

# CS 110

## Computer Architecture

I/O: DMA, Disks, Networking

Instructor:

Sören Schwertfeger and Chundong Wang

<https://robotics.shanghaitech.edu.cn/courses/ca/21s>

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkeley's CS61C

# Review

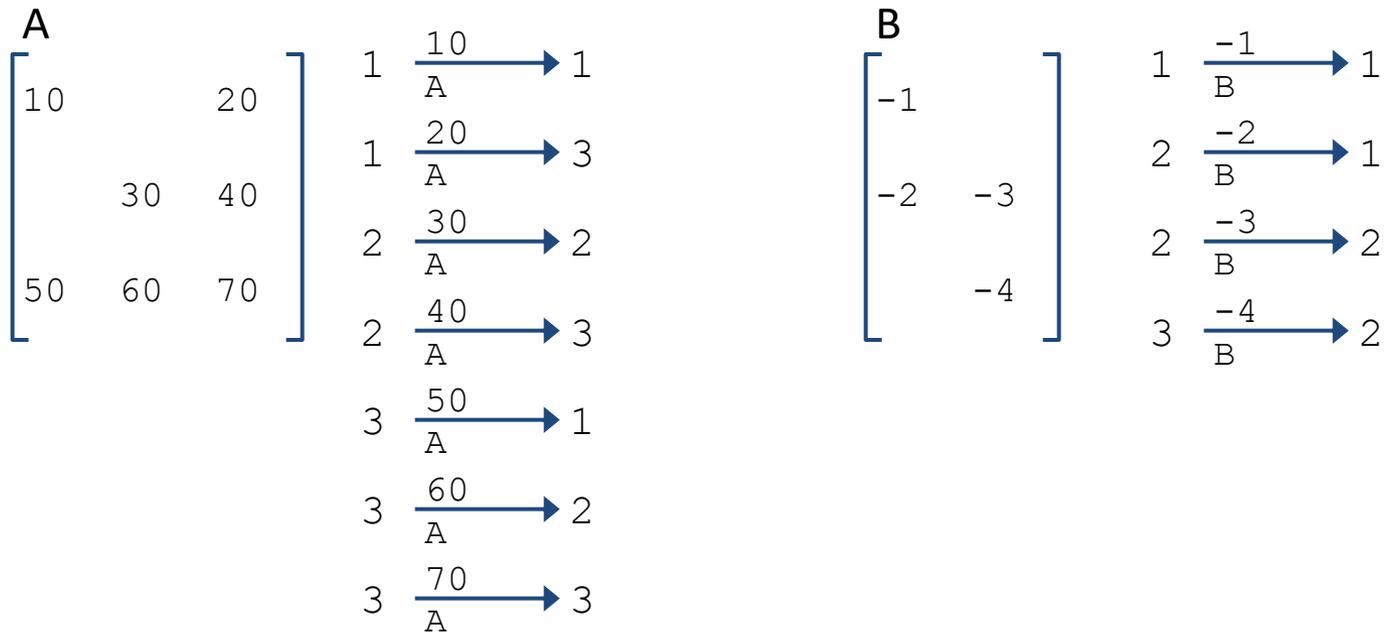
- WSC
  - A giant computer
- Map/Reduce
  - Map
  - Reduce

# Review: Sparse Matrices with Map/Reduce

$$\begin{matrix} \text{A} \\ \left[ \begin{array}{ccc} 10 & & 20 \\ & 30 & 40 \\ 50 & 60 & 70 \end{array} \right] \end{matrix} \times \begin{matrix} \text{B} \\ \left[ \begin{array}{cc} -1 & \\ -2 & -3 \\ & -4 \end{array} \right] \end{matrix} = \begin{matrix} \text{C} \\ \left[ \begin{array}{cc} -10 & -80 \\ -60 & -250 \\ -170 & -460 \end{array} \right] \end{matrix}$$

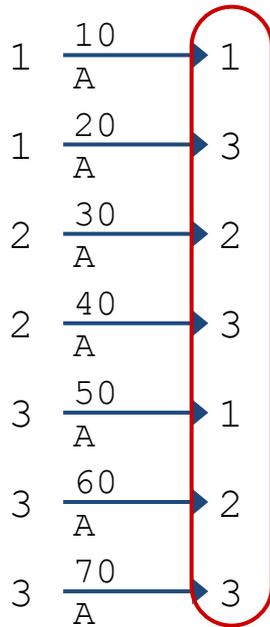
- Task: Compute product  $C = A \cdot B$
- Assume most matrix entries are 0
- Motivation
  - Core problem in scientific computing
  - Challenging for parallel execution
  - Demonstrate expressiveness of Map/Reduce

# Computing Sparse Matrix Product

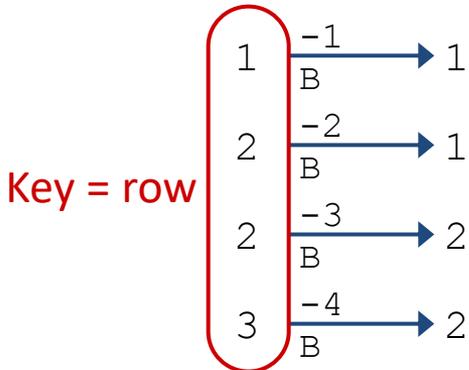


- Represent matrix as list of nonzero entries  
    ⟨row, col, value, matrixID⟩
- Strategy
  - **Phase 1: Compute all products  $a_{i,k} \cdot b_{k,j}$**
  - **Phase 2: Sum products for each entry  $i,j$**
  - **Each phase involves a Map/Reduce**

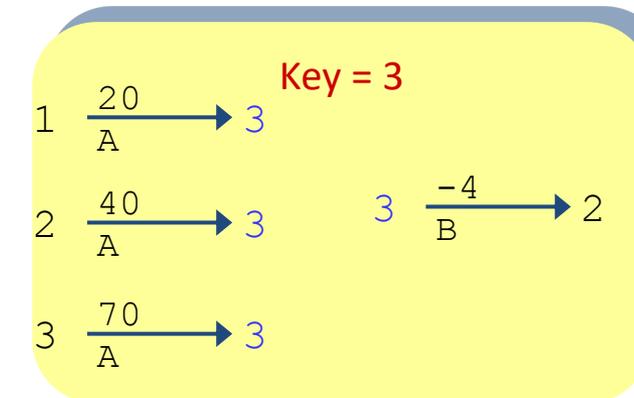
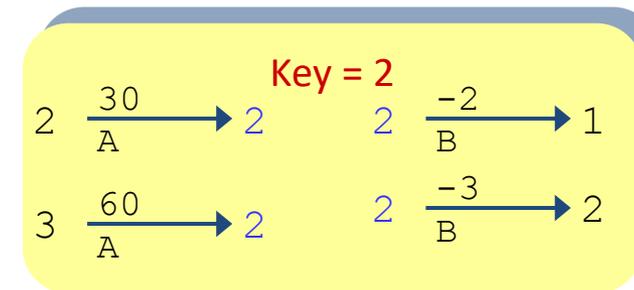
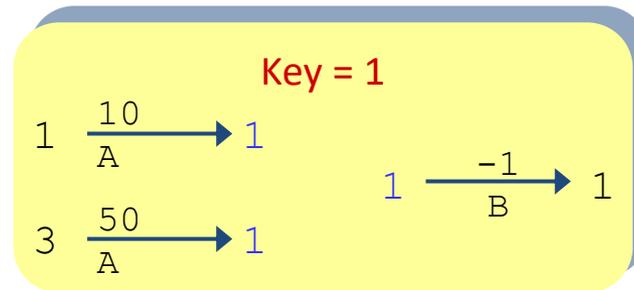
# Phase 1 Map of Matrix Multiply



Key = col



Key = row



– Group values  $a_{i,k}$  and  $b_{k,j}$  according to key  $k$

# Phase 1 “Reduce” of Matrix Multiply

Key = 1

$$\begin{array}{l} 1 \xrightarrow[A]{10} 1 \\ 3 \xrightarrow[A]{50} 1 \end{array} \quad \times \quad \begin{array}{l} 1 \xrightarrow[B]{-1} 1 \end{array}$$

Key = 2

$$\begin{array}{l} 2 \xrightarrow[A]{30} 2 \\ 3 \xrightarrow[A]{60} 2 \end{array} \quad \times \quad \begin{array}{l} 2 \xrightarrow[B]{-2} 1 \\ 2 \xrightarrow[B]{-3} 2 \end{array}$$

Key = 3

$$\begin{array}{l} 1 \xrightarrow[A]{20} 3 \\ 2 \xrightarrow[A]{40} 3 \\ 3 \xrightarrow[A]{70} 3 \end{array} \quad \times \quad \begin{array}{l} 3 \xrightarrow[B]{-4} 2 \end{array}$$

$$1 \xrightarrow[C]{-10} 1$$

$$3 \xrightarrow[C]{-50} 1$$

$$2 \xrightarrow[C]{-60} 1$$

$$2 \xrightarrow[C]{-90} 2$$

$$3 \xrightarrow[C]{-120} 1$$

$$3 \xrightarrow[C]{-180} 2$$

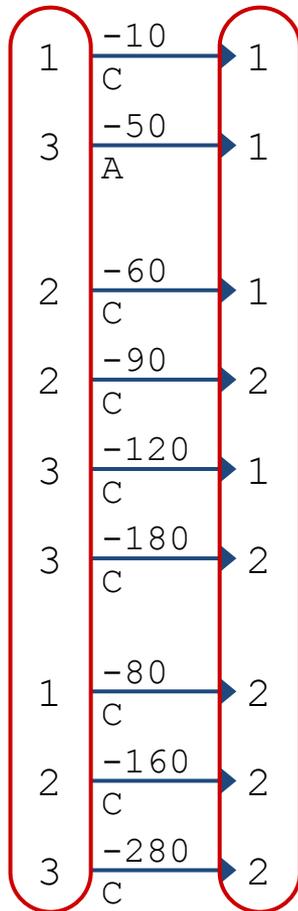
$$1 \xrightarrow[C]{-80} 2$$

$$2 \xrightarrow[C]{-160} 2$$

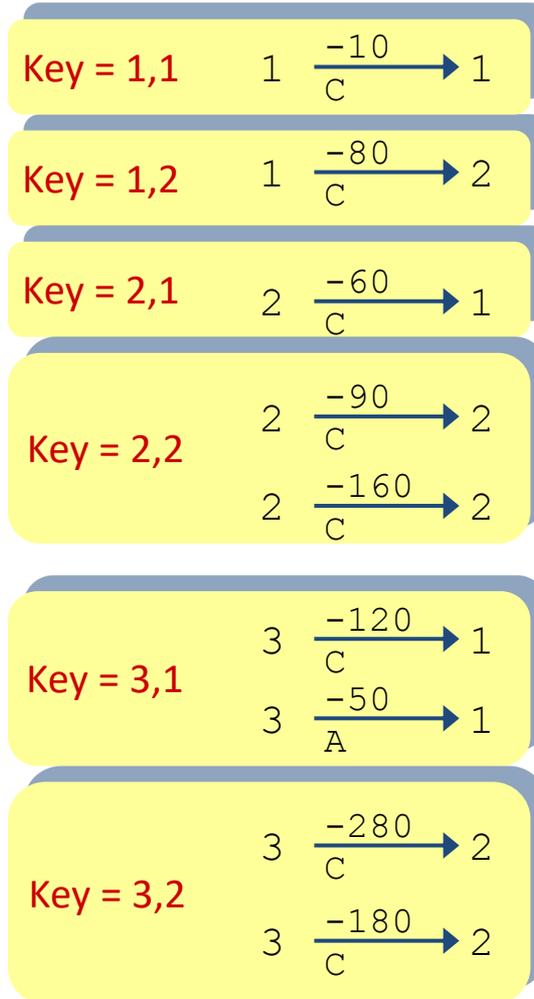
$$3 \xrightarrow[C]{-280} 2$$

– Generate all products  $a_{i,k} \cdot b_{k,j}$

# Phase 2 Map of Matrix Multiply



Key = row,col



– Group products  $a_{i,k} \cdot b_{k,j}$  with matching values of  $i$  and  $j$

# Phase 2 Reduce of Matrix Multiply

Key = 1,1     $1 \xrightarrow{\frac{-10}{C}} 1$

Key = 1,2     $1 \xrightarrow{\frac{-80}{C}} 2$

Key = 2,1     $2 \xrightarrow{\frac{-60}{C}} 1$

Key = 2,2     $2 \xrightarrow{\frac{-90}{C}} 2$

$2 \xrightarrow{\frac{-160}{C}} 2$

Key = 3,1     $3 \xrightarrow{\frac{-120}{C}} 1$

$3 \xrightarrow{\frac{-50}{A}} 1$

Key = 3,2     $3 \xrightarrow{\frac{-280}{C}} 2$

$3 \xrightarrow{\frac{-180}{C}} 2$

$1 \xrightarrow{\frac{-10}{C}} 1$

$1 \xrightarrow{\frac{-80}{C}} 2$

$2 \xrightarrow{\frac{-60}{C}} 1$

$2 \xrightarrow{\frac{-250}{C}} 2$

$3 \xrightarrow{\frac{-170}{C}} 1$

$3 \xrightarrow{\frac{-460}{C}} 2$

C

$$\begin{bmatrix} -10 & -80 \\ -60 & -250 \\ -170 & -460 \end{bmatrix}$$

– Sum products to get final entries

# Lessons from Sparse Matrix Example

- Associative matching is powerful communication primitive
  - Intermediate step in Map/Reduce
- Similar Strategy Applies to Other Problems
  - Shortest path in graph
  - Database join
- Many Performance Considerations
  - *Pairwise Element Computation with MapReduce* (HPDC '10, by Kiefer, Volk, Lehner from TU Dresden)
  - Should do systematic comparison to other sparse matrix implementations (L19 of CS110-21s)

# Review: I/O

- “Memory mapped I/O”: Device control/data registers mapped to CPU address space
- CPU synchronizes with I/O device:
  - Polling
  - Interrupts
- “Programmed I/O”:
  - CPU execs lw/sw instructions for all data movement to/from devices
  - CPU spends time doing 2 things:
    1. Getting data from device to main memory
    2. Using data to compute

# Working with real devices

- “Memory mapped I/O”: Device control/data registers mapped to CPU address space
- CPU synchronizes with I/O device:
  - Polling
  - Interrupts
- ~~“Programmed I/O”~~: **DMA**
  - ~~CPU execs lw/sw instructions for all data movement to/from devices~~
  - CPU spends time doing ~~2 things~~:
    1. ~~Getting data from device to main memory~~
    2. Using data to compute

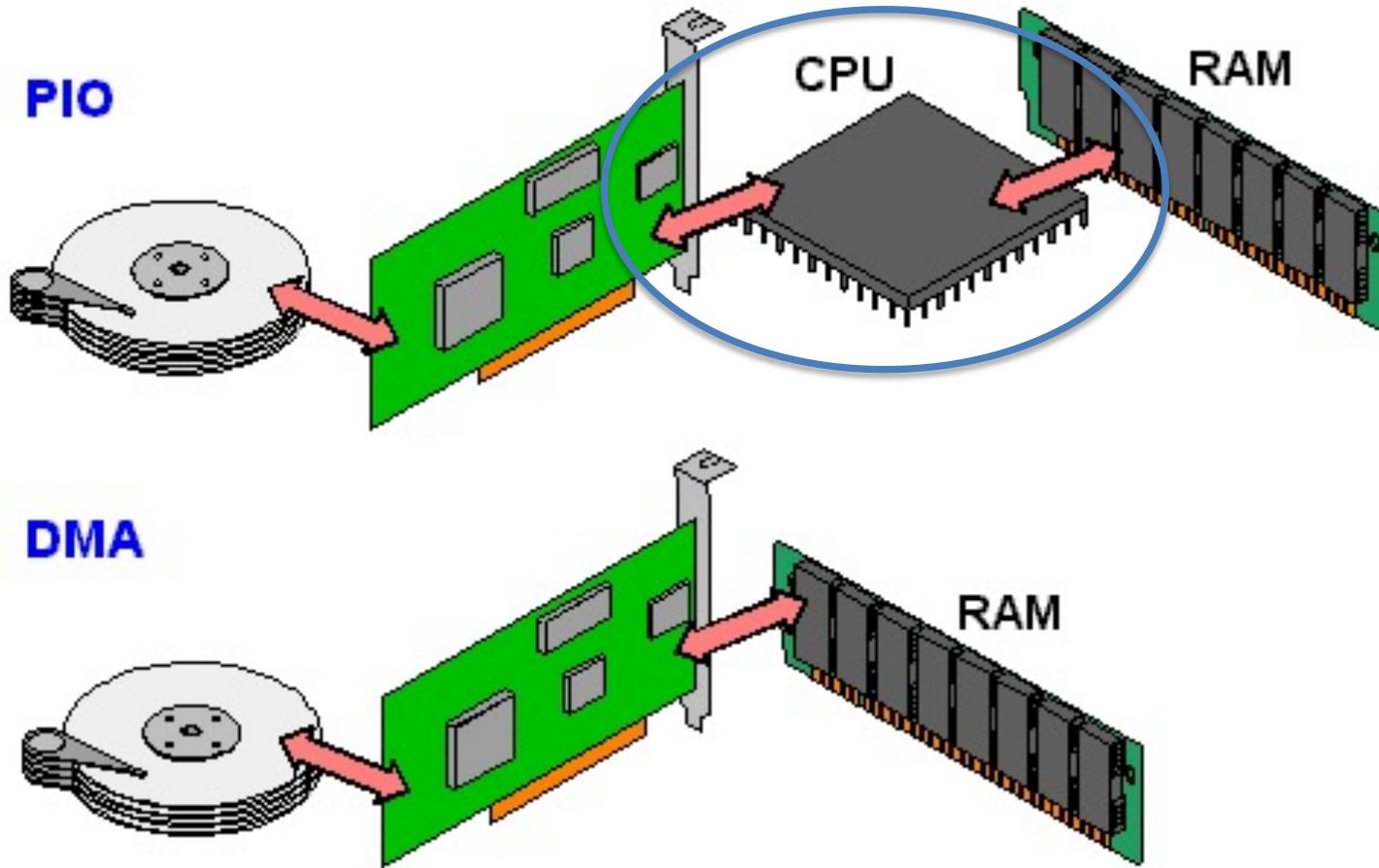
# Agenda

- Direct Memory Access (DMA)
- Disks
- Networking

# What's wrong with Programmed I/O?

- Not ideal because ...
  1. CPU has to execute all transfers, could be doing other work
  2. Device speeds don't align well with CPU speeds
  3. Energy cost of using beefy general-purpose CPU where simpler hardware would suffice
- Until now CPU has sole control of main memory

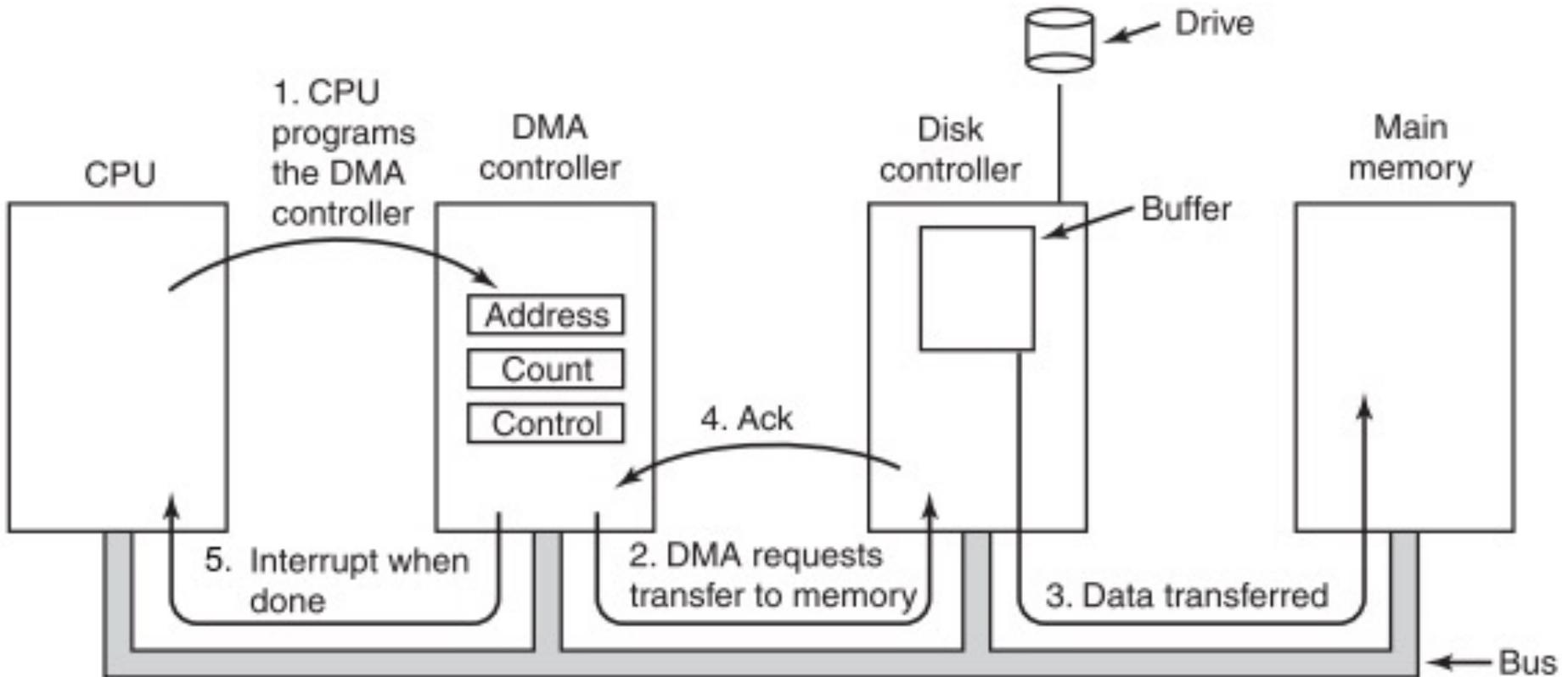
# PIO vs. DMA



# Direct Memory Access (DMA)

- Allows I/O devices to directly read/write main memory
- New Hardware: the DMA Engine
- DMA engine contains registers written by CPU:
  - Memory address to place data
  - # of bytes
  - I/O device #, direction of transfer
  - Unit of transfer, amount to transfer per burst

# Operation of a DMA Transfer



**Figure 5-4.** Operation of a DMA transfer.

[From Section 5.1.4 Direct Memory Access in *Modern Operating Systems* by Andrew S. Tanenbaum, Herbert Bos, 2014]

# DMA: Incoming Data

1. Receive interrupt from device
2. CPU takes interrupt, begins transfer
  - Instructs DMA engine/device to place data @ certain address
3. Device/DMA engine handle the transfer
  - CPU is free to execute other things
4. Upon completion, Device/DMA engine interrupt the CPU again

# DMA: Outgoing Data

1. CPU decides to initiate transfer, confirms that external device is ready
2. CPU begins transfer
  - Instructs DMA engine/device that data is available @ certain address
3. Device/DMA engine handle the transfer
  - CPU is free to execute other things
4. Device/DMA engine interrupt the CPU again to signal completion

# DMA: Some new problems

- Where in the memory hierarchy do we plug in the DMA engine? Two extremes:
  - Between CPU and L1:
    - Pro: Free coherency
    - Con: Trash the CPU's working set with transferred data
  - Between Last-level cache and main memory:
    - Pro: Don't mess with caches
    - Con: Need to explicitly manage coherency

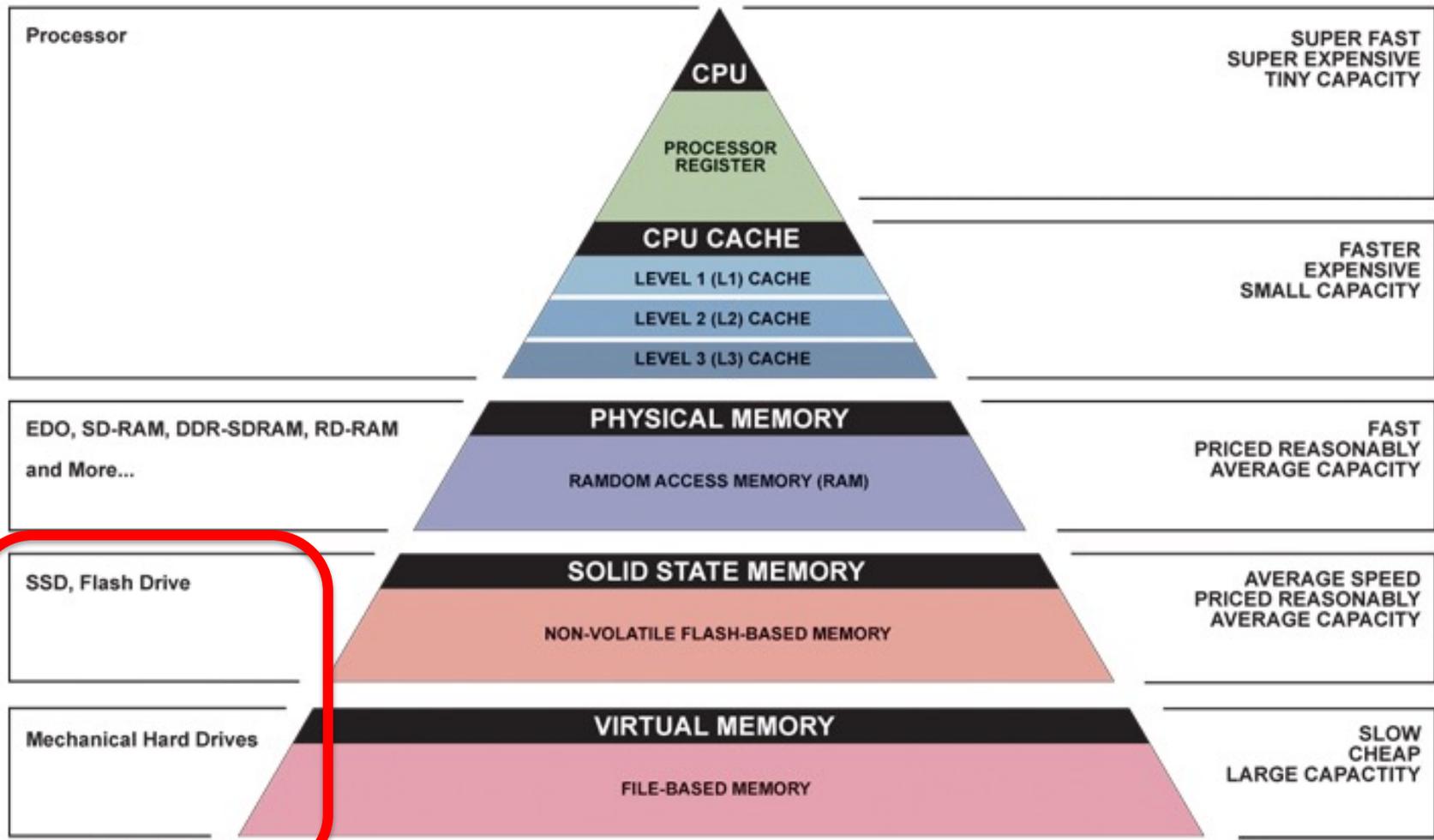
# DMA: Some new problems

- How do we arbitrate between CPU and DMA Engine/Device access to memory?
- Three options:
  - Burst Mode
    - Start transfer of data block, CPU cannot access memory in the meantime
  - Cycle Stealing Mode
    - DMA engine transfers a byte, releases control, then repeats - interleaves processor/DMA engine accesses
  - Transparent Mode
    - DMA transfer only occurs when CPU is not using the system bus

# Agenda

- Direct Memory Access (DMA)
- Disks
- Networking

# Computer Memory Hierarchy



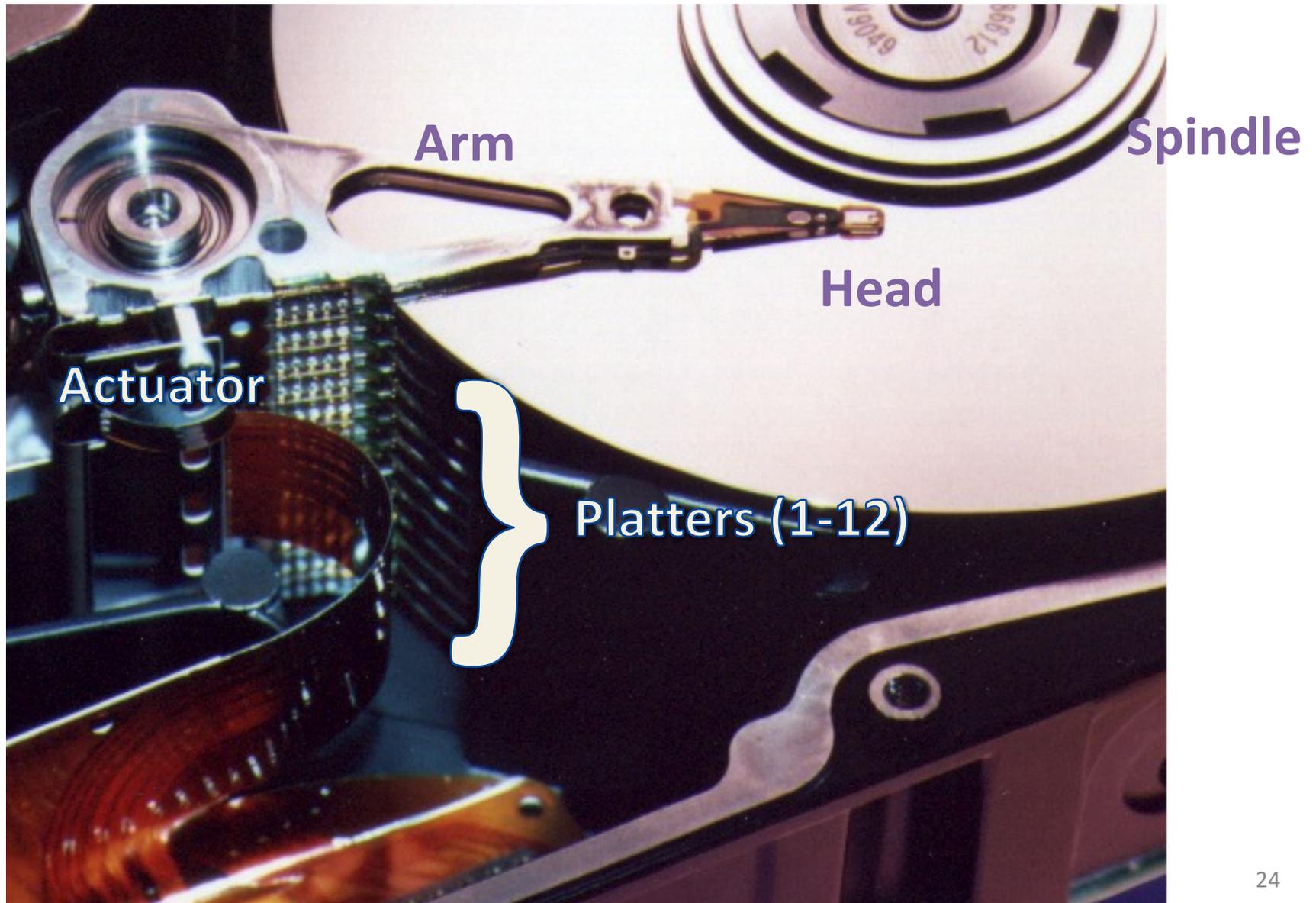
**Today**

▲ Simplified Computer Memory Hierarchy  
Illustration: Ryan J. Leng

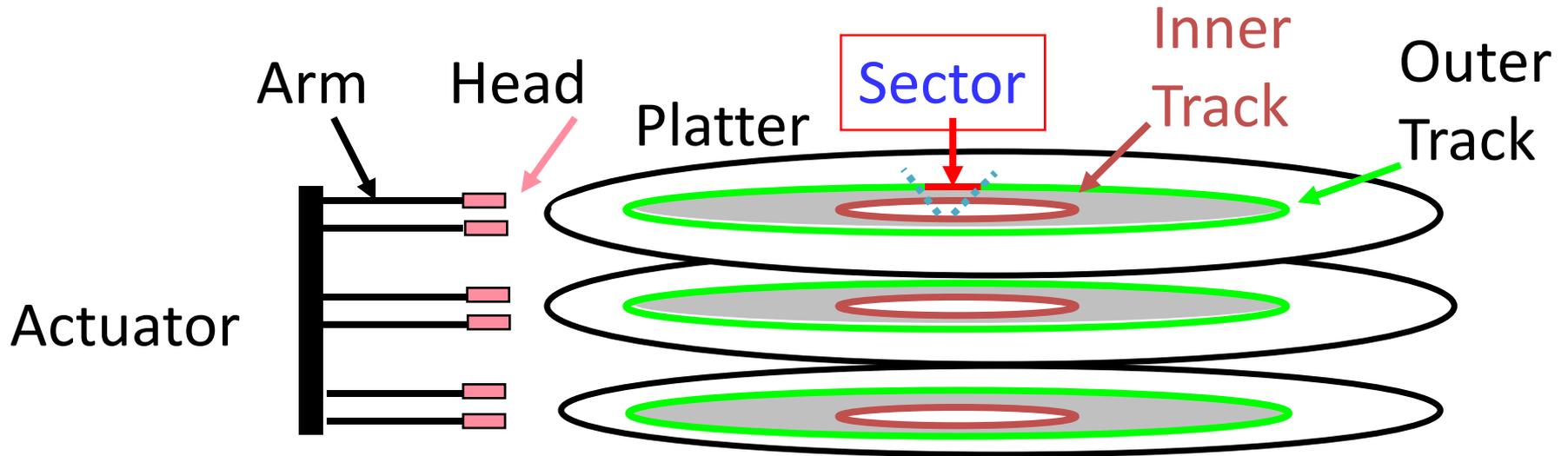
# Magnetic Disk – common I/O device

- A kind of computer memory
  - Information stored by magnetizing ferrite material on surface of rotating disk
    - similar to tape recorder except digital rather than analog data
- A type of non-volatile storage
  - Retains its value without applying power to disk.
- Magnetic Disk
  1. Hard Disk Drives (HDD) – faster, more dense, non-removable.
- Purpose in computer systems (Hard Drive):
  1. Working file system + long-term backup for files
  2. Secondary “backing store” for main-memory. Large, inexpensive, slow level in the memory hierarchy (virtual memory)

# Photo of Disk Head, Arm, Actuator



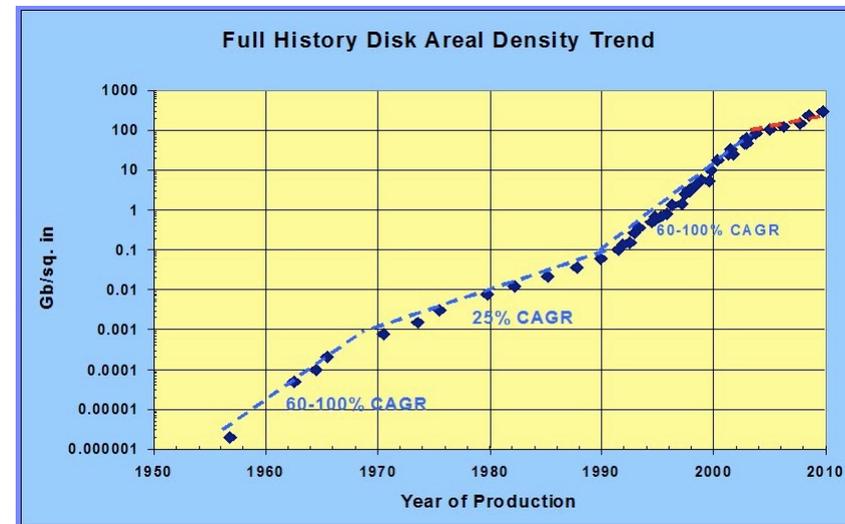
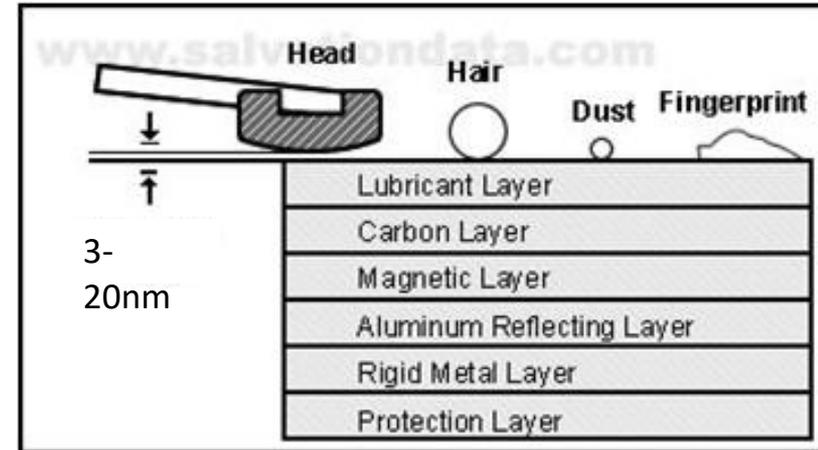
# Disk Device Terminology



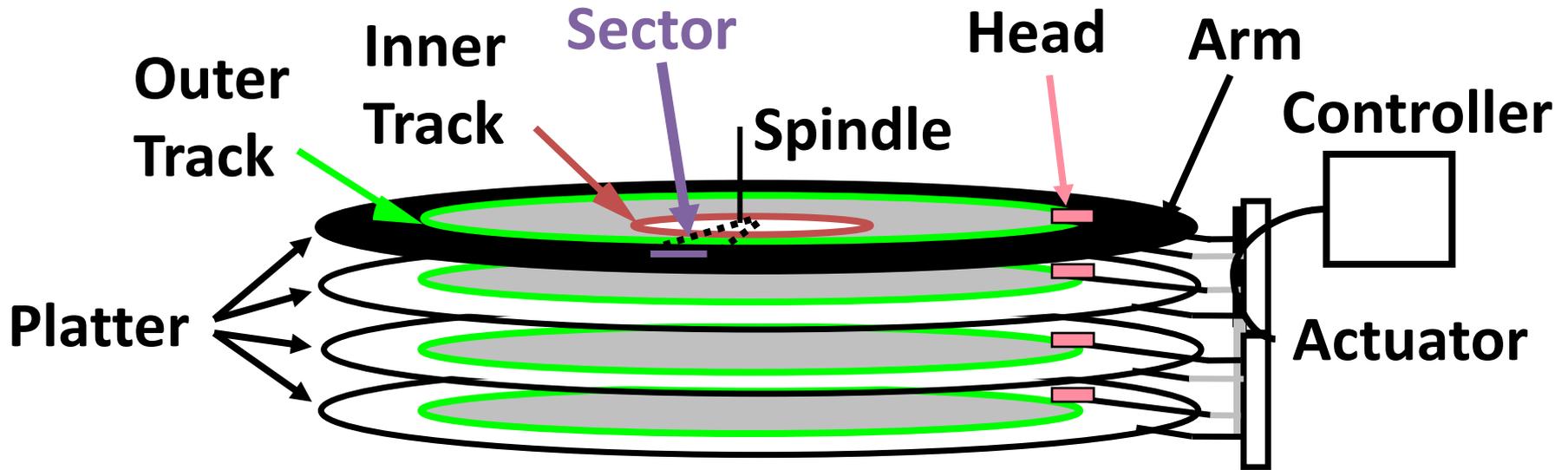
- Several platters, with information recorded magnetically on both surfaces (usually)
- Bits recorded in **tracks**, which in turn divided into **sectors** (e.g., 512 Bytes)
- **Actuator** moves **head** (end of **arm**) over track ("**seek**"), wait for **sector** rotate under **head**, then read or write

# Hard Drives are Sealed. Why?

- The closer the head to the disk, the smaller the “spot size” and thus the denser the recording.
  - Measured in Gbit/in<sup>2</sup>
  - ~900 Gbit/in<sup>2</sup> is state of the art
  - Started out at 2 Kbit/in<sup>2</sup>
  - ~450,000,000x improvement in ~60 years
- Disks are sealed to keep the dust out.
  - Heads are designed to “fly” at around 3-20nm above the surface of the disk.
  - 99.999% of the head/arm weight is supported by the air bearing force (air cushion) developed between the disk and the head.



# Disk Device Performance (1/2)



- **Disk Access Time = Seek Time + Rotation Time + Transfer Time + Controller Overhead**
  - Seek Time = time to position the head assembly at the proper track
  - Rotation Time = time for the disk to rotate to the point where the first sectors of the block to access reach the head
  - Transfer Time = time taken by the sectors of the block and any gaps between them to rotate past the head

# Disk Device Performance (2/2)

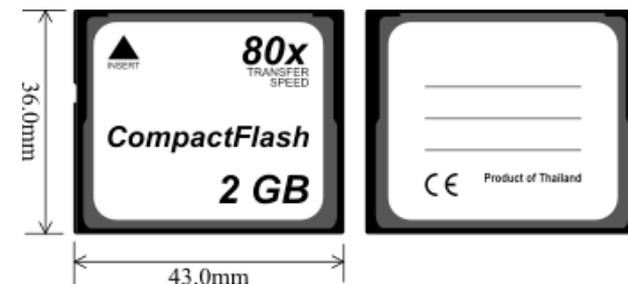
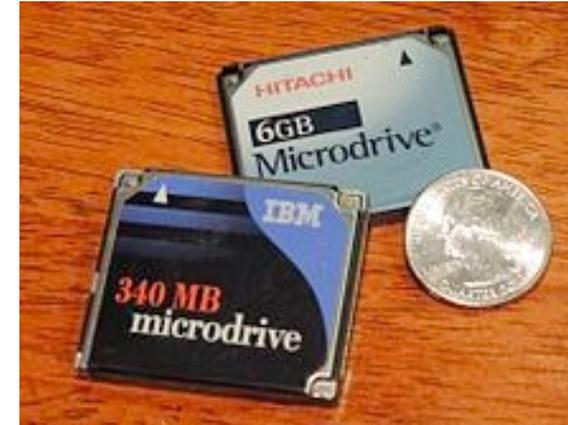
- Average values to plug into the formula:
- Rotation Time: Average distance of sector from head?
  - 1/2 time of a rotation
    - 7200 Revolutions Per Minute → 120 Rev/sec
    - 1 revolution = 1/120 sec → 8.33 milliseconds
    - 1/2 rotation (revolution) → 4.17 ms
- Seek time: Average no. tracks to move arm?
  - Number of tracks / 3
    - Check Page 9 at <http://pages.cs.wisc.edu/~remzi/OSFEP/file-disks.pdf>
  - Then, seek time = number of tracks moved × time to move across one track

# But wait!

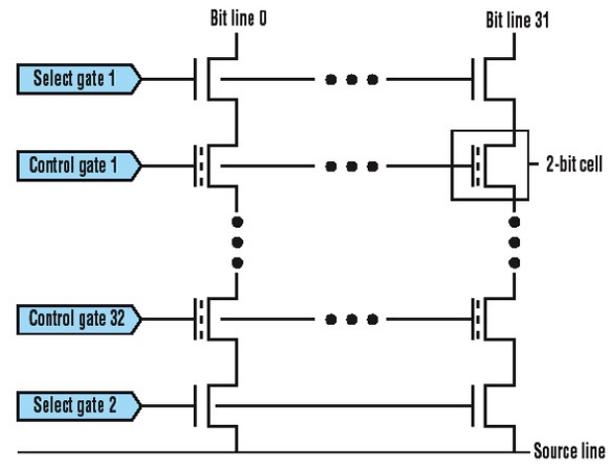
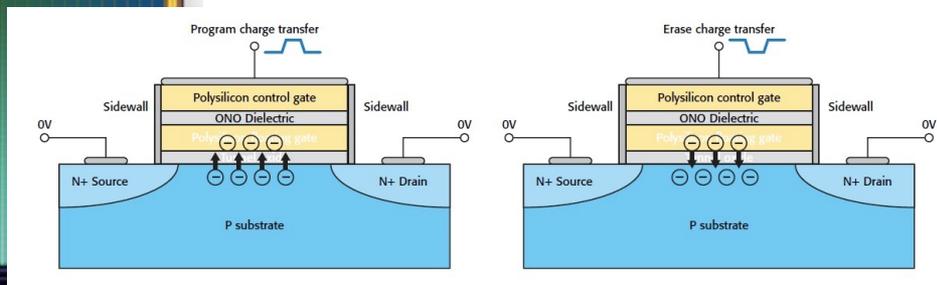
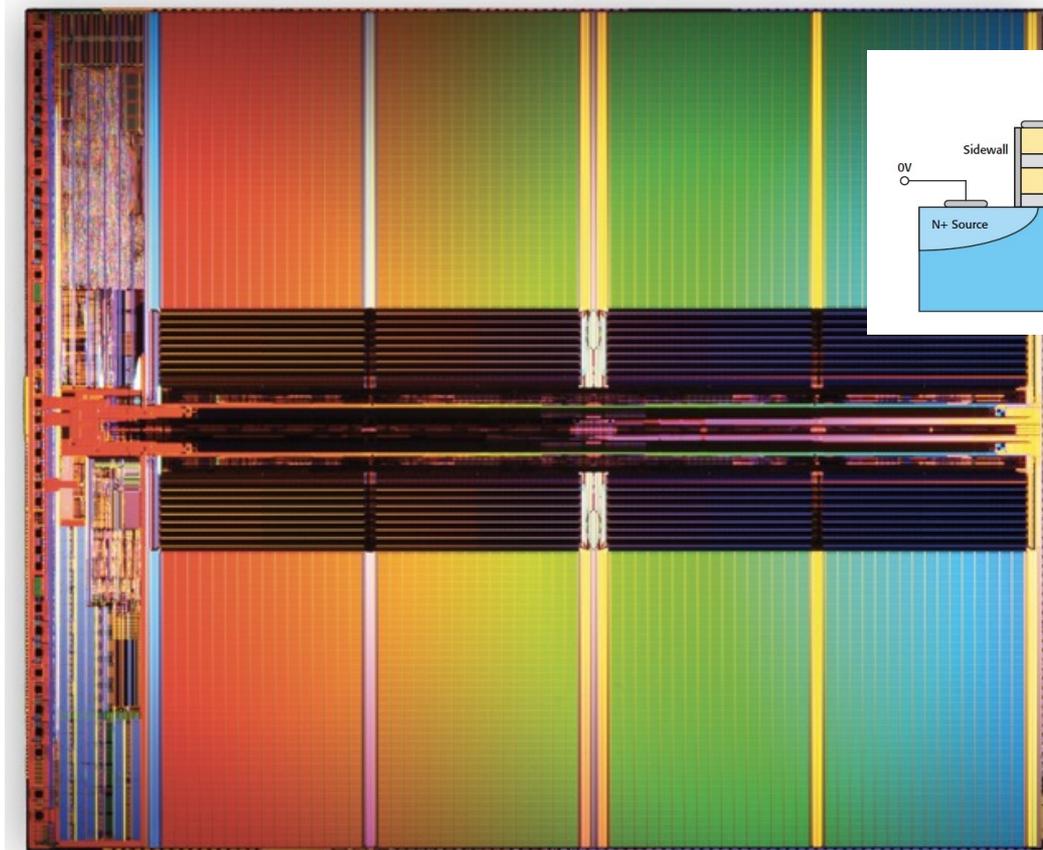
- Performance estimates are different in practice:
- Many disks have on-disk caches, which are completely hidden from the outside world
  - Previous formula completely replaced with on-disk cache access time

# Where does Flash memory come in?

- >15 years ago: Microdrives and Flash memory (e.g., CompactFlash) went head-to-head
  - Both non-volatile (retains contents without power supply)
  - Flash benefits: lower power, seldom crashes (no moving parts, need to spin  $\mu$ drives up/down)
  - Disk cost = fixed cost of motor + arm mechanics, but actual magnetic media cost very low
  - Flash cost = most cost/bit of flash chips
  - Over time, cost/bit of flash came down, became cost competitive



# Flash Memory / SSD Technology



2. Micron's triple-level cell (TLC) flash memory stores 3 bits of data in each transistor.

In the basic functional block used in multilevel NAND flash memories, 32 rows of bit lines and 32 control-gate lines form a building block that's repeated many times to form the memory array. The select gate lines are used with the control gate lines to control access to the array.

- NMOS transistor with an additional conductor between gate and source/drain which “traps” electrons. The presence/absence is a 1 or 0
- Memory cells can withstand a limited number of program-erase cycles. Controllers use a technique called *wear leveling* to distribute writes as evenly as possible across all the flash blocks in the SSD.

# What did Apple put in its iPods?

Toshiba flash  
2 GB

Samsung flash  
16 GB

Toshiba **1.8-inch HDD**  
80, 120, 160 GB

Toshiba flash  
32, 64, 128 GB



shuffle



nano



classic



touch

# Flash Memory in Smart Phones

iPhone 7: up to 256 GB



# Flash Memory in Laptops – Solid State Drive (SSD)

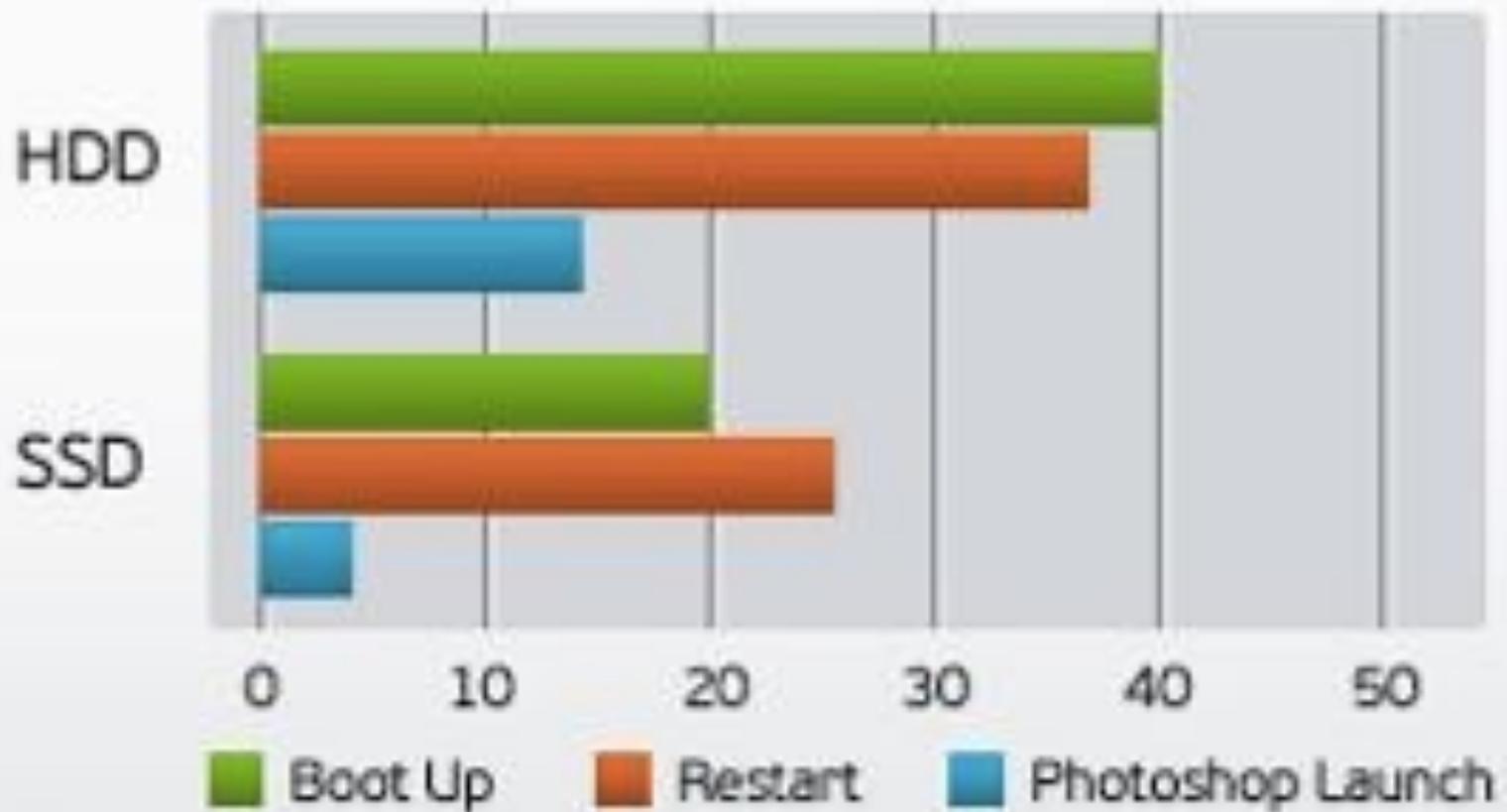


capacities up to 2TB

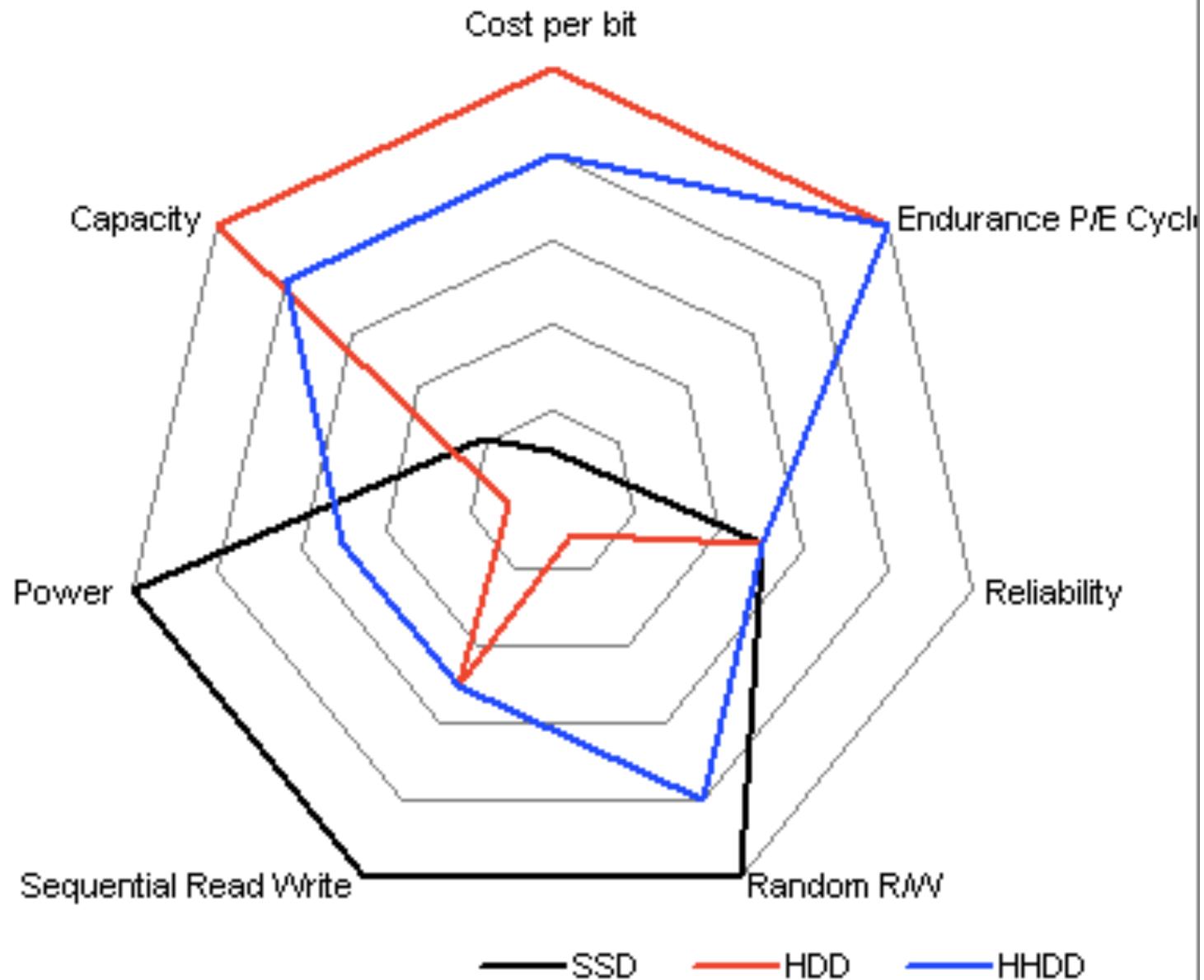


# HDD vs SSD speed

Time in Seconds (Lower is Better)



# Hard Drive vs. SSD vs. Hybrid HDD



# Agenda

- Direct Memory Access (DMA)
- Disks
- Networking

# Networks: Talking to the Outside World

- Originally sharing I/O devices between computers
  - E.g., printers
- Then communicating between computers
  - E.g., file transfer protocol (FTP)
- Then communicating between people
  - E.g., e-mail
- Then communicating between networks of computers
  - E.g., file sharing, www, ...

# The Internet (1962)

- History
  - 1963: JCR Licklider, while at DoD's ARPA, writes a memo describing desire to connect the computers at various research universities: Stanford, Berkeley, UCLA, ...
  - 1969 : ARPA deploys 4 "nodes" @ UCLA, SRI, Utah, & UCSB
  - 1973 Robert Kahn & Vint Cerf invent TCP, now part of the Internet Protocol Suite
- Internet growth rates
  - Exponential since start!

The image is a composite graphic. On the left is a black and white portrait of JCR Licklider, with the text "Lick" below it. On the right is a network diagram showing four nodes: SRI (node #2), Utah (node #4), UCSB (node #3), and UCLA (node #1). Each node is connected to a central hub. Associated with each node is a computer model: 940 for SRI, PDP 10 for Utah, 360 for UCSB, and Sigma 7 for UCLA. In the top right corner is a table titled "ASCII Alphabet" with columns for letters A-M and N-Z, and their corresponding 7-bit binary representations. At the bottom right is a portrait of Vint Cerf with the quote "Revolutions like this don't come along very often".

ASCII Alphabet			
A	1000001	N	1001110
B	1000010	O	1001111
C	1000011	P	1010000
D	1000100	Q	1010001
E	1000101	R	1010010
F	1000110	S	1010011
G	1000111	T	1010100
H	1001000	U	1010101
I	1001001	V	1010110
J	1001010	W	1010111
K	1001011	X	1011000
L	1001100	Y	1011001
M	1001101	Z	1011010

"Lick"

"Revolutions like this don't come along very often"

Vint Cerf

[www.greatachievements.org/?id=3736](http://www.greatachievements.org/?id=3736)

[en.wikipedia.org/wiki/Internet\\_Protocol\\_Suite](http://en.wikipedia.org/wiki/Internet_Protocol_Suite)

# The World Wide Web (1989)

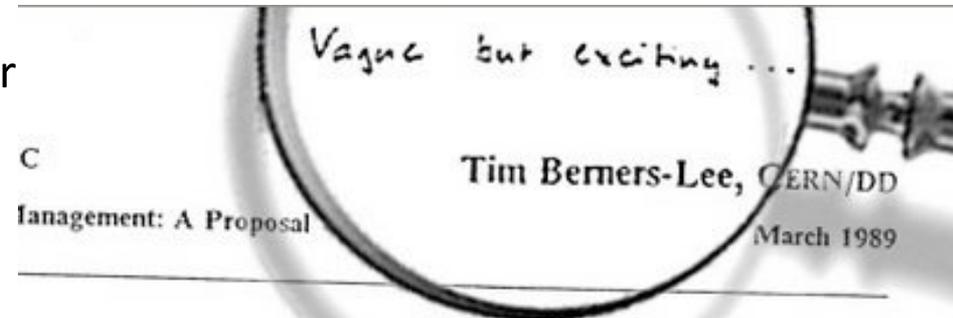
- “System of interlinked hypertext documents on the Internet”
- History
  - 1945: Vannevar Bush describes hypertext system called “memex” in article
  - 1989: Sir Tim Berners-Lee proposed and implemented the first successful communication between a Hypertext Transfer Protocol (HTTP) client and server using the internet.
  - ~2000 Dot-com entrepreneurs rushed in, 2001 bubble burst
- Today : Access anywhere!



Tim Berners-Lee



World's First web server in 1990

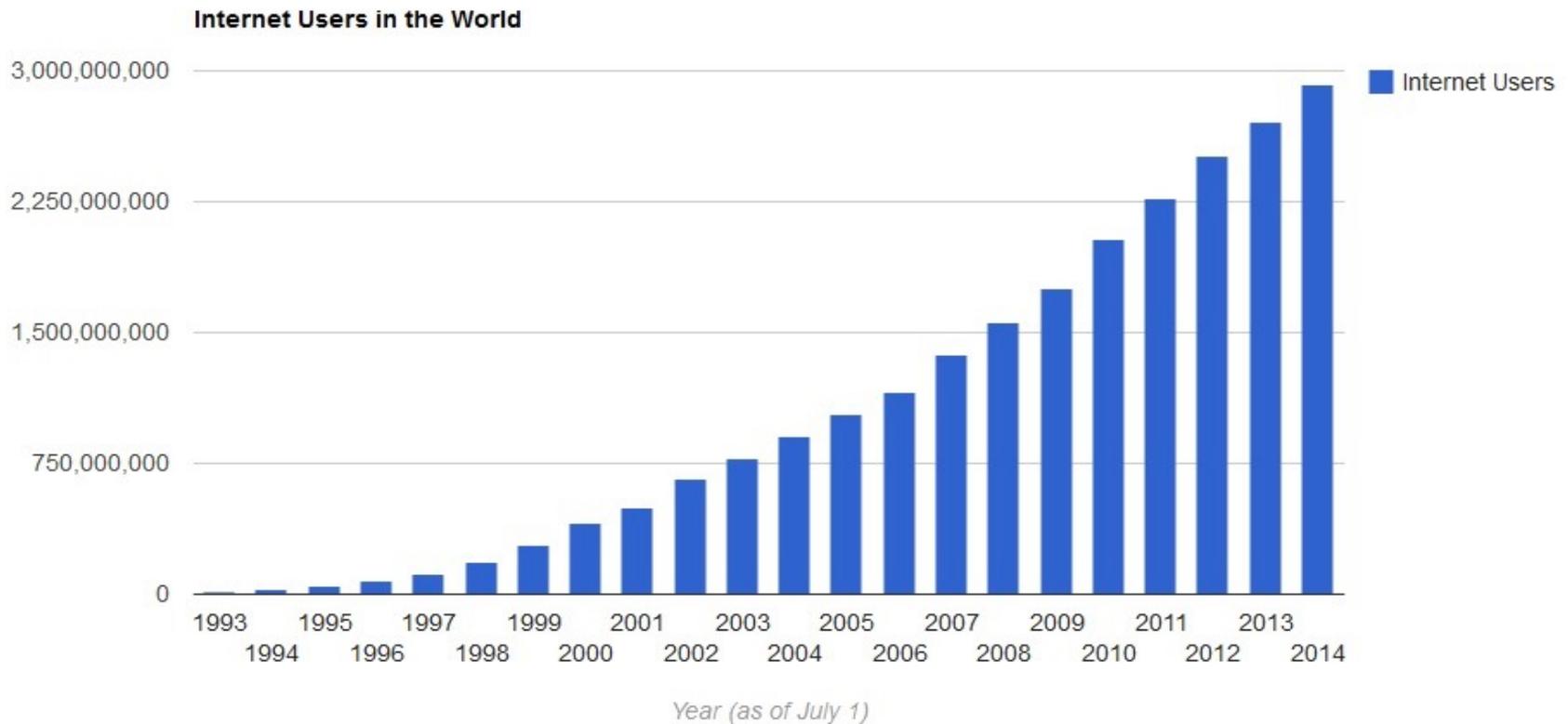


Information Management: A Proposal

On 14th September 1987, the first email was sent from China to Germany.

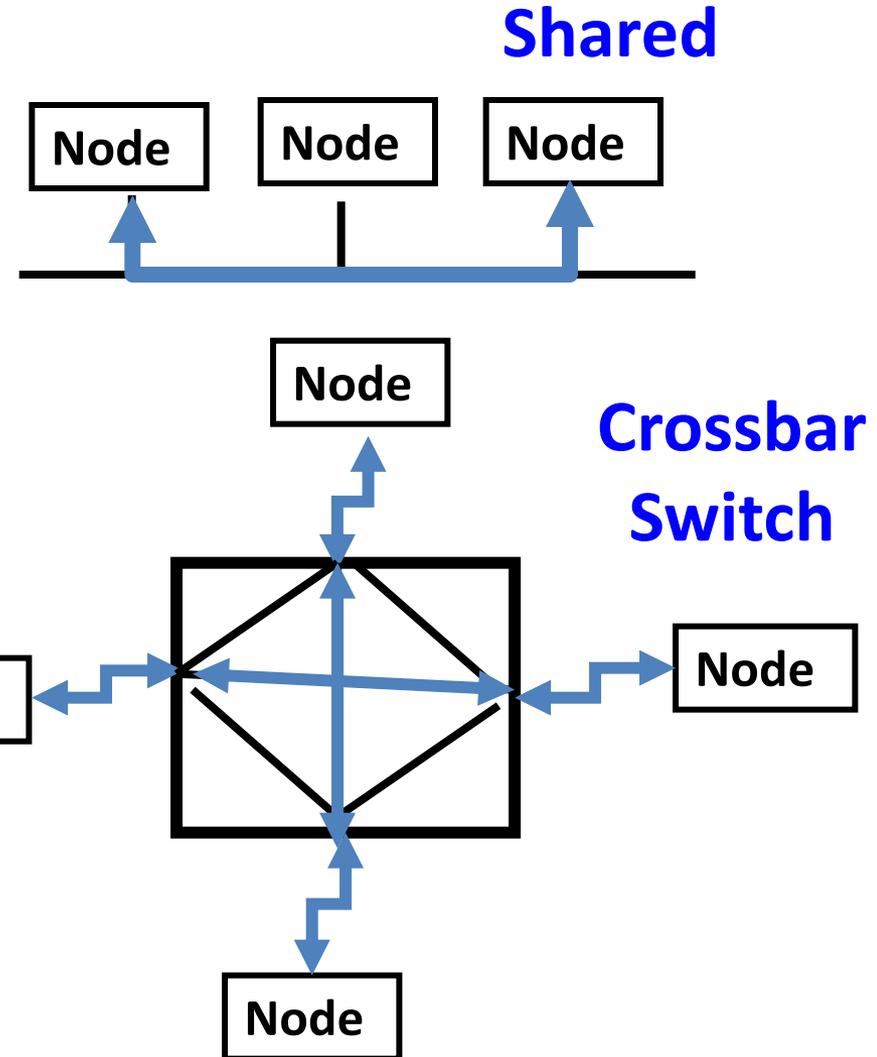


# Growth ... exponential



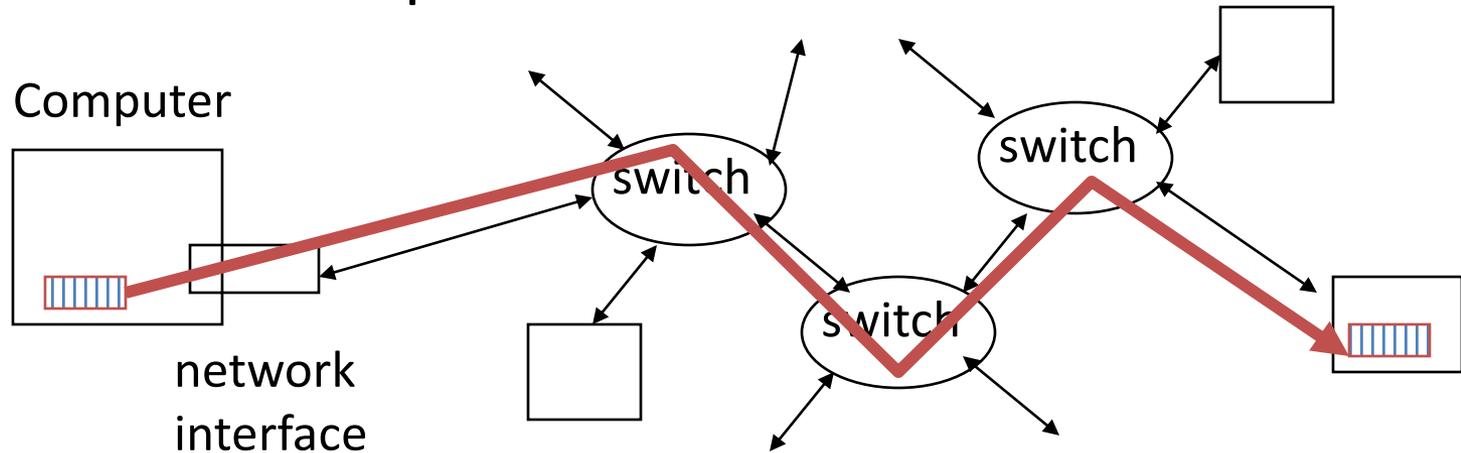
# Shared vs. Switch-Based Networks

- Shared vs. Switched:
  - **Shared:** 1 at a time (CSMA/CD)
  - **Switched:** pairs ("point-to-point" connections) communicate at same time
- Aggregate bandwidth (BW) in switched network is many times that of shared:
  - point-to-point faster since no arbitration, simpler interface



# What makes networks work?

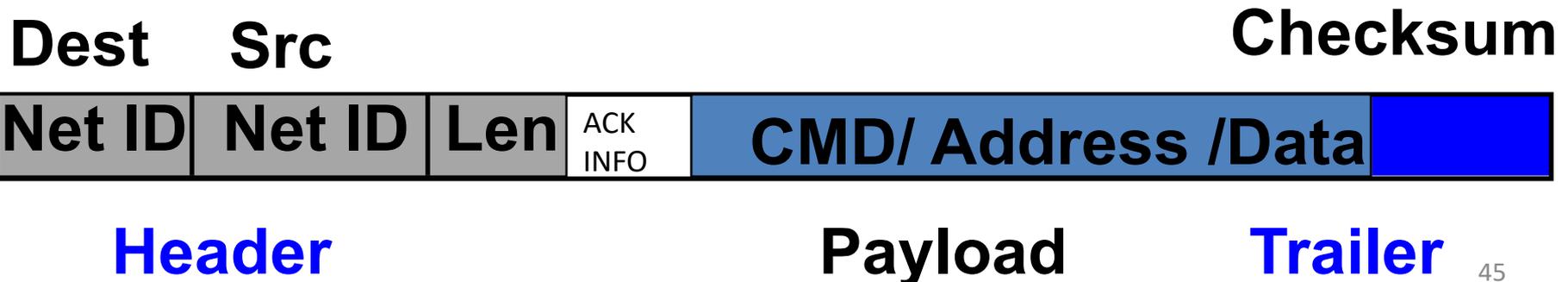
- Links connecting switches and/or routers to each other and to computers or devices



- Ability to name the components and to route packets of information - messages - from a source to a destination
- Layering, redundancy, protocols, and encapsulation as means of abstraction (big idea in Computer Architecture)

# Software Protocol to Send and Receive

- SW Send steps
  - 1: Application copies data to OS buffer
  - 2: OS calculates checksum, starts timer
  - 3: OS sends data to network interface HW and says start
- SW Receive steps
  - 3: OS copies data from network interface HW to OS buffer
  - 2: OS calculates checksum, if OK, send ACK; if not, [delete message](#) (sender resends when timer expires)
  - 1: If OK, OS copies data to user address space, & signals application to continue



# *Protocols* for Networks of Networks?

What does it take to send packets across the globe?

- Bits on wire or air
- Packets on wire or air
- Delivery packets within a single physical network
- Deliver packets across multiple networks
- Ensure the destination received the data
- Create data at the sender and make use of the data at the receiver

# Protocol for Networks of Networks?

Lots to do and at multiple levels!

Use abstraction to cope with complexity of communication

- Hierarchy of layers:
  - Application (chat client, game, etc.)
  - Transport (TCP, UDP)
  - Network (IP)
  - Data Link Layer (Ethernet)
  - Physical Link (copper, wireless, etc.)

# Protocol Family Concept

- *Protocol*: packet structure and control commands to manage communication
- *Protocol families (suites)*: a set of cooperating protocols that implement the network stack
- Key to **protocol families** is that communication occurs **logically** at the same level of the protocol, called **peer-to-peer**...  
...but is **implemented via services** at the next lower level
- **Encapsulation**: carry higher level information within lower level “envelope”

# Inspiration...

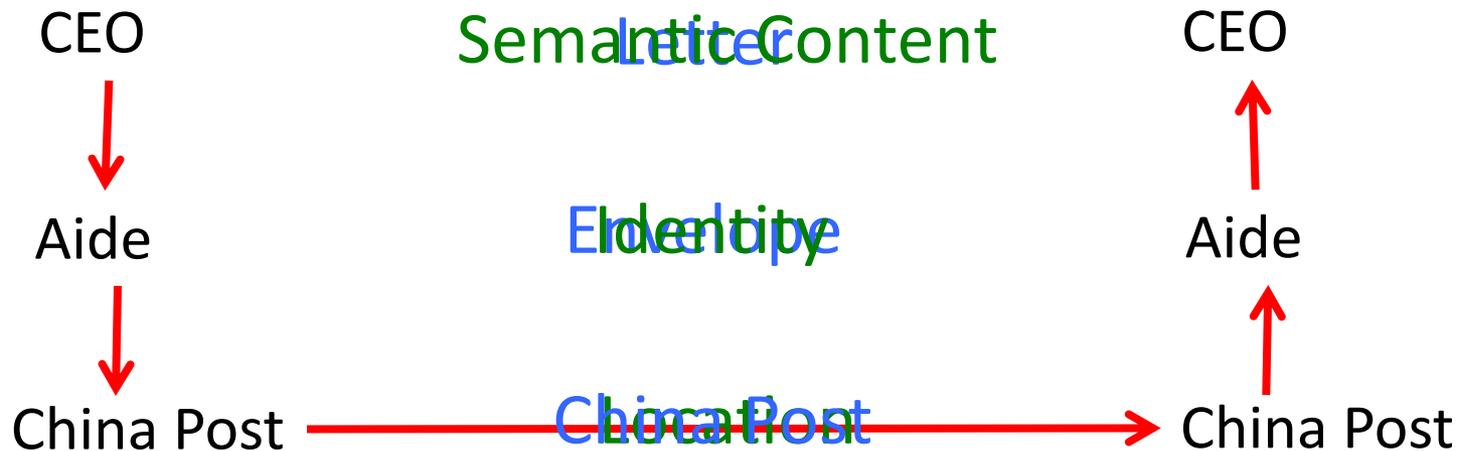
- CEO A writes letter to CEO B
  - Folds letter and hands it to assistant
- ~~Assistant~~ **Dear Jack,**
  - Puts letter in envelope with CEO B's full name
  - Takes to China Post
- ~~China Post Office~~ **It is ready.**
  - Puts letter in larger envelope
  - Puts name and street address on China Post envelope
  - Puts package on China Post delivery truck
- China Post delivers to other company

# The Path of the Letter

“Peers” on each side understand the same things

No one else needs to

Lowest level has most packaging



# “And in conclusion...”

- I/O gives computers their 5 senses
- I/O speed range is 100-million to one
- Polling vs. Interrupts
- DMA to avoid wasting CPU time on data transfers
- Disks for persistent storage, replaced by flash
- Networks: computer-to-computer I/O
  - Protocol suites allow networking of heterogeneous components. Abstraction!!!