Week 5: performance | QOL

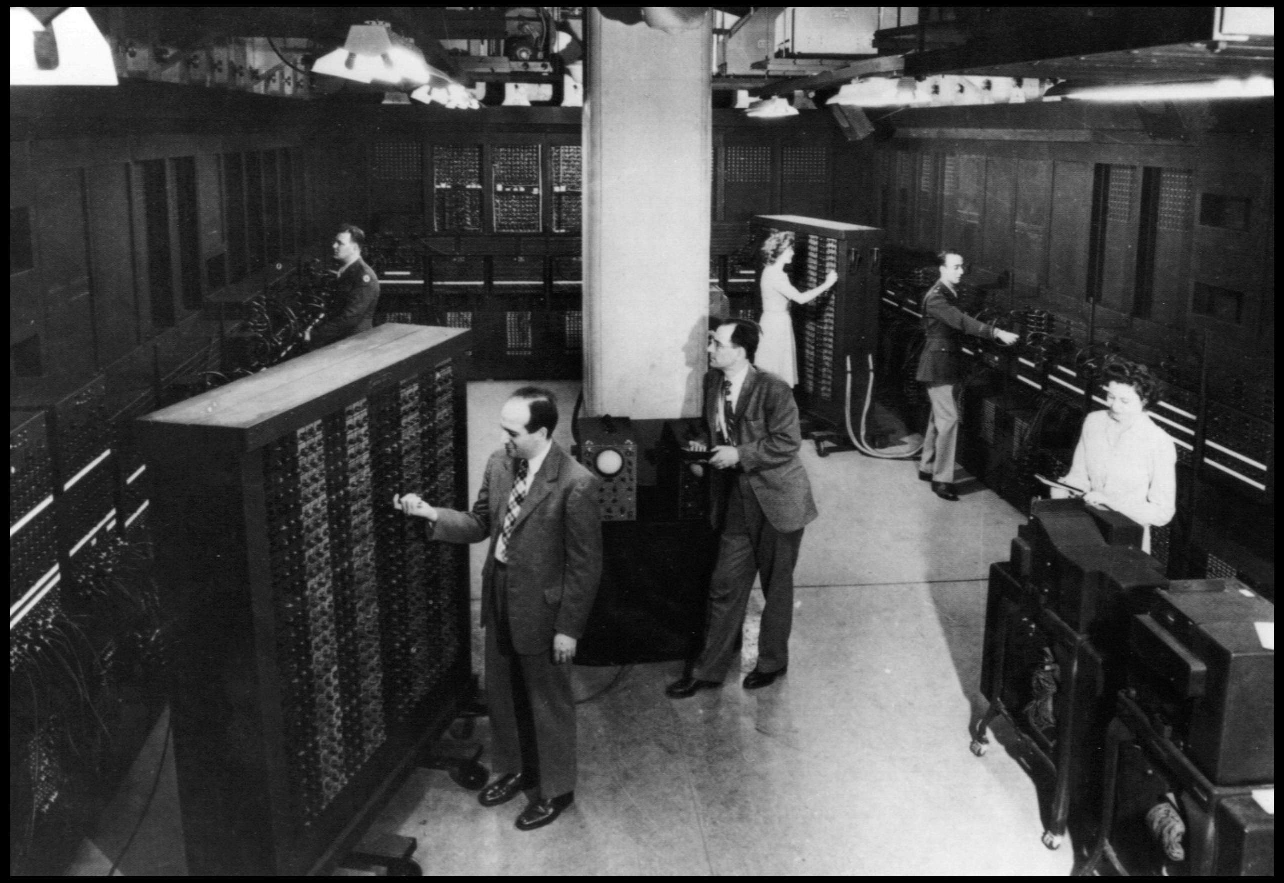
NRSC 7657 Workshop in Advanced Programming for Neuroscientists

course business

Performance

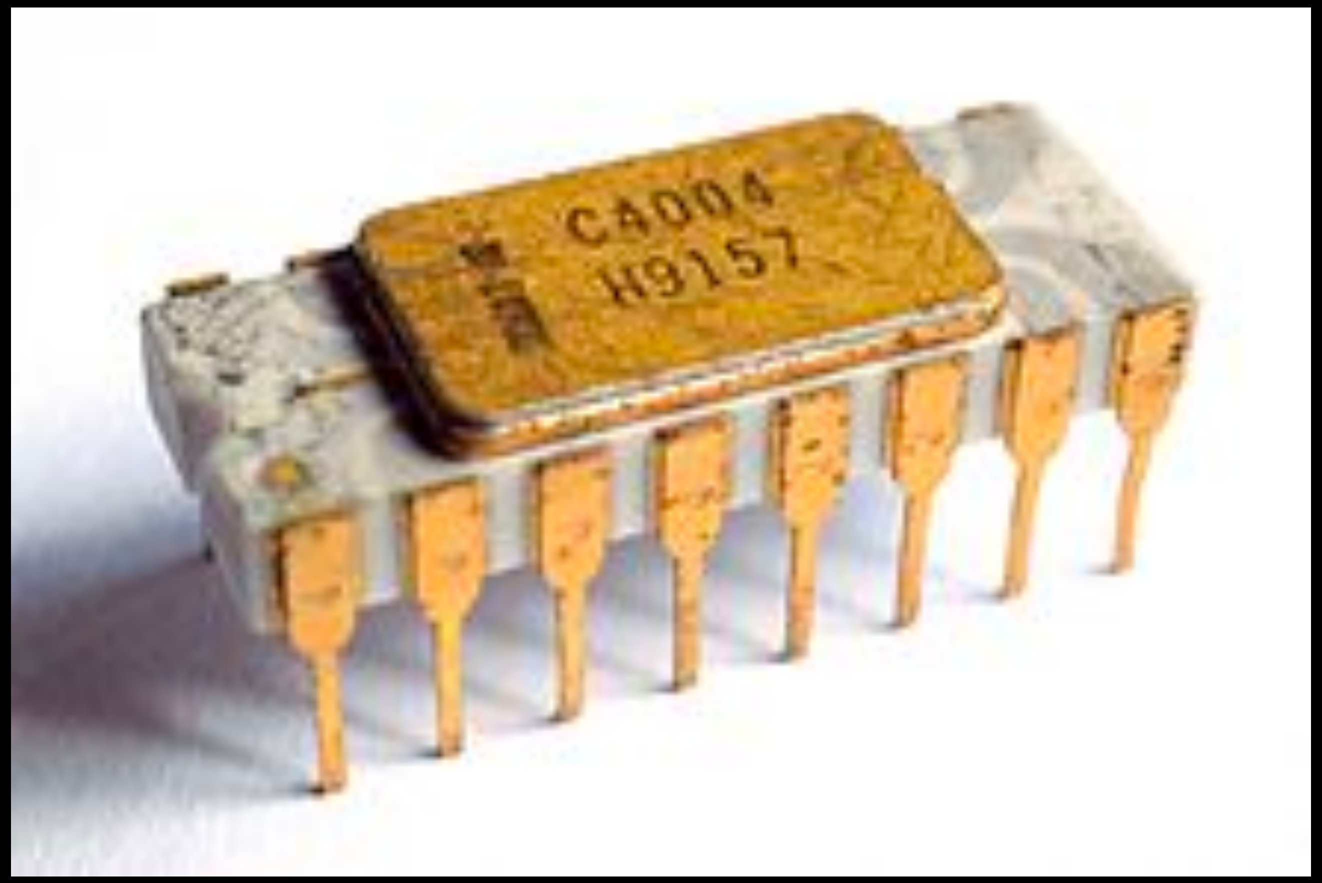
Processors: digital computers

vacuum tubes



ENIAC
18,000 vacuum tubes
5,000,000 hand-soldered joints
27 tons
150 kW

Transistors



Intel 4004
2250 transistors
MOS gates
< 27 tons
1W

Integrated circuits: CPUs

your computer, your phone
< 100,000 transistors
many Integrates gates
also < 27 tons
~100s of watts

Performance

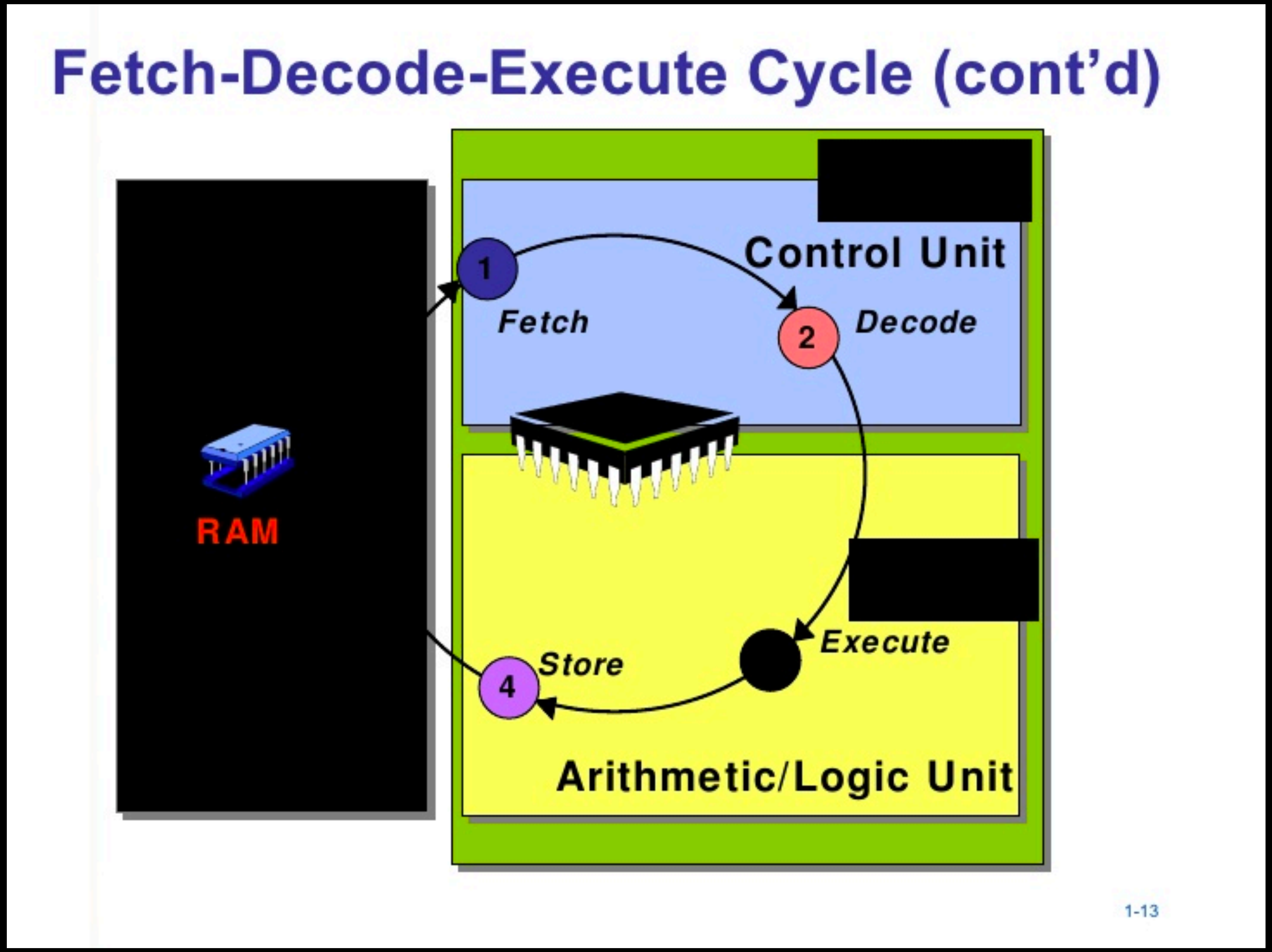
Processors: CPU

- The CPU is (mostly) what is doing the heavy lifting when you ask python to do some calculation.
- The python interpreter converts your code to machine language (a set of instructions for what memory addresses to operate on), and then:
    1. Puts that machine language set of instruction in RAM
    2. the CPU does its thing (fetch-decode-execute-store)

Performance

Processors: CPU machine cycle

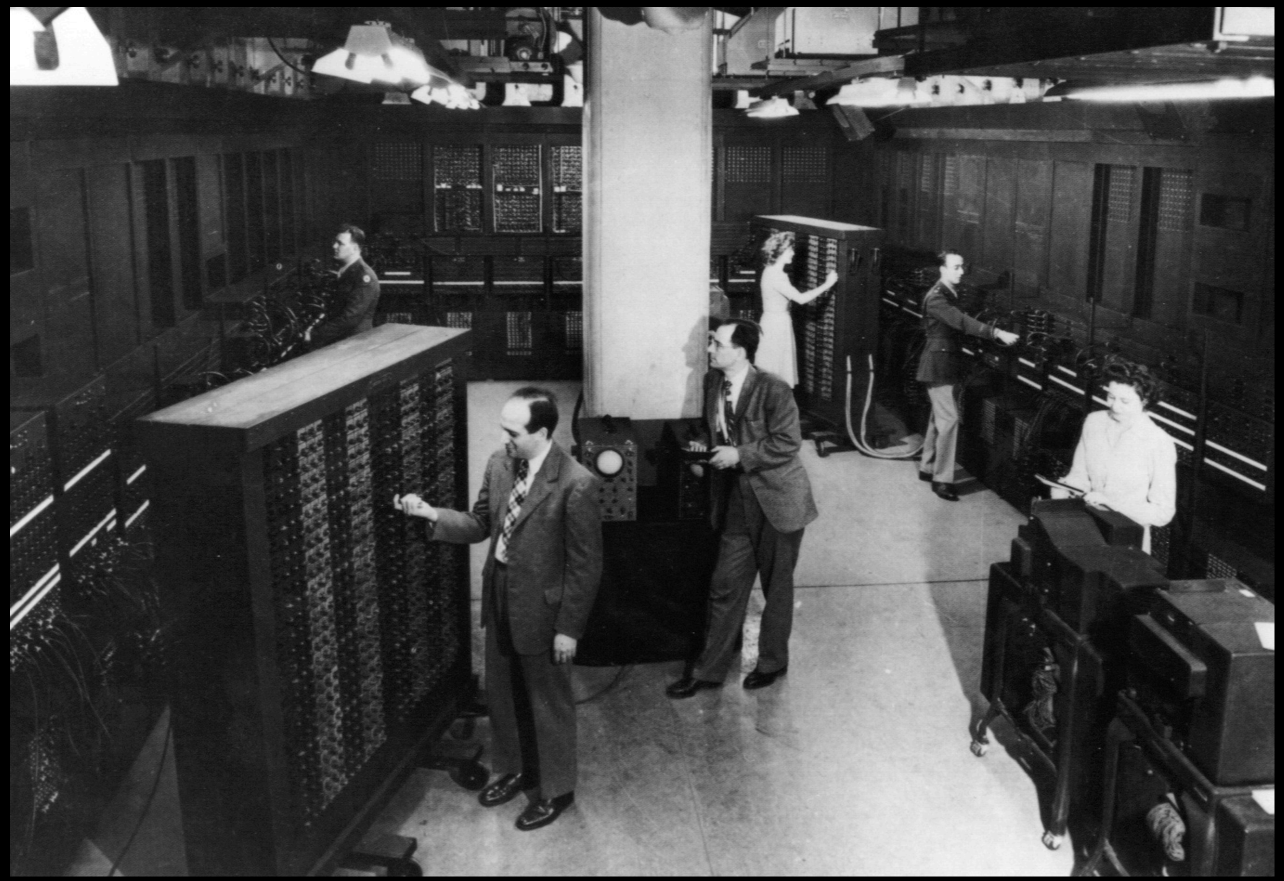
- Clock speed



https://www.slideshare.net/kitturashmikittu/computer-basics-and-io-devices

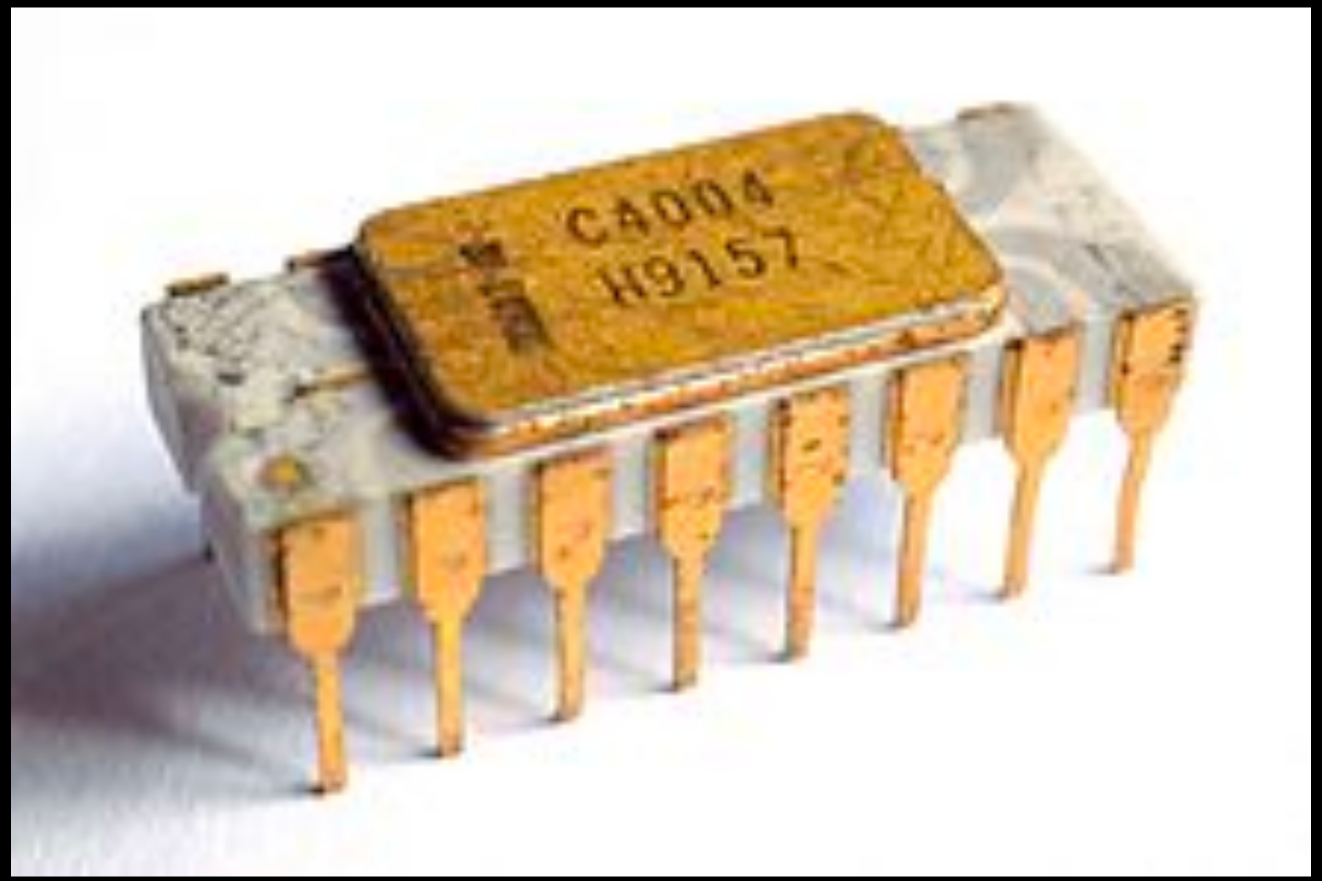
Performance
Processors: digital computers

vacuum tubes



ENIAC
100 kHz
Machine cycle: 200 µs

Transistors



Intel 4004
740 kHz
Machine cycle: 10.8 µs

Integrated circuits: CPUs

your computer, your phone
2 - 4Ghz
0.3 - 3 ns

Performance

Iteration: serial just like any other line

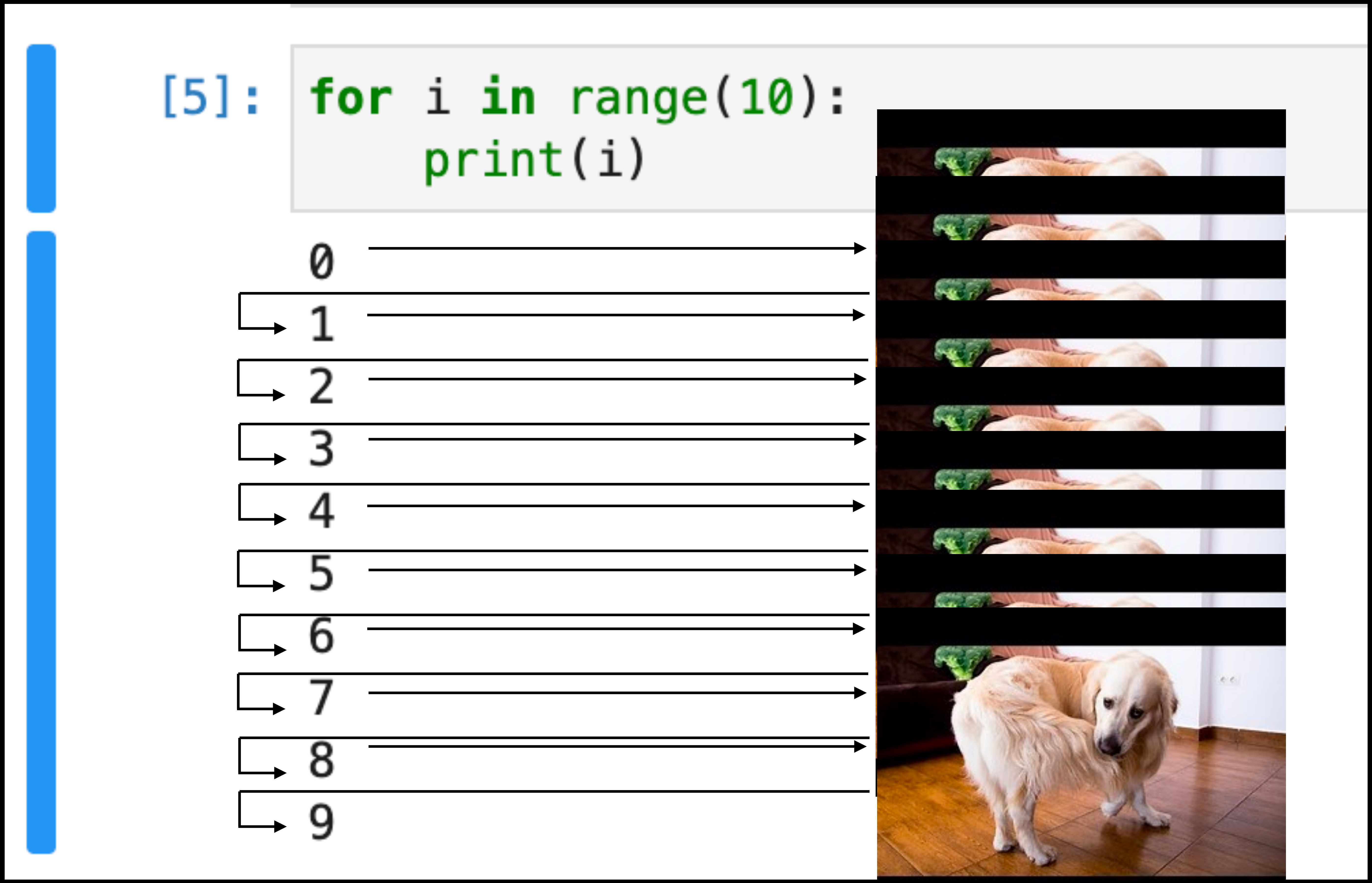
```
[2]: a = 2
```

```
[3]: b= 2
```

```
[4]: c = a + b
```

Performance

Iteration: serial just like any other line



Performance

Iteration: serial just like any other line

```
[7]:  print(0)
      0
[8]:  print(1)
      1
[9]:  print(2)
      2
[10]: print(3)
      3
[11]: print(4)
      4
[12]: print(5)
      5
[13]: print(6)
      6
[14]: print(7)
      7
[15]: print(8)
      8
[16]: print(9)
      9
```

Performance

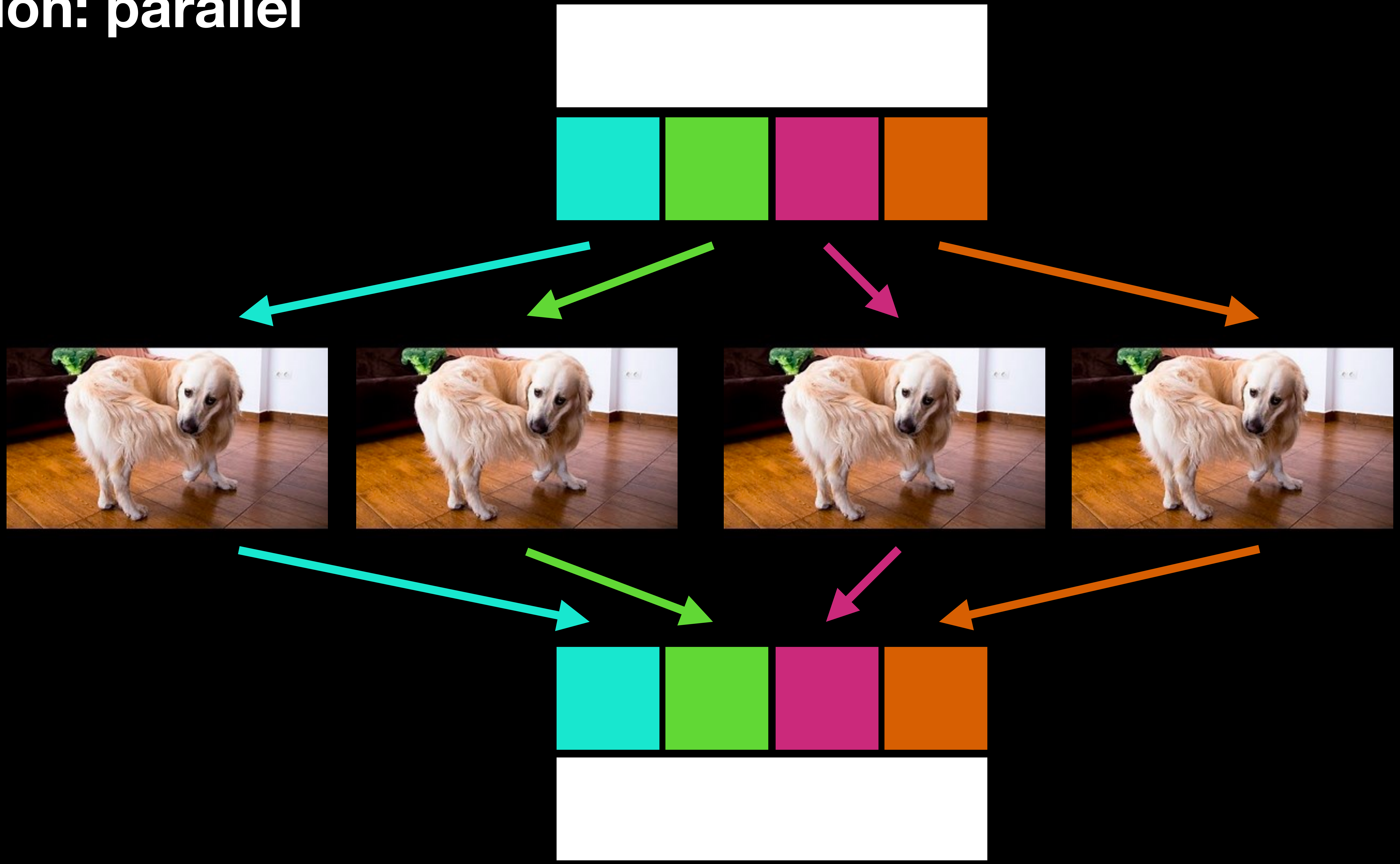
Iteration: data types, speeding it up

- Sorting tuples
- filtering
- Built-in methods (including filtering)
- List comprehension
- Avoiding it (but don’t try too hard unless you need to)

[to be covered in .ipynb]

Performance

Iteration: parallel



Performance

Parallel CPU usage

- Split a computation into parts, ship it off to a number of processors, stitch the results back together
- Not all computations benefit from this, but some do. The computations that be broken into parts that don't depend on each other are called **"embarrassingly parallel"**
  - **Arrays where you do the same thing to chunks**
  - **Deep learning**

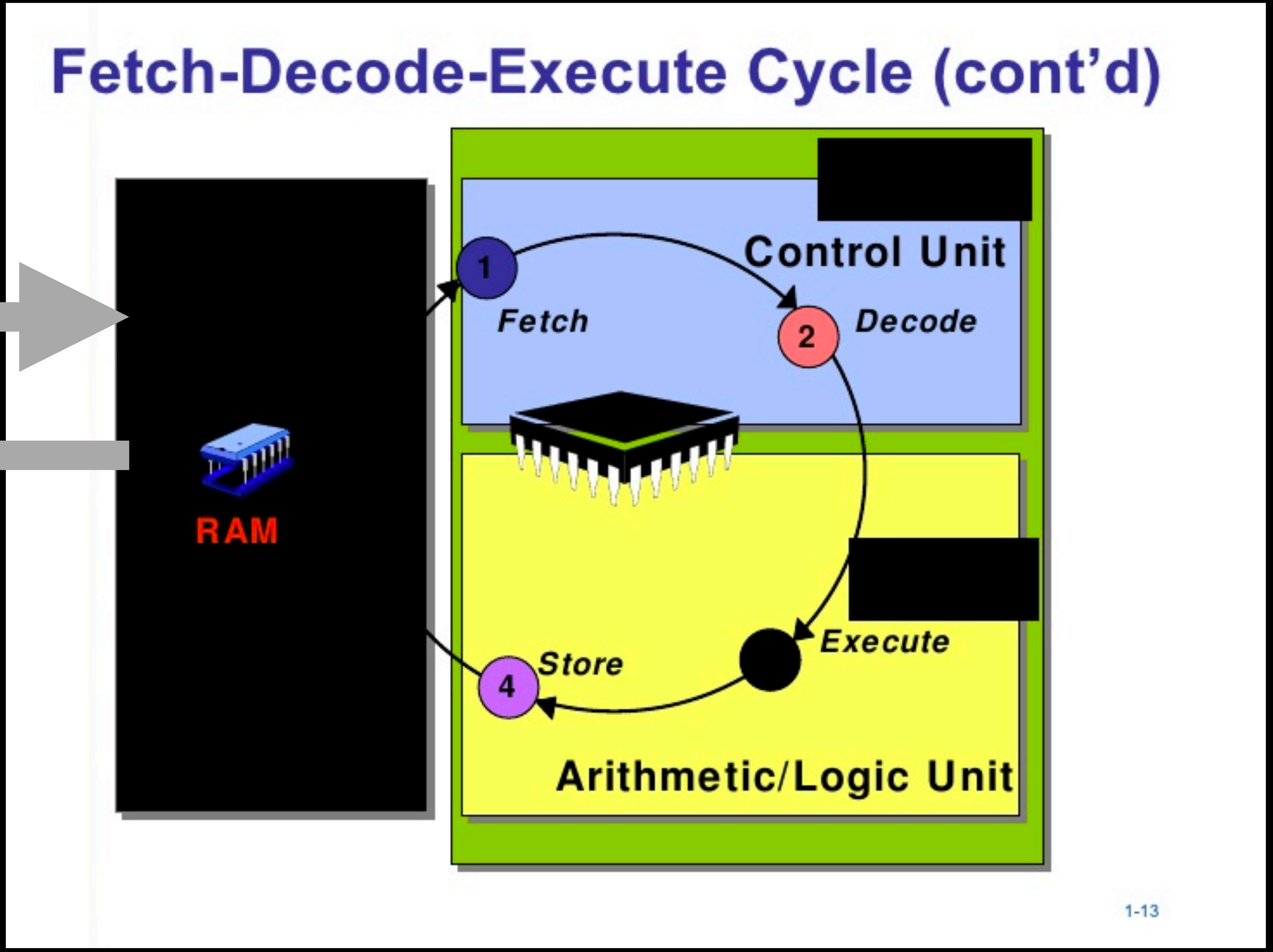
- `multiprocessing`
- `joblib`
- `asyncio`
- `dask`

Performance

Parallel CPU usage

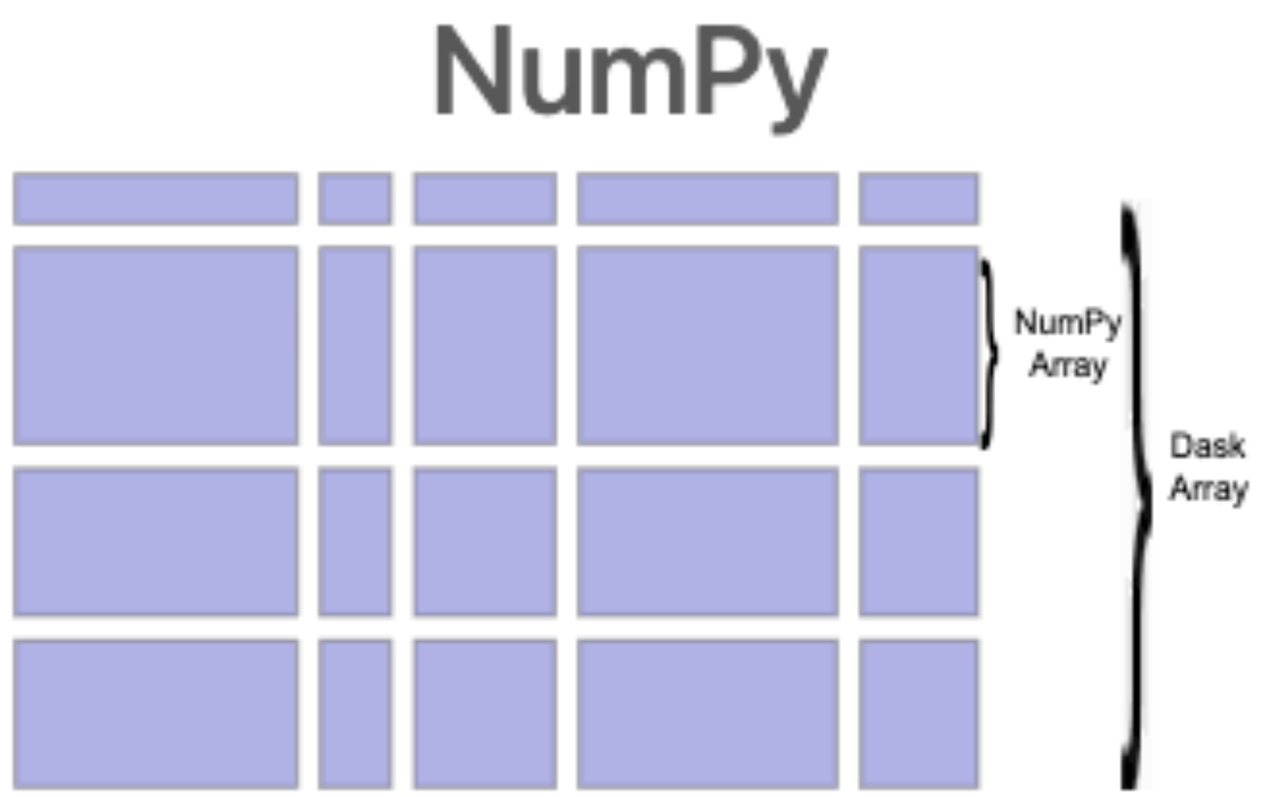
With standard package helpers

- `multiprocessing`
- `joblib`
- `asyncio`
- `dask`

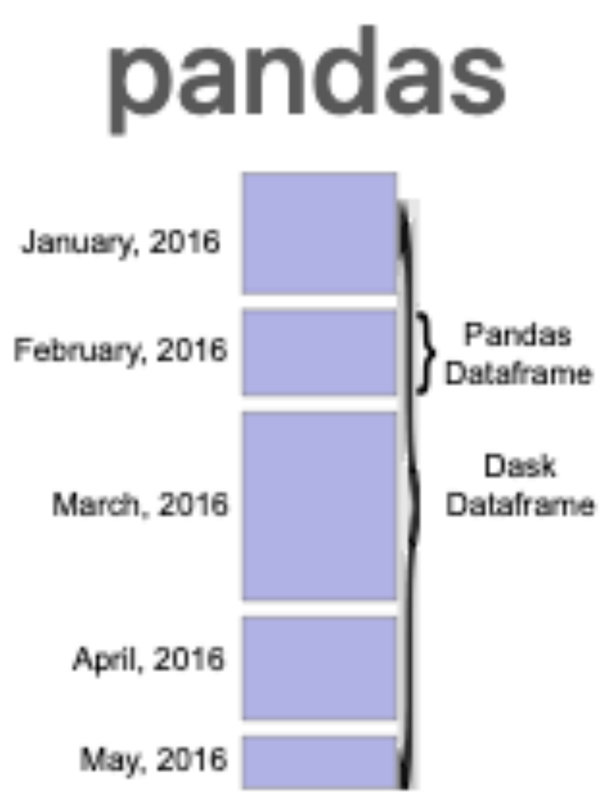


Performance

Parallel CPU usage: `dask`



Dask arrays scale NumPy workflows, enabling multi-dimensional data analysis in earth science, satellite imagery, genomics, biomedical applications, and machine learning algorithms.



Dask dataframes scale pandas workflows enabling applications in time series, business intelligence, and general data munging on big data.

Google Cloud VMs

class **dask_cloudprovider.gcp.GCPCluster**(*projectid=None, zone=None, network=None, machine_type=None, on_host_maintenance=None, source_image=None, docker_image=None, ngpus=None, gpu_type=None, filesystem_size=None, disk_type=None, auto_shutdown=None, bootstrap=True, preemptible=None, debug=False, **kwargs*) [source]

Cluster running on GCP VM Instances.

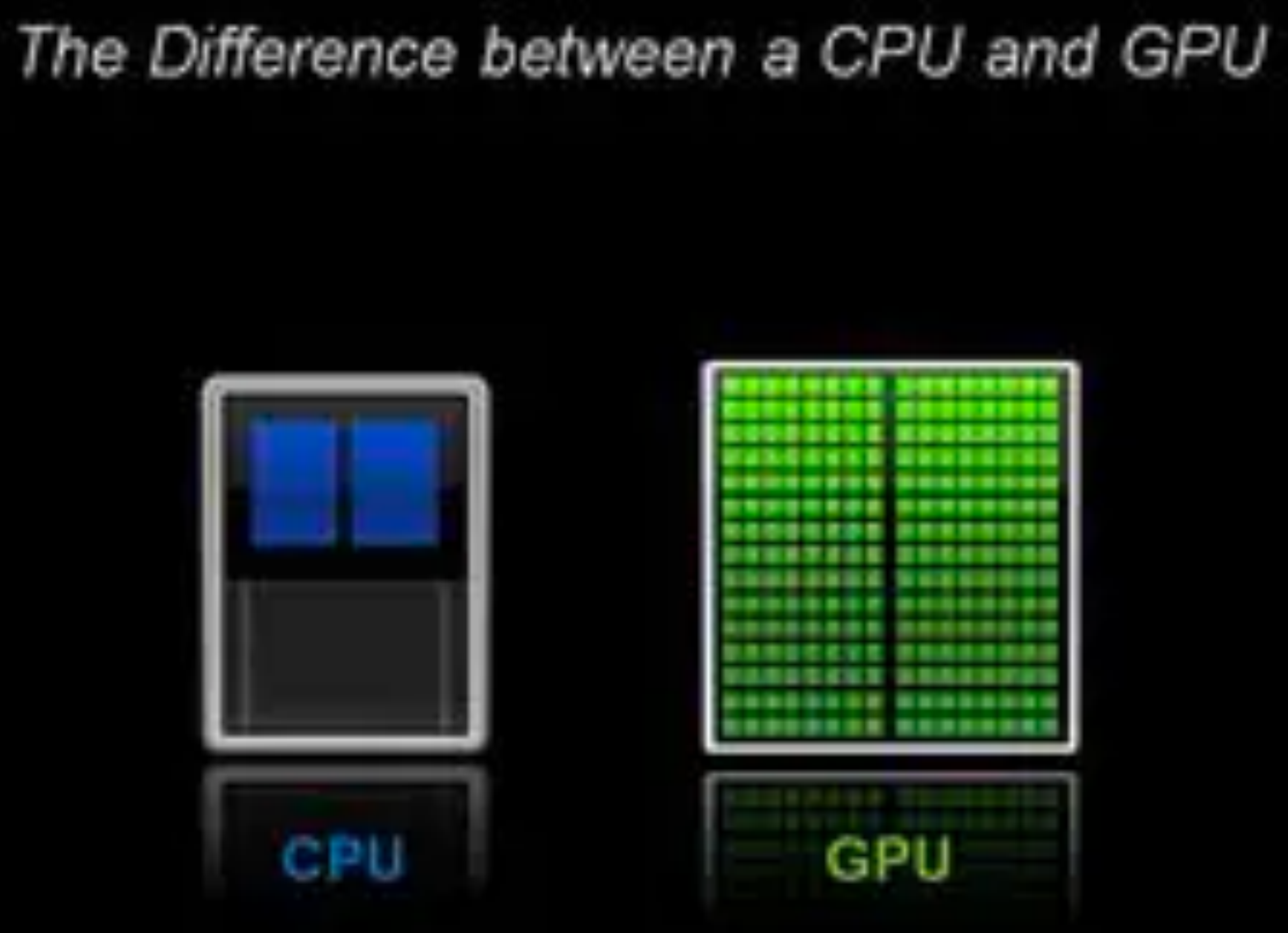
```python
# Arrays implement the NumPy API
import dask.array as da
x = da.random.random(size=(10000, 10000),
                     chunks=(1000, 1000))
x + x.T - x.mean(axis=0)
```

```python
# Dataframes implement the pandas API
import dask.dataframe as dd
df = dd.read_csv('s3://.../2018-*-*.csv')
df.groupby(df.account_id).balance.sum()
```

```python
# Dask-ML implements the scikit-learn API
from dask_ml.linear_model \
  import LogisticRegression
lr = LogisticRegression()
lr.fit(train, test)
```

Performance

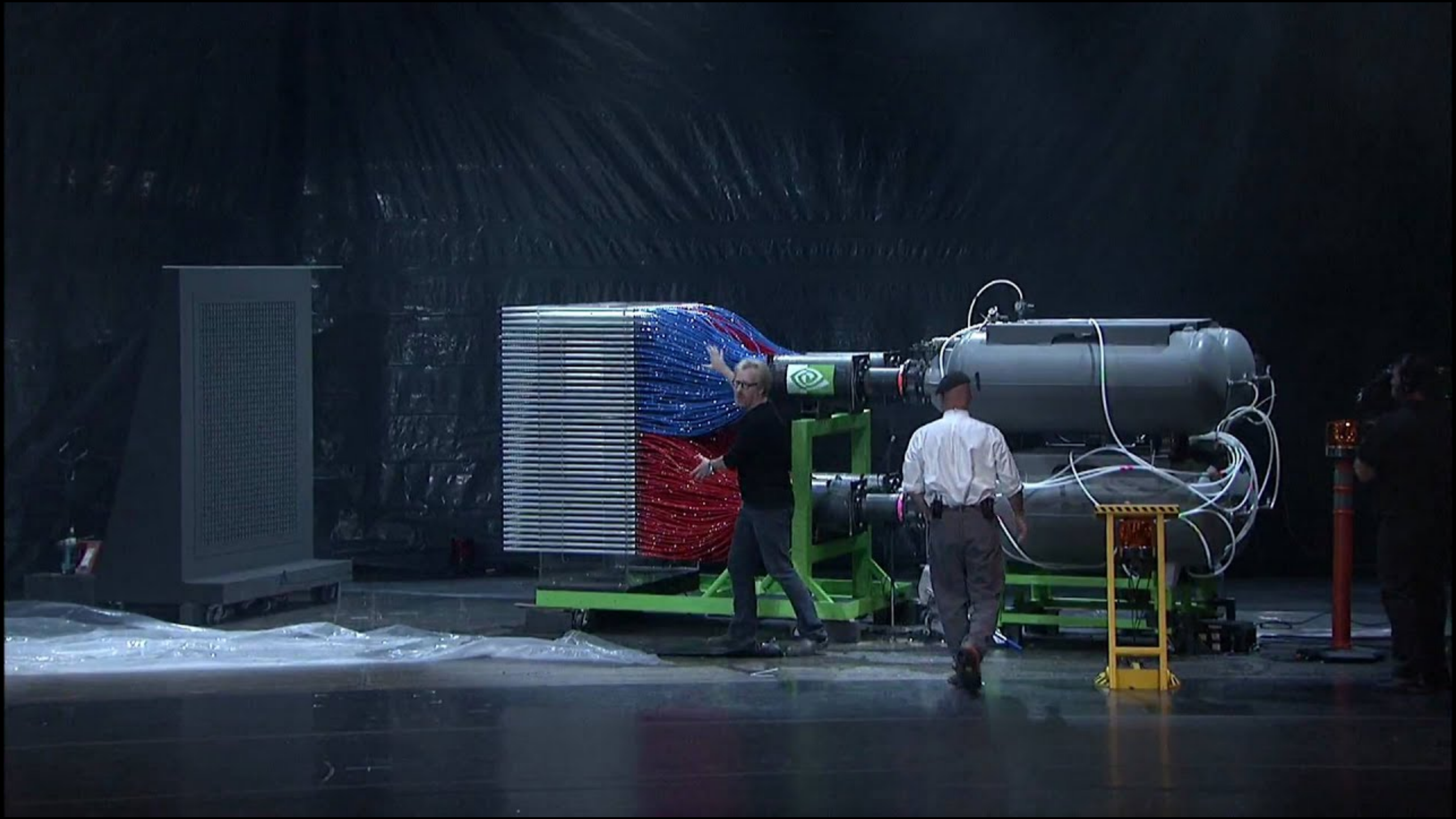
GPUs: many tiny cores for parallel things



- nVidia

Performance

GPUs: many tiny cores for parallel things



nVidia, https://youtu.be/-P28LKWTzrI

Performance

GPUs: many tiny cores for parallel things

- `PyTorch`
- `TensorFlow`
- MATLAB: Parallel Processing

Performance

Profiling

- `%%time` and `%%timeit`
  - Note: `%%timeit` will run multiple times by default, so use some caution

Code. QOL

Tips and tricks

- `tqdm`
- `os, sys, glob`
- `pprint`