Introduction to Computing Systems - Scientific Computing's Perspective

Le Yan
HPC @ LSU

Why We Are Here

- For researchers, understand how your instrument works is a key element for success
- Maximize productivity
  - If computing systems will be your (primary) research instrument
- Not just research
  - A driver doesn't have to be a mechanic, but
  - Having mechanic knowledge will be very helpful for drivers to fulfil their goals

Productivity: Have An Expectation for Performance

- Would you be suspicious
  - If it takes 5 minutes to open Microsoft Word on your laptop?
  - If it takes 15 minutes to download a song to your phone?
  - It takes 15 hours to assembly and align a 5GB sequence?

Productivity: You Are Suspicious. Now What?

- Fortnite (or Youtube or whatever app) is laggy. What would you do?
  - Check network?
  - Other programs running on your computer?
  - Fan stopped running?
  - …?

Productivity Matters in Scientific Computing

- 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 lots of pre-processing and post-processing

Productivity Matters in Scientific Computing

- 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 lots of pre-processing and post-processing

CENTER FOR COMPUTATION & TECHNOLOGY

Productivity Matters in Scientific Computing

- 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 lots of pre-processing and post-processing

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

CENTER FOR COMPUTATION & TECHNOLOGY

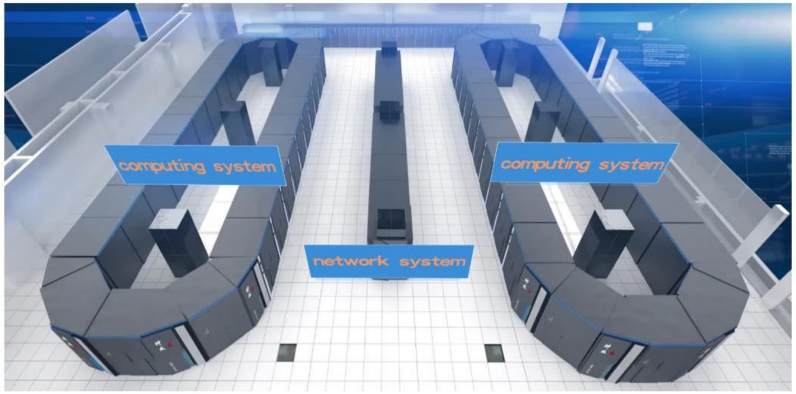
Productivity Matters in Scientific Computing

- 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 lots of pre-processing and post-processing

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

There are Similar Problems on Supercomputers

- Supercomputers are truly “super”
  - The biggest has more than 10 million CPU cores and consumes 15 MW when fully loaded
- But, being “super” doesn’t necessarily make performance issues go away



Goals

- Through a tour of computing systems, establish a basic understanding how their components work, and its implication on performance
- Learn a few useful Linux command line tools

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

What Do Computers Do?

- To compute, of course.
- What does it mean exactly?

What Do Computers Do?

- To compute, of course.
- What does it mean exactly?
- Computers process data, transforming it from one form to another
  - Example: open a web page on your laptop
  - Example: play video games
  - Example: analyze genome sequence

What Do Computers Do?

- In each of the examples
  - Data of some form is obtained from some places
    - Input devices (mouse and keyboard)
    - Other computers and devices
  - The data is processed by the computer
    - And presented (or stored) in the processed form

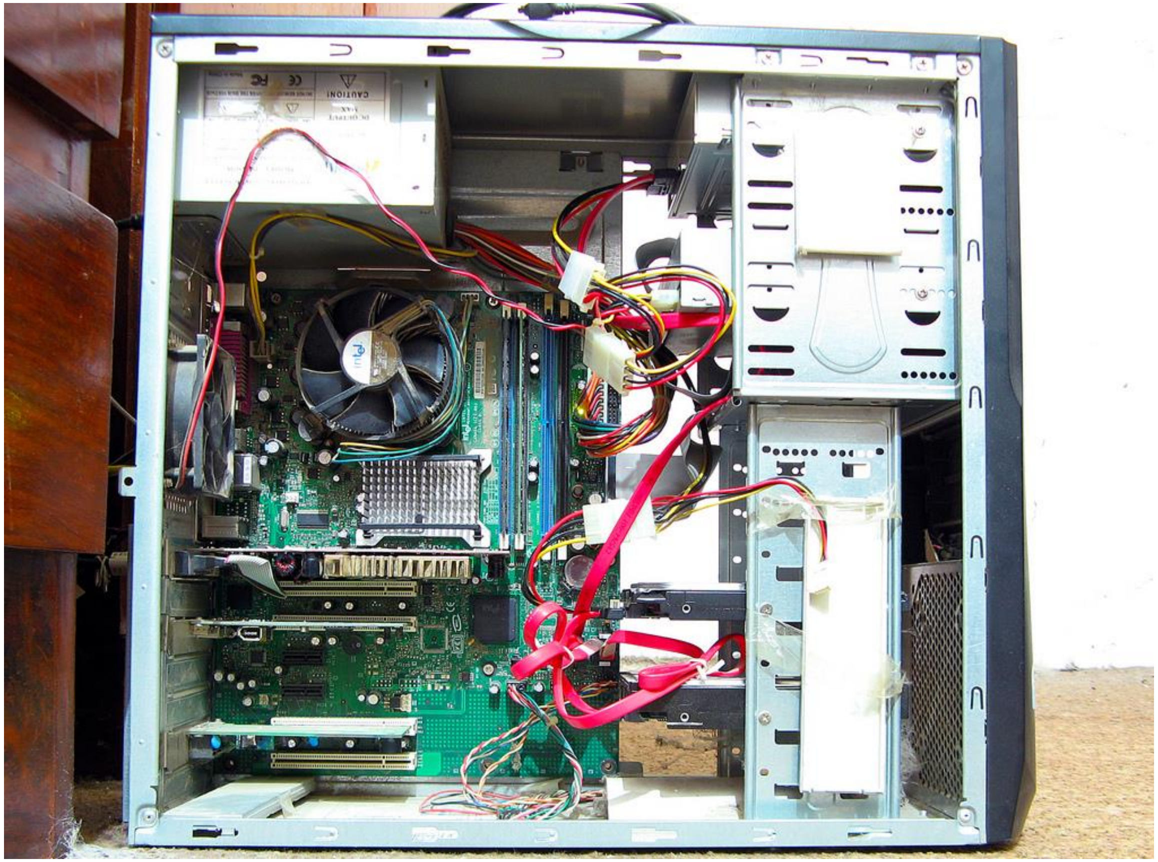
Computers vs. Manufacturing Plants

Components of Manufacturing Plants

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

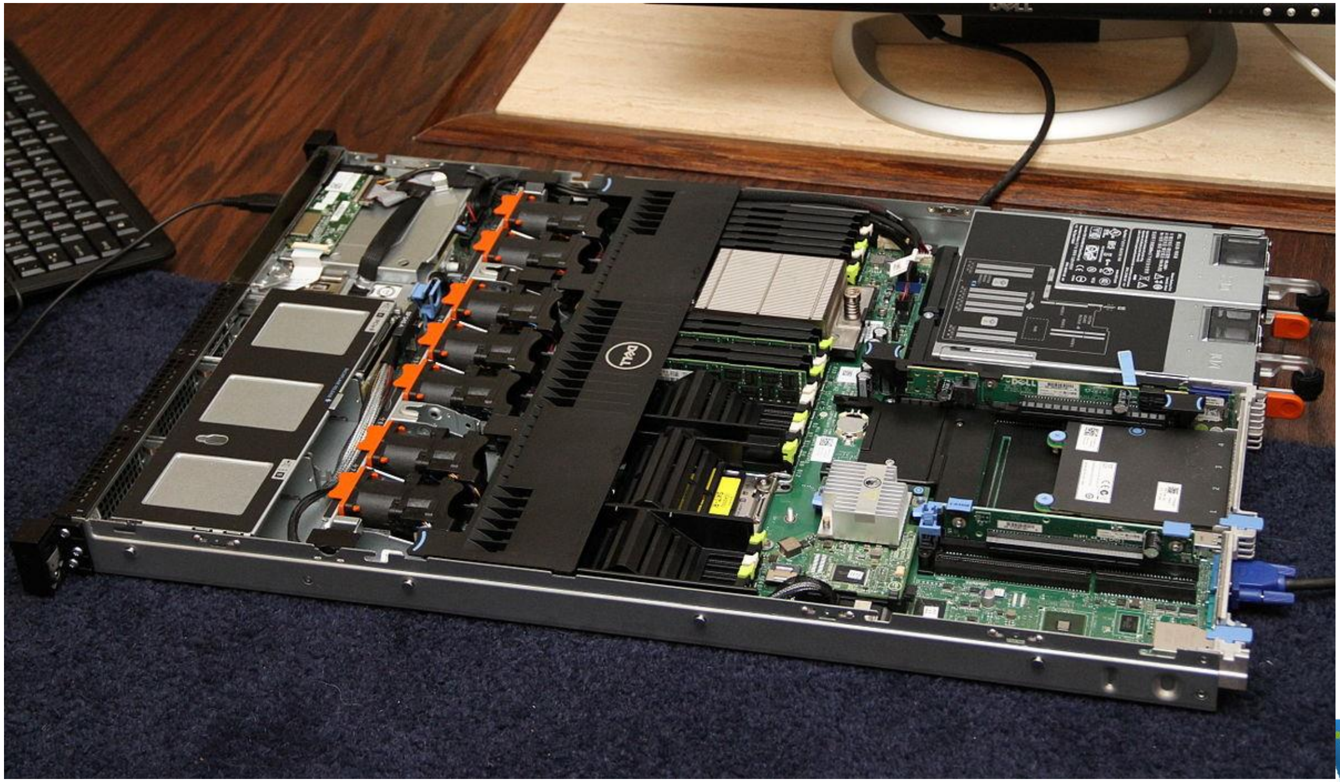
Components of Computers

- Recognize anything?



Component of Computers

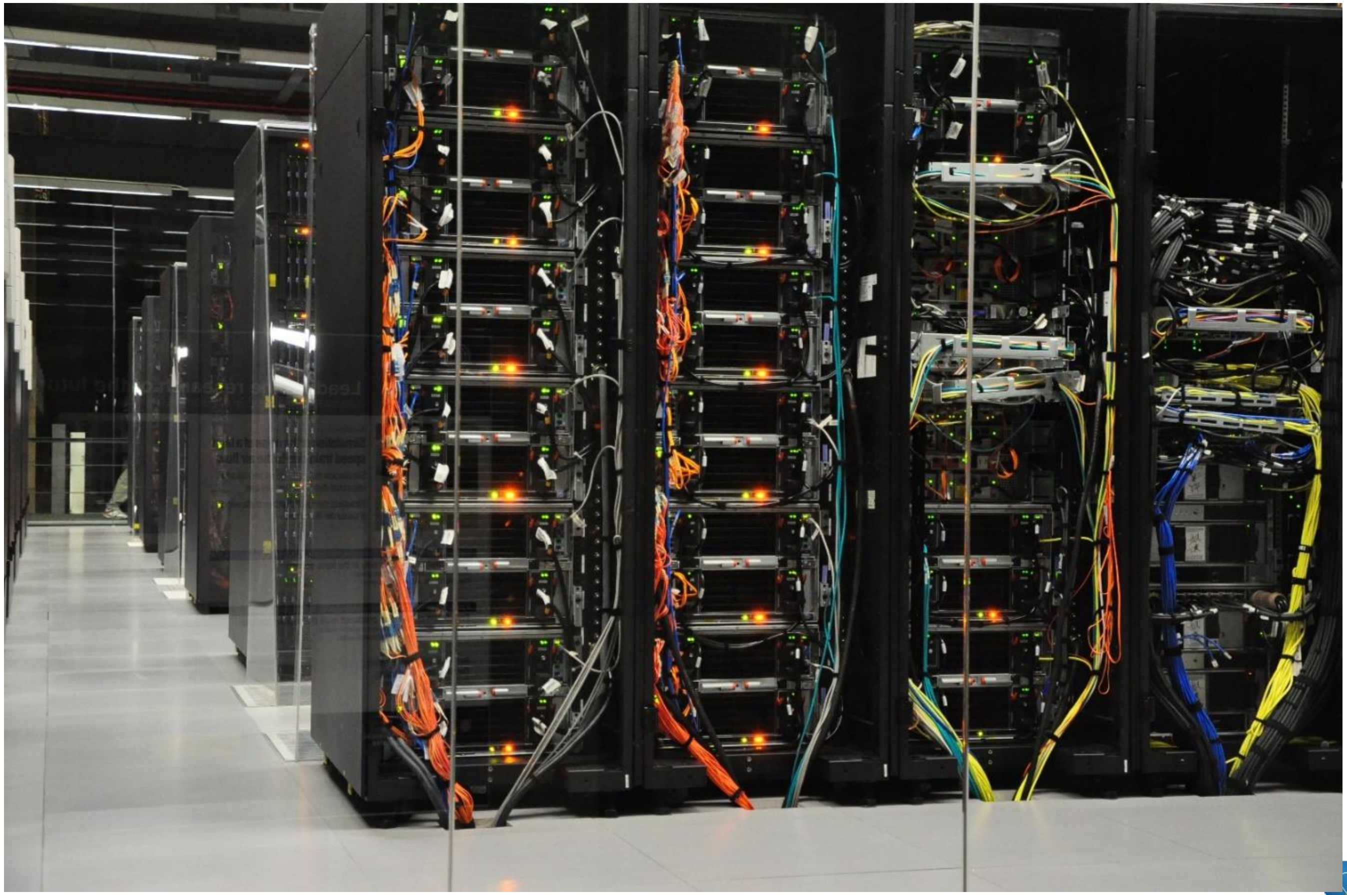
- What about this?



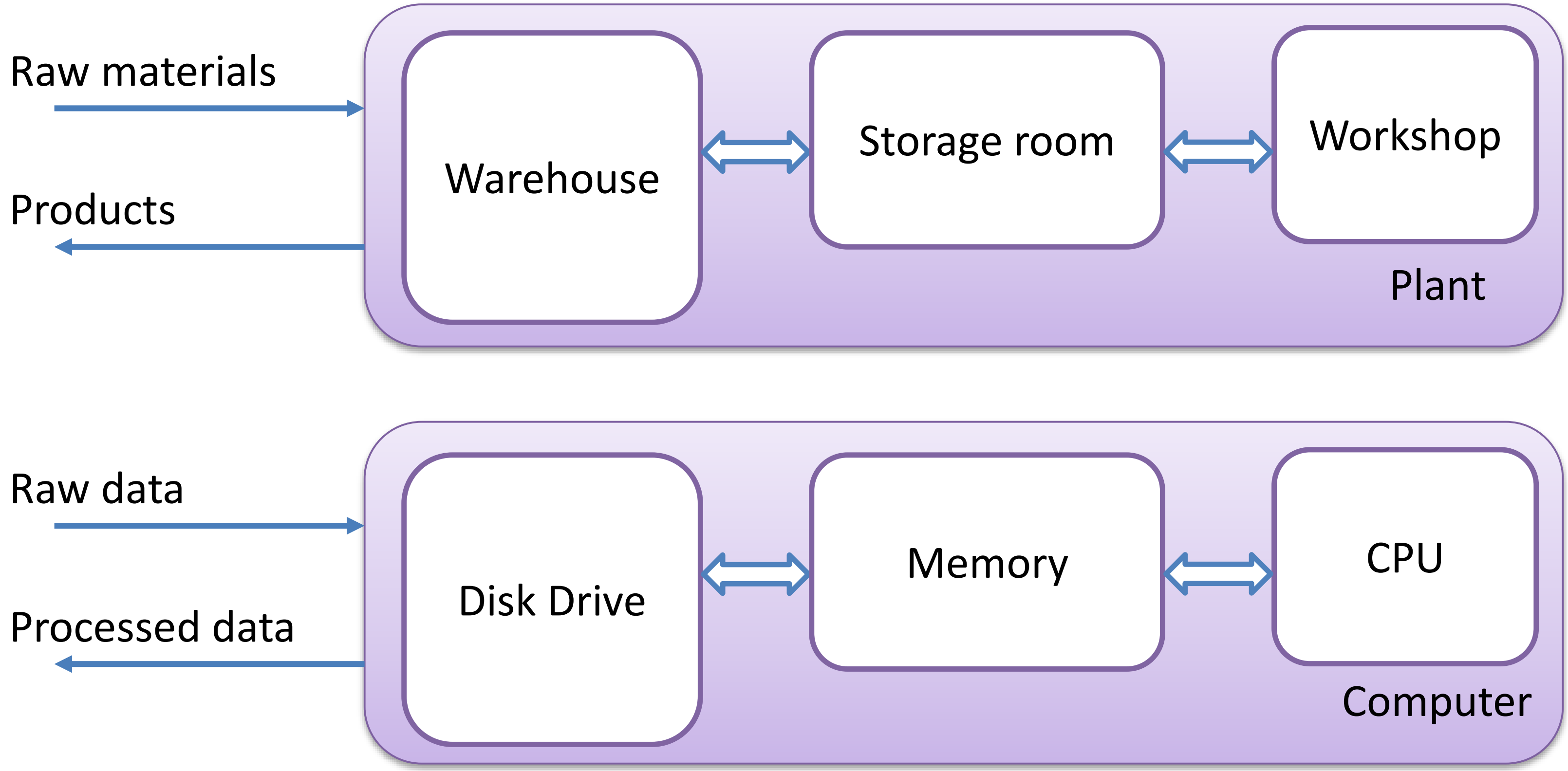
Component of Computers

- And this?

Component of Computers



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

CENTER FOR COMPUTATION & TECHNOLOGY

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 missing?

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)

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

Data – Know Your Raw Materials

- All information in a computing system is represented by a bunch of 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
    - …

CENTER FOR COMPUTATION & TECHNOLOGY

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	ö	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	

LSU CENTER FOR COMPUTATION & TECHNOLOGY

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

CENTER FOR COMPUTATION & TECHNOLOGY

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

Measuring Data: 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
  - Capacity, e.g. memory (~GB) and hard drive (~TB)
  - Bandwidth, e.g. network (~Gb)
  - Data size, e.g. reference genomes (~GB)
- Knowing these numbers helps put things into perspective
  - For instance, how fast can raw materials (data) can be processed by the workshops (CPU)? How long does it take to transfer this much materials (data) from warehouse (hard drive) to staging area (memory)?

Command Line Interface

- In scientific computing, we often need to run programs from the command line
  - With some flavor of the Linux/Unix operating systems
- How to access the command line with Jupyter Notebook
  - On the Jupyter Notebook page, click on “new”, then select “terminal” in the pull down menu

Useful Commands

- `file`: reveals the type of data contained in a computer file.
  - Syntax: `file <file name>`
- `cat`: reads files sequentially and print them to standard output.
  - Syntax: `cat <file name>`
- `ls`: lists the names and attributes of the files in the current working directory
  - Syntax: `ls [-l] [<file name>]`

Exercise

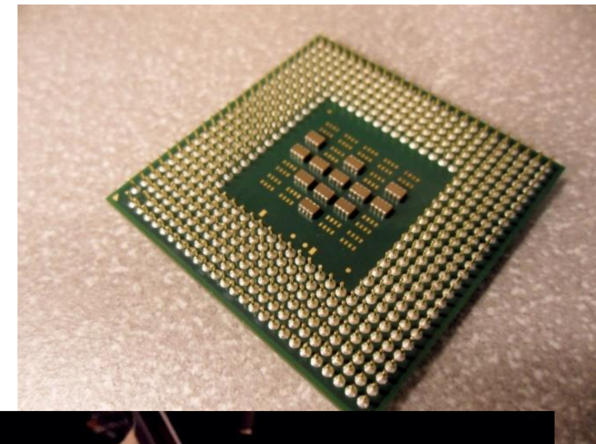
- Open the terminal in Jupyter Notebook
- Type `cd lbrnloniworkshop2018/day1_intro`
- Type `python hello.py`
- Find out the size of `hello.py`.
- Find out the type (text or binary) of `hello.py`.
- Print the content of `hello.py`.

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

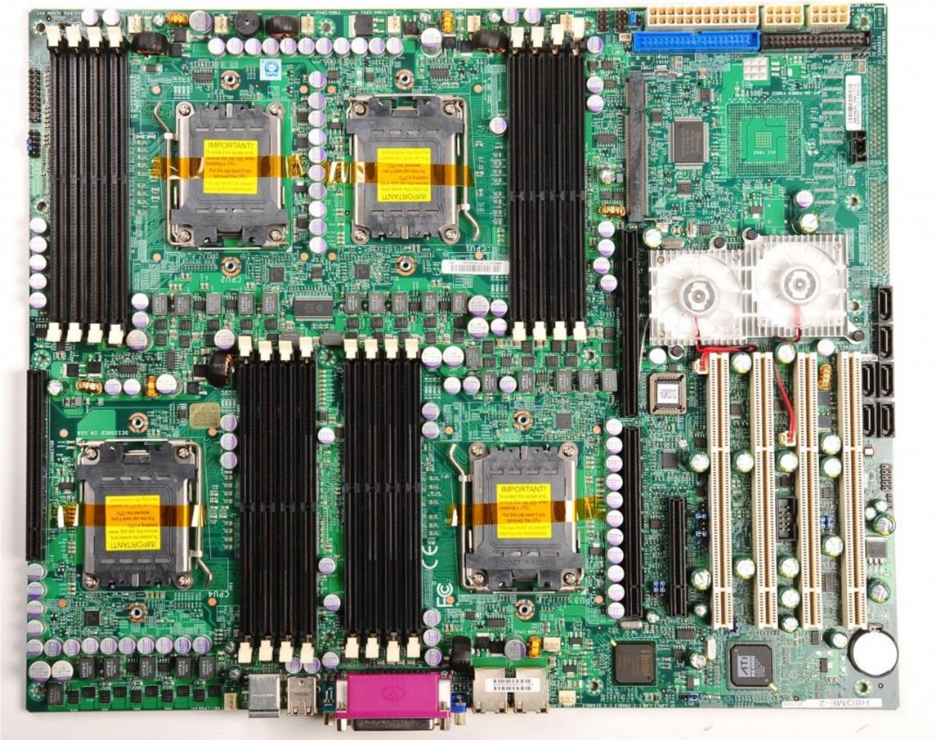
Central Processing Unit (CPU) (1)

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

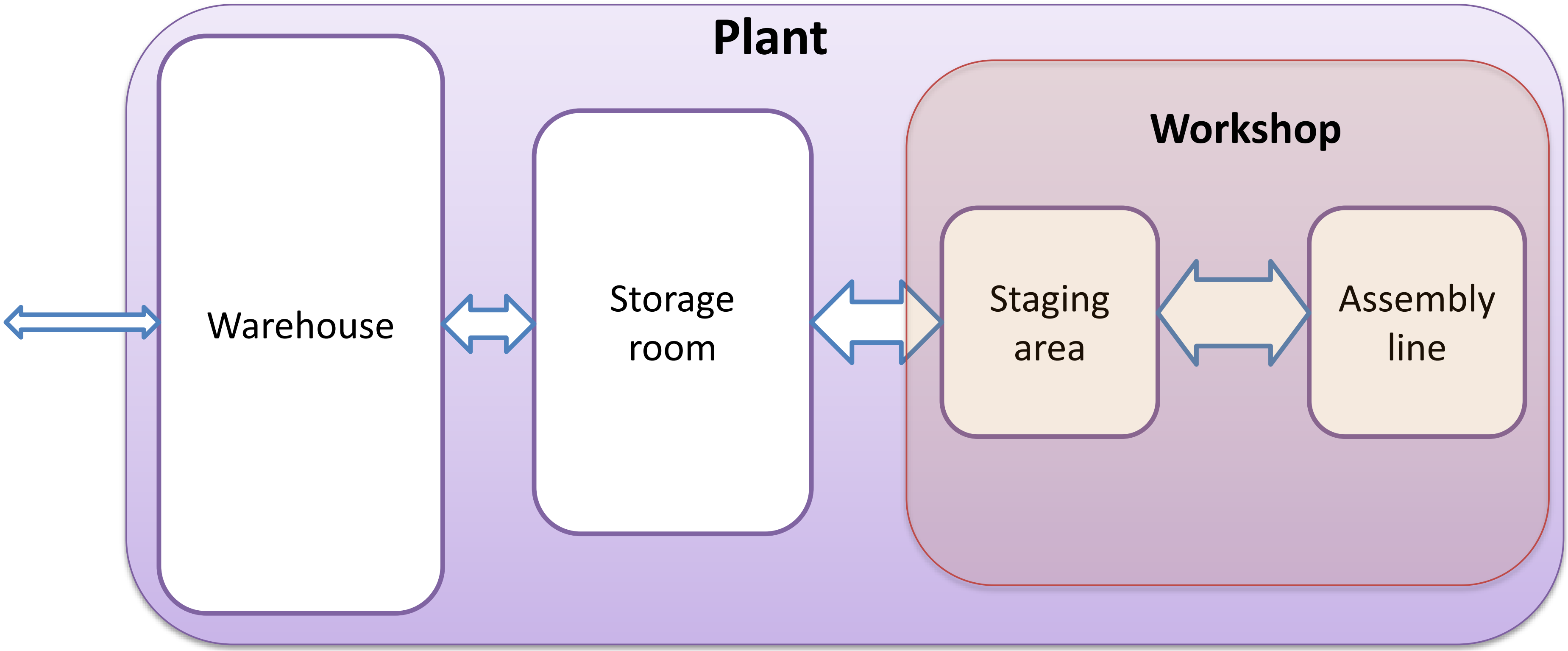


Central Processing Unit (CPU) (2)

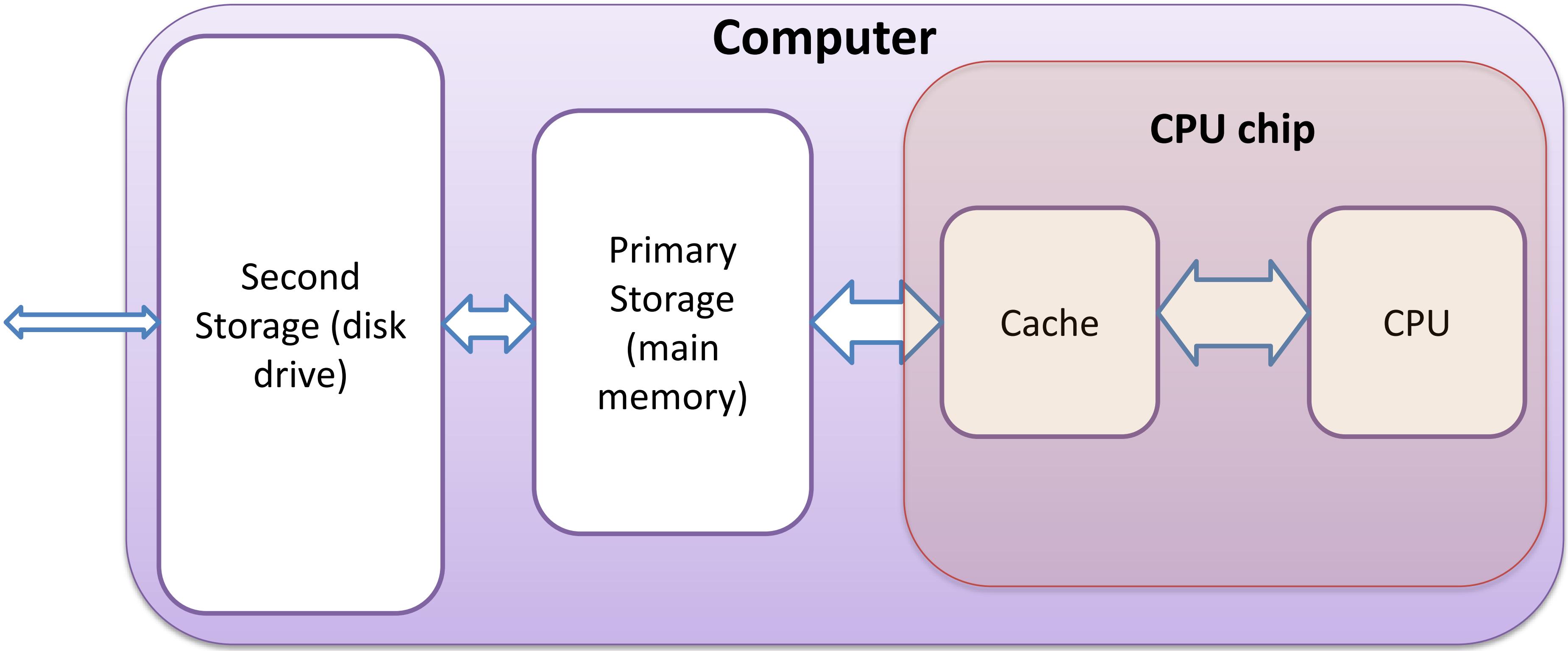
- A computer can have more than one CPU
- How many CPU sockets are shown in this picture?



A Closer Look: Workshop



A Closer Look: CPU



How CPU's 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

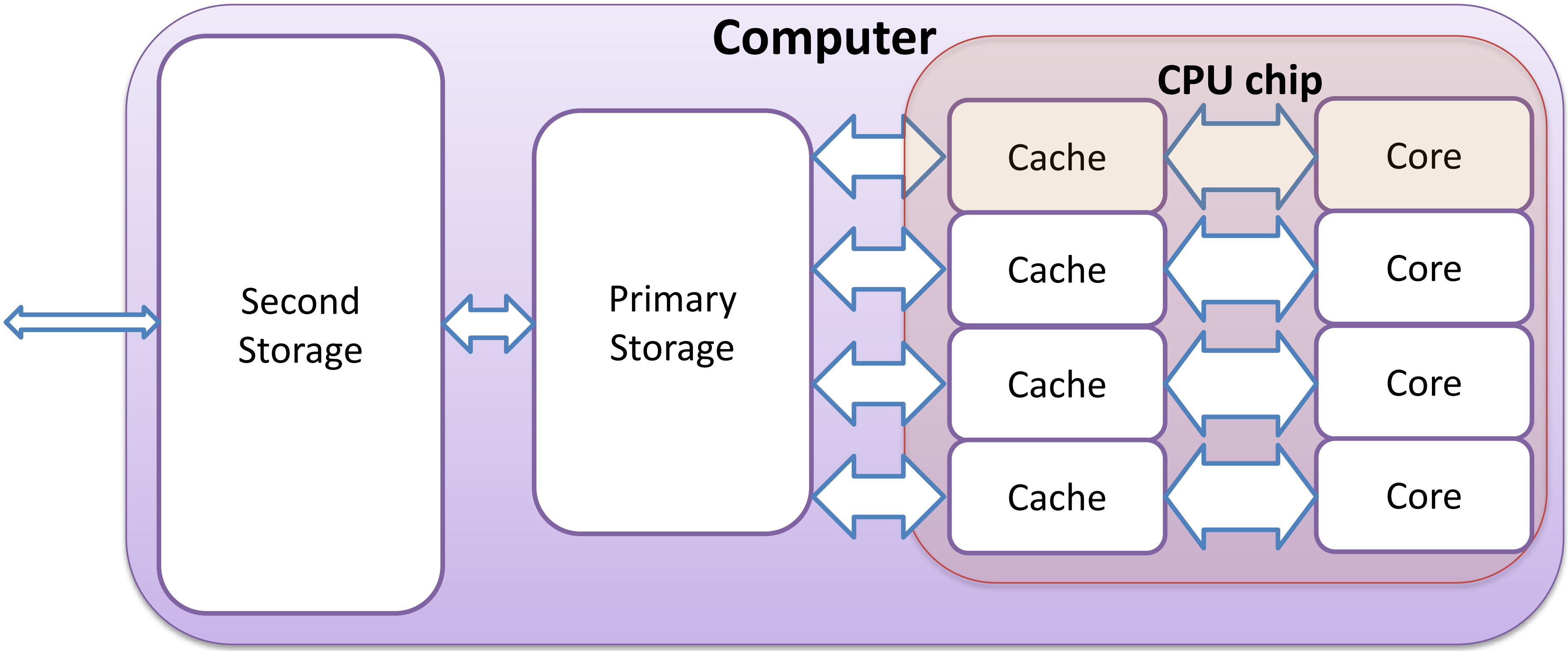
Concurrency

- 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

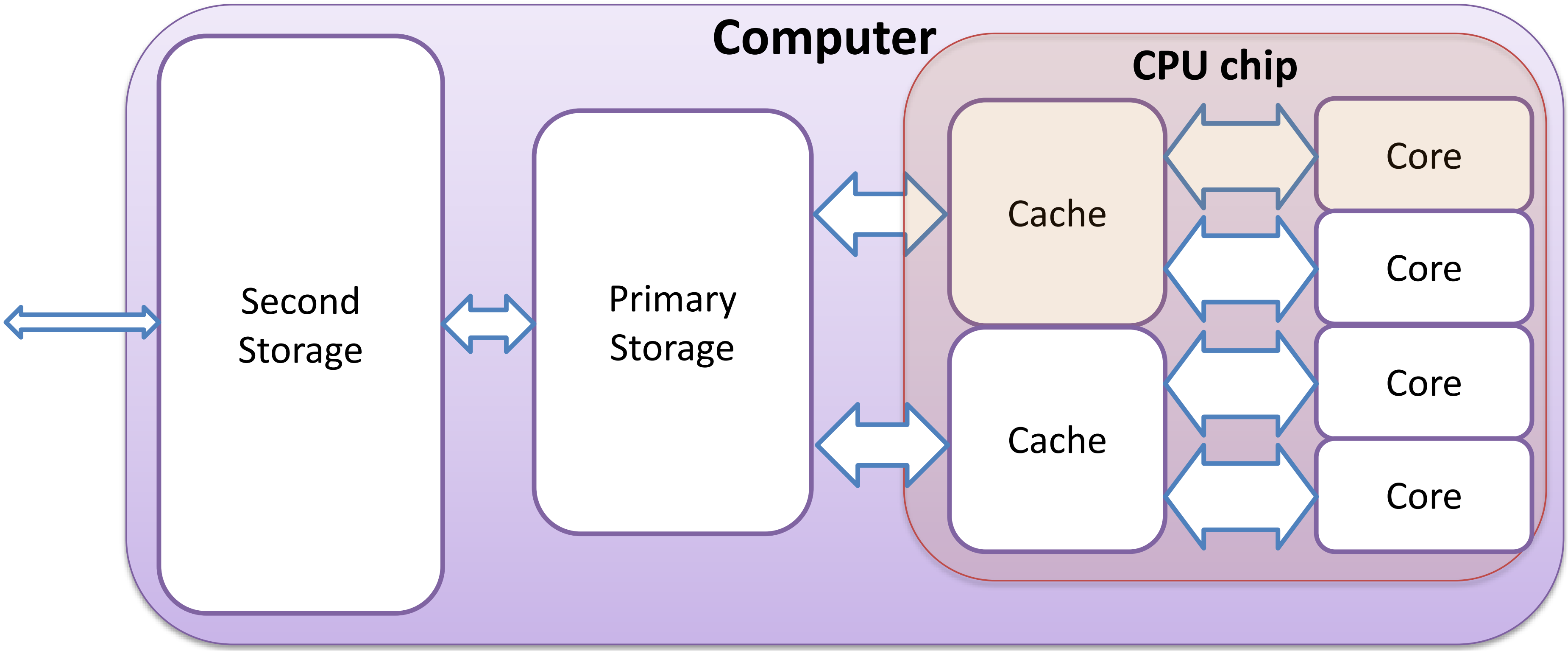
Concurrency

- 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 CPU's (Skylake family) could have anywhere between 2 to 28 cores
  - Instruction level: e **Pipelining** than one instruction per clock cycle by o stages of execution
  - Vectorization: perf **Multiple parts on each assembly lin** instruction

Different Design with MultiCores (1)



Different Design with MultiCores (2)



FLOPS

- FLOPS = **FL**oating **O**perations **P**er **S**econd
  - Another way of measuring how fast a computer is ("The way" in scientific computing)
- 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

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

CENTER FOR COMPUTATION & TECHNOLOGY

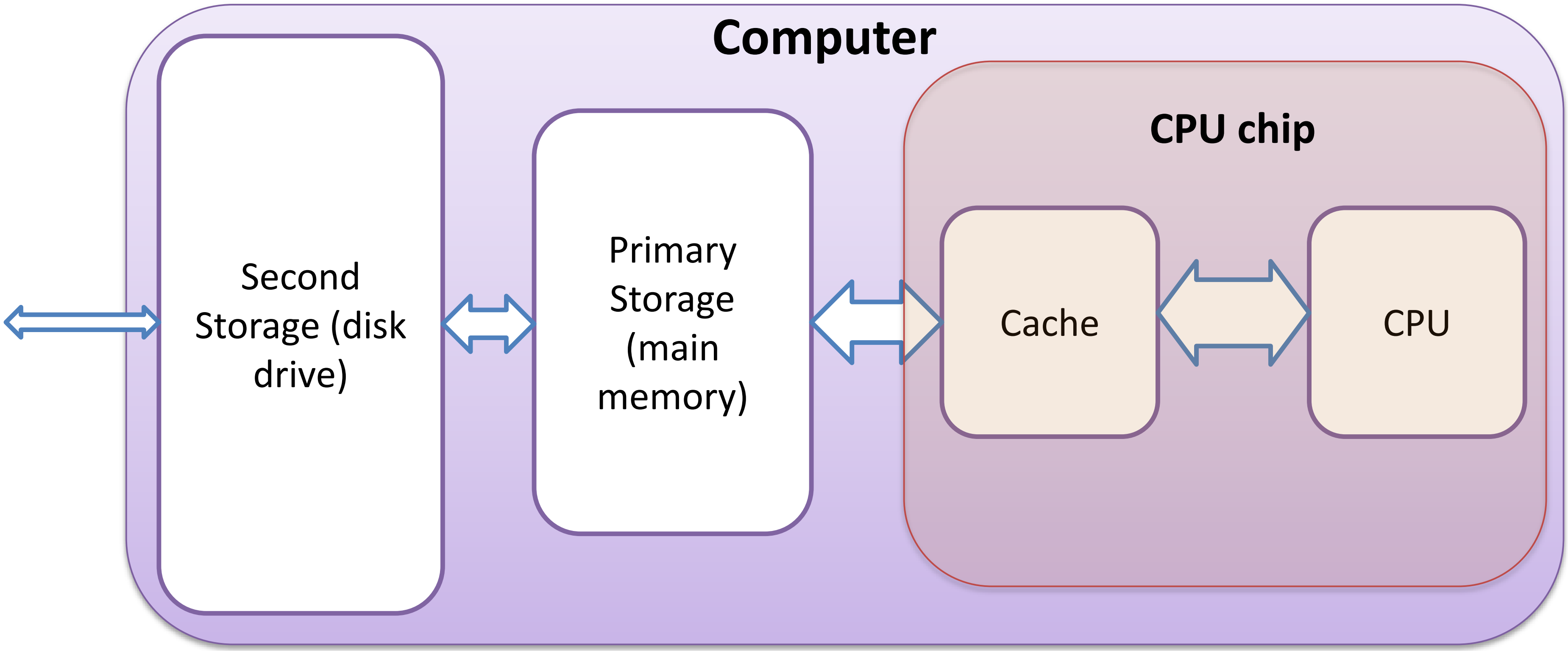
Exercise

- Explain this sentence: “A dual socket server is equipped with two 10-core Intel processors @ 2.60 GHz.”
- Print the content of file `/proc/cpuinfo` and find out how many sockets and cores there are on the server where your Jupyter Notebook is running.

Outline

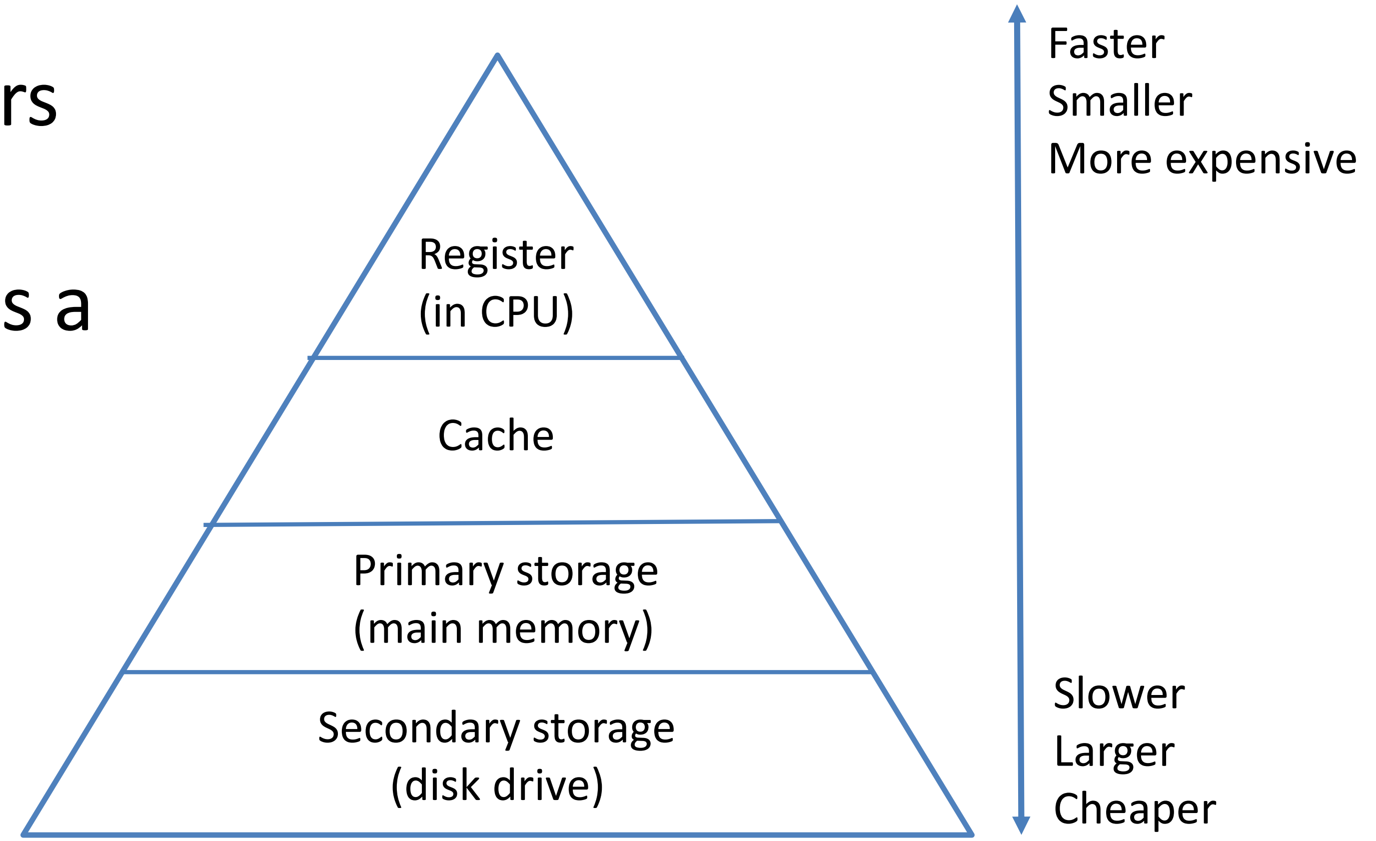
- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

Storage Hierarchy



Storage Hierarchy

- In computers storage is organized as a hierarchy

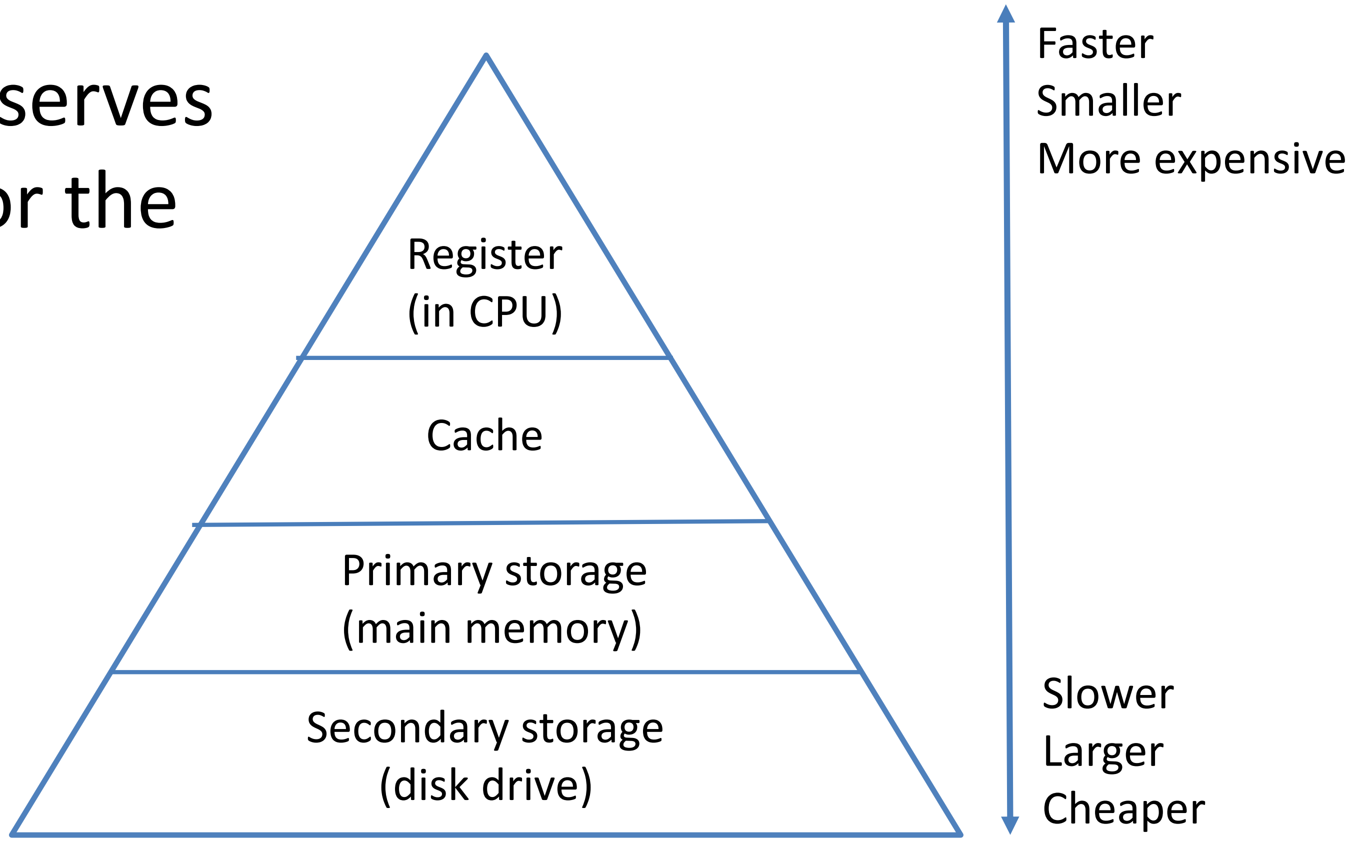


Why Use Cache?

- 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

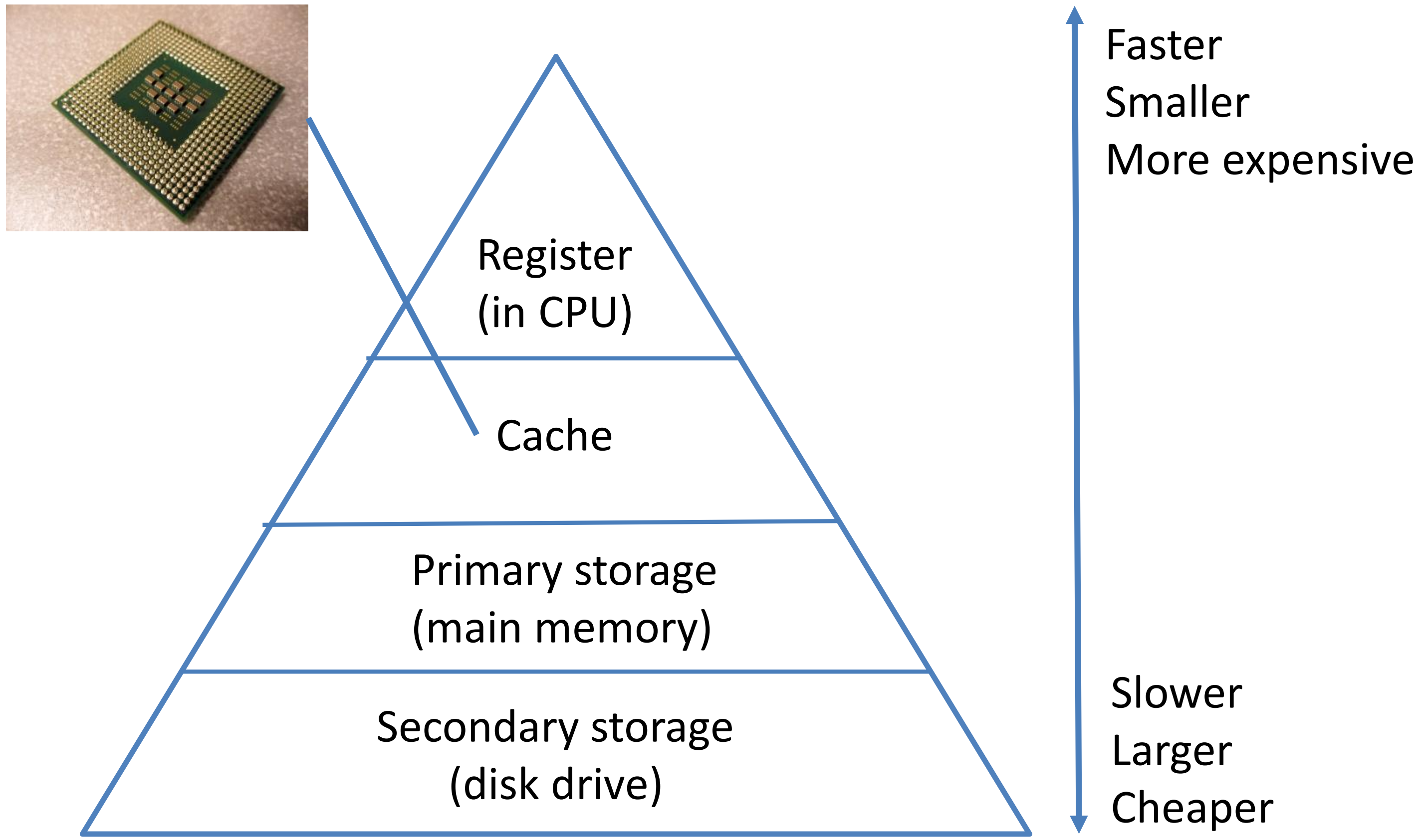
Storage Hierarchy

- Each level serves as cache for the next level



Storage Hierarchy: Devices

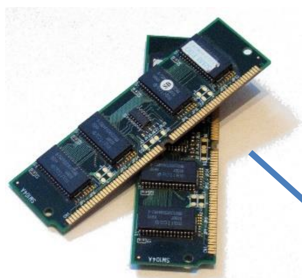
Cache: **S**tatic **R**andom **A**ccess **M**emory (SRAM)



Storage Hierarchy: Devices

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

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



Register
(in CPU)

Cache

Primary storage
(main memory)

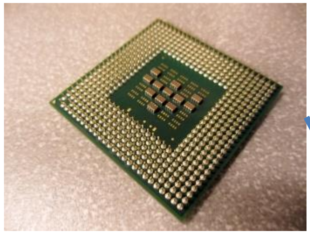
Secondary storage
(disk drive)

Faster
Smaller
More expensive

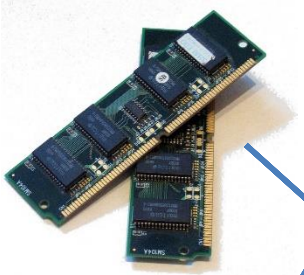
Slower
Larger
Cheaper

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

Register
(in CPU)

Cache

Primary storage
(main memory)

Secondary storage
(disk drive)

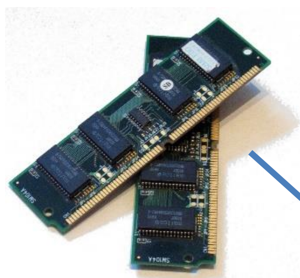
Faster
Smaller
More expensive

Slower
Larger
Cheaper

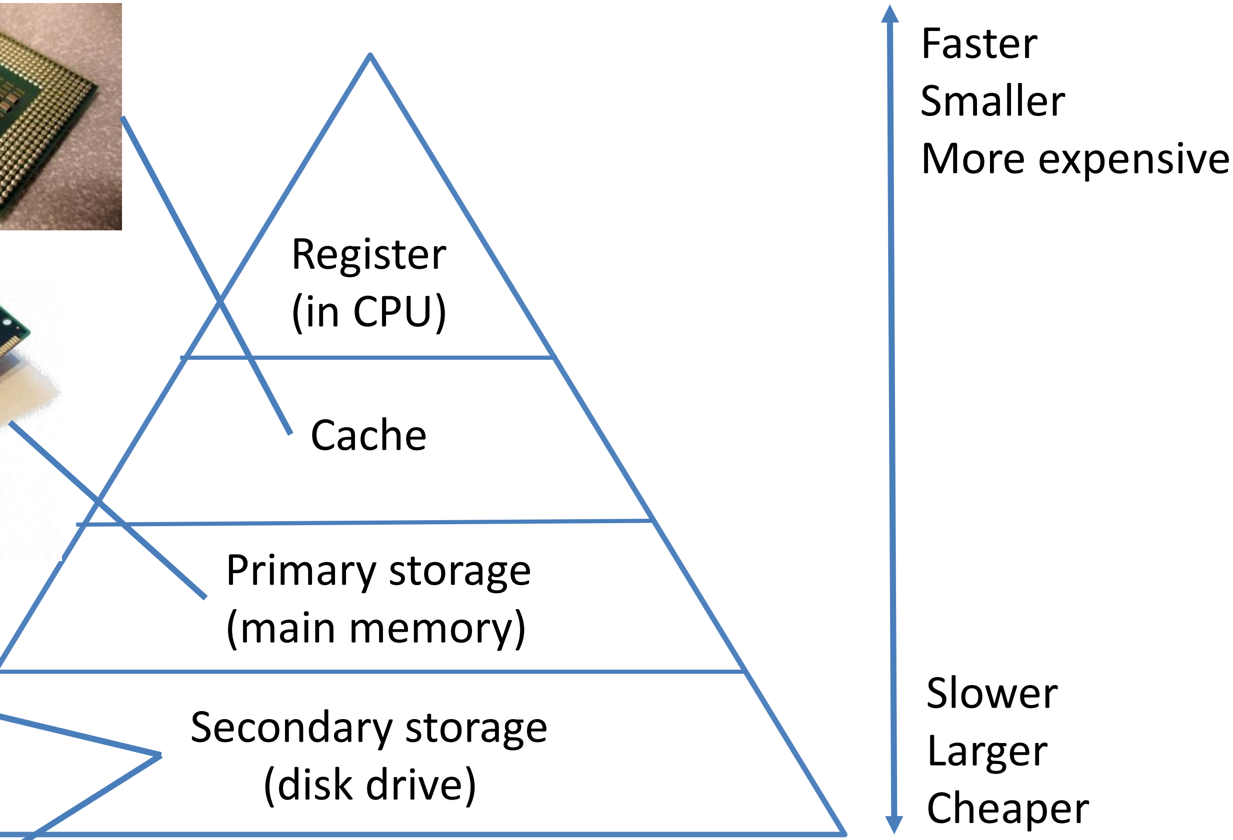
Storage Hierarchy: Devices

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

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



Spinning Drive



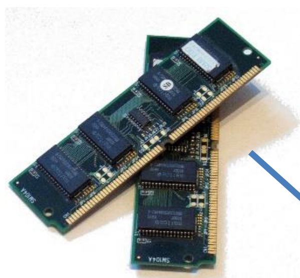
Solid State Drive (SSD)

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)

Register (in CPU)

Cache

Primary storage (main memory)

Secondary storage (disk drive)

Faster
Smaller
More expensive

Slower
Larger
Cheaper

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)

Command

- `free`: displays amount of free and used memory in the system
  - Syntax: `free -h`
- `lstopo`: shows the topology of the system
- `du` ("dist free"): shows the amount of available disk space
  - Syntax: `du -h`

LSU CENTER FOR COMPUTATION & TECHNOLOGY

Exercise

- Find out the total size of the main memory and how much is being used

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

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.

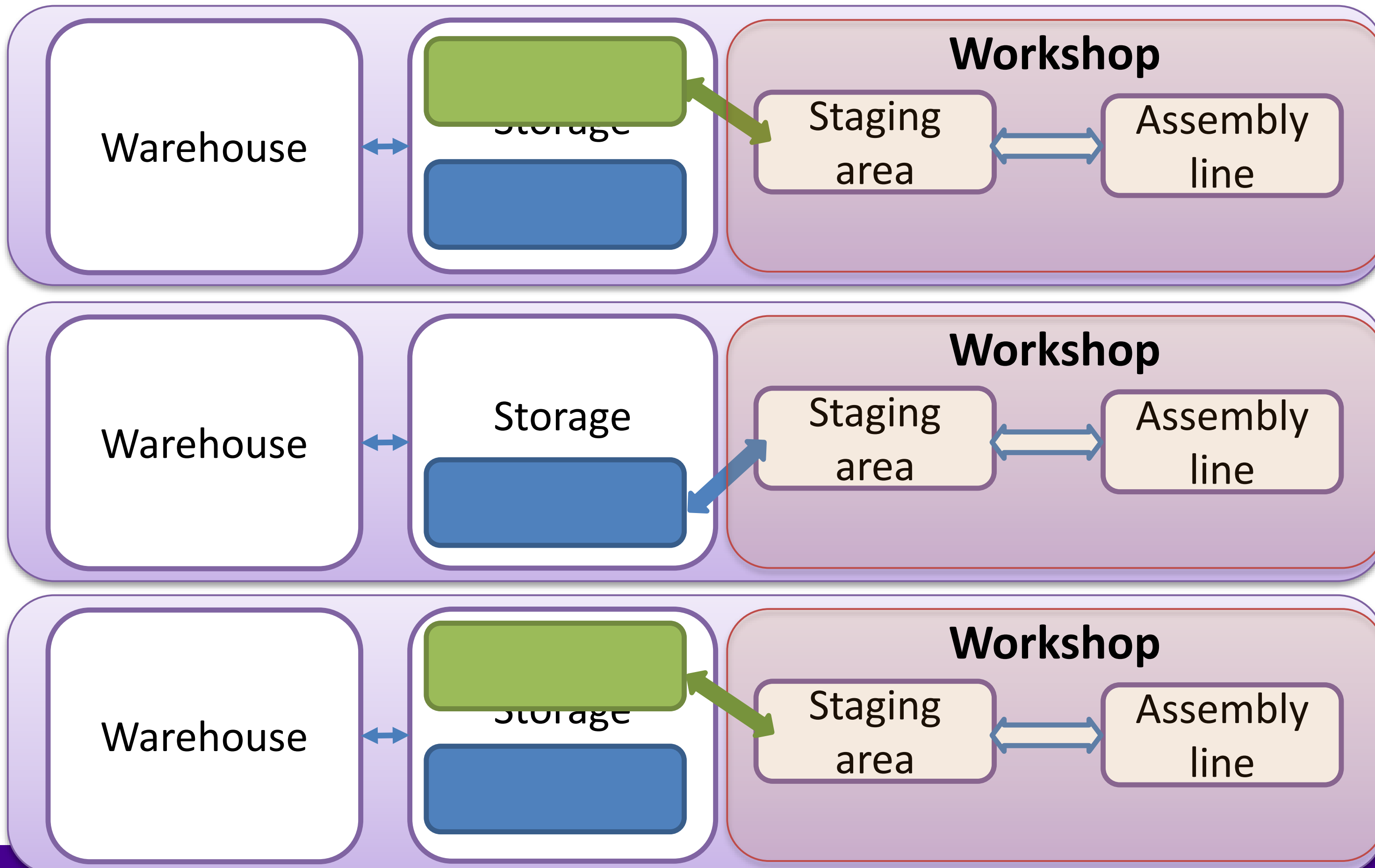
Producing A

Material for A is delayed

Producing B

Material for A is restored

Producing A



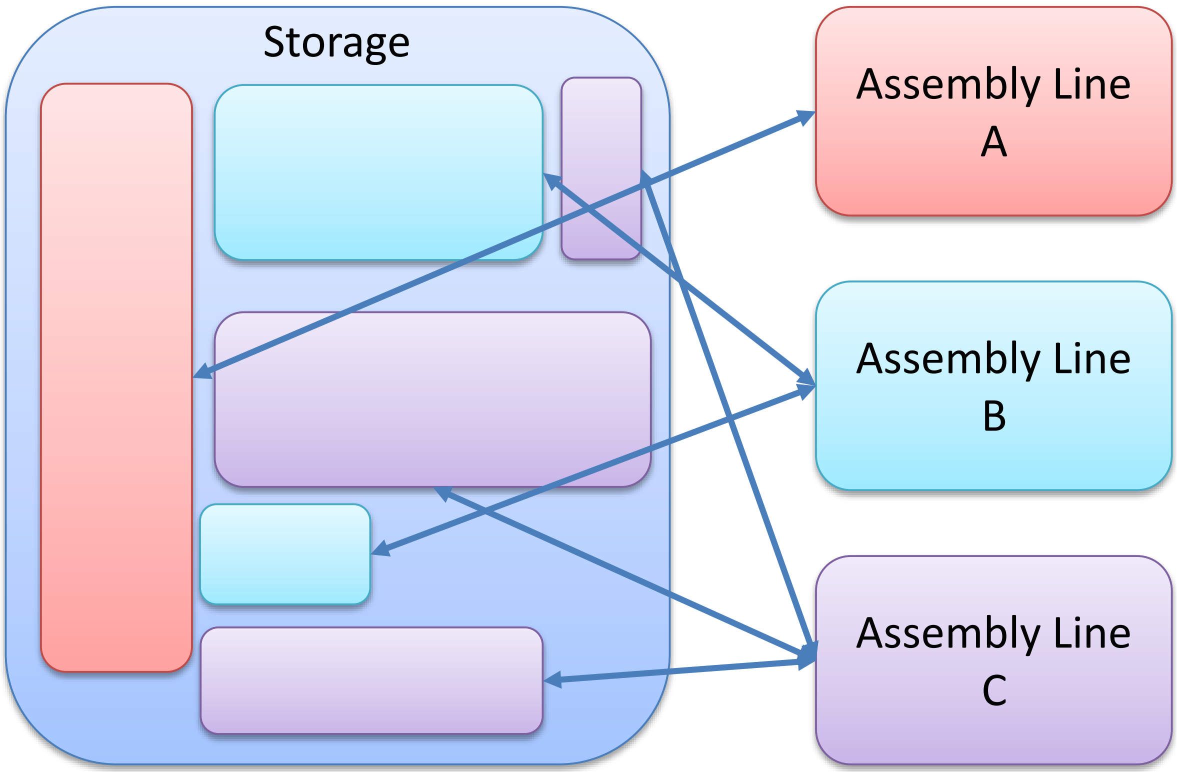
Thread

- 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

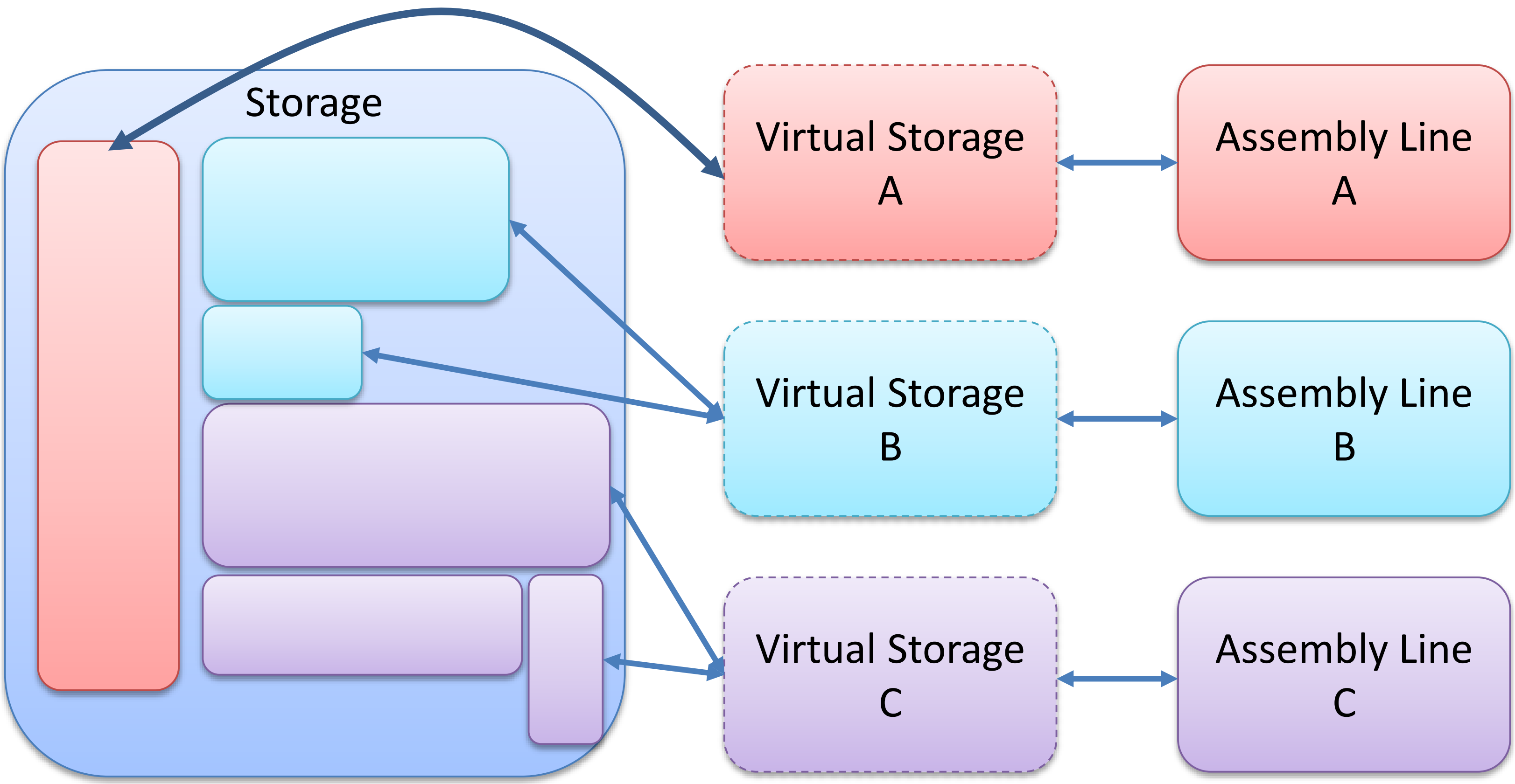
OS Memory Management: Virtual Memory (1)

- 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

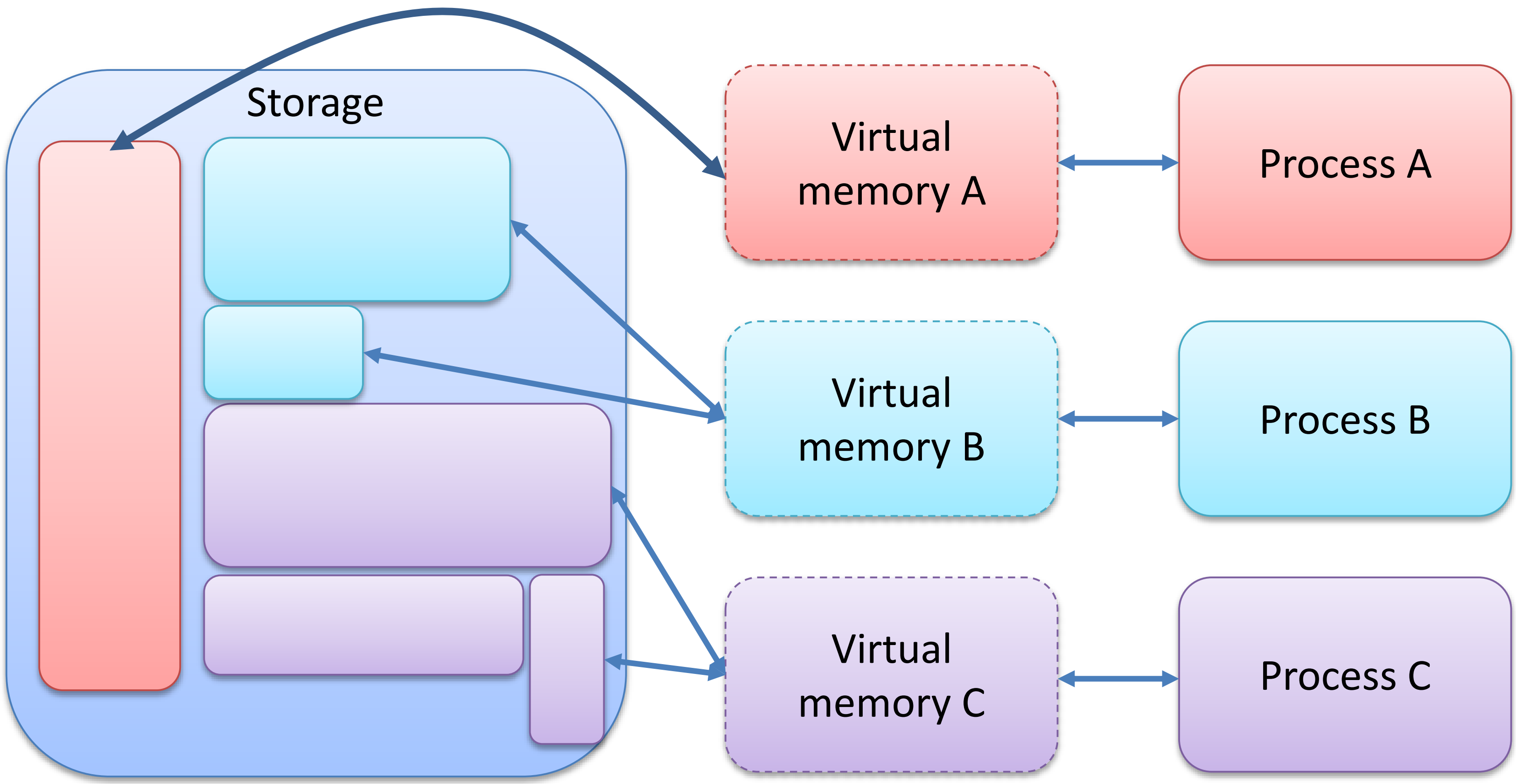
Without Virtual Memory



With Virtual Memory



With Virtual Memory



OS Memory Management: Swapping

- What if the computer is out of memory?

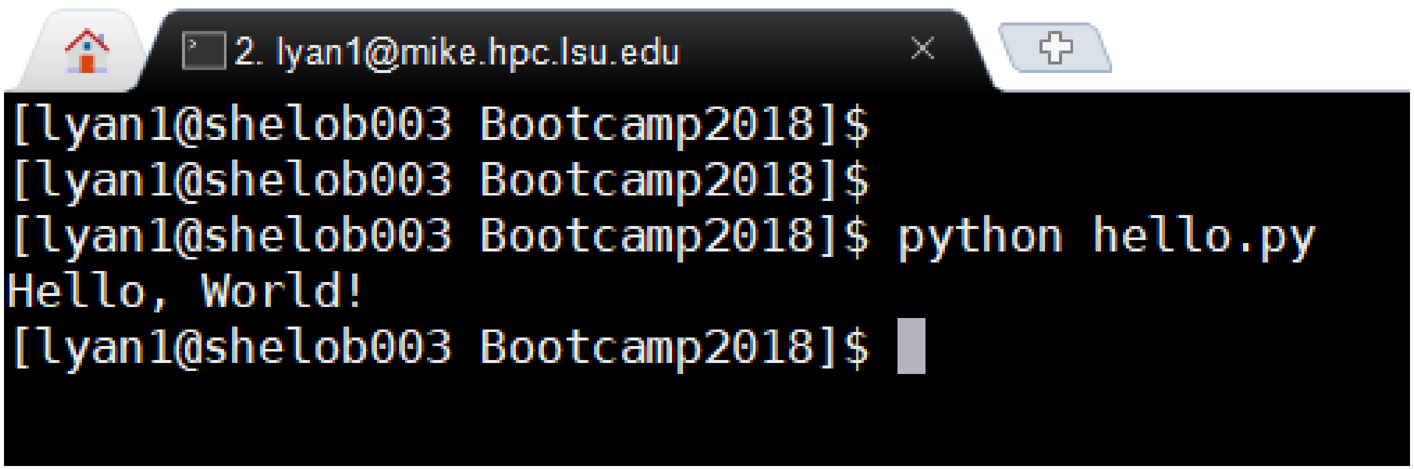
OS Memory Management: Swapping

- 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

CENTER FOR COMPUTATION & TECHNOLOGY

Operating Systems
Provide User Interface

- Shell is part of the operating system
  - Provide a interface for users to interact with the computer



Useful Commands for System Monitoring

- `top`: displays information about CPU and memory utilization.
  - Syntax: `top`
- `ps` ("process status"): displays the currently-running processes.
  - Syntax: `ps aux` or `ps -ef`

CENTER FOR COMPUTATION & TECHNOLOGY

Output from top

Each process has an unique ID

Memory and swap space utilization

Average CPU utilization for last 1, 5 and 15 minutes

```
top - 00:39:20 up 70 days, 14:44,  1 user,  load average: 0.00, 0.01, 0.00
Tasks: 684 total,   1 running, 683 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni, 99.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65945028k total, 26573424k used, 39371604k free,   164032k buffers
Swap: 134217724k total,    10188k used, 134207536k free, 21413936k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
17197 lyan1     20   0 27992 1860 1048 R  0.7  0.0   0:00.32 top
 2629 root      20   0     0    0    0 S  0.3  0.0  26:22.73 kondemand/1
 2630 root      20   0     0    0    0 S  0.3  0.0  29:03.62 kondemand/2
    1 root      20   0 33664 1292 1064 S  0.0  0.0   0:02.68 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.31 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:05.59 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:03.36 ksoftirqd/0
```

Owner of the process

Virtual memory used

Physical memory used

CPU and memory utilization for each process

Program name

Look for potential issues

Large swap space used?

Is there still free memory?

```
top - 00:39:20 up 70 days, 14:44,  1 user,  load average: 0.00, 0.01, 0.00
Tasks: 684 total,   1 running, 683 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni, 99.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65945028k total, 26573424k used, 39371604k free,   164032k buffers
Swap: 134217724k total,    10188k used, 134207536k free, 21413936k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
17197 lyan1     20   0 27992 1860 1048 R  0.7  0.0   0:00.32 top
 2629 root      20   0     0    0    0 S  0.3  0.0  26:22.73 kondemand/1
 2630 root      20   0     0    0    0 S  0.3  0.0  29:03.62 kondemand/2
    1 root      20   0 33664 1292 1064 S  0.0  0.0   0:02.68 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.31 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:05.59 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:03.36 ksoftirqd/0
```

Load average too high or too low? Normal range should be 0 to <number of cores>.

Exercise

- Find out the process ID of Jupyter Notebook.
- Find out how much virtual and physical memory Jupyter Notebook uses.

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

Files

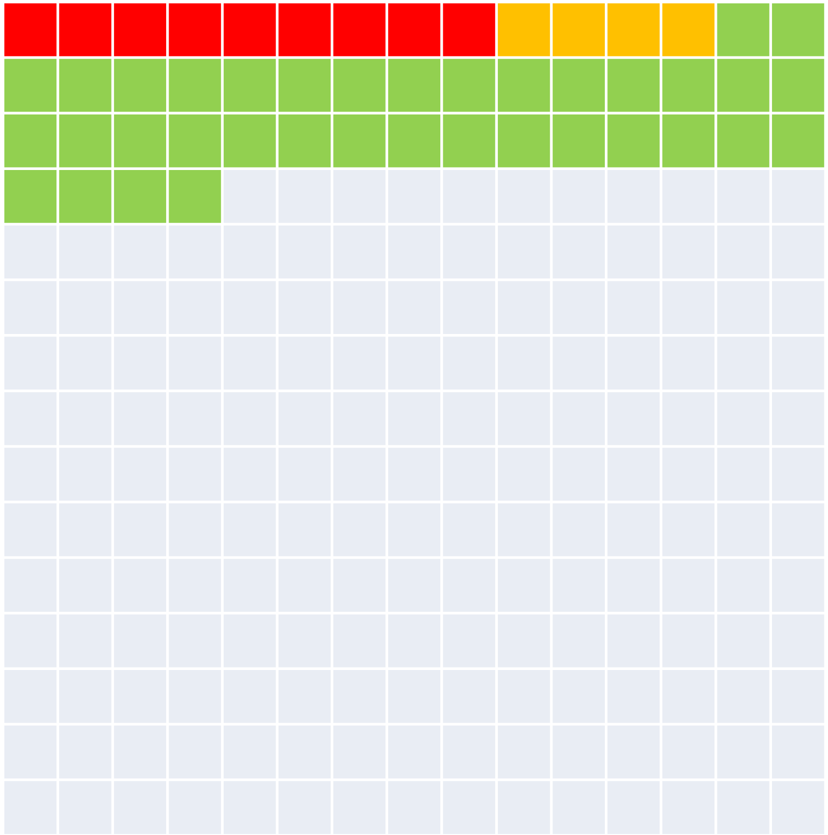
- A sequence of bytes
  - An abstraction used by OS to represent all I/O devices and to present a uniform view to applications
  - Allows the development of portable applications without knowing details of underlying technology
- From the OS point of view, files include
  - Normal files on disks
  -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

How Files Are Stores on Disks

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

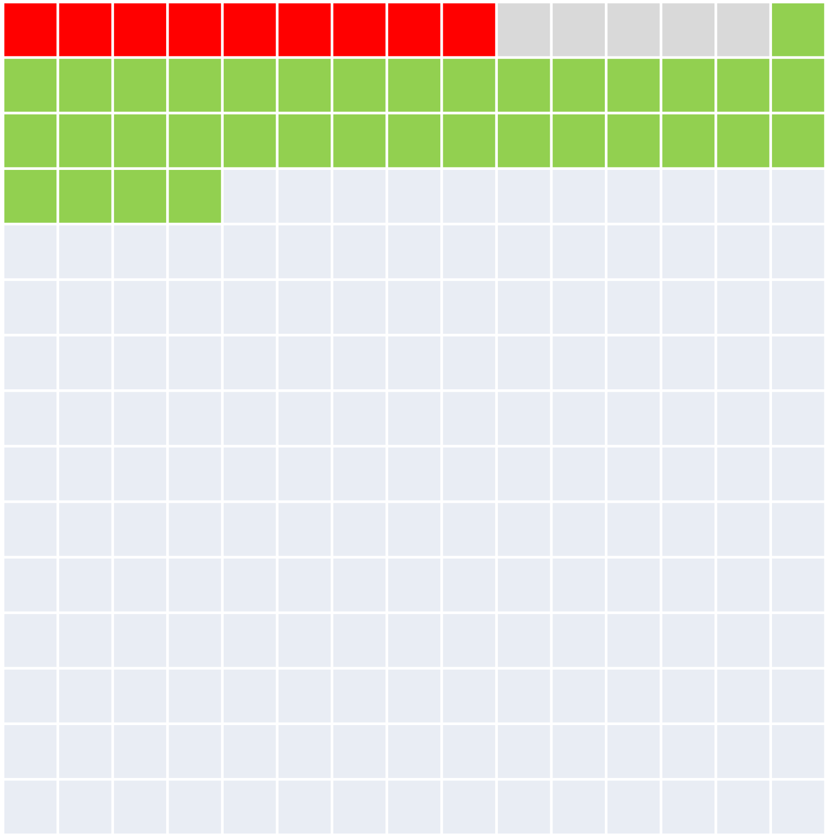
How Files Are Stores on Disks

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



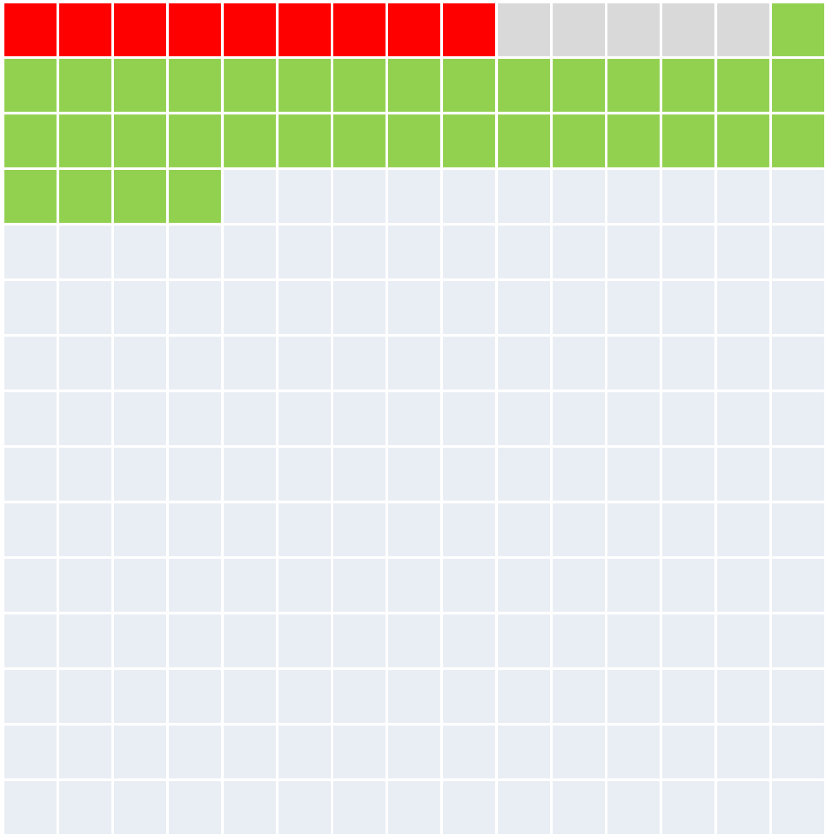
How Files Are Stores on Disks

- Then the yellow file was removed



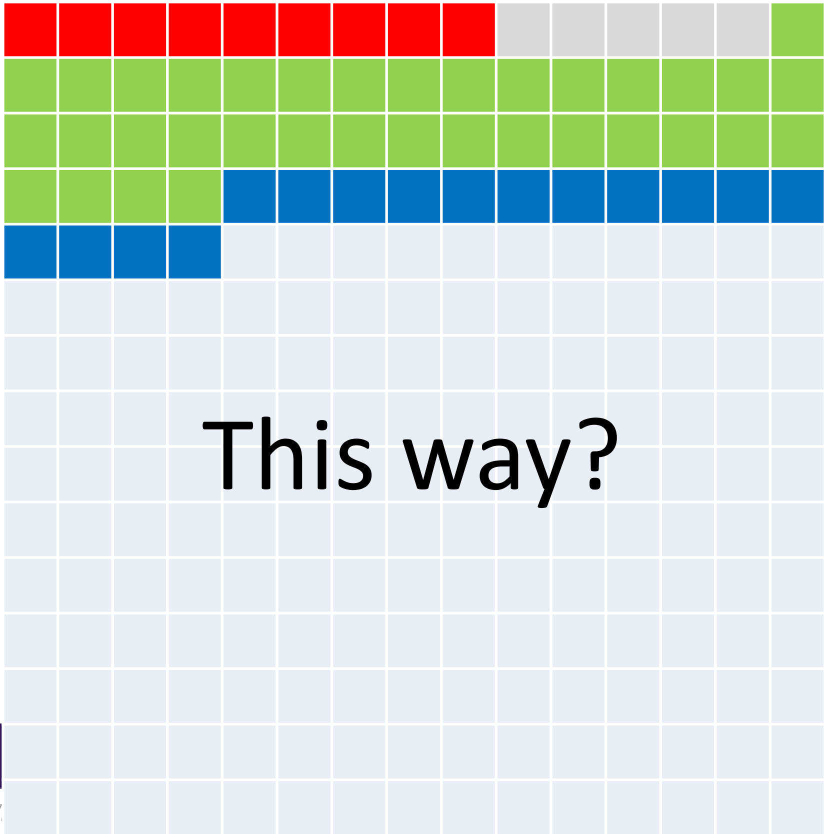
How Files Are Stores on Disks

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



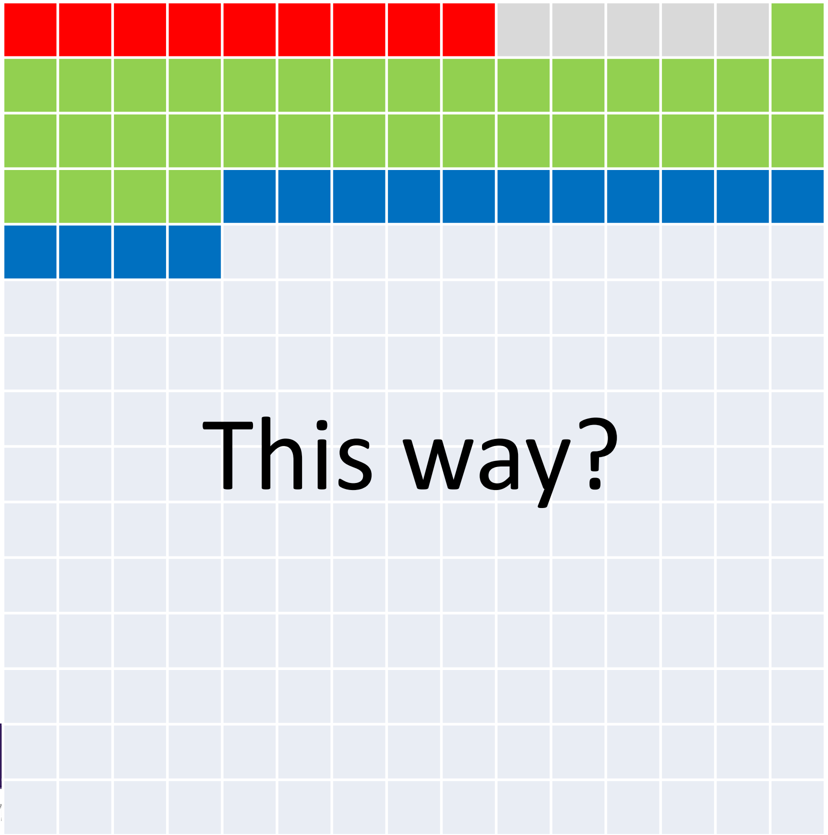
How Files Are Stores on Disks

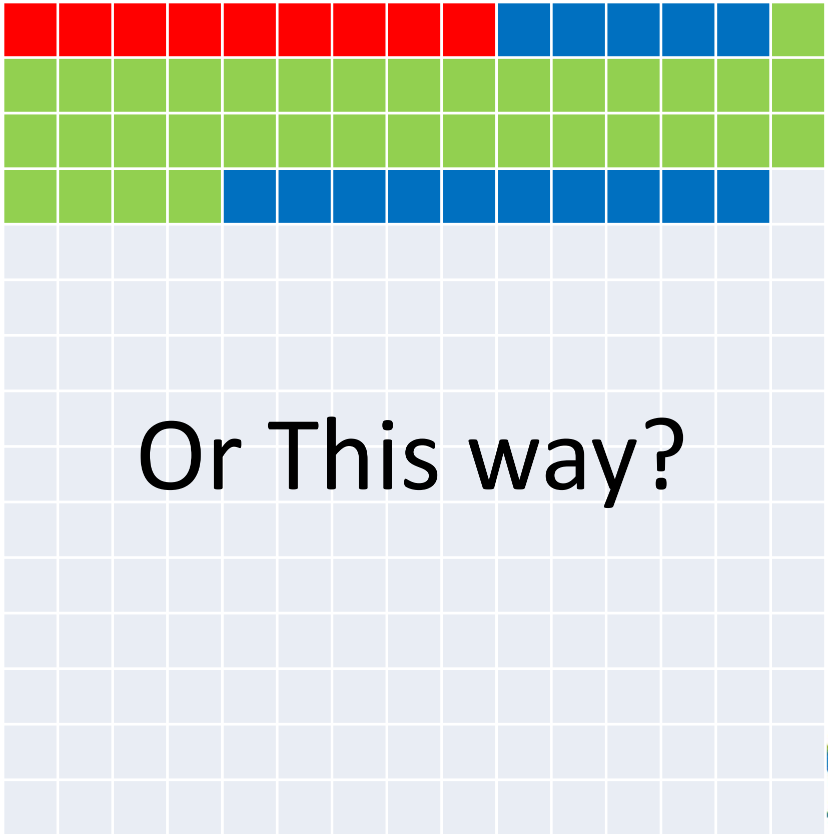
- Now, what would happen if another sequence of 15 bytes need to be written to the disk?



How Files Are Stores on Disks

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



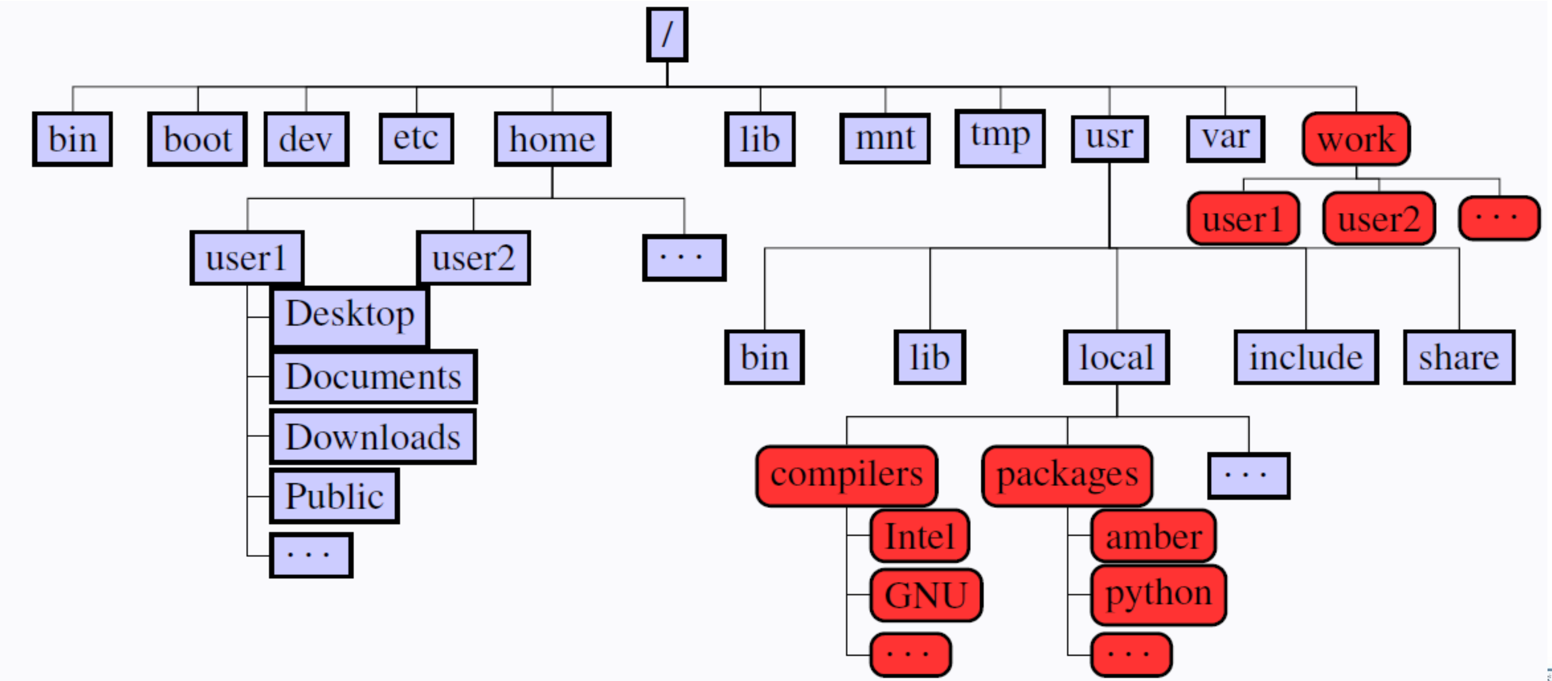


File Systems

- 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 file system simply erase its records and the space it occupies does not reset to a “empty” state
- Performance consideration: bookkeeping itself takes time and resources
  - E.g. not a good idea to put 1 million small files in the same directory.

Directory Tree

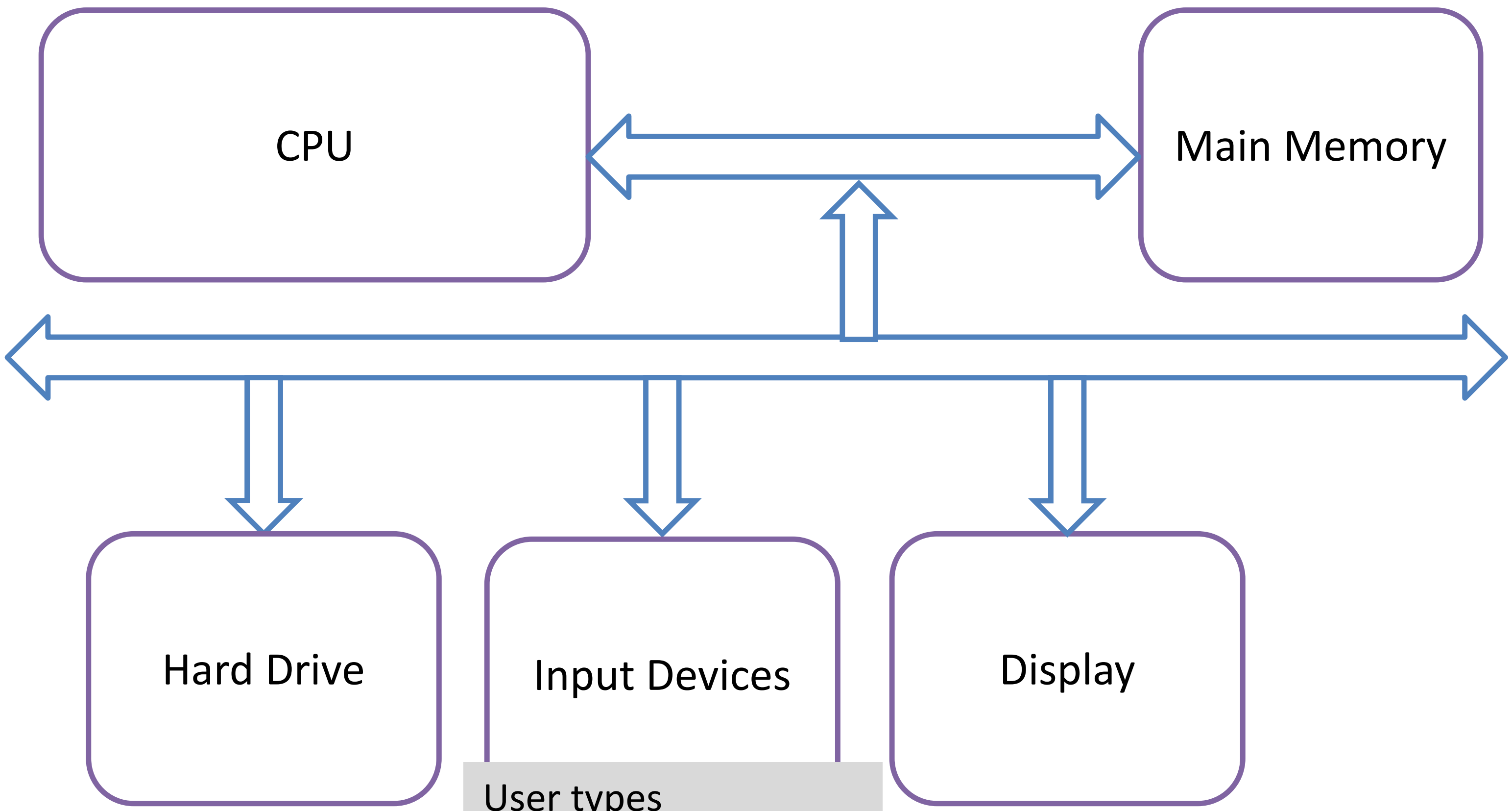
- Files are organized in an inverted tree structure, aka "directory tree"



Path

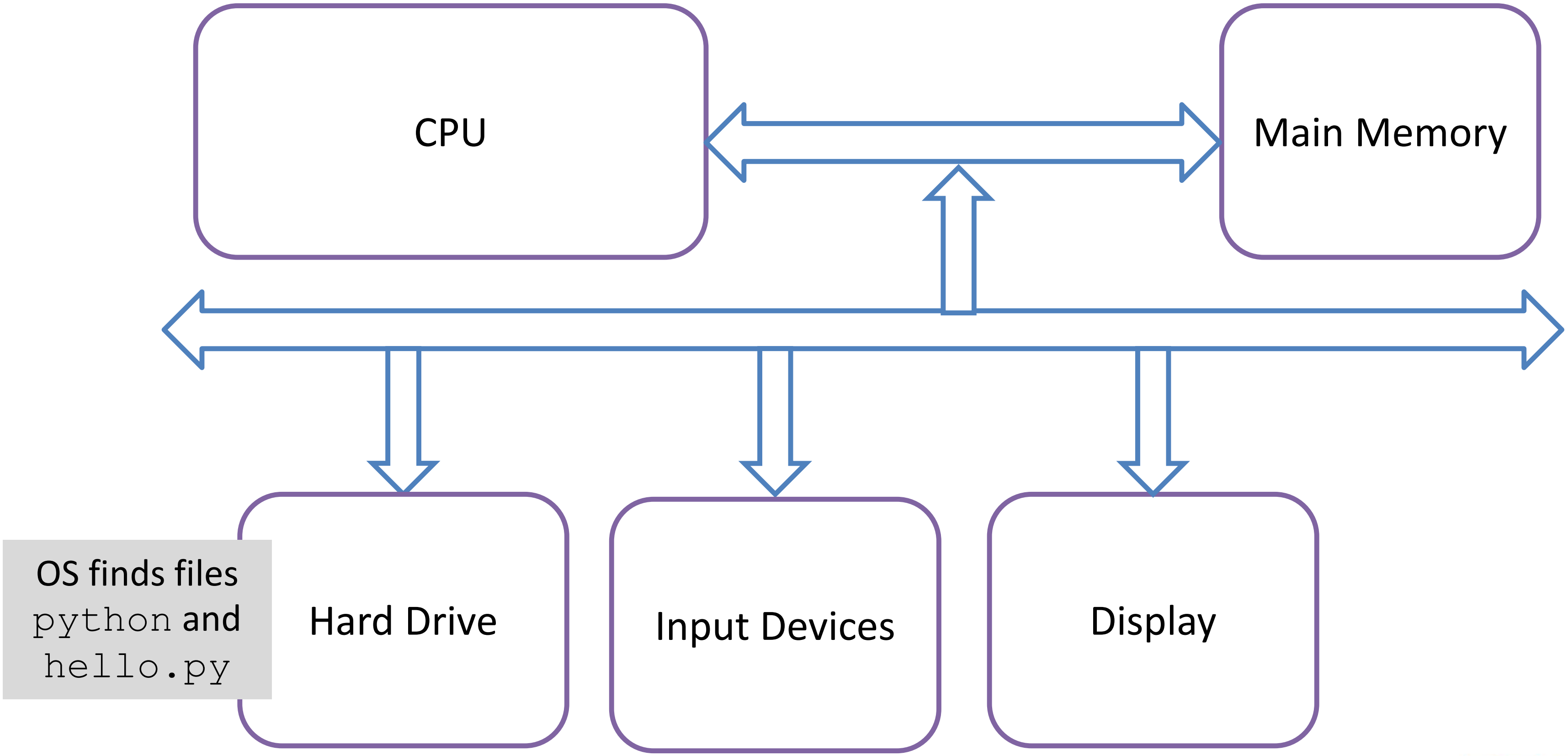
- Path indicates a location in the directory tree
  - In Linux, "/" denotes the "root" directory
- Absolute path
  - Paths starting with "/" are defined uniquely and does NOT depend on the current working directory
    - Example: `/home/lyan1` is unique
- Relative path
  - Paths not starting with "/" are relative to current working directory
  - They are not unique
    - "`lbrnloni2018`" is not unique: it points to "`/home/lyan1/lbrnloni2018`" if current working directory is "`/home/lyan1`"; if current working directory is "`/work/lyan1`", then it points to "`/work/lyan1/lbrnloni2018`"
  - Shortcuts
    - `.` (single dot) is the current working directory
    - `..` (double dots) is one directory up

Hello World Revisited: What Happened

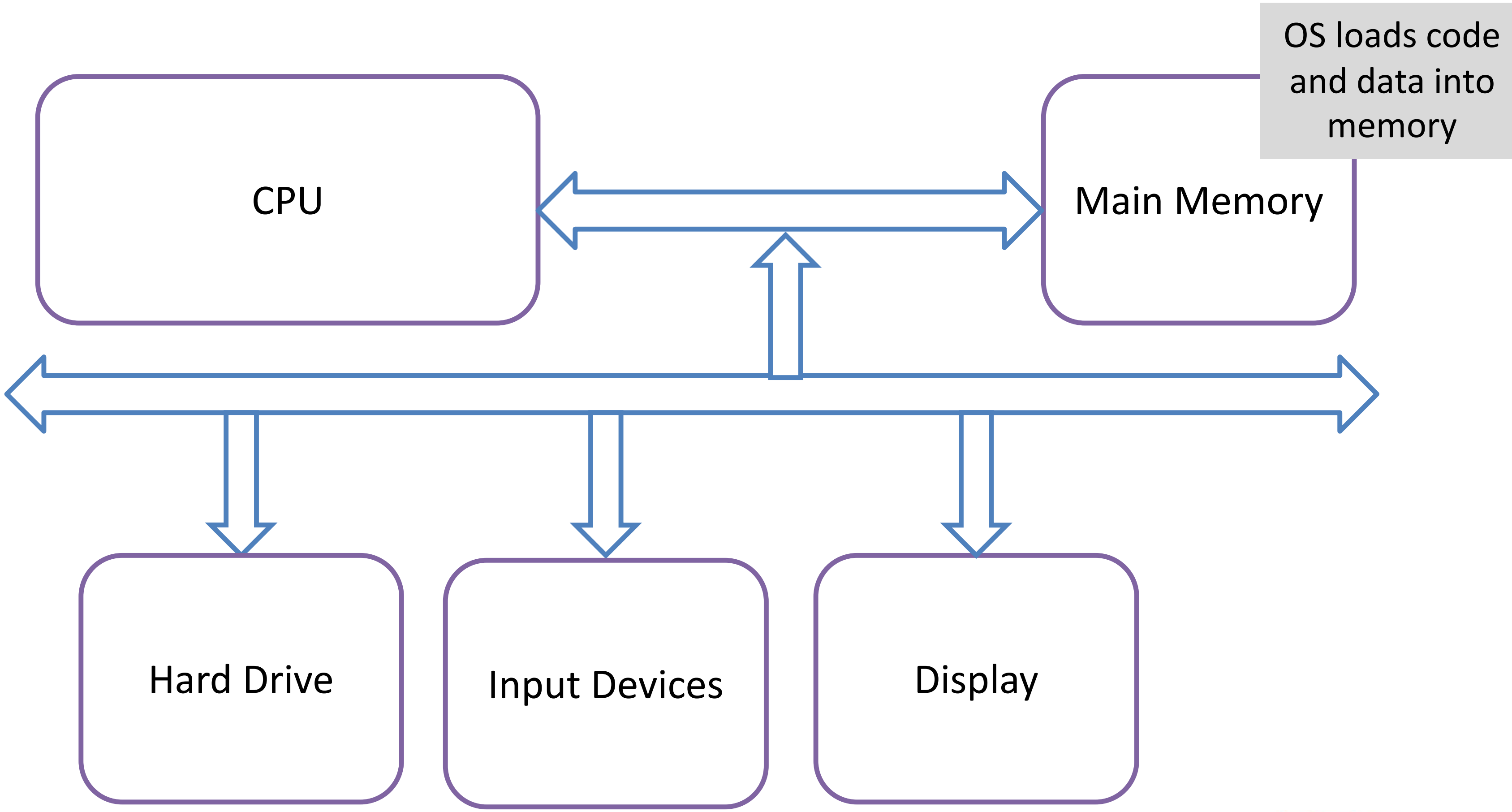


User types
`python hello.py`

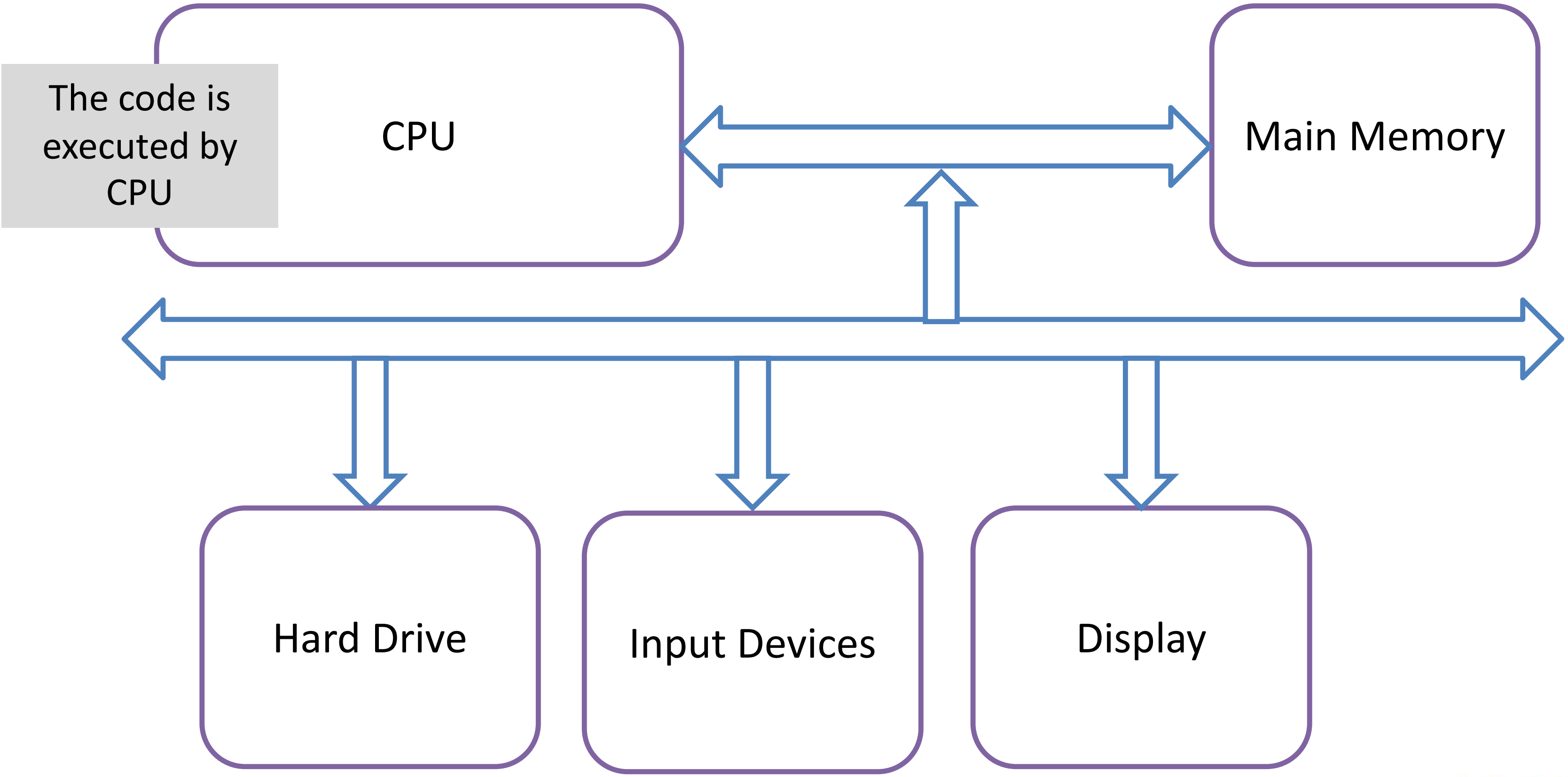
Hello World Revisited: What Happened



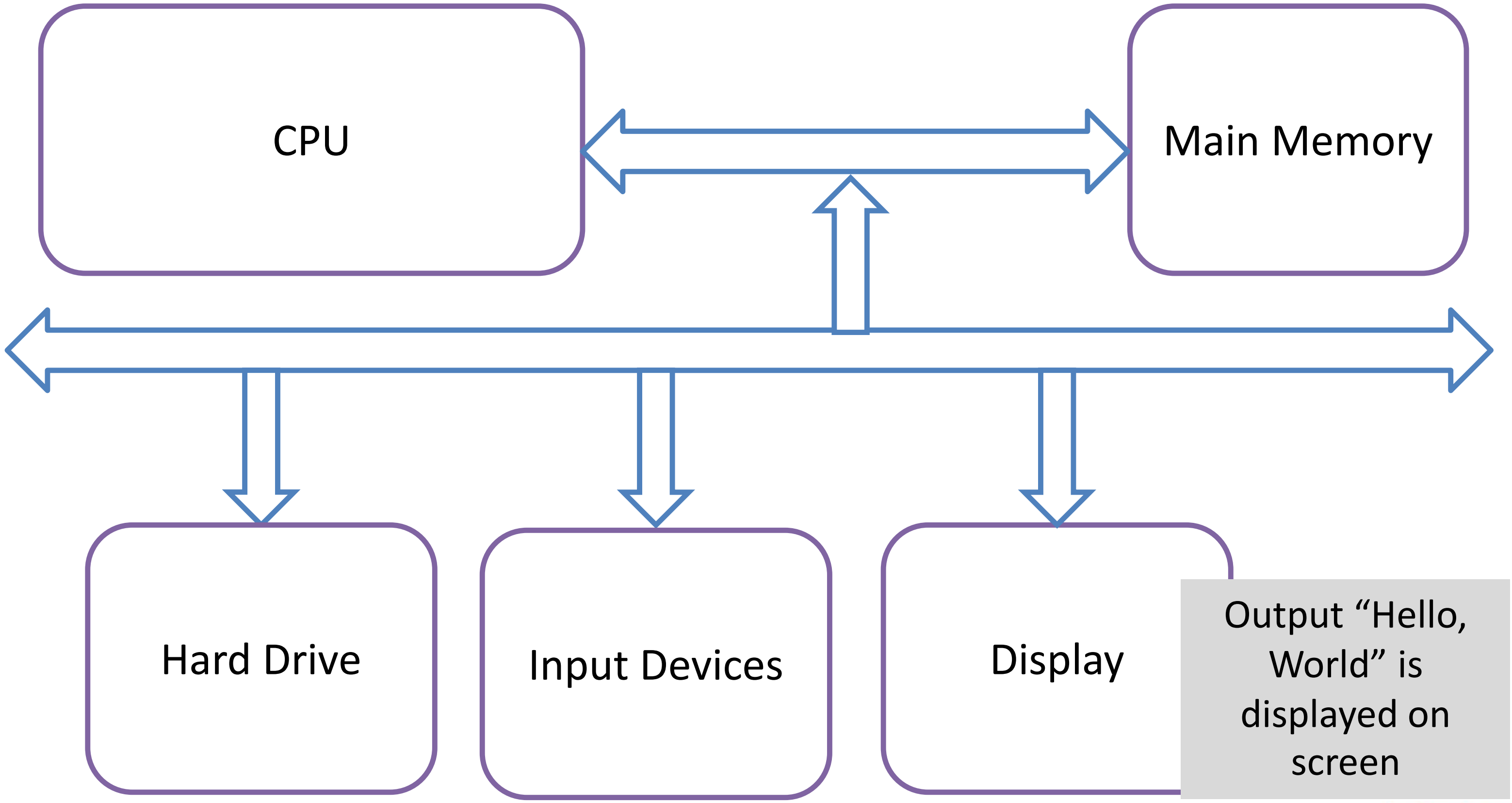
Hello World Revisited: What Happened



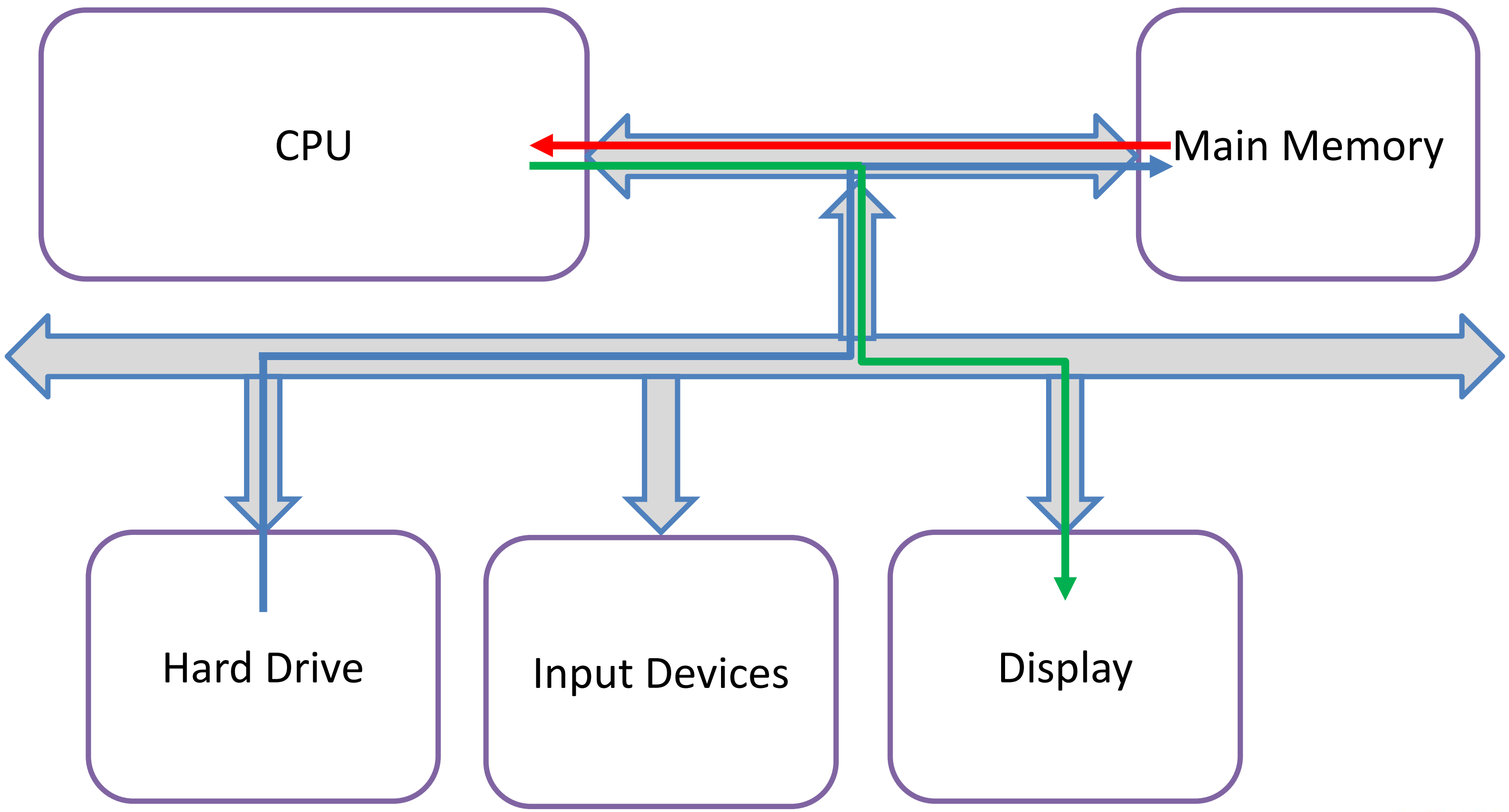
Hello World Revisited: What Happened



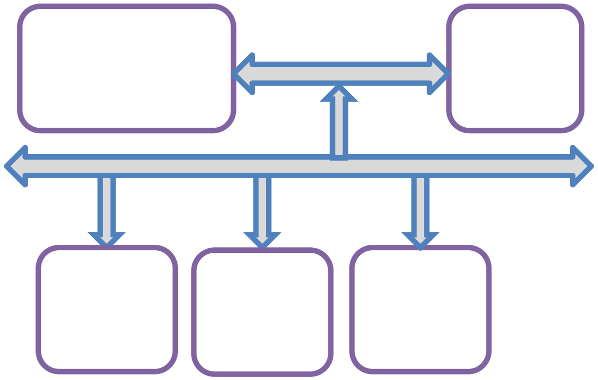
Hello World Revisited: What Happened



Data Transfer



Data Transfer 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

Useful Commands (1)

- `pwd`: displays current working directory
  - Syntax: `pwd`
- `cd`: changes current working directory
  - Syntax: `cd <path to directory>`
- `mkdir`: creates a directory
  - Syntax: `mkdir <path to directory>`
- `cp`: copies files or directories
  - Use “-r” flag when copying directories
  - Syntax: `cp [-r] <path to source> <path to destination>`

CENTER FOR COMPUTATION & TECHNOLOGY

Useful Commands (2)

- `rm`: removes files and directories
  - Syntax: `rm [-r] <path to file>`
- `mv`: moves files and directories
  - Can be used to rename files as well
  - Syntax: `mv <source path> <destination path>`
- `du`: displays disk usage information
  - Syntax: `du -h [<path to directory>]`

Exercise

- At the command prompt, `cd` to directory "`/work/<user name>`"
  - Ex: if your user name is "hpctrn011", the directory should be "`/work/hpctrn011`"
- Create a new directory "`bowtie2_yeast`" and cd into it
- Copy the following two files to your new directory
  - `/work/lyan1/Bootcamp2018/yeast/yeast_ref.fa`
  - `/work/lyan1/Bootcamp2018/yeast/read1.fastq`
- Find out how much space they occupy

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

How Do We Improve Performance?

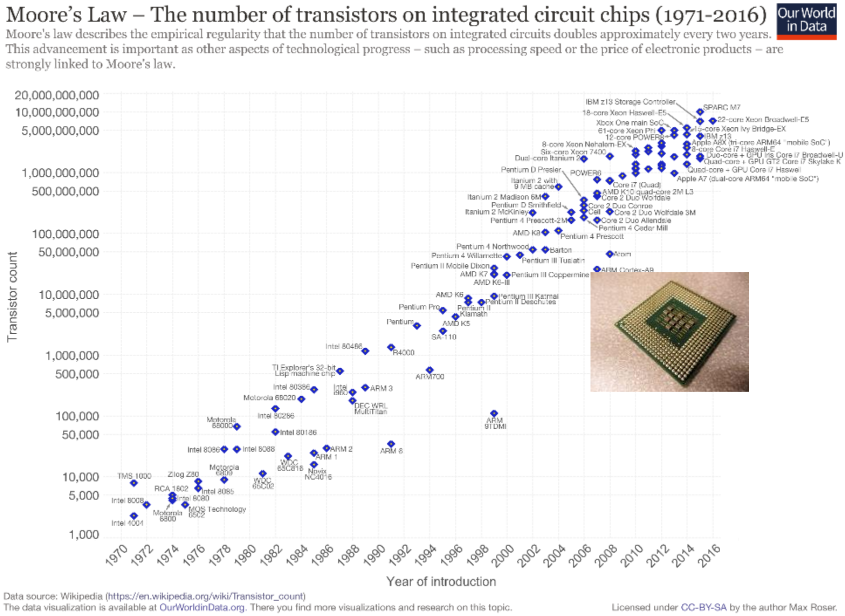
- Better performance = produce more product in a given period of time
- How to get better performance?

How Do We Improve Performance?

- Better performance = produce more product in a given period of time
- How to get better performance?
  - Faster processing
  - More processing units

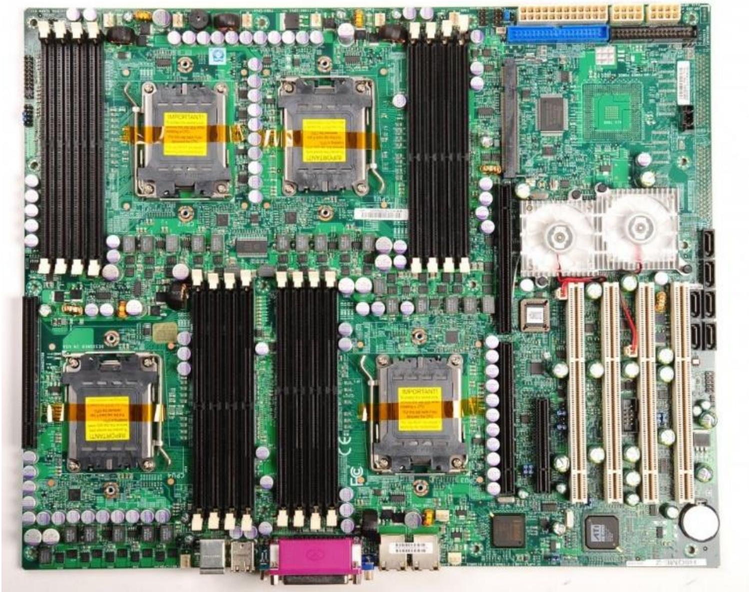
Moore's Law

- The "faster processing" route
- Moore's Law
  - Number of transistors on a CPU chip doubles every 18 months
    - Means more clock cycles per second
  - 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

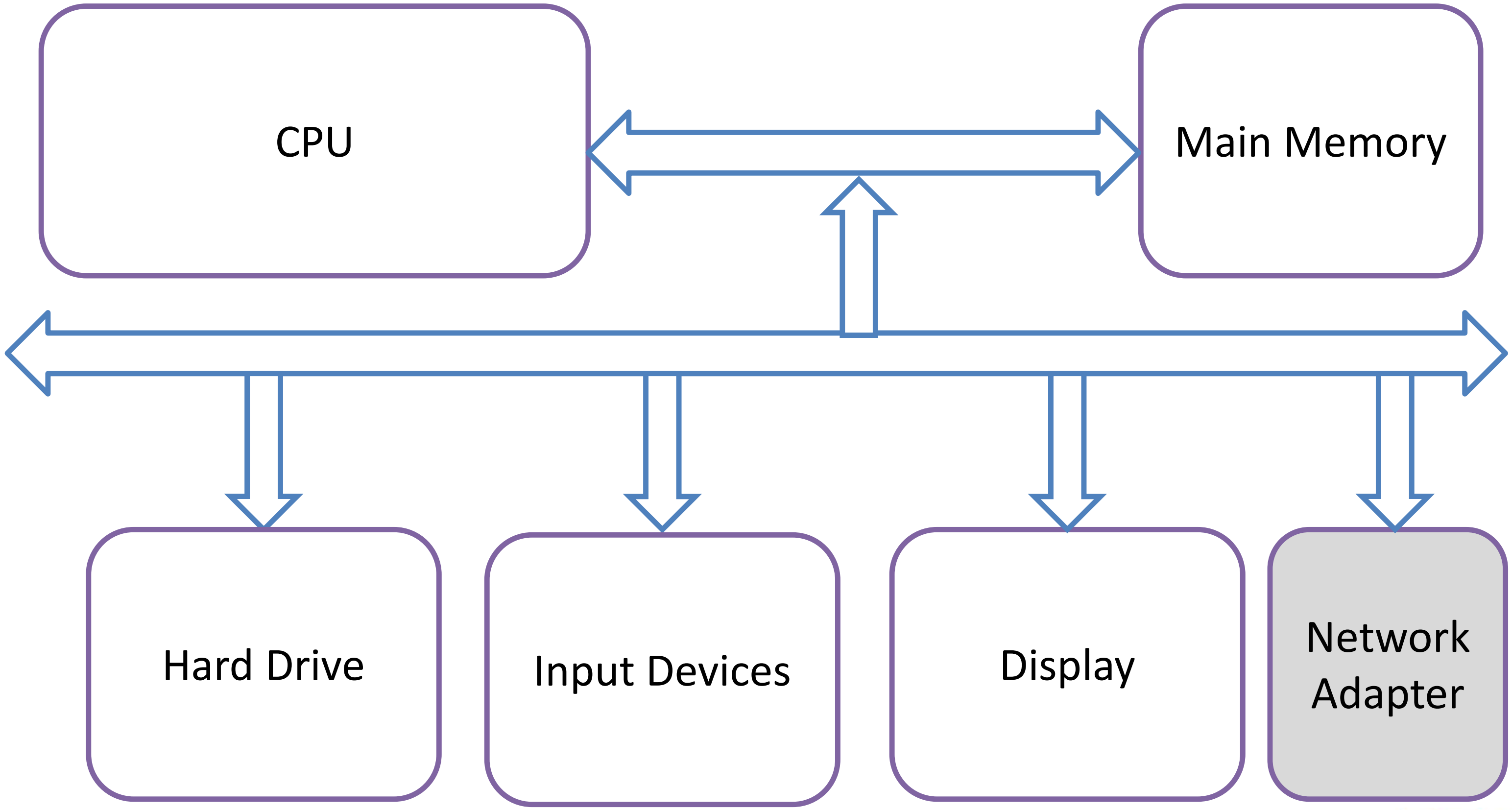


At the `top` output screen, press "`1`" to display load on each CPU core

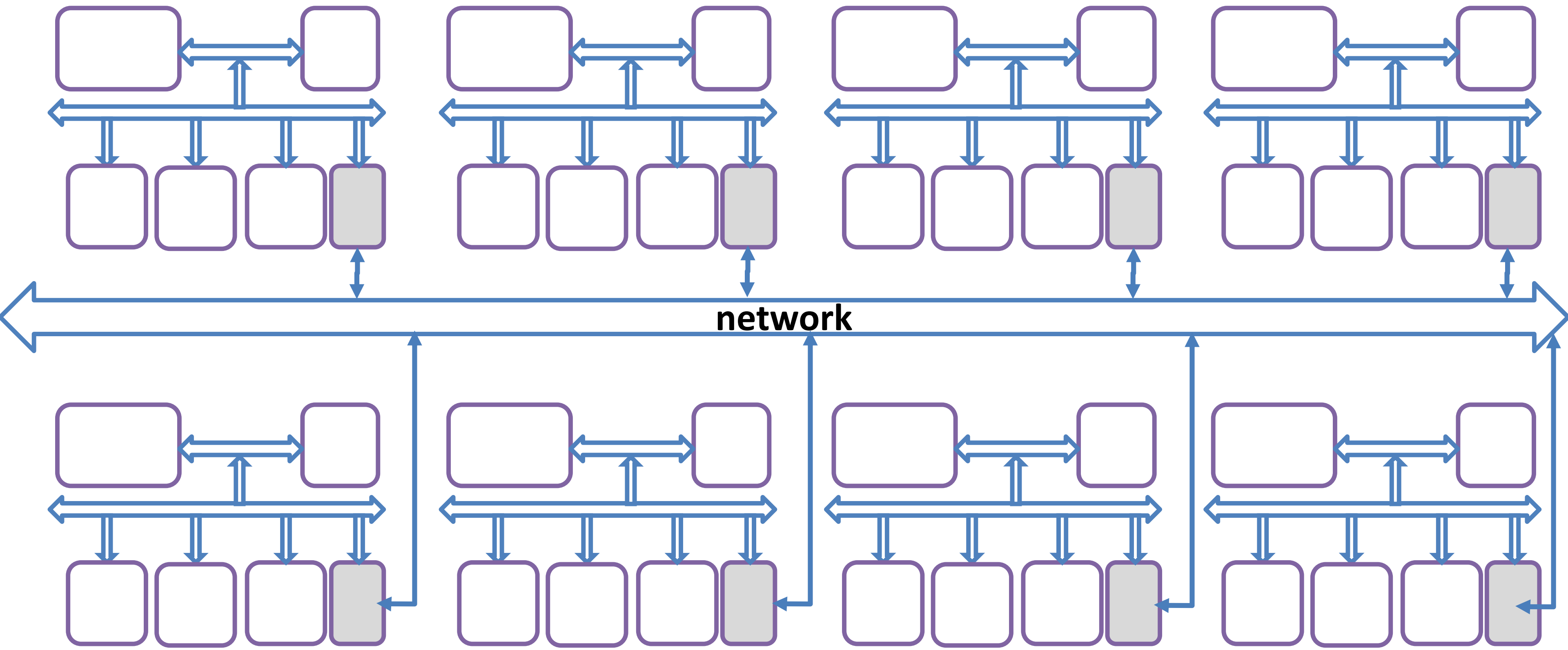
```
top - 16:42:59 up 46 days,  7:53,  0 users,  load average: 0.20, 0.10, 2.71
Tasks: 671 total,   1 running, 670 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 98.0%us,  1.7%sy,  0.0%ni,  0.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
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,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  :  0.0%us,  0.7%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 98.7%us,  1.3%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
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
```

Two cores are busy in this example.

Extend Parallelism over Network

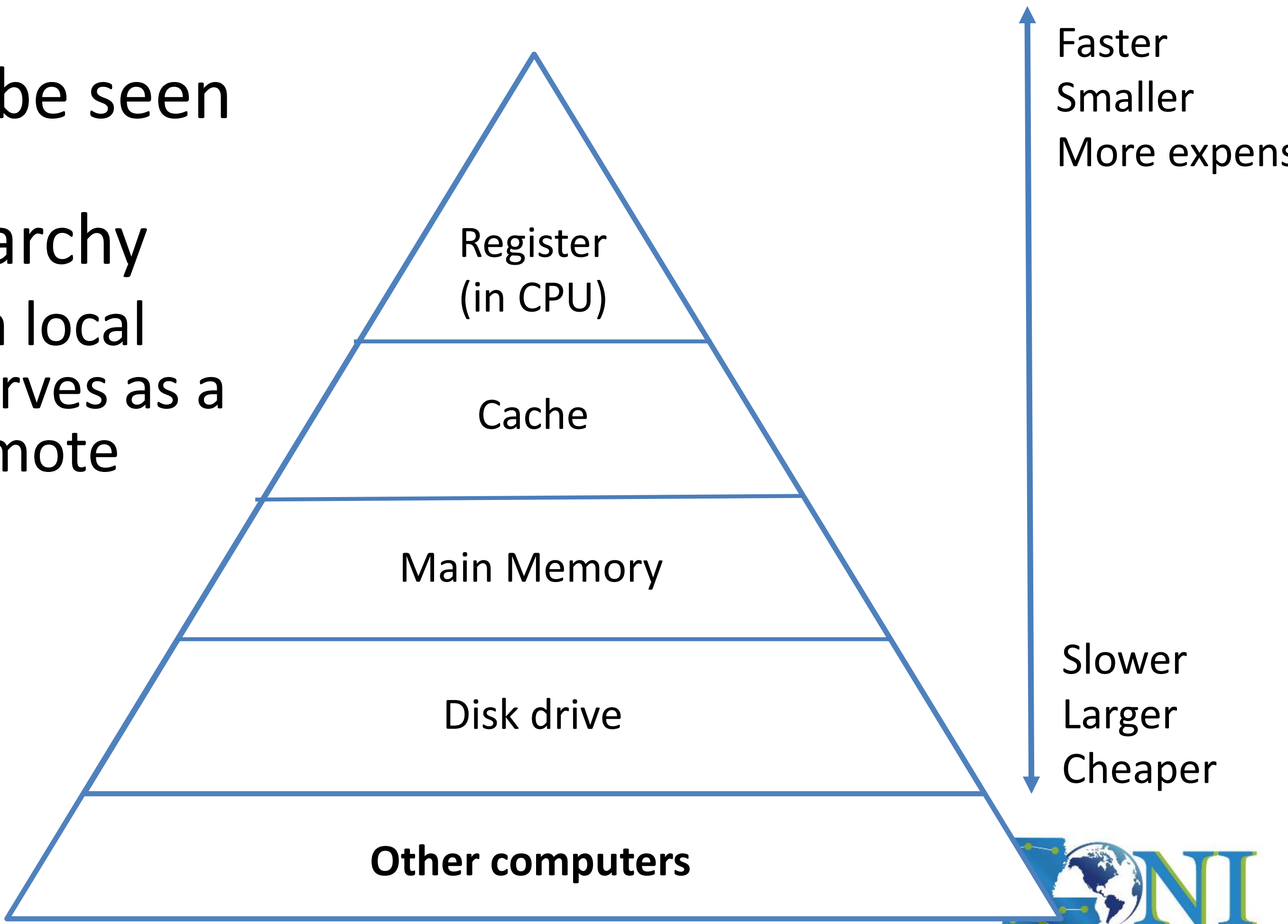


Extend Parallelism over Network



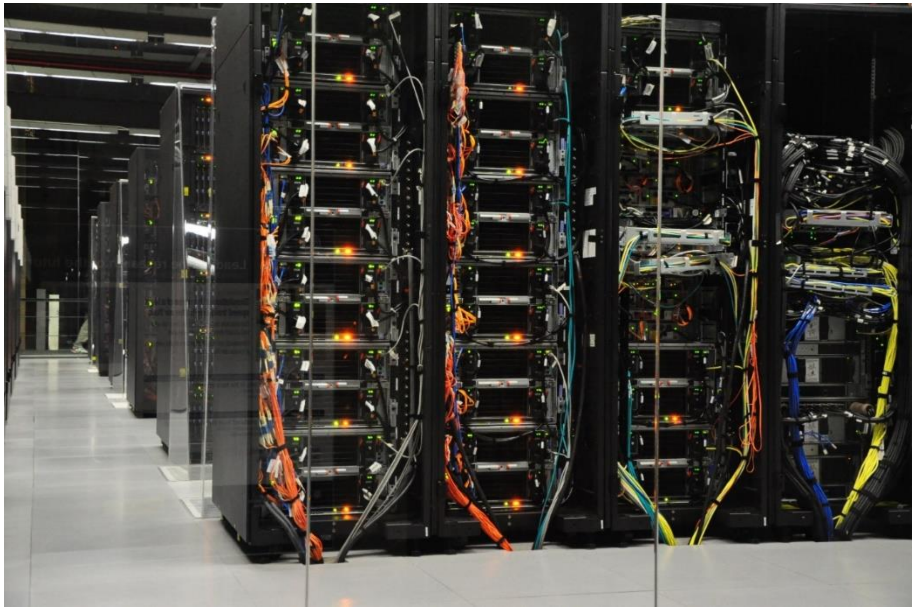
Memory Hierarchy (Updated)

- Network can be seen as part of the memory hierarchy
  - Disk drive on local computer serves as a cache for remote systems



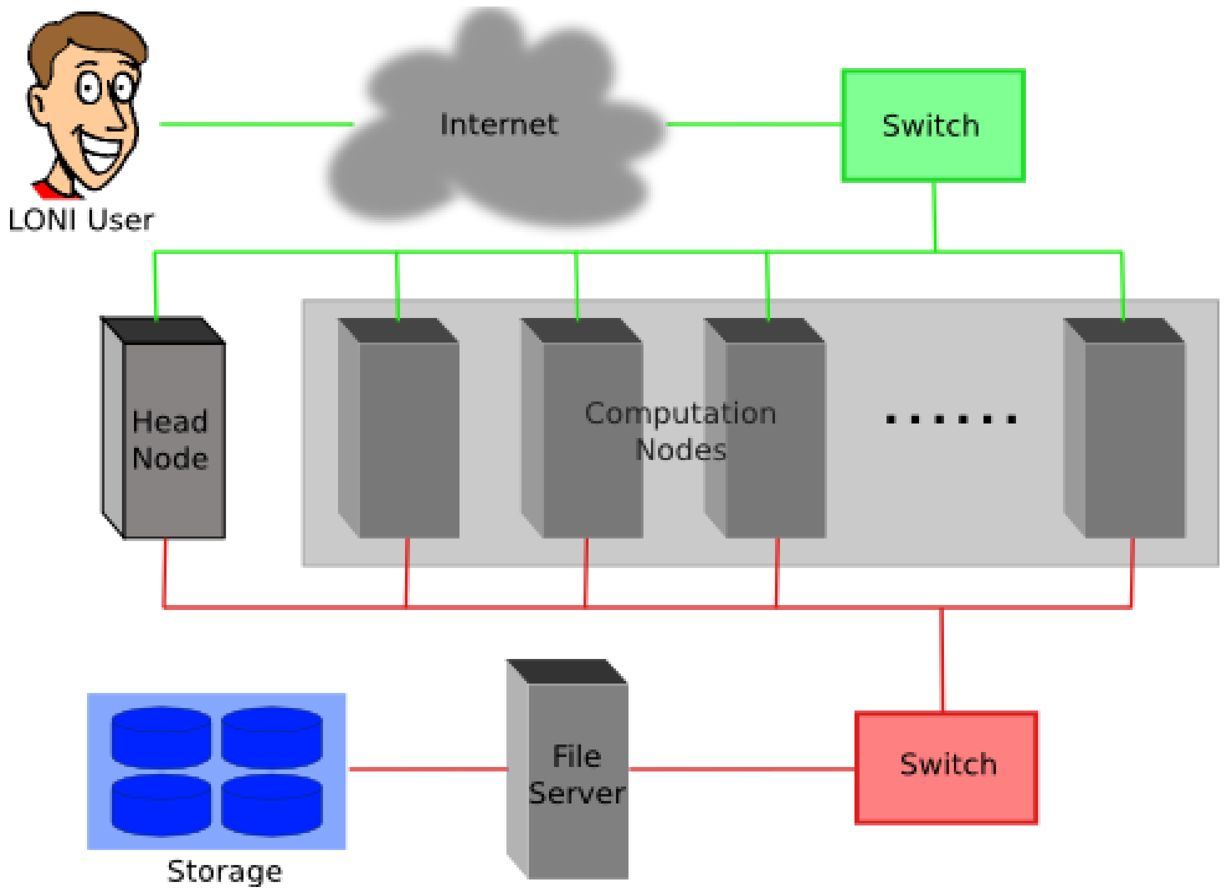
Extend Parallelism over Network: Clusters

- The majority of supercomputers are clusters
  - Individual servers connected by high speed network
  - Servers are known as “nodes” in a cluster
  - They server different purposes
    - Login node
    - Compute node
    - I/O node



CENTER FOR COMPUTATION & TECHNOLOGY

“Super Mike 2” Cluster



```
active jobs------------------------
JOBID               USERNAME      STATE PROCS   REMAINING            STARTTIME

874220               ahar121    Running    32    00:01:26  Wed May 23 17:03:00
874219               ahar121    Running    32    00:01:43  Wed May 23 17:03:17
875714                 lyan1    Running    16     1:38:25  Sat May 26 16:39:59
875646               sbhask3    Running    16     2:00:08  Sat May 26 09:01:42
875647               sbhask3    Running    16     2:00:30  Sat May 26 09:02:04
875648               sbhask3    Running    16     2:02:06  Sat May 26 09:03:40
875649               sbhask3    Running    16     2:02:28  Sat May 26 09:04:02
875651               sbhask3    Running    16     2:02:56  Sat May 26 09:04:30
875666               sbhask3    Running    32     6:29:55  Sat May 26 13:31:29
875667               sbhask3    Running    16     6:30:25  Sat May 26 13:31:59
874342               ahar121    Running    32     7:33:52  Thu May 24 00:35:26
874343               ahar121    Running    32     7:37:10  Thu May 24 00:38:44
875623               liangjh    Running   512     9:58:16  Fri May 25 18:59:50
875671               batmanu    Running    32    21:06:03  Sat May 26 14:07:37
875041               rosalyn    Running    16    23:51:13  Thu May 24 16:52:47
875458               karunya    Running    32  1:06:36:20  Fri May 25 11:37:54
875192               ahar121    Running    32  1:07:48:09  Fri May 25 00:49:43
875194               stiwar8    Running   128  1:08:05:51  Fri May 25 01:07:25
875196                sbale1    Running   256  1:08:45:49  Fri May 25 01:47:23
875206                 karki    Running    32  1:15:26:48  Fri May 25 08:28:22
875209                 karki    Running    32  1:15:35:55  Fri May 25 08:37:29
875210                 karki    Running    32  1:15:43:35  Fri May 25 08:45:09
875211                 karki    Running    32  1:15:47:18  Fri May 25 08:48:52
875235              sbshakya    Running    16  1:16:02:35  Fri May 25 09:04:09
875342               rosalyn    Running    16  1:17:38:49  Fri May 25 10:40:23
875392               rosalyn    Running    16  1:17:43:49  Fri May 25 10:45:23
875396               rosalyn    Running    16  1:17:49:28  Fri May 25 10:51:02
```

Outline

- What do computers do?
- Components of computing systems
  - Data
  - CPU
  - Storage hierarchy
  - Operating systems
  - Files
  - Parallel processing
- Putting it together

Putting It Together: A Case Study

- Apply the knowledge and tools you learned during the boot camp to the management your scientific computing workflow
- See project “Sequence Alignment with Bowtie2” for details