On the road to the SKA interdisciplinary workshop

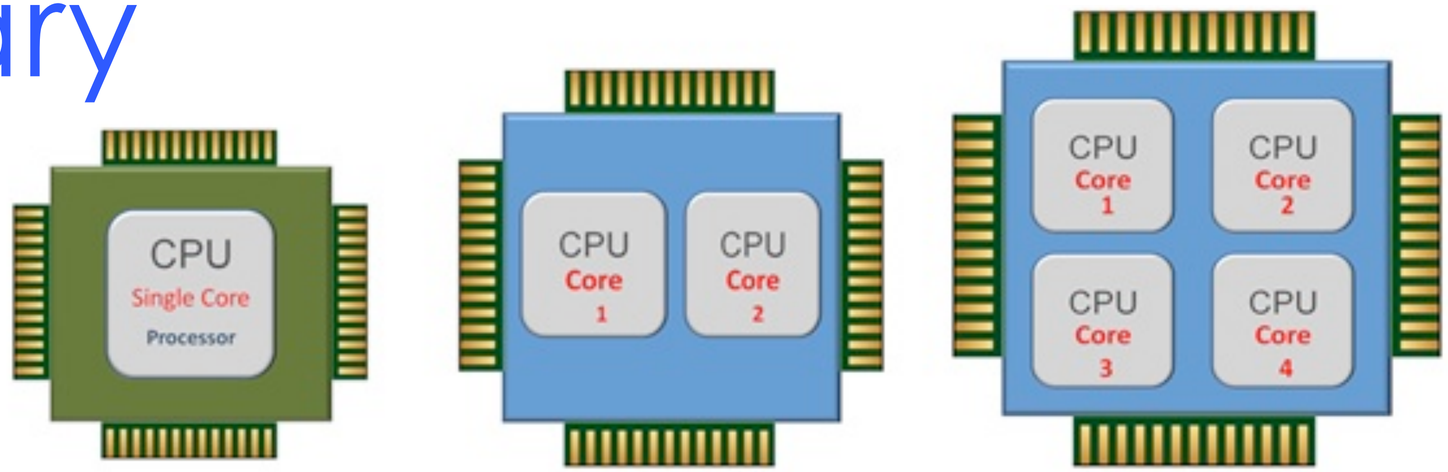
High-Performance Computing with Python

High performance computing (HPC)

- **HPC**: minimize time-to-solution of software workflow
- **Acceleration**: On a single computer, make a process run as quickly as possible
- **Parallelization**: Use many processes running at the same time

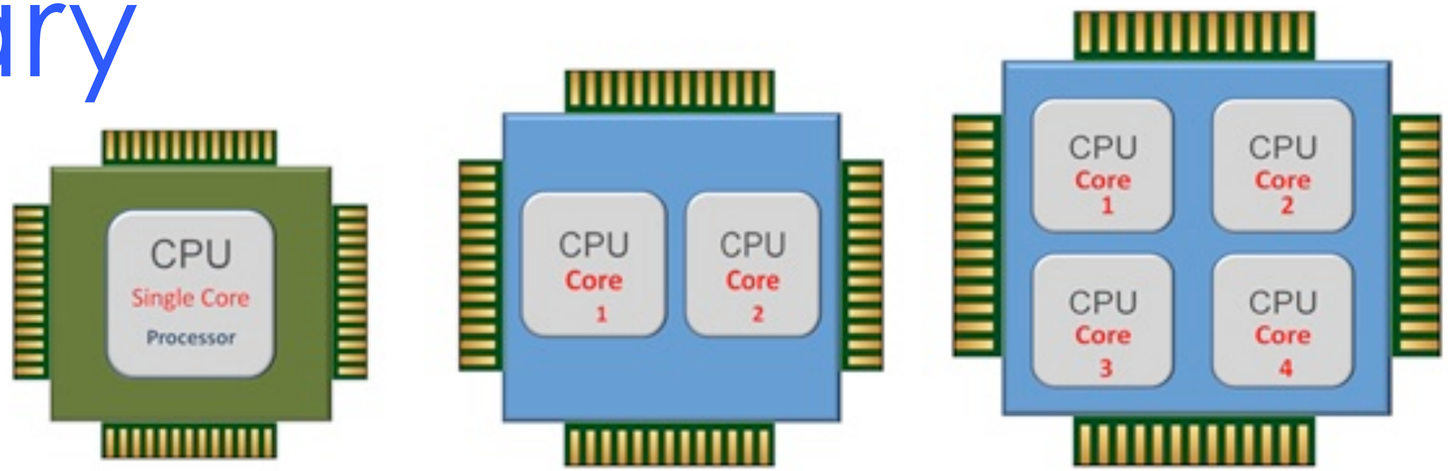
HPC vocabulary

- **Core**: The fundamental execution unit. A modern CPU has multiple cores.
- **Processor (Socket)**: The physical chip that houses one or more cores.



HPC vocabulary

- **Core**: The fundamental execution unit. A modern CPU has multiple cores.
- **Processor (Socket)**: The physical chip that houses one or more cores.

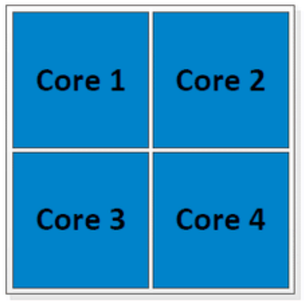


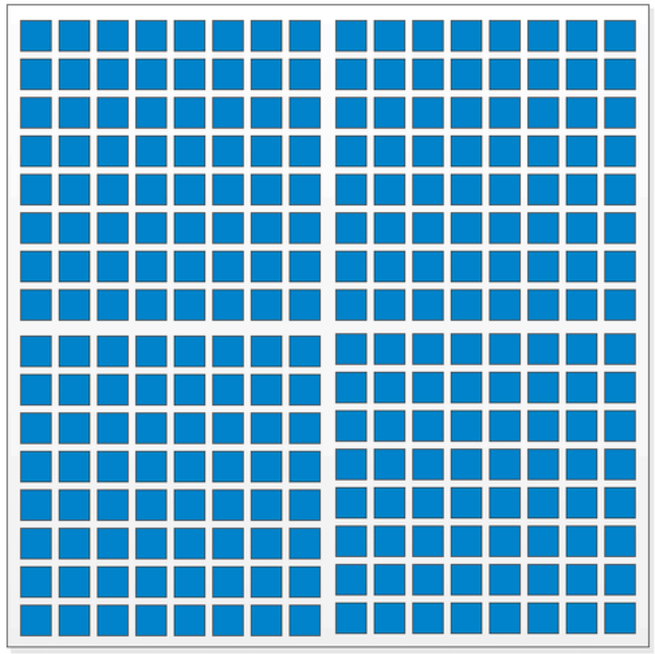
GPU

(Hundreds of cores)

CPU

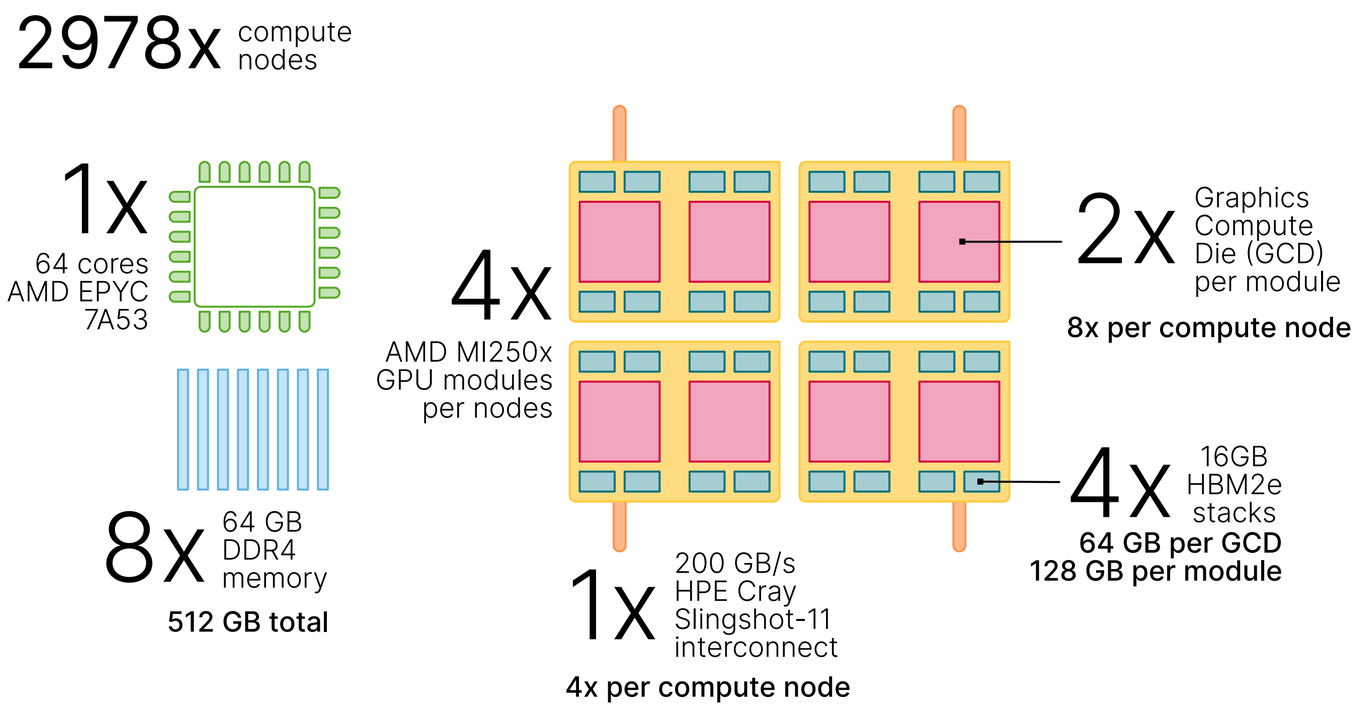
(Multiple cores)





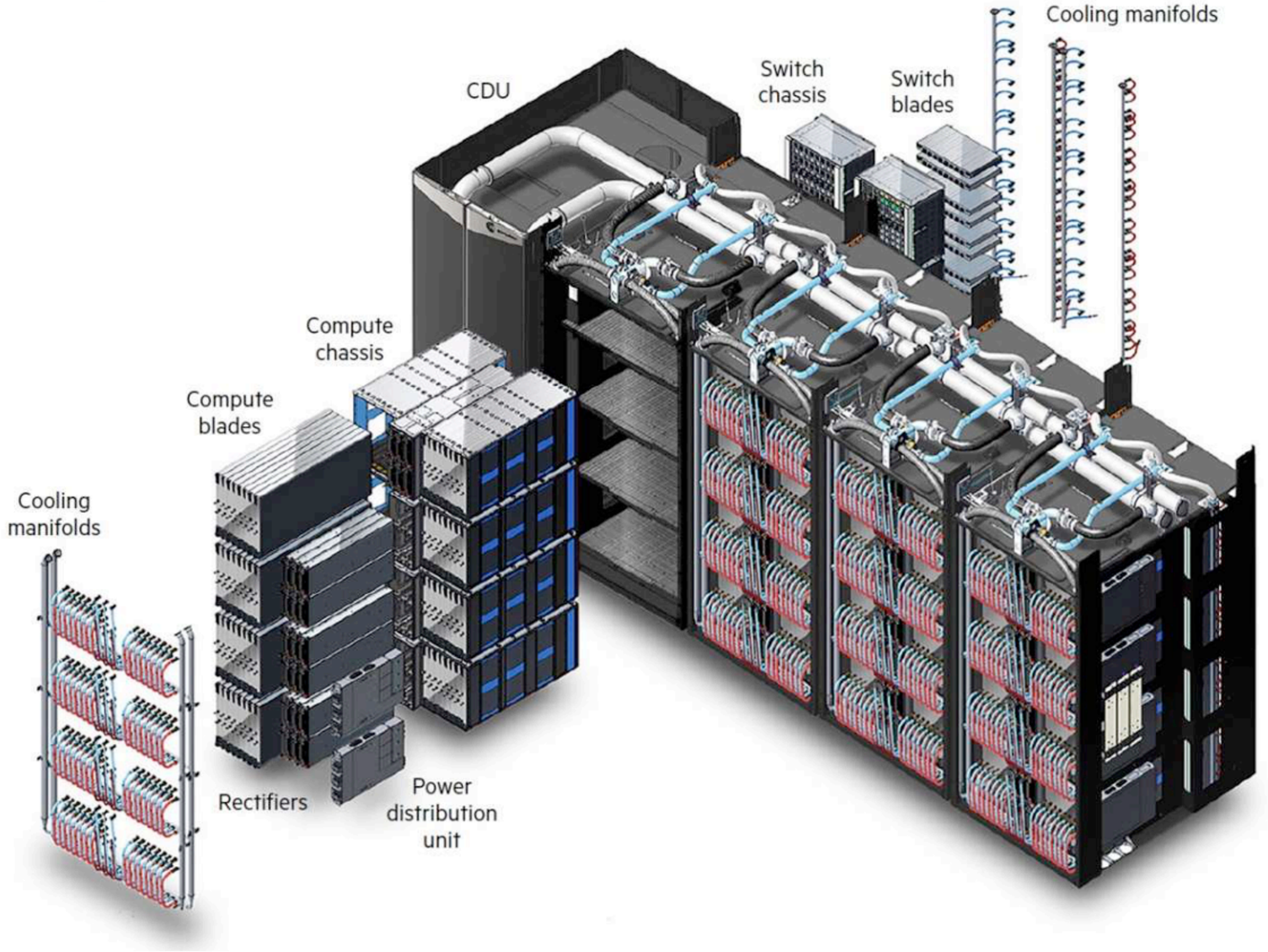
HPC vocabulary

- **Core**: The fundamental execution unit. A modern CPU has multiple cores.
- **Processor (Socket)**: The physical chip that houses one or more cores.
- **Node**: A single physical server containing processors and memory.

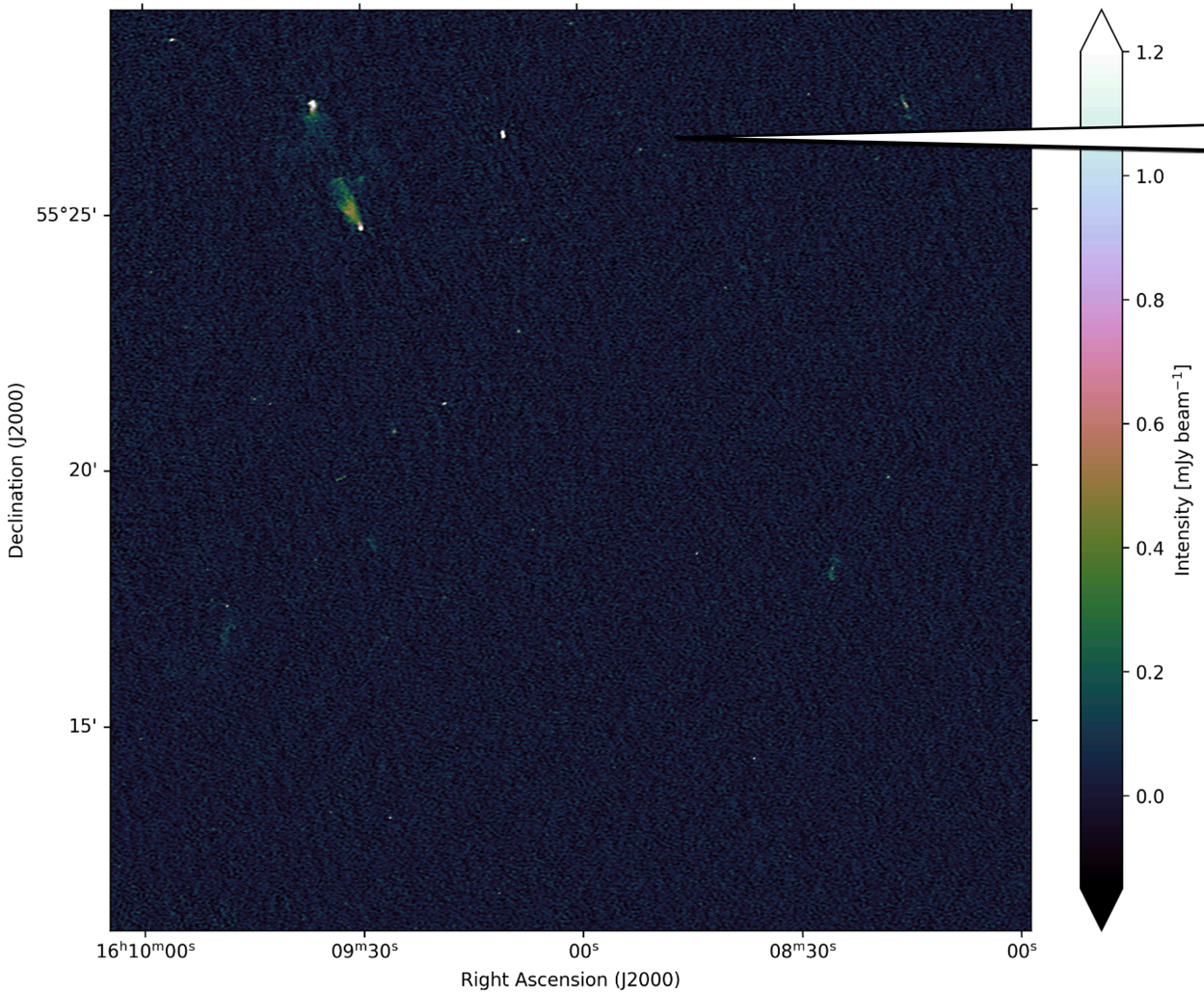


HPC vocabulary

- **Core**: The fundamental execution unit. A modern CPU has multiple cores.
- **Processor (Socket)**: The physical chip that houses one or more cores.
- **Node**: A single physical server containing processors and memory.
- **Cluster**: A collection of many nodes networked together.

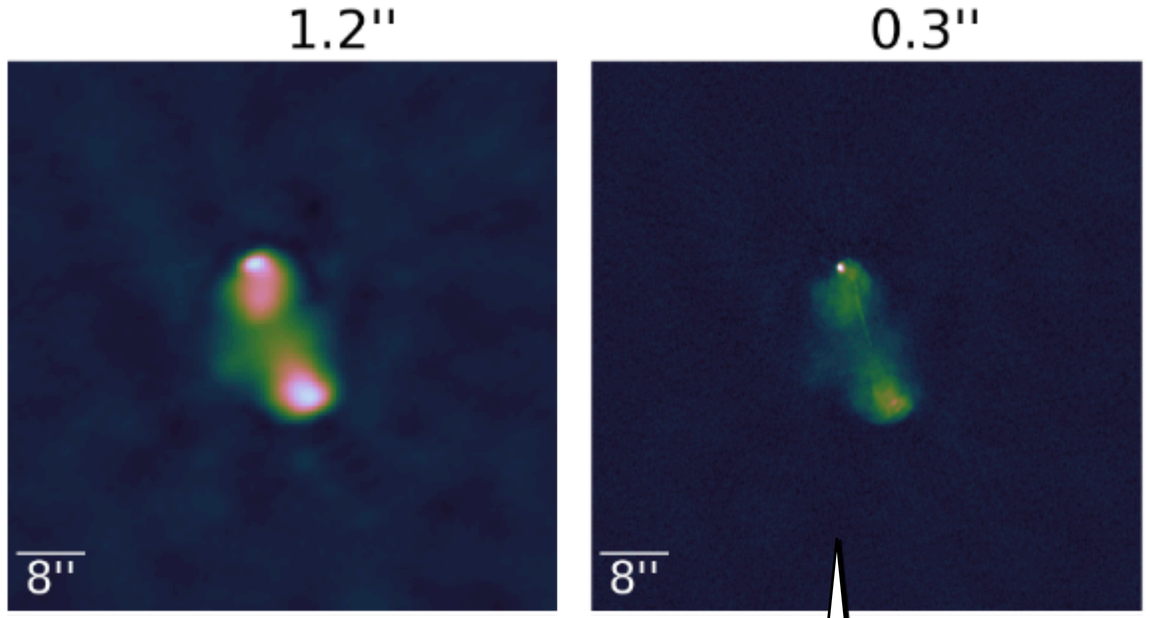


HPC Challenge of Radio astronomy



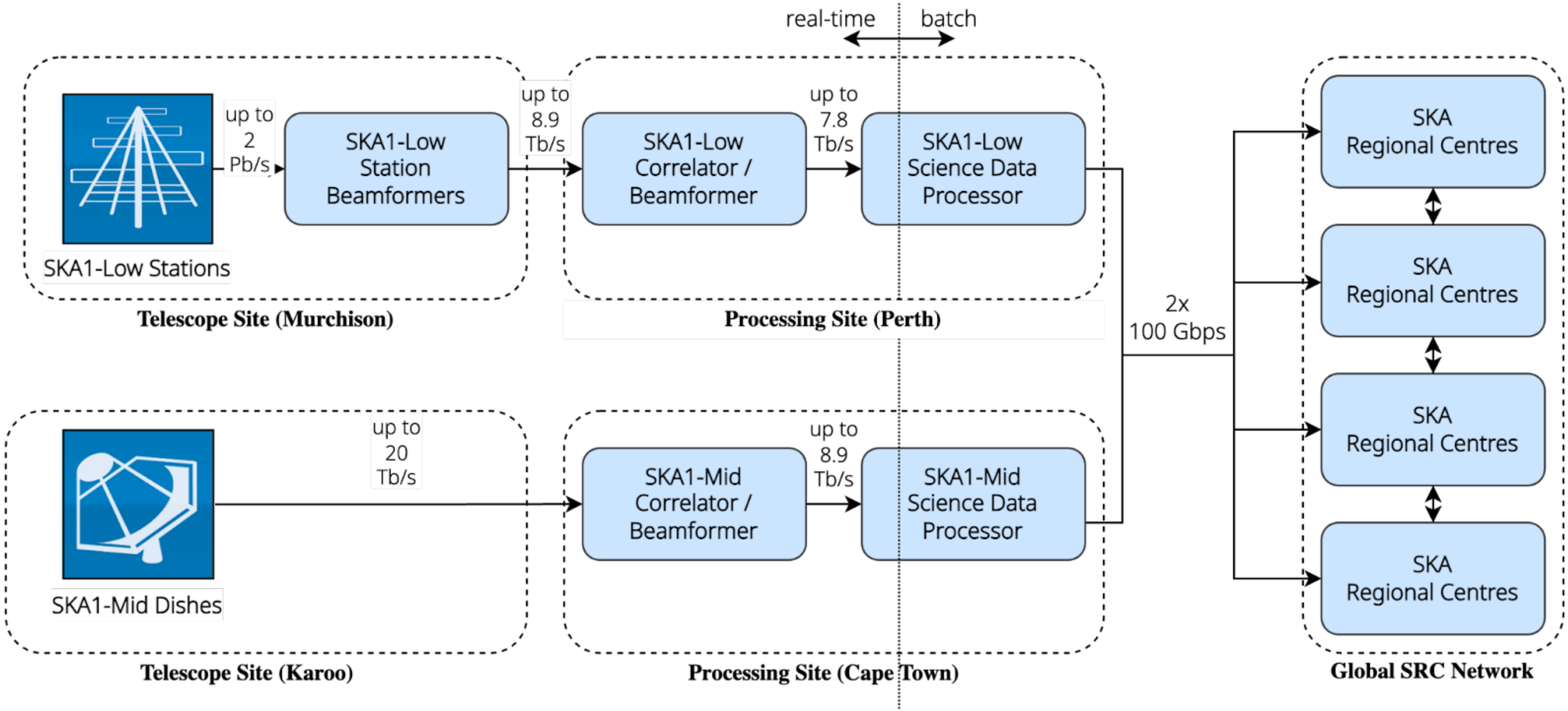
52,000 core hours (13,000 hours for imaging)
~6 days to image ~8 hour observation

Ye+ 2023, arXiv:2309.16560



250,000 core hours for 0.3" field

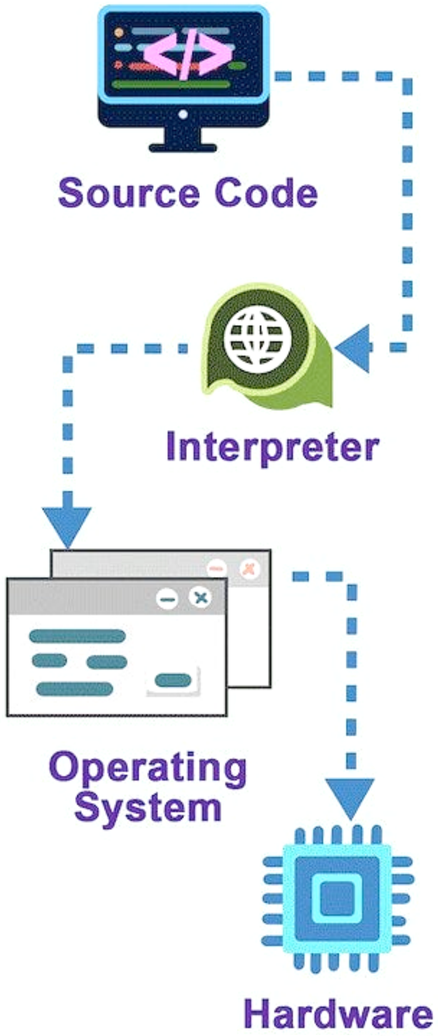
HPC Challenge of the Radio astronomy

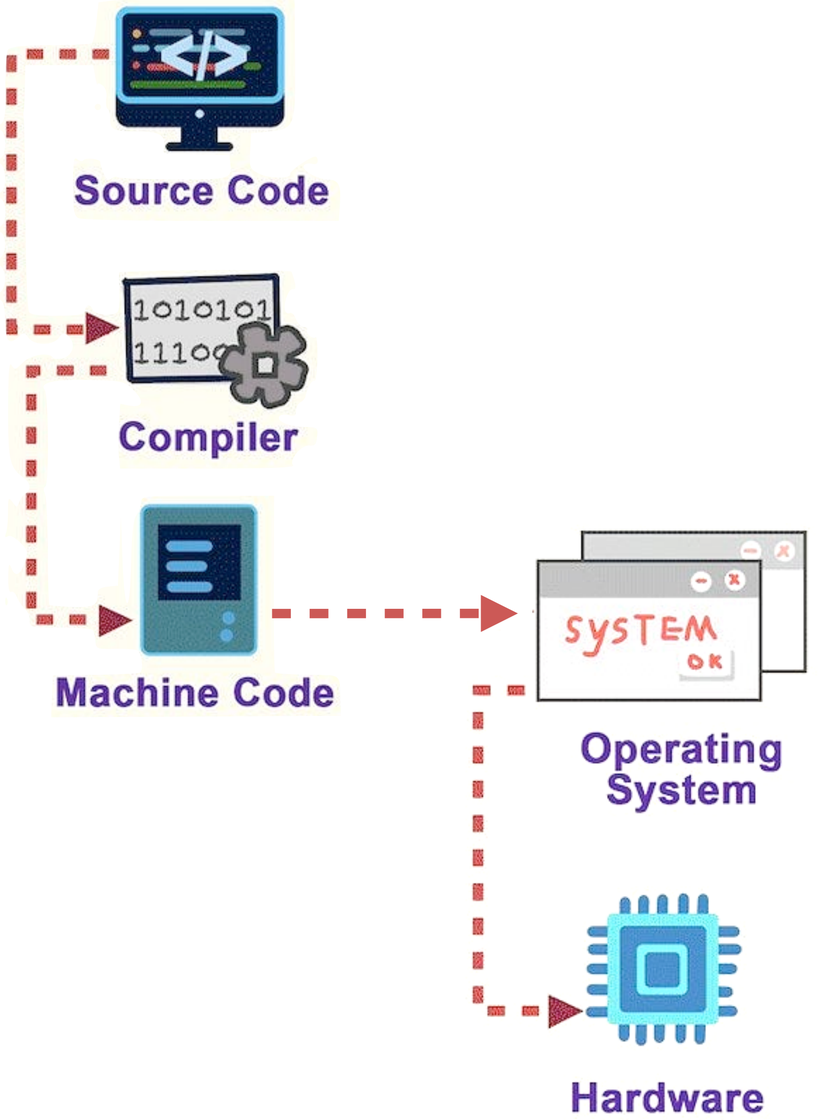


Topics covered today

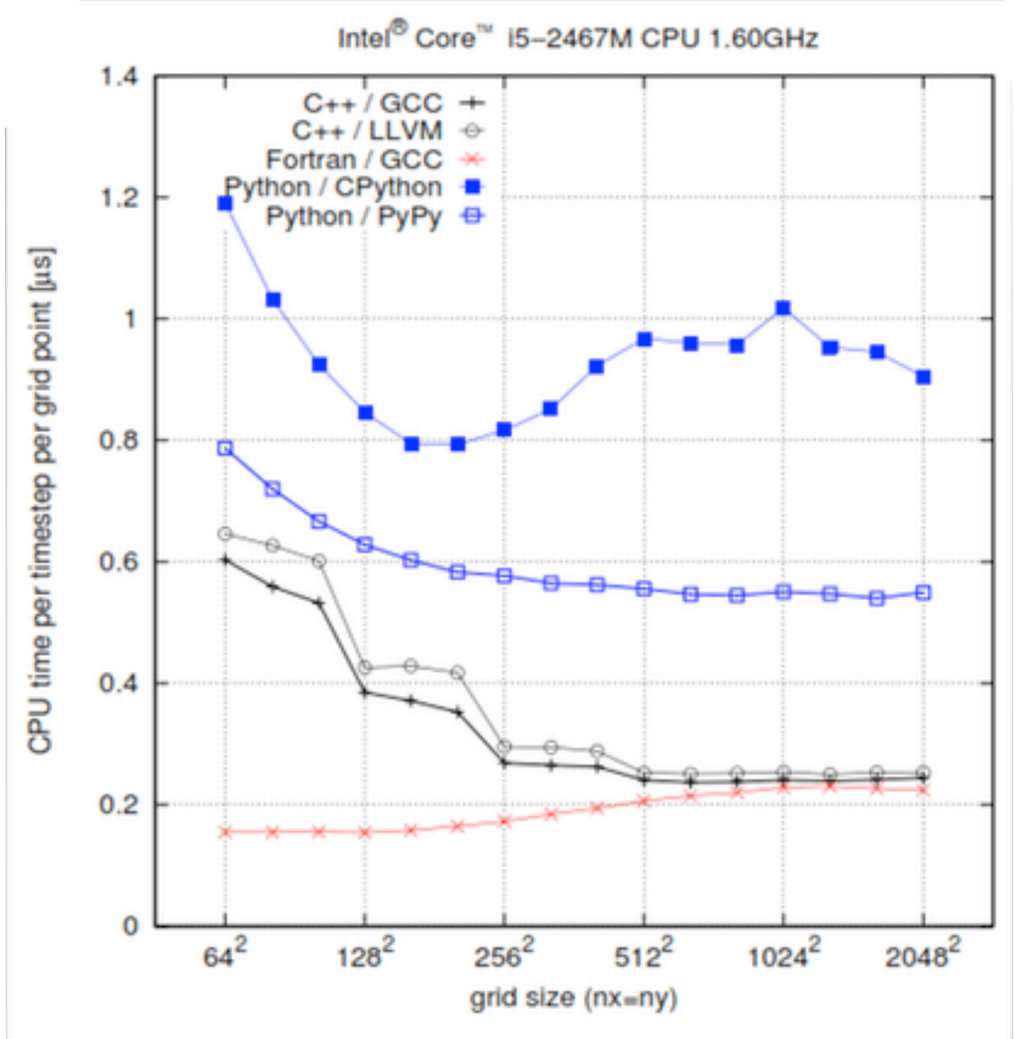
- **Acceleration**:
  - **Numpy** & vectorization
  - **Cupy** & GPU computing
  - **Numba** & just-in-time compilation
- **Parallelization**:
  - **MPI**: bottom-up parallelization
  - **Dask**: top-down parallelization
- **Github repository**: github.com/Road2SKA/python_hpc_tutorial

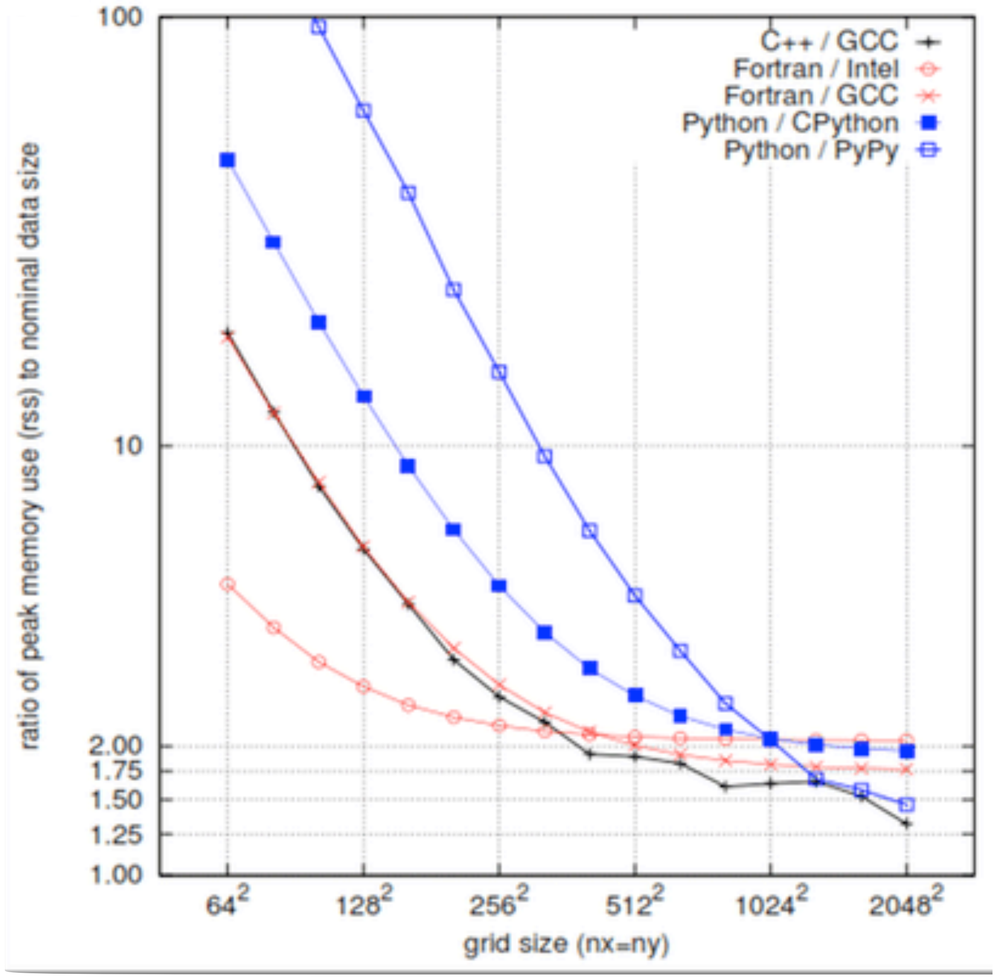
Python vs C++/Fortran





Python vs C++/Fortran

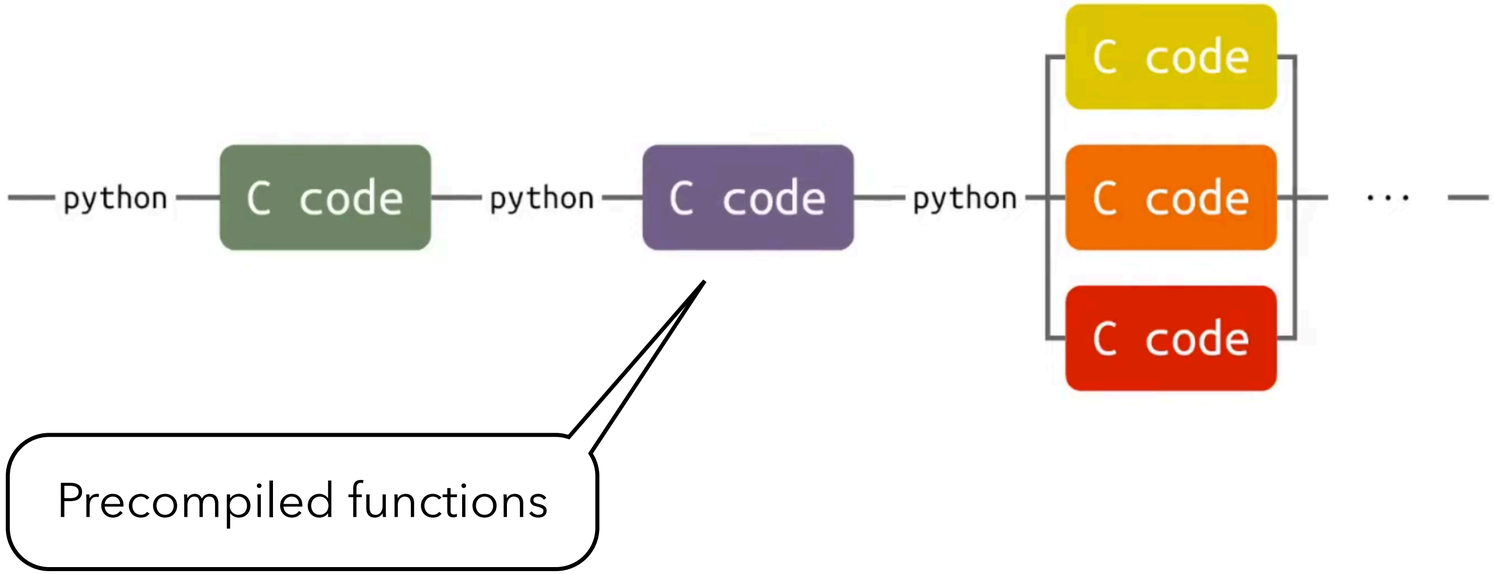




V. Guzman, Horacio. (2014). Modeling peak interaction forces of soft matter with dynamic AFM in liquid (http://hdl.handle.net/10486/663401).

Why python?

- Very flexible, most operations can be **overloaded**
- Easy to connect it with other languages like C and Fortran
- Wide range of libraries available: numpy, scipy, astropy, etc

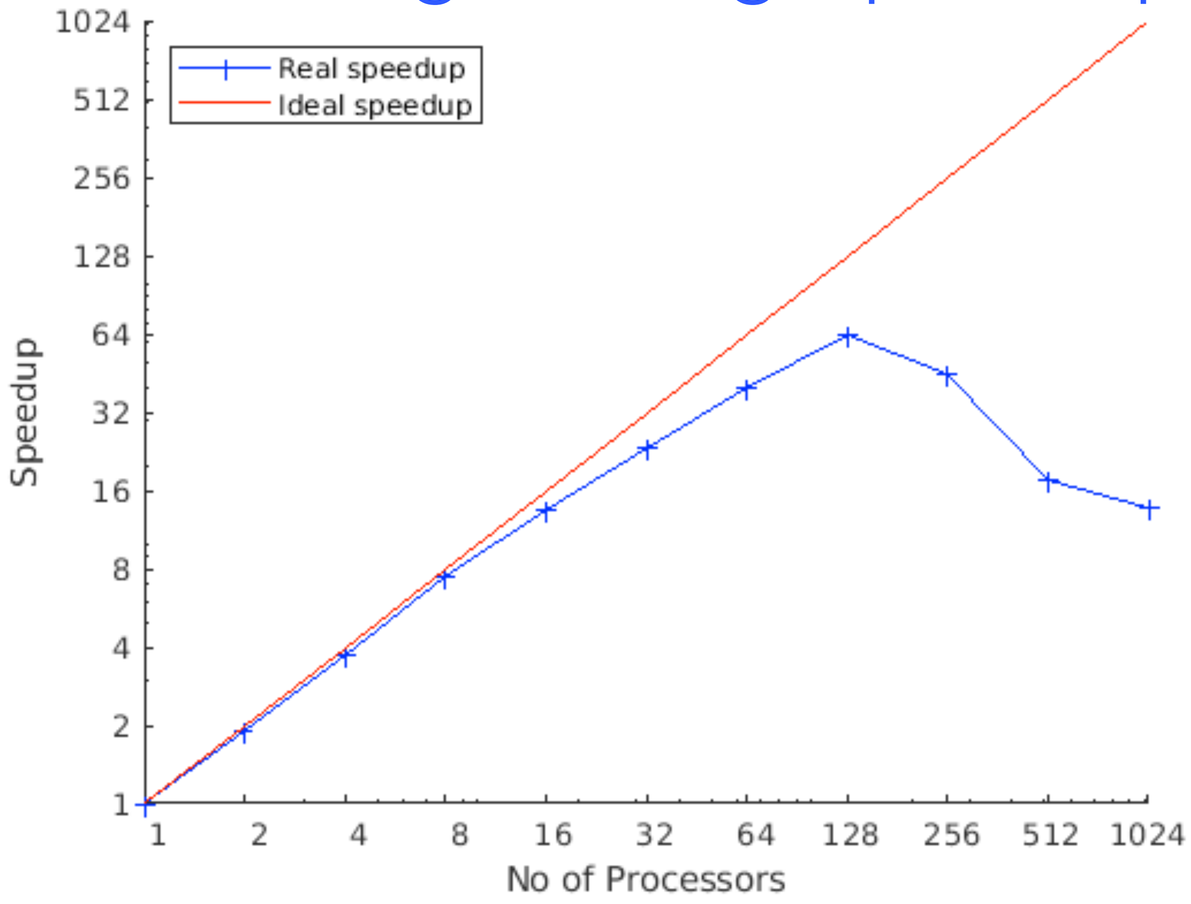


How to benchmark in python

- **Profiling**: analyzing code to understand the execution time breakdown & memory consumption
- A **bottleneck** is an especially slow part of the code. Speeding up a bottleneck speeds up the entire execution time
- **Benchmarking**: measuring the execution time or memory consumption with respect to a reference implementation

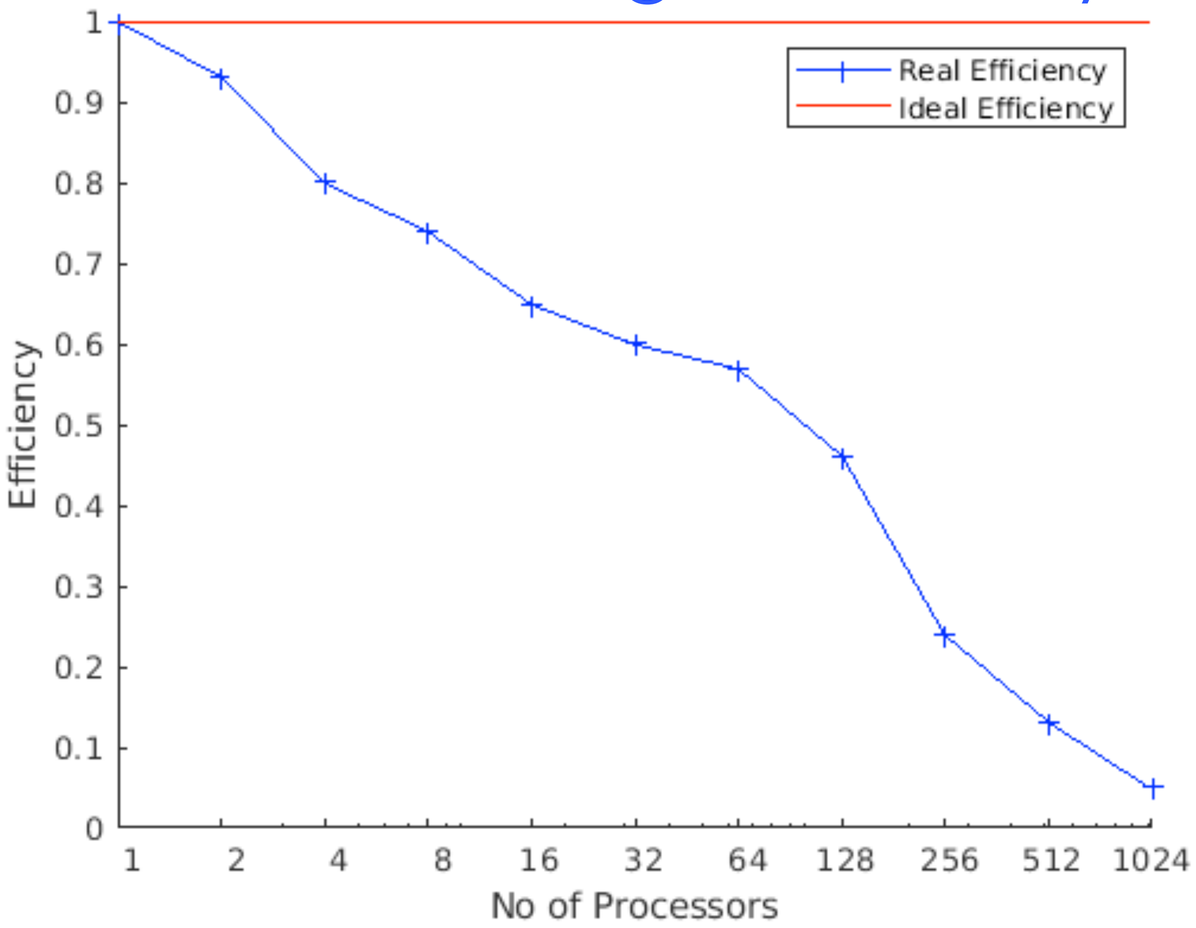
0_python_profile.ipynb

Strong Scaling: speedup



Keep the problem size the same, increase the amount of resources

Weak Scaling: Efficiency

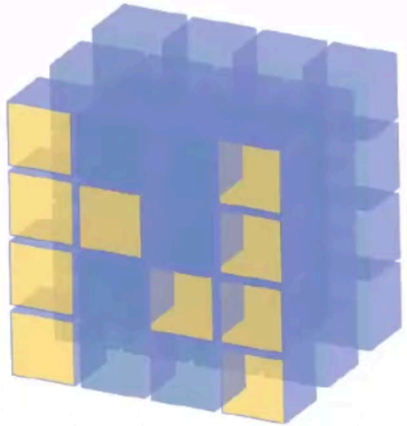


Increase the problem size the same, and proportionally increase the resources

Want to stay at high efficiency to maximize our use of resources


ACCELERATION

NumPy

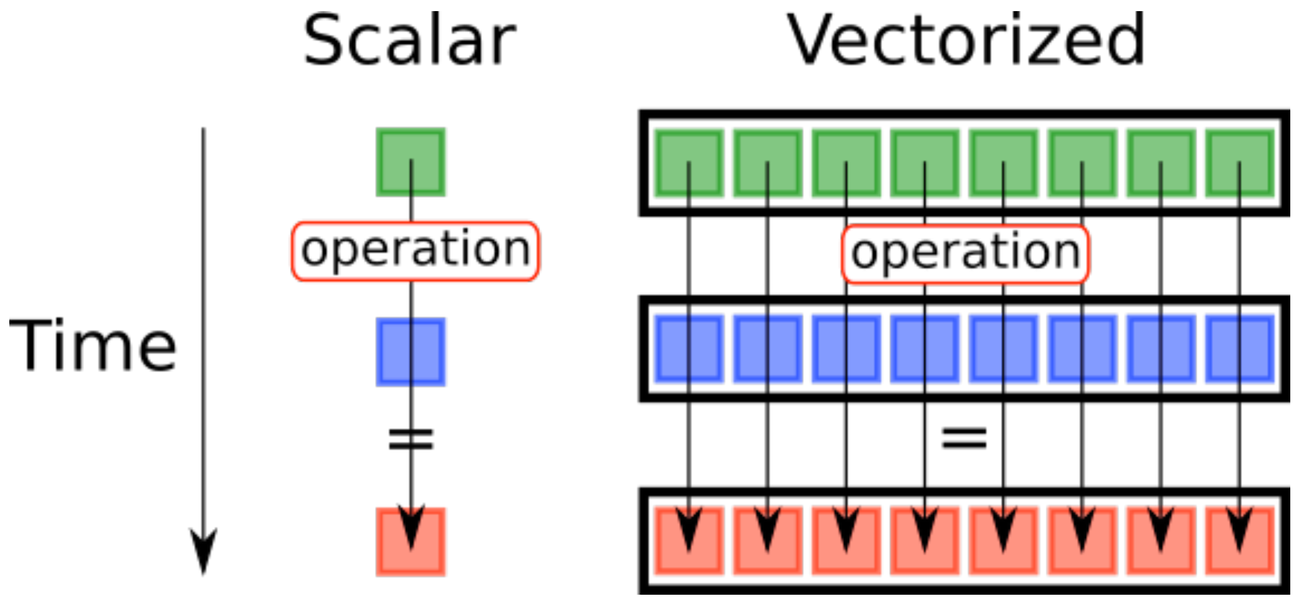


NumPy is a Python library that adds support for large multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- Behind the scenes, Numpy calls high-performance libraries like BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage)

NumPy & vectorization

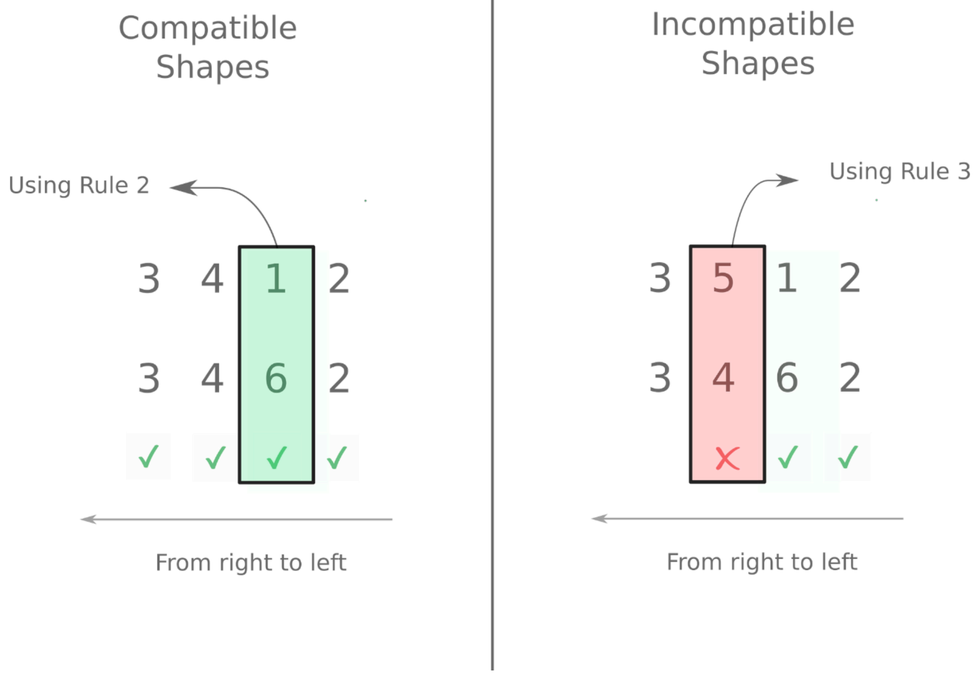
- **Vectorization**: Rewriting a loop so that instead of processing a single element of an array **N** times, it processes **K** elements of the array simultaneously **N**/**K** times.
- Often done automatically by the compiler
- Typical values of **K** for a CPU is 4-8



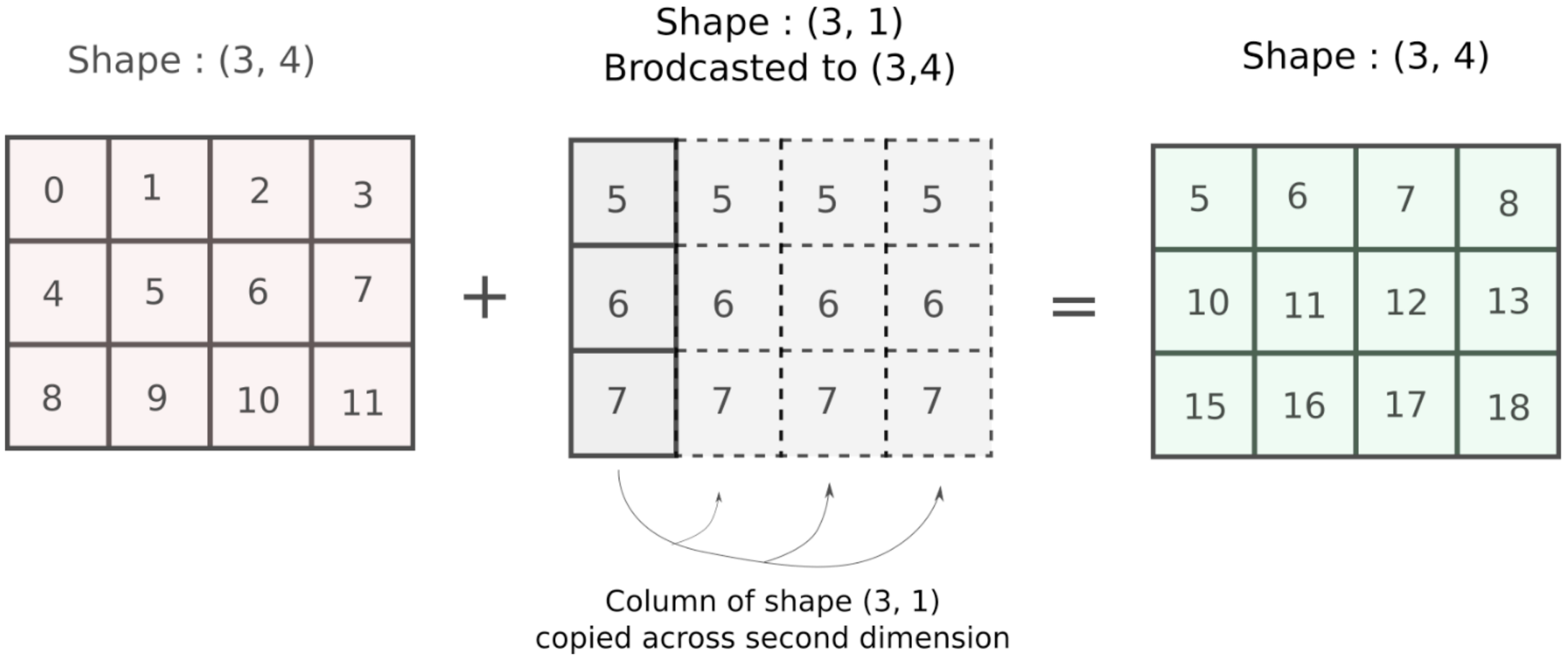
NumPy & broadcasting

Rules of broadcasting:

1. NumPy compares the shape of the two arrays dimension-by-dimension from right to left
2. Two dimensions are said to be compatible if **both of them are equal, or either one of them is 1.**
3. If both the dimensions are unequal and neither of them is 1, then NumPy will throw an error and halt.



NumPy & broadcasting

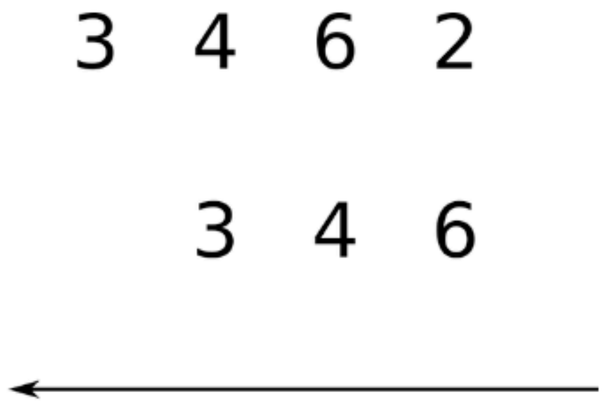


NumPy & broadcasting

Quiz: which of these array multiplications will work?

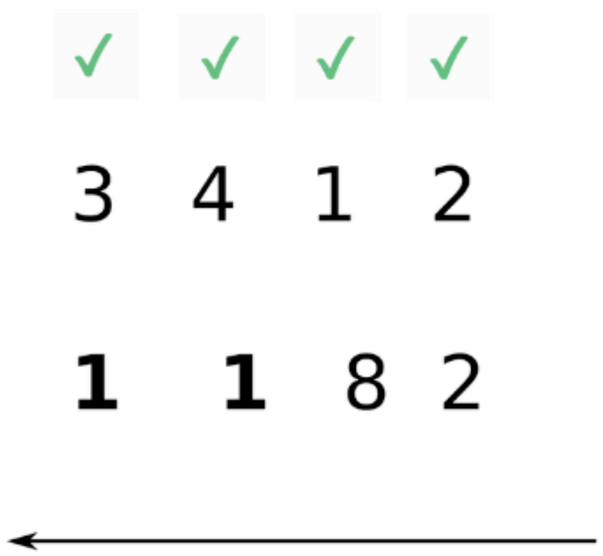
1

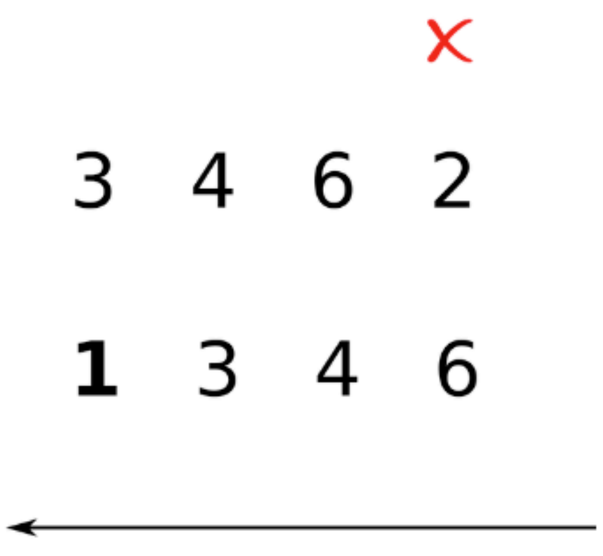
2



3

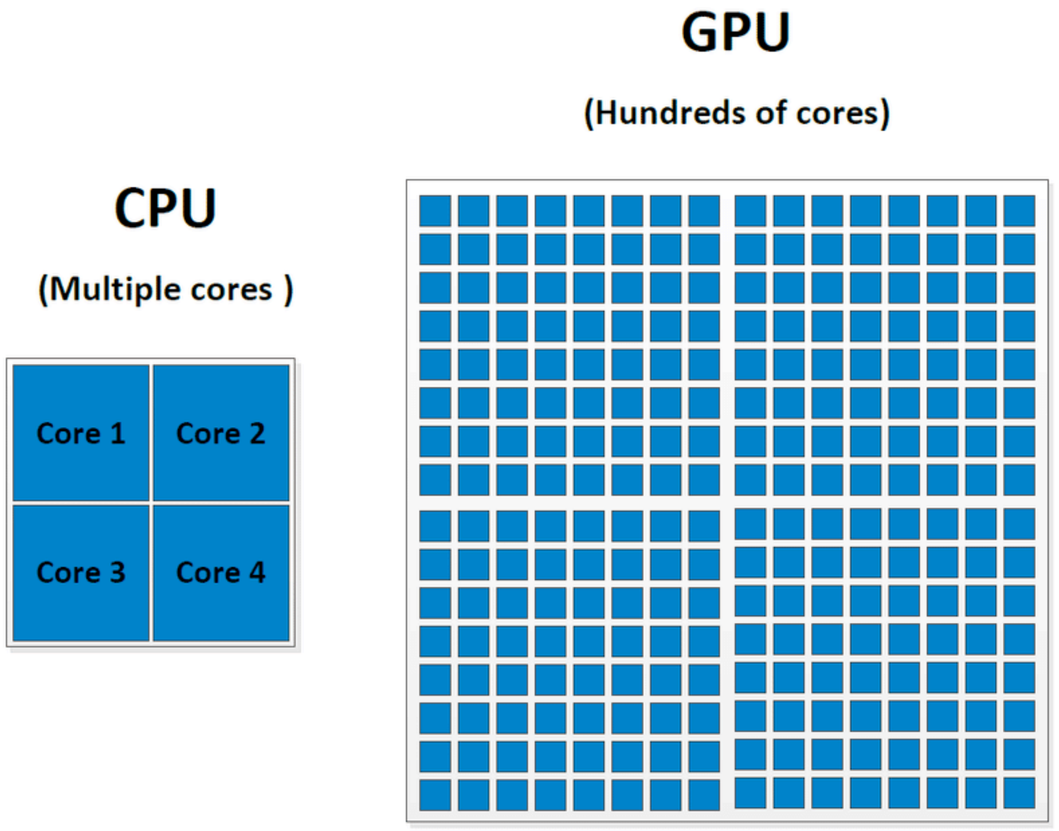
NumPy & broadcasting





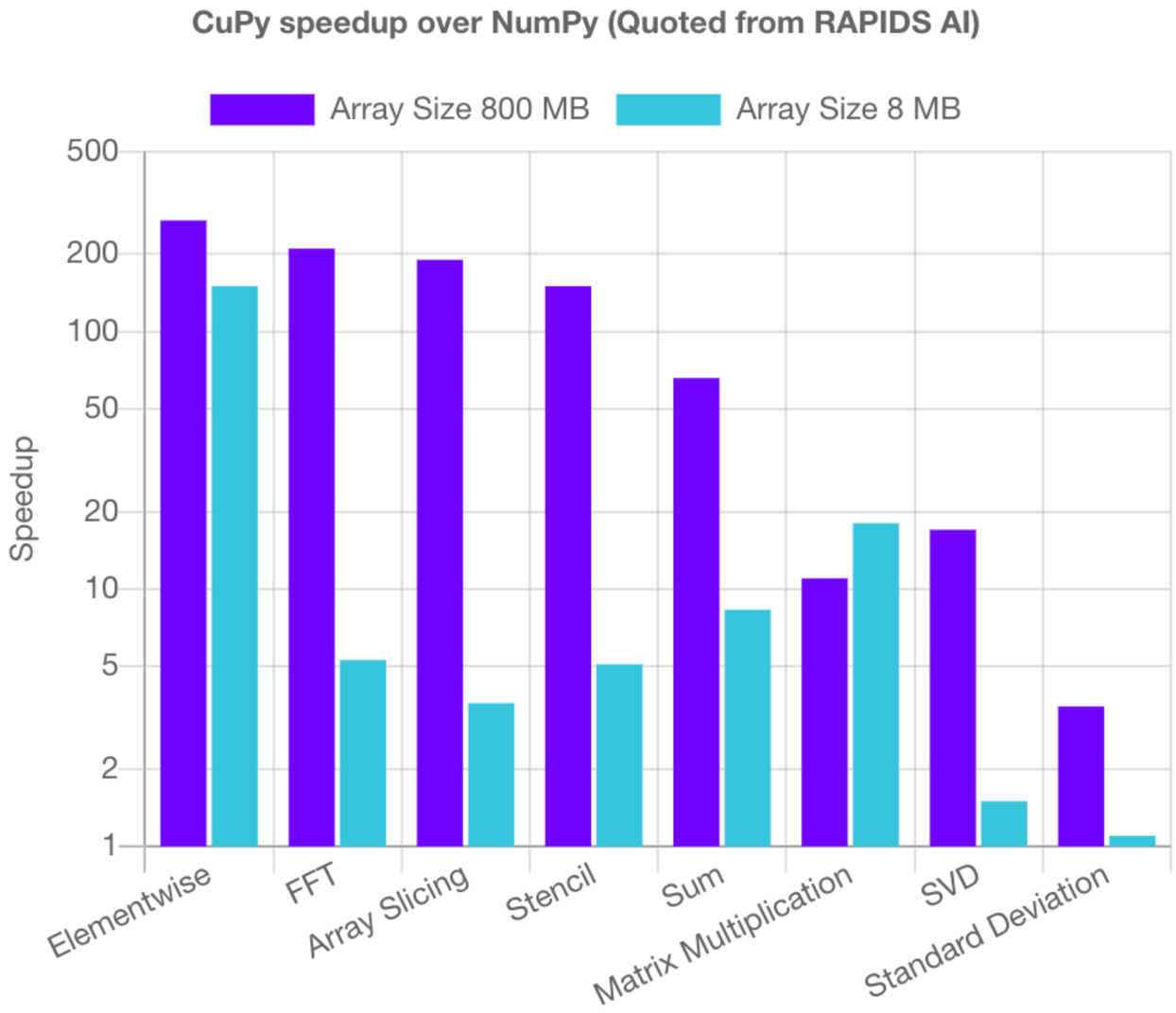
GPU computing

- **CPU**: C++, Fortran, etc
- **GPU** needs instructions compiled from hardware-specific language
  - NVIDIA GPU: **CUDA**
  - AMD GPU: **ROCm**



GPU computing

CuPy



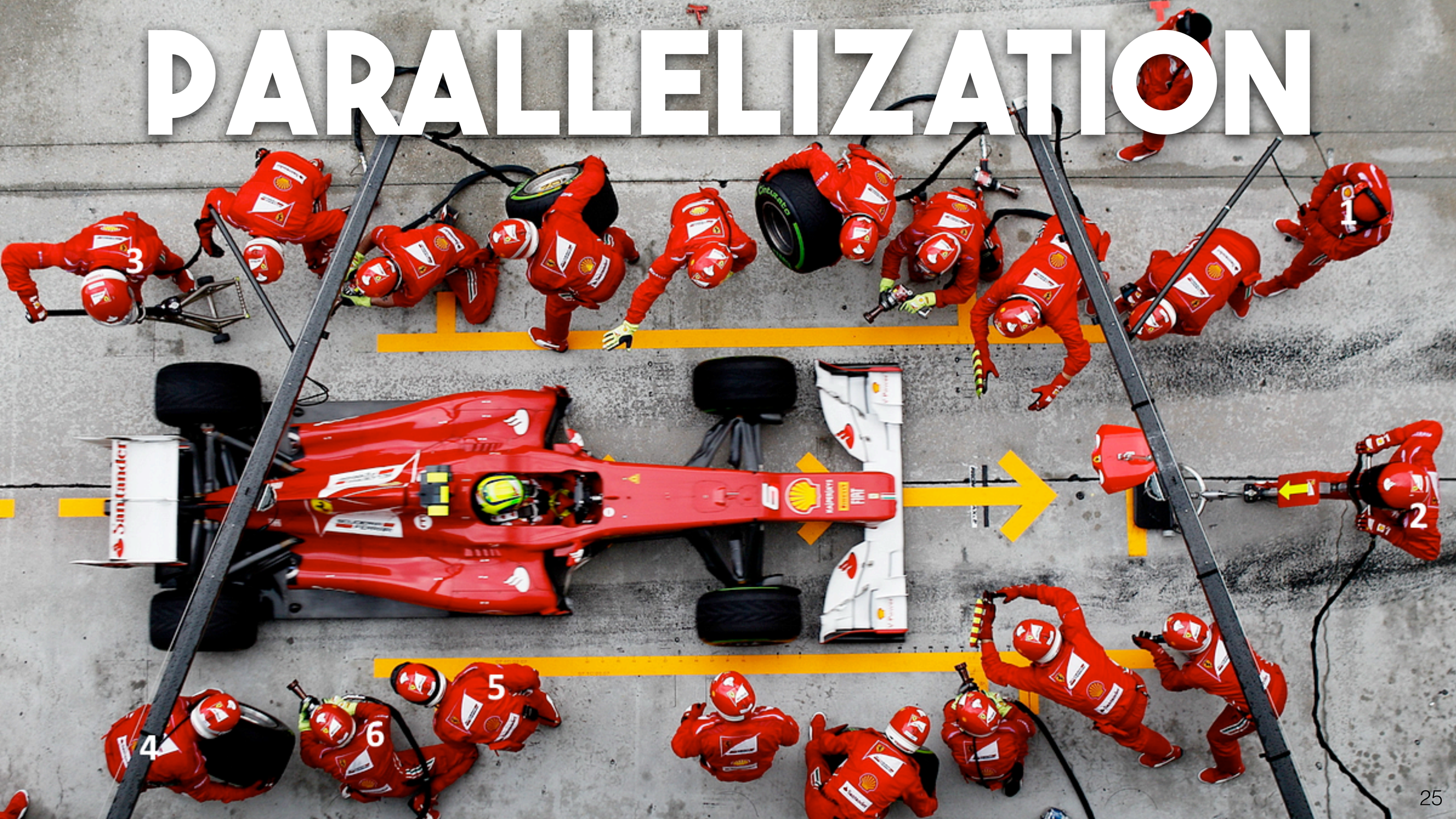
Just-in-time compilation (JIT)

- What if I don’t already have a pre-compiled code?
  => **Use just-in-time (JIT) compilation**

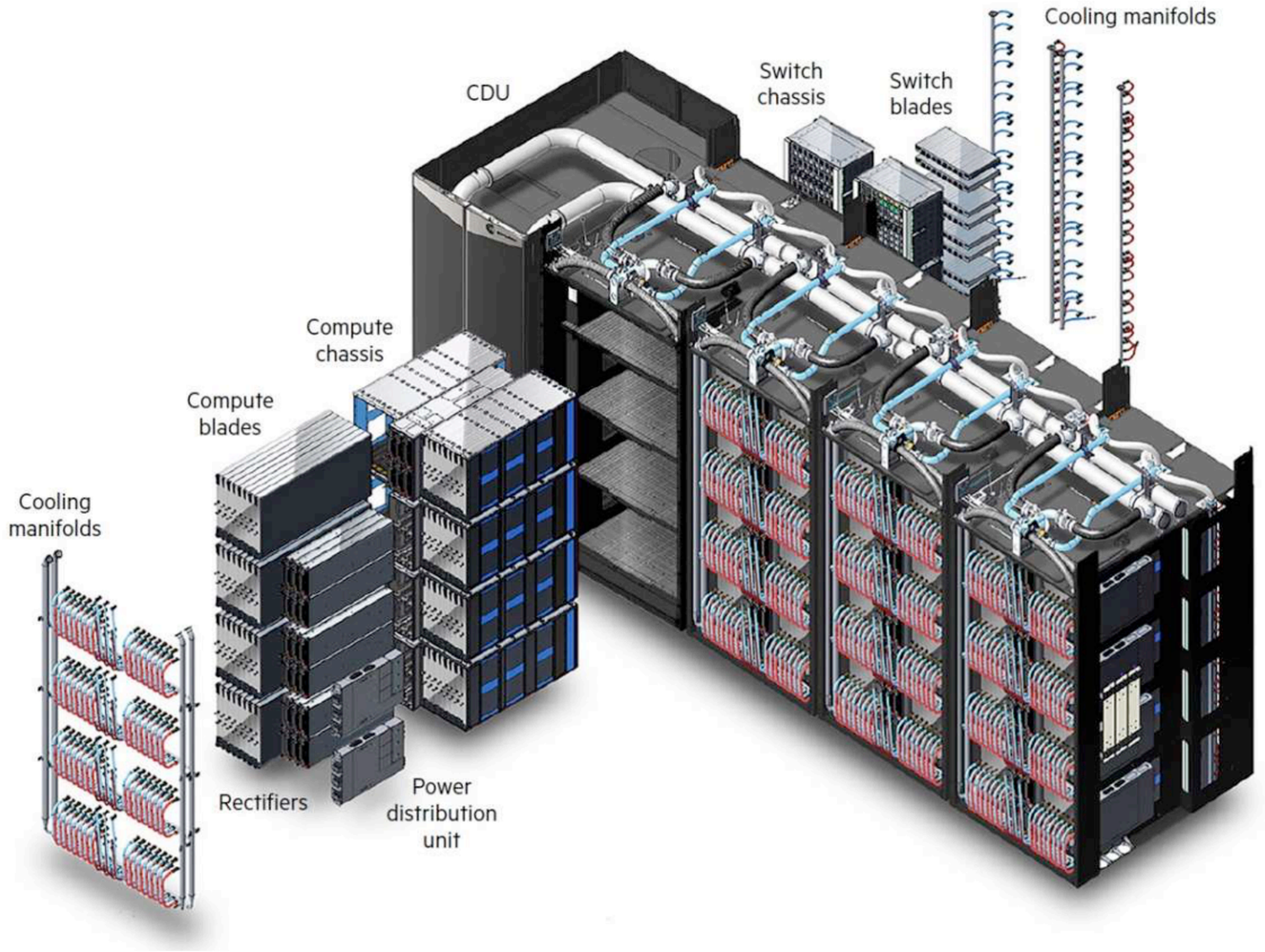
- **Numba** is a python library which translates Python functions to optimized machine code at runtime

- There’s also **numba-CUDA** to use JIT with GPUs: https://nvidia.github.io/numba-cuda/

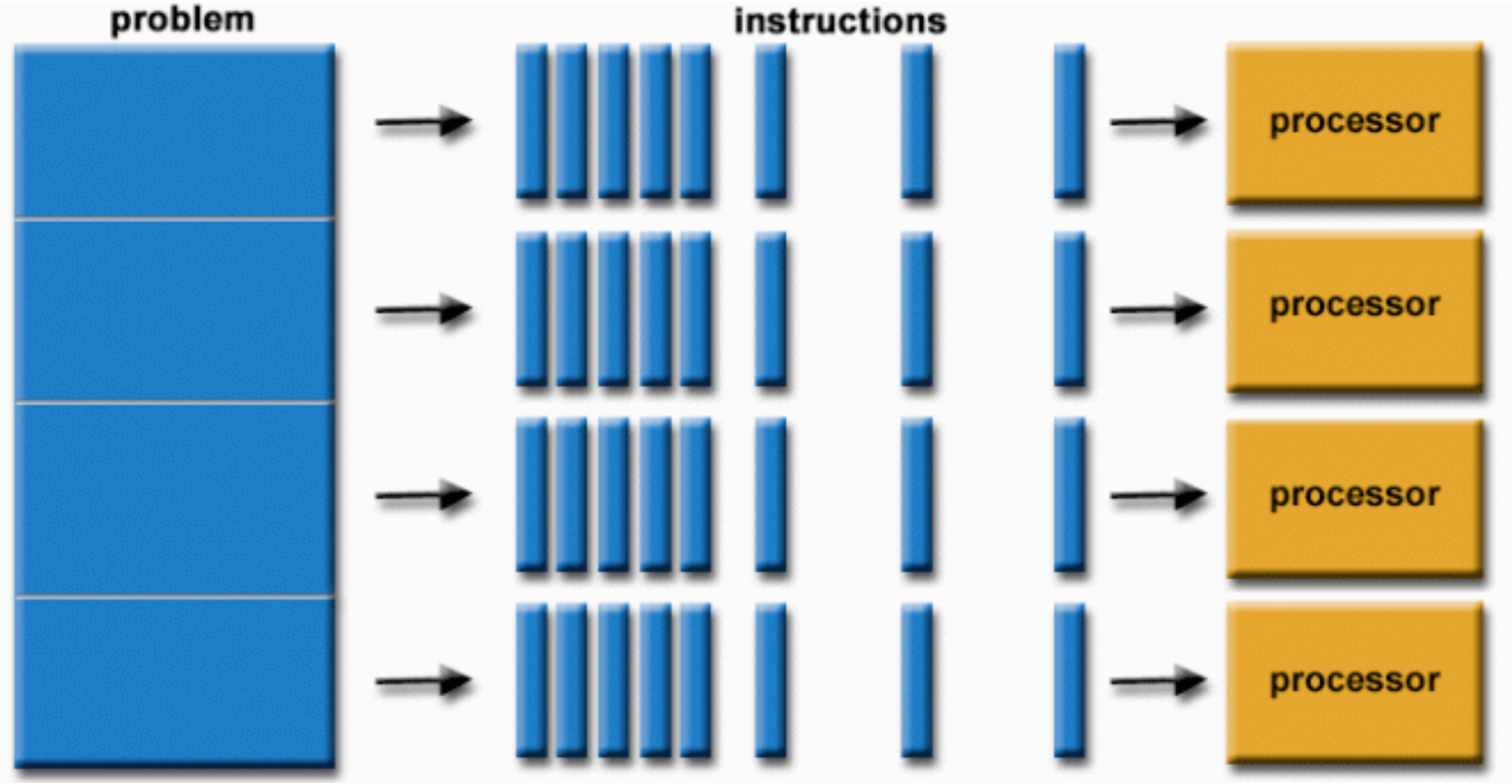
PARALLELIZATION



PARALLELIZATION



PARALLELIZATION

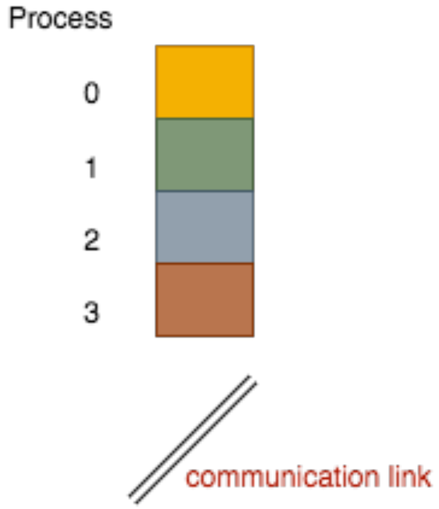


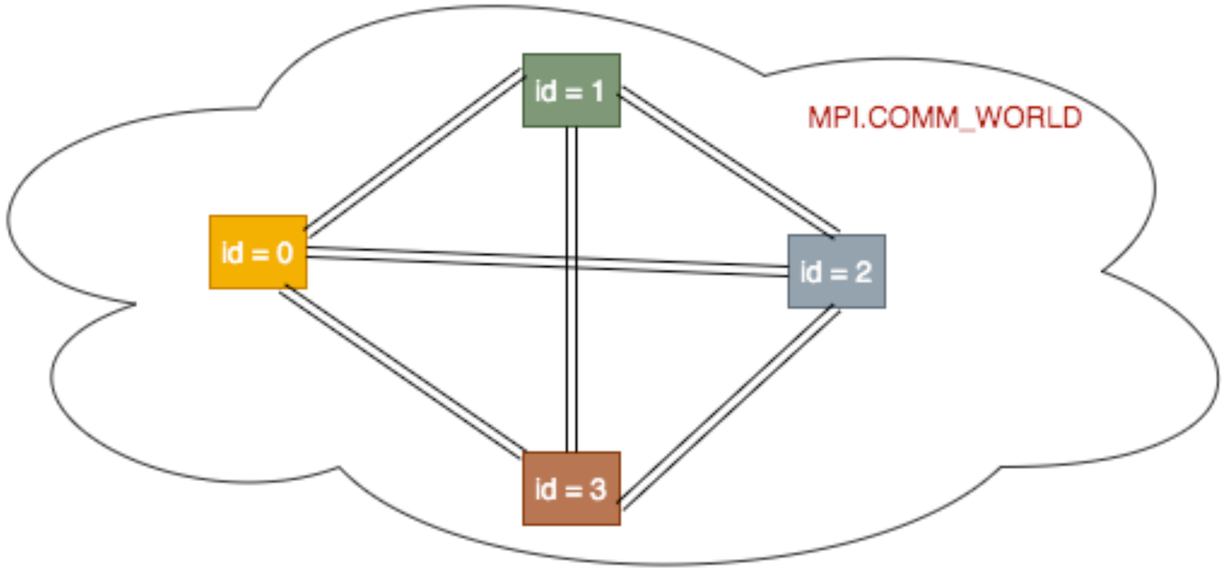
Message passing interface (MPI)

Message Passing Interface is a standardized and portable message-passing system designed to function on a wide variety of parallel computers.

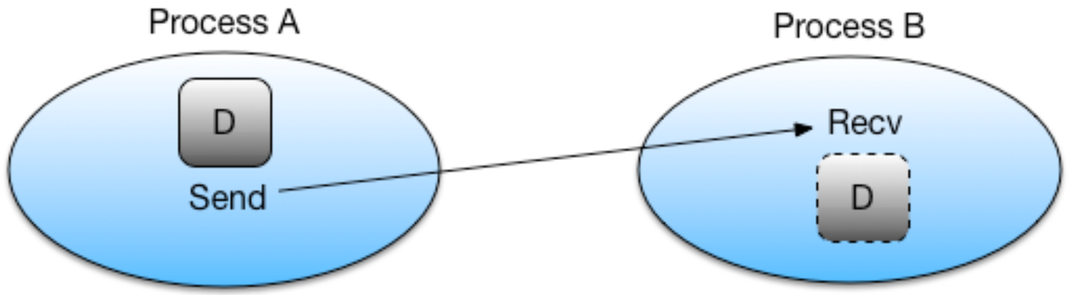
mpi4py/mpi4py

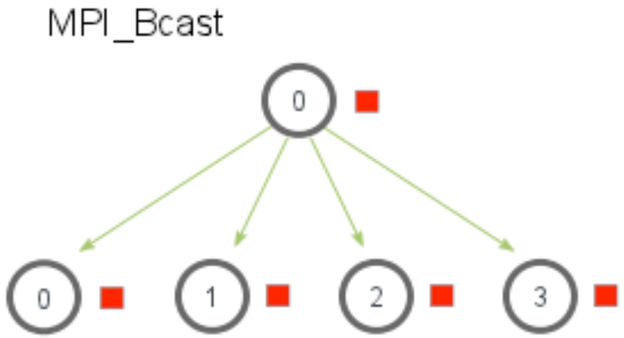
Python bindings for MPI

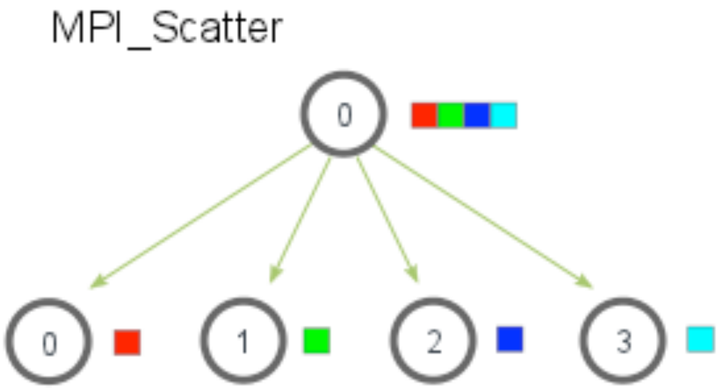


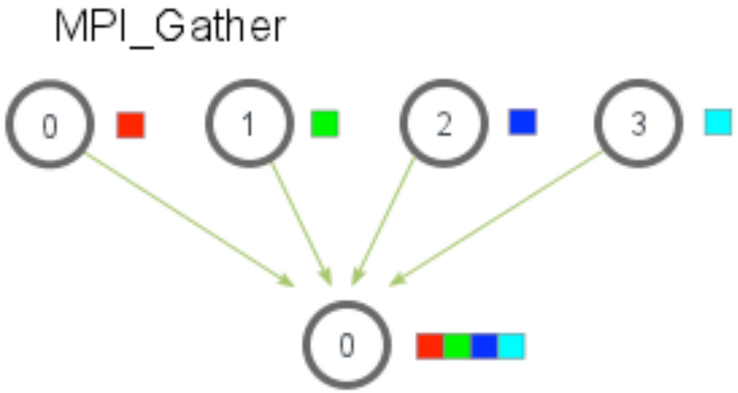


Message passing interface (MPI)









Dask

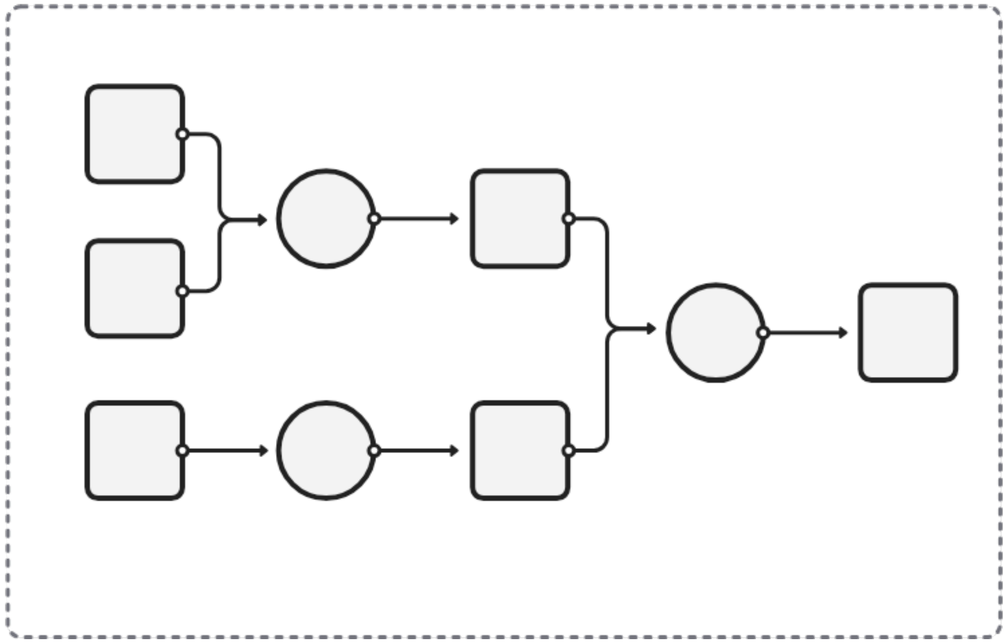
dask

Collections

(create task graphs)

- Dask Array
- Dask DataFrame
- Dask Bag
- Dask Delayed
- Futures

Task Graph

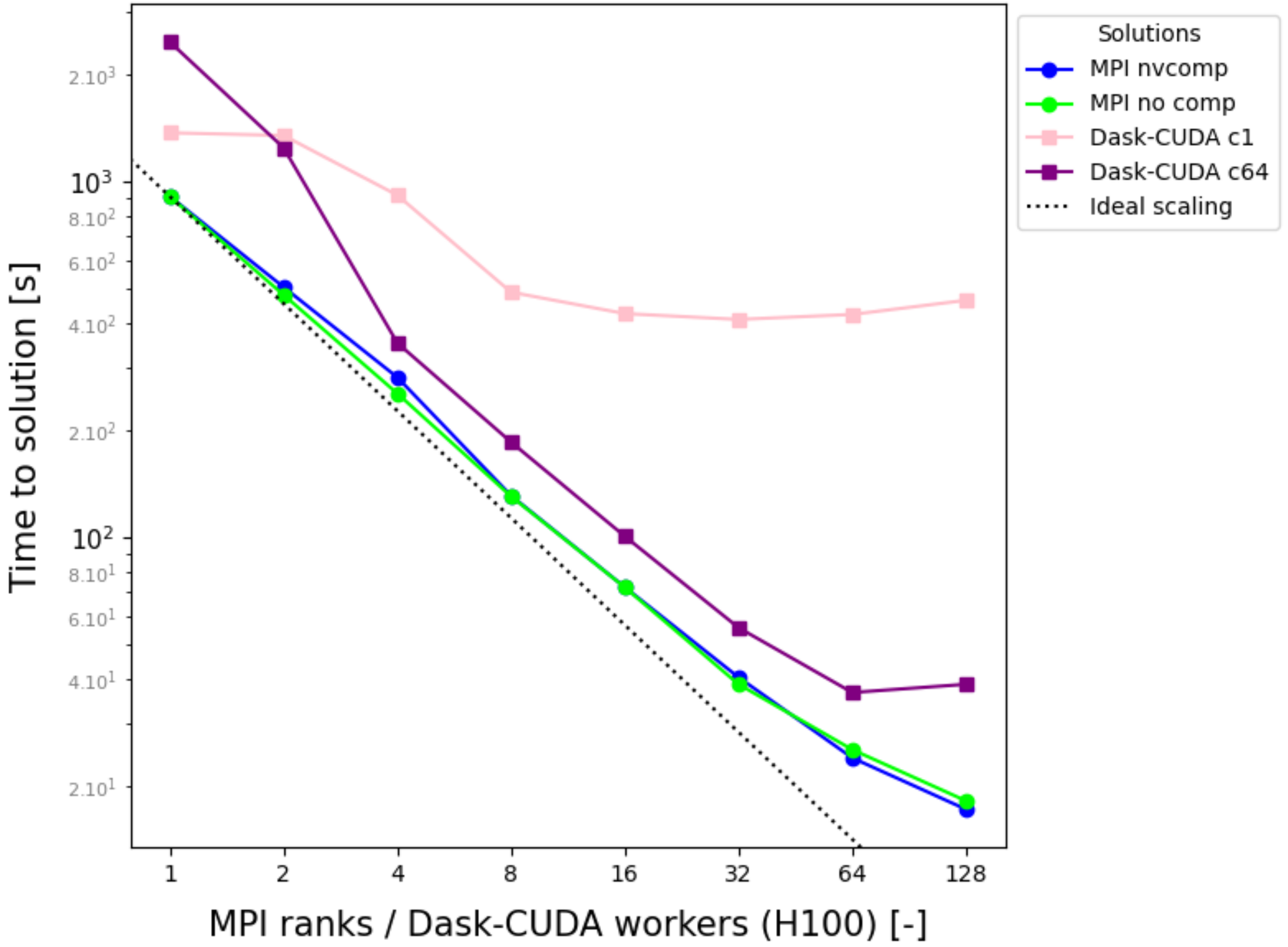


Schedulers

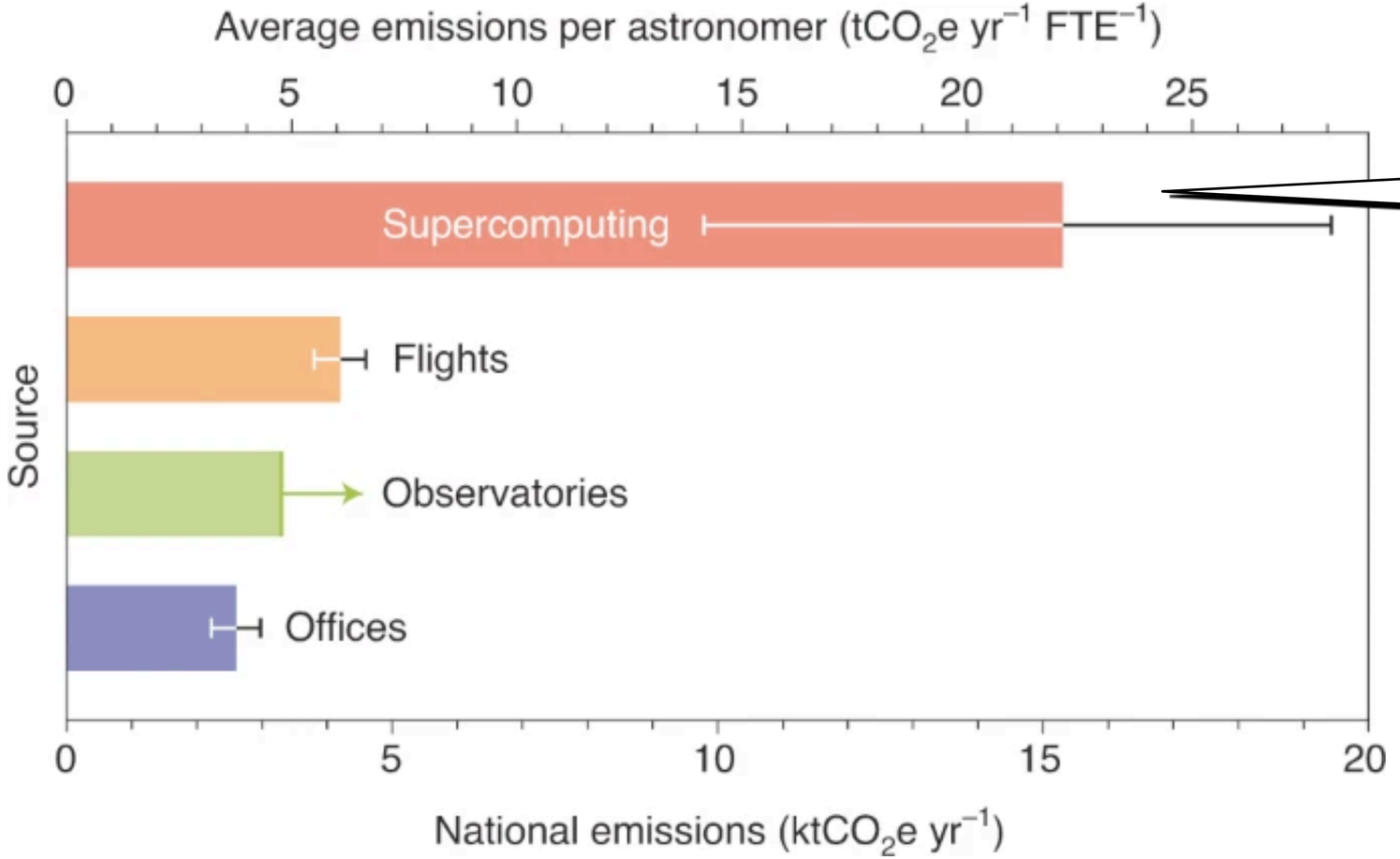
(execute task graphs)

- Single-machine (threads, processes, synchronous)
- Distributed

Dask vs MPI


Solutions
MPI nvcomp
MPI no comp
Dask-CUDA c1
Dask-CUDA c64
Ideal scaling
Time to solution [s]
2.10^3
10^3
8.10^2
6.10^2
4.10^2
2.10^2
10^2
8.10^1
6.10^1
4.10^1
2.10^1
1
2
4
8
16
32
64
128
MPI ranks / Dask-CUDA workers (H100) [-]

THE ENVIRONMENTAL IMPACT



~40% Computational/theoretical astrophysics (simulations)
~30% radio astronomy data reduction
~30% optical & infrared

A. Stevens et al., "The imperative to reduce carbon emissions in astronomy" Nature Astronomy 2020

For reference, per-capita carbon emissions*….
in Australia: 17 tCO_2/yr
in South Africa: 6.5 tCO_2/yr

*https://www.worldometers.info/co2-emissions/co2-emissions-per-capita/