

# [544] Intro to Big Data Systems

Tyler Caraza-Harter

# Outline

## Course Overview

- **Introductions**
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

## Resources

- Overview
- Compute
- Memory
- Storage
- Network

## Deployment

# Introductions

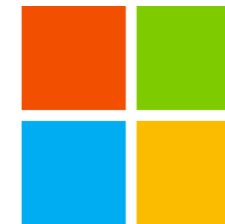
Tyler Caraza-Harter

- Long time Badger
- Email: [tharter@wisc.edu](mailto:tharter@wisc.edu)
- Just call me “Tyler” (he/him)



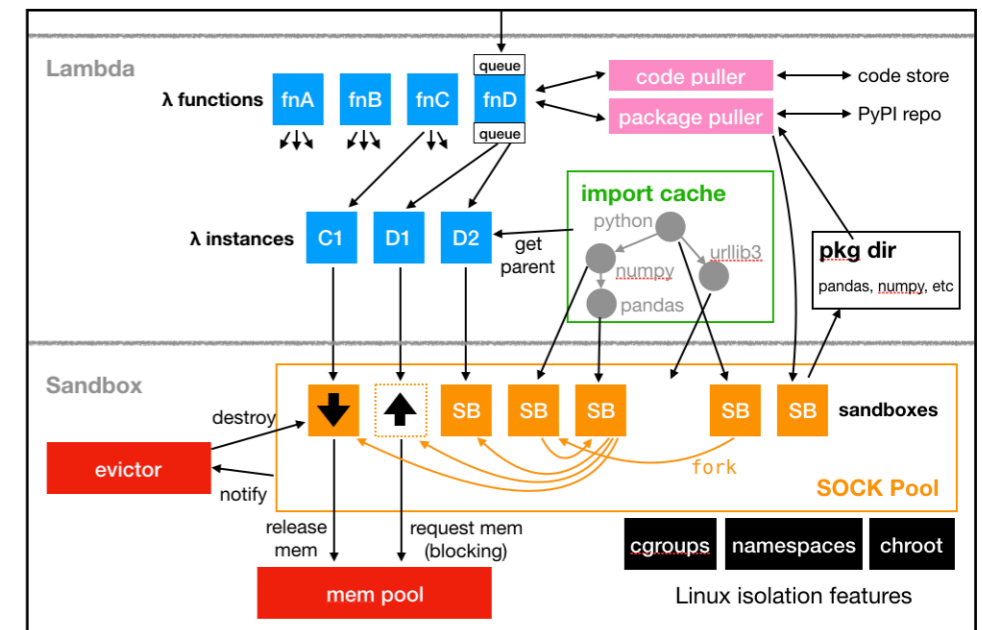
Industry experience

- Worked at Microsoft on SQL Server and Cloud
- Other internships/collaborations: Qualcomm, Google, Facebook, Tintri, Bauplan



Open source

- OpenLambda (serverless cloud platform)
- <https://github.com/open-lambda/open-lambda>



# Who are You?

Year in school? Major?

What CS courses have people taken before?

- 320? 400? 537/564/640?

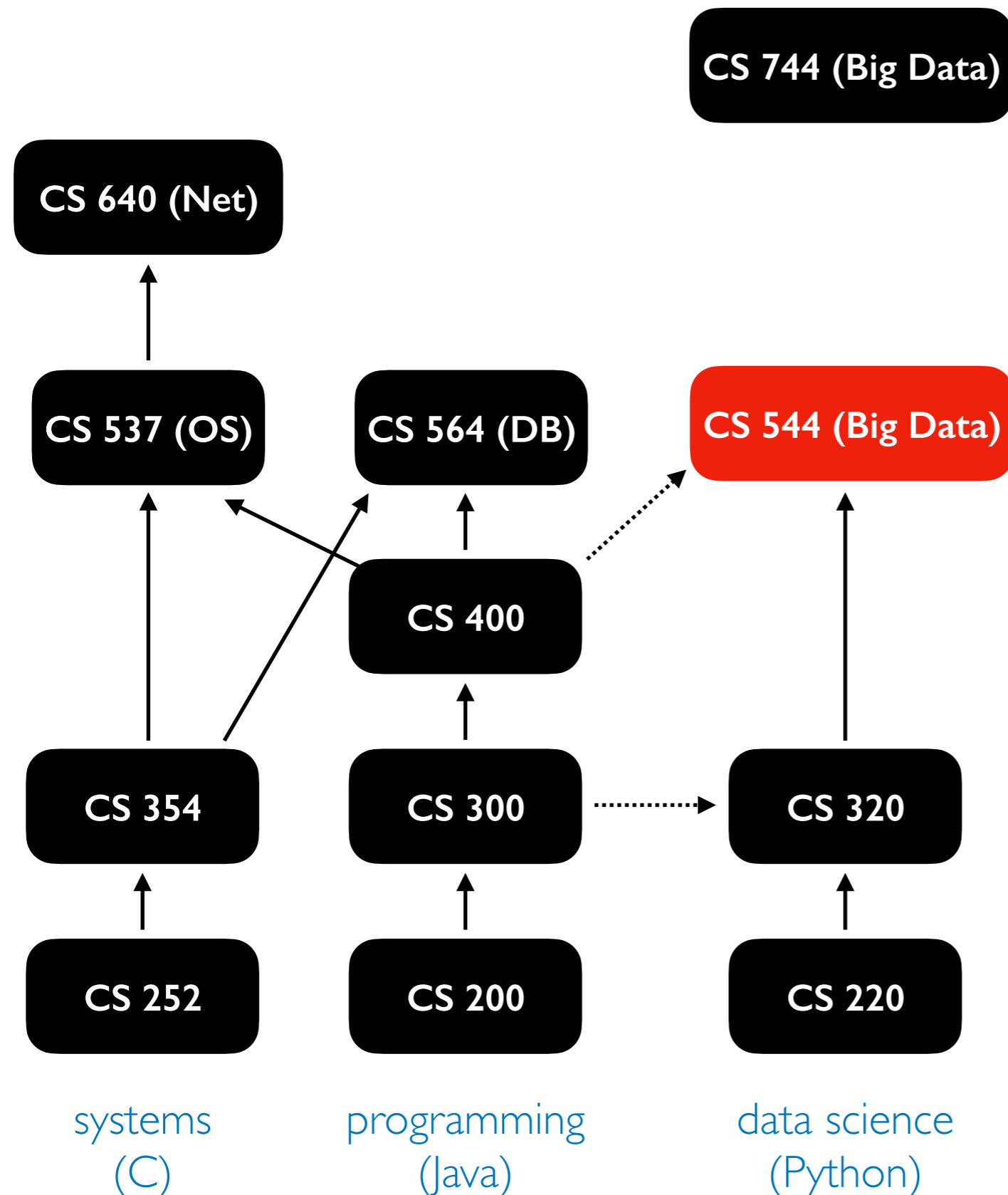
Please fill this form (**due today**):

<https://forms.gle/3fmvXgVYQm7JLCmh6>

Why?

- Help me get to know you
- Get extra credit

# Related courses



- most coding will be in Python (400 folks will need to pick this up)
- first third of course will cover some foundations from operating systems, networking, and databases
- 744 will cover some similar systems, but from the research perspective (544 is hands on)

# What are "systems"?

Some major categories of software

- analysis code (run once, get results)
- applications (long running, maybe a website)
- **systems** (manage **resources**, like storage space)

Other code uses systems. For example, without an operating system, your analysis code couldn't read files.

Whatever kind of programming you doing,  
knowing how systems work with help you **use resources better!**

# What are "*big data systems*"?

Some major categories of software

- analysis code (run once, get results)
- applications (long running, maybe a website)
- **systems** (manage **resources**, like storage space)

Other code uses systems. For example, without an operating system, your analysis code couldn't read files.

**As data grows, we need to optimize our code and/or use more resources**

Big data systems manage resources that are:

- **distributed** (cluster of machines)
- **specialized** (e.g., GPUs)

# What will you learn in 544?

## Learning objectives

- **Deploy** distributed systems for data storage and analytics
- Demonstrate competencies with tools and processes necessary for **loading data** into distributed storage systems
- Write programs that use distributed platforms to **efficiently analyze large datasets**
- Produce meaning from large datasets by **training machine learning models** in parallel or on distributed systems
- **Measure resource usage** and overall cost of running distributed programs
- Optimize distributed analytics programs to **reduce resource consumption** and program runtime
- Demonstrate competencies with **cloud services** designed to store or analyze large datasets



# What will you learn today?

## Learning objectives

- recall course **logistics and policies**
- describe different kinds of **hardware resources**
- compare **scale up** to **scale out** approaches
- compare different approaches for running code on a CPU

# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab, Anki

## Resources

- Overview
- Compute
- Memory
- Storage
- Network

## Deployment

# Main Websites

1

<https://tyler.caraza-harter.com/cs544/f24/schedule.html>

- schedule, course content, how to get help
- links to all other resources/tools
- some lecture recordings (review only)

2

<https://github.com/cs544-wisc/f24>

- project specifications
- lecture demo code

3

Canvas

- announcements
- quizzes
- grade summaries
- zoom office hours

# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab, Anki

## Resources

- Overview
- Compute
- Memory
- Storage
- Network

## Deployment

1

TopHat (me asking you questions during lecture)

- Optional, but earn extra credit for correct answers!

2

Piazza (asking questions of **general interest**)

- our goal: responses <24 hours
- don't post >5 lines of project code

3

Email (asking questions of **individual interest**)

- everybody will be assigned a TA contact (544 has 8 this semester)
- our goal: responses <2 business days
- feel free to escalate by CC'ing instructor on same thread after 2 days
- if contacting multiple staff members about same issue, please keep all on the same thread (don't start multiple threads)

4

GitLab

- you'll be given a **private** repo for each project
- we'll provide feedback on GitHub

5

Anki Flash Cards

- memory terms, basic ideas using flash cards and spaced repetition

# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

## Resources

- [Overview](#)
- Compute
- Memory
- Storage
- Network

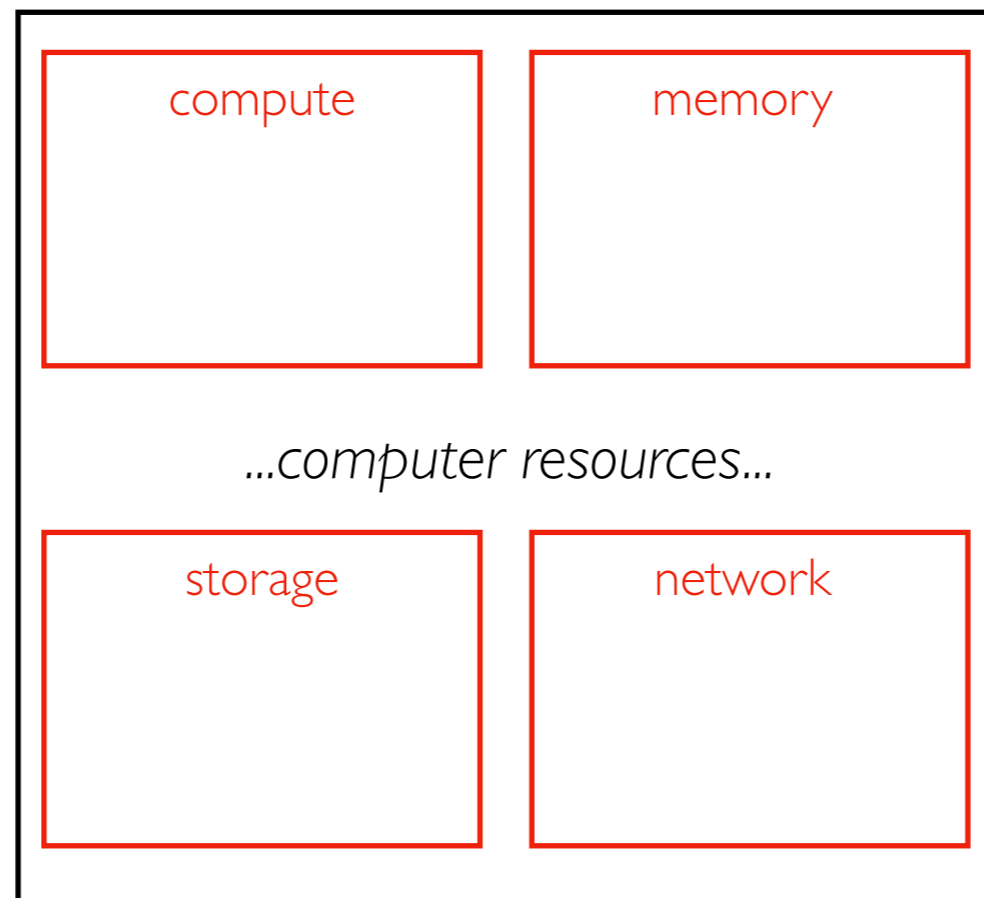
## Deployment

# Categories of resources

**Systems:** software for managing computer resources

Other kinds of software (analysis code, applications) rely on systems.

a computer:

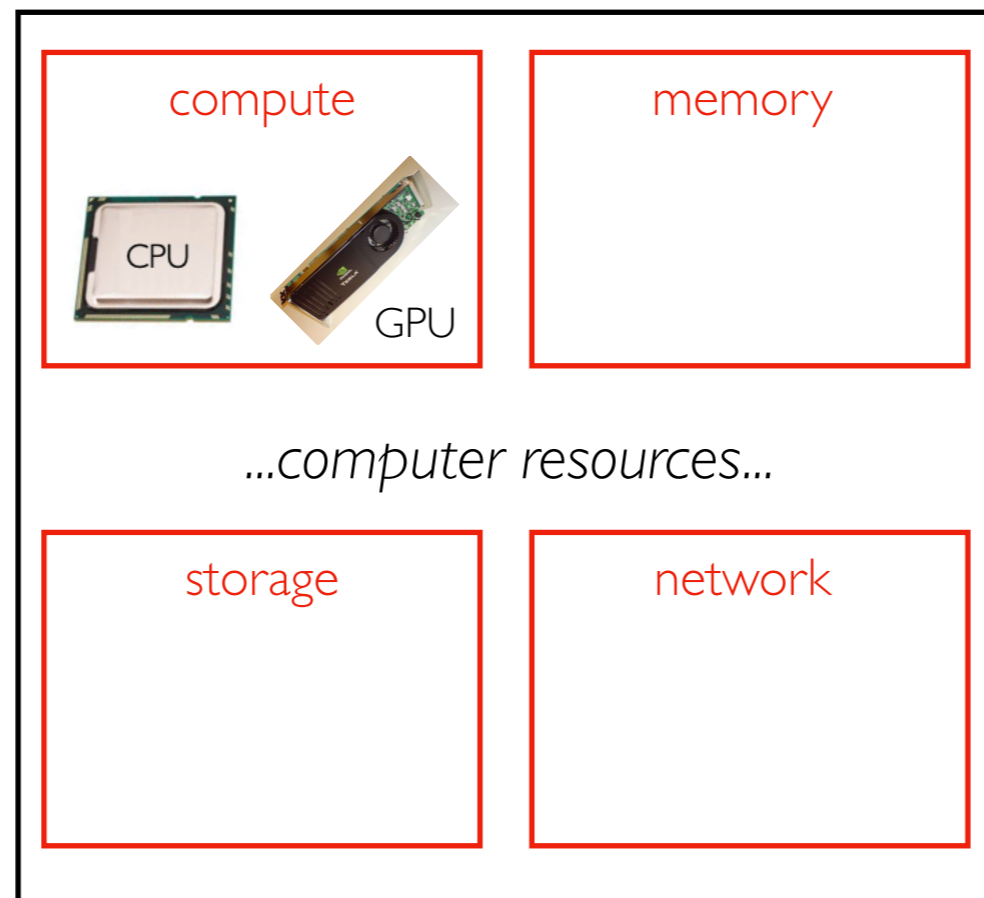


# Categories of resources

**Systems:** software for managing computer resources

Other kinds of software (analysis code, applications) rely on systems.

a computer:



computational resources  
execute code

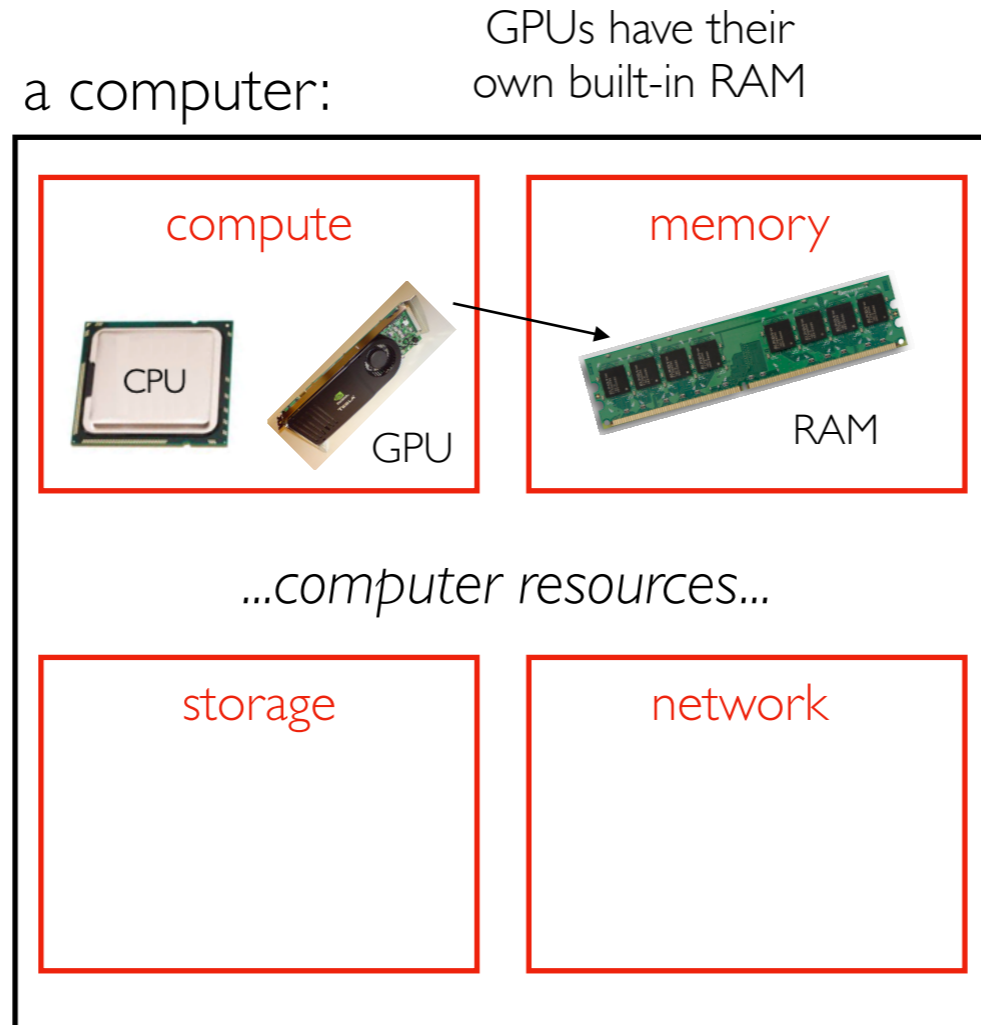
central processing unit (CPU), graphics processing unit (GPU)



# Categories of resources

**Systems:** software for managing computer resources

Other kinds of software (analysis code, applications) rely on systems.



computational resources  
execute code

memory holds data  
for active usage

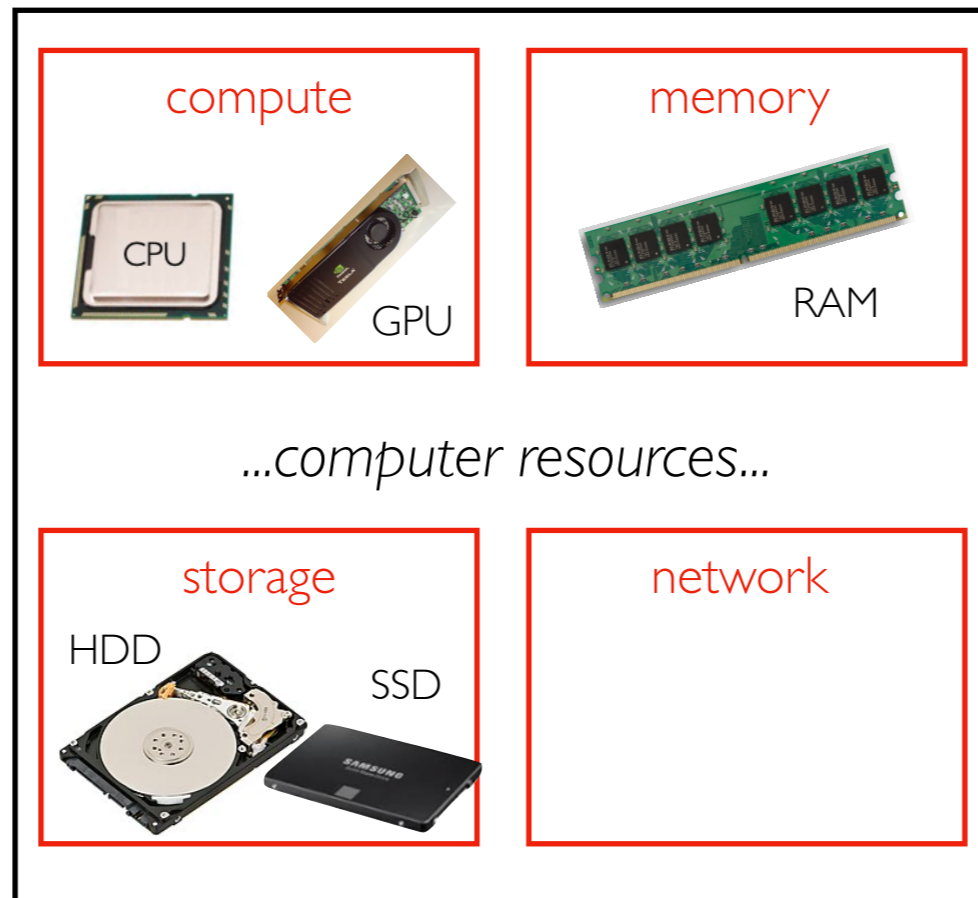
random-access memory (RAM)

# Categories of resources

**Systems:** software for managing computer resources

Other kinds of software (analysis code, applications) rely on systems.

a computer:



computational resources  
execute code

memory holds data  
for active usage

storage holds  
long-term data

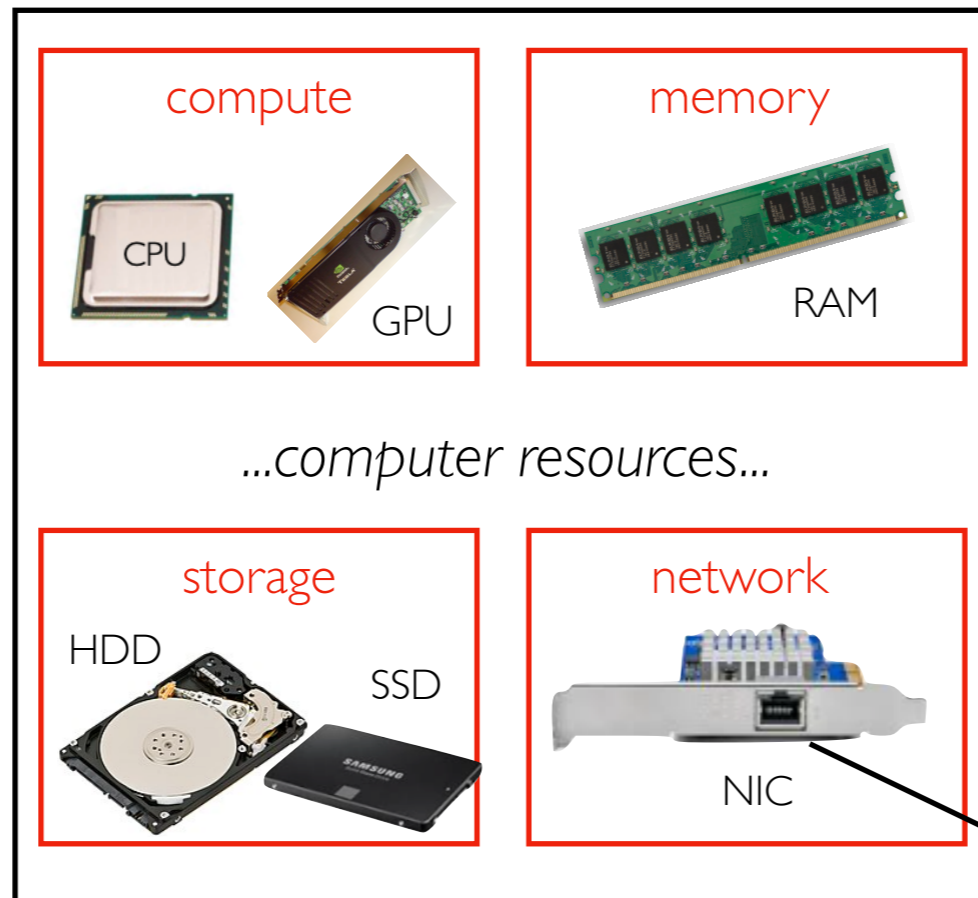
hard disk drive (HDD), solid-state disk (SSD)

# Categories of resources

**Systems:** software for managing computer resources

Other kinds of software (analysis code, applications) rely on systems.

a computer:



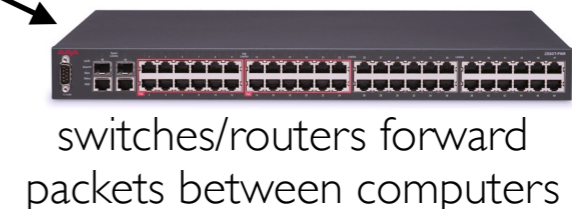
computational resources  
execute code

memory holds data  
for active usage

storage holds  
long-term data

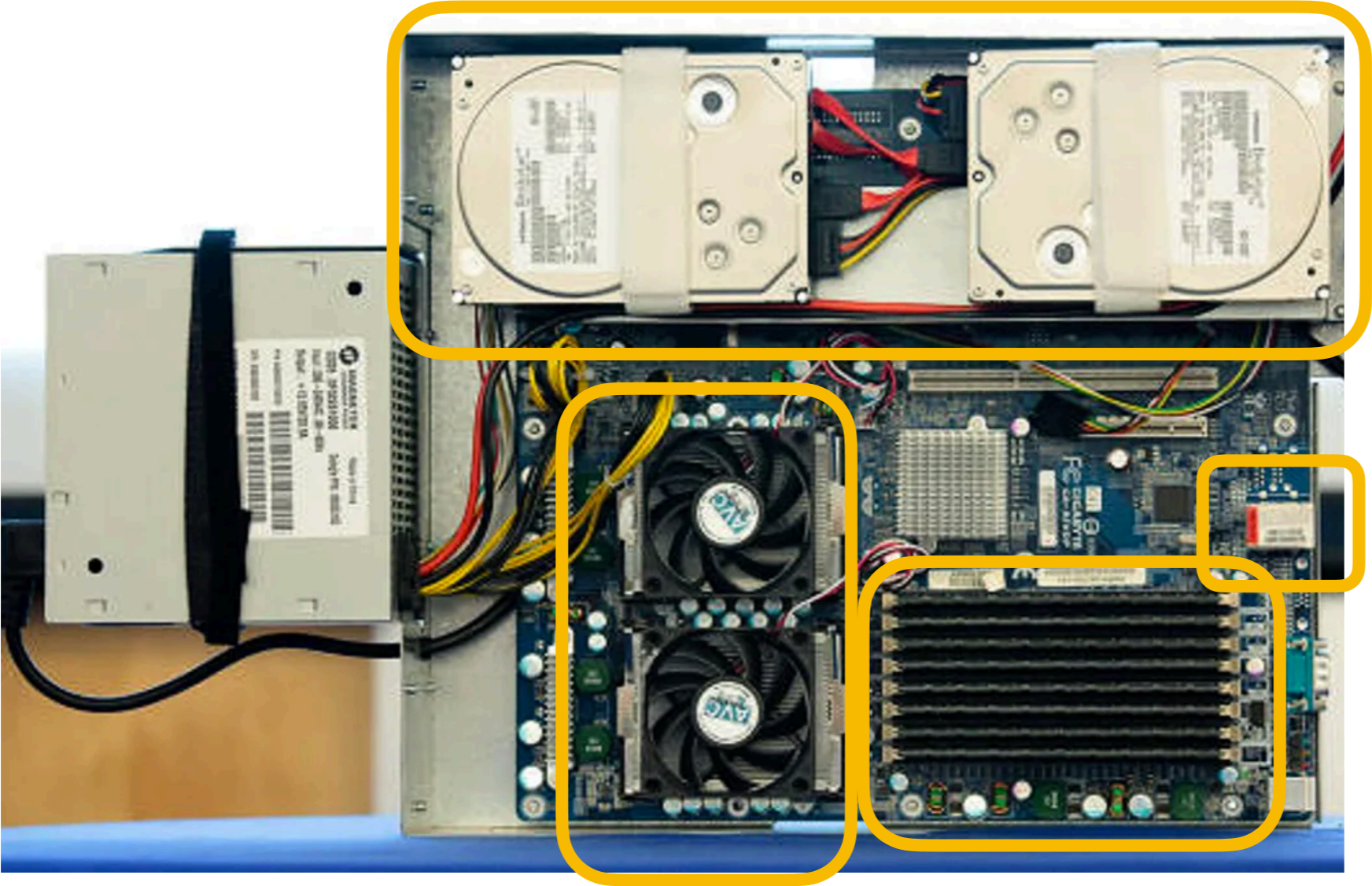
network provides  
communication between  
computers

network interface card (NIC)



# A real server

Hard Drives



NIC

CPUs

RAM

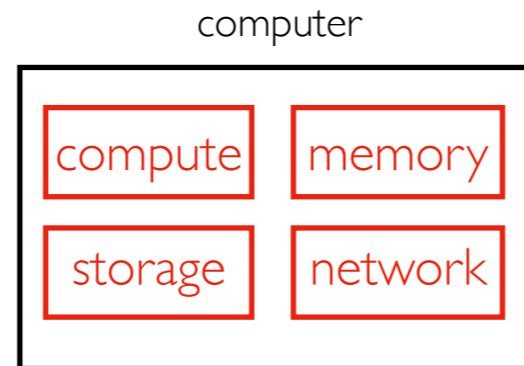
# Big Data

Potential problems as datasets grow

- might run too slowly
- might not be able to run at all (for example, not enough memory)

Solutions:

- more efficient code
- use more resources



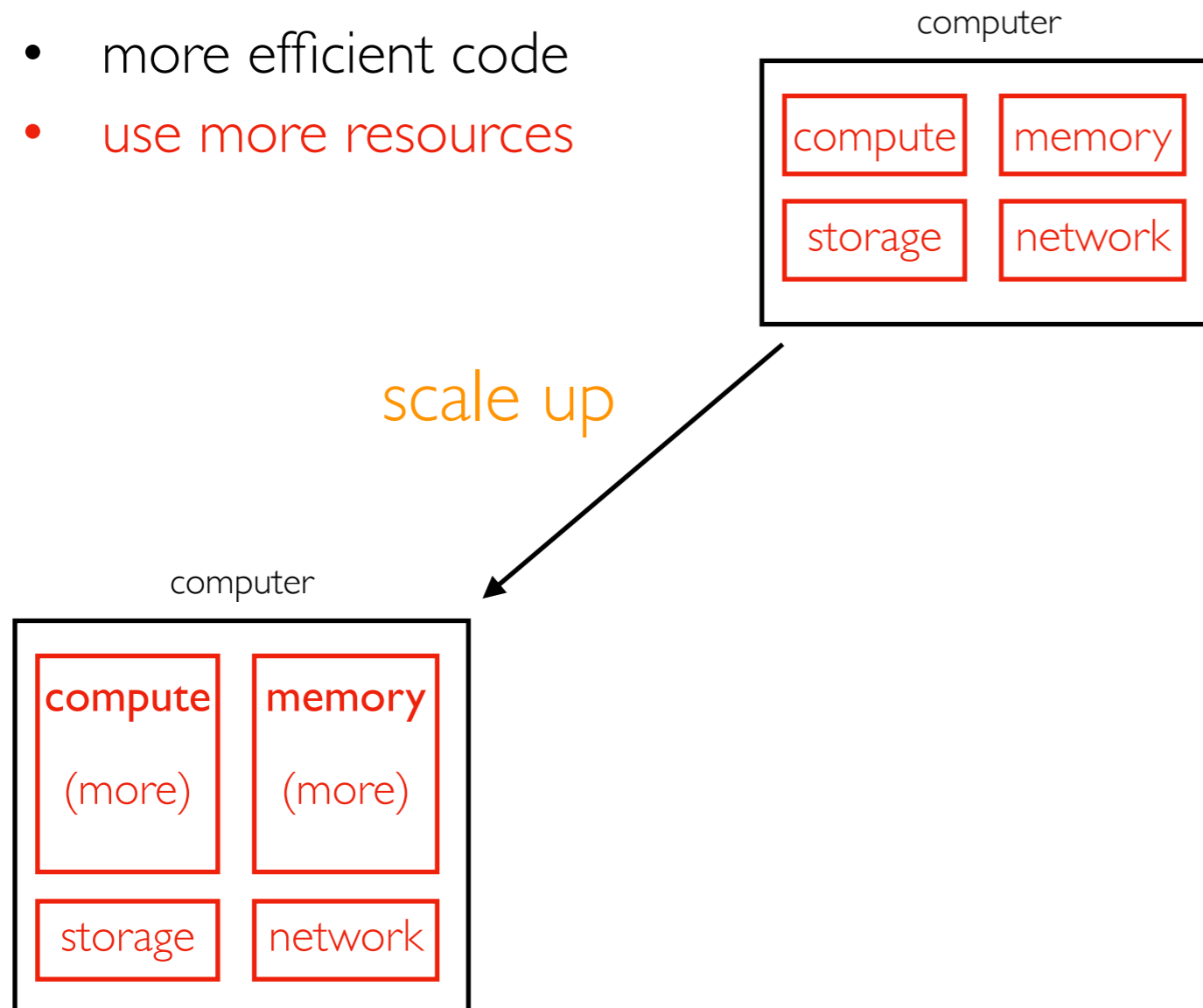
# Big Data

Potential problems as datasets grow

- might run too slowly
- might not be able to run at all (for example, not enough memory)

Solutions:

- more efficient code
- use more resources



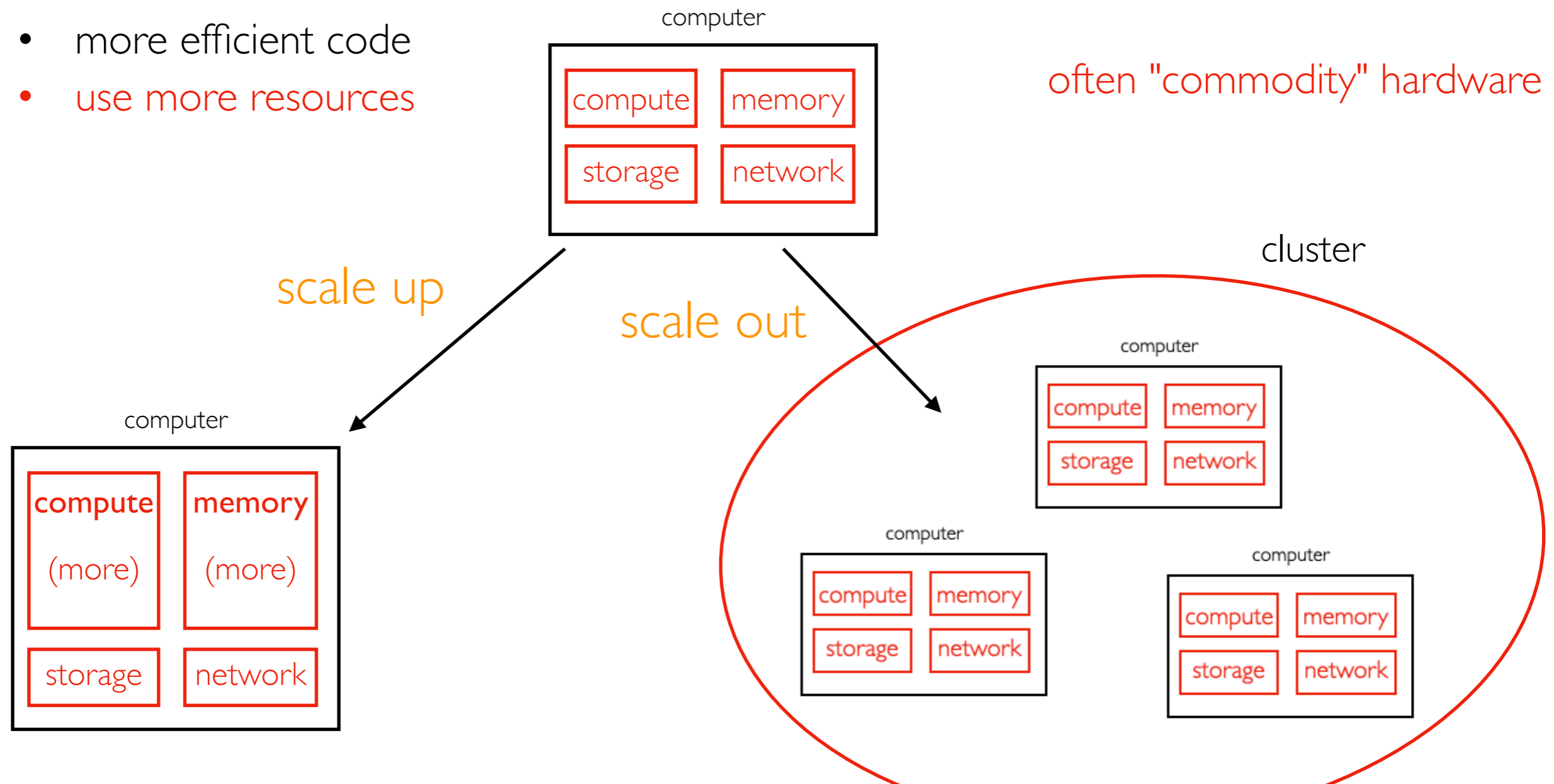
# Big Data

Potential problems as datasets grow

- might run too slowly
- might not be able to run at all (for example, not enough memory)

Solutions:

- more efficient code
- use more resources



# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

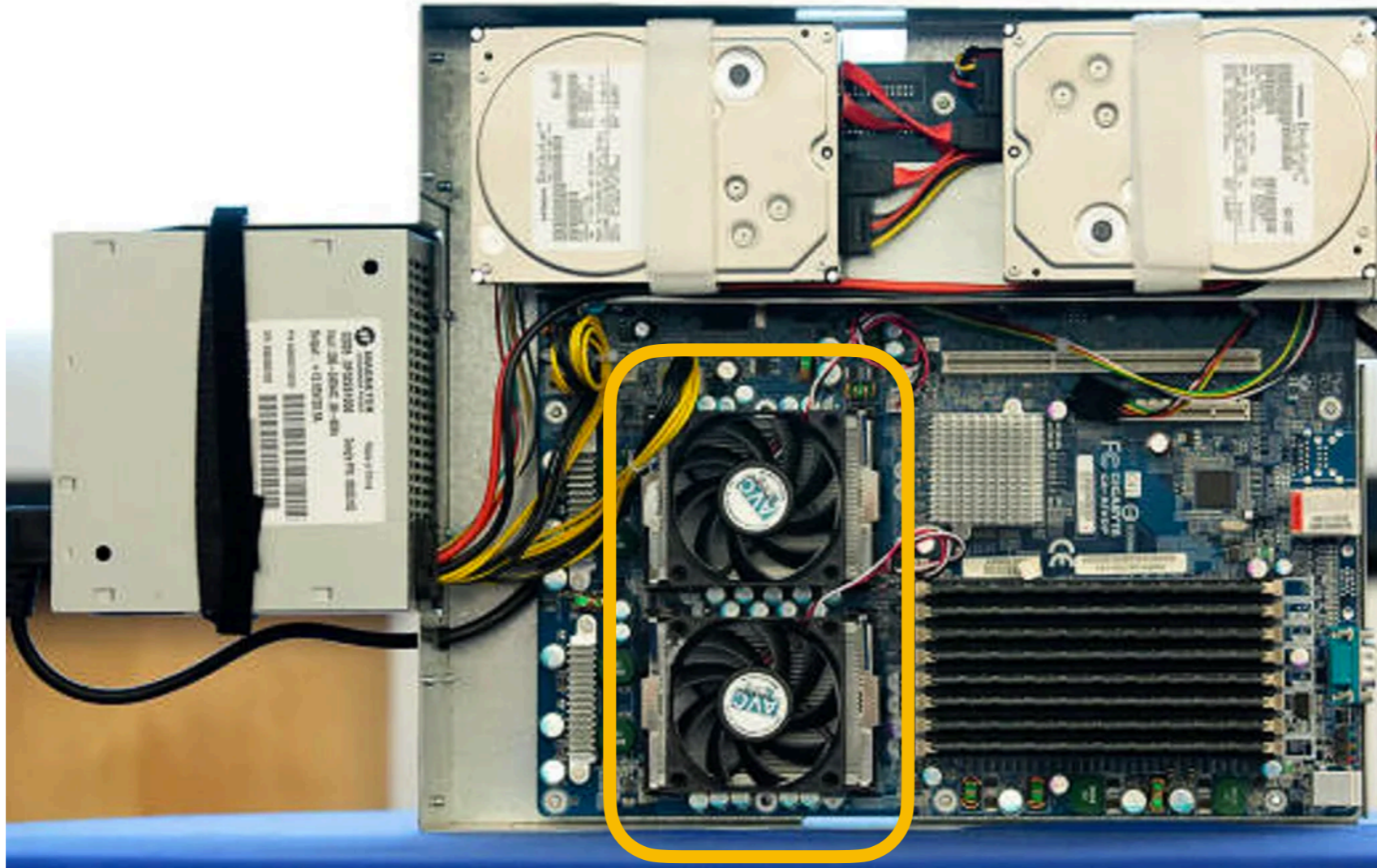
## Resources

- Overview
- **Compute**
- Memory
- Storage
- Network

## Deployment



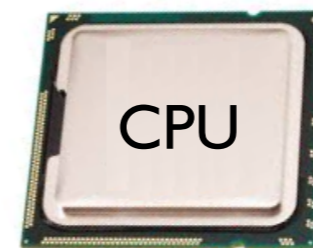
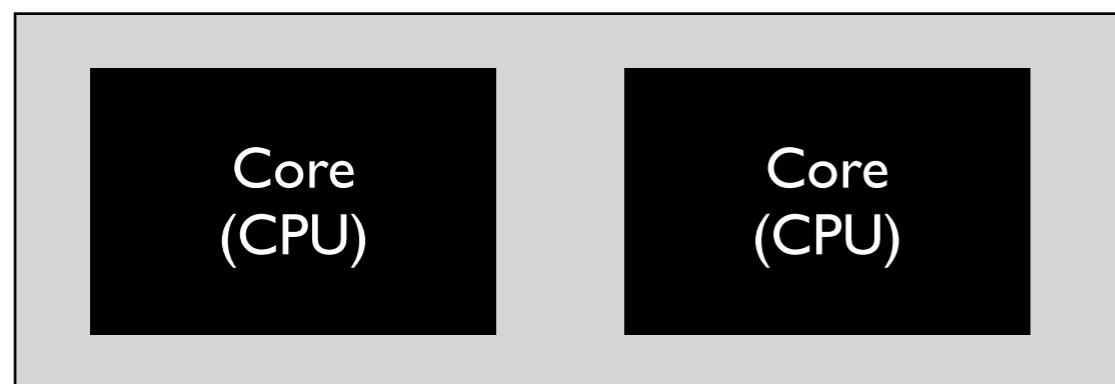
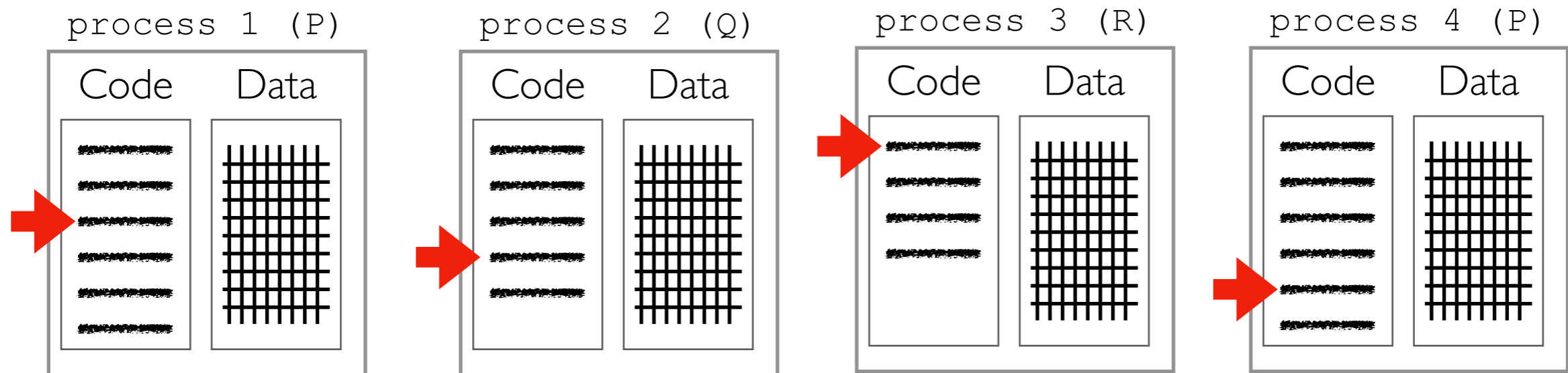
# Compute



Some computers have multiple CPUs. Modern CPUs typically have multiple cores. Each core works like a CPU and runs programs by executing instructions.

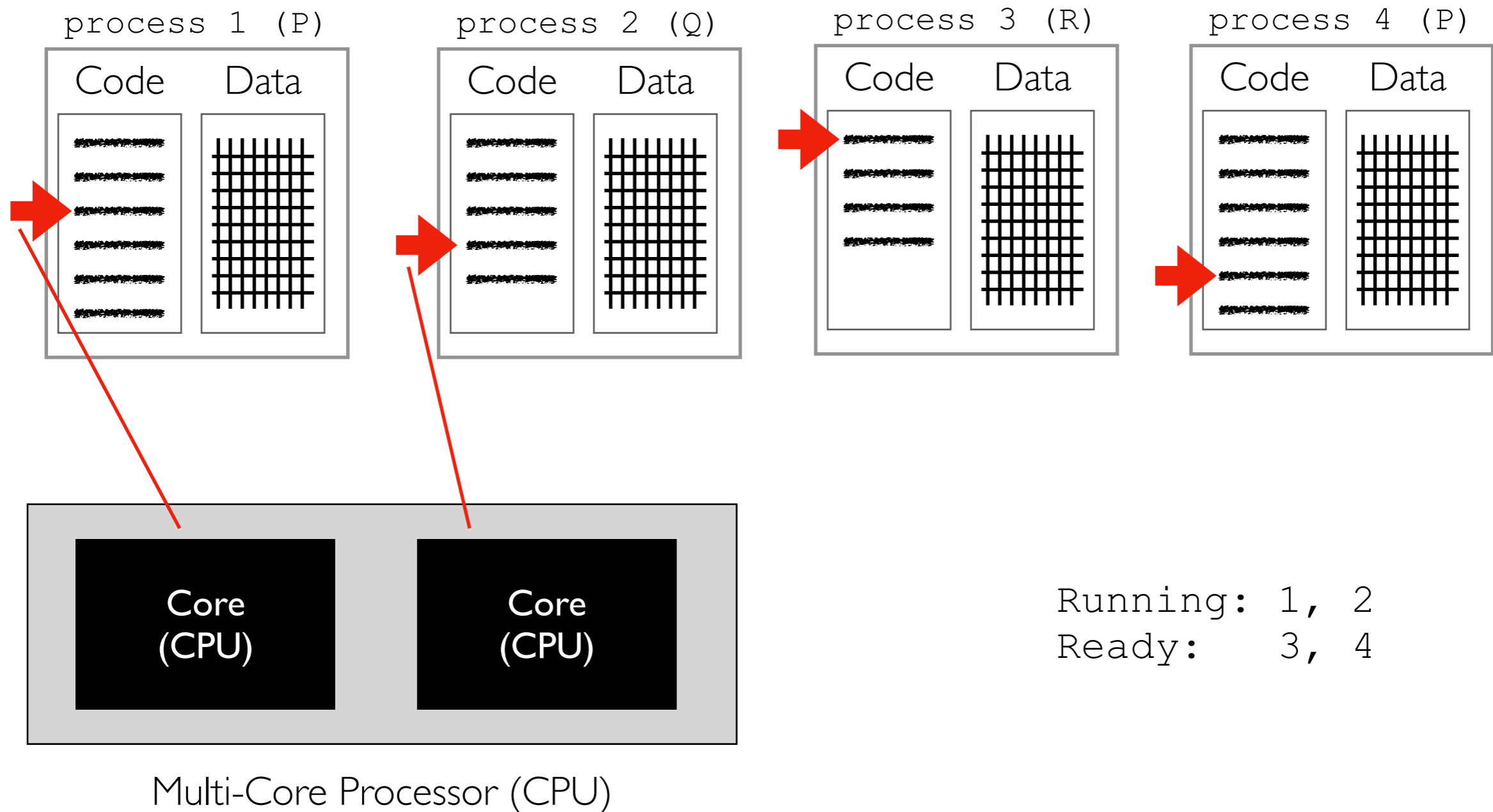
**How do cores run machine code?**

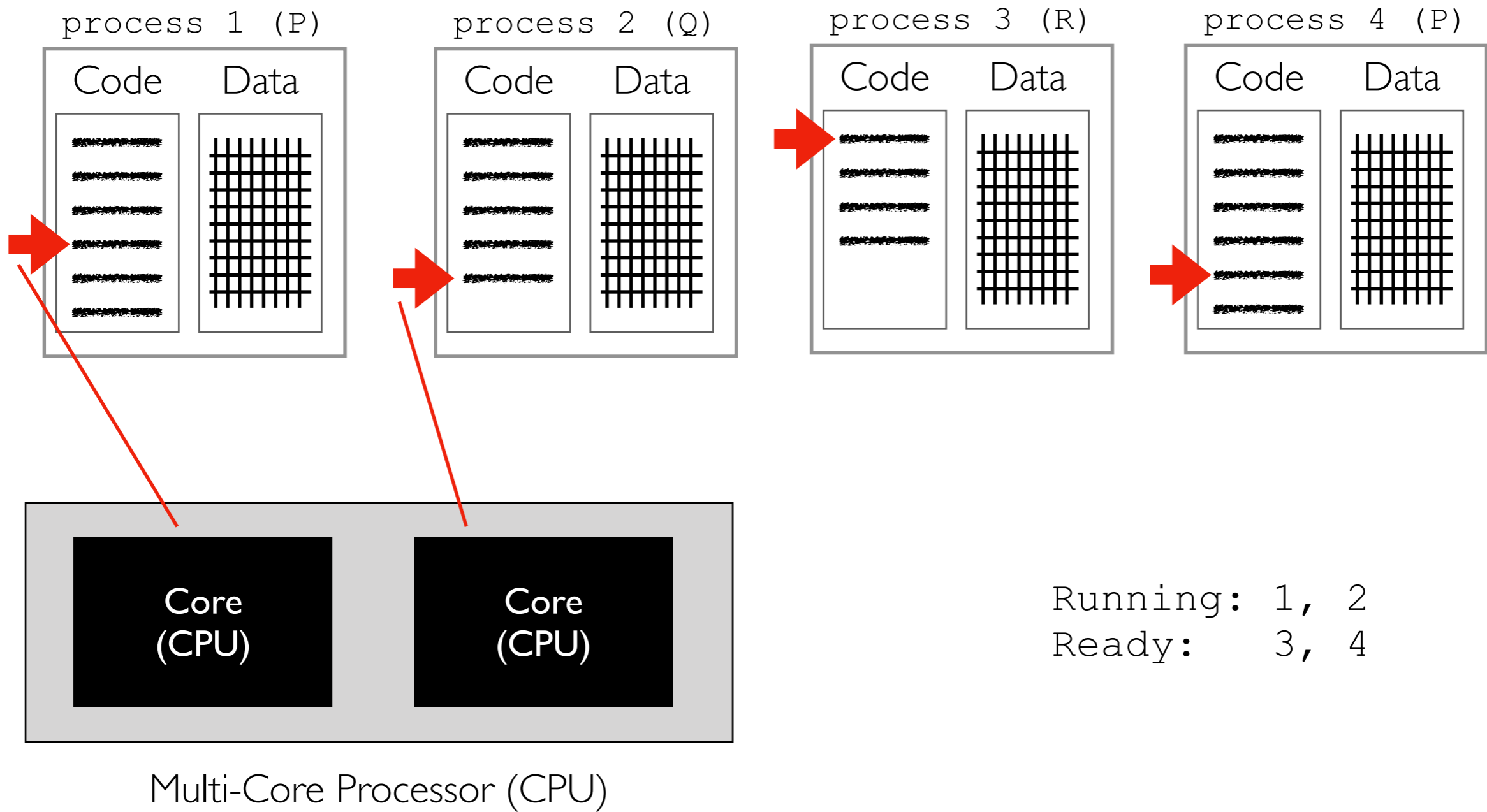
the operating system "schedules" tasks on cores  
(decides when they get to run)

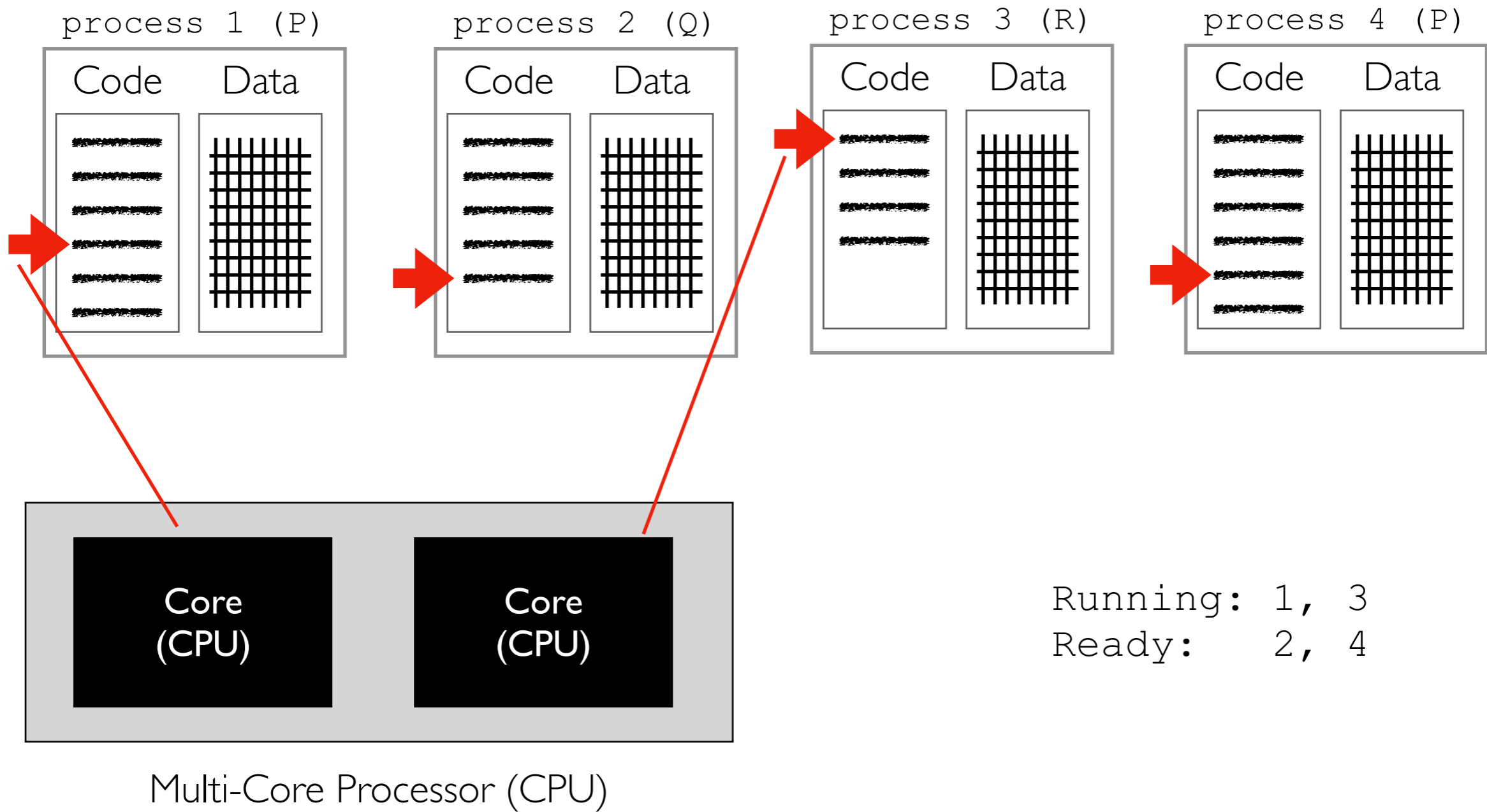


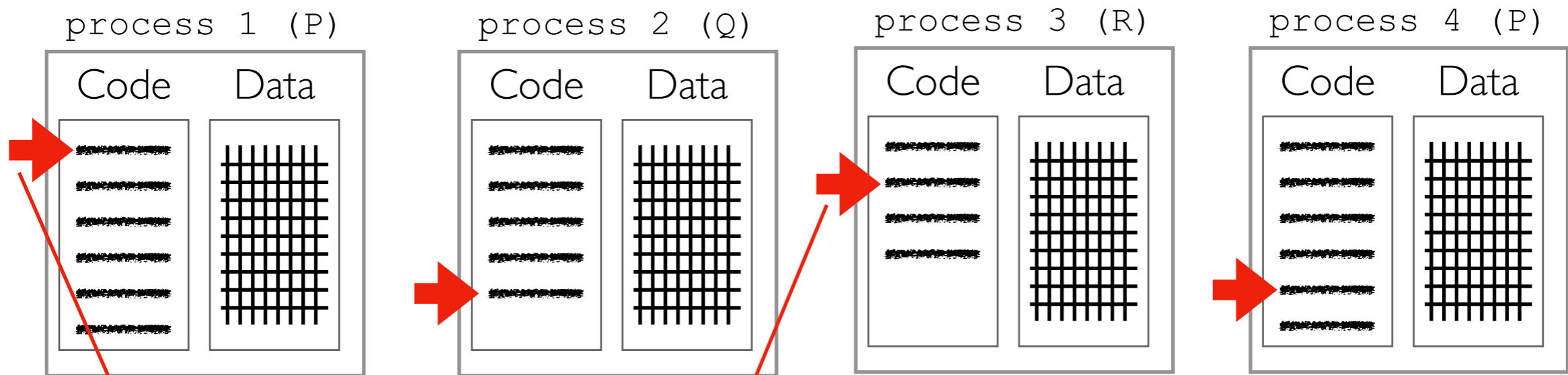
Multi-Core Processor (CPU)

a core can execute instructions for one (or in some cases two) tasks at a time



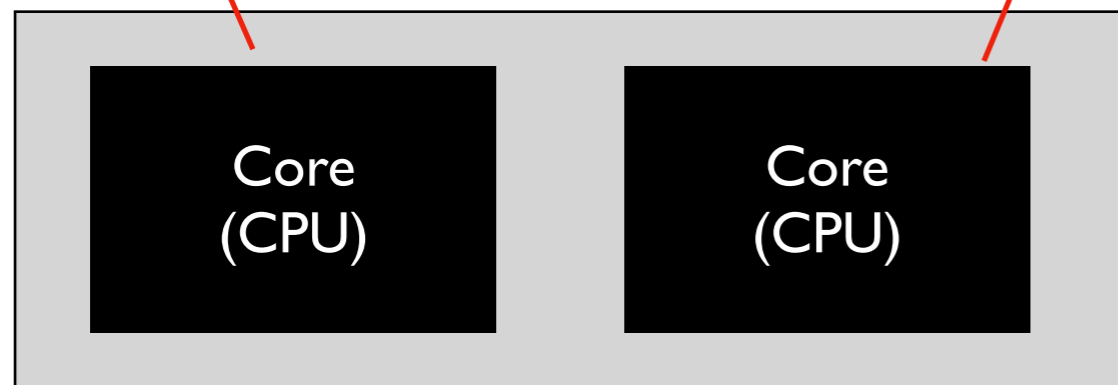






the more cores we have, the more tasks we can run simultaneously (that is, in "parallel")

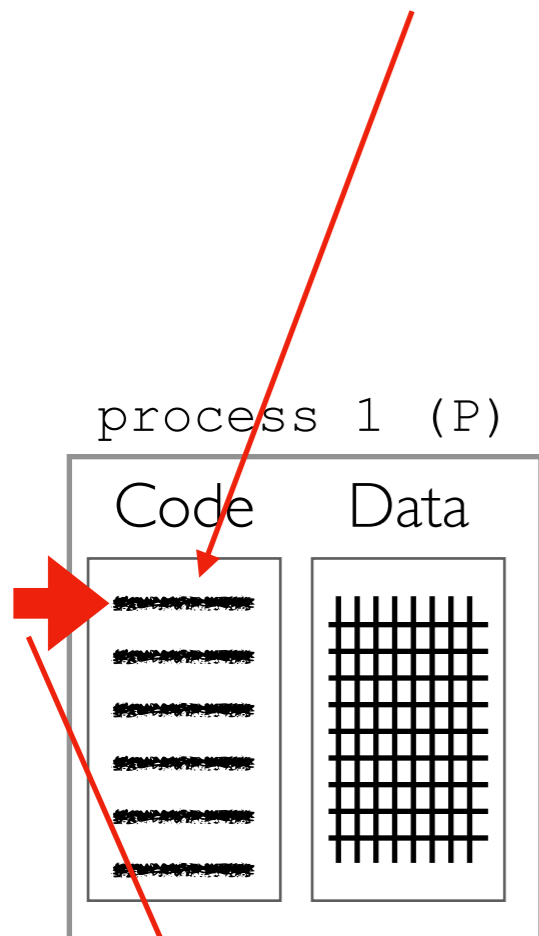
Running: 1, 3  
Ready: 2, 4



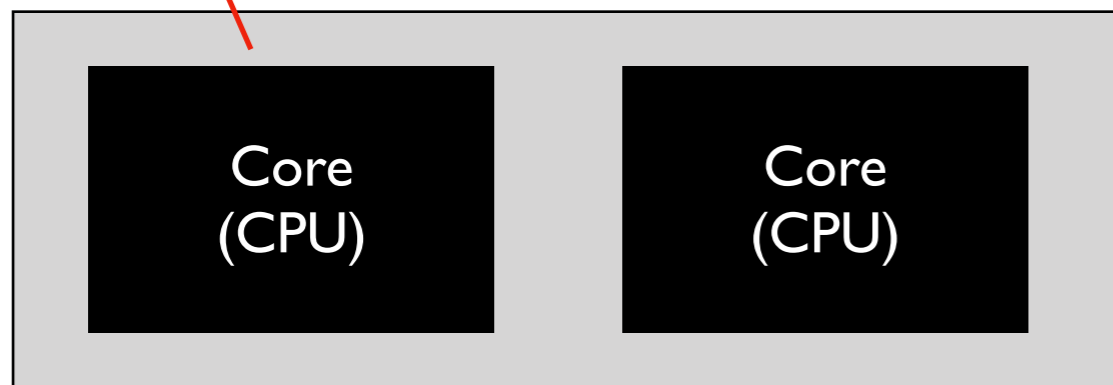
Multi-Core Processor (CPU)

these instructions are in "machine code"  
that the CPU can understand

<https://pages.cs.wisc.edu/~deppeler/cs354/reference/x86-cheat-sheet.pdf>



these instructions are  
represented as 1's and 0's  
the CPU understands



Multi-Core Processor (CPU)

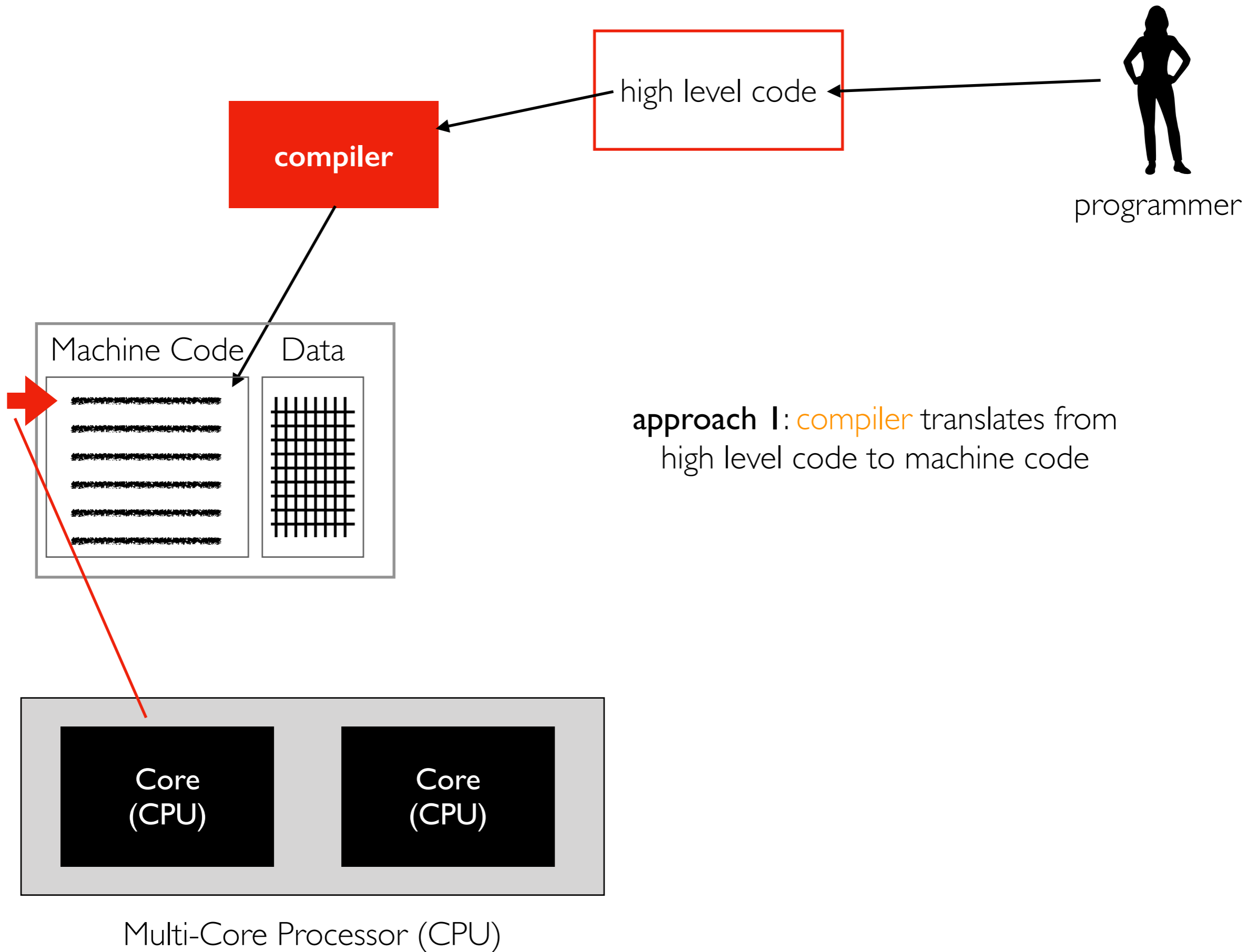
```
arithmetic  
two operand instructions  
addl src,dst  dst = dst + src  
subl src,dst  dst = dst - src  
imull src,dst dst = dst * src  
sall src,dst  dst = dst << src (aka shll)  
sarl src,dst  dst = dst >> src (arith)  
shrl src,dst  dst = dst >> src (logical)  
xorl src,dst  dst = dst ^ src  
andl src,dst  dst = dst & src  
orl  src,dst  dst = dst | src  
one operand instructions  
incl dst      dst = dst + 1  
decl dst      dst = dst - 1  
negl dst      dst = -dst  
notl dst      dst = ~dst  
arithmetic ops set CCs implicitly  
cf=1 if carry out from msb  
zf=1 if dst==0,  
sf=1 if dst < 0 (signed)  
of=1 if two's complement  
      (signed) under/overflow
```

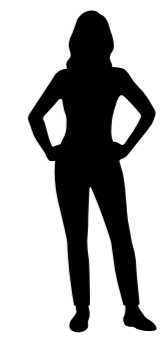
how do we bridge the gap between "high level"  
code (Python/Java/etc) and machine code?



how do we bridge the gap between "high level"  
code (Python/Java/etc) and machine code?

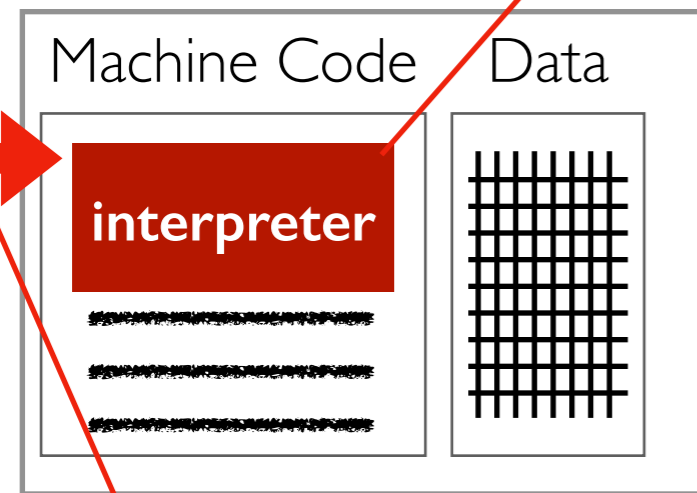
*Note: we'll primarily write Python this semester, but it helps to explore this in general to understand how systems like Spark work (which is written in Scala and uses the Java Virtual Machine)*



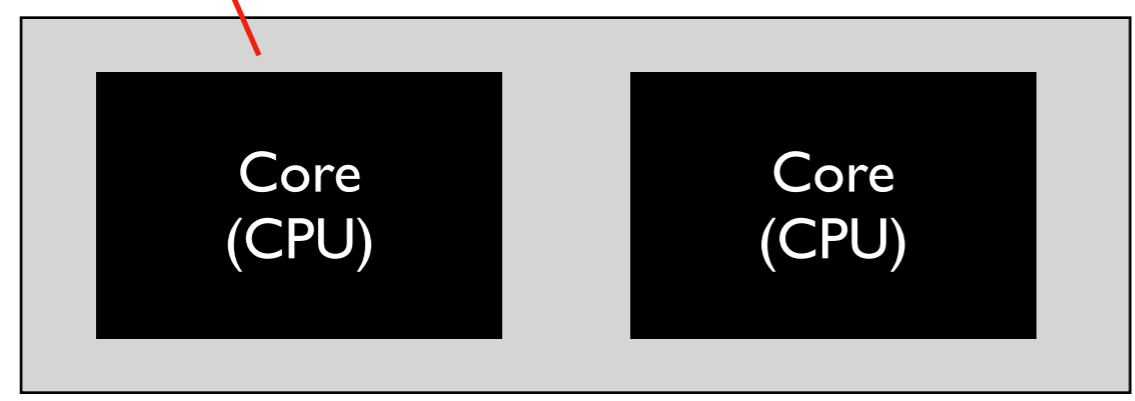


programmer

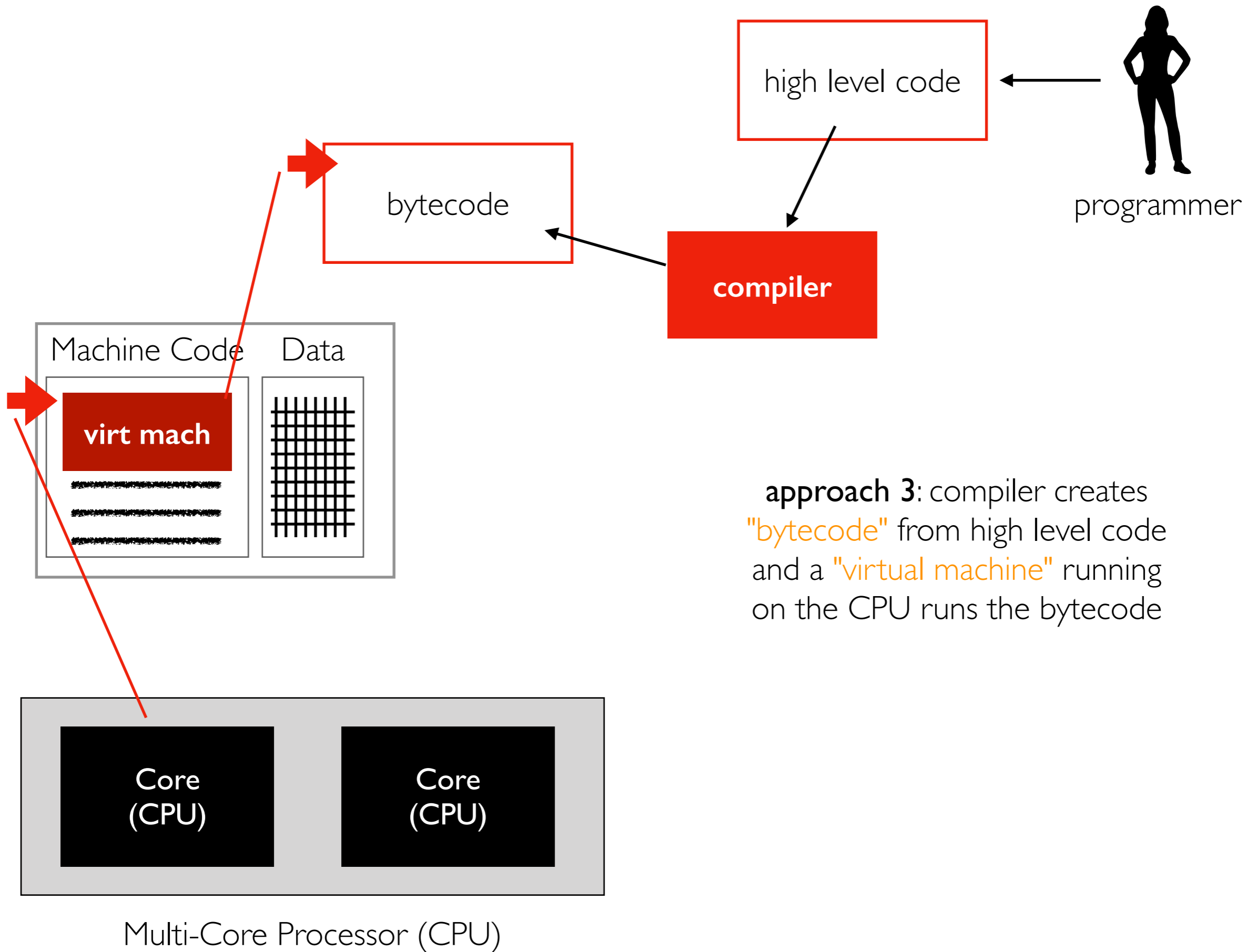
high level code



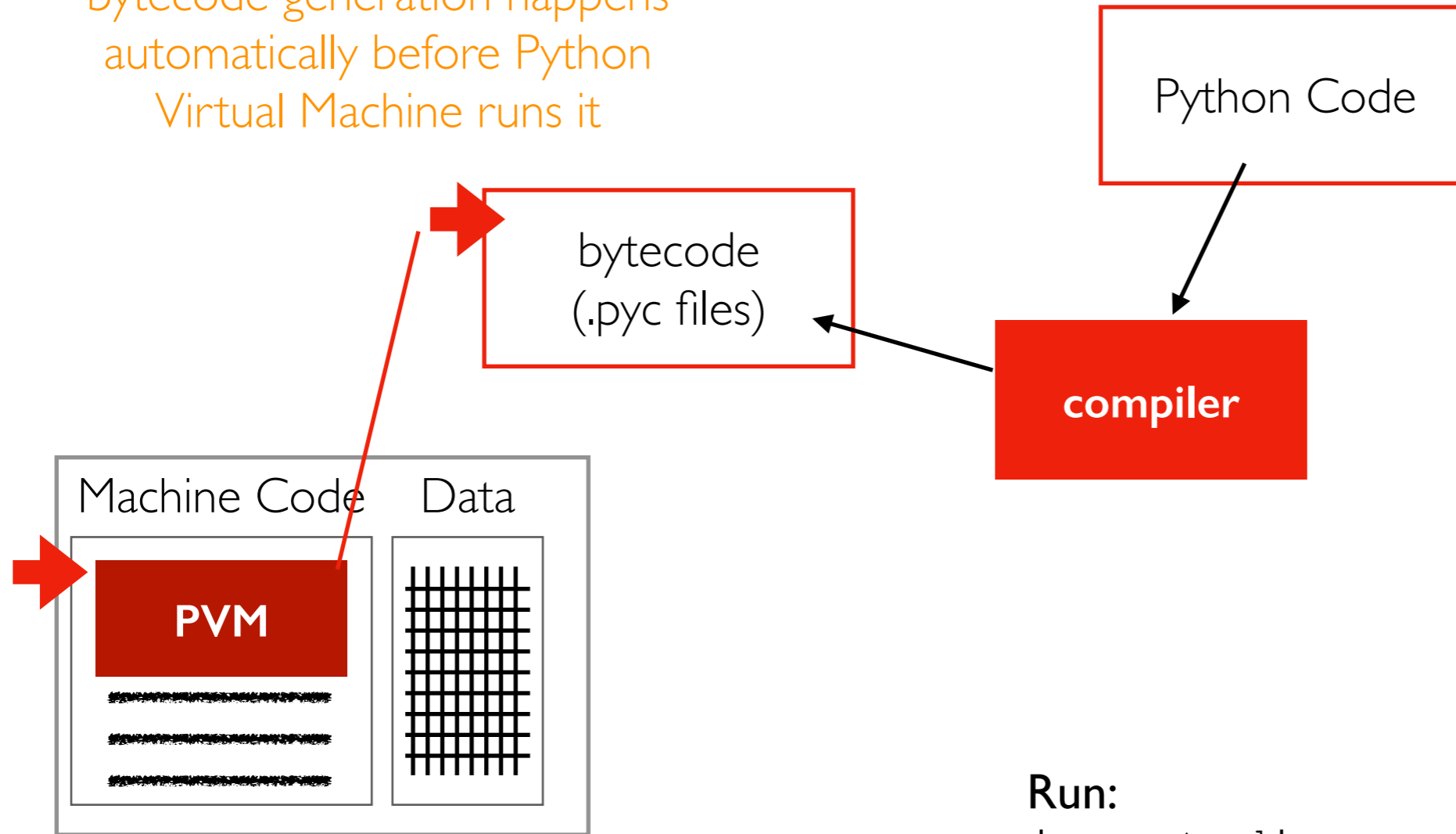
approach 2: CPU runs an **interpeter** program that loops over programmer's code and runs it



Multi-Core Processor (CPU)

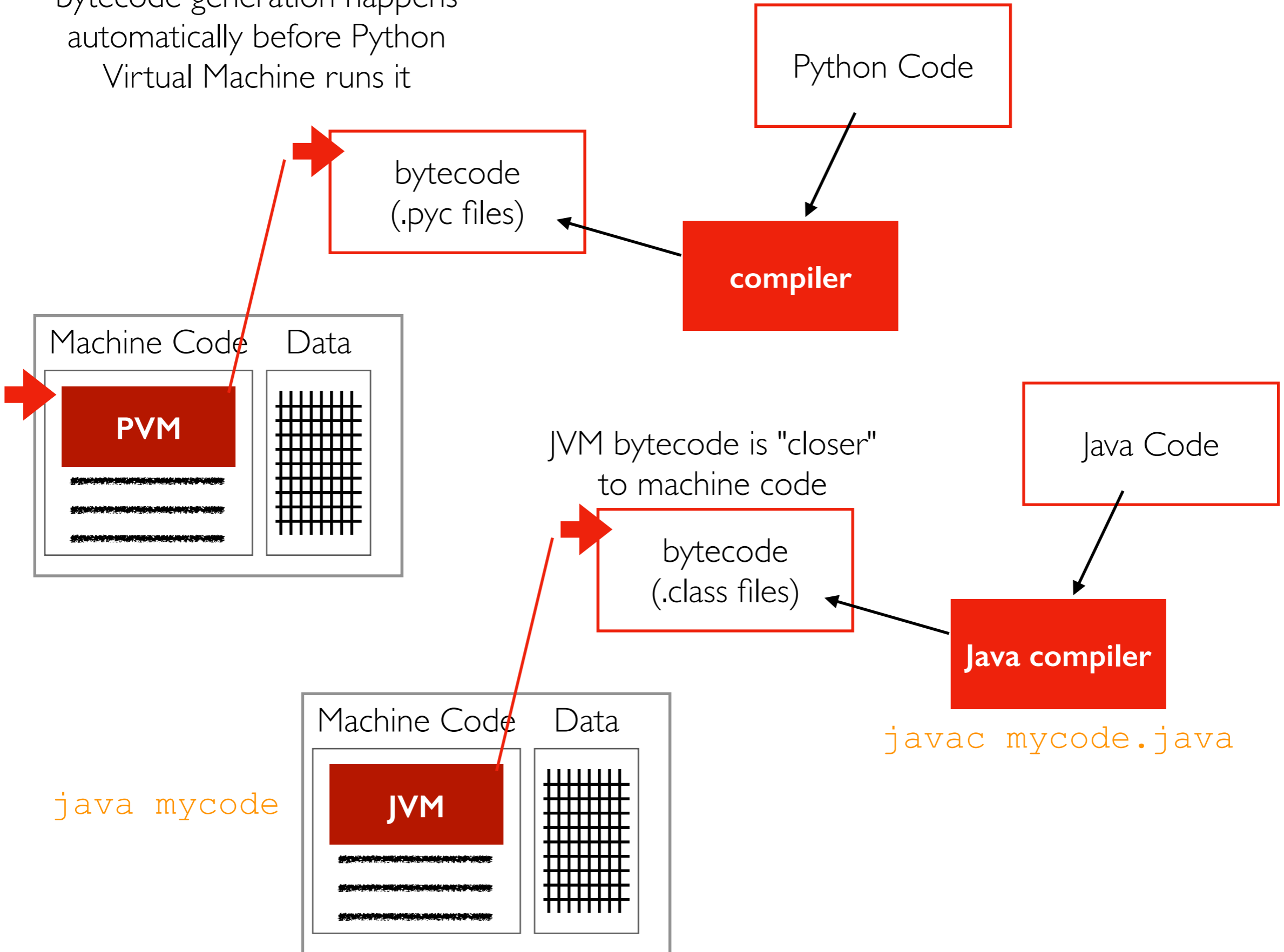


when you run "python3 ..."  
bytecode generation happens  
automatically before Python  
Virtual Machine runs it

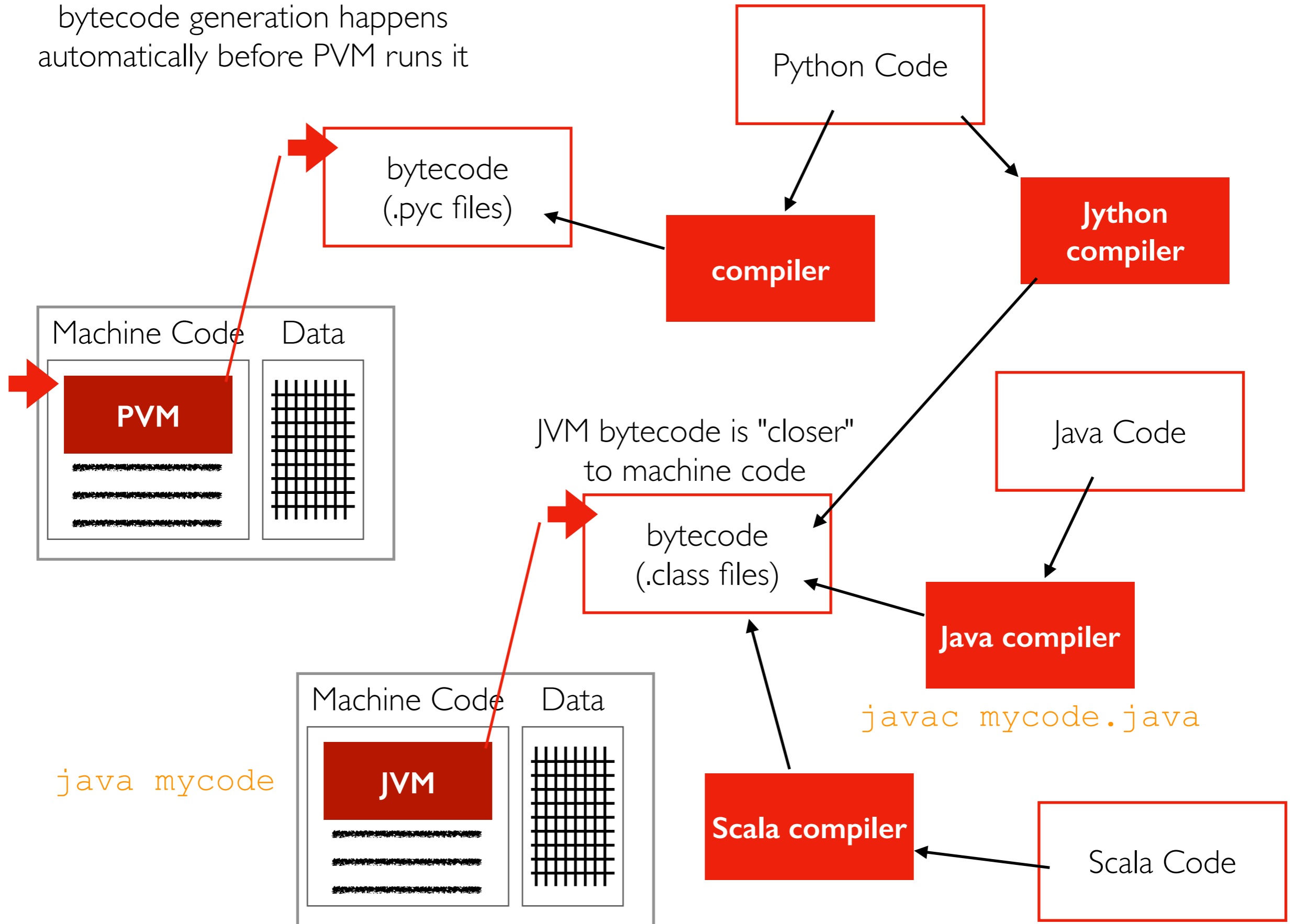


```
Run:  
import dis  
dis.dis("z = x + y * 2")
```

when you run "python3 ..."  
bytecode generation happens  
automatically before Python  
Virtual Machine runs it



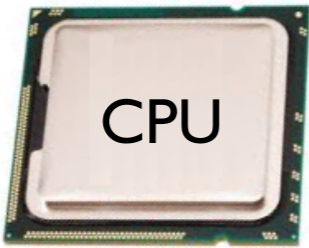
when you run "python3 ..."  
bytecode generation happens  
automatically before PVM runs it



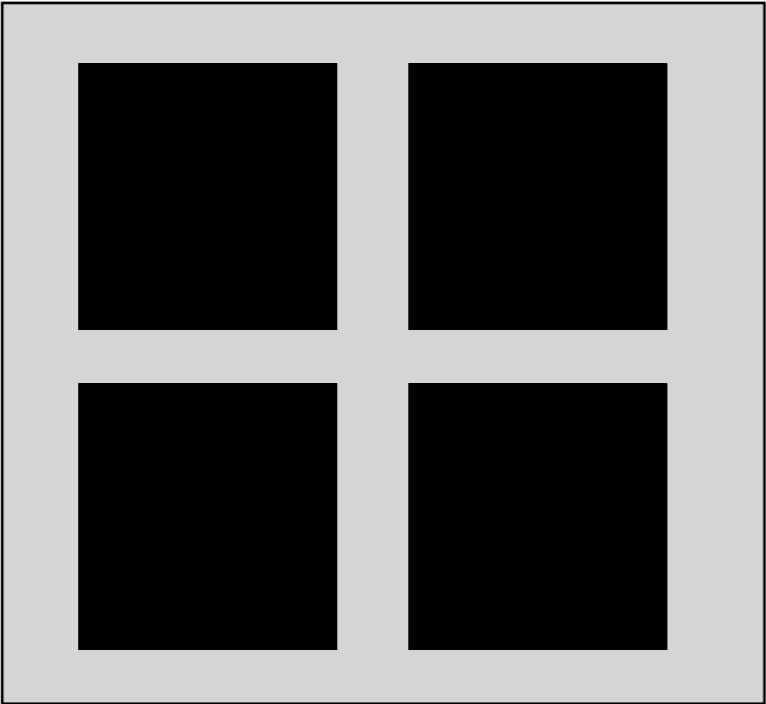
**What are alternatives to CPUs for compute?**



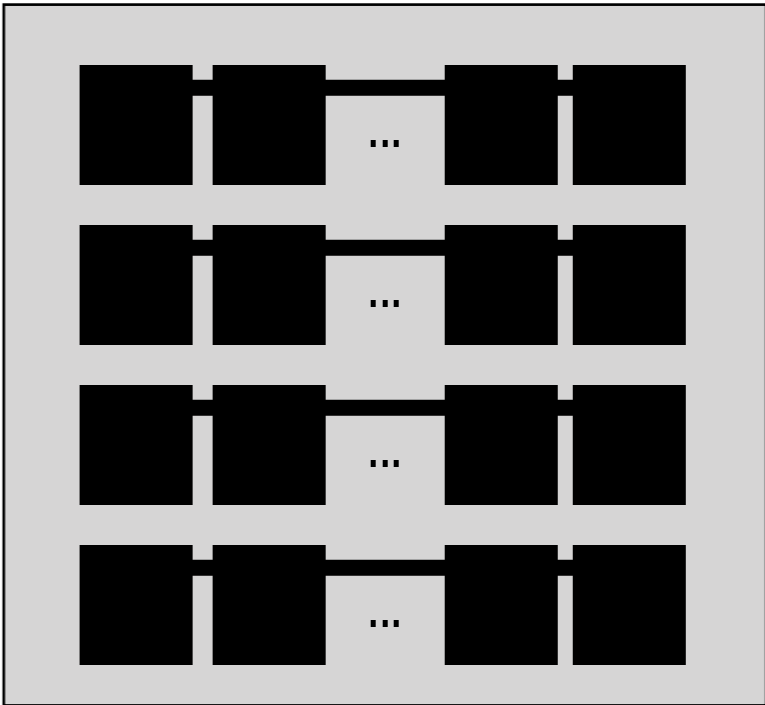
GPUs (graphical processing units) are an alternative compute resource.



[https://en.wikipedia.org/wiki/Nvidia\\_Tesla](https://en.wikipedia.org/wiki/Nvidia_Tesla)



few cores that are fast,  
flexible, independent

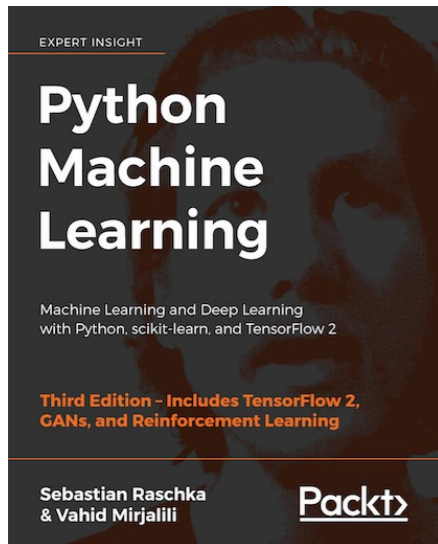


many cores that are slow,  
float-optimized, coordinated

# GPU vs. CPU: Cost Comparison

TOP HAT

The GPU is 30% cheaper but 28x faster at floating-point operations!



Specifications	Intel® Core™ i7-6900K Processor Extreme Ed.	NVIDIA GeForce® GTX™ 1080 Ti
Base Clock Frequency	3.2 GHz	< 1.5 GHz
Cores	8	3584
Memory Bandwidth	64 GB/s	484 GB/s
Floating-Point Calculations	409 GFLOPS	11300 GFLOPS
Cost	~ \$1000.00	~ \$700.00

<https://sebastianraschka.com/books.html>

Resource metric: **FLOPS** (floating-point operations per second)

- floating-point ops: add, mult, etc (how to weight?)
- prefixes: K (thousand), M (million), G (billion), T (trillion)
- ➔ how many TFLOPS does the above GPU provide?

# Outline

## Course Overview

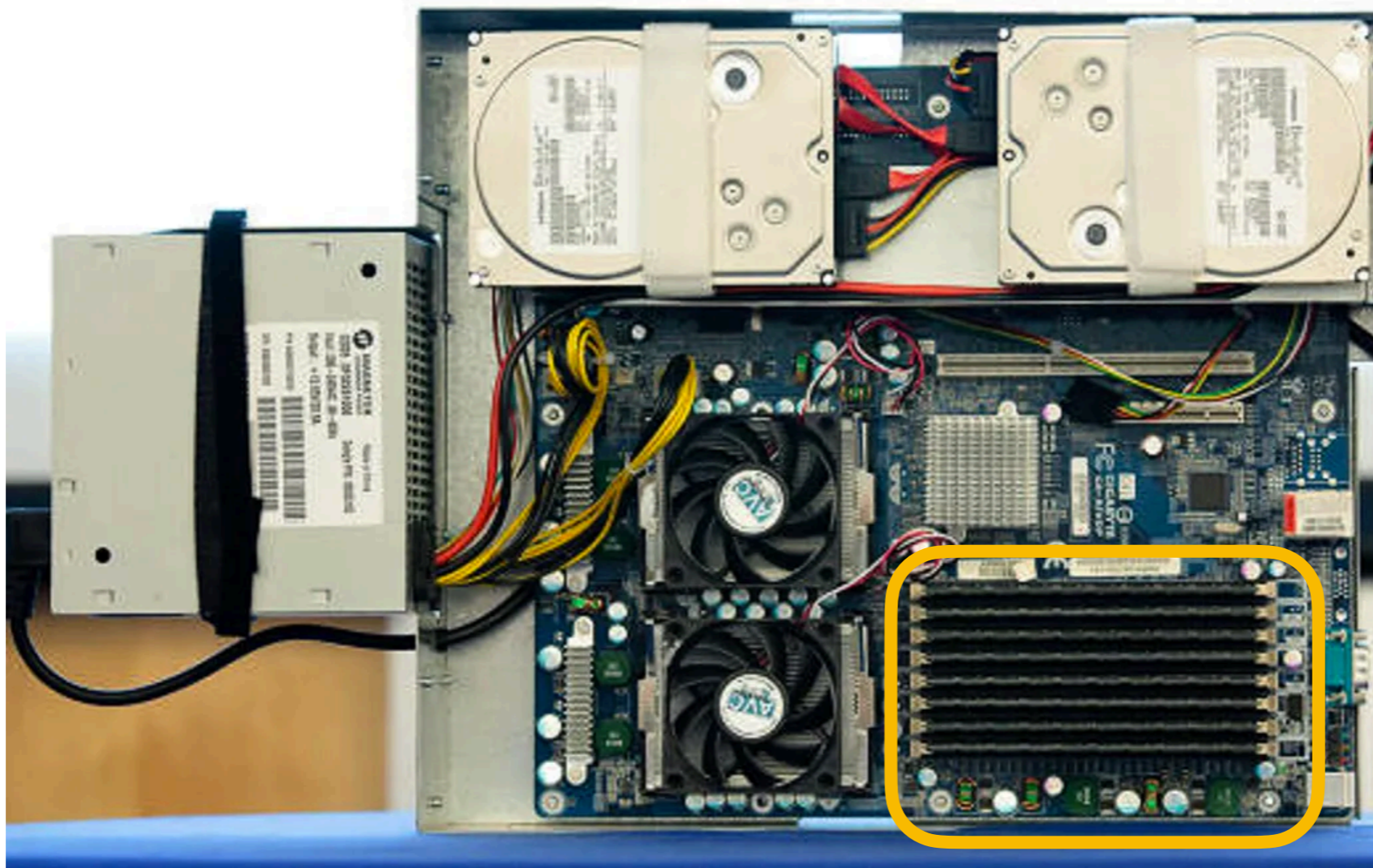
- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

## Resources

- Overview
- Compute
- **Memory**
- Storage
- Network

## Deployment

# RAM: Random Access Memory



# What is "Random"?

"Random" means we can jump around and access data from different locations efficiently.

In contrast, some devices that hold data are only efficient **sequentially**:



# Bits

RAM holds bits.

A "bit" holds a 0 or a 1 (two possible values).

2 bits can hold 00, 01, 10, 11 (four possible values).

N bits can hold  $2^N$  possible values.

# Representation

Different encodings/representations decide what a combination of bits mean.

## **bits**

000	A	0	-4	
001	B	1	-3	
010	C	2	-2	colors
011	D	3	-1	images
100	E	4	0	floats
101	F	5	1	etc.
110	G	6	2	
111	H	7	3	

# Bytes

A byte is 8 bits, so can hold  $2^8 = 256$  possible values.

RAM is "byte addressible"

- each byte of data has it's own address the CPU can use to access it
- extracting a single bit from a byte actually involves more steps than using the whole byte

Units:

- 1 KB = 1024 bytes (or sometimes 1000 bytes)
- 1 MB = 1024 KB (or sometimes 1000 KB)
- 1 GB = 1024 MB (or sometimes 1000 MB)
- 1 TB = 1024 GB (or sometimes 1000 GB)



# RAM Characteristics

## Characteristics

- **small** (for example, an e2-medium VM has 4 GB)
- **volatile** (contents lost upon reboot)
- **fast** (much faster than storage devices)

## Some uses

- actively used data (e.g., Python list, program code, DataFrame)
- copies of "hot" data (frequently accessed) from storage

# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

## Resources

- Overview
- Compute
- Memory
- **Storage**
- Network

## Deployment

# Block devices: storing 0s and 1s

Hard Disk Drives (HDDs)



- 0s/1s stored on spinning magnetized platter
- moving head reads/writes data

Solid State Disks (SSDs)



- 0s/1s stored in charged cells
- no moving parts (faster)

Both are "block devices"

- data is read/written in blocks of many bytes (for example, 0.5 KB)
- reading 1 byte or 1 block takes same time

# HDD and SSD Characteristics

## Characteristics

- **large** (> 1 TB devices are affordable)
- **nonvolatile** (contents lost upon reboot)
- **slow** (much slower than memory)

## Some uses

- large datasets
- data that needs to be preserved long term

# Metrics

## Capacity

- how much data can be stored?
- measured in bytes (for example, 500 GB)

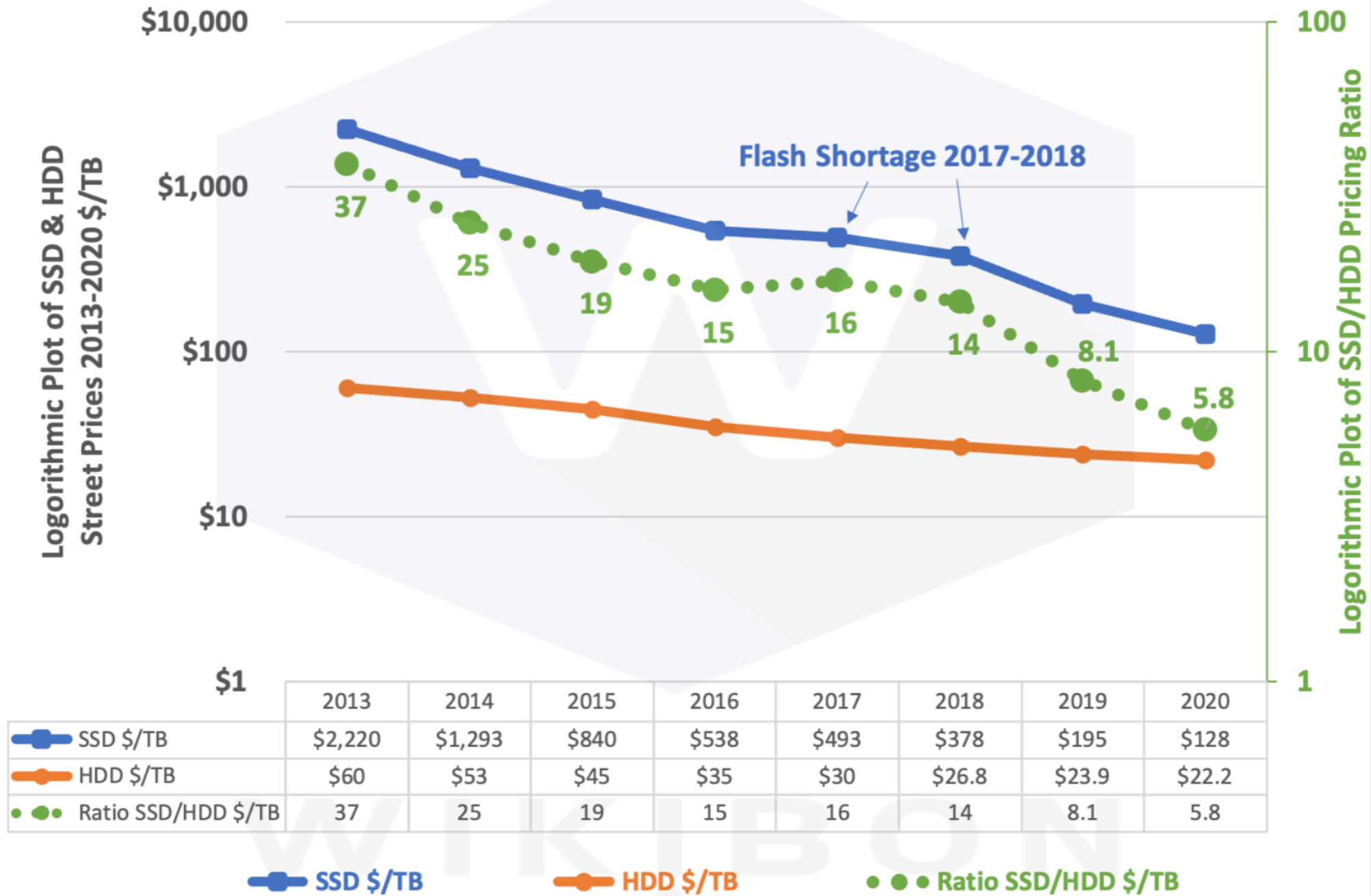
## Throughput

- how fast can data be read/written?
- measure in bytes/second (for example, 200 MB/s)
- throughput will depend on access pattern (for example, spinning disks have low throughput for random accesses)

## Latency

- how long does it take to do one I/O (e.g., 10 ms)

# Price/Terabyte of SSD & HDD are Converging Rapidly




Source: © Wikibon, 2021

Figure 3 – Flash & HDD Pricing 2013 – 2020

Source: © Wikibon, 2021. Wikibon uses historical data is from multiple sources, including IDC, Gartner, Kitguru, Nidec, Trendfocus, and Wells Fargo LLC

# Update: Amazon's "Overall Pick" in Aug 2024

**Overall Pick** ⓘ



Seagate IronWolf 8TB NAS Internal Hard Drive HDD – 3.5 Inch SATA 6Gb/s 7200 RPM 256MB Cache for RAID...

★★★★☆ 29,051  
1K+ bought in past month

**\$176<sup>54</sup>**

Or \$58.85/month for 3 months (no fees or interest)


✓prime One-Day  
FREE delivery **Tomorrow 10 AM - 3 PM**

**Add to cart**

More Buying Choices  
\$131.08 (24 used & new offers)

**\$22/TB**

**Overall Pick** ⓘ



SAMSUNG 990 EVO SSD 1TB, PCIe Gen 4x4, Gen 5x2 M.2 2280 NVMe Internal Solid State Drive, Speeds Up to...

★★★★★ 827  
10K+ bought in past month

**\$69<sup>99</sup>** List: \$149.99

Or \$23.33/month for 3 months (no fees or interest)

FREE delivery for Prime members

**Add to cart**

More Buying Choices  
\$67.19 (25 used & new offers)

**\$70/TB**  
(3.2x more expensive)

# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

## Resources

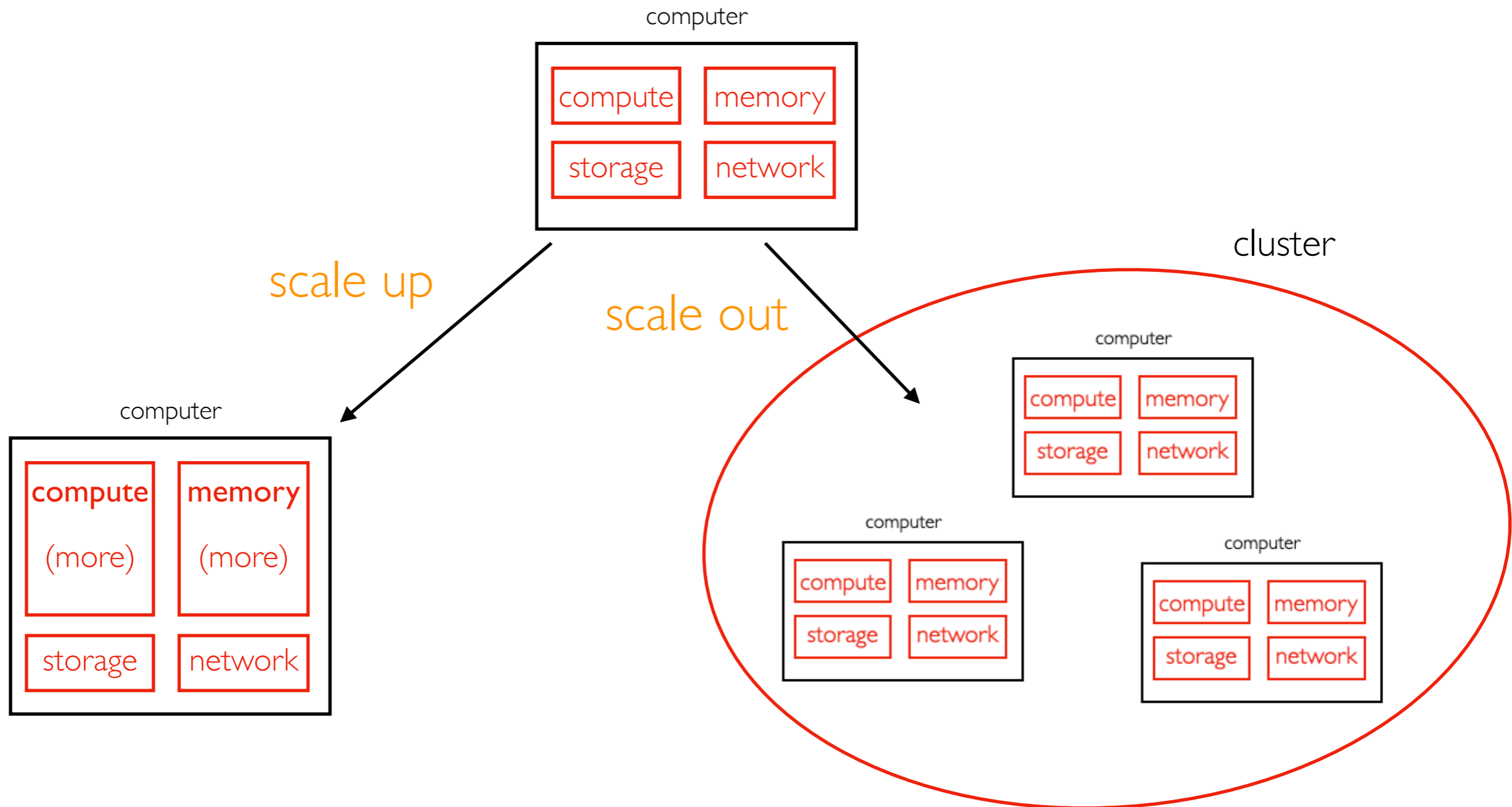
- Overview
- Compute
- Memory
- Storage
- **Network**

## Deployment



# Network

When scaling out, many **nodes** (computers) will be communicating via a network.



# Network



Server



Rack

<https://www.dotmagazine.online/issues/digital-infrastructure-and-transforming-markets/data-center-models>

[https://buy.hpe.com/us/en/servers/proliant-dl-servers/proliant-dl10-servers/proliant-dl20-server/hpe-proliant-dl20-gen10-plus-e-2336-2-9ghz-6-core-1p-16gb-u-4sff-500w-rps-server/p44115-b21?ef\\_id=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd\\_JECmdyqNMAm2CKLSm\\_F-z6jTYDTQaAgMTEALw\\_wcB:G:s&s\\_kwid=AL!13472131331628972784!!gl318267171339!!1707918369!67076417419&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd\\_JECmdyqNMAm2CKLSm\\_F-z6jTYDTQaAgMTEALw\\_wcB](https://buy.hpe.com/us/en/servers/proliant-dl-servers/proliant-dl10-servers/proliant-dl20-server/hpe-proliant-dl20-gen10-plus-e-2336-2-9ghz-6-core-1p-16gb-u-4sff-500w-rps-server/p44115-b21?ef_id=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB:G:s&s_kwid=AL!13472131331628972784!!gl318267171339!!1707918369!67076417419&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB)

[https://www.server-rack-online.com/gl910ent-4048sss.html?](https://www.server-rack-online.com/gl910ent-4048sss.html?utm_medium=shoppingengine&utm_source=googlebase&utm_source=google&utm_medium=cpc&adpos=&scid=scplpgl910ent-4048sss&sc_intid=gl910ent-4048sss&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB)

[utm\\_medium=shoppingengine&utm\\_source=googlebase&utm\\_source=google&utm\\_medium=cpc&adpos=&scid=scplpgl910ent-4048sss&sc\\_intid=gl910ent-4048sss&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd\\_JECmdyqNMAm2CKLSm\\_F-z6jTYDTQaAgMTEALw\\_wcB](https://www.server-rack-online.com/gl910ent-4048sss.html?utm_medium=shoppingengine&utm_source=googlebase&utm_source=google&utm_medium=cpc&adpos=&scid=scplpgl910ent-4048sss&sc_intid=gl910ent-4048sss&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFjsG49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB)

# Network



Server



Data Center

<https://www.dotmagazine.online/issues/digital-infrastructure-and-transforming-markets/data-center-models>

[https://buy.hpe.com/us/en/servers/proliant-dl-servers/proliant-dl10-servers/proliant-dl20-server/hpe-proliant-dl20-gen10-plus-e-2336-2-9ghz-6-core-1p-16gb-u-4sff-500w-rps-server/p/44115-b21?ef\\_id=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd\\_JECmdyqNMAm2CKLSm\\_F-z6jTYDTQaAgMTEALw\\_wcB:G:s&s\\_kwid=AL!13472131331628972784!!g318267171339!!1707918369!67076417419&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd\\_JECmdyqNMAm2CKLSm\\_F-z6jTYDTQaAgMTEALw\\_wcB](https://buy.hpe.com/us/en/servers/proliant-dl-servers/proliant-dl10-servers/proliant-dl20-server/hpe-proliant-dl20-gen10-plus-e-2336-2-9ghz-6-core-1p-16gb-u-4sff-500w-rps-server/p/44115-b21?ef_id=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB:G:s&s_kwid=AL!13472131331628972784!!g318267171339!!1707918369!67076417419&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB)

[https://www.server-rack-online.com/gl910ent-4048sss.html?](https://www.server-rack-online.com/gl910ent-4048sss.html?utm_medium=shoppingengine&utm_source=googlebase&utm_source=google&utm_medium=cpc&adpos=&scid=scplpg910ent-4048sss&sc_intid=gl910ent-4048sss&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB)

[utm\\_medium=shoppingengine&utm\\_source=googlebase&utm\\_source=google&utm\\_medium=cpc&adpos=&scid=scplpg910ent-4048sss&sc\\_intid=gl910ent-4048sss&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd\\_JECmdyqNMAm2CKLSm\\_F-z6jTYDTQaAgMTEALw\\_wcB](https://www.server-rack-online.com/gl910ent-4048sss.html?utm_medium=shoppingengine&utm_source=googlebase&utm_source=google&utm_medium=cpc&adpos=&scid=scplpg910ent-4048sss&sc_intid=gl910ent-4048sss&gclid=Cj0KCQiAt66eBhCnARIsAKf3ZNFJsg49UV6Zm33R7lkRqi-XOd_JECmdyqNMAm2CKLSm_F-z6jTYDTQaAgMTEALw_wcB)

# Topology

Example configuring Hadoop File System (HDFS) to store data based on network topology:

## python Example

```
#!/usr/bin/python3
# this script makes assumptions about the physical environment.
# 1) each rack is its own layer 3 network with a /24 subnet, which
# could be typical where each rack has its own
# switch with uplinks to a central core router.
#
#
#          +-----+
#          |core router|
#          +-----+
#         /         \
# +-----+         +-----+
# |rack switch|     |rack switch|
# +-----+         +-----+
# | data node |     | data node |
# +-----+         +-----+
# | data node |     | data node |
# +-----+         +-----+
#
#
```

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/RackAwareness.html>

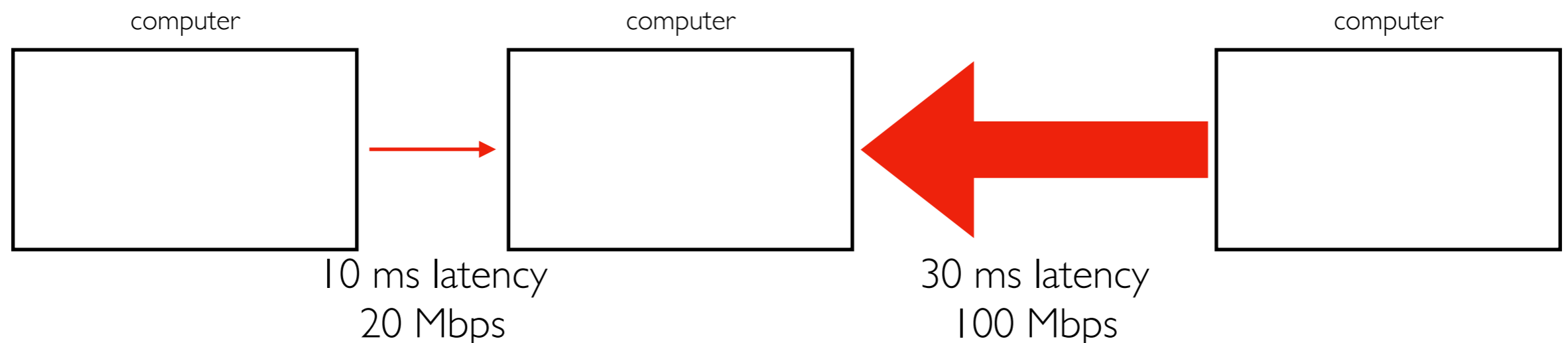
# Metrics

## Latency

- how long does it take to send messages between two points
- seconds, milliseconds (ms), etc

## Bandwidth/Throughput

- how many **bits** can be sent per second?
- Mbps (mega bits per second -- note lower case "b")
- What is faster, 10 Mbps or 10 MB/s?



# Outline

## Course Overview

- Introductions
- Main sites: [tyler.caraza-harter.com](http://tyler.caraza-harter.com), Canvas, GitHub
- Other tools: Email, TopHat, Piazza, GitLab

## Resources

- Overview
- Compute
- Memory
- Storage
- Network

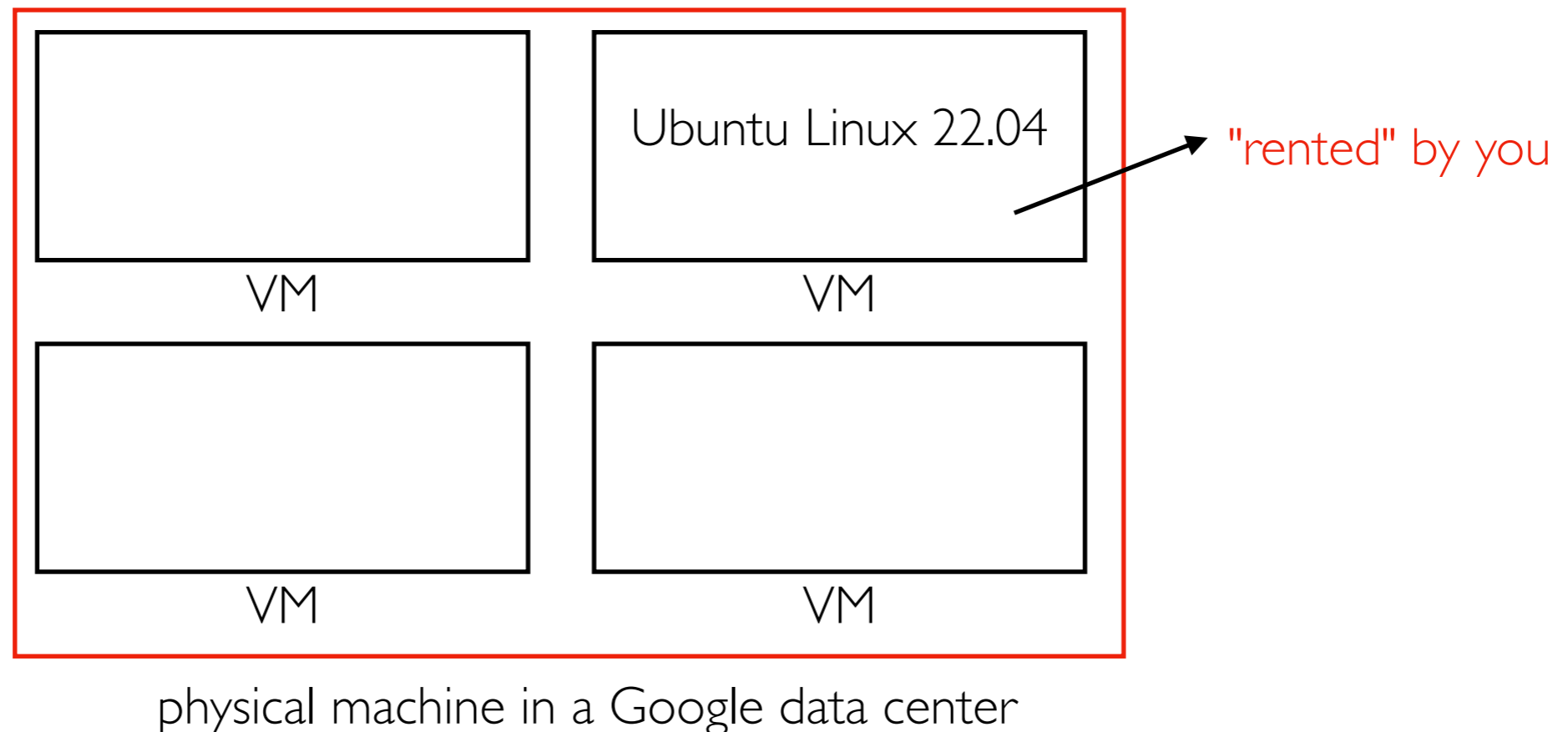
## Deployment

# Deployment

Deployment means running code somewhere

- often a major undertaking when working with clusters

We'll be deploying systems on Google Cloud Platform (GCP) this semester using virtual machines (VMs).



# Cloud Budget Plan

We'll be announcing how to get \$50 of free credit for the semester.

You'll use different VMs/resources at different points. Follow the plan here to create the right types.

<https://github.com/cs544-wisc/f24/blob/main/projects.md#compute-setup>

Important: check your credits regularly to make sure you're not using too much too soon based on the "cumulative" column. If you accidentally configured to use too much, fix the issue, then find ways to catch up (like shutting down your VM overnight).



# VM Maintenance Responsibilities

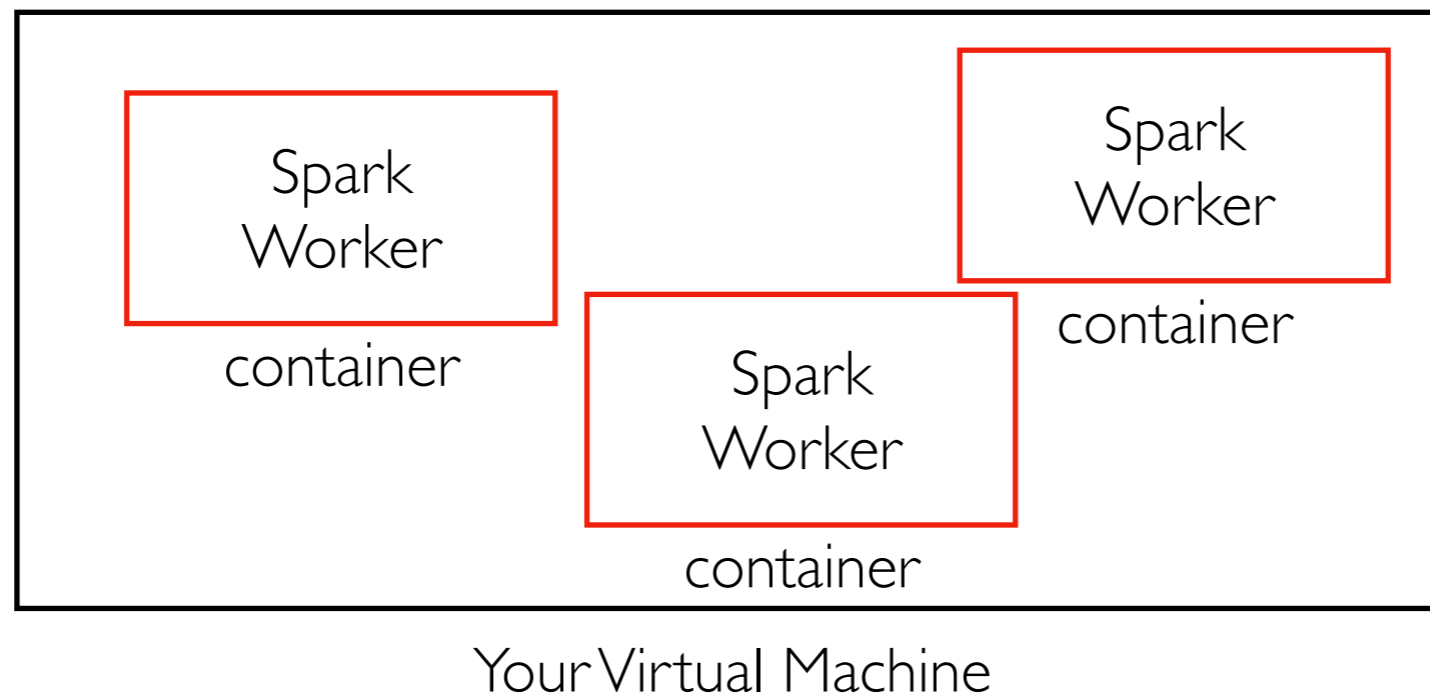
Some of your responsibilities:

- creating/re-creating the VM (choosing a region, etc)
- setting up SSH
- installing software
- rebooting after updates
- backing up work on GitHub (private) or with scp
- resolving issues directly with Google (e.g., false positive crypto mining)

# Docker containers

Containers are a lightweight alternative to virtual machines.

You'll run Docker containers this semester to have your own "mini cluster"



Resources of the "cluster" are limited to those of a single VM, so we'll scale projects accordingly. But the techniques will apply to large clusters and datasets.

# Conclusion

Systems manage **resources** like compute, memory, storage, and networking.

Big data systems use specialized or distributed resources to make it faster to work with large datasets.

We'll **deploy** these systems using containers and VMs.

Tasks for next time:

- read syllabus, become familiar with course websites
- introduce yourself via the welcome form
- redeem Google credit and create first VM