



信息科学与技术学院

School of Information Science and Technology

CS 110

Computer Architecture

More I/O

Instructors:

Chundong Wang, Siting Liu & Yuan Xiao

Course website: <https://toast->


[lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html](https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html)

School of Information Science and Technology (SIST)

ShanghaiTech University

2025/5/29

Administratives

- Final exam, June 12th 8am-10am; you can bring **3**-page A4-sized double-sided cheat sheet, **handwritten** only! (**Teaching center 201/202/203**); the whole course will be covered. **NO** electronic devices (no smart watches, no calculators, etc.) **Answer sheet** will be provided! **Only what is written on the answer sheet will be marked and graded.**
- All the assignments have been released! 
 - Project 3 ddl today!
 - Project 4 released, ddl June 3rd. **Will be checked the 17th week during lab sessions. The Longan nano board will also be collected during the lab session.**
 - HW 7 ddl tomorrow!
 - **HW 8 ddl June 5th.**
 - Lab 14 released, to check May 27th, 29th & June 4th (Lab Session 1 only, 1D104); **Prepare in advance!**
- Discussion May 30th & June 6th on *Final Review*.

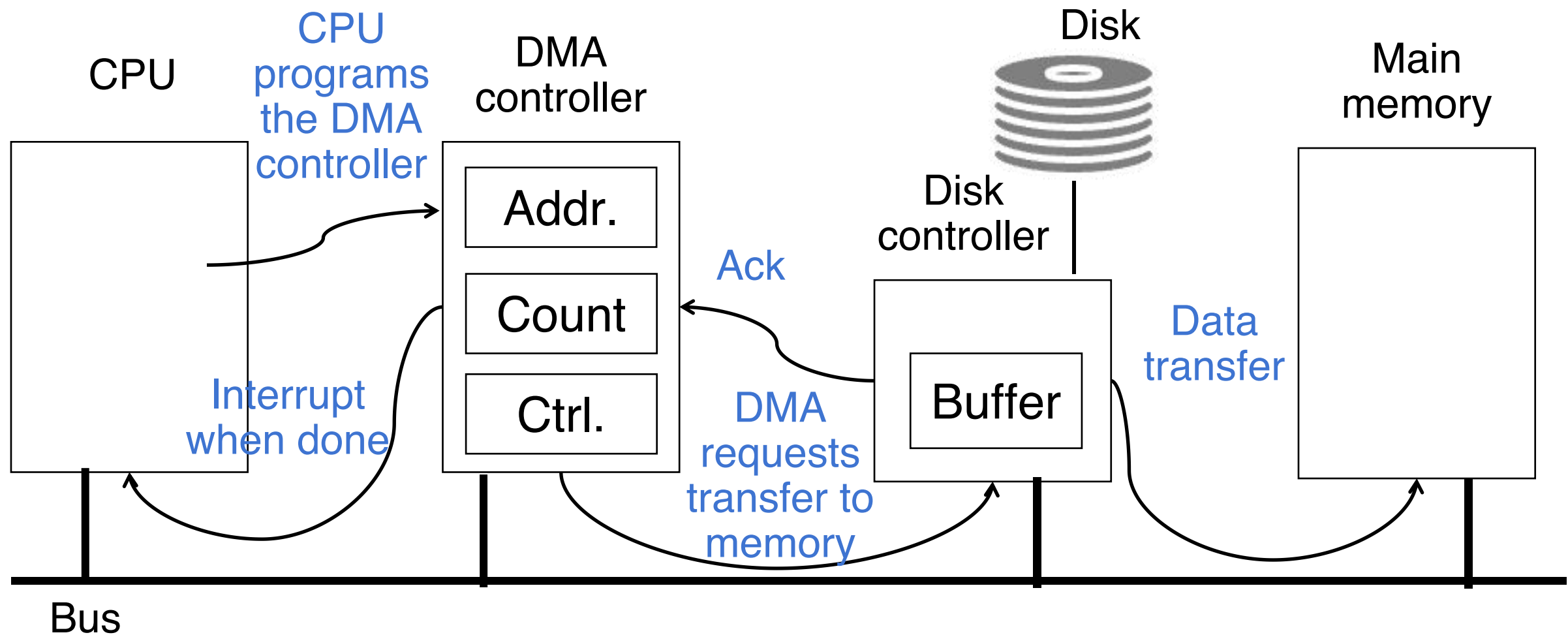
Review: I/O

- “Memory mapped I/O”: Device control/data registers mapped to CPU address space
- CPU synchronizes with I/O device:
 - Polling: wastes processor resources;
 - Interrupts: Nothing to do with no I/O activity; high cost when lots of I/O - expensive saving states and thrashing caches;
 - e.g. mouse and keyboard; what about high data rate (e.g. network, disk)?
- “Programmed I/O”:
 - CPU executes `lw/sw` instructions for all data movement to/from devices
 - CPU spends time doing 2 things:
 - Get data from device to main memory
 - Use data to compute
 - Not ideal, CPU can do something more important and complex

Direct Memory Access (DMA)

- ~~“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
- Allow 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

DMA Transfer



DMA: Incoming Data

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

DMA: Outgoing Data

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

DMA: New Problems

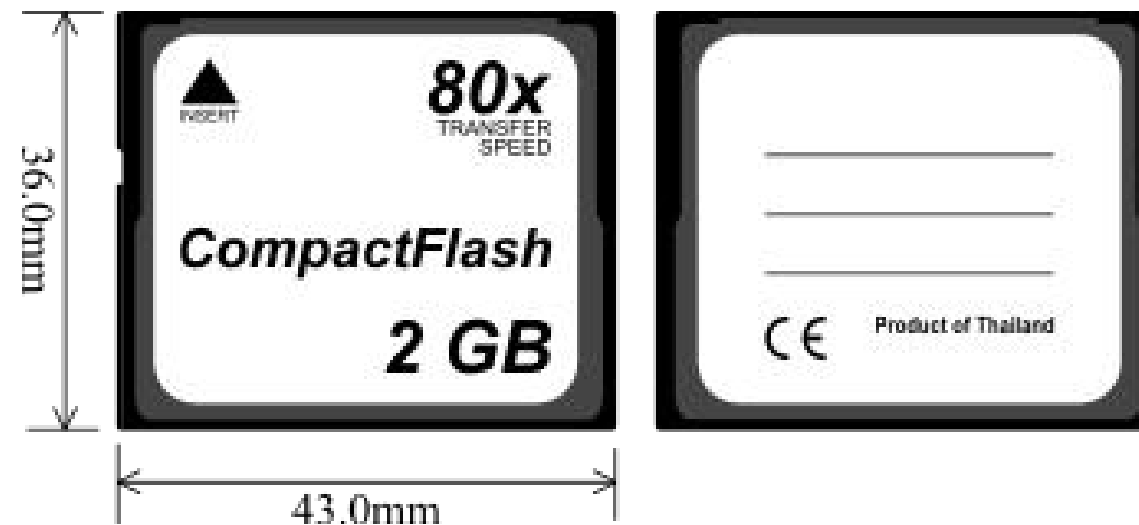
- Where in the memory hierarchy do we plug in the DMA engine?
- Two extremes:
 - Between CPU and L1:
 - Pro: Free coherency
 - Con: Thrash the CPU's working set with transferred data
 - Between Last-level cache and main memory:
 - Pro: Don't mess with caches
 - Con: Need to explicitly manage coherency

DMA: 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

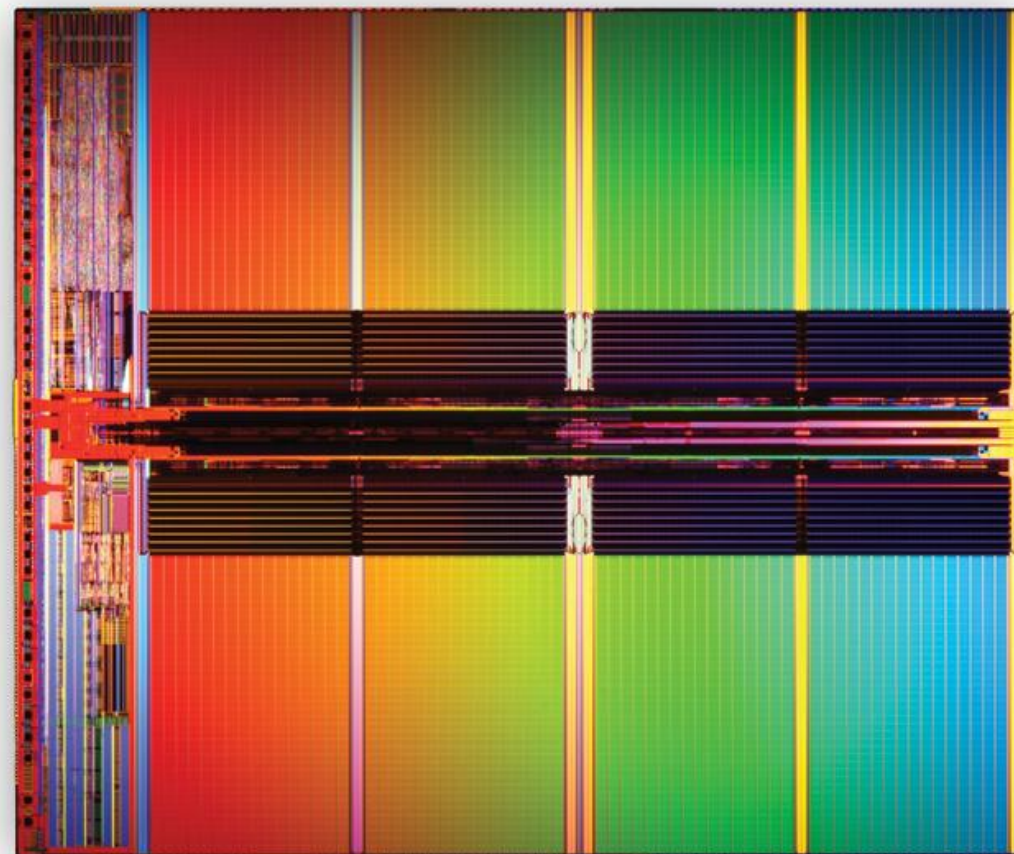
Common I/O device: SSD (Flash Memory)

- >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 μ 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

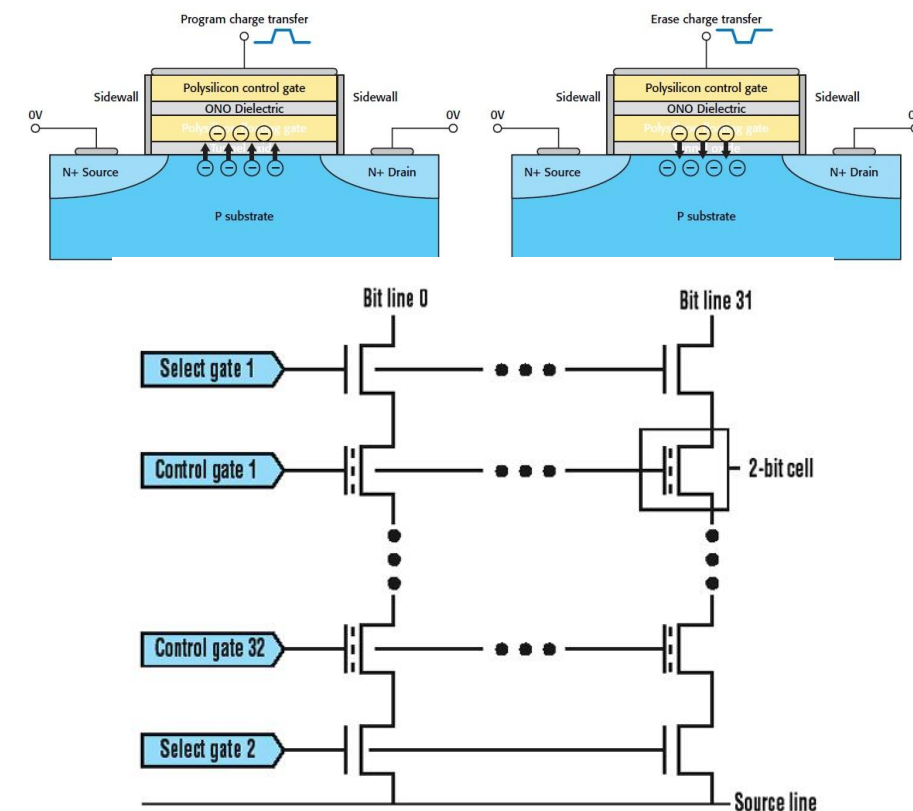


Flash Memory, SSD Technology

- 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 **solid-state drive (SSD)**;
- Even compute using flash memory, more in EE219.

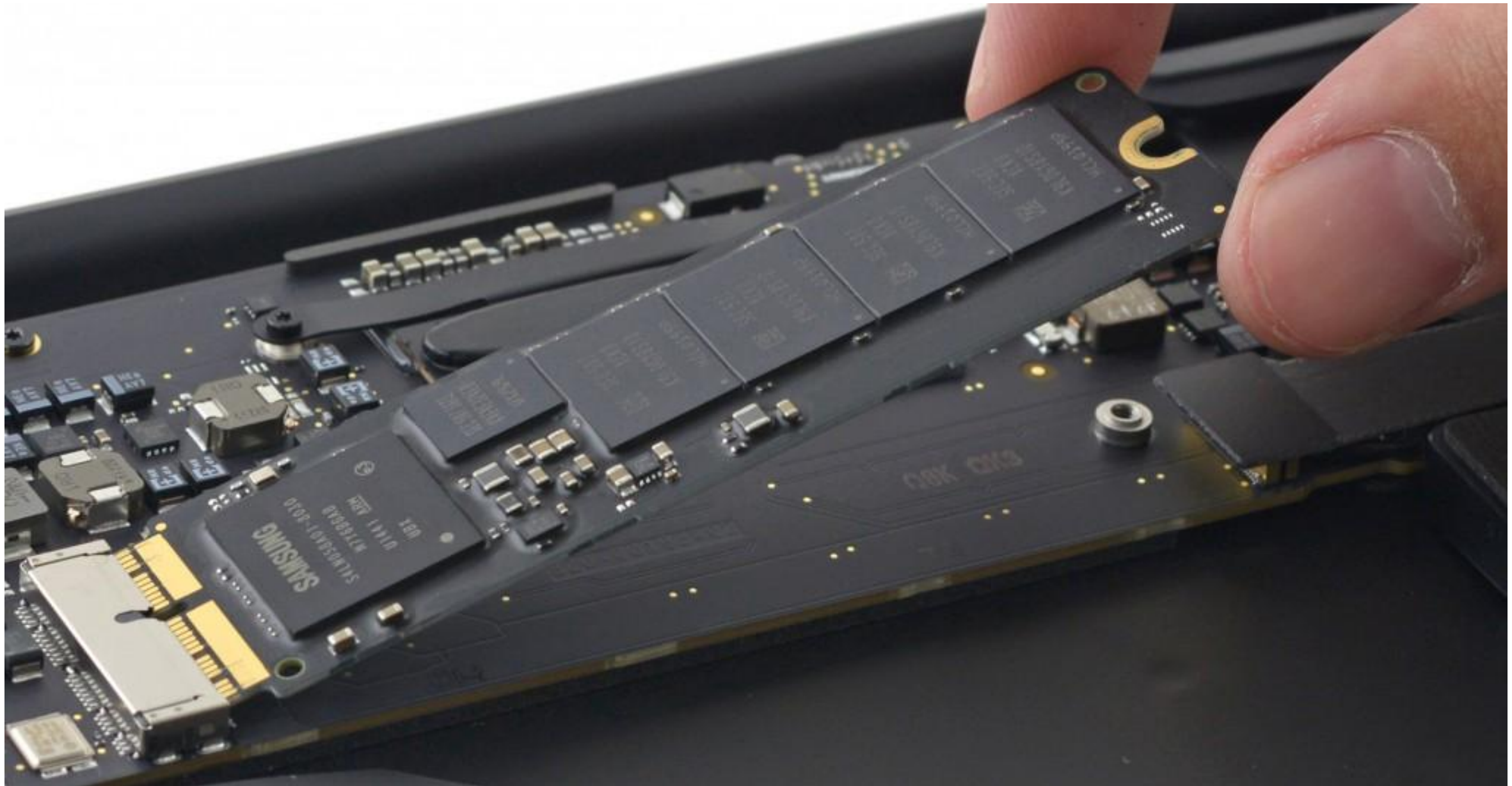


2. Micron's triple-level cell (TLC) flash memory stores 3 bits of data in each transistor.

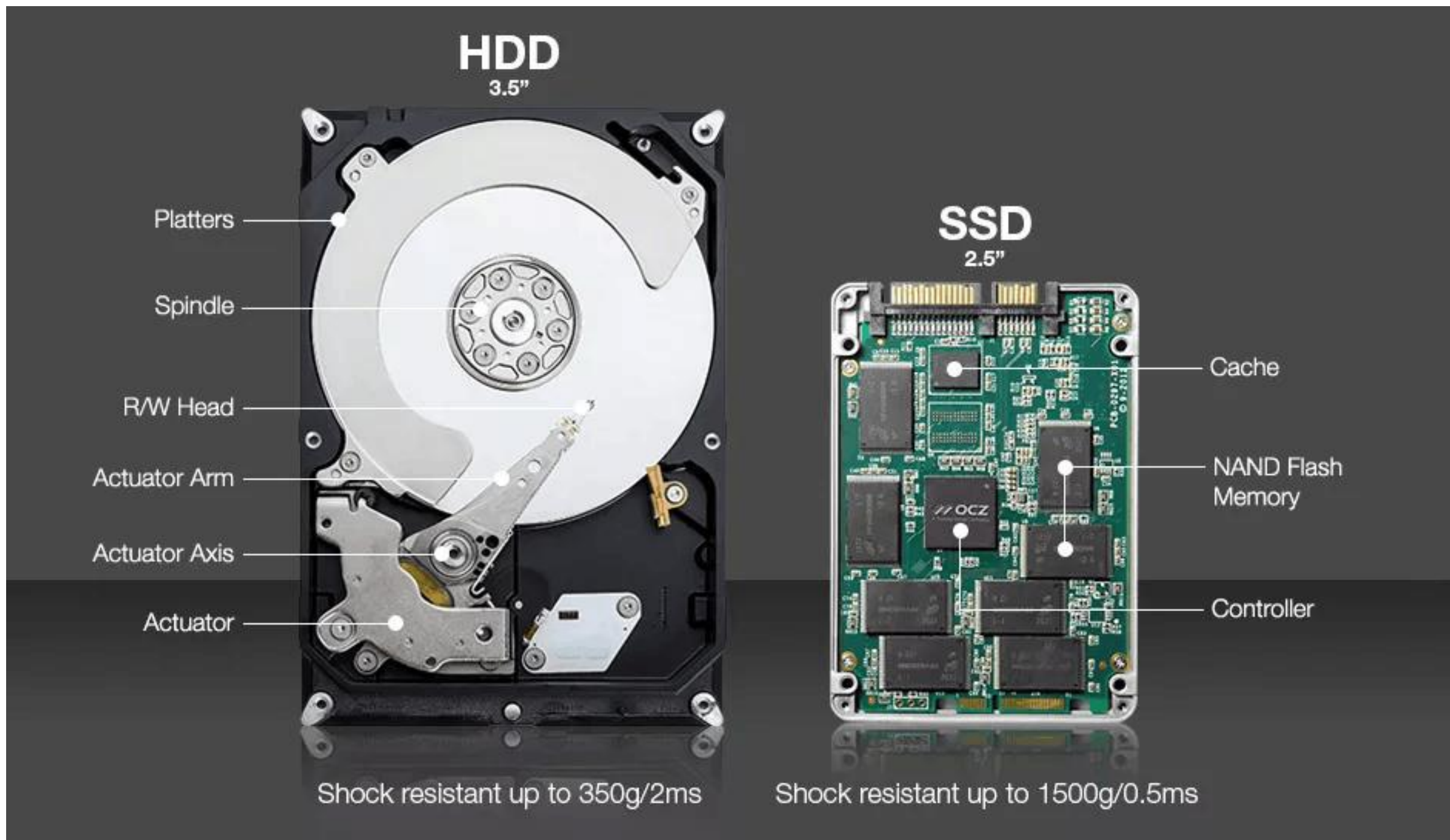


In the basic functional block used in multilevel NAND flash memories, 32 rows of bit lines and 32 control-gate lines form a building block that's repeated many times to form the memory array. The select gate lines are used with the control gate lines to control access to the array.

SSD in Real Computing Systems



HDD vs. SSD



HDD vs. SSD

	HDD	SSD
Cost per bit	Cheaper	
Capacity	Larger	
Durable		More durable (shock-resistant)
Performance		Faster
Power consumption		Lower
Size		More compact
Endurance	Better	

Common I/O Devices 2: Networking

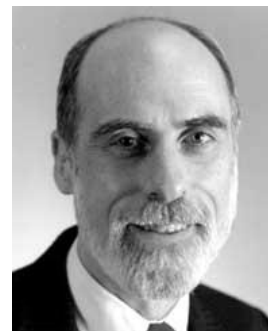
- Originally sharing I/O devices between computers
 - E.g., printers
- Then communicating between computers
 - E.g., file transfer protocol (FTP)
- Then communicating between people
 - E.g., e-mail
- Then communicating between networks of computers
 - E.g., file sharing, www, ...

The Internet (1962)

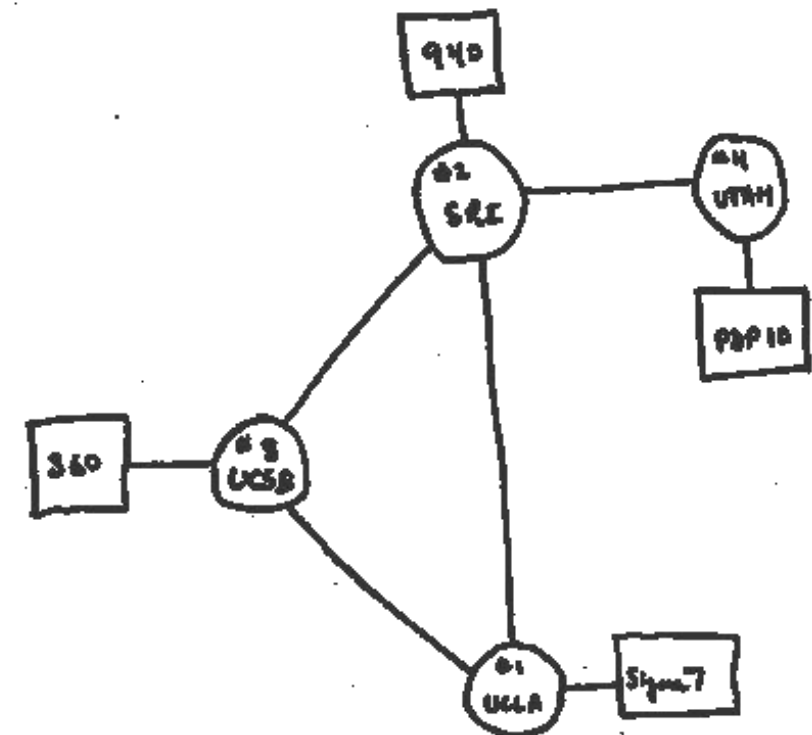
- History
- **1963**: JCR Licklider, while at DoD's ARPA, writes a memo describing desire to connect the computers at various research universities: Stanford, Berkeley, UCLA, ...
- **1969** : ARPA deploys 4 "nodes" @ UCLA, SRI, Utah, & UCSB
- **1973** Robert Kahn & Vint Cerf invent TCP, now part of the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



"Lick"



Vint Cerf



The World Wide Web (1989)

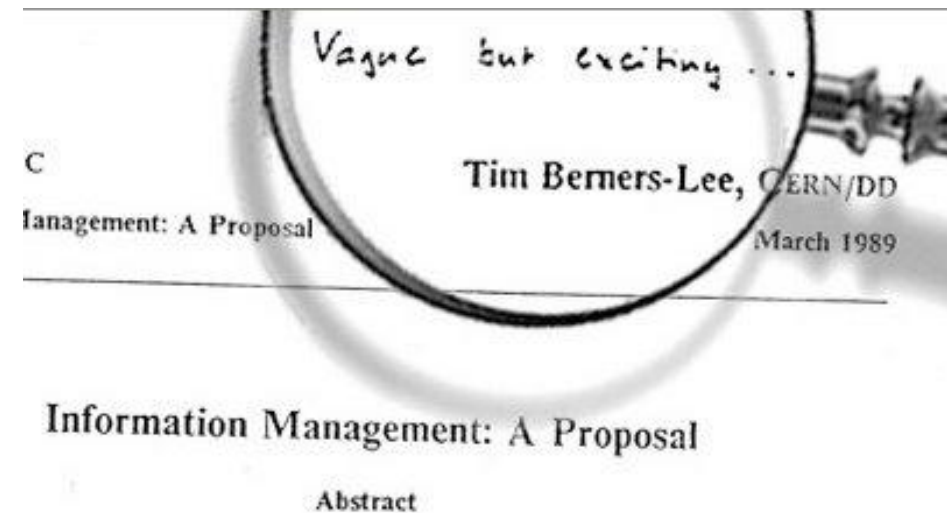
- “System of interlinked hypertext documents on the Internet”
- History
 - **1945**: Vannevar Bush describes hypertext system called “memex” in article
 - **1989**: Sir Tim Berners-Lee proposed and implemented the first successful communication between a Hypertext Transfer Protocol (HTTP) client and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



Tim Berners-Lee

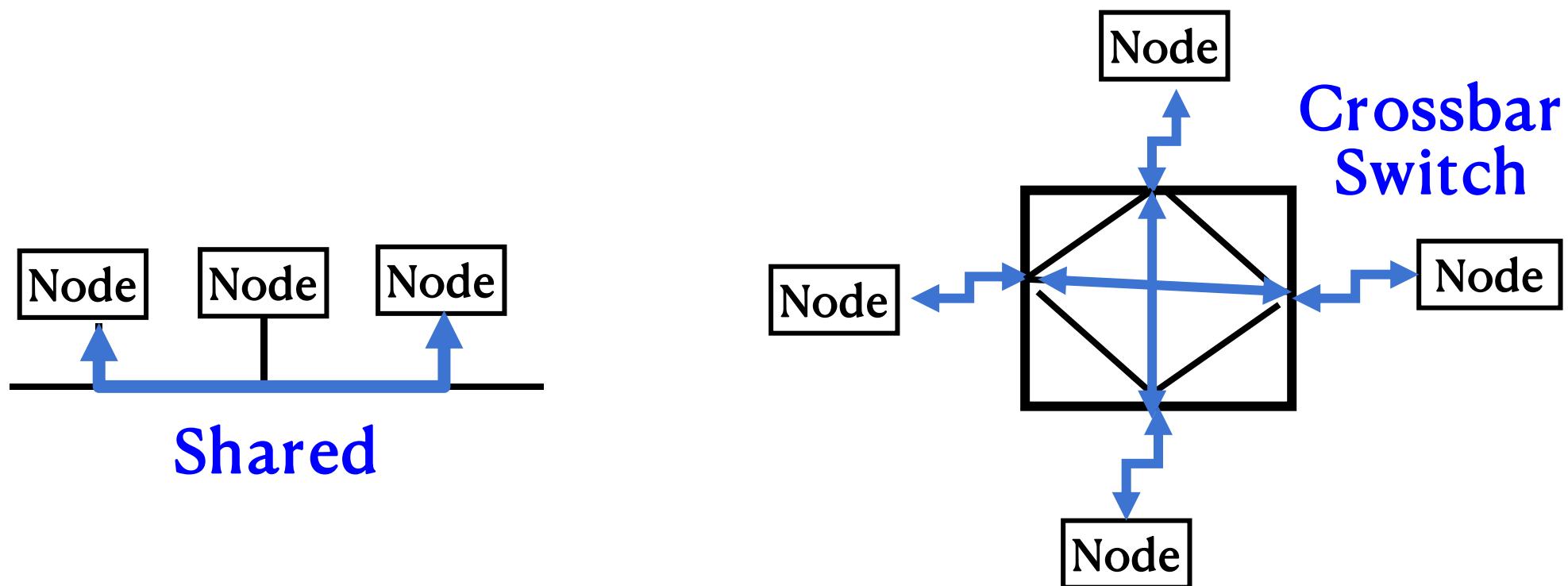


World's First web server in 1990



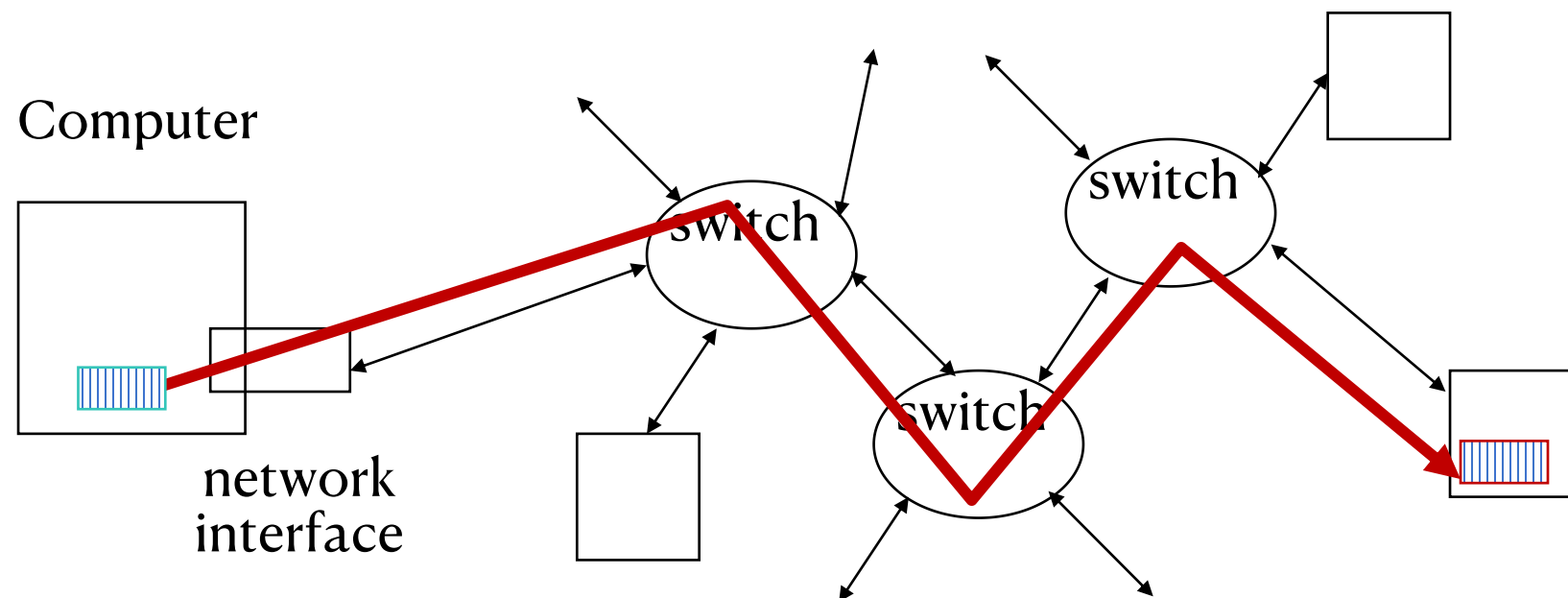
Shared vs. Switch-Based Networks

- Shared vs. Switched:
 - Shared: 1 at a time
 - 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



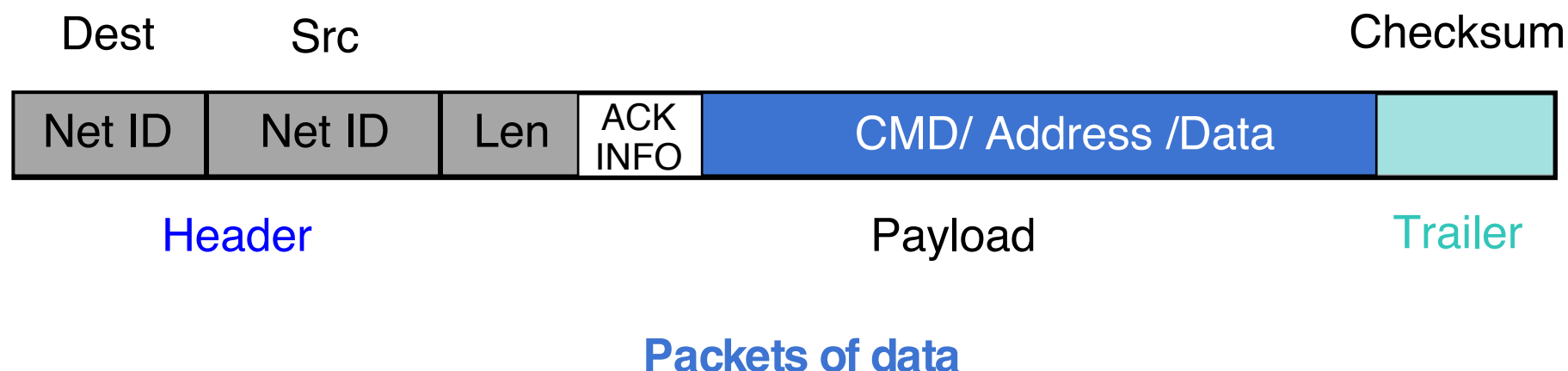
What Makes Networks Work?

- Links connect 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)



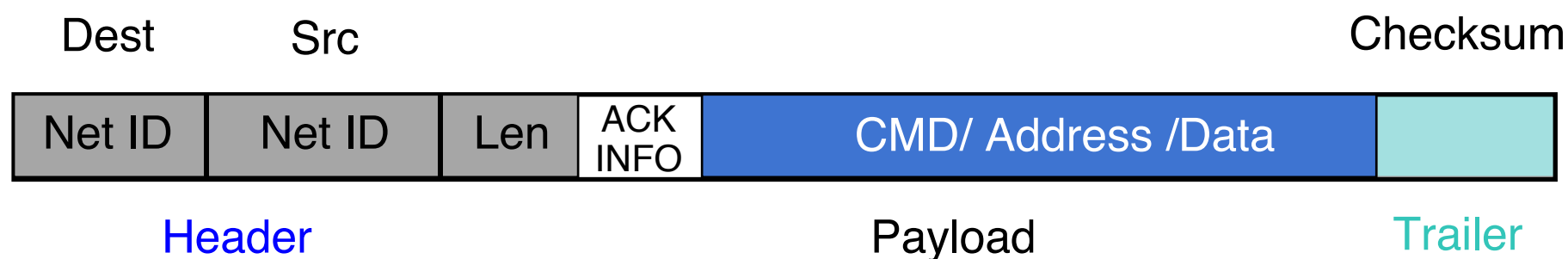
Software Protocol to Send and Receive

- 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



Packets of data

Protocols 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 wires, wireless, etc.)

Protocol Family Concept

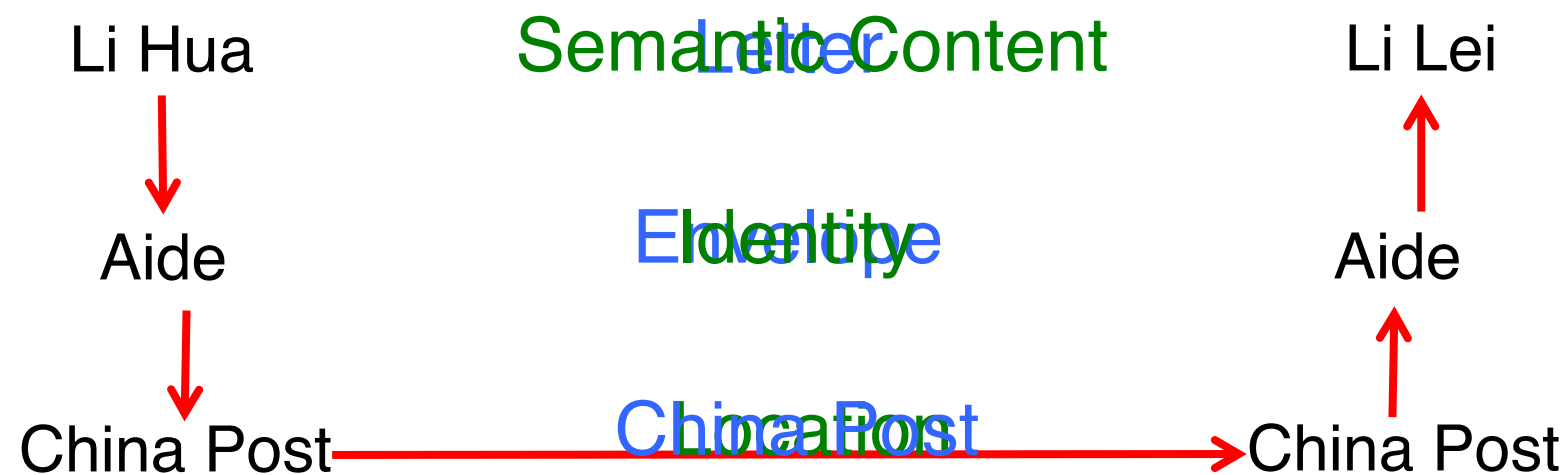
- Protocol: packet structure and control commands to manage communication
- **Protocol families (suites)**: a set of cooperating protocols that implement the network stack
- Key to protocol families is that communication occurs **logically** at the same level of the protocol, called peer-to-peer...but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

Analogy: Send a Letter

- Li Hua writes letter to Li Lei
 - Folds letter and hands it to assistant
- Assistant:
 - Puts letter in envelope with Li Lei's full name
 - Takes to China Post
- China Post Office
 - 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

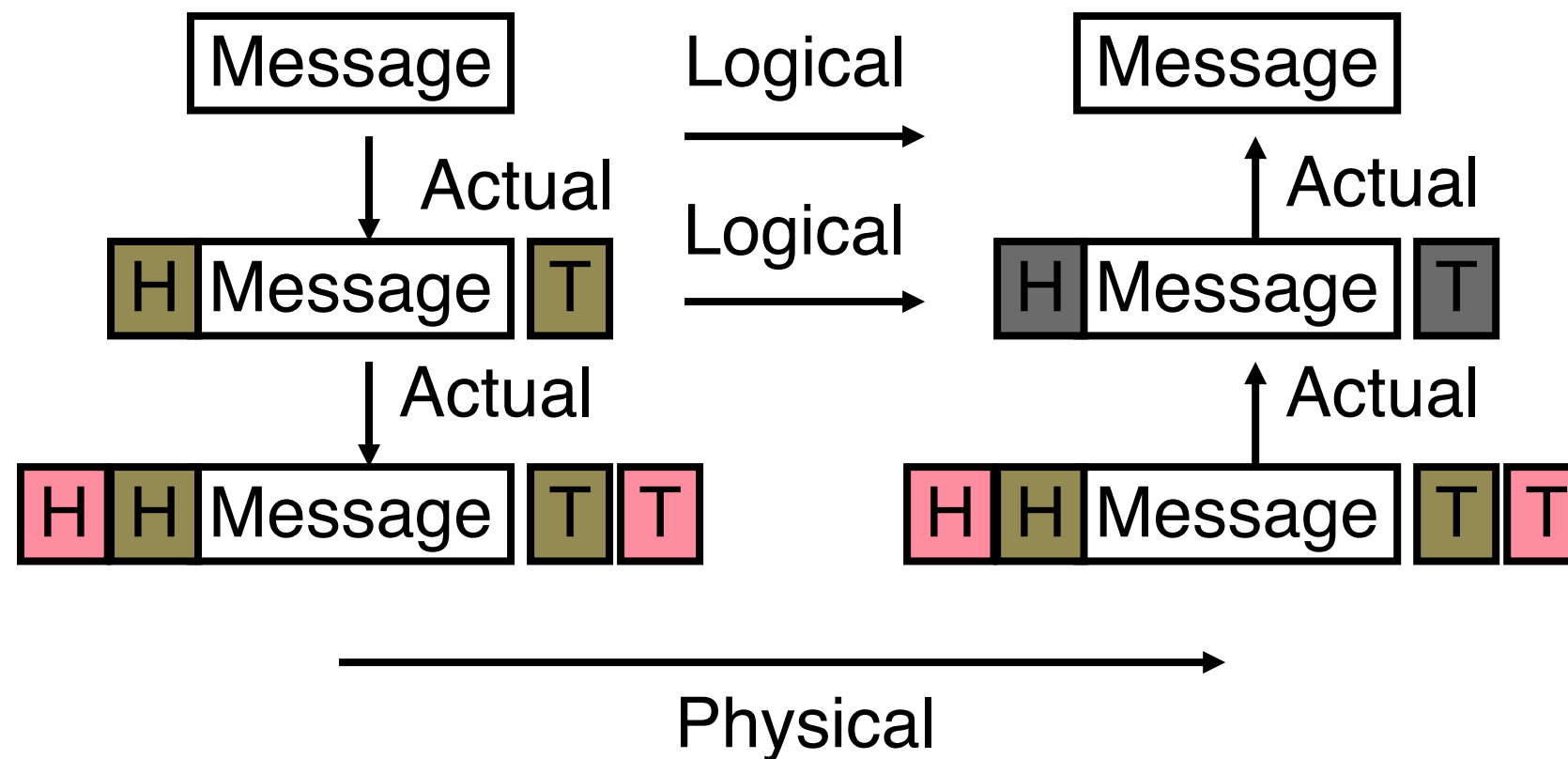
The Path of the Letter

- “Peers” on each side understand the same things
- Lowest level has most packaging



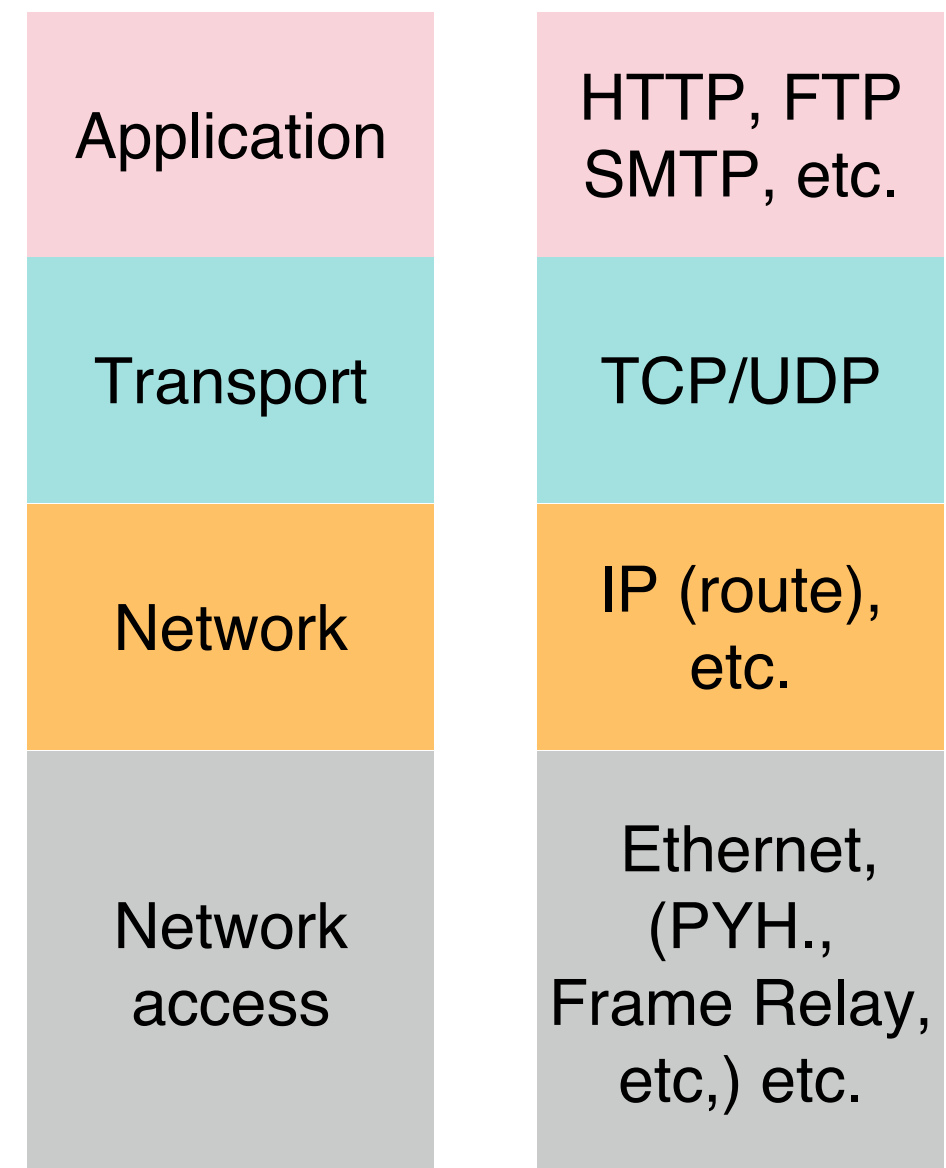
Protocol Family Concept

- Each lower level of stack “encapsulates” information from layer above by adding header and trailer.



Most Popular Protocol for Network of Networks

- Transmission Control Protocol/Internet Protocol (TCP/IP) stack
- This protocol family is the basis of the Internet, a WAN (wide area network) protocol
 - IP makes best effort to deliver
 - Packets can be lost, corrupted
 - TCP guarantees delivery
 - TCP/IP so popular it is used even when communicating locally: even across homogeneous LAN (local area network)
 - UDP/IP: video or sound streaming; video call....
 - More in CS120



Conclusion

- 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!!!



信息科学与技术学院

School of Information Science and Technology

CS 110

Computer Architecture

Warehouse Scale Computing (WSC)

Instructors:

Chundong Wang, Siting Liu & Yuan Xiao

Course website: <https://toast->

[lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html](https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html)

School of Information Science and Technology (SIST)

ShanghaiTech University

2025/5/29

Parallelism Overview

Today's Lecture

- **Parallel Requests**
Assigned to computer
e.g., Search “CS110”
- **Parallel Threads**
Assigned to core
e.g., Lookup, Ads
- **Parallel Instructions**
>1 instruction @ one time
e.g., 5 pipelined instructions
- **Parallel Data**
>1 data item @ one time
e.g., Add of 4 pairs of words
- **Hardware descriptions**
All gates @ one time
- **Programming Languages**

Software

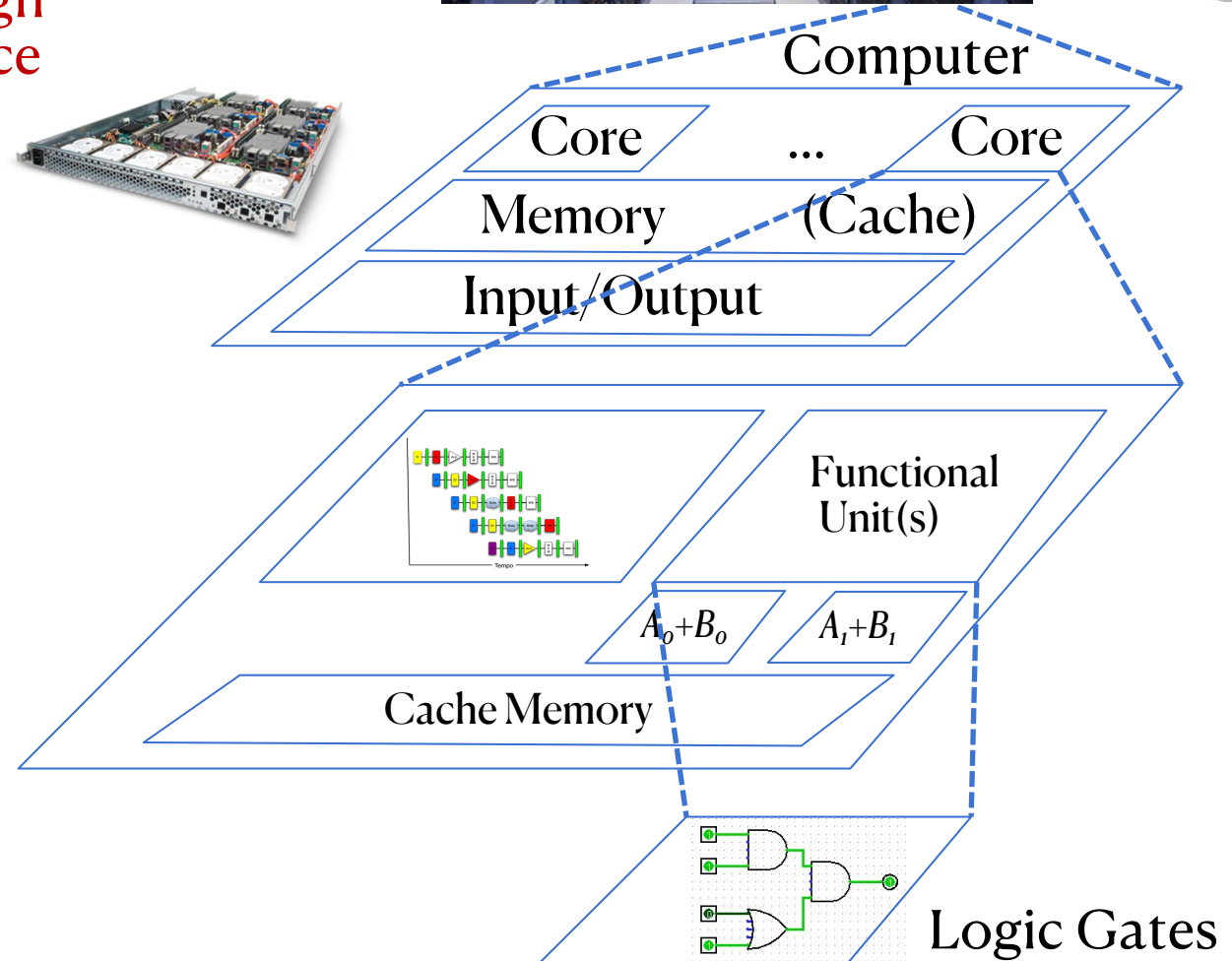
Hardware

**Harness
Parallelism &
Achieve High
Performance**

Warehouse
Scale
Computer



Smart
Phone



Google's WSC

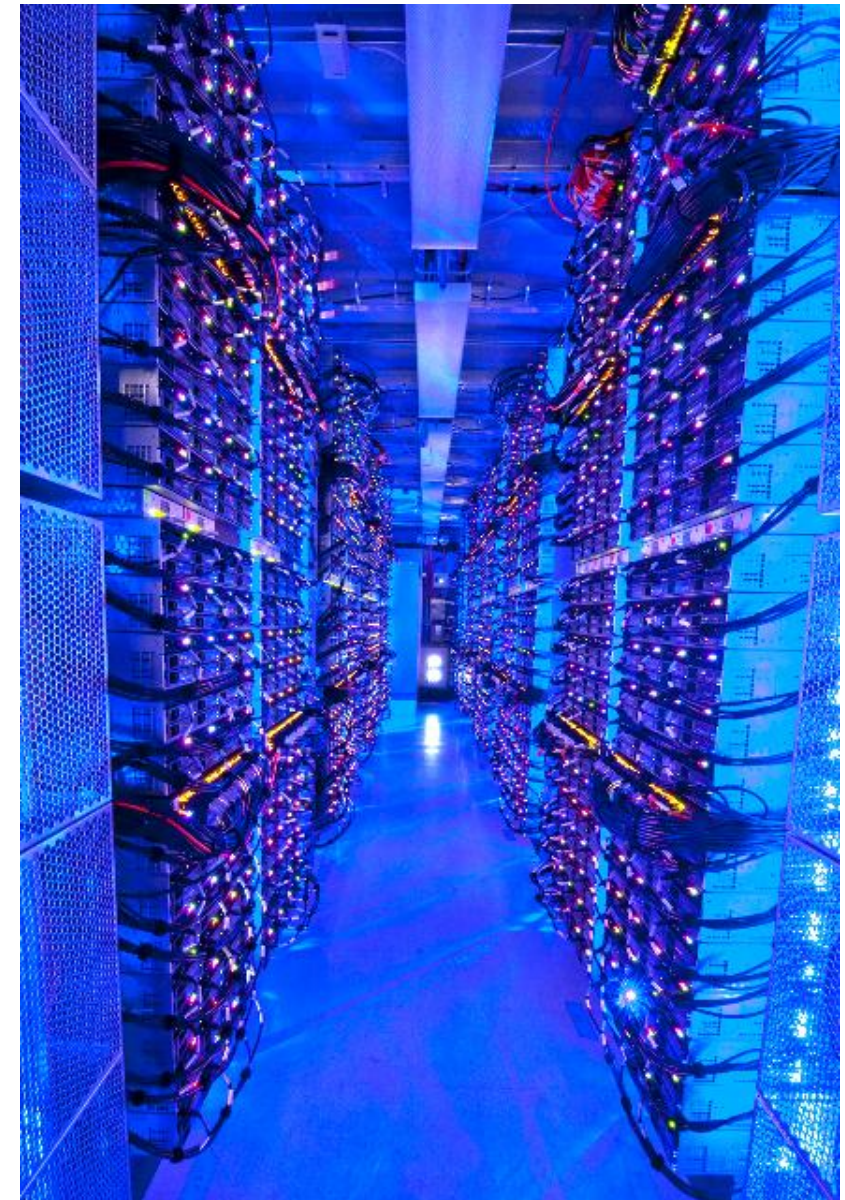


Containers in WSCs

Inside WSC



Inside container



Array, Rack, Server



A Giant Computer

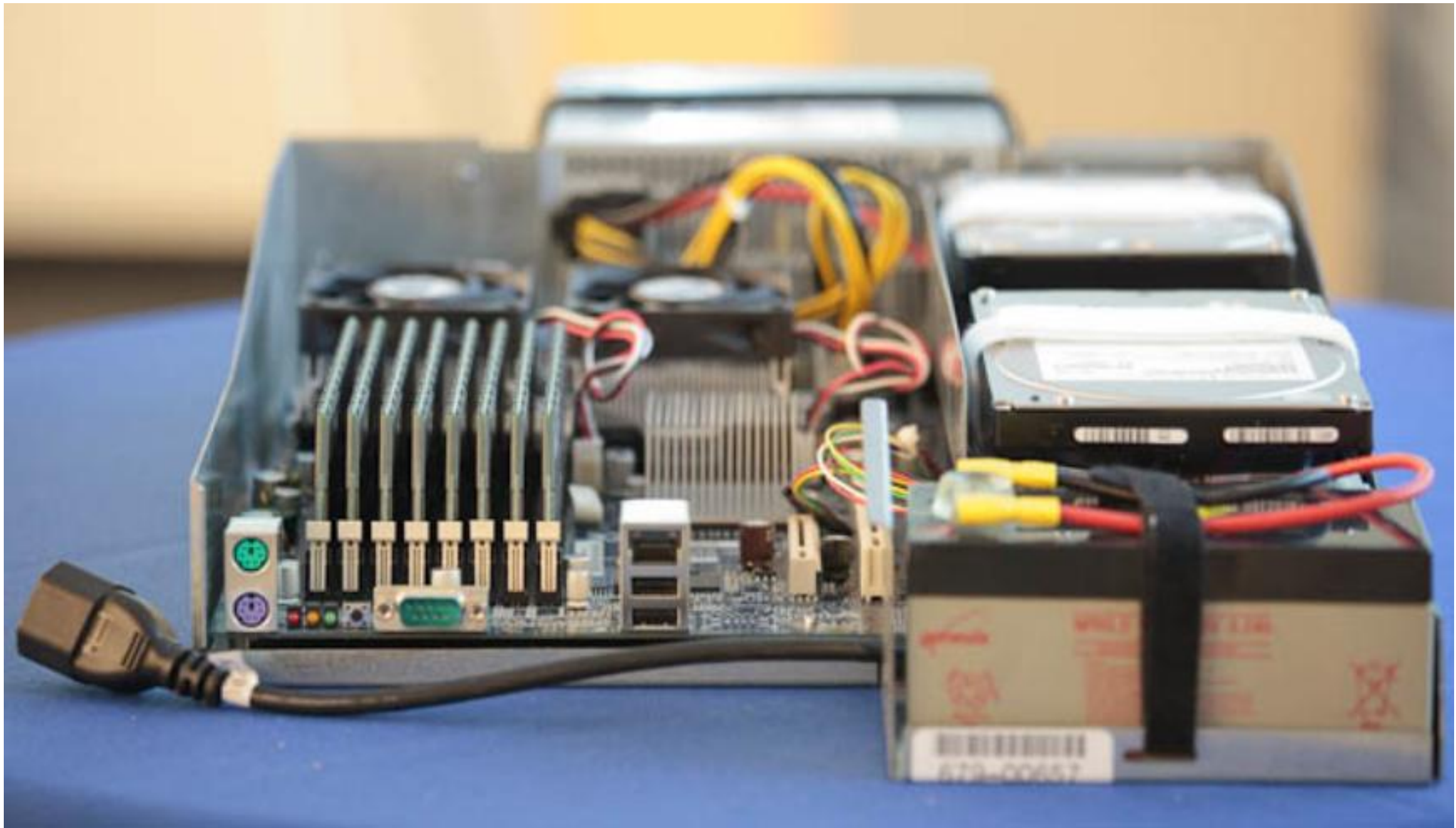
- Sunway TaihuLight

系统峰值性能	125.436PFlops
实测持续运算性能	93.015PFlops
处理器型号	"申威26010" 众核处理器
整机处理器个数	40960个
实整机处理器核数	10649600个
系统总内存	1310720 GB
操作系统	Raise Linux
编程语言	C、C++、Fortran
并行语言及环境	MPI、OpenMP、OpenACC等
SSD存储	230TB
在线存储	10PB, 带宽288GB/s
近线存储	10PB, 带宽32GB/s



<http://www.nscdw.cn/swsource/5d2fe23624364f0351459262>

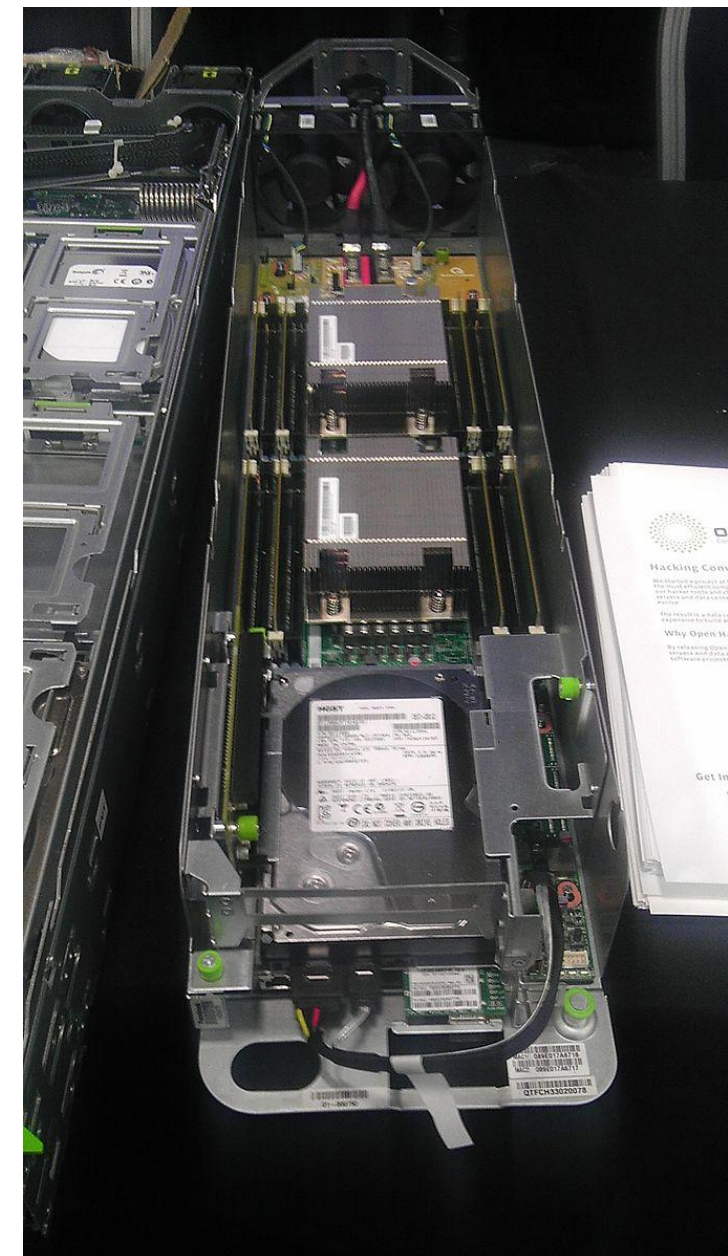
Google Server Internals





Open Compute Project

- Share designs of data center products
 - Facebook, Intel, Nokia, Google, Apple, Microsoft, Seagate Technology, Dell, Cisco, Goldman Sachs, Lenovo, ...
- Design and enable the delivery of the most efficient server, storage and data center hardware designs for scalable computing.
- Openly sharing ideas, specifications and other intellectual property is the key to maximizing innovation and reducing operational complexity
- All Facebook Data Centers are 100% OCP



Warehouse-Scale Computers

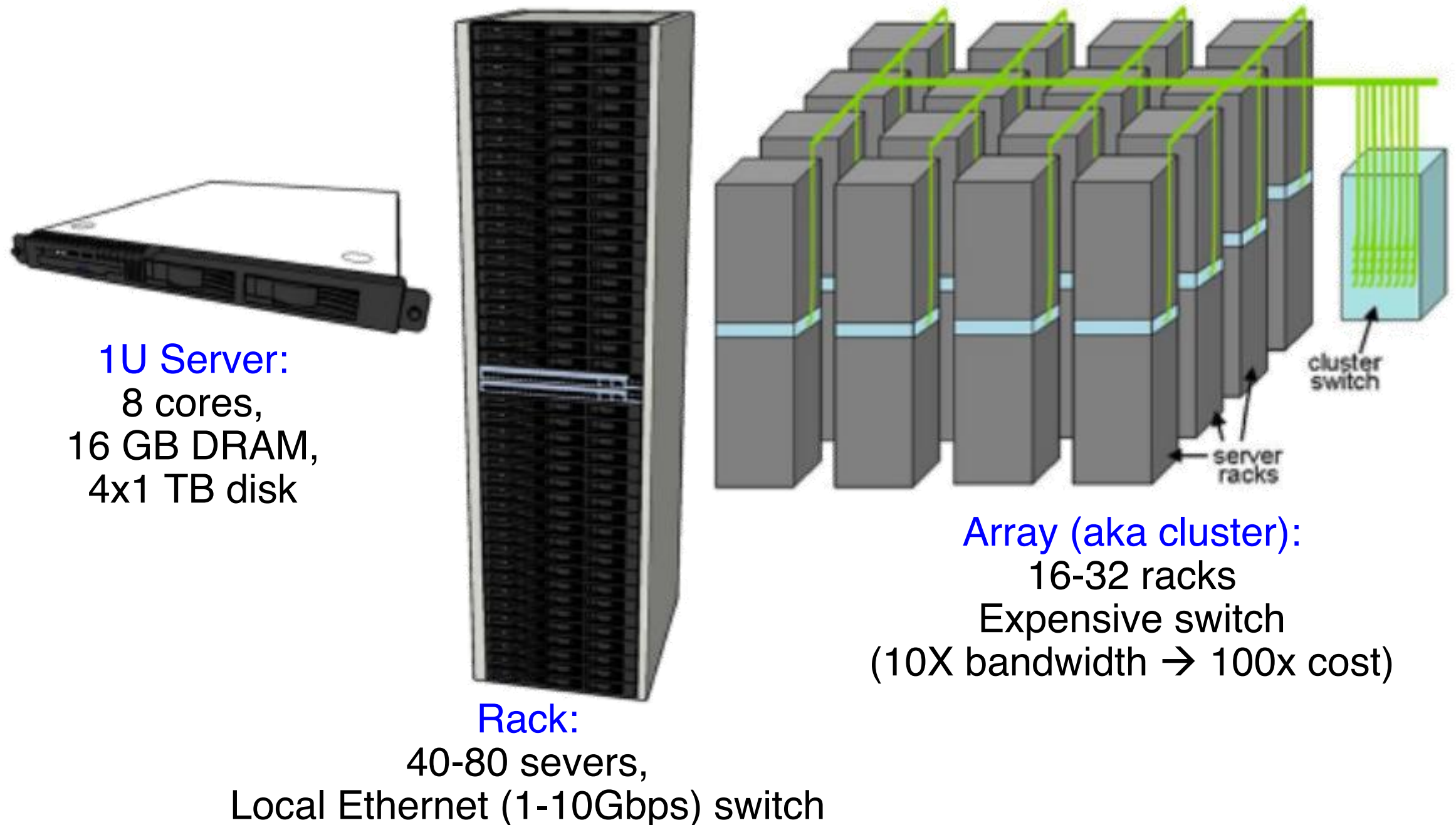
- Datacenter
 - Collection of 10,000 to 100,000 servers
 - Networks connecting them together
- **Single gigantic** machine
- Very large applications (Internet service): search, email, video sharing, social networking
- Very high availability
- “...WSCs are no less worthy of the expertise of computer systems architects than any other class of machines” Barroso and Hoelzle, 2009

Unique to WSCs

- Ample Parallelism
 - Request-level Parallelism: e.g., web search
 - Data-level Parallelism: e.g., image classifier training
- Scale and its Opportunities/Problems
 - Scale of economy: low per-unit cost
 - Cloud computing: rent computing power with low costs (e.g., AWS)
 - High # of failures
 - e.g.: 4 disks/server, annual failure rate: 4% WSC of 50,000 servers: 1 disk fail/hour
- Operation Cost Count
 - Longer life time (>10 years)
 - **Cost of equipment purchases << cost of ownership**

$$\frac{50000 \times 4 \times 4\%}{365 \times 24} \approx 0.913$$

WSC Architecture



WSC Storage Hierarchy

- Lower latency to DRAM in another server than local disk
- Higher bandwidth to local disk than to DRAM in another server

1U Server:

DRAM: 16GB, 0.1 μ s, 20GB/s

Disk: 2TB, 10⁴ μ s, 200MB/s

Rack(80 servers):

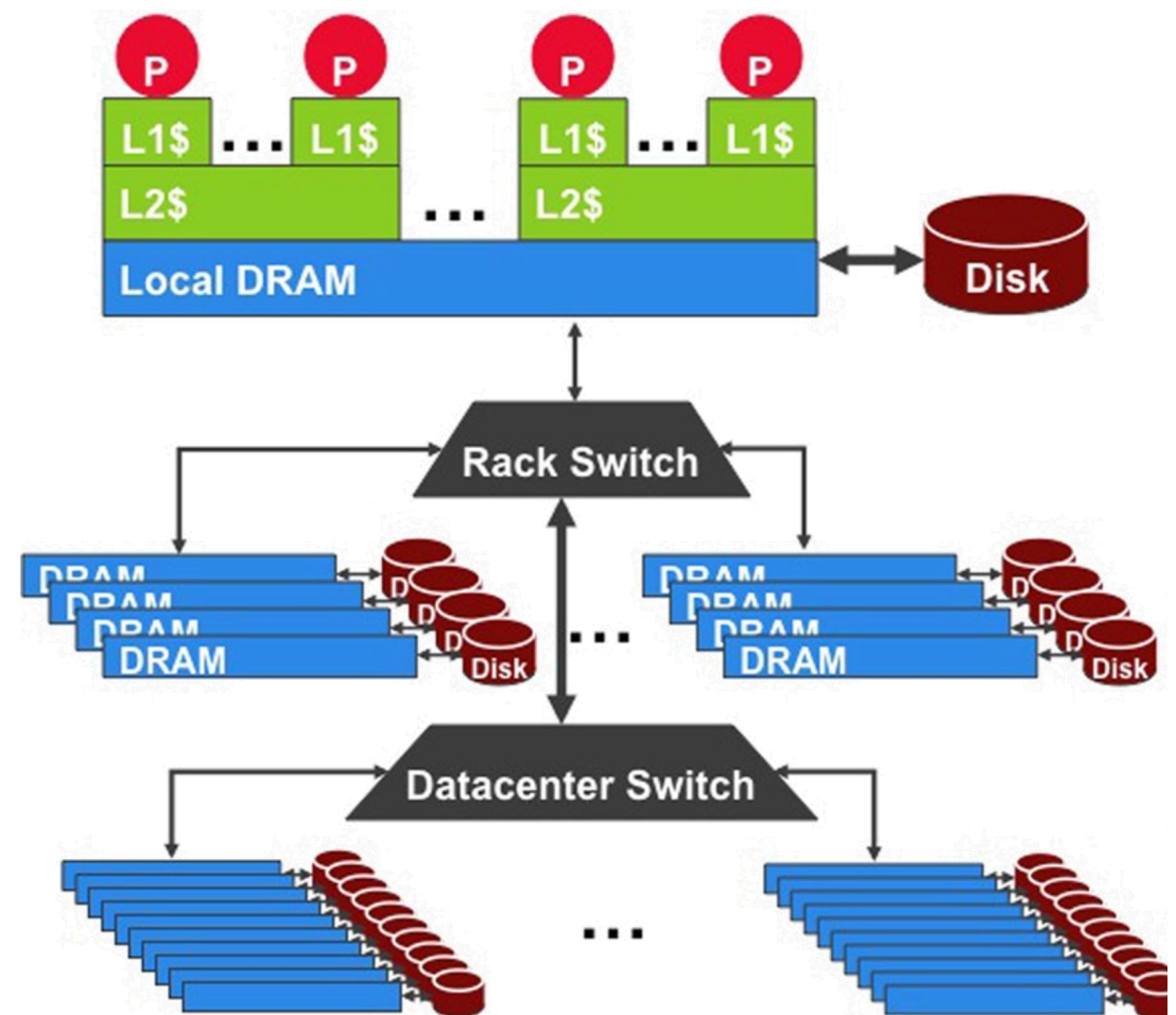
DRAM: 1TB, 300 μ s, 100MB/s

Disk: 160TB, 11ms, 100MB/s

Array(30 racks):

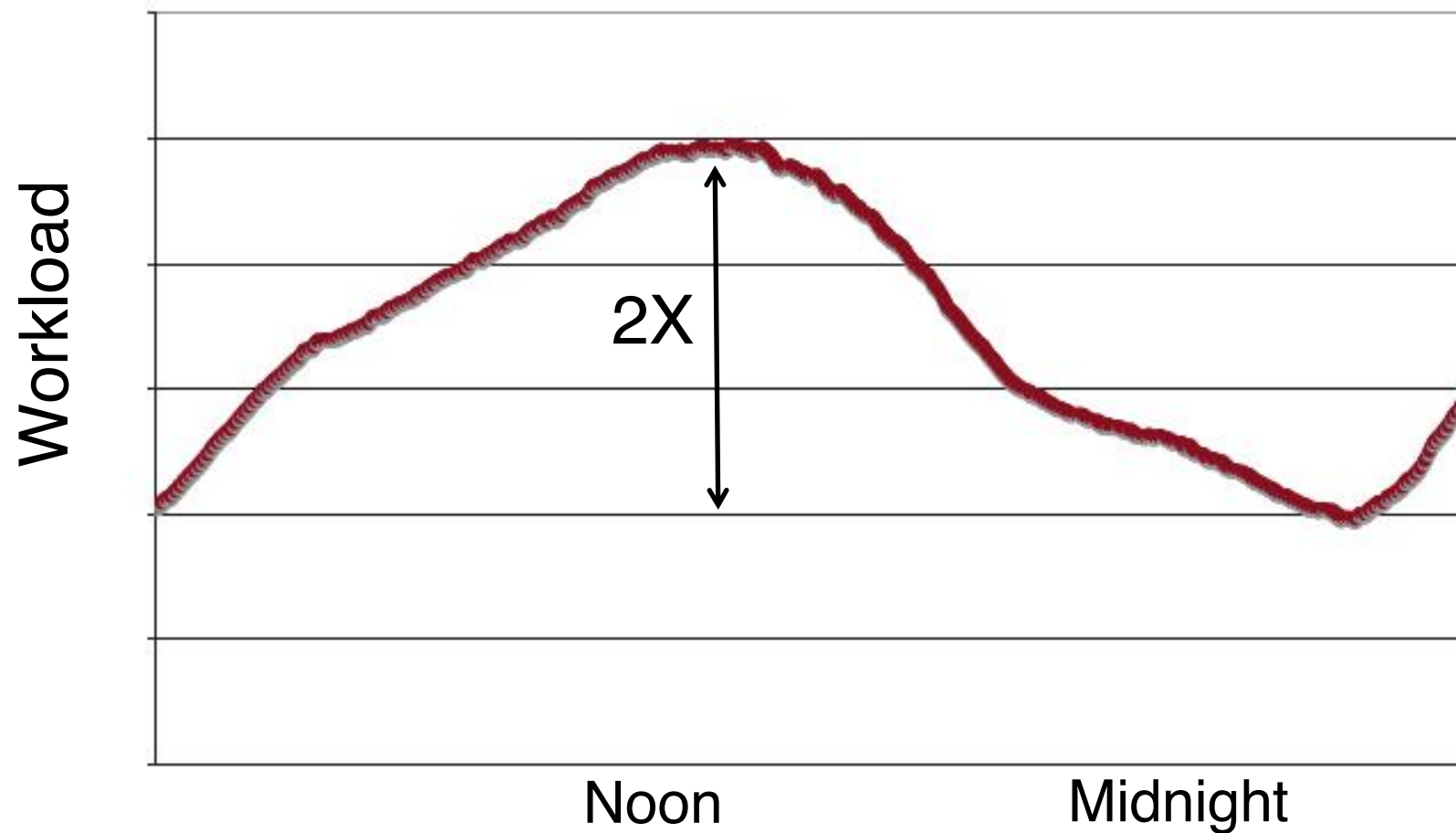
DRAM: 30TB, 500 μ s, 10MB/s

Disk: 4.80PB, 12ms, 10MB/s



Workload Variation

- Online service: Peak usage 2X off-peak



Impact on WSC software

- Latency, bandwidth → Performance
 - Independent data set within an array
 - Locality of access within server or rack
- High failure rate → Reliability, Availability
 - Preventing failures is expensive
 - Cope with failures gracefully
- Varying workloads → Scalability, Availability
 - Scale up and down gracefully
- More challenging than software for single computers!

Power Usage Effectiveness

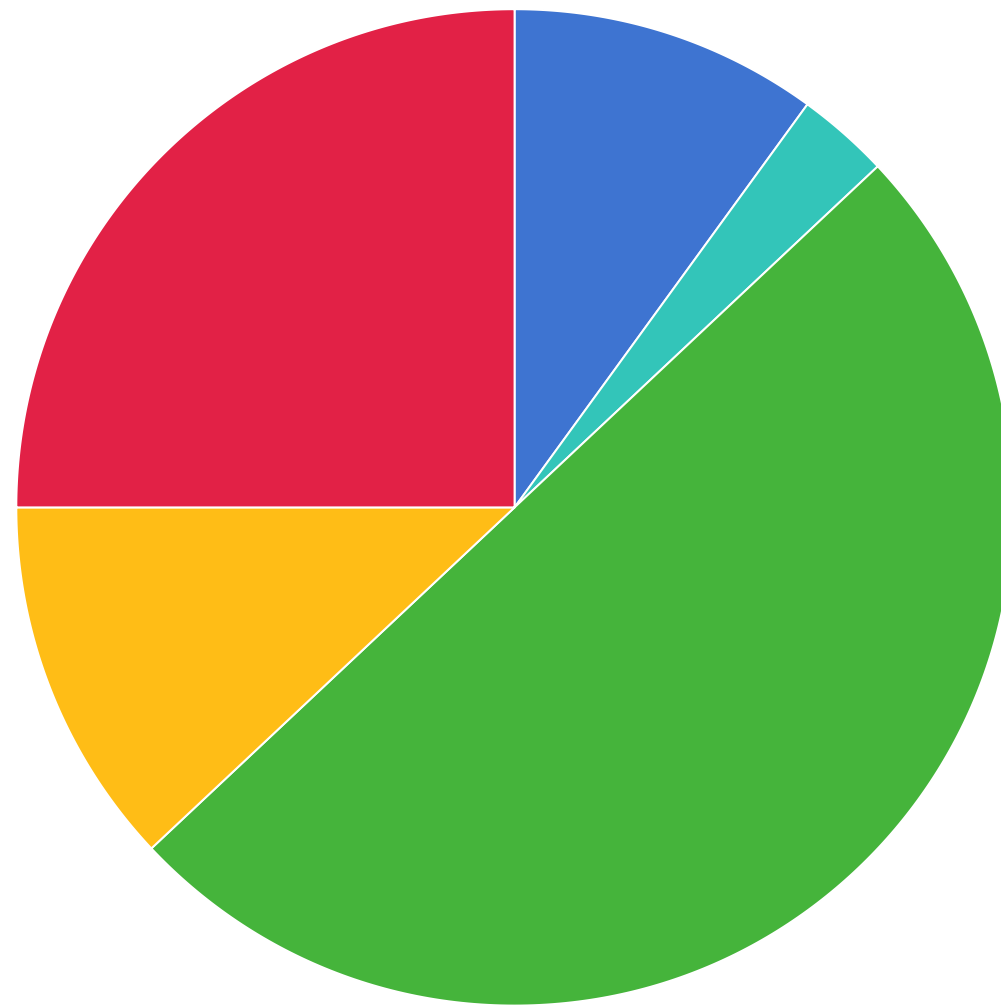
- Energy efficiency
 - Primary concern in the design of WSC
 - Important component of the total cost of ownership
- Power Usage Effectiveness (PUE):

$$\frac{\text{Total Building Power}}{\text{IT Equipment Power}}$$

- A power efficiency measure for WSC
- Not considering efficiency of servers, networking
- Perfection = 1.0
- Google WSC's PUE = 1.2

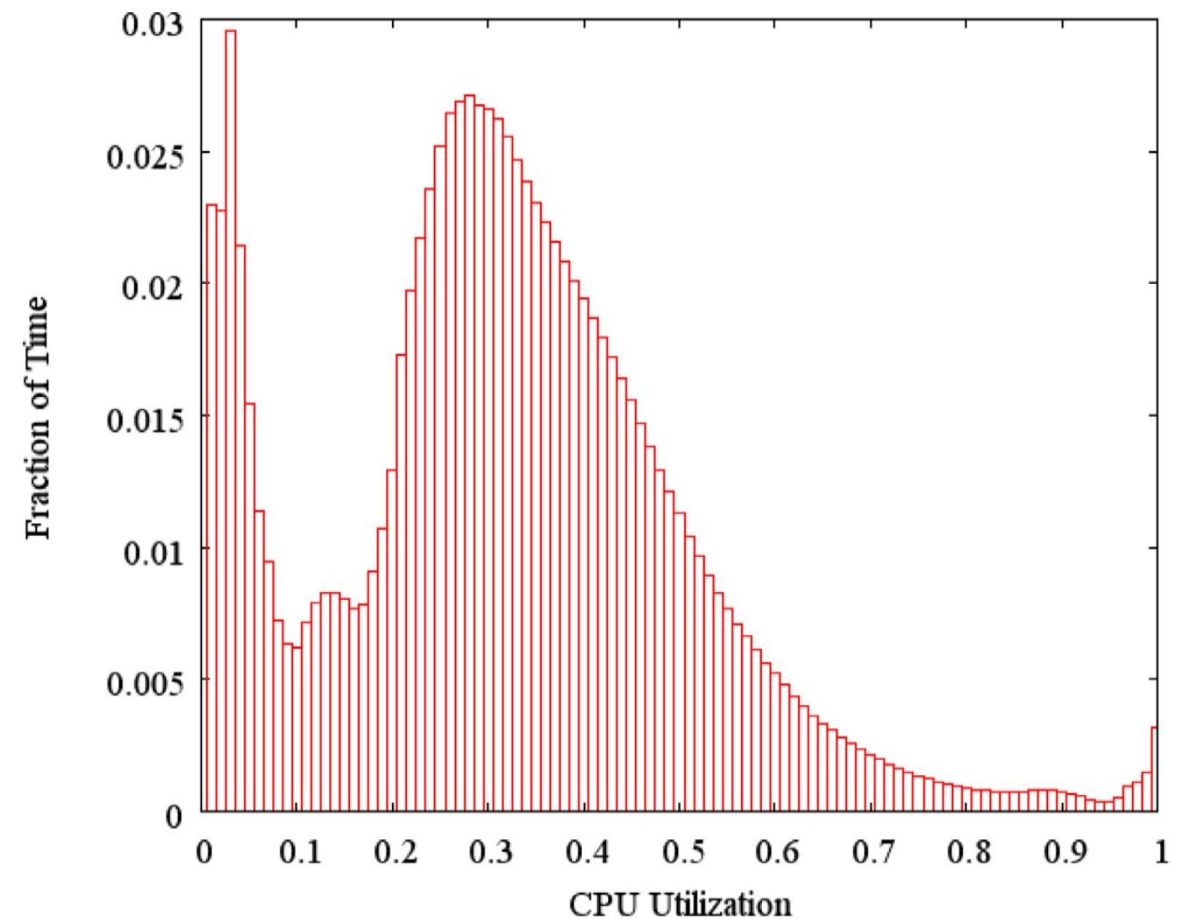
Where Data Center Power Goes

- Electricity Transformer/UPS
- Lighting, etc.
- IT Equipment
- Air Movement
- Cooling



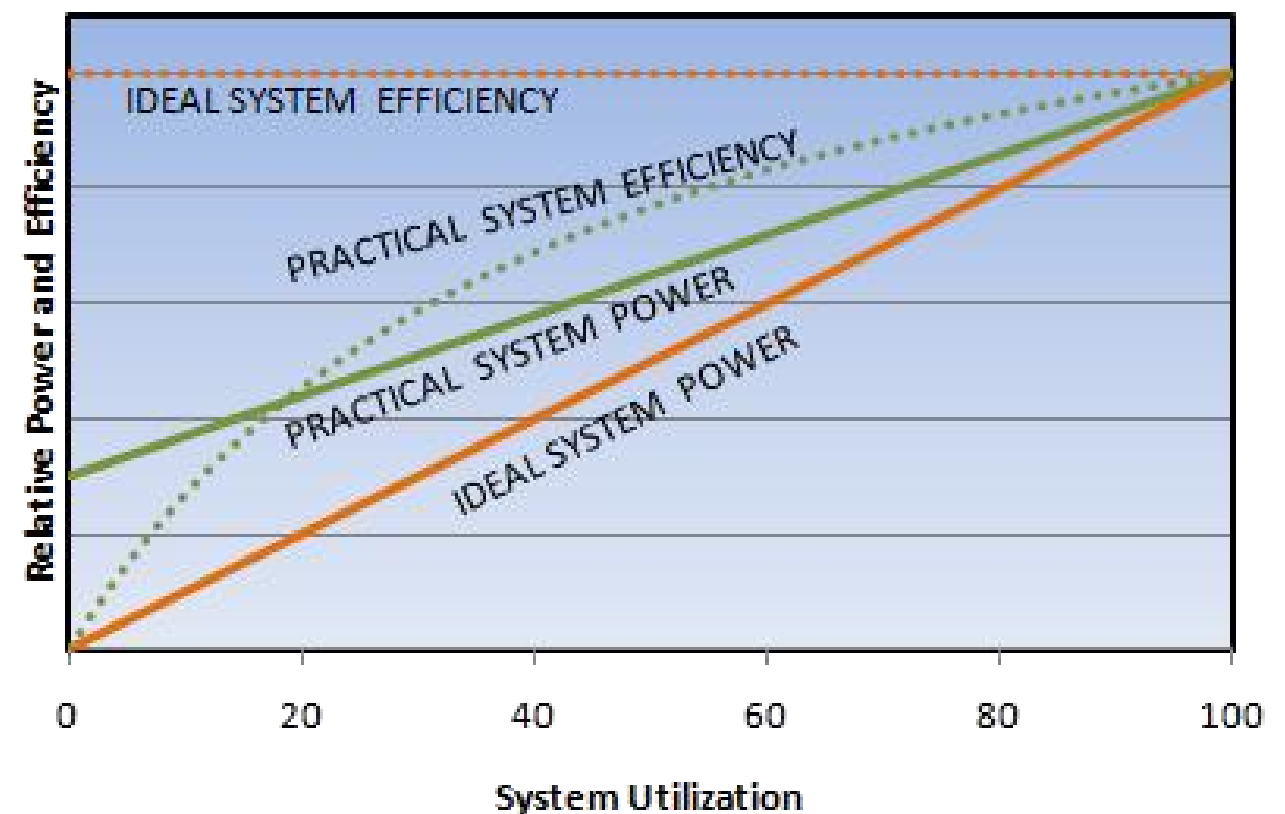
Load Profile of WSCs

- Average CPU utilization of 5,000 Google servers, 6 month period
- Servers rarely idle or fully utilized, operating most of the time at 10% to 50% of their maximum utilization



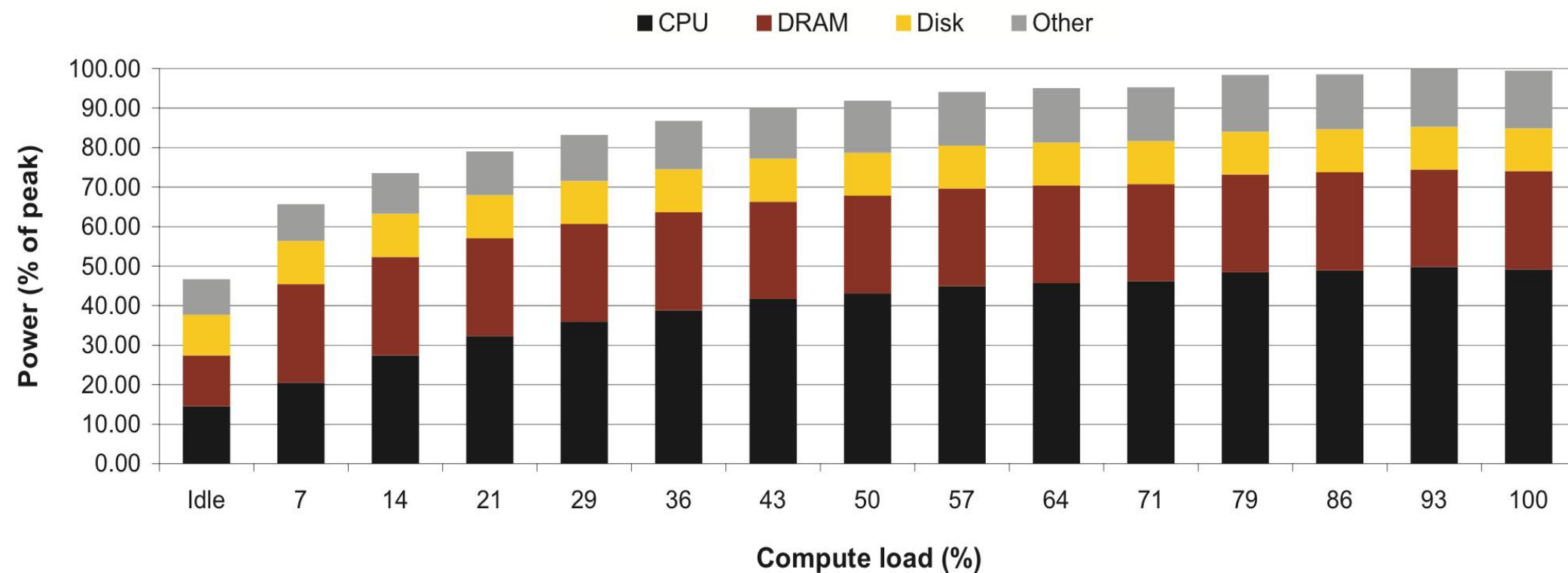
Energy-Proportional Computing: Design Goal of WSC

- Energy = Power x Time
- Efficiency = Computation/Energy
- Desire:
 - Consume almost no power when idle (Doing nothing well)
 - Gradually consume more power as the activity level increases



Cause of Poor Energy Proportionality

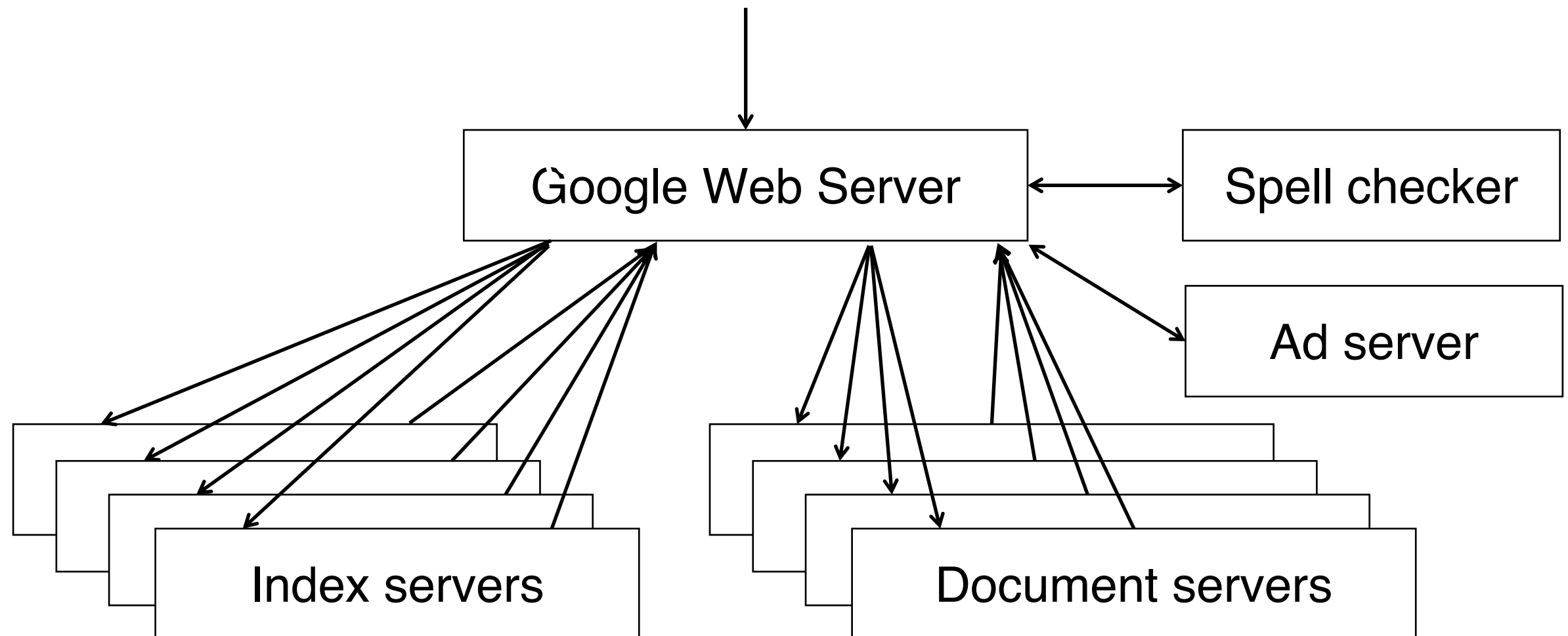
- DRAM, disks, networking: 70% at idle!
- Need to improve the energy efficiency of peripherals




More Parallelism

- Request-Level Parallelism (RLP)
 - Hundreds of thousands of requests per sec.
 - Popular Internet services like web search, social networking, ...
 - Such requests are largely independent
 - Often involve read-mostly databases
 - Rarely involve read-write sharing or synchronization across requests
 - Computation easily partitioned across different requests and even within a request

Google Query-Serving Architecture



Anatomy of a Web Search



Na Zha 2

AllImagesVideosNewsShort videosForumsWebMore

Tools

Any time

All results

Advanced Search





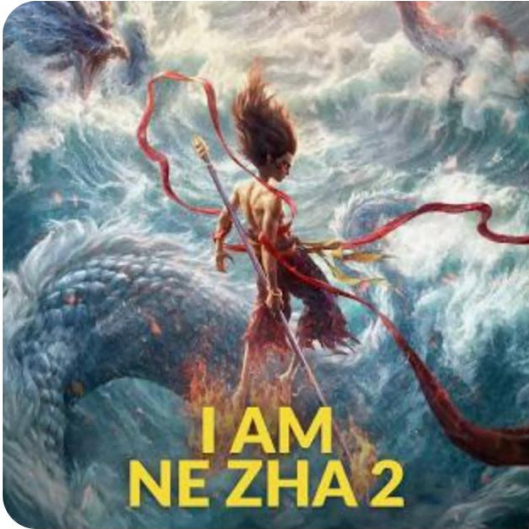
About 15,100,000 results (0.33s)

These are results for **Ne Zha 2**
Search instead for [Na Zha 2](#)

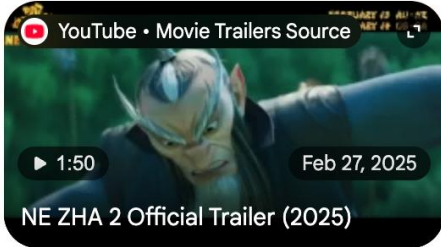
Ne Zha 2

Not Rated2025 · Fantasy/Animation · 2h 24m

OverviewReviewsCast





3DVF
Ne Zha 2 Confirms the Phenomenon: We Saw the Chinese...
A Chinese phenomenon coming soon to America. Since its release in China, Ne...
3 weeks ago




YouTube · Movie Trailers Source
1:50
Feb 27, 2025
NE ZHA 2 Official Trailer (2025)

Ratings


 IMDb8.1/10

 Rotten Tomatoes96%

 Wikipedia
https://en.wikipedia.org/wiki/Ne_Zha_2

Ne Zha 2

Ne Zha 2 is a 2025 Chinese animated fantasy action adventure film written and directed by Jiaozi. The direct sequel to *Ne Zha* (2019), it is based on the ...



About

8.1/10
IMDb

96%
Rotten Tomatoes

63%
Metacritic

94% liked this movie
Google users

After the catastrophe, although the souls of Nezha and Aobing were saved, their bodies would soon be shattered. Taiyi Zhenren planned to use the seven-colored lotus to rebuild their bodies.

Release date: February 14, 2025 (USA)
Director: [Yu Yang](#)

People also ask

Is Ne Zha 2 coming out?

Why is Ne Zha 2 so successful?

Anatomy of a Web Search (cont'd)

- Google “Ne Zha 2”
 - Direct request to “closest” Google WSC
 - Front-end load balancer directs request to one of many arrays (cluster of servers) within WSC
 - Within array, select one of many Google Web Servers (GWS) to handle the request and compose the response pages
 - GWS communicates with Index Servers to find documents that contains the search word, “Ne Zha 2”
 - Return document list with associated relevance score

Anatomy of a Web Search (cont'd)

- In parallel, Ad system: run ad auction for bidders on search terms
- Use docids (Document IDs) to access indexed documents
- Compose the page
 - Result document extracts (with keyword in context) ordered by relevance score
 - Sponsored links (along the top) and advertisements (along the sides)

Anatomy of a Web Search (cont'd)

- Implementation strategy
 - Randomly distribute the entries
 - Make many copies of data (a.k.a. “replicas”)
 - Load balance requests across replicas
- **Redundant copies** of indices and documents
 - Breaks up search hot spots, e.g., “Ne Zha 2”
 - Increases opportunities for **request-level parallelism**
 - Makes the system more **tolerant of failures**

Data-level Parallelism in WSC

- SIMD
 - Supports data-level parallelism in a single machine
 - Additional instructions & hardware
 - e.g., Matrix multiplication in memory
- DLP on WSC
 - Supports data-level parallelism across multiple machines
 - MapReduce & scalable file systems

Problem Statement

- How to process large amounts of raw data (crawled documents, request logs, ...) every day to compute derived data (inverted indices, page popularity, ...), when computation is conceptually simple but input data is large and distributed across 100s to 1000s of servers, so as to finish in reasonable time?
- Challenge: Parallelize computation, distribute data, tolerate faults without obscuring simple computation with complex code to deal with issues

Solution: MapReduce

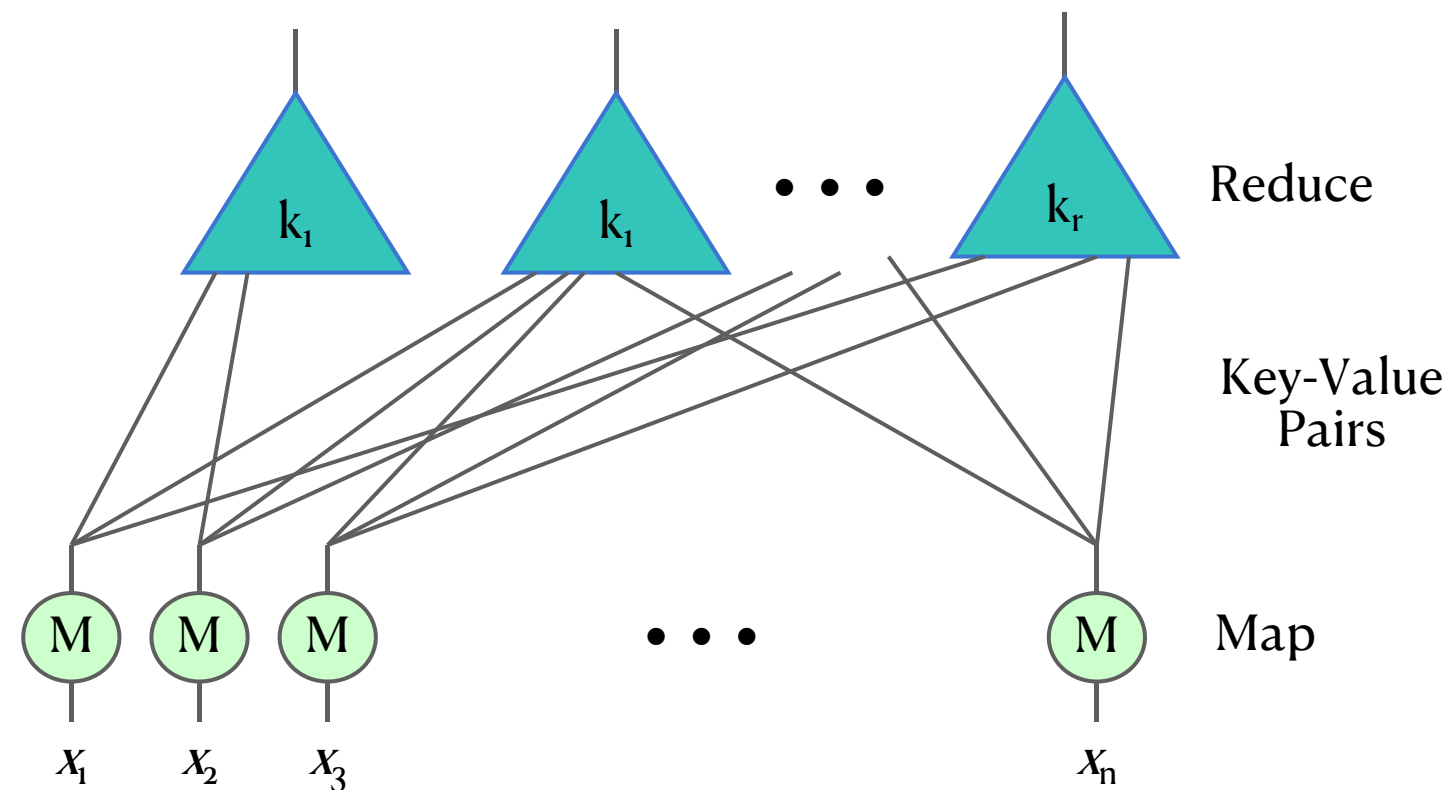
- Simple data-parallel programming model and implementation for processing large datasets
- Users specify the computation in terms of
 - a *map* function, and
 - a *reduce* function
- Underlying runtime system
 - Automatically parallelize the computation across large scale clusters of machines
 - Handles machine failure
 - Schedule inter-machine communication to make efficient use of the networks

MapReduce: Real Applications

- At Google
 - Index construction for Google Search
 - Article clustering for Google News
 - Statistical machine translation
 - For computing multi-layers street maps
- At Yahoo!
 - “Web map” powering Yahoo! Search
 - Spam detection for Yahoo! Mail
- At Facebook
 - Data mining
 - Ad optimization
 - Spam detection

MapReduce Programming Model

- Map computation across many objects
 - E.g., 10^{10} Internet web pages
- Aggregate results in many different ways
- System deals with issues of resource allocation & reliability



Inspiration: Map & Reduce Functions

- Calculate: $\sum_{i=1}^4 n^2$

- In Python

```
from functools import reduce
```

```
A = [1, 2, 3, 4]
```

```
def square(x):
```

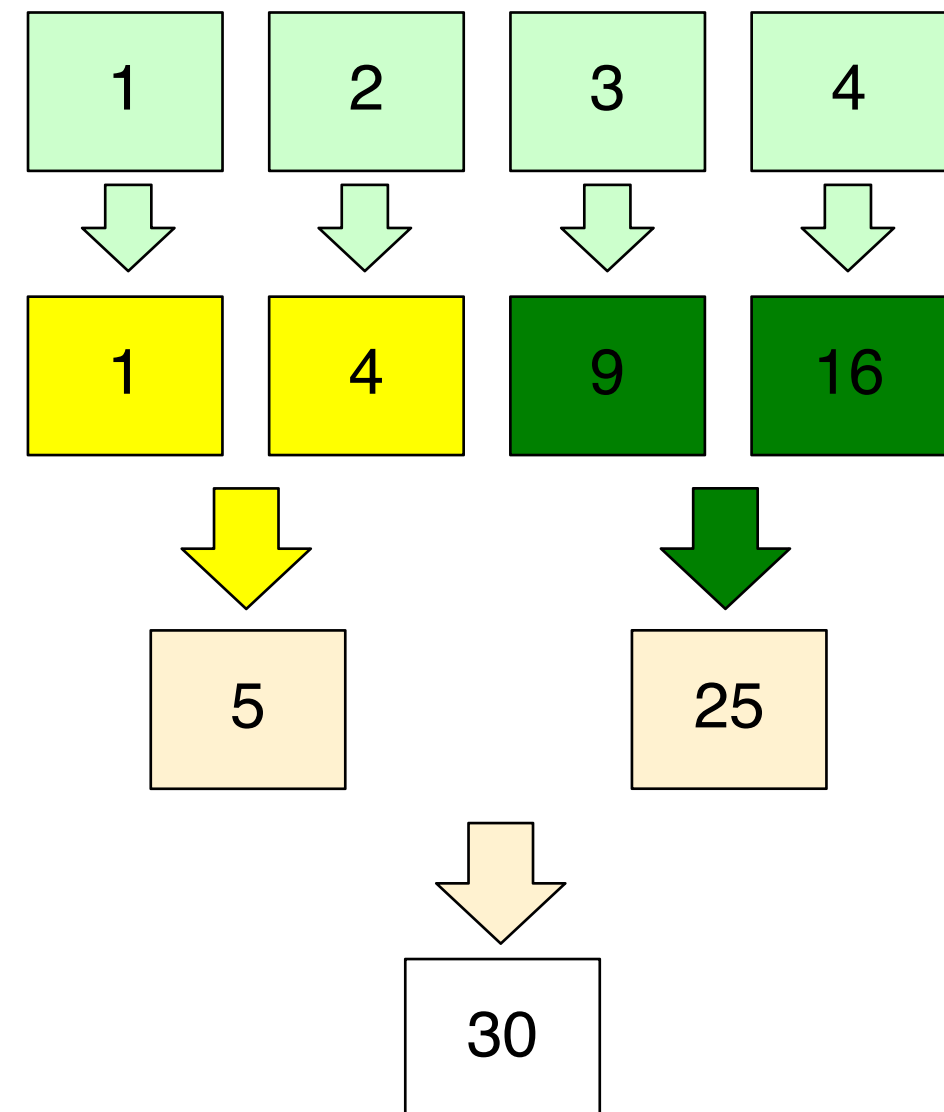
```
    return x * x
```

```
def sum(x, y):
```

```
    return x + y
```

```
reduce(sum,
```

```
    map(square, A))
```



MapReduce Programming Model

- **Map:** $(in_key, in_value) \rightarrow list(inter_key, inter_val)$
`map(in_key, in_val):`
`// DO WORK HERE`
`emit(inter_key, inter_val)`
 - Slice data into “shards” or “splits” and distribute to workers
 - Compute set of intermediate key/value pairs
- **Reduce:** $(inter_key, list(inter_value)) \rightarrow list(out_value)$
`reduce(inter_key, list(inter_val)):`
`// DO WORK HERE`
`emit(out_key, out_val)`
 - Combines all intermediate values for a particular key
 - Produces a set of merged output values (usually just one)

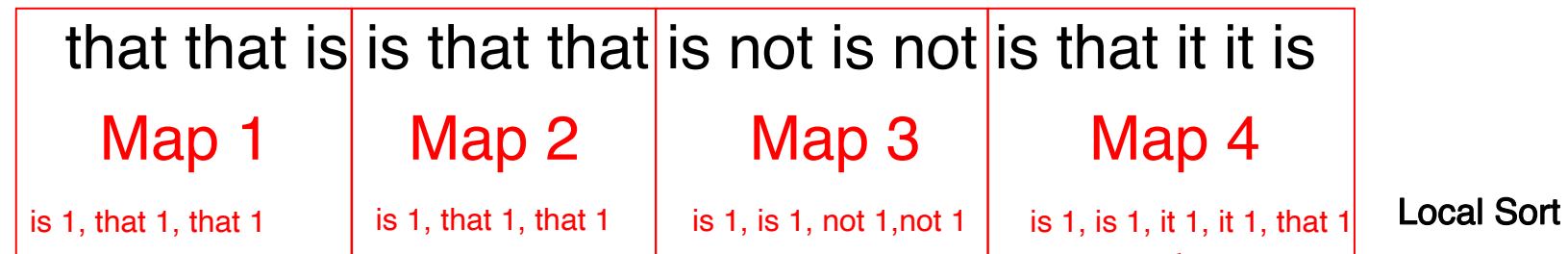
MapReduce Word Count Example

- **Distribute**

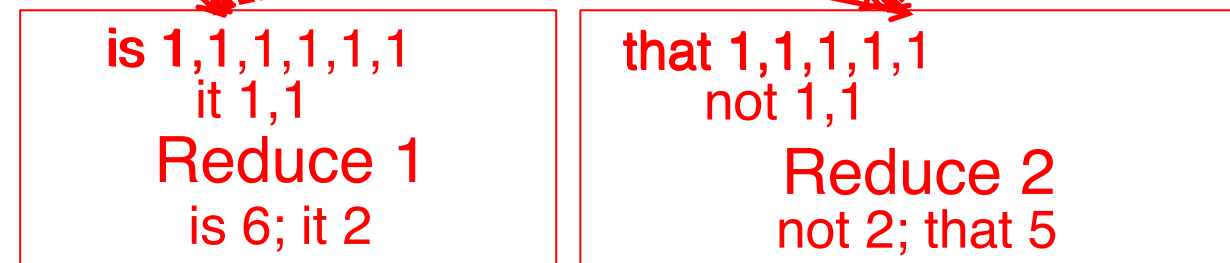
that that is	is that that	is not is not	is that it it is	Local Sort
Map 1	Map 2	Map 3	Map 4	
that 1, that 1, is 1 is 1, that 1, that 1	is 1, that 1, that 1 is 1, that 1, that 1	is 1, not 1, is 1, not 1 is 1, is 1, not 1, not 1	is 1, that 1, it 1, it 1, is 1 is 1, is 1, it 1, it 1, that 1	

MapReduce Word Count Example

- Distribute**

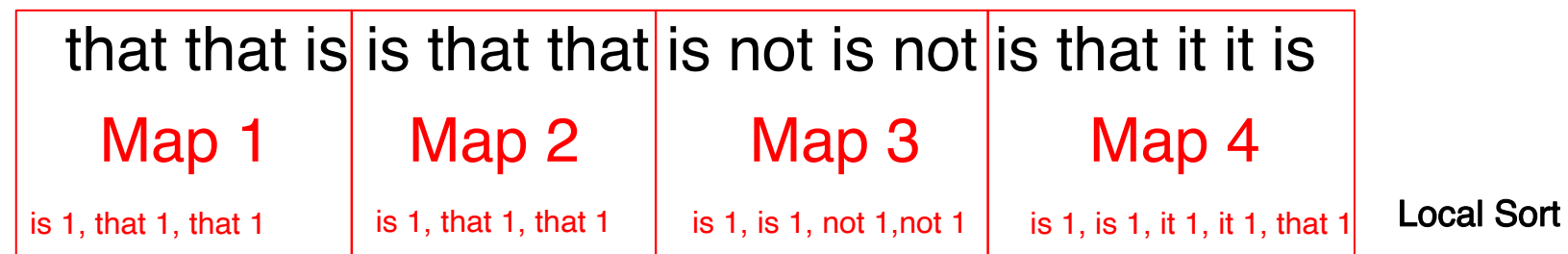


- Shuffle**

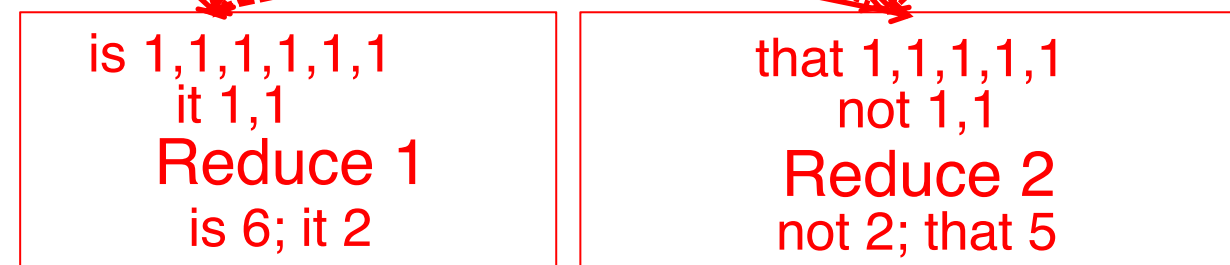


MapReduce Word Count Example

- Distribute**



- Shuffle**



- Collect**

is 6; it 2; not 2; that 5

Big Data Framework: Hadoop & Spark

- Apache Hadoop
 - Open-source MapReduce Framework
 - Hadoop Distributed File System (HDFS)
 - Hadoop YARN Resource Management
 - MapReduce Java APIs
 - more than half of the Fortune 50 used Hadoop (2013)
- Apache Spark
 - Fast and general engine for large-scale data processing.
 - Running on HDFS
 - Provides Java, Scala, Python APIs for
 - Database
 - Machine learning
 - Graph algorithm



Conclusion

- Warehouse-Scale Computers (WSCs)
 - New class of computers
 - Scalability, energy efficiency, high failure rate
- Cloud Computing
 - Benefits of WSC computing for third parties
 - “Elastic” pay as you go resource allocation
- Request-Level Parallelism
 - High request volume, each largely independent of other
 - Use replication for better request throughput, availability
- MapReduce Data Parallelism
 - Map: Divide large data set into pieces for independent parallel processing
 - Reduce: Combine and process intermediate results to obtain final result
 - Hadoop, Spark

Happy Dragon Boat Festival!

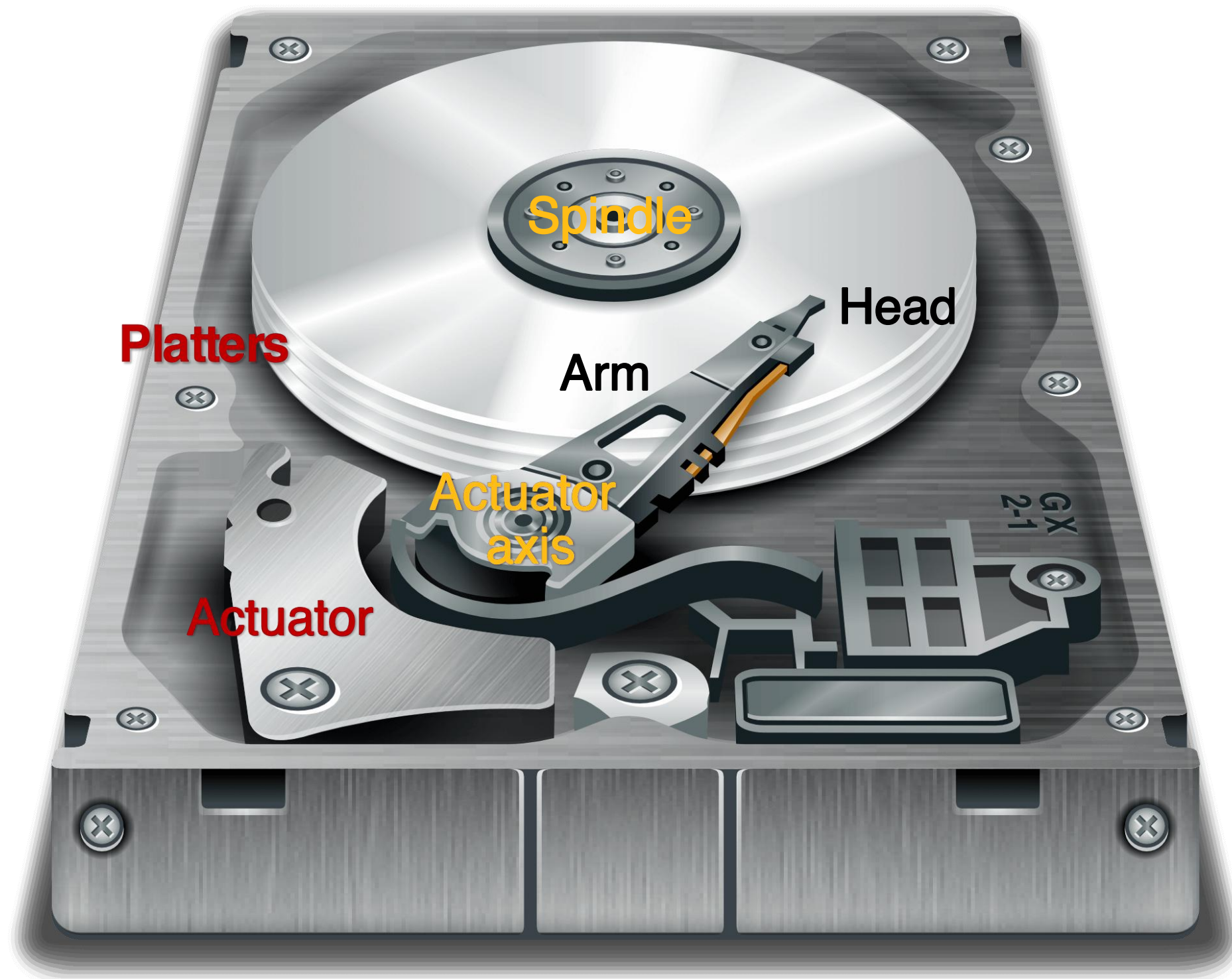


Slides after this page are appendix

Common I/O Devices: Magnetic Disk

- 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
 - **Hard Disk Drives (HDD)** – faster compared to tape, more dense, non-removable.
- Purpose in computer systems (Hard Drive):
 - Working file system + long-term backup for files
 - Secondary “backing store” for main-memory. Large, inexpensive, slow level in the memory hierarchy (virtual memory)

Internal Look



Internal Look

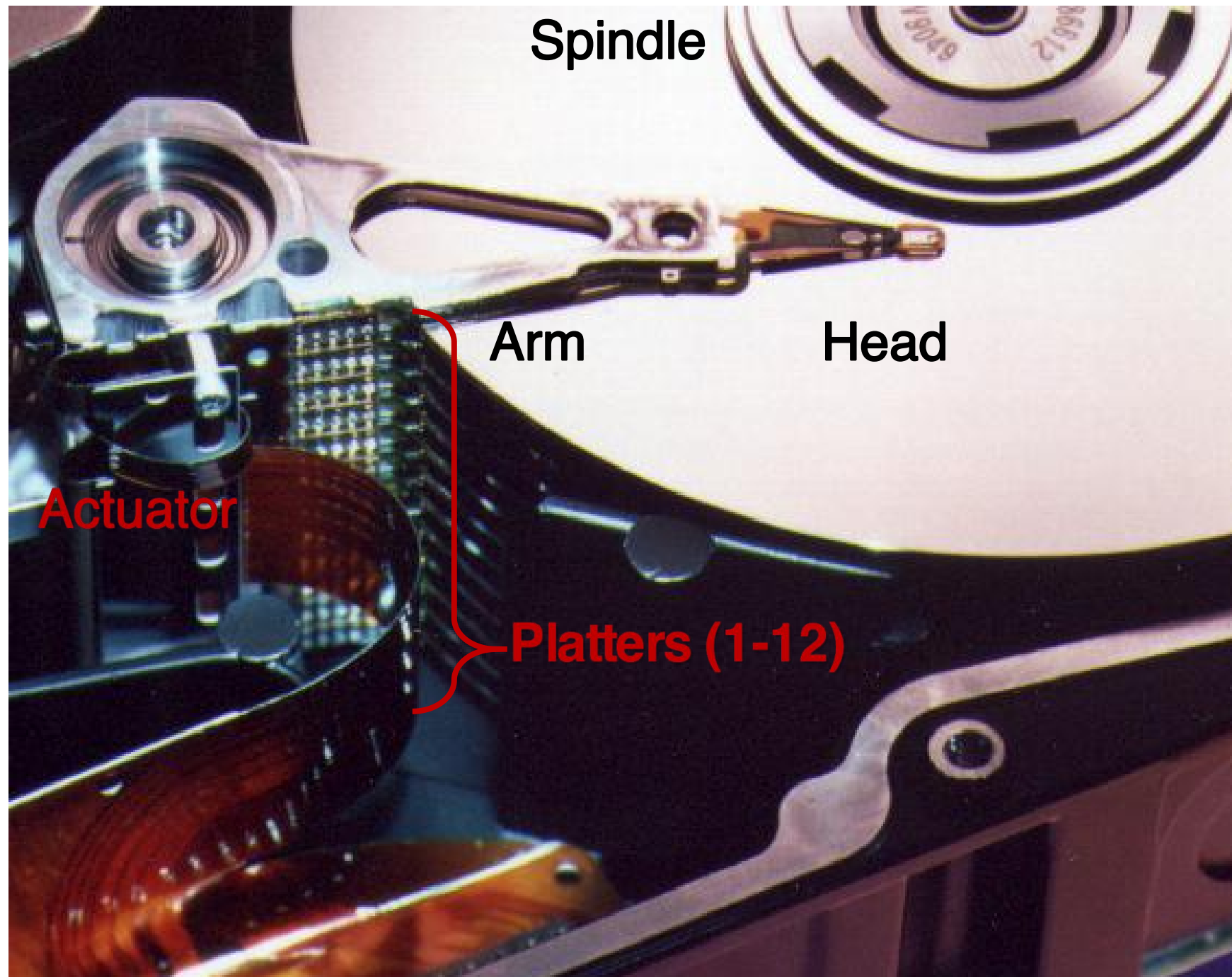
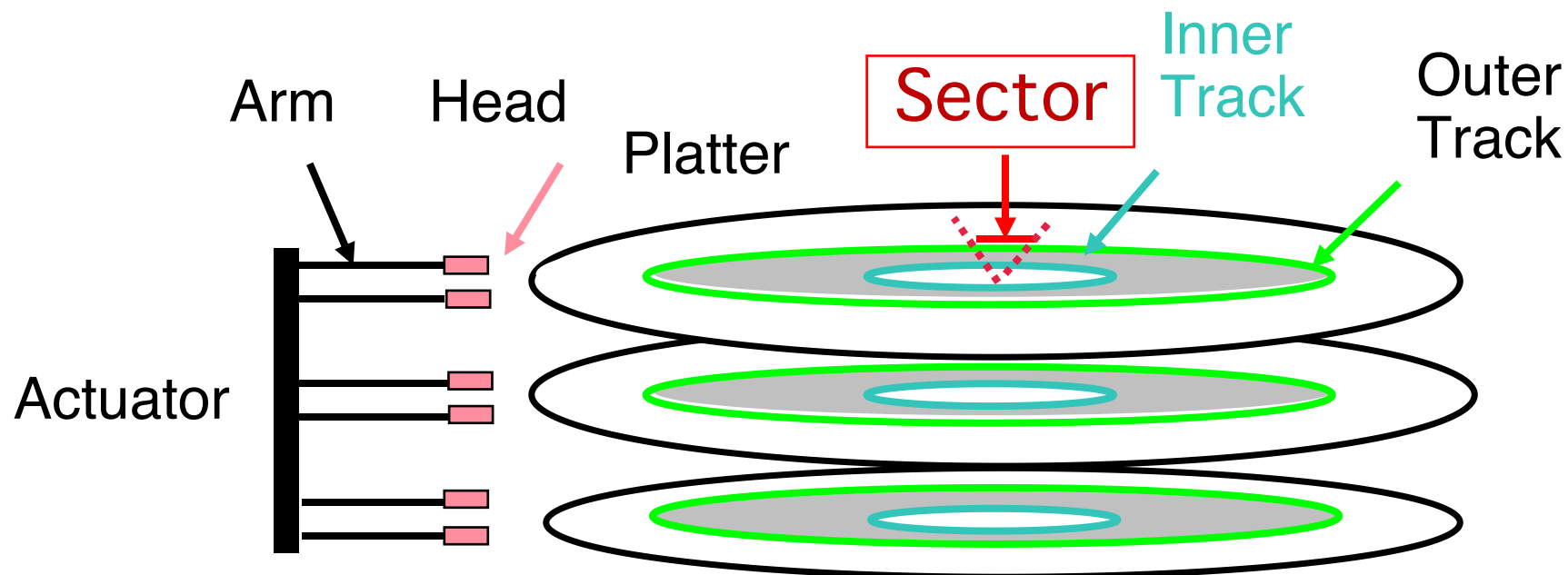


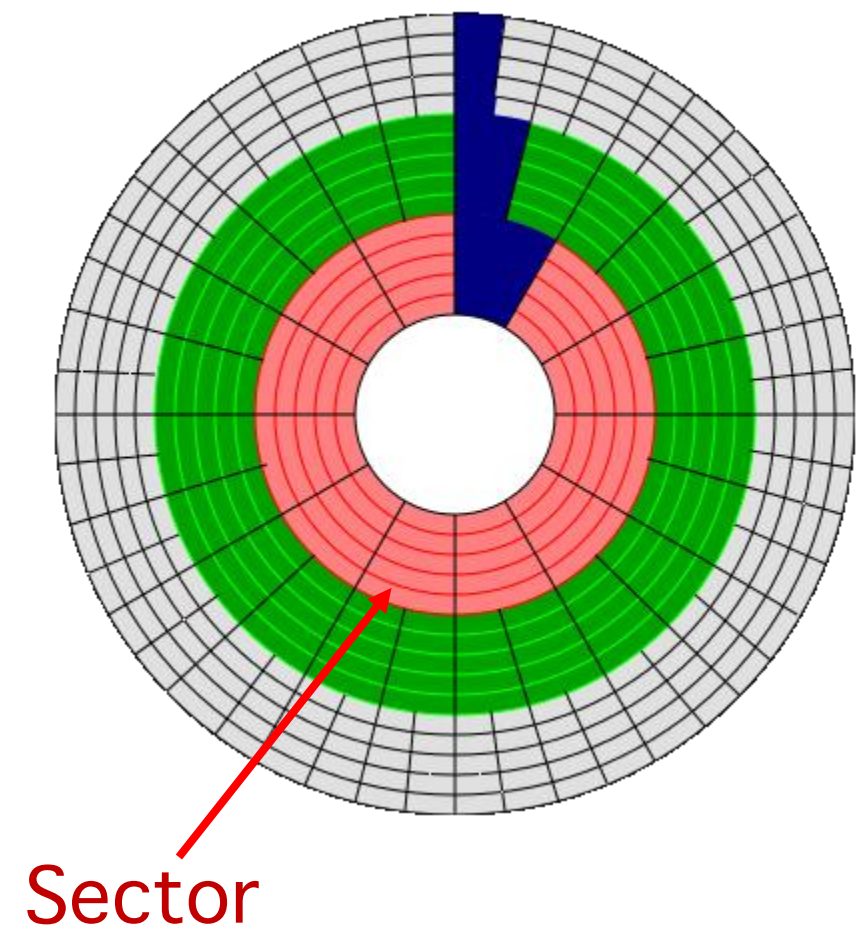
Photo of Disk Head, Arm, Actuator

Hard Disk Drive Terminologies

- Several platters, with information recorded magnetically usually on both surfaces
- 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

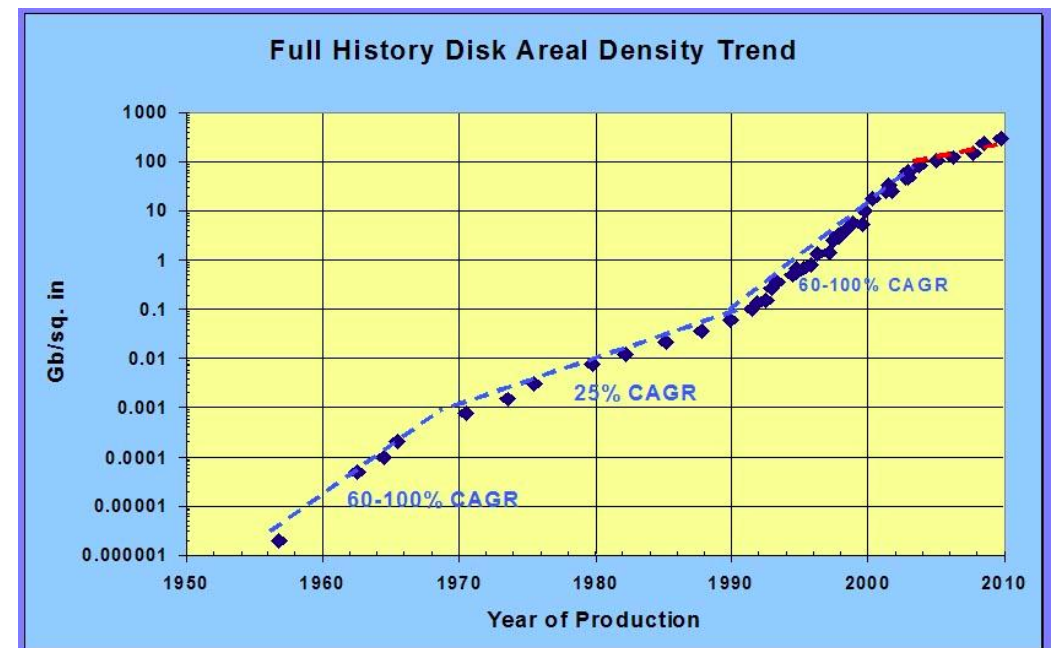
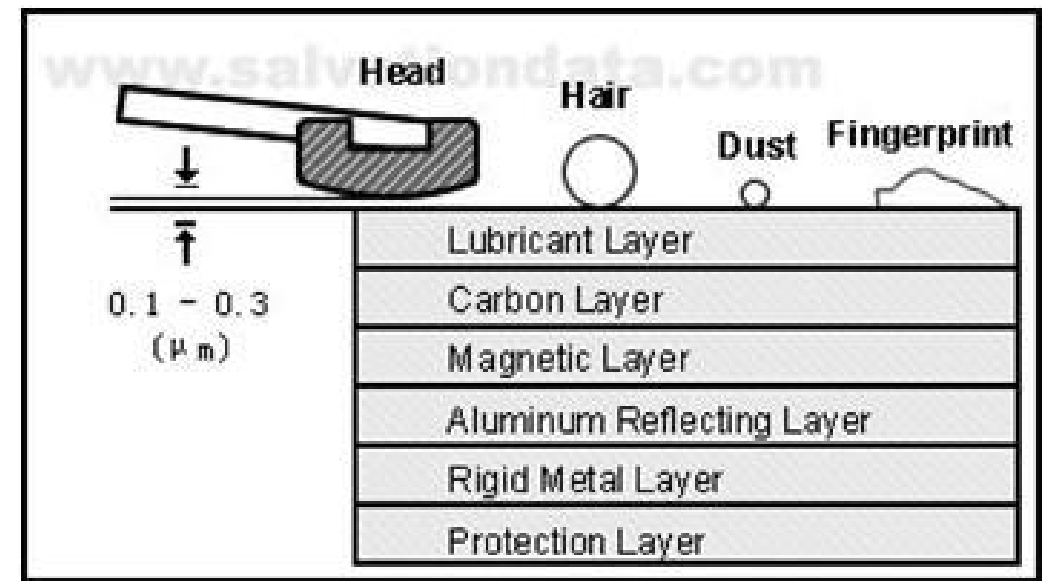


Platter bird's-eye view



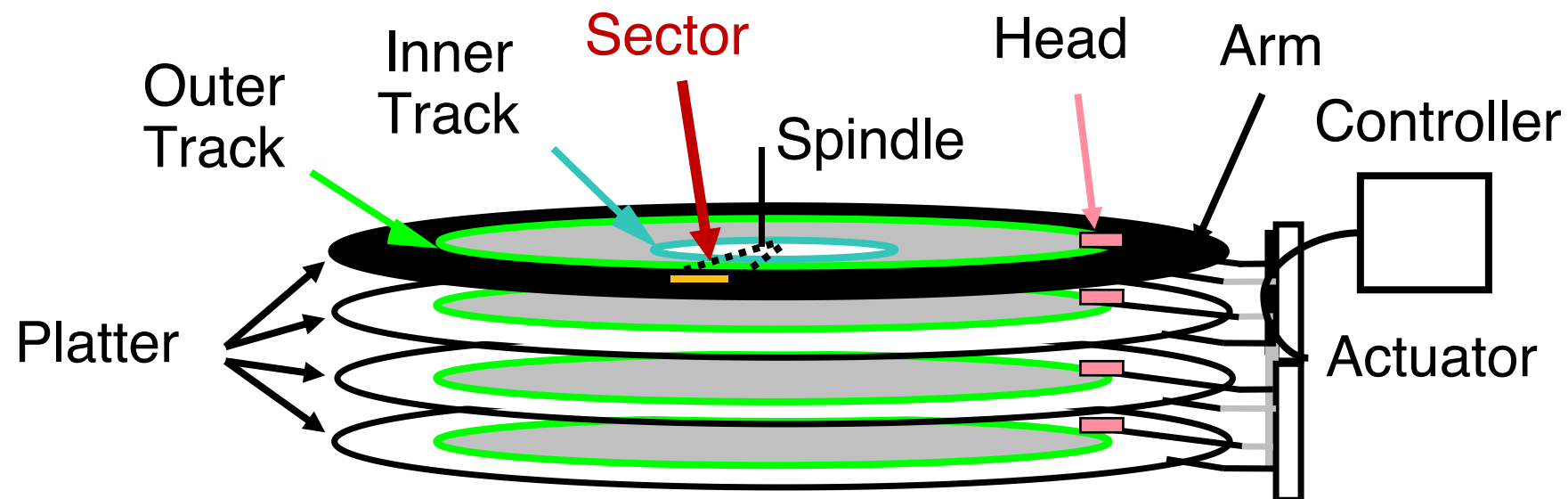
Hard Drives are Sealed. Why?

- The closer the head to the disk, the smaller the “spot size” and thus the denser the recording.
 - Measured in Gbit/in²
 - ~900 Gbit/in² is state of the art
 - Started out at 2 Kbit/in²
 - ~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 \times 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

Cloud Computing: Scale of Economy

May 2017 AWS Instances & Prices

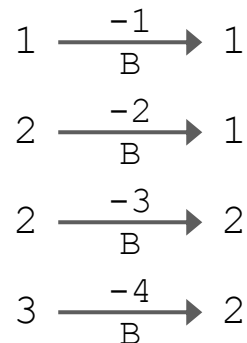
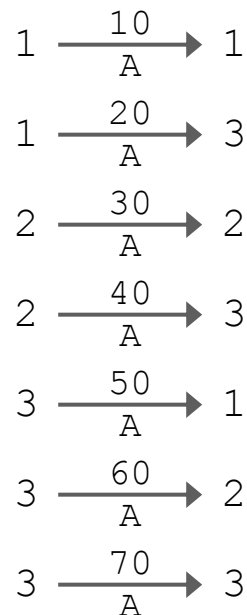
- Closest computer in WSC example is Standard Extra
- At these low rates, Amazon EC2 can make money!
 - even if used only 50% of time
- Virtual Machine (VM) plays an important role

Name	Memory	vCPUs	Storage	Arch	Network Performance	Linux On Demand
M1 General Purpose Small	1.7 GB	1	160 GB	32/64-bit	Low	\$0.044 hourly
M1 General Purpose Medium	3.75 GB	1	410 GB	32/64-bit	Moderate	\$0.087 hourly
M1 General Purpose Extra Large	15.0 GB	4	1680 GB	64-bit	High	\$0.35 hourly
C1 High-CPU Medium	1.7 GB	2	350 GB	32/64-bit	Moderate	\$0.13 hourly
C1 High-CPU Extra Large	7.0 GB	8	1680 GB	64-bit	High	\$0.52 hourly
I2 Extra Large	30.5 GB	4	800 GB	64-bit	Moderate	\$0.853 hourly
I2 Double Extra Large	61.0 GB	8	1600 GB	64-bit	Moderate	\$1.705 hourly
M4 Large	8.0 GB	2	EBS only	64-bit	Moderate	\$0.108 hourly
M4 Extra Large	16.0 GB	4	EBS only	64-bit	High	\$0.215 hourly
M4 16xlarge	256.0 GB	64	EBS only	64-bit	20 Gigabit	\$3.447 hourly
General Purpose GPU Extra Large	61.0 GB	4	EBS only	64-bit	High	\$0.9 hourly
General Purpose GPU 16xlarge	732.0 GB	64	EBS only	64-bit	20 Gigabit	\$14.4 hourly
X1 Extra High-Memory 16xlarge	976.0 GB	64	1920 GB	64-bit	10 Gigabit	\$6.669 hourly

Example: Sparse Matrix Multiplication

- Compute matrix multiplication with plenty of 0's
- Challenging for parallel execution
- Demonstrate expressiveness of Map/Reduce

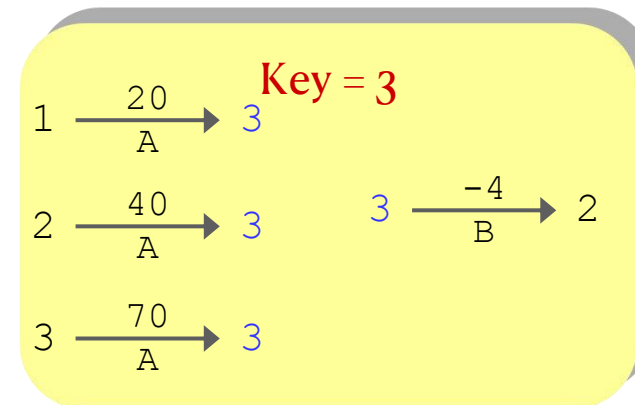
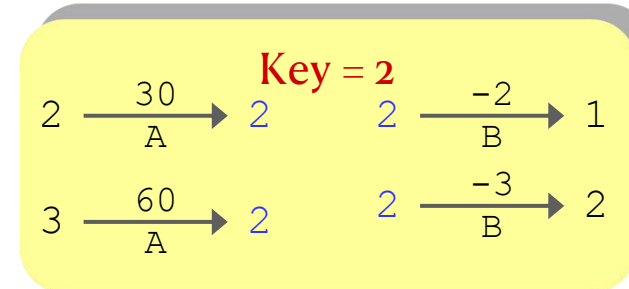
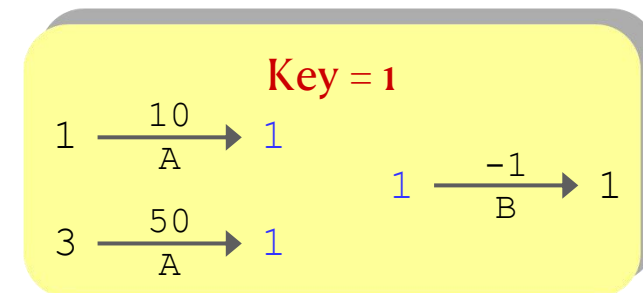
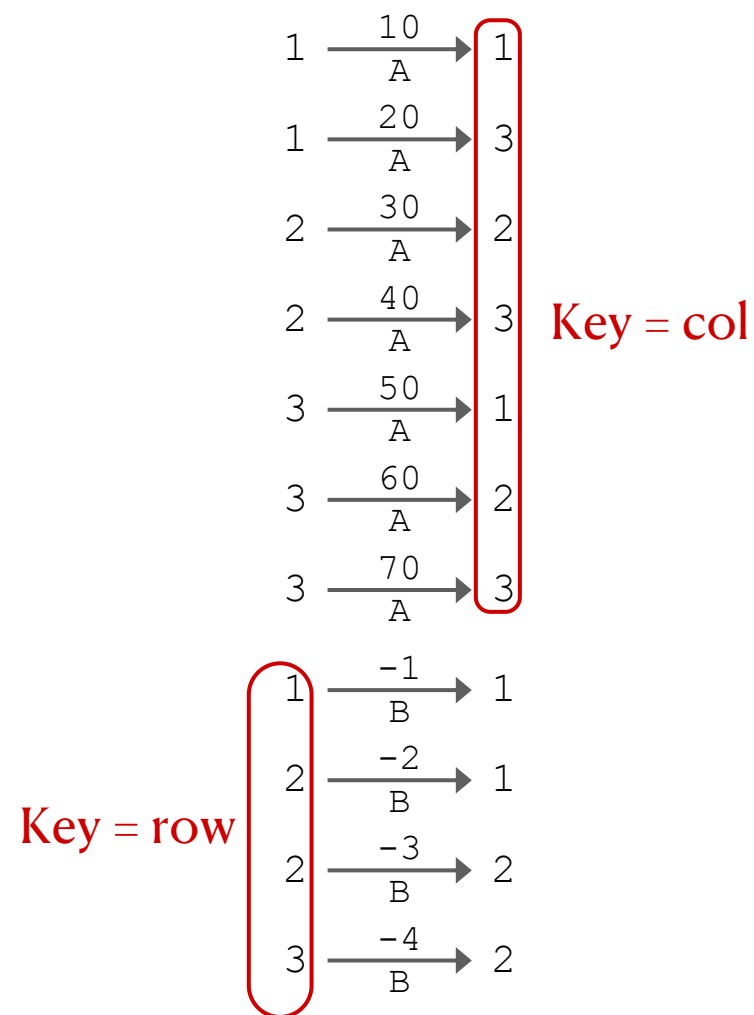
$$\begin{matrix} & A & & B & & C \\ \begin{bmatrix} 10 & & 20 \\ & 30 & 40 \\ 50 & 60 & 70 \end{bmatrix} & \times & \begin{bmatrix} -1 \\ -2 & -3 \\ & & -4 \end{bmatrix} & = & \begin{bmatrix} -10 & -80 \\ -60 & -250 \\ -170 & -460 \end{bmatrix}
 \end{matrix}$$



- Represent matrix as list of nonzero entries
 $\langle \text{row, col, value, matrixID} \rangle$
- 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

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



Phase 1 “Reduce” of Matrix Multiply

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

Key = 1

$$\begin{array}{lcl} 1 & \xrightarrow[A]{10} & 1 \\ 3 & \xrightarrow[A]{50} & 1 \end{array} \quad \mathbf{X} \quad \begin{array}{lcl} 1 & \xrightarrow[B]{-1} & 1 \end{array}$$

Key = 2

$$\begin{array}{lcl} 2 & \xrightarrow[A]{30} & 2 \\ 3 & \xrightarrow[A]{60} & 2 \end{array} \quad \mathbf{X} \quad \begin{array}{lcl} 2 & \xrightarrow[B]{-2} & 1 \\ 2 & \xrightarrow[B]{-3} & 2 \end{array}$$

Key = 3

$$\begin{array}{lcl} 1 & \xrightarrow[A]{20} & 3 \\ 2 & \xrightarrow[A]{40} & 3 \\ 3 & \xrightarrow[A]{70} & 3 \end{array} \quad \mathbf{X} \quad \begin{array}{lcl} 3 & \xrightarrow[B]{-4} & 2 \end{array}$$

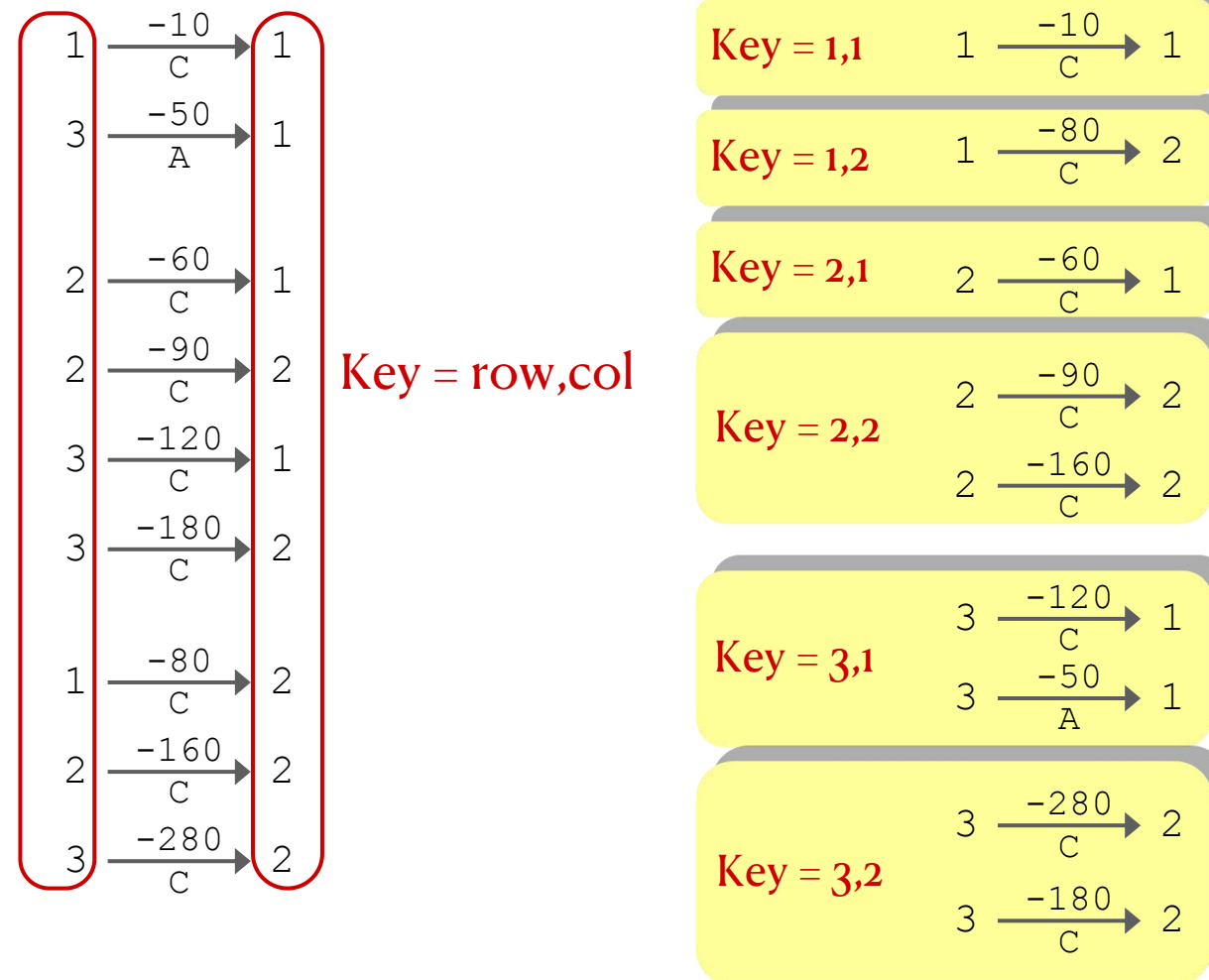
$$\begin{array}{lcl} 1 & \xrightarrow[C]{-10} & 1 \\ 3 & \xrightarrow[A]{-50} & 1 \end{array}$$

$$\begin{array}{lcl} 2 & \xrightarrow[C]{-60} & 1 \\ 2 & \xrightarrow[C]{-90} & 2 \\ 3 & \xrightarrow[C]{-120} & 1 \\ 3 & \xrightarrow[C]{-180} & 2 \end{array}$$

$$\begin{array}{lcl} 1 & \xrightarrow[C]{-80} & 2 \\ 2 & \xrightarrow[C]{-160} & 2 \\ 3 & \xrightarrow[C]{-280} & 2 \end{array}$$

Phase 2 “Map” of Matrix Multiply

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



Phase 2 “Reduce” of Matrix Multiply

- Sum partial sums to get final results

Key = 1,1 $1 \xrightarrow[\text{C}]{-10} 1$

Key = 1,2 $1 \xrightarrow[\text{C}]{-80} 2$

Key = 2,1 $2 \xrightarrow[\text{C}]{-60} 1$

Key = 2,2 $2 \xrightarrow[\text{C}]{-90} 2$
 $2 \xrightarrow[\text{C}]{-160} 2$

Key = 3,1 $3 \xrightarrow[\text{C}]{-120} 1$
 $3 \xrightarrow[\text{A}]{-50} 1$

Key = 3,2 $3 \xrightarrow[\text{C}]{-280} 2$
 $3 \xrightarrow[\text{C}]{-180} 2$

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

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

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

$2 \xrightarrow[\text{C}]{-250} 2$

$3 \xrightarrow[\text{C}]{-170} 1$

$3 \xrightarrow[\text{C}]{-460} 2$

C

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

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