CSE 451: Operating Systems
Winter 2026

Module 15
I/O System and Secondary Storage

Gary Kimura

A quick word about Pset 3

- Not enough time to cover everything in the lectures
- You'll need to gleam some material from the textbook
- Pset, respect to Storage look at
  - Magnetic Disks
  - Flash Storage
  - Defragmentation and TRIM operations
  - RAID
- With respect to Memory management look at
  - Super pages
  - Inverted Page Tables
- Why? Because they might be on the pset, and we may not have time to talk about them in class

What's Ahead

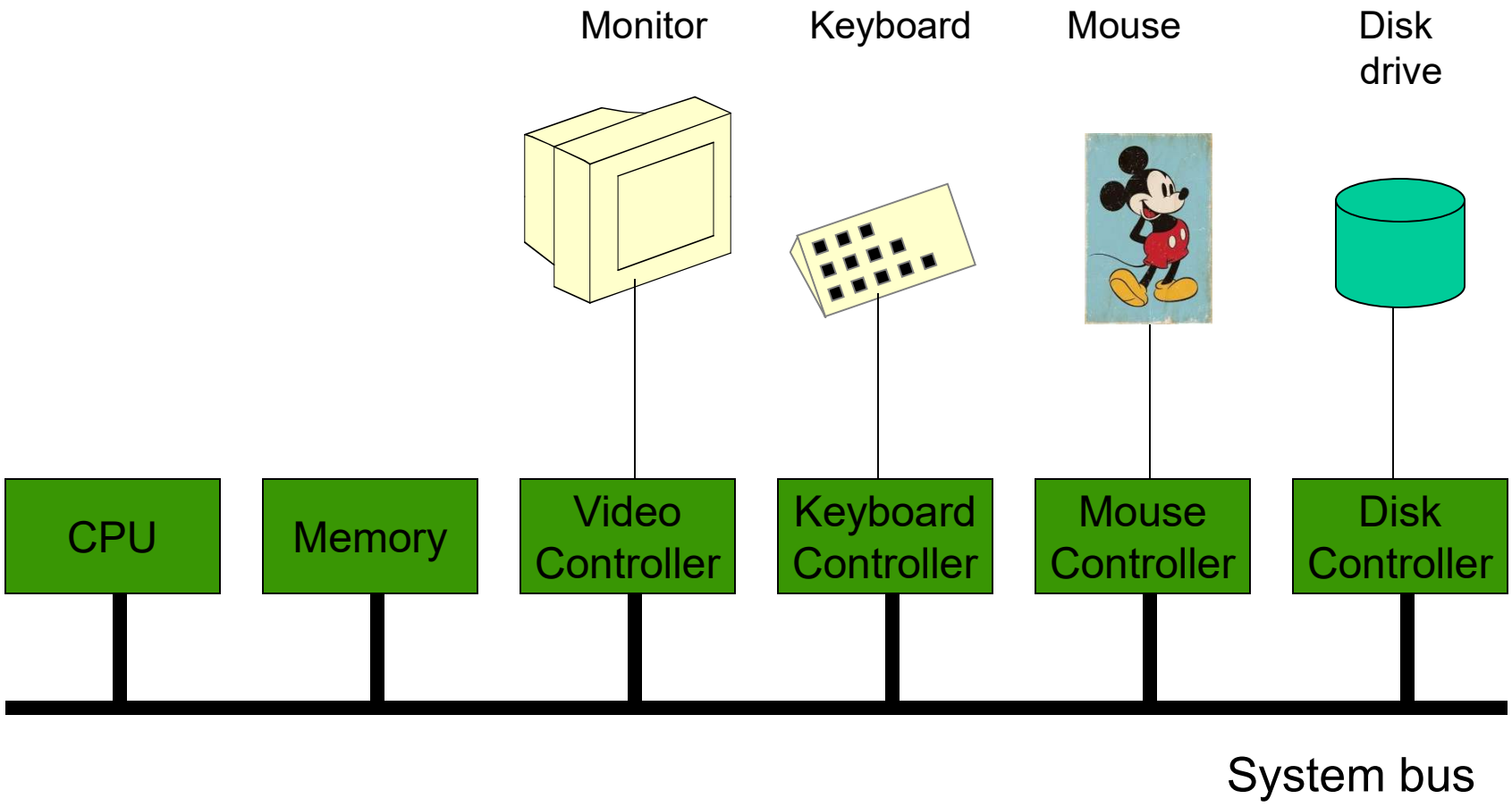
- High level view of the I/O System
- We will concentrate on two major I/O components
- Secondary Storage
  - Disk Drives and SSD
- File System
  - Objectives
  - Programming Interface
  - On-Disk Structure (Storage layout)

- And skip over a lot of other stuff, that we just don't have time to cover

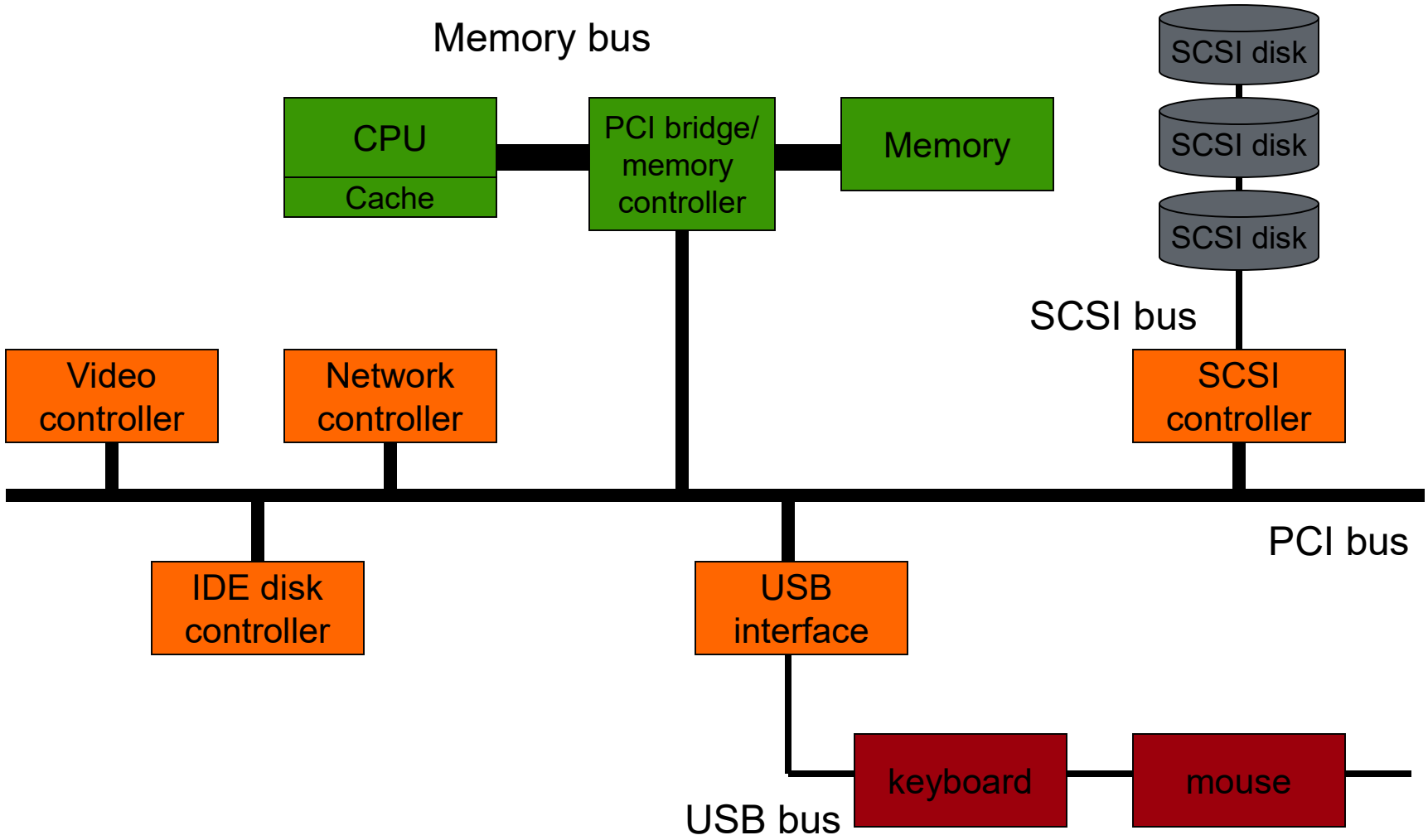
But first the I/O System Hardware Environment

- Major components of a computer system:
  CPU, memories (primary/secondary), I/O system
- I/O devices:
  - Block devices – store information in fixed-sized blocks;
    typical sizes: 128-4096 bytes
  - Character devices – delivers/accepts stream of characters (bytes)
- Device controllers:
  - Connects physical device to system bus (Minicomputers, PCs)
  - Mainframes use a more complex model:
    Multiple buses and specialized I/O computers (I/O channels)
- Communication:
  - Memory-mapped I/O, controller registers
  - Direct Memory Access – DMA
- In the old days, before smart controllers, but we'll skip that part…

I/O Hardware - Single Bus



I/O Hardware - Multiple Buses



Diversity among I/O Devices

The I/O System has to consider device characteristics:

- Data rate:
  - may vary by several orders of magnitude
- Complexity of control:
  - exclusive vs. shared devices
- Unit of transfer:
  - stream of bytes vs. block-I/O
- Data representations:
  - character encoding, error codes, parity conventions
- Error conditions:
  - consequences, range of responses
- Applications:
  - impact on resource scheduling, buffering schemes
- Bottom line: It's complex

The I/O System's role

The I/O System is essentially the glue that holds this all together.
By providing

1. A programming interface for applications to use (hopefully not too big, also not too cumbersome). Some functions are straightforward, like open, close, read, and write. And some are device specific like ioctl's.

2. A way for the myriad of hardware devices to attach to the system through device drivers.

3. Each instance of device has a "unique" name, so that the application can reference the device.

In the Windows, its I/O System, File Systems, and Device Drivers are the largest pieces of code in the OS.

Organization of the I/O Function (General characterization)

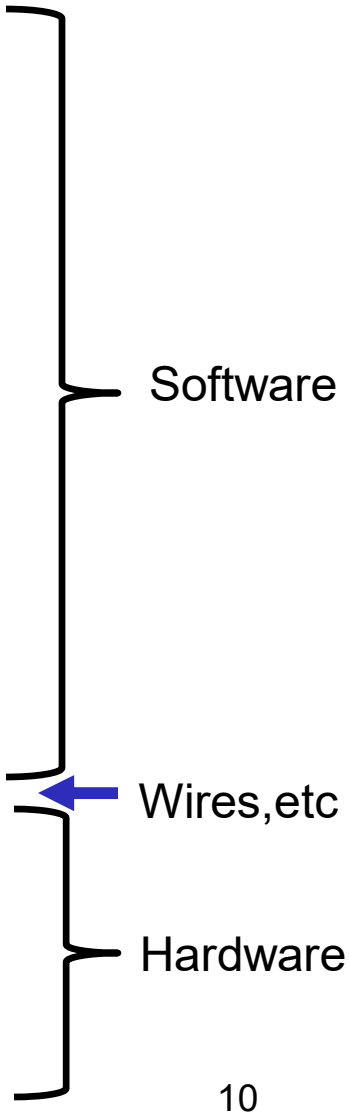
- Programmed I/O with polling:
  - The processor issues an I/O command on behalf of a process
  - The process busy waits for completion of the operation before proceeding
- Interrupt-driven I/O:
  - The processor issues an I/O command and continues to execute
  - The I/O module interrupts the processor when it has finished I/O
  - The initiator process may be suspended pending the interrupt
- Direct memory access (DMA):
  - A DMA module controls exchange of data between I/O module and main memory. What type of address is specified?
  - The processor requests transfer of a block of data from DMA and is interrupted only after the entire block has been transferred

Functional Layers (a simplified view)

- User Level API (stuff like open, close, read, and write

- I/O System (glue that connects this all together in the kernel)
- Maybe a File System (to add some structure to the device)
- Device Driver (knows how to interface with a particular piece of hardware)

- Device Controller (typically the firmware that is included with the piece of hardware)
- The actual Hardware



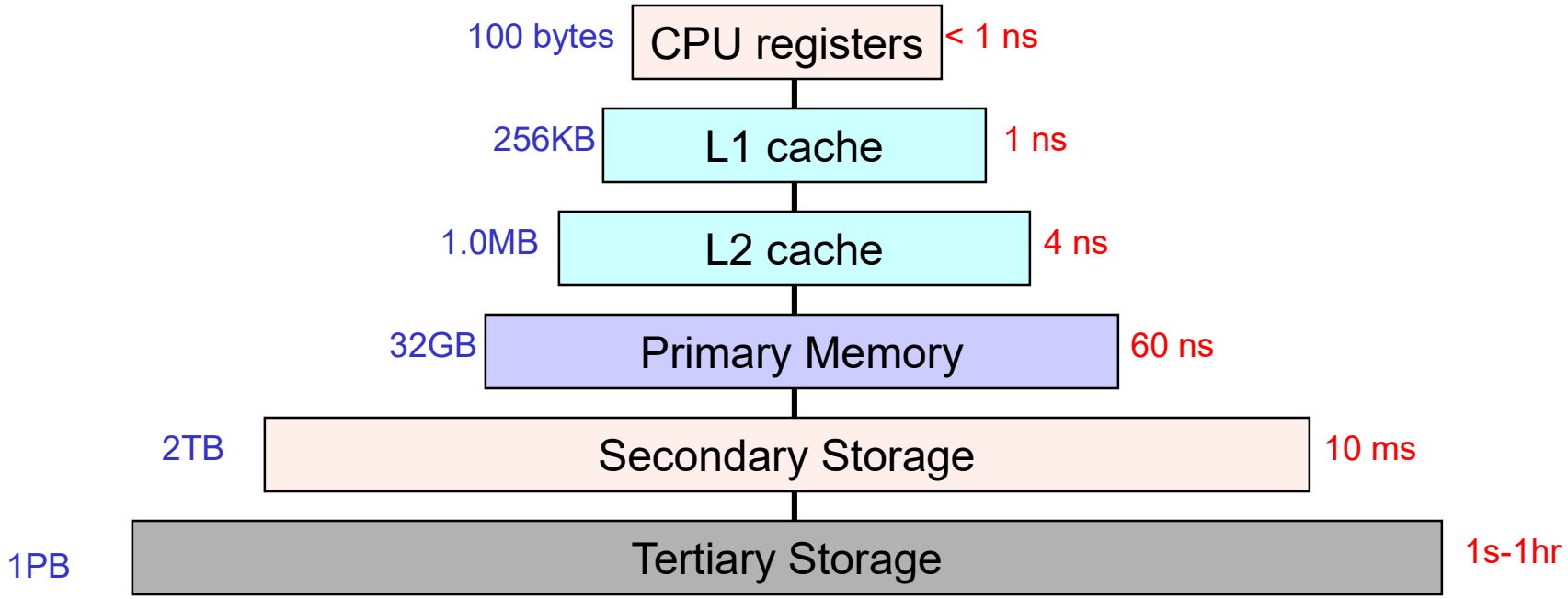
I/O System and Secondary Storage

- That was our quick overview of the I/O System.

  There is a lot more to discuss if time permits. Such as following a full I/O request from the time the User issues the request, until it causes some action at the hardware level, and then the return trip back to the User.

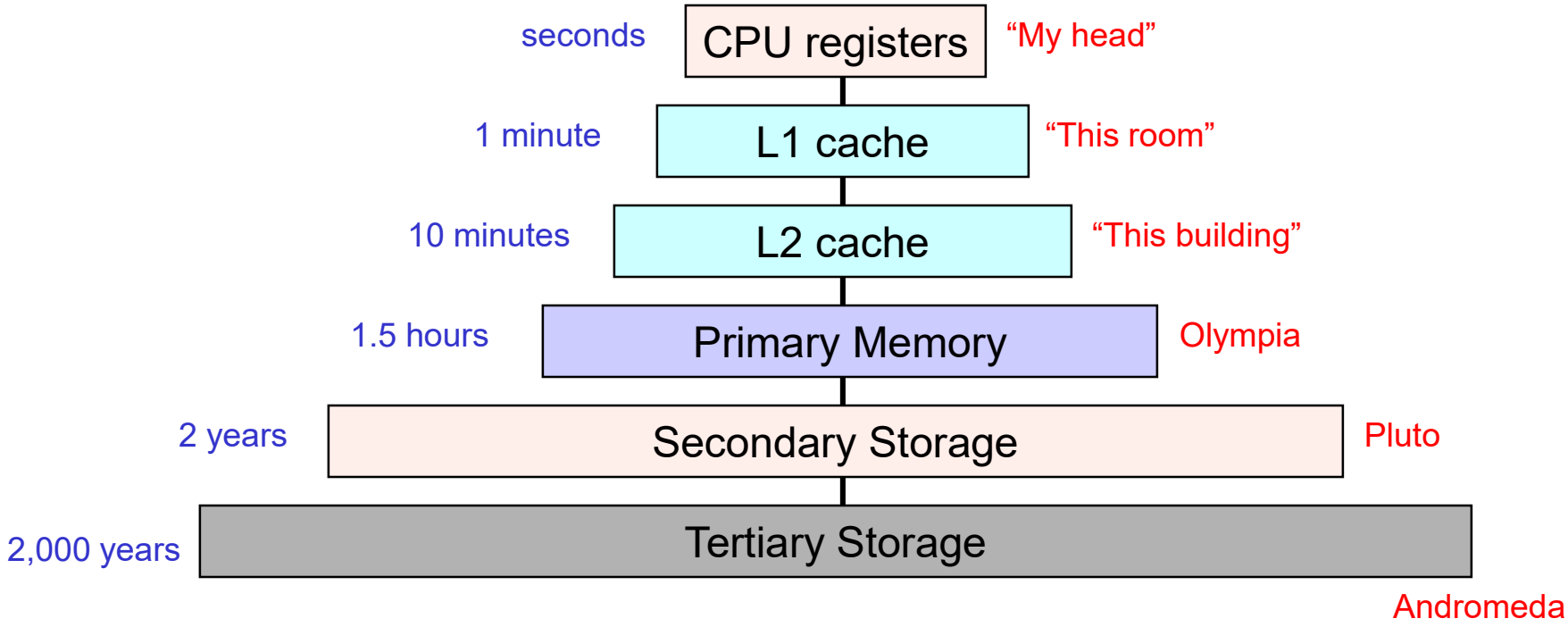
- But now onto Secondary Storage

Memory hierarchy

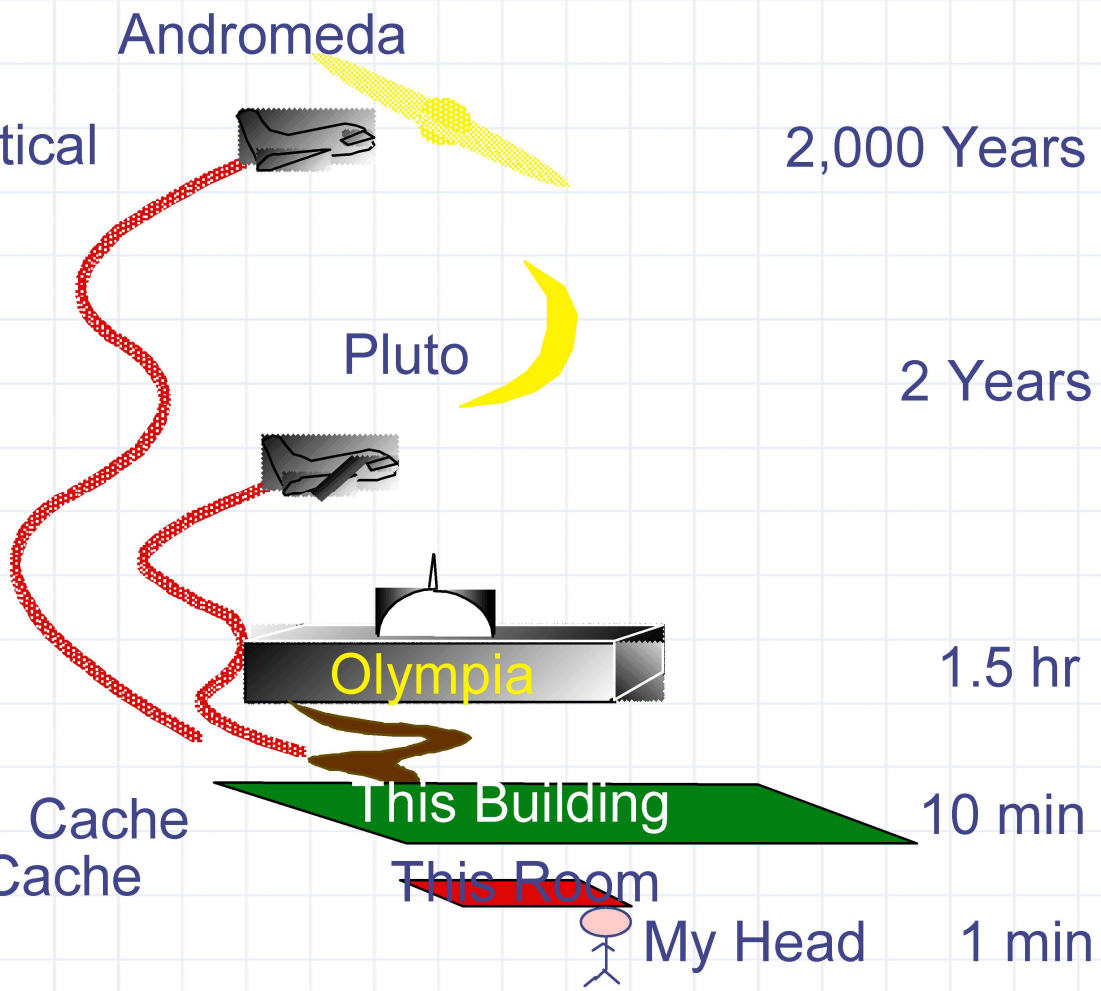


- Each level acts as a cache of lower levels

Memory hierarchy: distance analogy



Storage Latency: How Far Away is the Data?



10^9	Tape /Optical Robot	2,000 Years
10^6	Disk	2 Years
100	Memory	1.5 hr
10	On Board Cache	10 min
2	On Chip Cache	
1	Registers	1 min

Another trip down memory lane …

IBM 2314

About the size of

6 refrigerators

8 x 29MB (M!)

Cost $252,000 in 1965
($2,4 Million in 2024)

Required similar-sized
air conditioner

Less than 0.006% the capacity of
this $100 4TB Hard Drive
(4”x6”x1” item)

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 persistent: data survives power loss
  - it’s slow: milliseconds to access
  - it occasionally fails
  - two prevalent form factors: Hard Disk Drive (HDD), and Solid-State Drive (SSD). What does the future hold?
  - it’s large: (some 2024 numbers)
    - 30TB HDD for about $700
    - 100TB SSD for $40,000
  - it’s cheap for smaller sizes: (some 2024 numbers)
    - 2TB HDD for $73, $0.04/GB
    - 500GB SSD for $50, $10/GB

More Observations

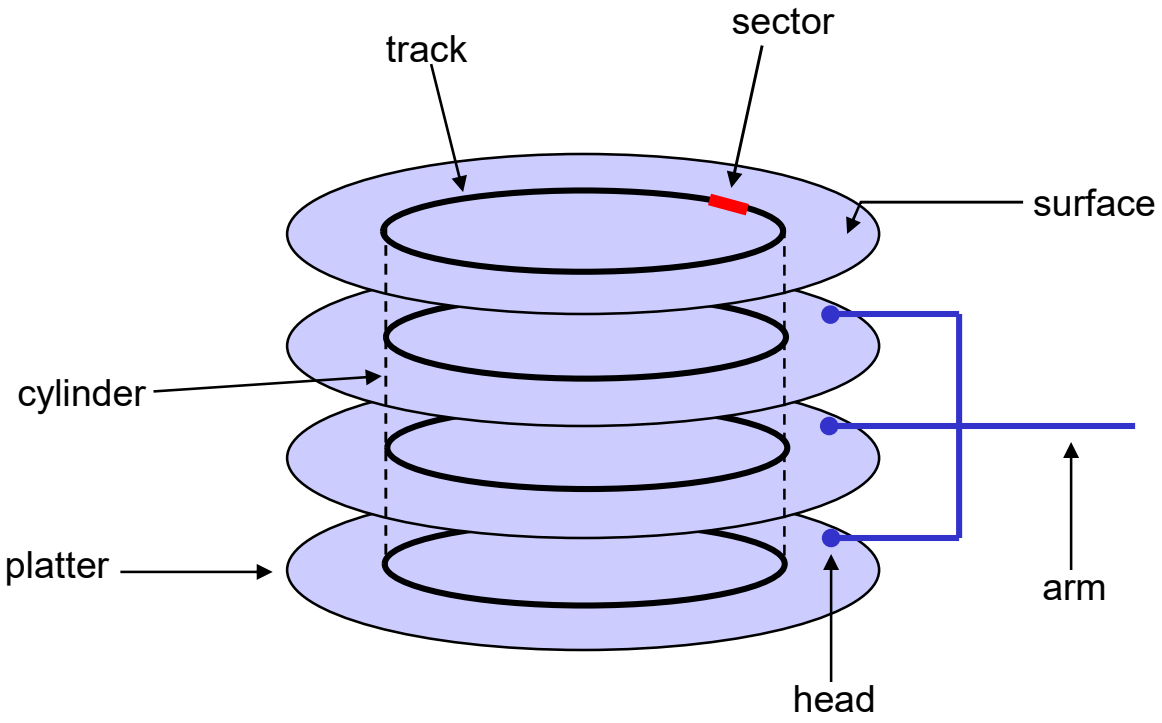
- About HDD and SDD, not including future storage technologies.
- HDD Capacity
  - Has grown from KB to TB
- SSD Trend
  - Supplanting HDD
  - More reliable but still more expensive per unit of storage
- HDD preceded SSD
  - Absent a major redesign of the OS the SSD was shoehorned into the HDD paradigm, even though SSD have different performance characteristics. (This is my personal observation)

Disks and the OS
(a troubled relationship from the beginning)

- 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



Interacting with disks

- In the old days…
  - OS would have to specify cylinder #, sector #, surface #, transfer size
    - i.e., OS needed to know all of the disk parameters
- Modern disks are more sophisticated
  - Disk controllers now provide a higher-level interface and "hide" the actual disk geometry from the OS
    - Disk Controller maps cylinder/surface/sector to a logical block [0..N] that is then exported to the OS.
    - The basic interaction between the OS and the disk is reading and writing logical blocks (typically 512-byte sized blocks, aka Sector).
  - As a result, physical parameters are hidden from OS
    - Both good and bad
    - But some behavioral information does leak through to the OS

Performance Issues

- The performance of HDD suffer from having to mechanically moving parts.
  - Performance is improved by limiting the amount of seeking needed to read or write data.
  - Defragmentation helps improve performance
- Without going into detail, erasing data on a SSD involves erasing an entire block.
  - Performance is improved if the SSD has a stockpile of "clean" ready to us blocks.
  - SSD can occasionally garbage collect (aka Trim) blocks that are no longer in use, making them ready for reuse.
  - Note: Defragmenting an SSD will only shorten its life.

Drives and the File System

- There is a lot more to say about HDD and their peculiar characteristics, but some of it is antiquated with SSD.
  - A common peculiarity of HDD is that accessing contiguous blocks is faster than noncontiguous blocks, and accessing lower numbered logical blocks typically are faster than higher numbered blocks. This is a geometry issue that SSD avoid.
  - SSD have large erasure blocks and do wear leveling that HDD do not do. This characteristic is pretty much hidden from the OS. But can/should the OS exploit this behavior?
- From the File System perspective secondary storage are sequentially numbered logical blocks that it can read and write.
- The File System is our next topic.