Introduction to Scientific Computing

Le Yan

HPC @ LSU

Goals

- Get familiar with scientific computing concepts and terminology
- Establish the basic understanding of computing systems from the perspective of an effective scientific computing practitioner

Outline

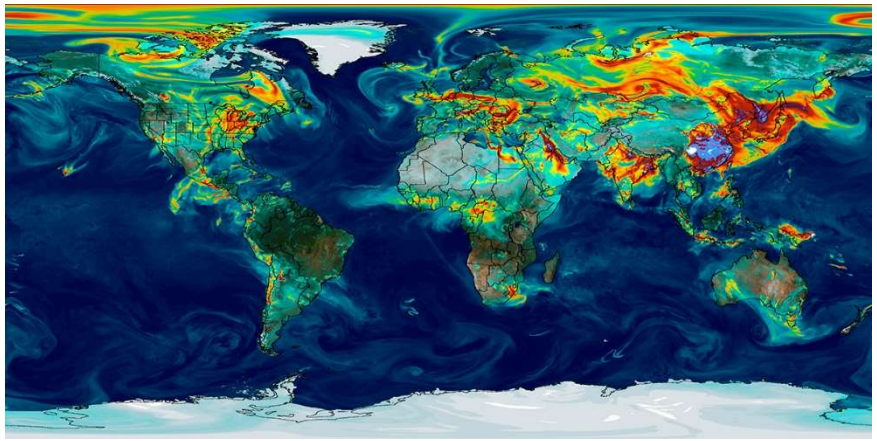
- Basic concepts and terminology
- Computing systems
- Parallel computing

What Is Scientific Computing

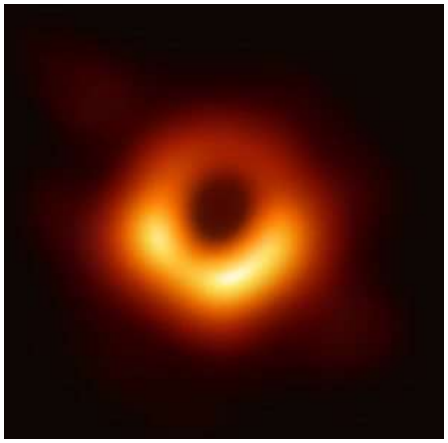
- Use computers and other computing technologies to gain scientific insights
  - Numerical simulations of physical phenomena
  - Data analytics
  - …
- Third pillar of science
  - In addition to “theoretical analysis” and “experiments”

Why Scientific Computing

- Sometimes other means are impossible



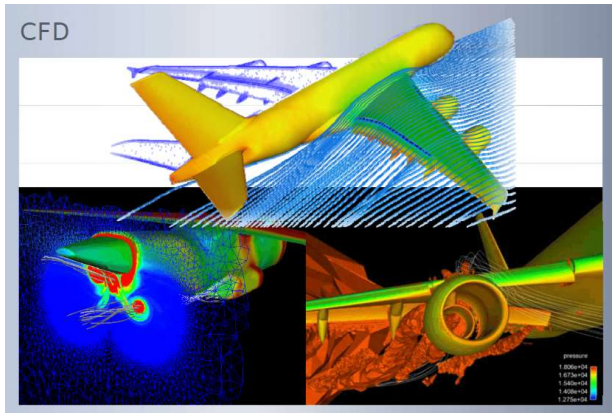
Climate research



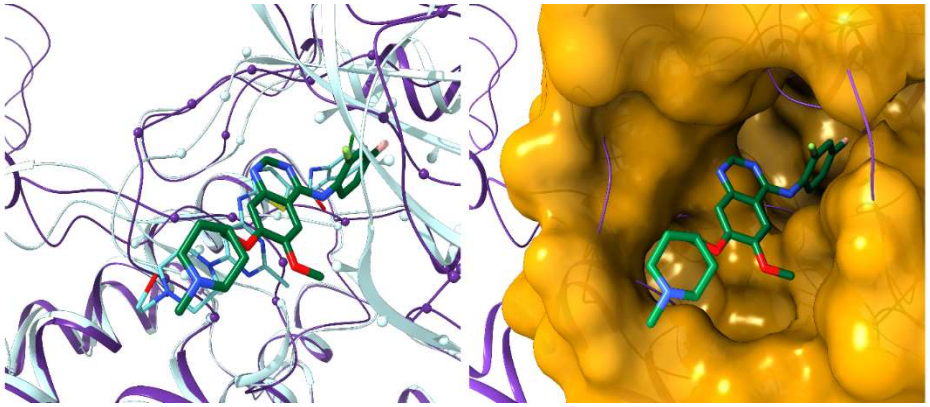
Astrophysics

Why Scientific Computing

– Sometimes other means are costly (time and money)



Aircraft design



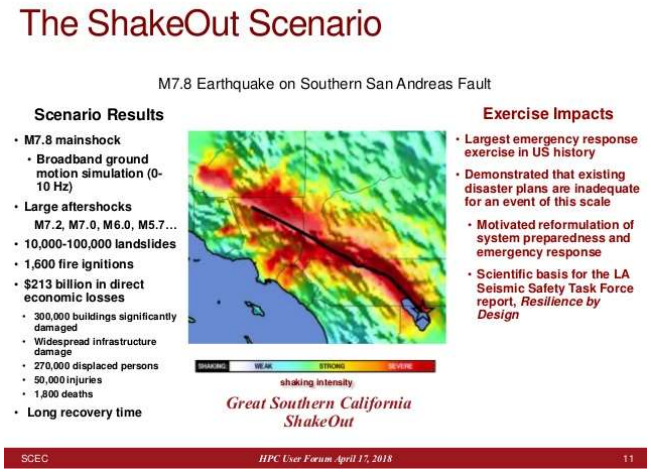
Drug discovery

Why Scientific Computing

- Sometimes experiments are dangerous or undesirable

In lieu of testing nuclear weapons, second-generation designers judge the condition of the aging stockpile based on tests of weapon subsystems, computer simulations of both physics phenomena (shown here) and weapon behavior, and knowledge gained from past nuclear tests. (Photo: Los A

Nuclear weapon tests



Natural disasters

Scientific Computing: An Example

What is the connection between Pringles and scientific computing?

Scientific Computing: An Example

What is the connection between Pringles and scientific computing?

P&G used computer simulation to design the shape of Pringles chips so that they do not fly off the fast-moving production line.

Elements Of Scientific Computing

Hardware

Software

python

OPEN MPI

ANSYS

TensorFlow

gatk

aws

Google Cloud

Microsoft Azure

LSU CENTER FOR COMPUTATION & TECHNOLOGY

LONI

High Performance Computing (HPC)

- It is great if a PC can satisfy your needs
- However
  - Many problems **cannot fit on a PC** - usually because they need more than a few GB of RAM, or more than a few TB of disk
  - Many problems **would take a very long time to run on a PC**, e.g. months or even years
- If a single computer does not cut it, how about using bunch of them?
  - That is high performance computing
- HPC systems, supercomputers, clusters: nowadays they more or less mean the same thing
  - A number of computers (nodes) connected via high speed network (interconnect) so they can solve a problem together

How Super Are Supercomputers?

- **Fl**oating-point **O**perations **P**er **S**econd (FLOPS) is the metric used to measure how powerful a supercomputer is
  - How many numbers it can crunch in a second
- The most powerful supercomputer today
  - Has millions of processing cores
  - Operates in the ~100 petaflops (10^{17}) range
  - Consumes 13 MegaWatts of power
    - US power consumption (per capita): ~1300 Watts
- You can find the 500 most powerful supercomputers in the world (in the public domain) at top500.org

TOP500® SUPERCOMPUTER SITES

Home / Lists / November 2018 / List

TOP500 List - November 2018

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

R_{peak} values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

previous | 1 | 2 | 3 | 4 | 5 | next

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,397,824	143,500.0	200,794.9	9,783
2	DOE/NNSA/LLNL United States	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM / NVIDIA / Mellanox	1,572,480	94,640.0	125,712.0	7,438
3	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
4	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	100,678.7	18,482
5	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	387,872	21,230.0	27,154.3	2,384

Science At Scale

- ACM Gordon Bell Prize recognizes outstanding achievement in high-performance computing applications
- The 2017 Winner
  - Simulated one of the most devastating earthquake in history
  - On the then No. 1 supercomputer in the world
  - With more than 10 million CPU cores
  - Reached a sustained speed of 1.9×10^{16} FLOPS

Super Cell Phones?

- Chances are your smartphone has 4, 6, even 8 cores
- The processing power would be in the neighborhood of a few hundred GigaFLOPS
  - 10^{11} operations per second
  - Would make the top 500 list 15 years ago (2004)

High Throughput Computing (HTC)

- The strength of supercomputers does not only lie in the capability of solving huge problems
- They are very good at processing many, many small problems as well
  - Running them concurrently will greatly shorten time-to-solution
- People sometimes refer to this paradigm as “high throughput computing”

Scientific Computing In Cloud

- Cloud – "someone else's computer"
- Scientific computing in cloud has been growing rapidly in recent years
- Strength
  - Elasticity
  - Flexibility
  - Resilience
- Weakness
  - Need to manage the cost
  - Still not very efficient for certain types of scientific computing workloads

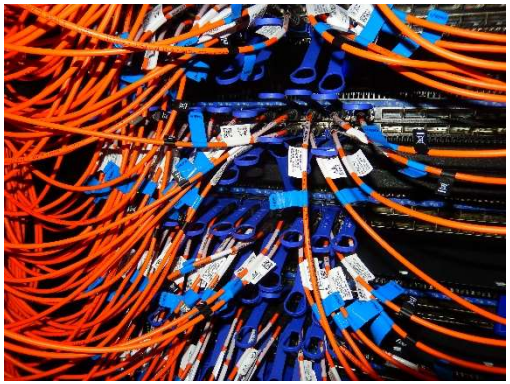
Scientific Computing Software

- Use third party applications
- Write your own code
  - Programming languages
  - Libraries
  - Development tools
- Often we need both

Louisiana Optical Network Infrastructure (LONI)

- State-of-the-art optical fiber network connecting Louisiana research institutions and other organizations to each other as well as national research networks
- Owned by Louisiana Board of Regents
- LONI means two things for its end users
  - **Free** high speed network (you are already using it)
  - **Free** high performance computing resources

LONI QB2 Cluster



Cluster	QB2
# of nodes	504
# of cores	10,080
Total memory	32 TBytes
Total storage	2.8 PBytes
Note	Most nodes are equipped with NVIDIA GPUs, ideal for deep learning workload

Software On LONI HPC Clusters

- **Molecular dynamics packages**: NAMD, Amber, GROMACS…
- **Statistics packages**: R
- **Bioinformatics applications**: Qiime, BLAST, Bowtie, HMMER, Mothur, SOAPdenovo, VELVET…
- **Compilers**: Intel, PGI, GNU…
- **Parallel processing libraries**: MPI, CUDA…
- **Numerical libraries**: MKL, FFTW, PETSc…
- **User installed packages**

User Consulting Services

- Answer HPC-related questions from users
- Enable users to use LONI HPC resource efficiently
  - Provide cluster support such as software installation and code development
  - Help resolve issues on clusters
  - Provide user training
- Collaborate with users on research projects

Other Computing Resources

- NSF XSEDE
  - Compute time for both research and education on supercomputers supported by XSEDE resource providers
  - Trainings
- DOE INCITE
  - Compute time for research on top-ranked supercomputers housed at DOE national laboratories
- Both free of charge
  - And competitive

Practicing Scientific Computing

- Find the right tools
  - Workstation vs. Clusters vs. Cloud
  - Writing own code vs. using third-party software
- Learn about the tools
  - Any limitation of capacity and functionality?
- Be aware of how the tools perform
  - Does my job finish in the expected amount of time?
  - Is there any stress sign from the hardware?

Computers As Instruments

- Understanding how the instruments work is a key element to achieve success
- If computing systems will be your (primary) research instrument, knowing how they work will help maximize your productivity

Productivity: Performance Awareness

- Would you be suspicious if it takes
  - 5 minutes to launch Microsoft Word on your laptop?
  - 15 minutes to download a file from Dropbox?
  - 15 hours to assembly and align a 5GB sequence?

Productivity: Performance Awareness

- Fortnite (or Youtube or whatever app) is laggy. What would you do?
  - Network?
  - Lots of other programs running on your computer?
  - Fans not running anymore?
  - …

Productivity Matters

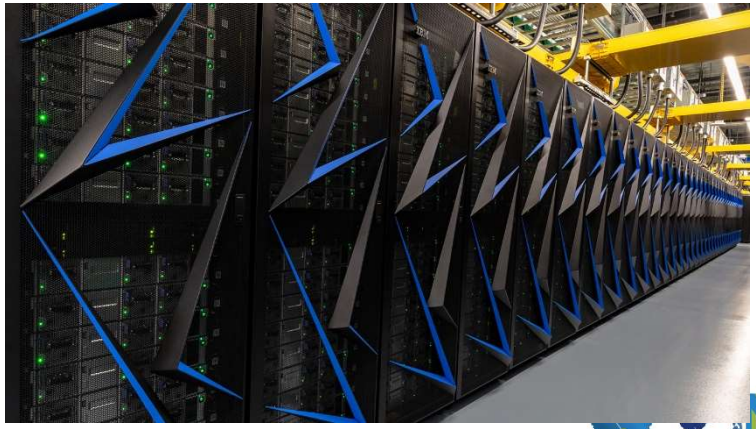
Know how long a workload should run (and know something is wrong when it takes longer)
Know where to look when something is wrong
Write Python scripts to automate pre- and post-processing

Productivity Matters

Know how long a workload should run (and know something is wrong when it takes longer)	Don't know how long a workload should run (and don't know something is wrong when it takes longer)
Know where to look when something is wrong	Don't know where to look when something is wrong
Write Python scripts to automate pre- and post-processing	Don't write python scripts to automate lots of pre-processing and post-processing, i.e. do everything by hand

On Supercomputers, Too

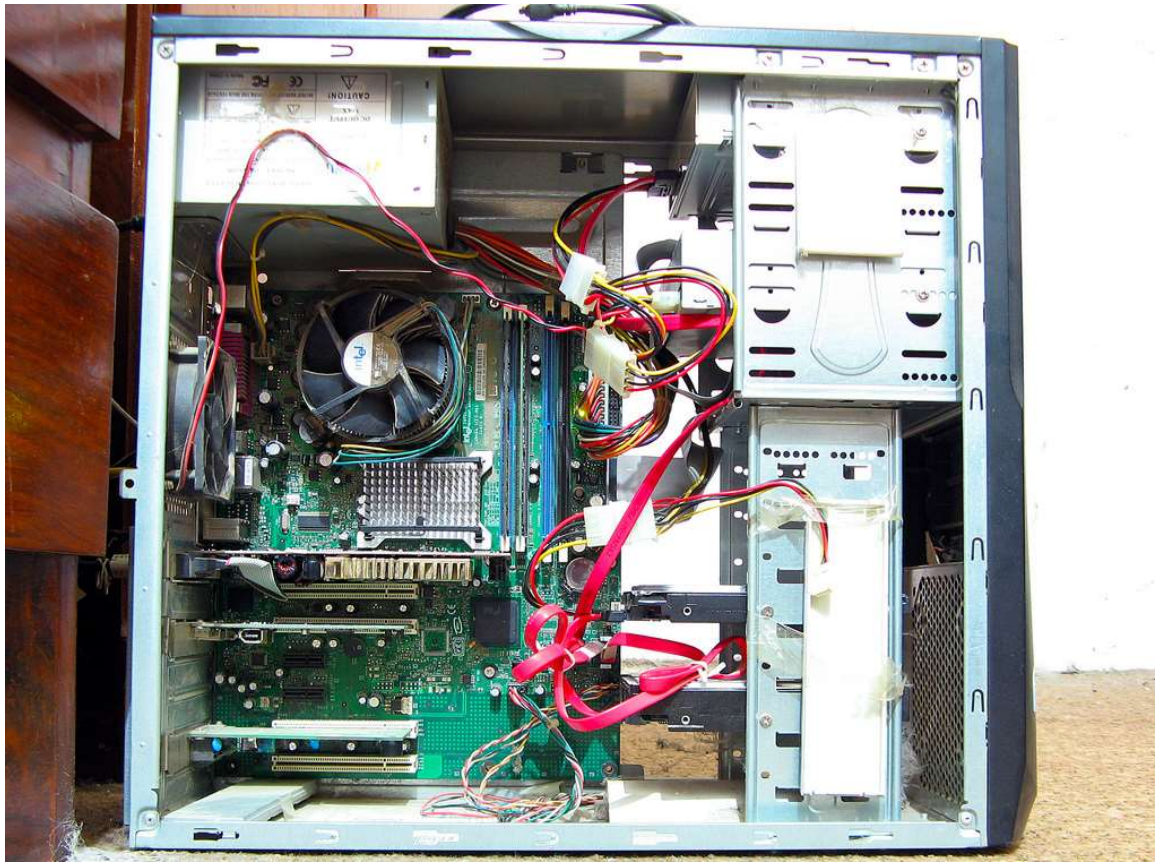
- Supercomputers are truly “super”
- But being “super” doesn’t necessarily make performance issues go away
  - Sometimes it make things even more complicated



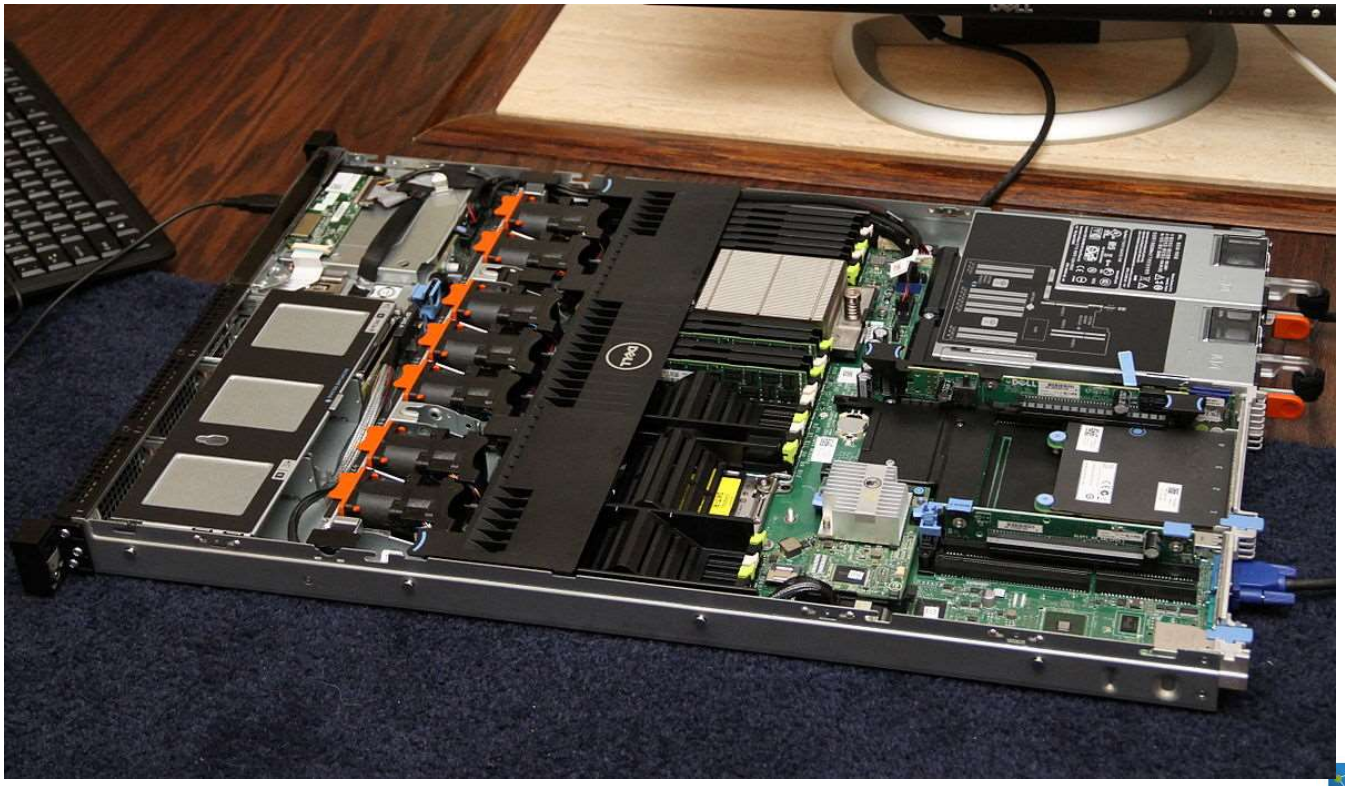
Outline

- Basic concepts and terminology
- Computing systems
- Parallel computing

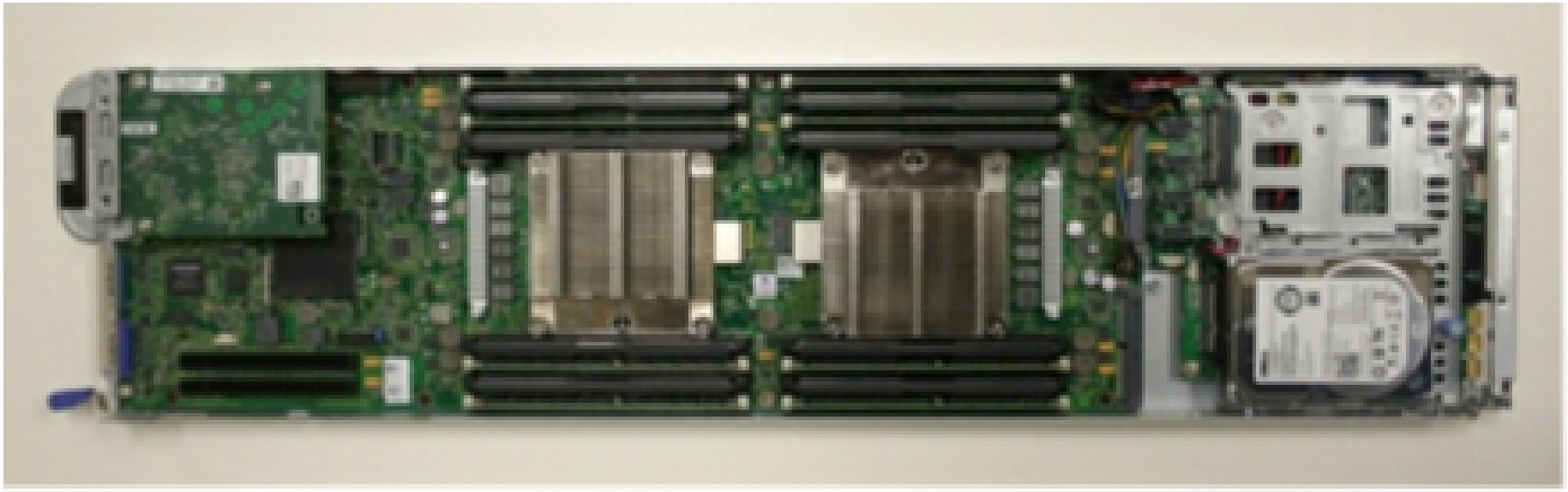
Computers Of All Shapes



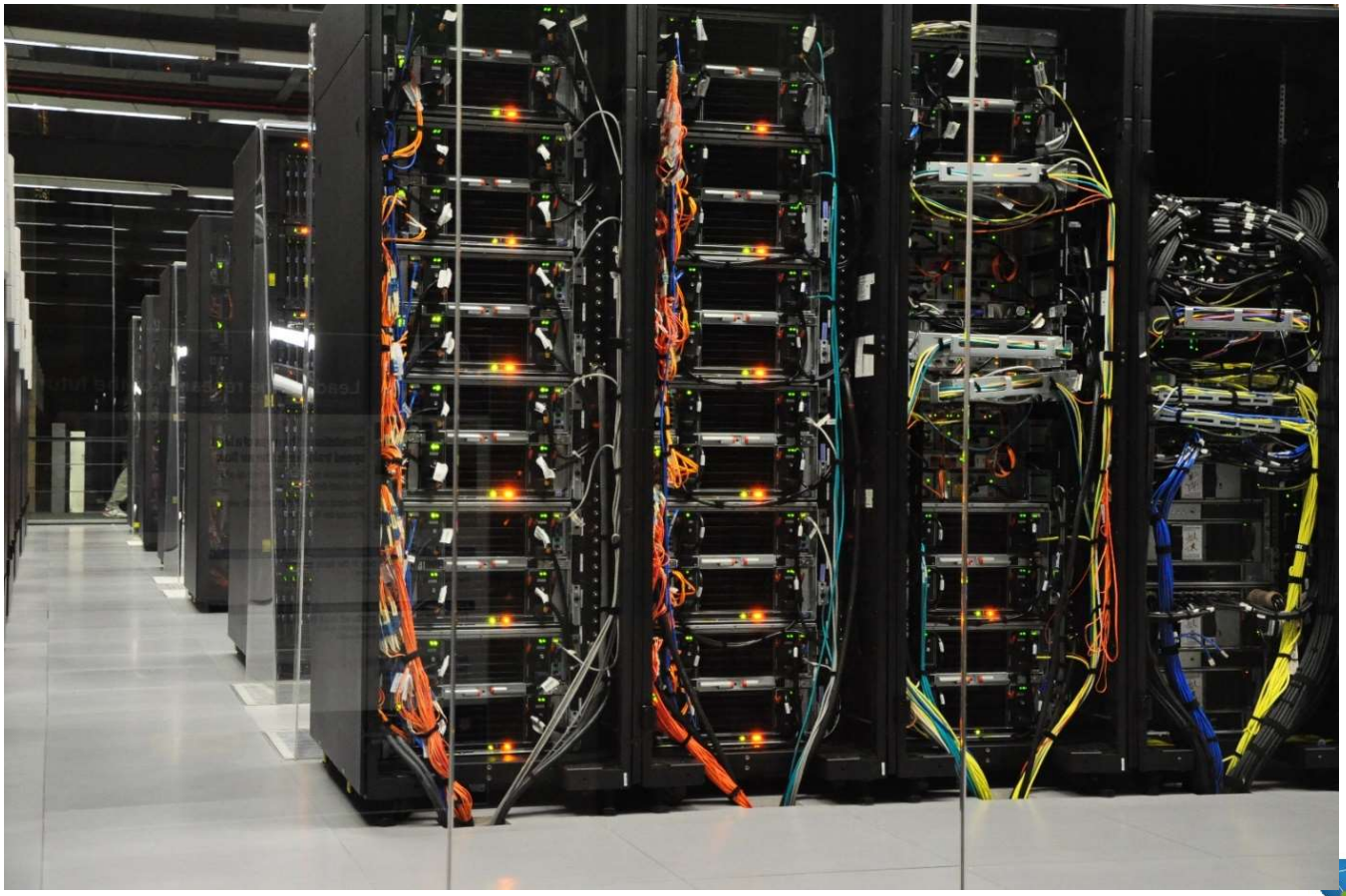
Computers Of All Shapes



Computers Of All Shapes



Computers Of All Shapes



What Do Computers Do?

What Do Computers Do?

- Computers process data, transforming it from one form to another
  - Data of some form is obtained from
    - Input devices (mouse and keyboard)
    - Other computers and devices
  - The data is then processed by the computer
    - And presented (or stored) in the processed form

What Do Computers Do?

- Computers process data, transforming it from one form to another
  - Data of some form is obtained from
    - Input devices (mouse and keyboard)
    - Other computers and devices
  - The data is then processed by the computer
    - And presented (or stored) in the processed form

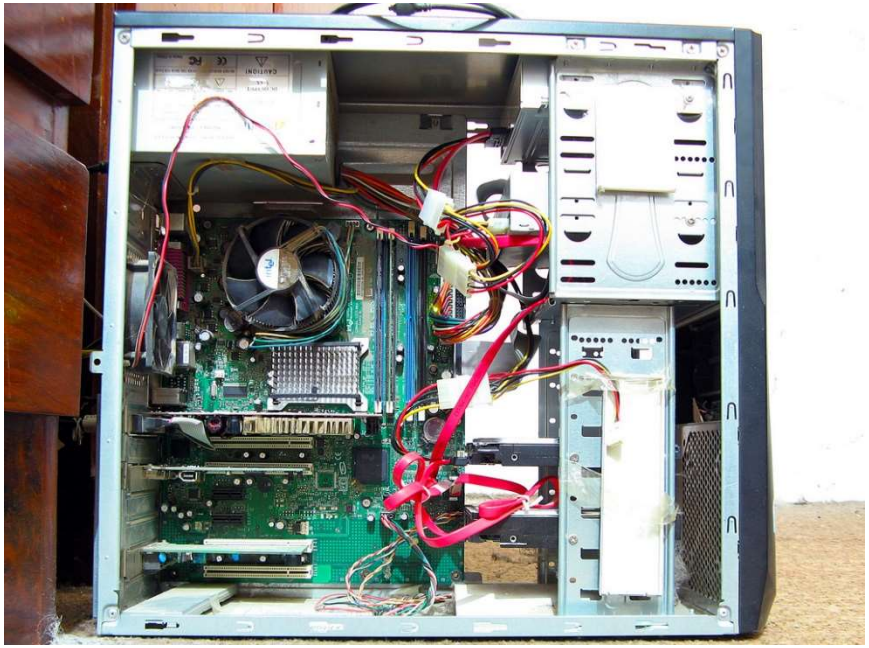
Scenario	Input data	Output data
Open a web page	Website URL	Webpage
Play a video game	Keystrokes	Graphics
Sequence assembly	Reads from sequencer	Assembled genome

Components Of Manufacturing Plants

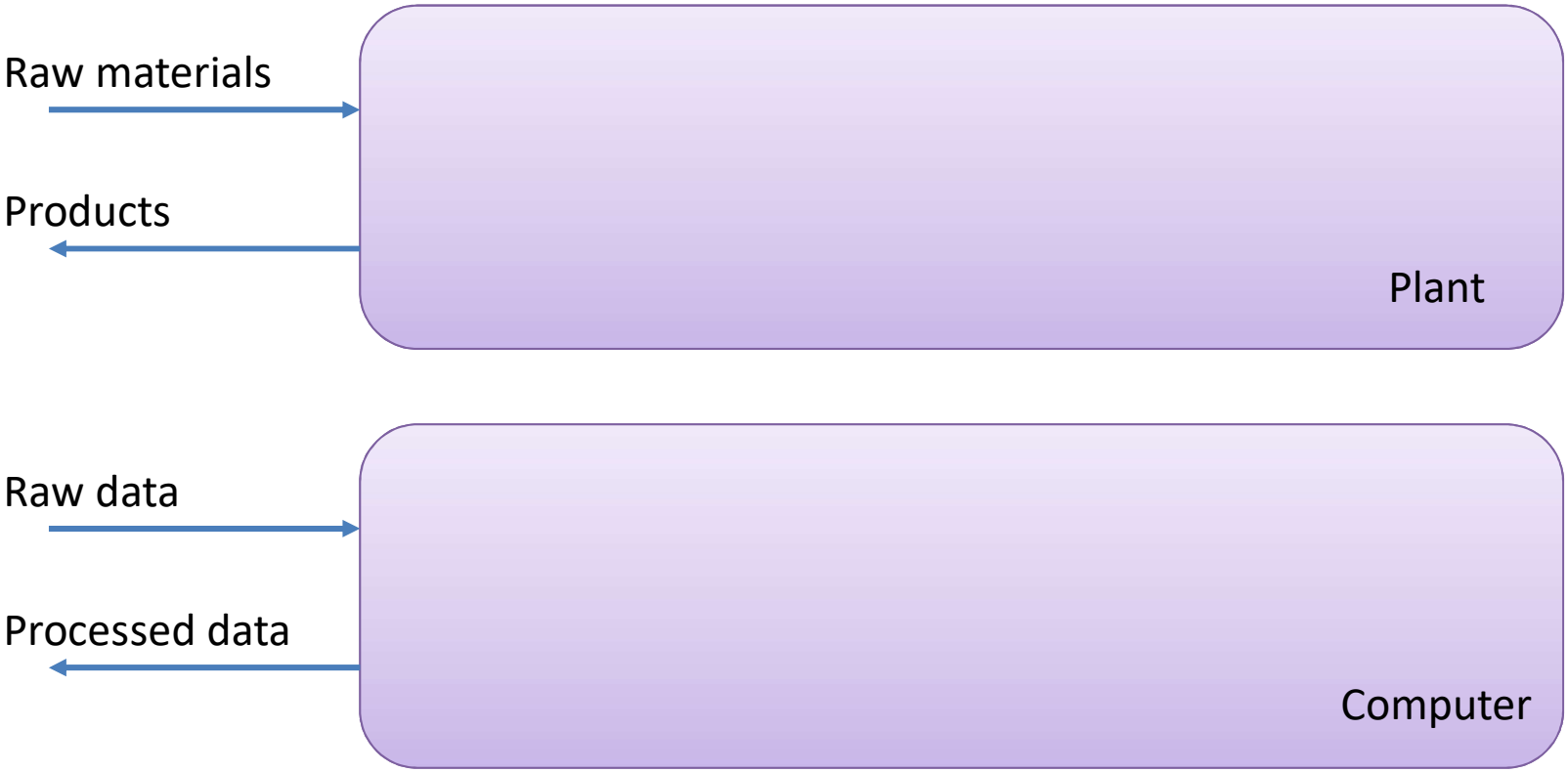
- Production workshops
  - Product assembly lines
  - Staging areas
- Warehouses
- Passages

Components Of Computers

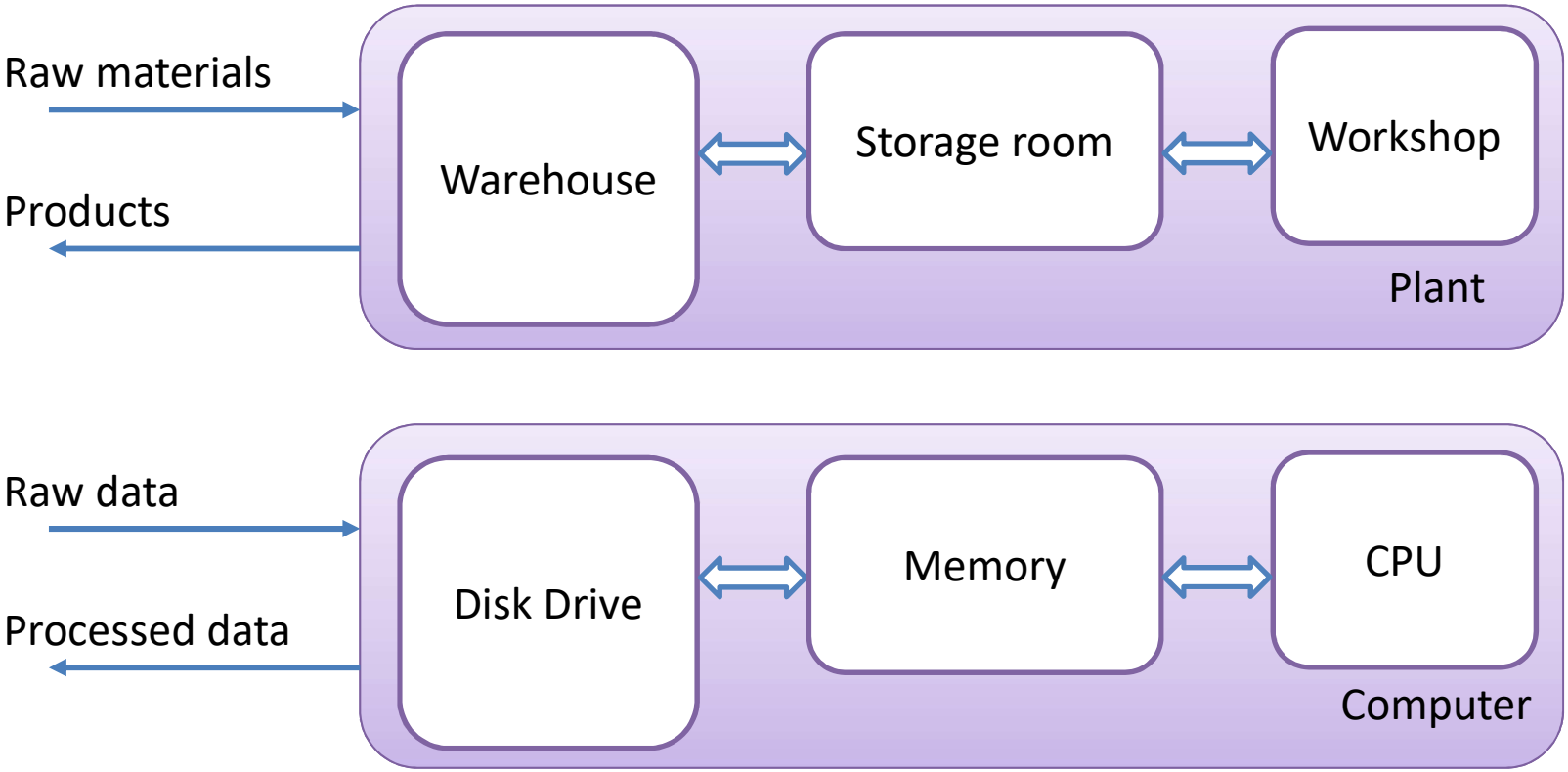
- Processors (CPU)
- Storage units
- Data bus



Computers vs. Manufacturing Plants



Computers vs. Manufacturing Plants



Computers vs. Manufacturing Plants

Manufacturing plants	Computers
Raw material	Data
Product	Data
Workshop	CPU
Storage space (warehouses, storage rooms, staging areas etc.)	Storage systems (hard drive, memory, cache etc.)
Passage/pathway	Data bus

Computers vs. Manufacturing Plants

Manufacturing plants	Computers
Raw material	Data
Product	Data
Workshop	CPU
Storage space (warehouses, storage rooms, staging areas etc.)	Storage systems (hard drive, memory, cache etc.)
Passage/pathway	Data bus

Anything else?

Computers vs. Manufacturing Plants

Manufacturing plants	Computers
Raw material	Data
Product	Data
Workshop	CPU
Storage space (warehouses, storage rooms, staging areas etc.)	Storage systems (hard drive, memory, cache etc.)
Passage/pathway	Data bus
Operation manual/plan	Software (another form of data)

Data – Know Your Raw Materials

- All information in a computing system is represented by bits
- Each **bit** is either 0 or 1
  - Denoted by lower case “b”
- They are arranged into 8-bit clunks, each of which is called a **byte**
  - Denoted by upper case “B”

The Meaning Of 0's And 1's

- The meaning of a sequence of 0's and 1's depends on context, i.e. how it is interpreted.
- Example: 01110101 (a sequence of 8 bits, or 1 byte)
  - Numeric (base-10 integer): 117
  - Text: lower case "u"
  - Color (8-bit grayscale): 
  - Could be a part of
    - 4-byte long integer
    - 4-byte long real number
    - 4-byte long color
    - …

Types Of Data

- Textual
  - Human readable
  - ASCII (American Standard Code for Information Interchange): a map between characters and one-byte integers.
- Binary
  - Machine readable

The ASCII code

American Standard Code for Information Interchange

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Textual vs Binary: Integers

1,333,219,569

Textual Representation

10 bytes (characters) or 13 bytes (incl. the commas)

Binary Representation

4 bytes (32 bits)

Textual vs Binary: Programs

```
print "Hello, world!"
```

- Computers do NOT understand this statement in its original form
- It needs to be translated to a binary form before being executed on a computer

Programming Languages

- Programming languages can be categorized as:
  - Compiled languages
    - Human readable programs have to be “compiled” into a binary executable by special software called “compiler” before they can be executed by computers
    - Examples are C/C++, Fortran etc.
  - Interpreted languages
    - Human readable programs can be executed “directly” using an “interpreter”
    - Examples are Python, Perl, R, Ruby etc.
  - Not a well-defined divide because, either way, computers are unable to execute human-readable instructions without some sort of human-to-machine translation

Know Your Numbers (1)

Prefix	Quantity
K (kilo)	1,000
M (mega)	1,000,000
G (giga)	1,000,000,000
T (tera)	1,000,000,000,000
P (peta)	1,000,000,000,000,000
E (exa)	1,000,000,000,000,000,000

Measuring Data: Know Your Numbers (2)

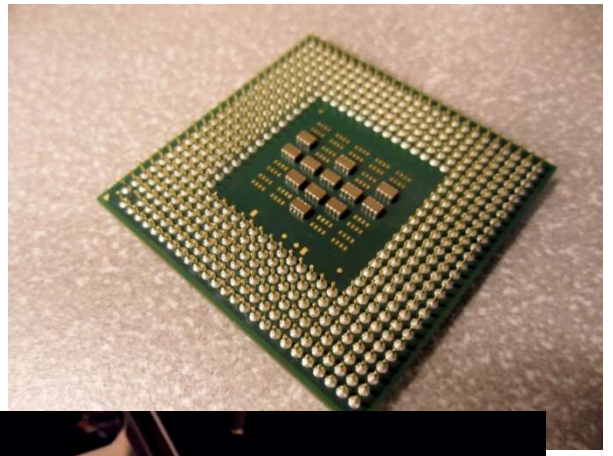
- Relevant numbers
  - Capability, e.g. CPU (GHz)
  - Capacity, e.g. memory (~GB) and hard drive (~TB)
  - Bandwidth, e.g. network (~Gb)
  - Data size, e.g. reference genomes (~GB)

Measuring Data: Know Your Numbers (3)

- Knowing these numbers helps put things into perspective
  - How fast can data can be processed by the CPU?
  - How long does it take to transfer data from hard drive to memory? From remote server over network?

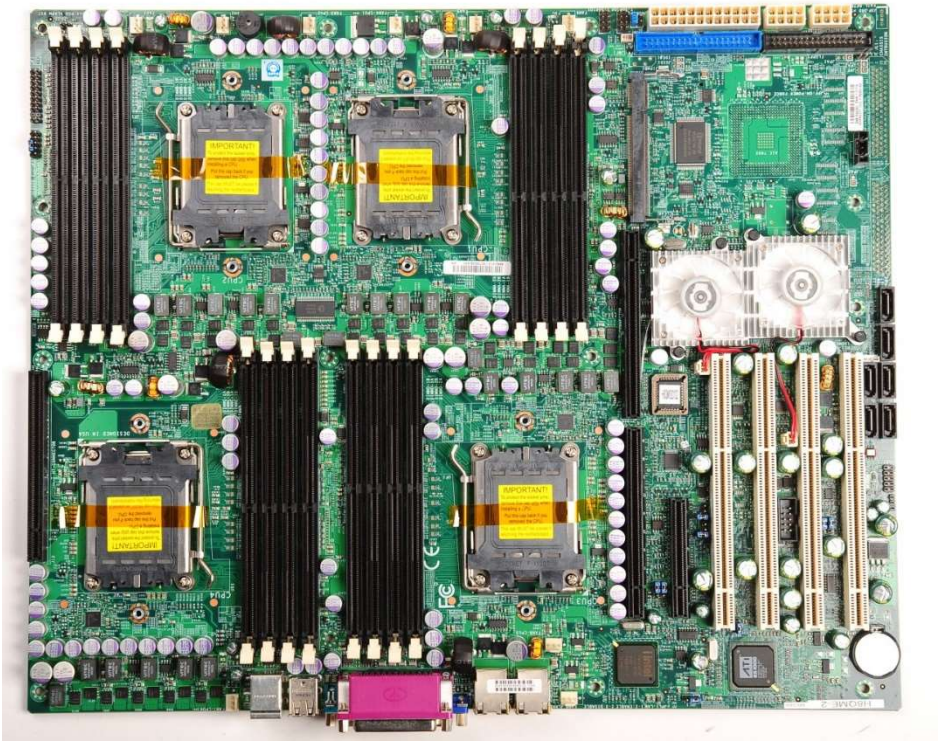
Central Processing Unit (CPU) (1)

- Or simply “processor”
  - A “chip” plugged into a “socket” on the motherboard
- This is where the numbers are crunched

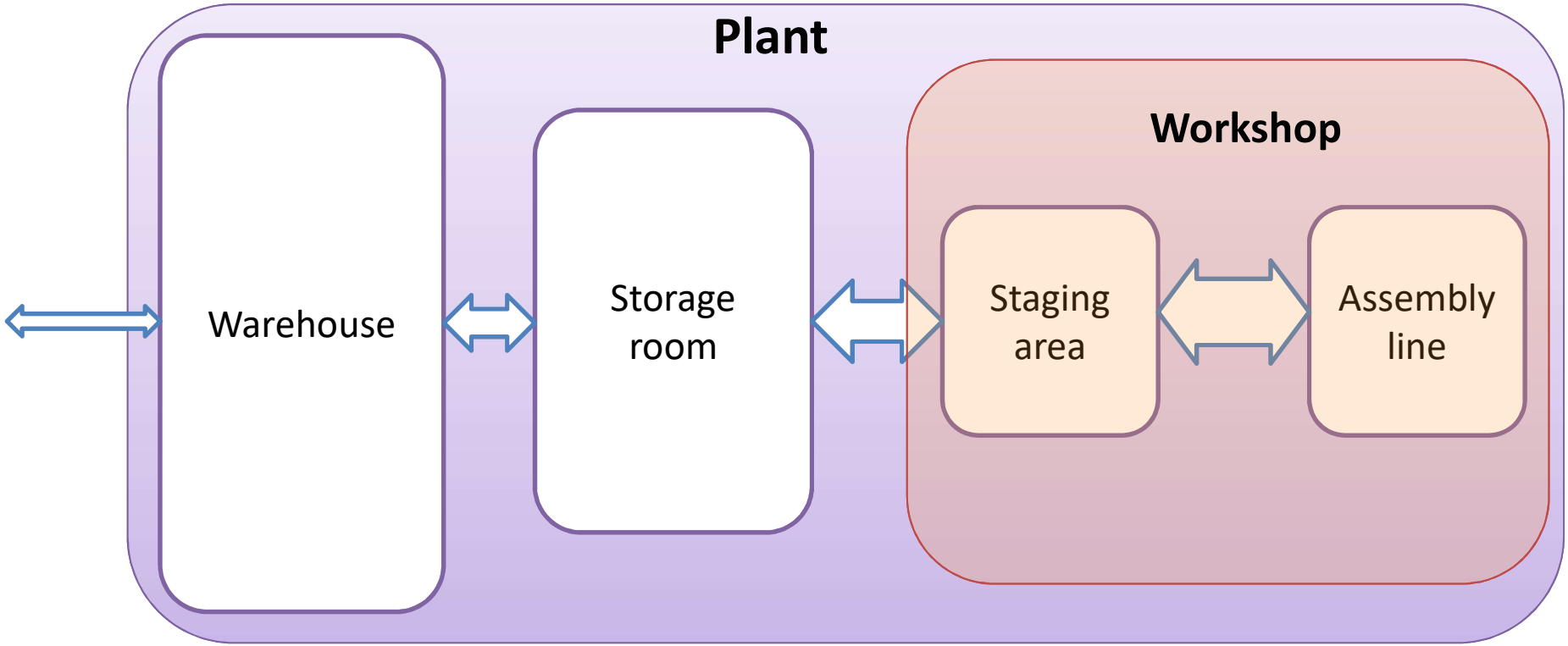


Central Processing Unit (CPU) (2)

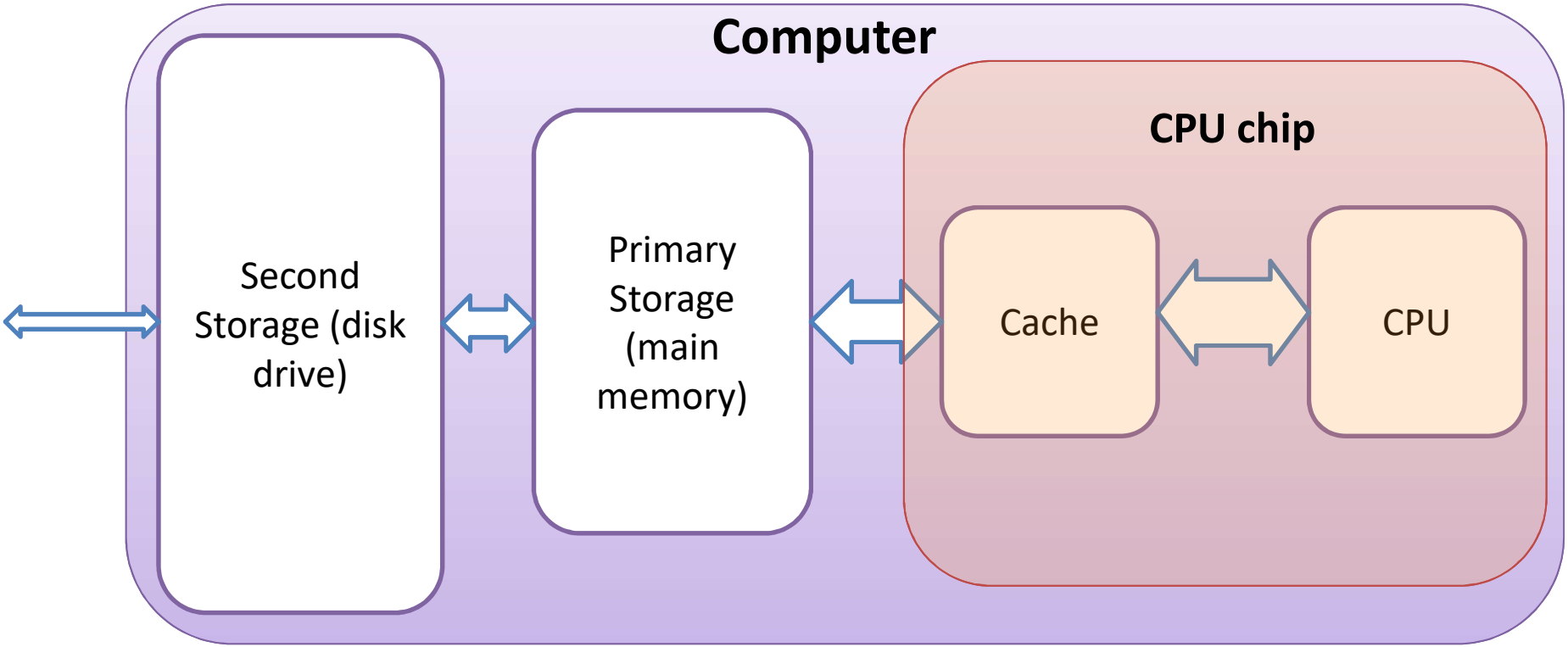
- A computer can have more than one CPU
- How many CPU sockets are shown in this picture?



A Closer Look: Workshop



A Closer Look: CPU



How CPUs Work

- CPU's function by executing instructions
  - In each clock cycle, the CPU fetch an instruction from main memory and execute it
  - The instructions can be load data, save data, operate on data etc.
- A CPU's speed is measured by how many clock cycles it has per second, aka "frequency"
  - Ex: 2.6 GHz = 2.6 x 10^9 cycles per second

To calculate A=B*C:
Step 1: load value of B to register
Step 2: load value of C to register
Step 3: Calculate B*C
Step 4: save value of A

Register And Cache

- Register: a very small storage device in CPU
  - For data and instructions that are needed immediately
- Cache: memory on CPU chip
  - A storage between CPU and main memory that can read and write data with higher throughput than the main memory
- More on them later

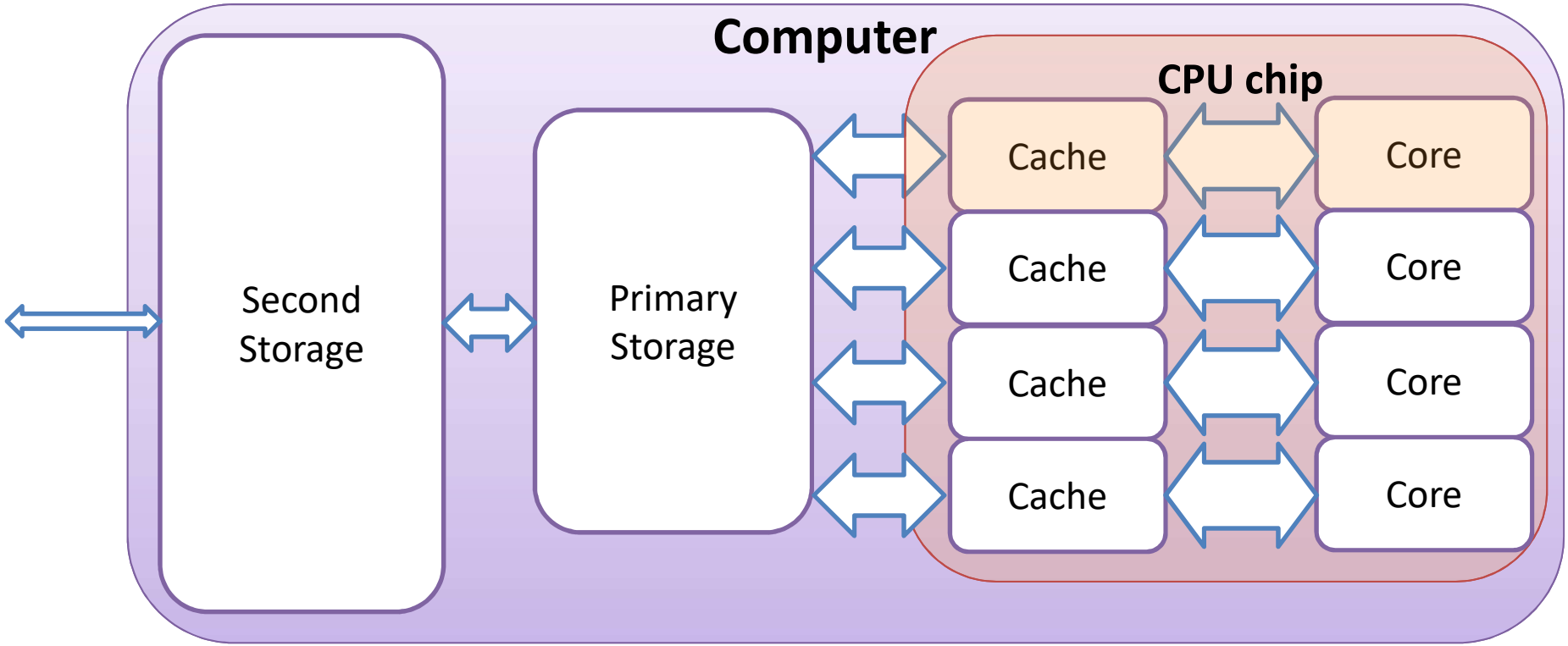
Concurrency

- Concurrency is a very important concept in modern computing systems, which greatly improves their performance
- There are multiple types/levels of concurrency
  - Core level: each chip may have multiple CPU's (cores)
    - Current generation of Intel CPU's (Skylake family) could have anywhere between 2 to 28 cores
  - Instruction level: execute more than one instruction per clock cycle by overlapping stages of execution
  - Vectorization: perform multiple operations with one instruction

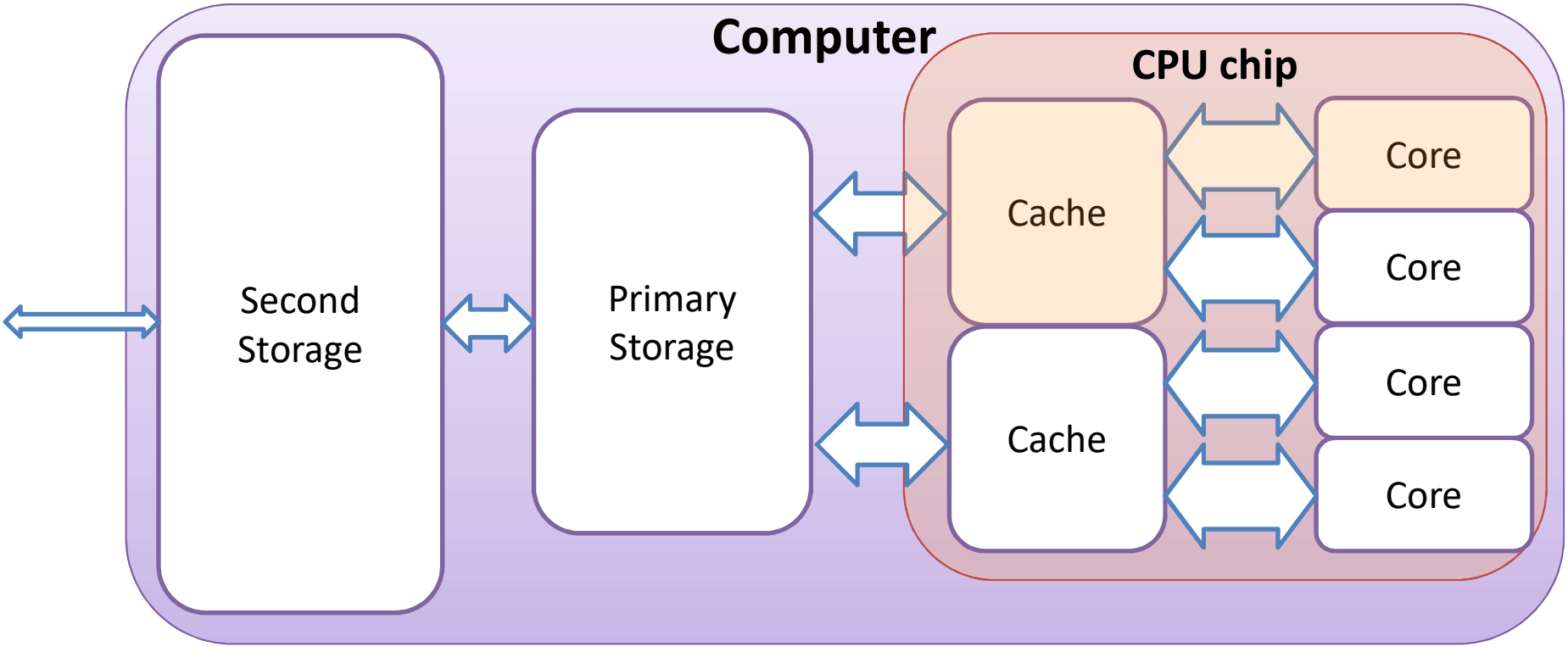
Concurrency

- Concurrency is a very important concept in modern computing systems, which greatly improves their performance
- There are multiple types/levels of concurrency
  - Core level: each ch **Multiple assembly lines**
    - Current generation of Intel CPUs (Skylake family) could have anywhere between 2 to 28 cores
  - Instruction level: e than one instruction per clock cycle by **Pipelining** stages of execution
  - Vectorization: perf **Multiple parts on each assembly lin** instruction

Different Design With MultiCores (1)



Different Design With MultiCores (2)



Measuring CPU Speed

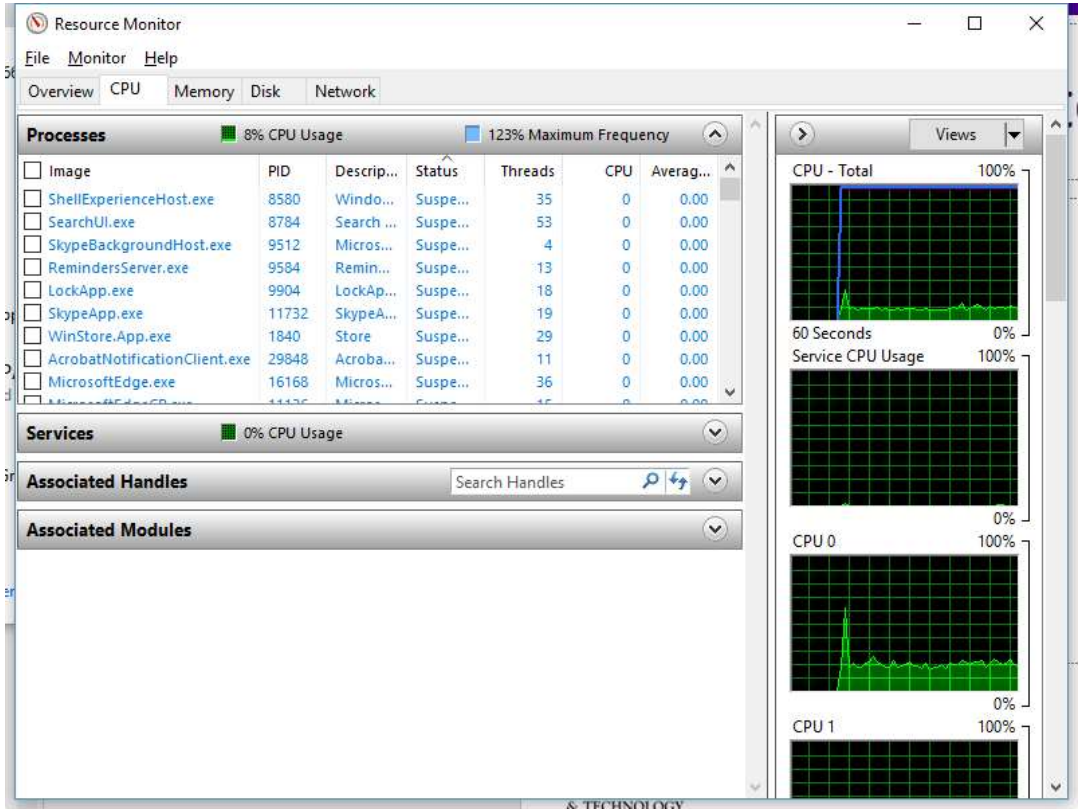
- FLOPS = **FL**oating **O**perations **P**er **S**econd
- With the concurrencies we discussed, the computing power of a CPU is:

  (Frequency) x (Number of operations per cycle) x (Number of cores)

- Ex: Intel Skylake 6148 CPU @ 2.40 GHz

  (2.4×10^9 Hz) x (16 operations/instruction) x (20 cores) = 768 GFLOPS

Monitoring CPU Usage



Tips for practitioners

Monitor CPU load and make sure that they are busy when processing your workload.

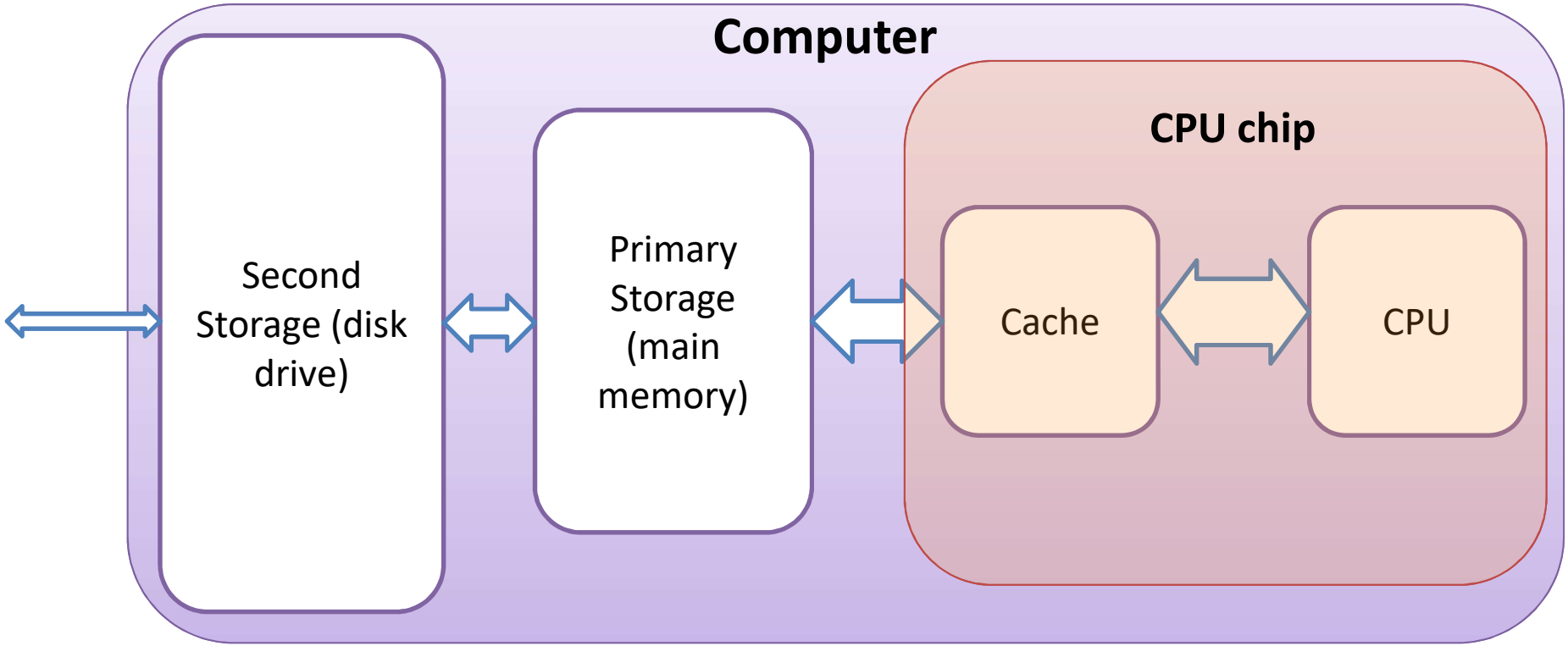
Monitoring CPU Usage

```
top - 16:42:59 up 46 days,  7:53,  0 users,  load average: 0.20, 0.10, 2.71
Tasks: 671 total,   1 running, 670 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 98.0%us,  1.7%sy,  0.0%ni,  0.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.3%us,  0.3%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,
Cpu5  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa
Cpu7  :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa
Cpu8  :  0.0%us,  0.7%sy,  0.0%ni, 99.3%id,  0.0%wa
Cpu9  : 98.7%us,  1.3%sy,  0.0%ni,  0.0%id,  0.0%wa
Cpu10 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa
Cpu11 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa
Cpu12 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,
Cpu13 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  32815016k total,  7513104k used, 25301912k free,    85308k buffers
Swap: 100663292k total,    23392k used, 100639900k free,  5412200k cached
```

Tips for practitioners

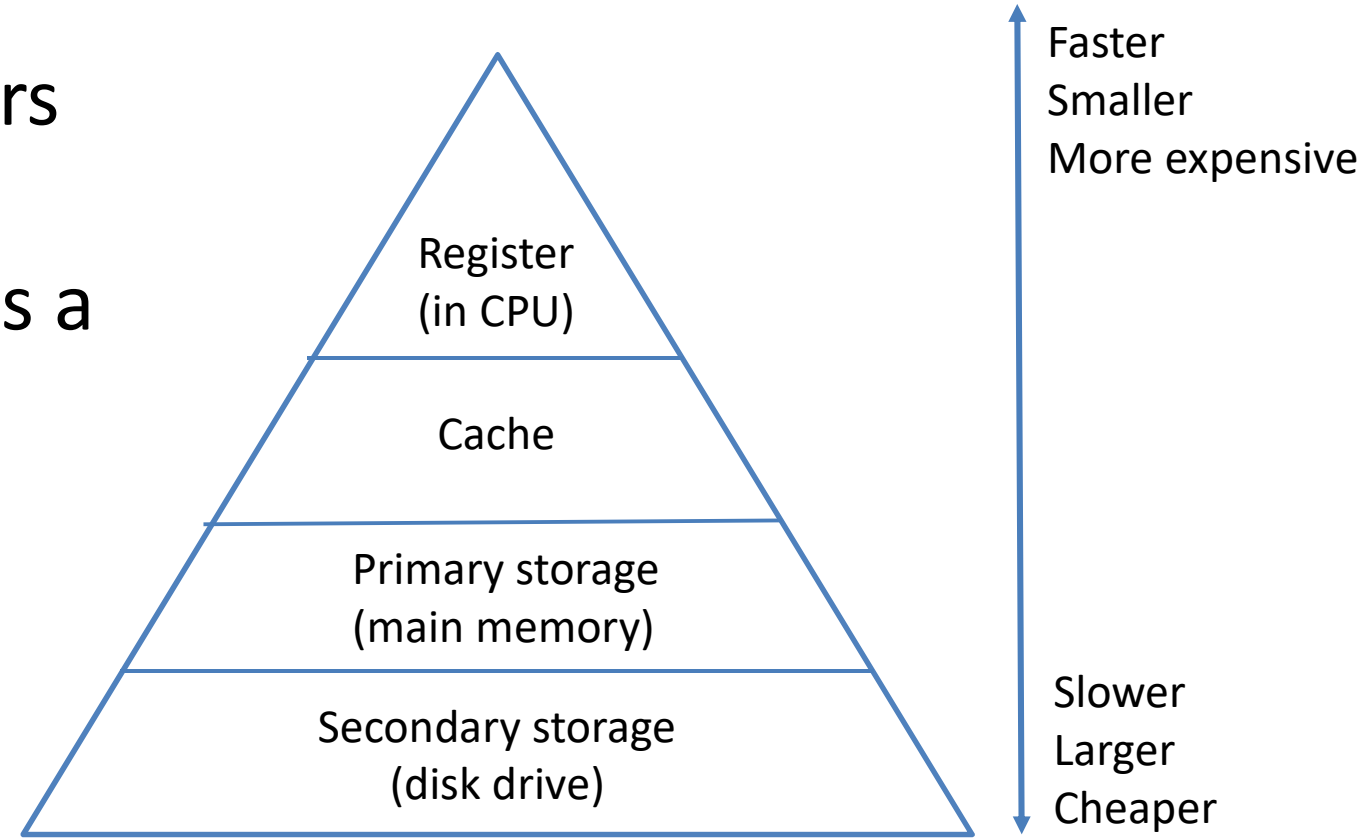
Monitor CPU load and make sure that they are busy when processing your workload.

Storage Hierarchy



Storage Hierarchy

- In computers storage is organized as a hierarchy

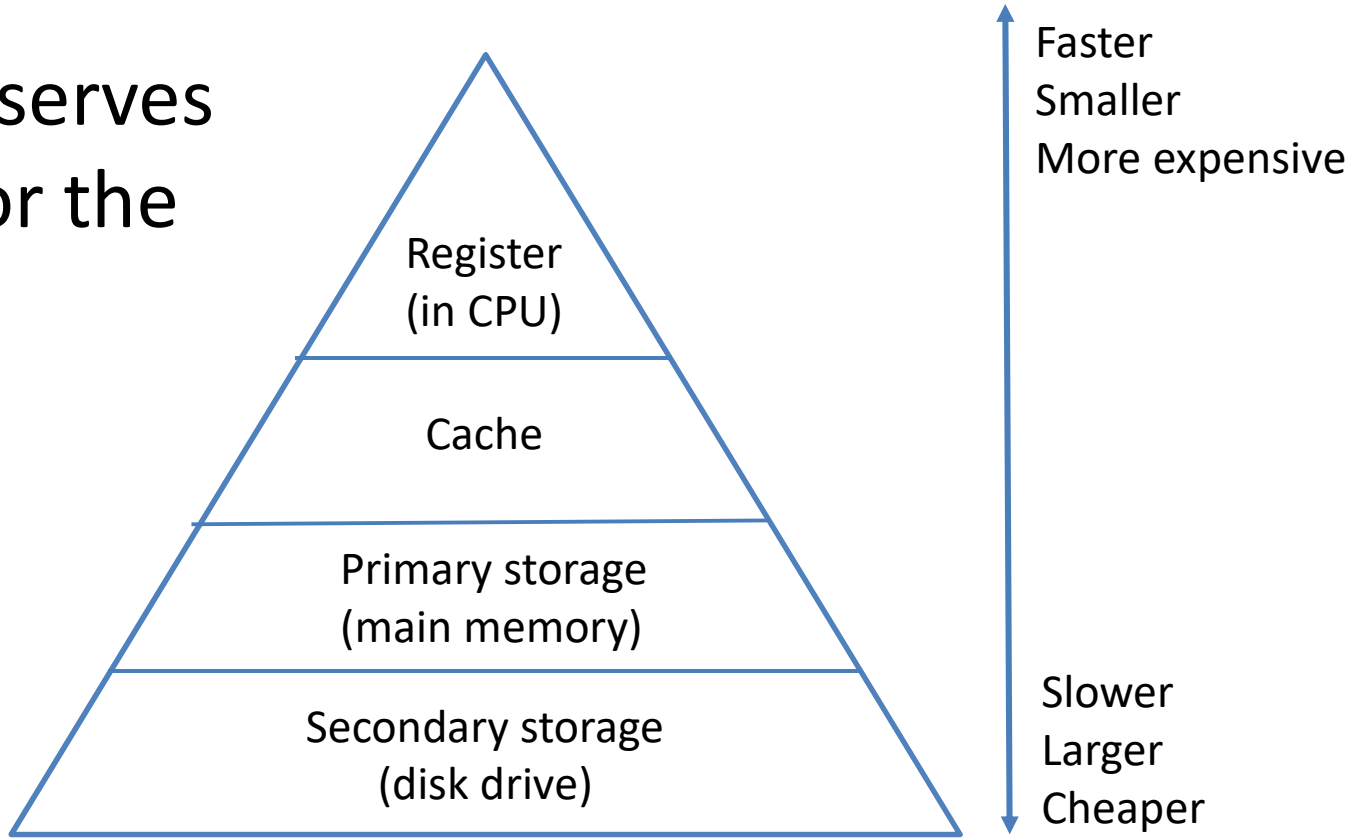


Why Use Cache?

- The problem of processor-memory gap – CPU's are capable of processing data at a much higher speed than the main memory can supply
  - If we opt to stick with slow memory, CPU's will always be almost hungry
  - On the other hand, ultra-fast memory that can keep up with CPU is very expensive to build
- Solution: insert cache between CPU and main memory
  - Cache serves as temporary staging area for data that the processor is likely to need in the near future.
  - We can have both a large memory and a fast one
    - Need to reuse data loaded into the fast memory as much as possible

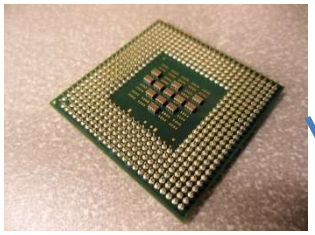
Storage Hierarchy

- Each level serves as cache for the next level



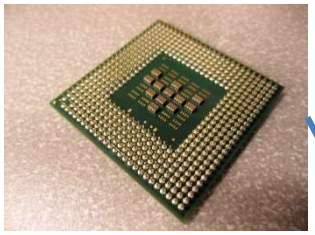
Storage Hierarchy: Devices

Cache: **S**tatic **R**andom **A**ccess **M**emory (SRAM)

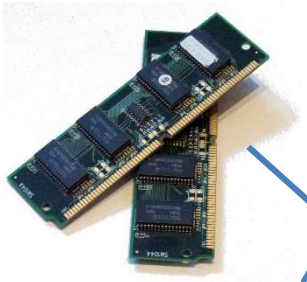


Storage Hierarchy: Devices

Cache: **S**tatic **R**andom **A**ccess **M**emory (SRAM)

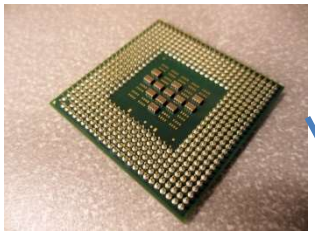


Main memory: **D**ynamic **R**andom **A**ccess **M**emory (DRAM)

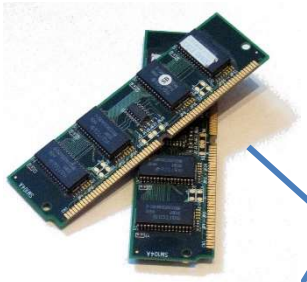


Storage Hierarchy: Devices

Cache: **S**tatic **R**andom **A**ccess **M**emory (SRAM)



Main memory: **D**ynamic **R**andom **A**ccess **M**emory (DRAM)



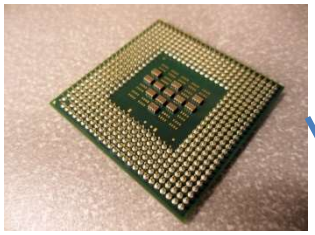
Spinning Drive

Faster
Smaller
More expensive

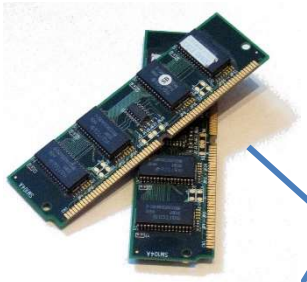
Slower
Larger
Cheaper

Storage Hierarchy: Devices

Cache: **S**tatic **R**andom **A**ccess **M**emory (SRAM)



Main memory: **D**ynamic **R**andom **A**ccess **M**emory (DRAM)



Spinning Drive

Faster
Smaller
More expensive

Slower
Larger
Cheaper

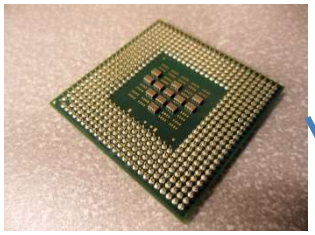
Solid State Drive (SSD)

LSU CENTER FOR COMPUTATION & TECHNOLOGY

LONI

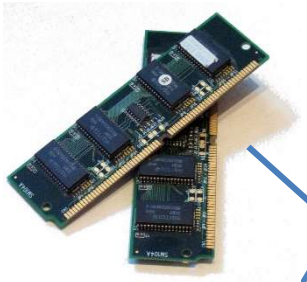
Storage Hierarchy: Devices

Cache: **S**tatic **R**andom **A**ccess **M**emory (SRAM)



Volatile

Main memory: **D**ynamic **R**andom **A**ccess **M**emory (DRAM)



Spinning Drive

Non-volatile

Solid State Drive (SSD)

Faster
Smaller
More expensive

Slower
Larger
Cheaper

LSU CENTER FOR COMPUTATION & TECHNOLOGY

LONI

Measuring Data Movement

- Latency: how long it takes to start and end a transfer
- Bandwidth: how fast data can be moved once a transfer is started

$$Transfer\ time = latency + \frac{Amount\ of\ data}{bandwidth}$$

Storage Hierarchy: Performance and Capacity

Level	Performance		Capacity
	Latency (cycles)	Bandwidth (per second)	
Register	0		O(KB)
Cache	O(1) – O(10)	O(10GB)-O(100GB)	O(100KB) – O(10MB)
Main memory	O(100)	O(1GB)	O(10GB)
Spinning Drive	O(10,000,000)	O(10MB)	O(1TB)
SSD	O(100,000)	O(1GB)	O(1TB)

Storage Hierarchy: Performance and Capacity

Level	Performance		Capacity
	Latency (cycles)	Bandwidth (per second)	
Register	0		O(KB)
Cache	O(1) – O(10)		
Main memory	O(100)		
Spinning Drive	O(10,000,000)		
SSD	O(100,000)	O(1GB)	O(1TB)

Tips for practitioners

- Be aware of the limitation of storage systems, e.g. the amount of available memory
- Eliminate unnecessary data movement

Operating Systems Manage Hardware

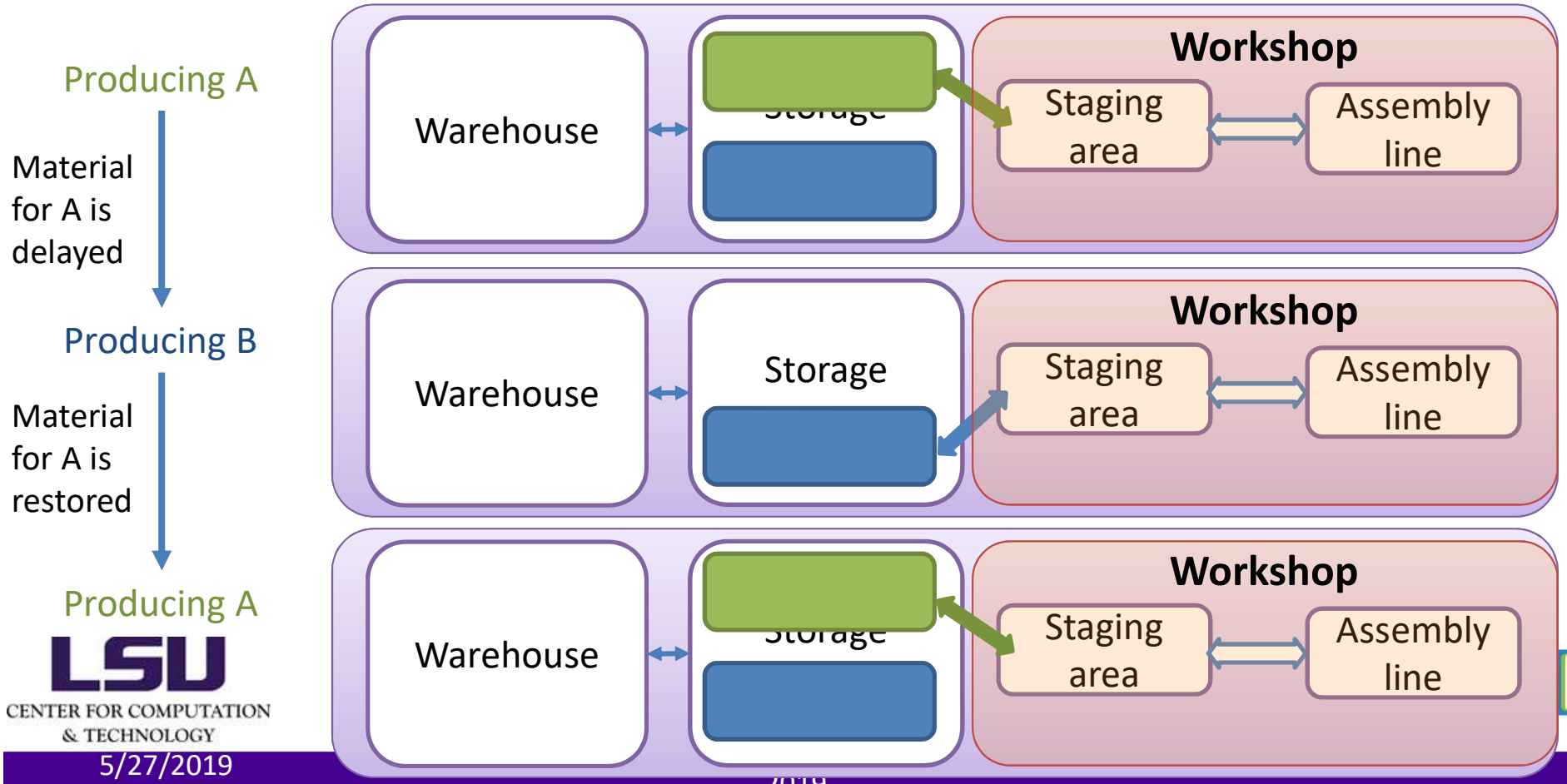
- Operating systems provide applications **with simple and uniform mechanisms** for manipulating hardware
  - So that applications themselves do not have to deal with the complicated and different hardware devices.
  - Ex: when writing a sequence of bytes to an output device, a program may perceive no difference no matter the device is spinning drive, SSD or even screen display

Operating Systems Manage Program Execution

- Operating systems use the concept of processes to manage running programs
- A **process** is an instance of a program in execution
- The OS provides an abstraction so that
  - It appears to every program that it is the only one running on the system, while
  - The OS manages multiple programs running concurrently on the same system and tries to maximize resource utilization

Why Use Processes?

Imagine that a plant produces many different products, the procedure of each of which is different but shares the same workshop.



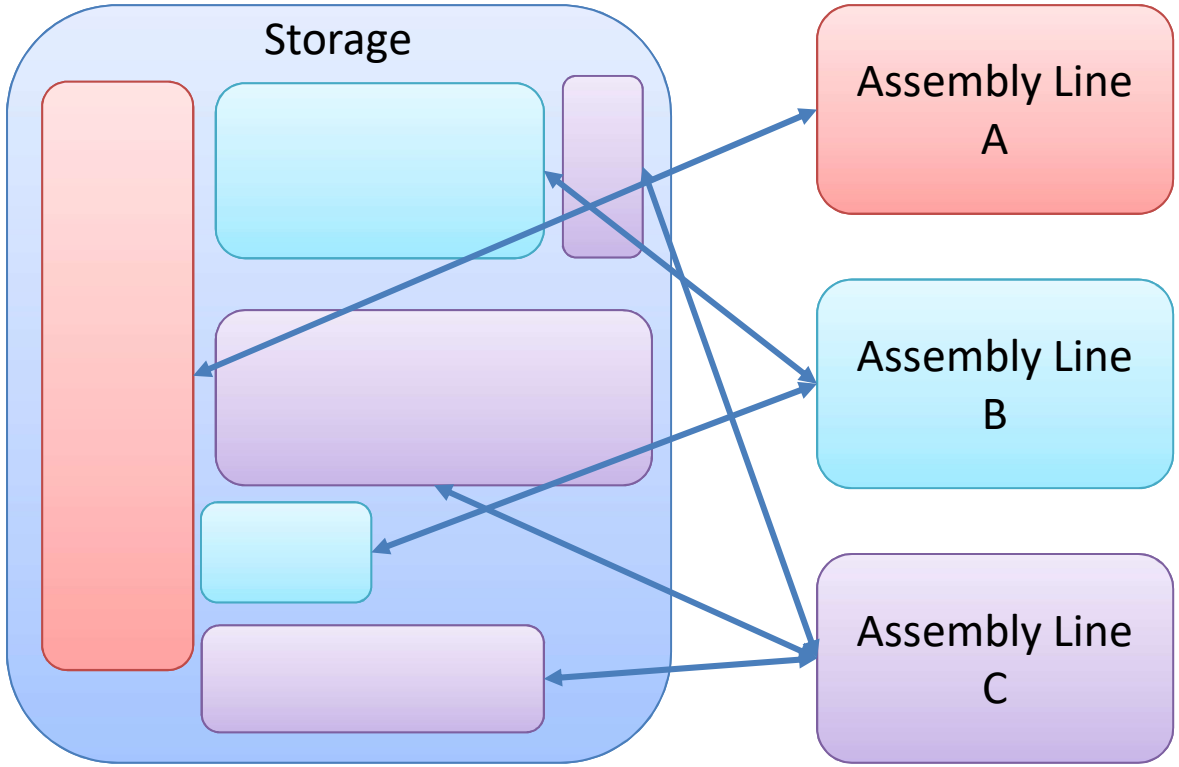
Thread

- A “light weight” instance of a running program
- Compared to processes, threads are less flexible, but it takes less efforts (faster) to switch between threads
  - Tradeoff: performance vs flexibility

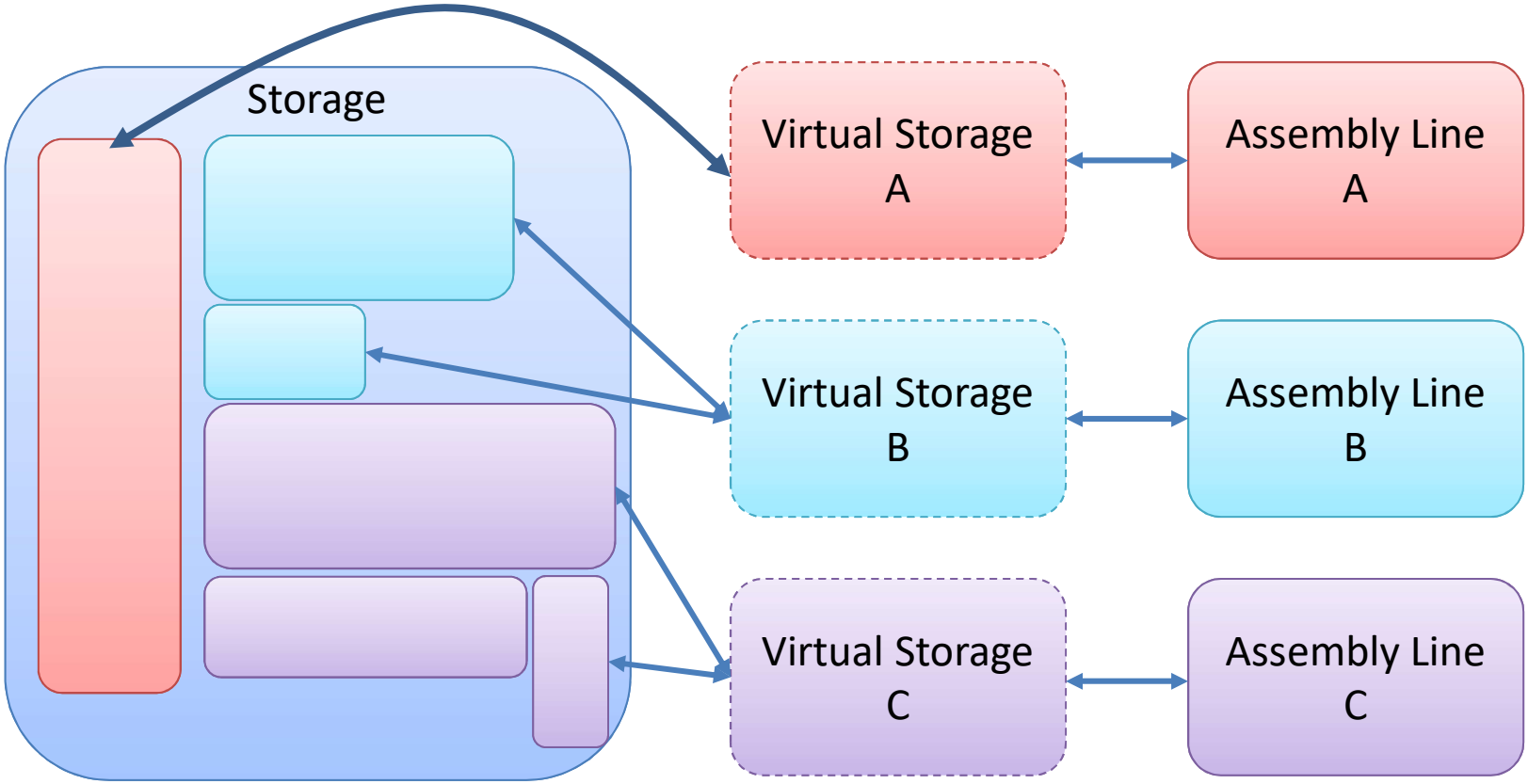
OS Memory Management: Virtual Memory (1)

- The abstraction used by the OS to manage memory
- Virtual memory gives all programs an illusion that they have exclusive use of memory
- Then OS maps virtual memory to physical memory

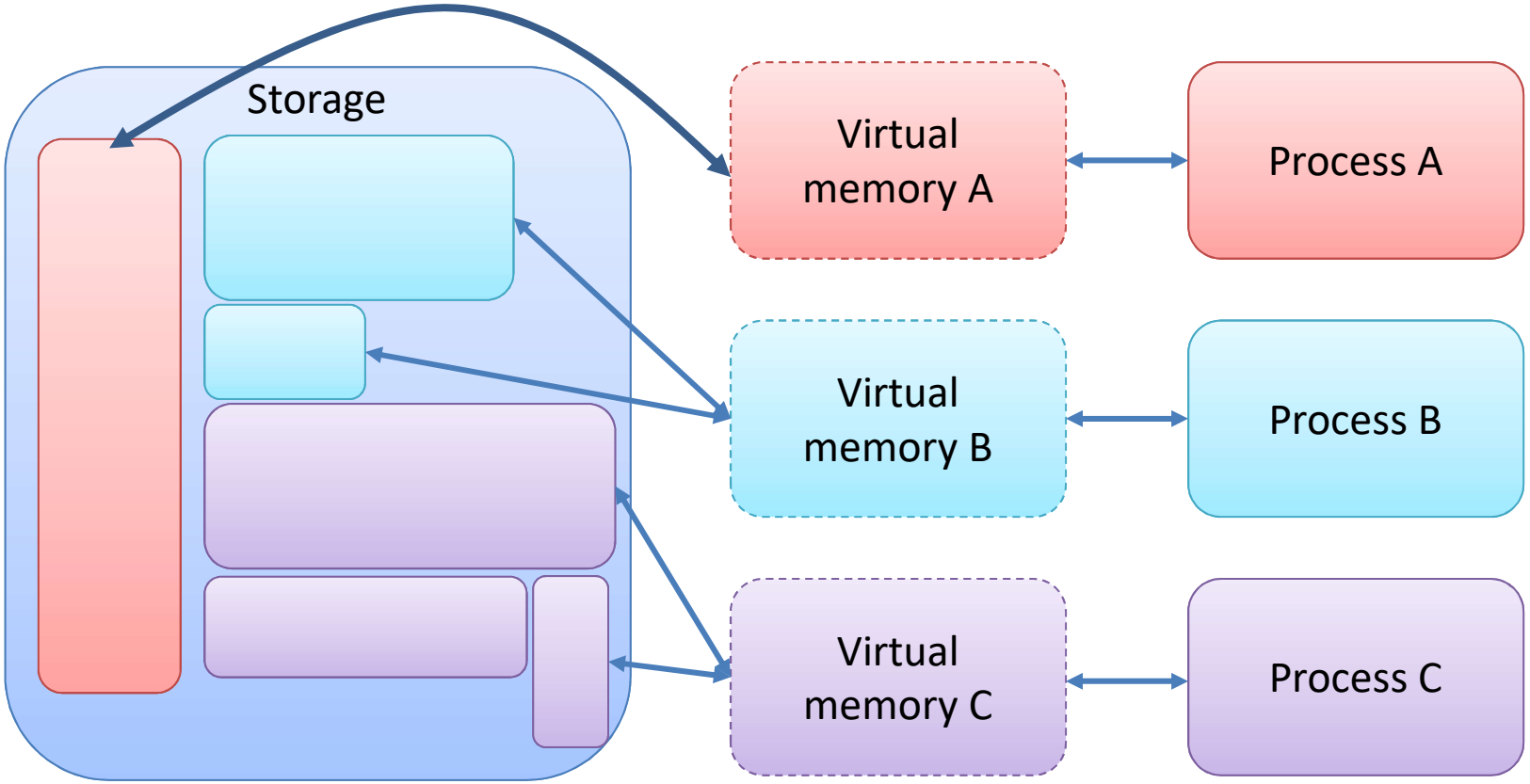
Without Virtual Memory



With Virtual Memory



With Virtual Memory



OS Memory Management: Swapping

- What if the computer is out of memory?

OS Memory Management: Swapping

- What if the computer is out of memory?
- It turns out that OS will try to move, or swap, some processes to secondary storage so others can run
  - Will be swapped back later
- Performance can be greatly impacted

OS Memory Management: Swapping

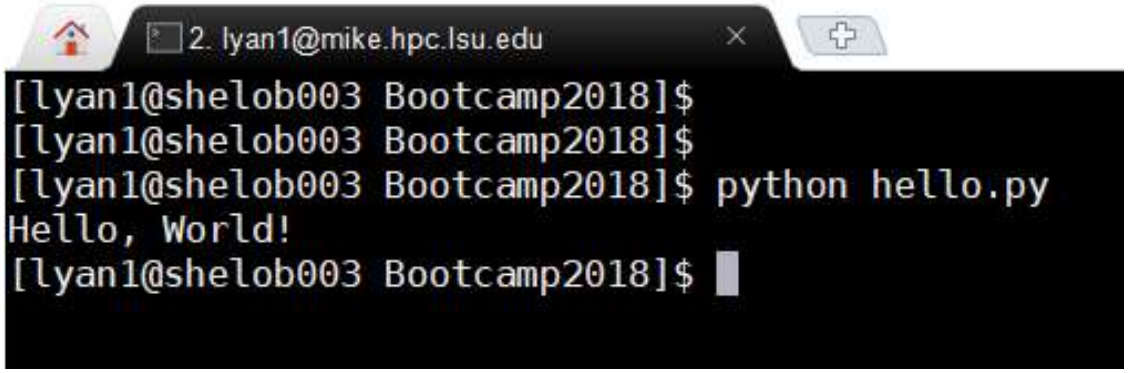
- What if the computer is out of memory?
- It turns out that OS will try to move, or swap, some processes to secondary storage so others can run
  - Will be swapped back later
- Performance can be great

Tips for practitioners

- Monitor how much memory your program is using.
- Be aware of how memory requirement changes with scenarios, e.g. input data size, model resolution.

Operating Systems Provide User Interface

- Graphic User Interface (GUI)
- Command Line Interface (CLI)
  - With some flavor of the Linux/Unix operating systems
  - Great tool for automation

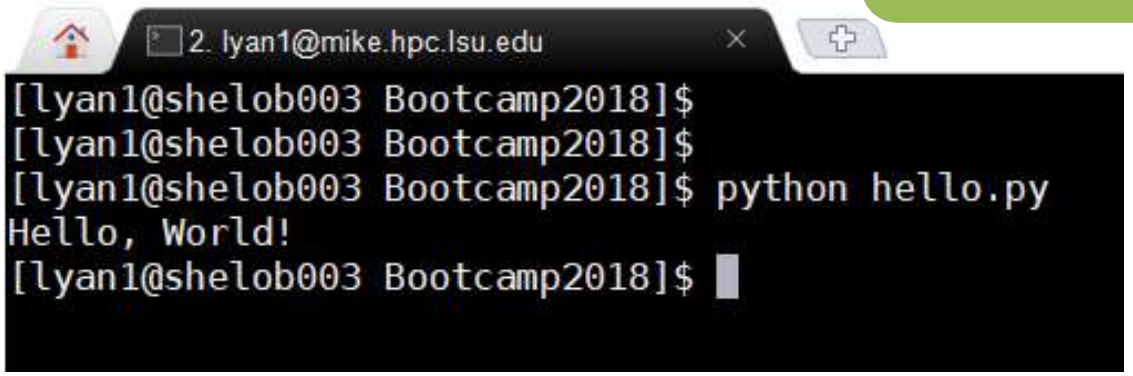


Operating Systems Provide User Interface

- Graphic User Interface (GUI)
- Command Line Interface (CLI)
  - With some flavor of the Linux/Unix operating systems
  - Great tool for automation

Tips for practitioners

Get familiar with CLI.



Files

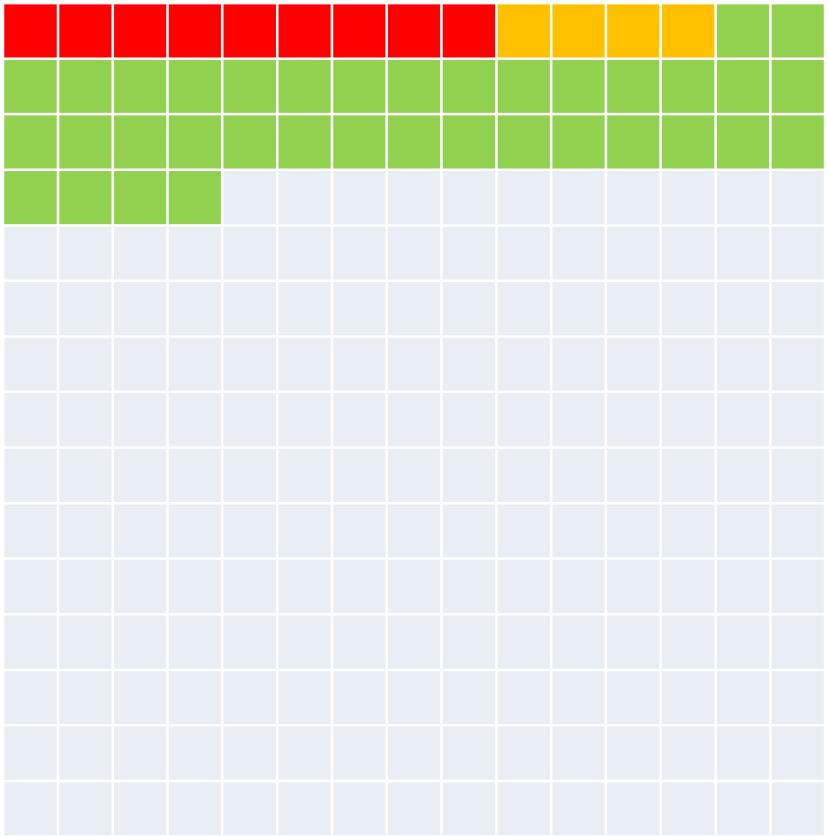
- A file is just a sequence of bytes
  - An abstraction used by OS to represent all I/O devices and to present a uniform view to applications
  - Allows the development of portable applications without knowing details of underlying technology
- From the OS point of view, files include
  - Normal files on disks
  - Keyboard
  - Display
  - All input/output devices
  - Even networks

How Files Are Stored On Disks

- Imagine your disk as a huge warehouse
  - Each grid can store one byte

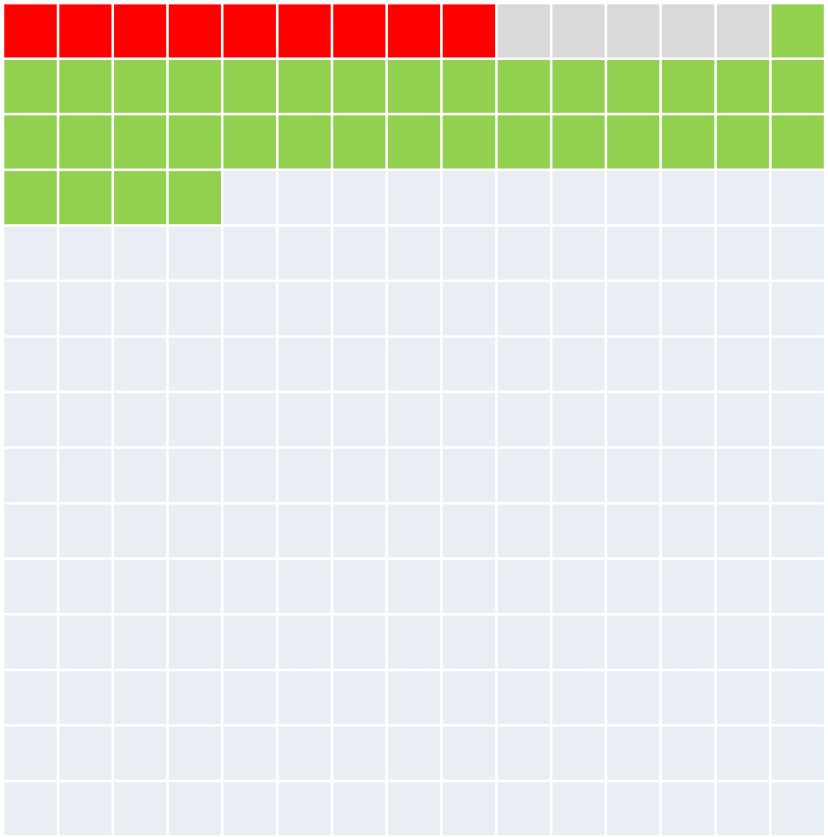
How Files Are Stored On Disks

- Now three files (i.e. sequences of bytes) are written to it



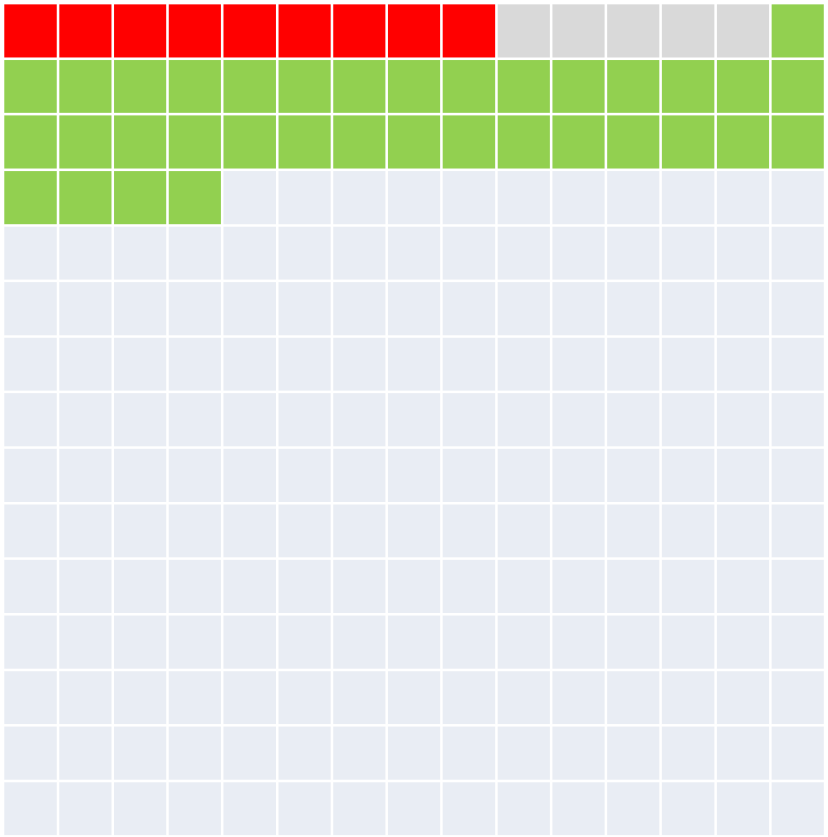
How Files Are Stored On Disks

- Then the yellow file was removed



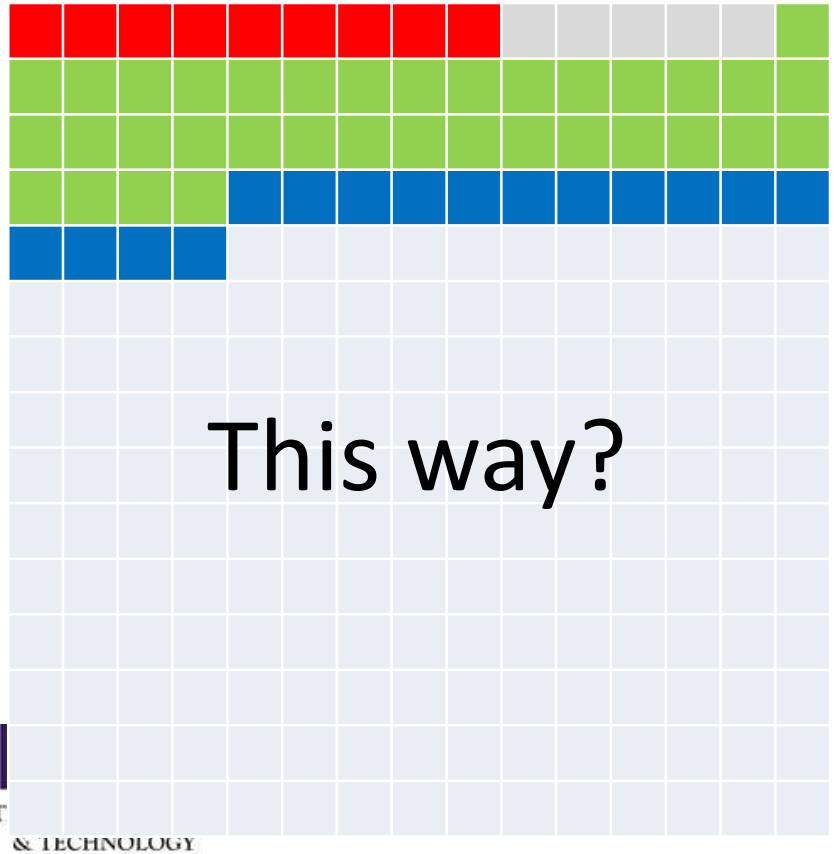
How Files Are Stored On Disks

- Now, what would happen if another sequence of 15 bytes need to be written to the disk?



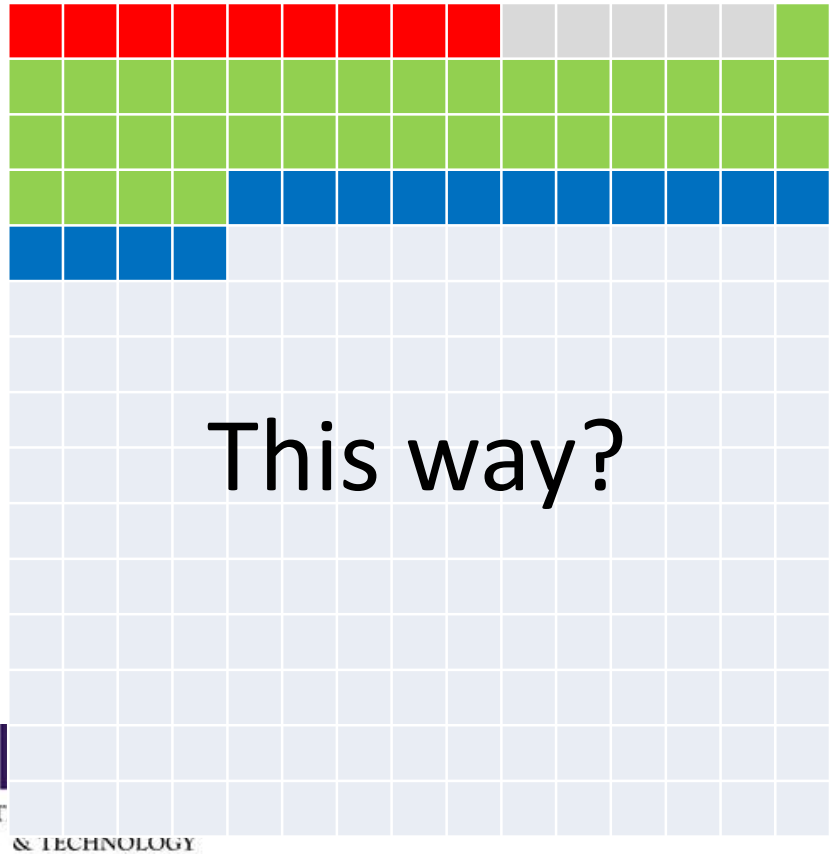
How Files Are Stored On Disks

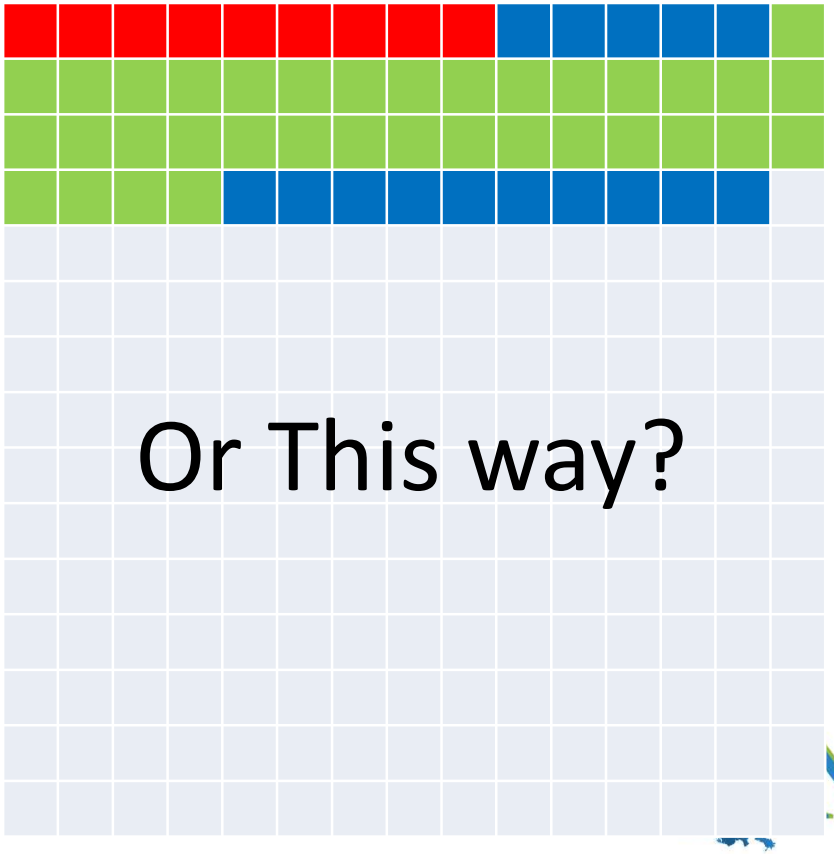
- Now, what would happen if another sequence of 15 bytes need to be written to the disk?



How Files Are Stores on Disks

- Now, what would happen if another sequence of 15 bytes need to be written to the disk?





File Systems

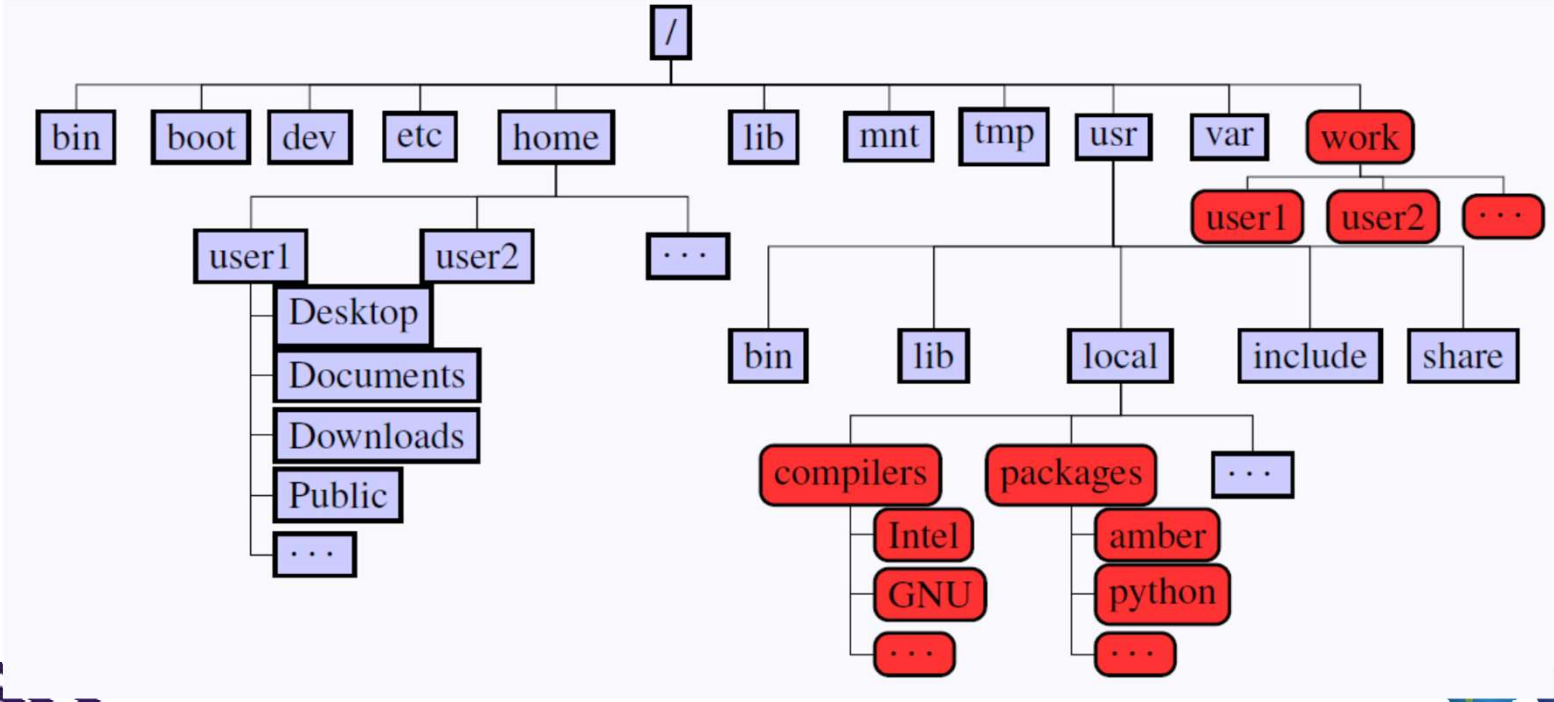
- File systems do bookkeeping for files on disks
  - Location, size etc.
- Details might be hidden from users
  - E.g. the bytes in a file might not be contiguous on the storage device
  - E.g. when a file is deleted, the fil
    records and the space it occupie
    state
- Performance consideration: boo
  and resources
  - E.g. not a good idea to put 1 million small files in the same directory.

Tips for practitioners

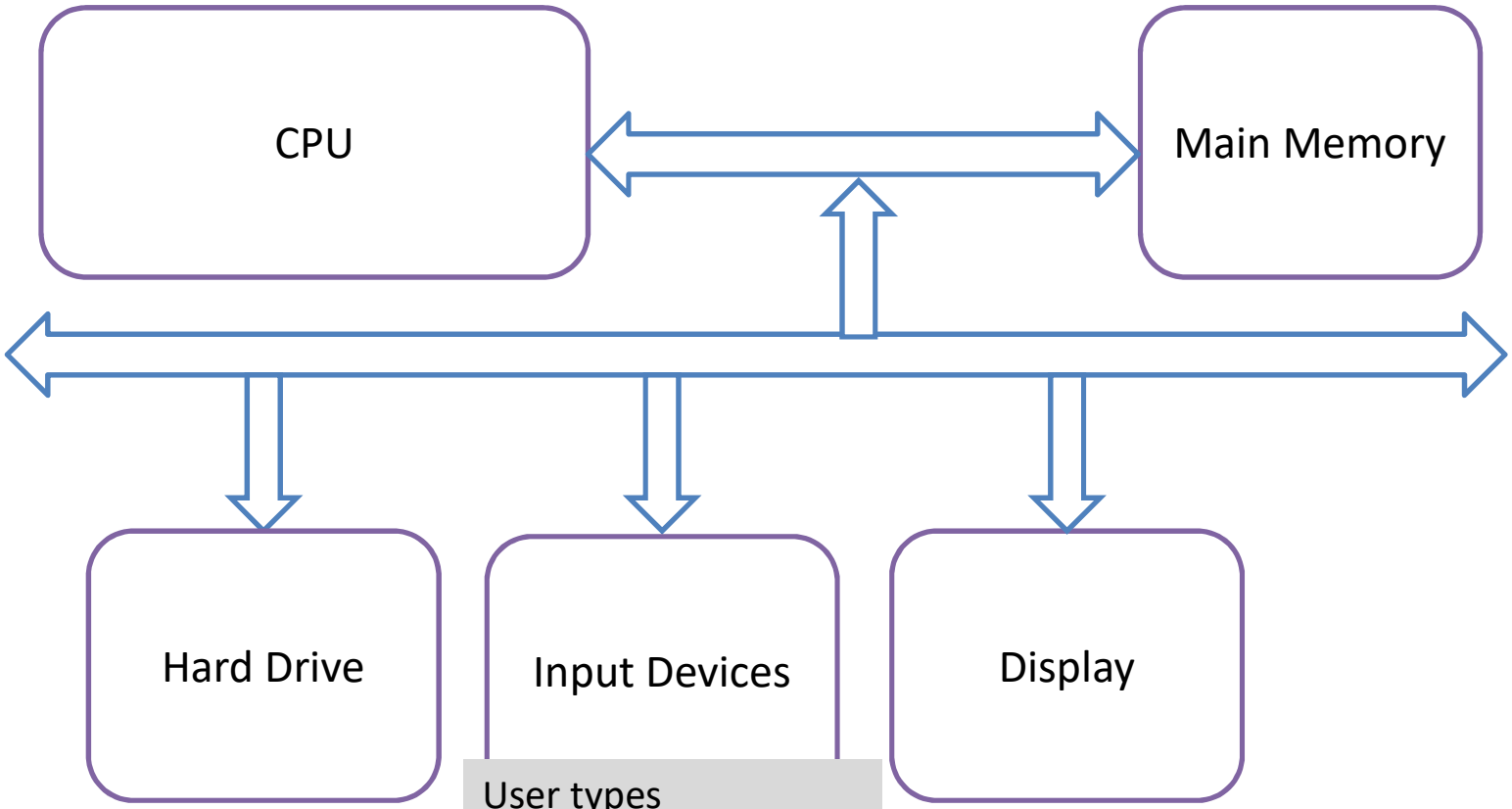
- Be mindful of the limitation of your file system, e.g. size and number of files

Directory Tree

- Files are organized in an inverted tree structure, aka “directory tree”



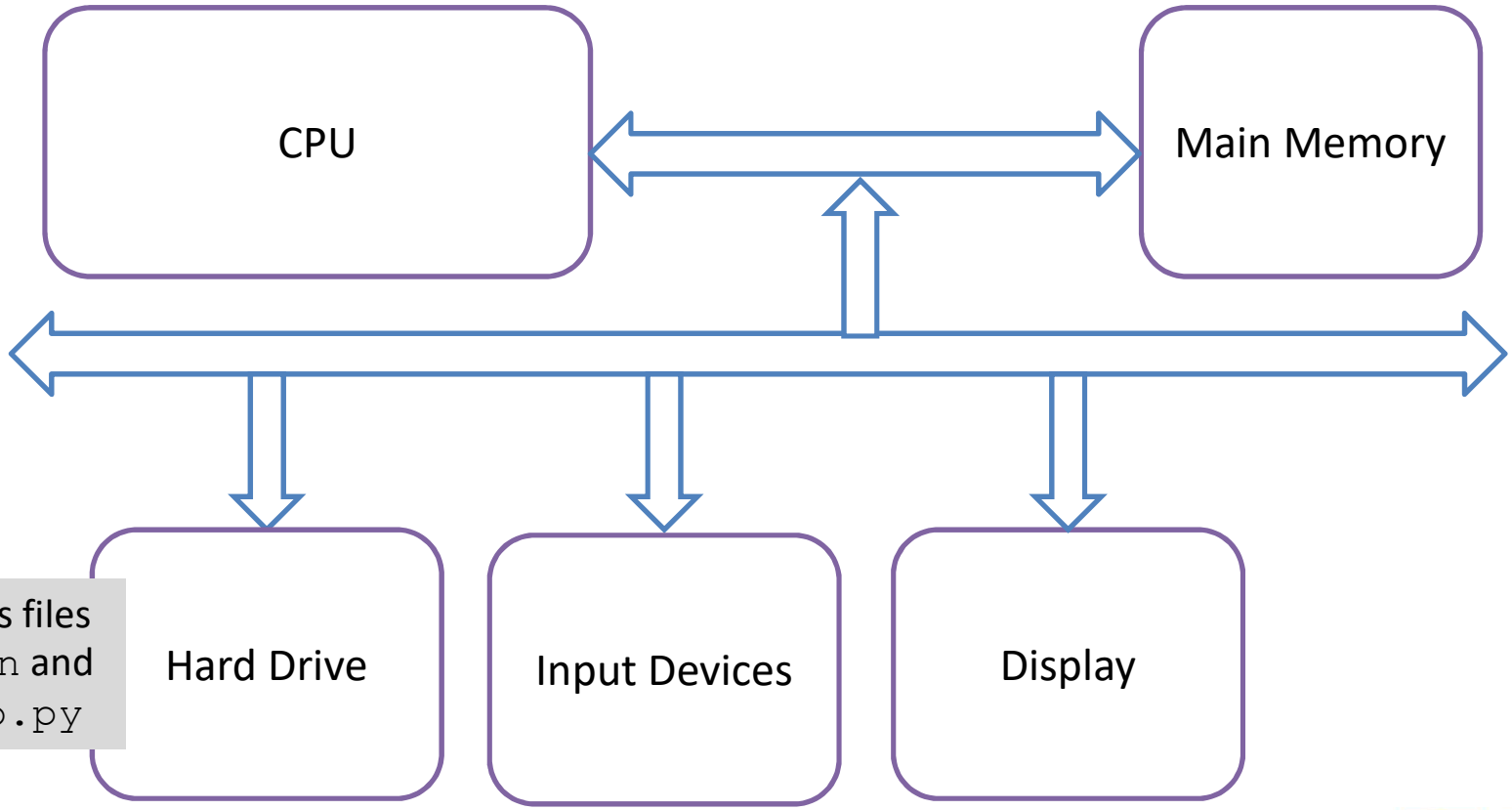
Running A Program: Step By Step



User types

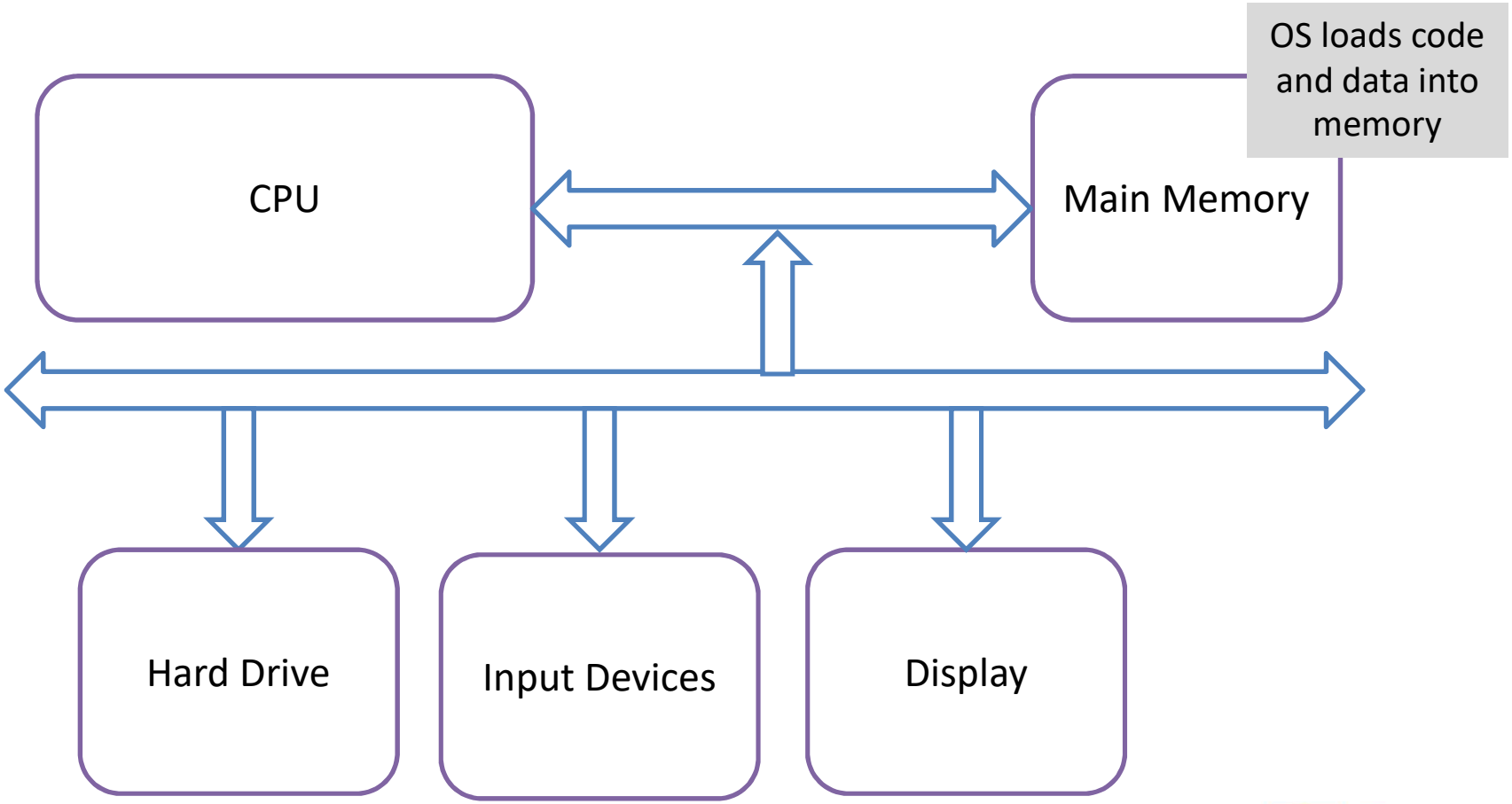
```
python hello.py
```

Running A Program: Step By Step

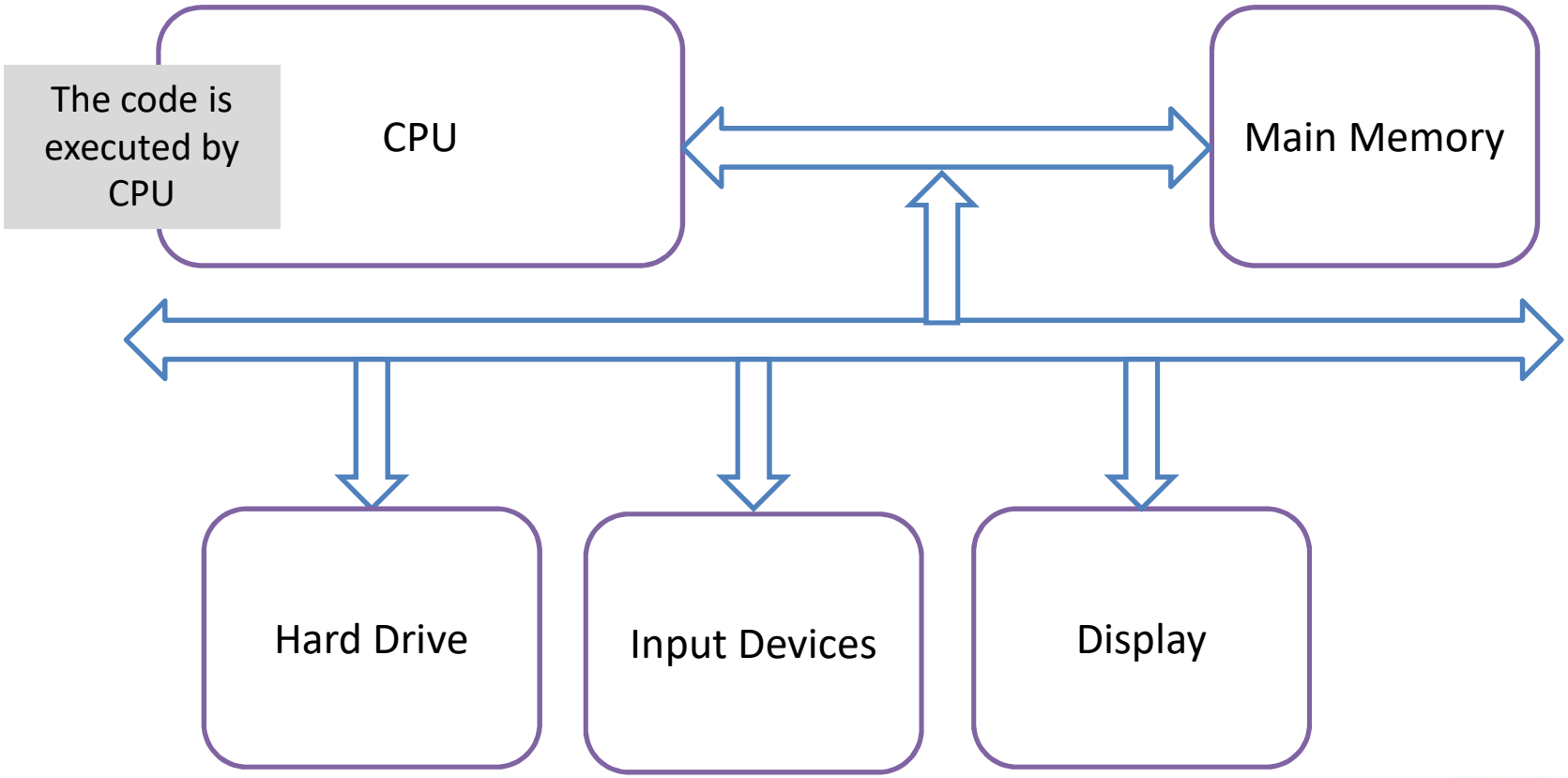


OS finds files `python` and `hello.py`

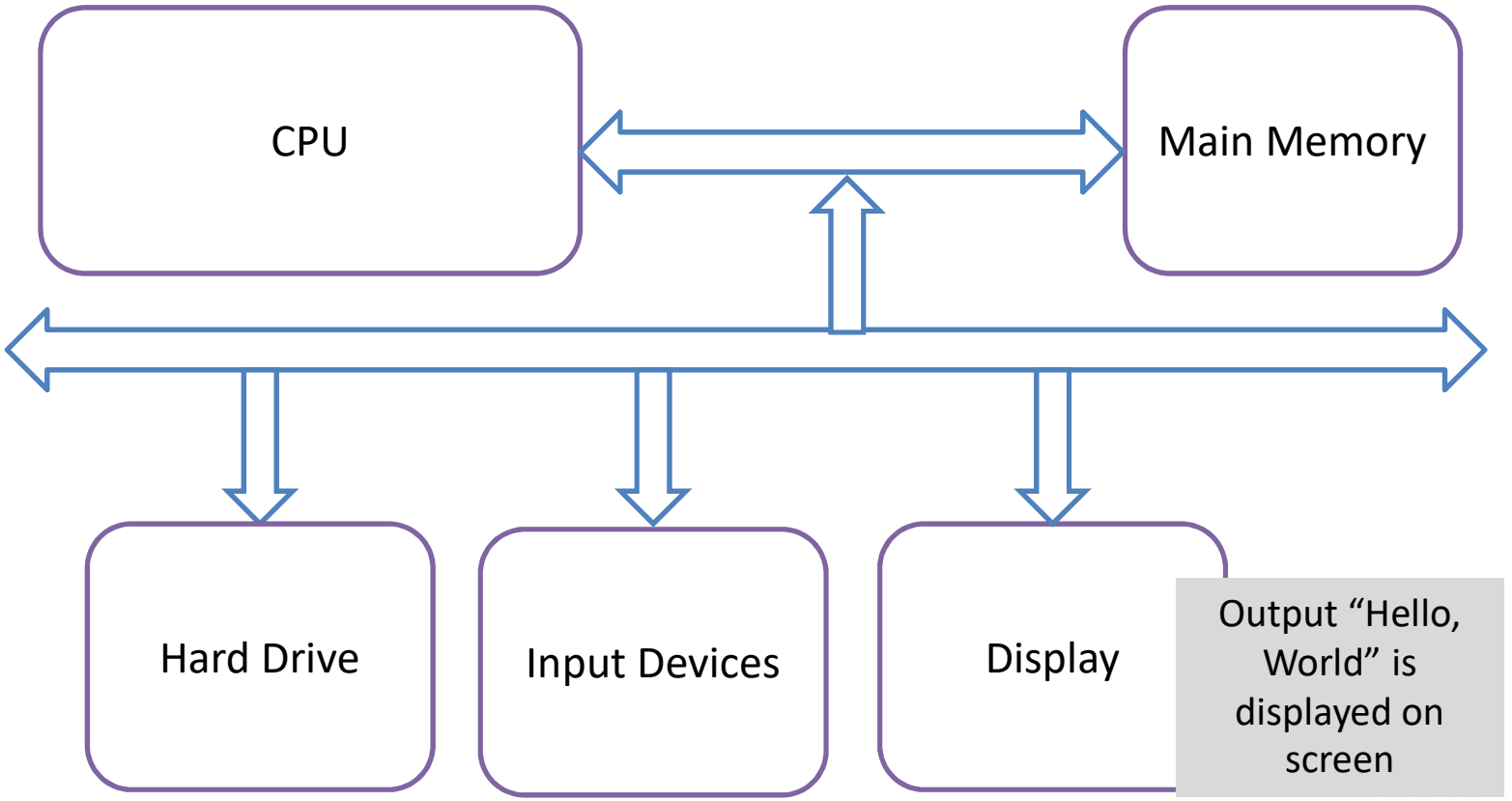
Running A Program: Step By Step



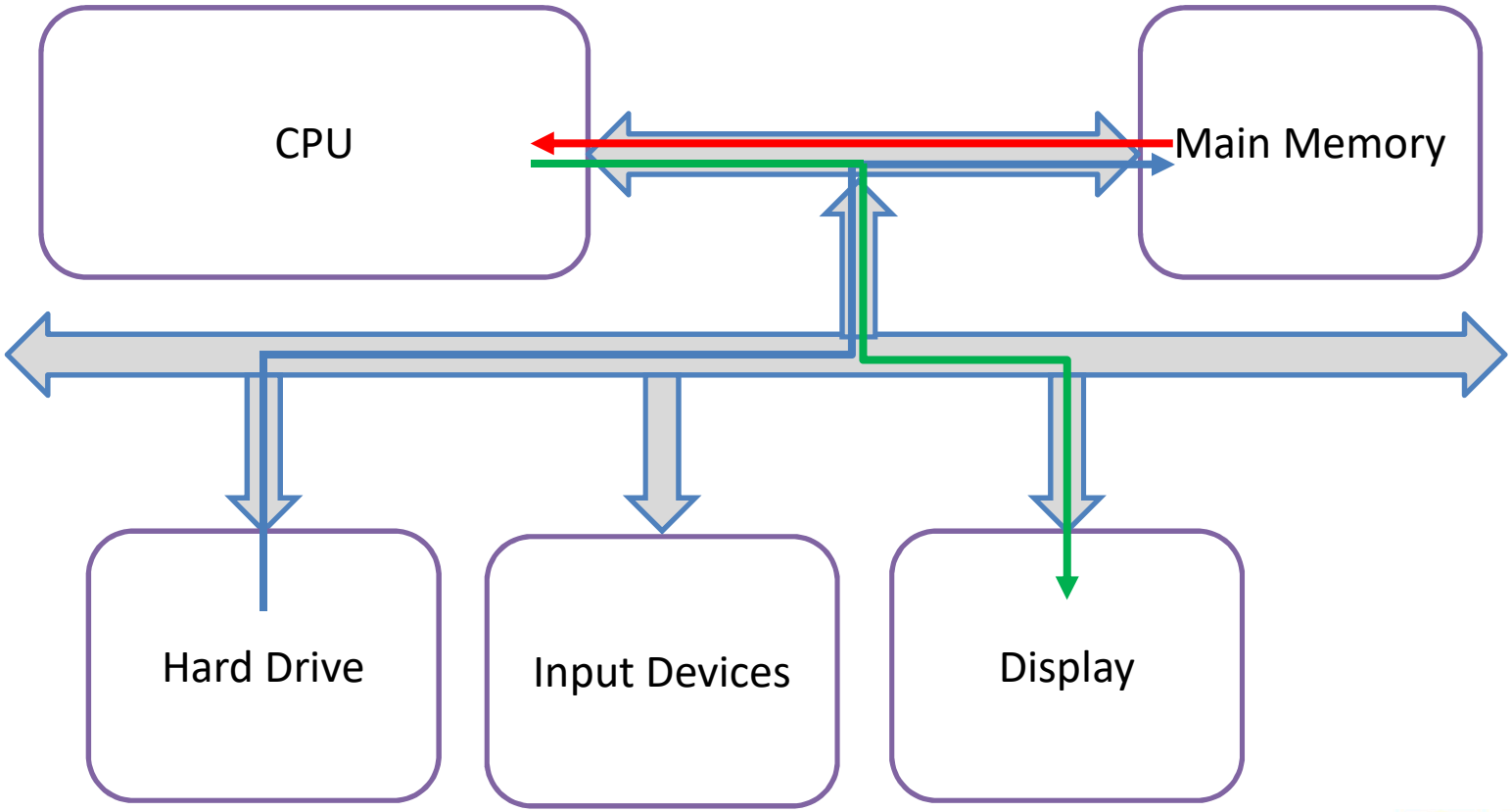
Running A Program: Step By Step



Running A Program: Step By Step

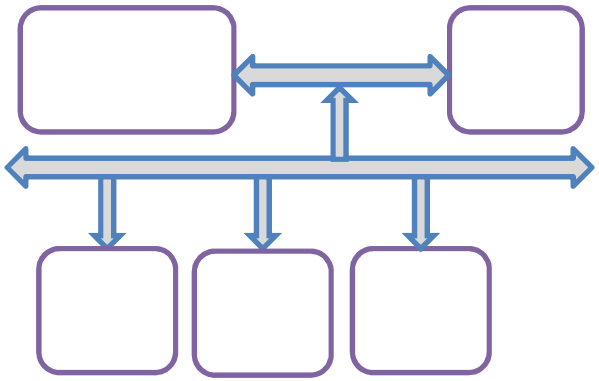


Running A Program: Data Movement



Data Movement is Overhead

- For something as small as `hello.py`, quite a bit of data transfer is involved
- All time spent in those "passages" is overhead
  - Therefore is bad for performance and should be avoided as much as possible



Tips for practitioners

Do not move (large amount of) data unless absolutely necessary.

Outline

- Basic concepts and terminology
- Computing systems
- Parallel computing

How Do We Improve Performance?

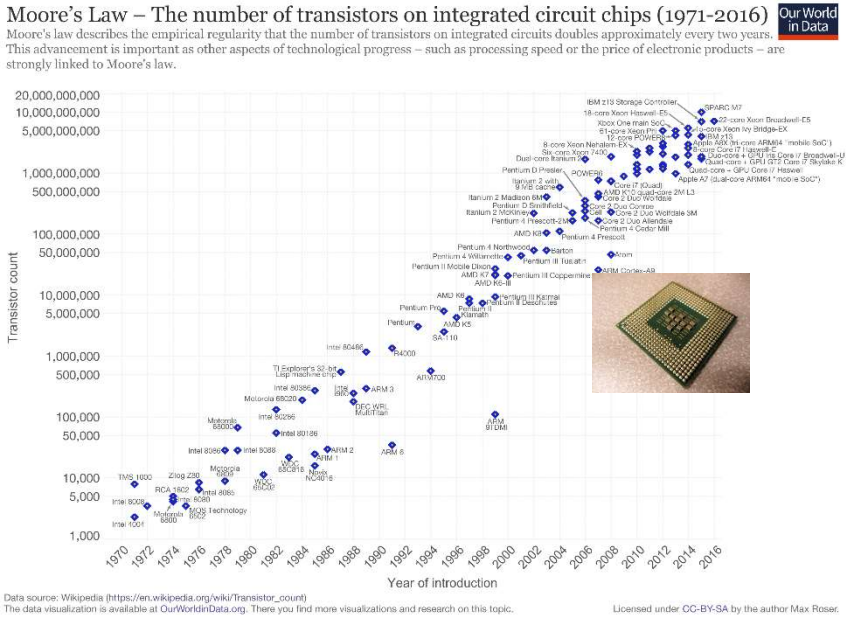
- Better performance is
  - To produce more products in a given period of time
  - To produce the same amount of products in less time
- How to get better performance?

How Do We Improve Performance?

- Better performance is
  - To produce more products in a given period of time
  - To produce the same amount of products in less time
- How to get better performance?
  - Faster processing for each processing unit
  - More processing units

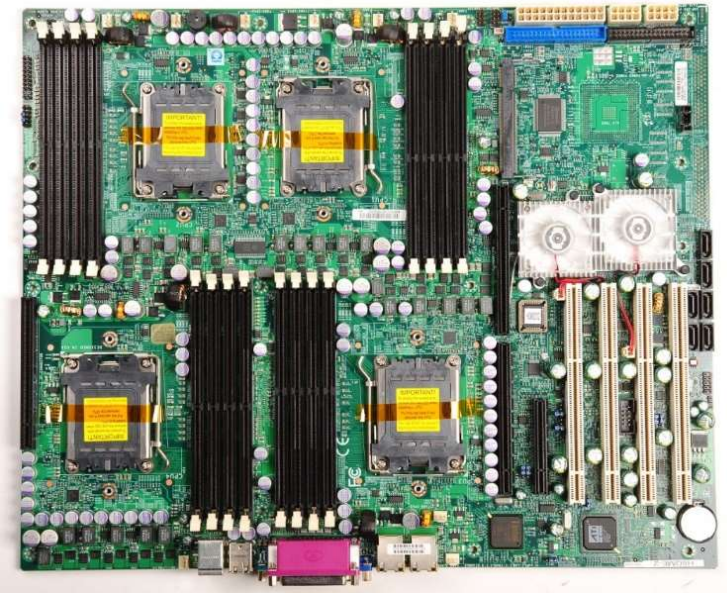
Moore's Law

- The "faster processing" route
- Moore's Law
  - Number of transistors on a CPU chip doubles every 18 months
    - Means more clock cycles per second
    - Programs will run twice faster every 18 months
  - Not any more: heat dissipation becomes an inhibitive problem

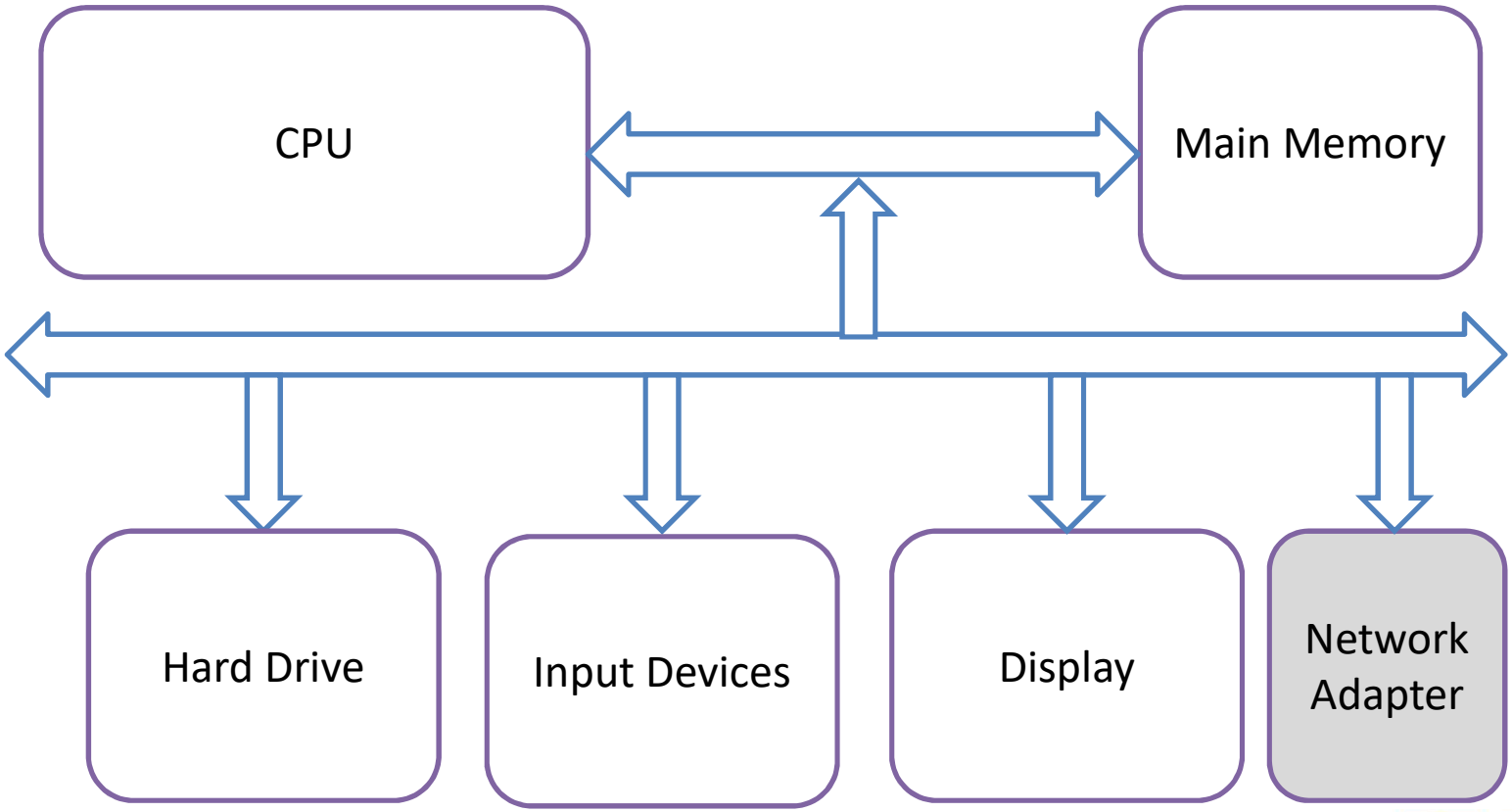


Parallel Processing

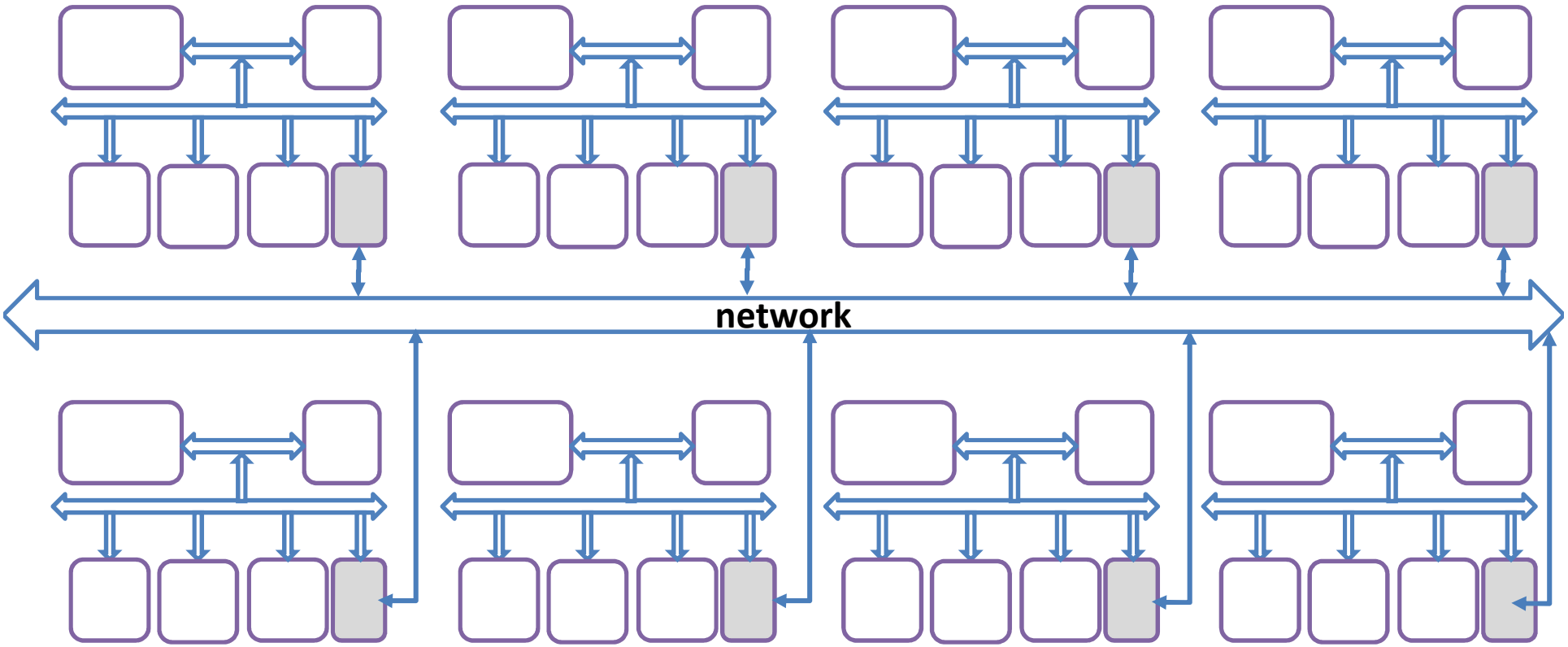
- The “more processing units” route
- Take advantage of parallelism and concurrency
  - Multiple data points per instruction
  - Multiple instructions per cycle
  - Multiple cores per CPU chip
  - Multiple CPU chips per computer



Extend Parallelism over Network

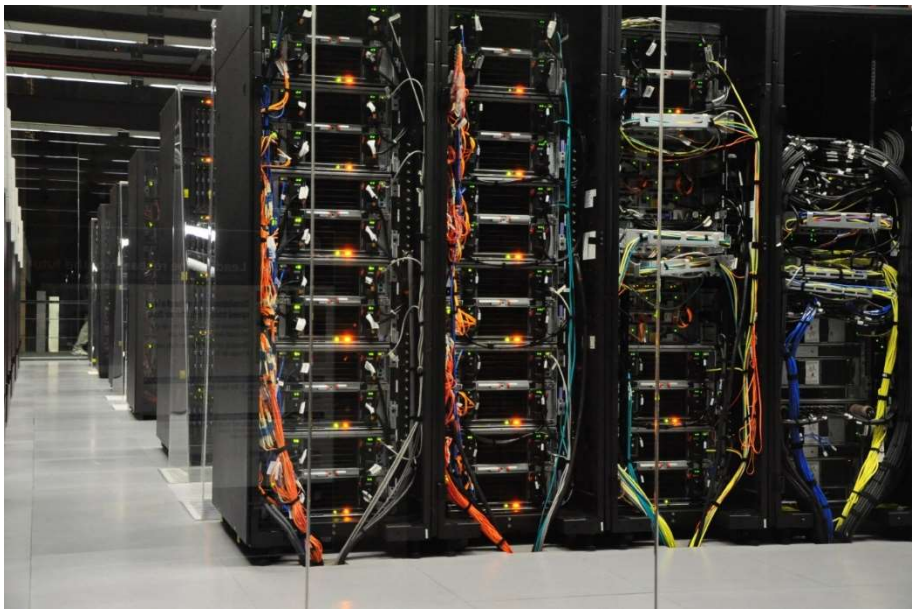


Extend Parallelism over Network



Extend Parallelism over Network: Clusters

- The majority of supercomputers are clusters
  - Individual servers connected by high speed network
  - Servers are known as “nodes” in a cluster



Parallel Programs

- Multiple processing units do NOT guarantee speedup
- You need programs that are aware of those units and capable of exploiting them
- If you use third party software, look for these keywords
  - Parallel, Multithreaded, MPI, OpenMP, etc.

Working In Parallel

- If it takes 4 hours for a worker to move 100 boxes from place A to place B, how long would it take 2 workers to perform the same task?

Working In Parallel

- If it takes 4 hours for a worker to move 100 boxes from place A to place B, how long would it take 4 workers to perform the same task?

Working In Parallel

- How about 100 workers? And 200?

Ideally…

Ideal world

# of Workers	Time (hours)	Worker * time
1	4	4
2	2	4
4	1	4
…	…	…
100	0.04	4

…But In Reality

Ideal world

# of Workers	Time (hours)	Worker * time
1	4	4
2	2	4
4	1	4
…	…	…
100	0.04	4

Reality

# of Workers	Time (hours)	Worker * time
1	4	4
2	2	4
4	1.1	4.4
…	…	…
100	0.2	20

Parallel Programs: Speedup

- Serial run-time = T_{serial}
- Parallel run-time = $T_{parallel}$
- Speedup of a parallel program

$$S = \frac{T_{serial}}{T_{parallel}}$$

Parallel Programs: Speedup

- Serial run-time = T_{serial}
- Parallel run-time = $T_{parallel}$
- Speedup of a parallel program

# of Workers	Time (hours)	Worker * time	Speedup
1	4	4	4/4 = 1
2	2	4	4/2 = 2
4	1.1	4.4	4/1.1 = 3.6
…	…	…	
100	0.2	20	4/0.2 = 20

Parallel Programs: Efficiency

- Number of workers = p
- Definition of efficiency:

$$E = \frac{S}{p} = \frac{\left(\frac{T_{serial}}{T_{parallel}} \right)}{p} = \frac{T_{serial}}{p \cdot T_{parallel}}$$

Parallel Programs: Efficiency

- Number of workers = p
- Definition of efficiency:

# of Workers	Time (hours)	Worker * time	Speedup	Efficiency
1	4	4	4/4 = 1	1/1 = 1
2	2	4	4/2 = 2	2/2 = 1
4	1.1	4.4	4/1.1 = 3.6	3.6/4 = 0.9
…	…	…	…	…
100	0.2	20	4/0.2 = 20	20/100 = 0.2

Parallel Programs: Scaling

- Ideally
  - Speedup is always equal to the number of workers
  - Efficiency is always equal to 100%
- But in reality
  - Efficiency is ever deceasing with increasing number of workers
    - When the efficiency is less than 1/(number of cores), what happens?

Parallel Programs: Scaling

- How the efficiency/speedup of a parallel program changes with the number of cores is called the “scaling behavior”
- The scaling behavior depends on the underlying hardware
- It depends on the problem size too

Parallel Programs: Scaling

- How the efficiency/speedup of a parallel program changes with the number of cores is called the "scaling
- The scaling behav
underlying hardw
- It depends on the

Tips for practitioners

- Make sure your program is capable of running in parallel
- Make sure your program is actually running in parallel
- Be mindful of the scaling behavior of your program so that it runs optimally

Questions?