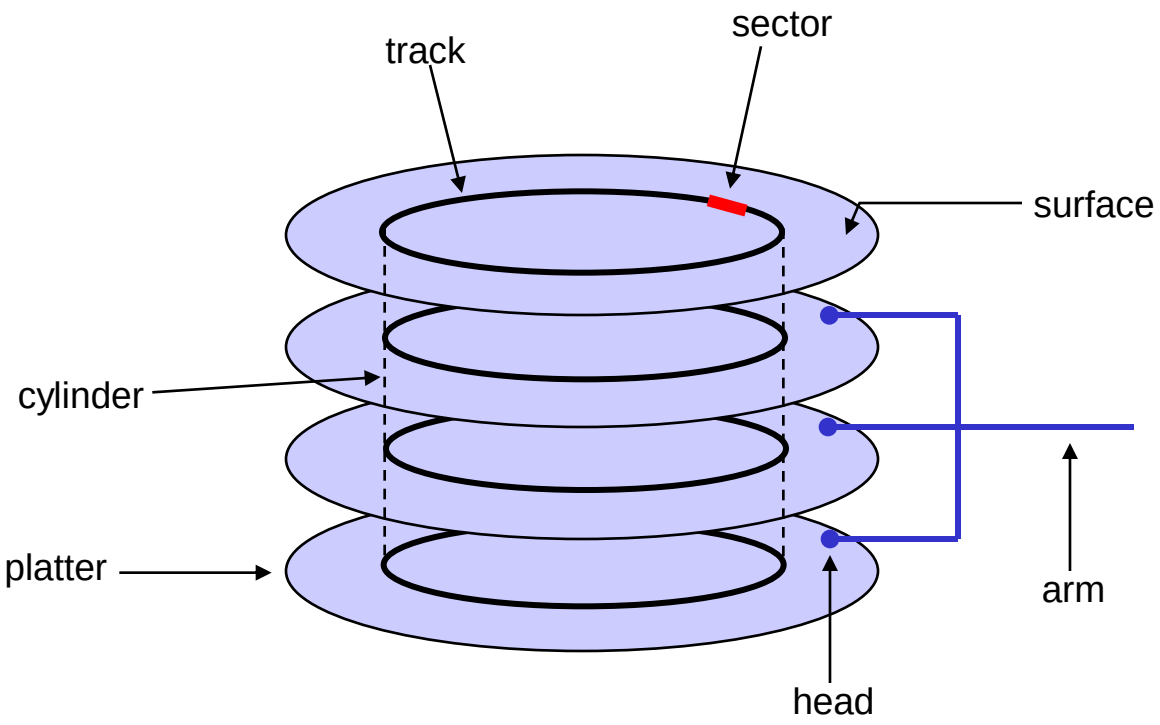
- OS provides abstractions to allow physical HW resources to be shared / protected
  - CPU sharing with threads (virtual CPU)
  - Memory sharing with virtual memory
  - Disk sharing with files

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
  - 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 (head, 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

Example disk characteristics

Lecture Year	2005	2010
Device	IBM Ultrastar 36XP drive	Seagate Constellation ES
Form Factor	3.5”	3.5”
Capacity	36Gb	2Tb
Rotation	7200	7200
Platters	10	6
Surfaces	20	12
Sector Sizes	512-736	512-528
Cylinders	11,494	123,418
Cache	4Mb	16Mb
Transfer Rates	17.9 MB/s (inner) 28.9 MB/s (outer)	75 MB/s (inner) 142 MB/s (outer)
Full Seek	14.5ms	17.5ms
Head Switch	0.3ms	0.3ms

Flash Memory

- No moving parts!
- Writes must be to "clean" cells; no update in place
  - Erasure block: 128–512 KB
  - Erasure time: Several milliseconds
- Write/read page (2-4KB)
  - 50-100 microseconds
- Flash firmware maps logical pages to a physical location
  - Allows firmware to move pages as needed
  - Wear-leveling (can only write a physical page a limited number of times)
  - Avoid pages that no longer work
  - Coalesce live pages during erasure
- Transparent to the device user
- Many file system implications