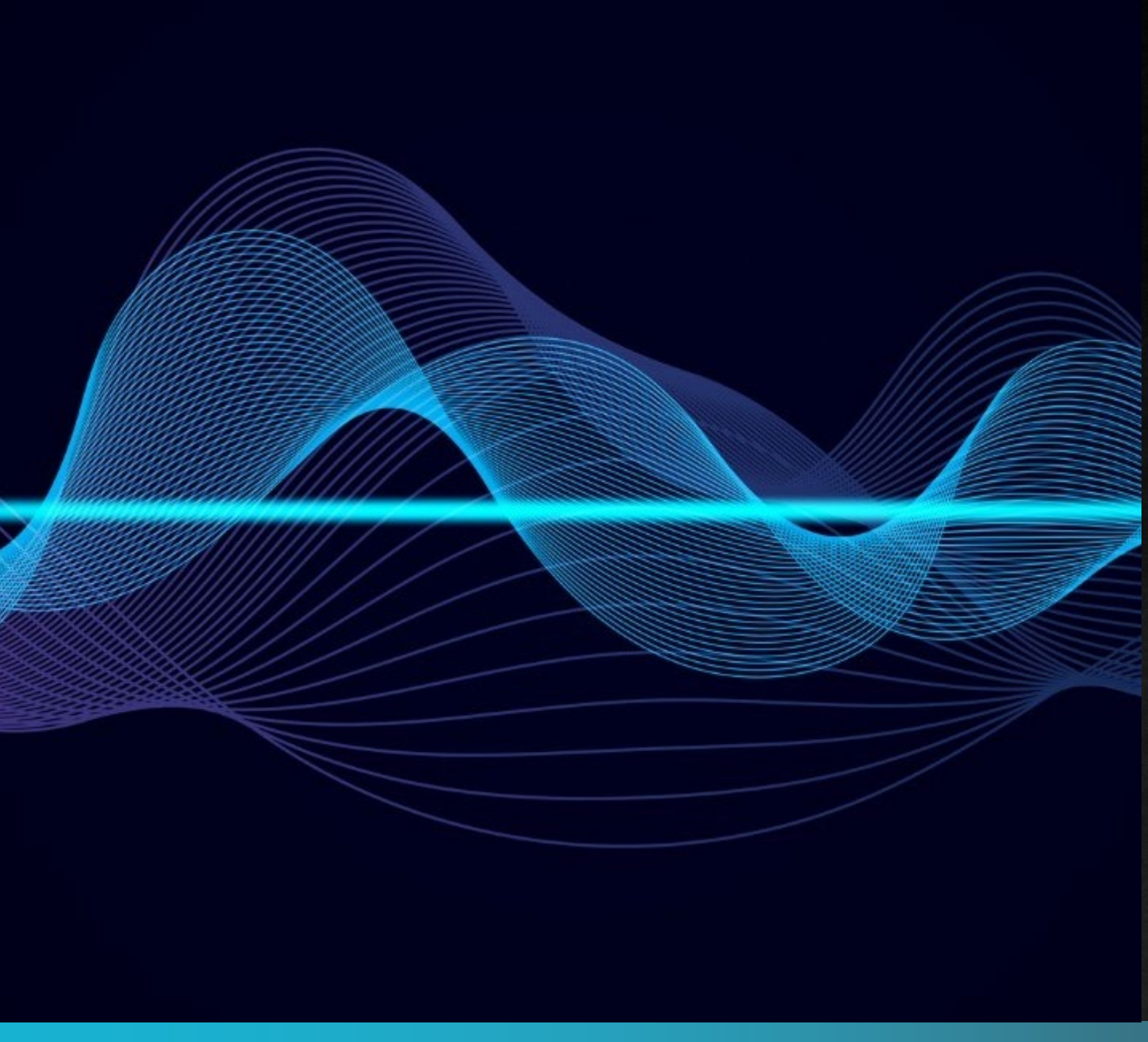


Computational Physics

PHYS 6260

Parallel Programming Intro

Announcements:

- HW6: Will post by Wednesday, due next Friday 2/28
- Project progress reports: Due Friday 3/7

Lecture Outline

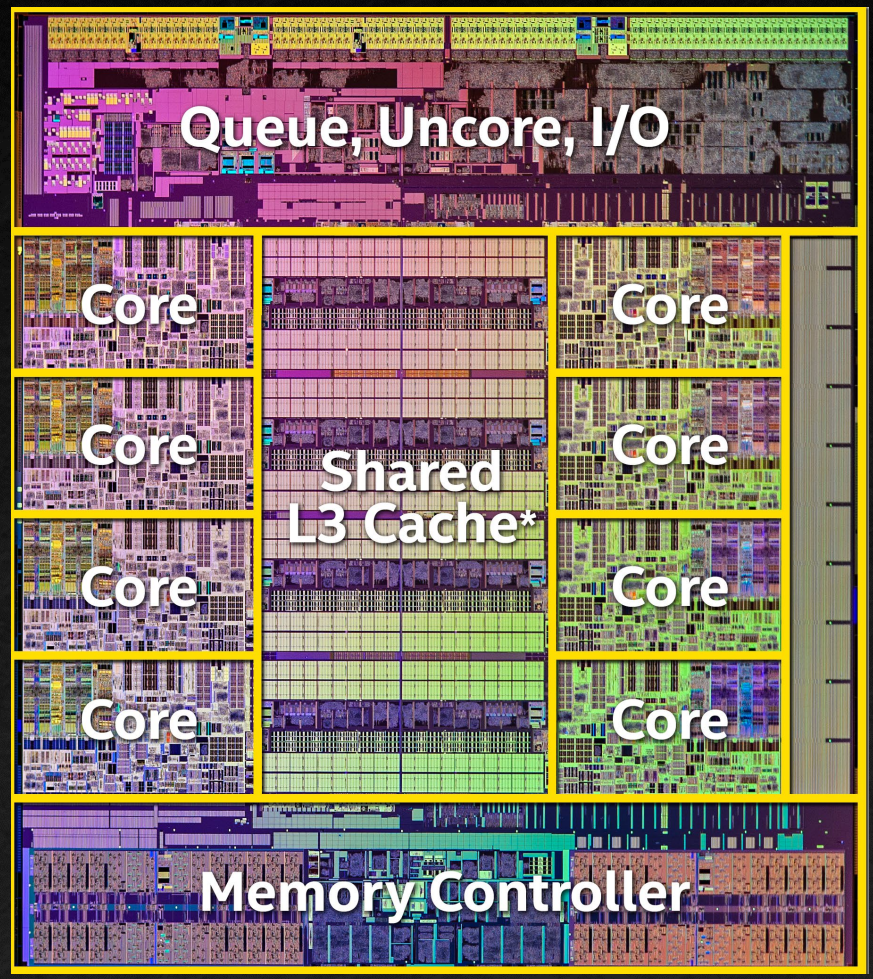
We will cover these topics

- Hardware architecture
- Parallel programming models
- Parallel efficiency

Multi-core architecture

- All CPUs have multiple cores, ranging from 2 to 128
- Graphics cards (GPUs) can be used for scientific computing, which have up to 16k lightweight cores
- There are also “data processing units” (DPUs), which is an nVIDIA term. They are specialized for machine learning and matrix operations.
  - ARM based (cell phone CPUs, Apple M1/2 CPUs)

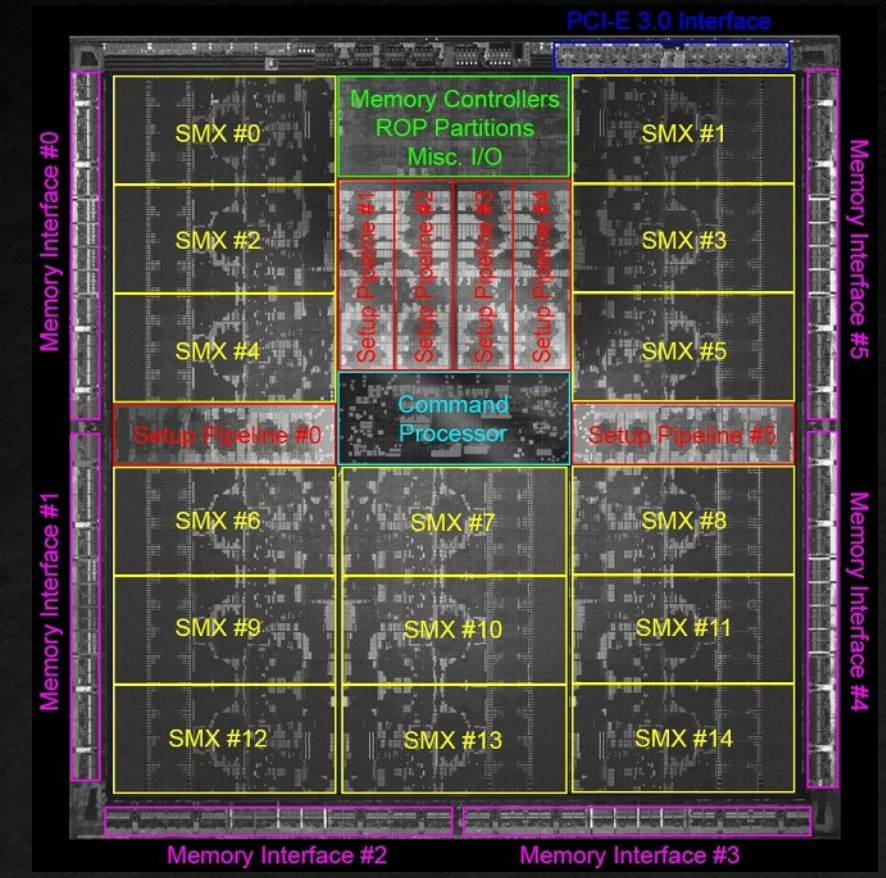
8-core CPU



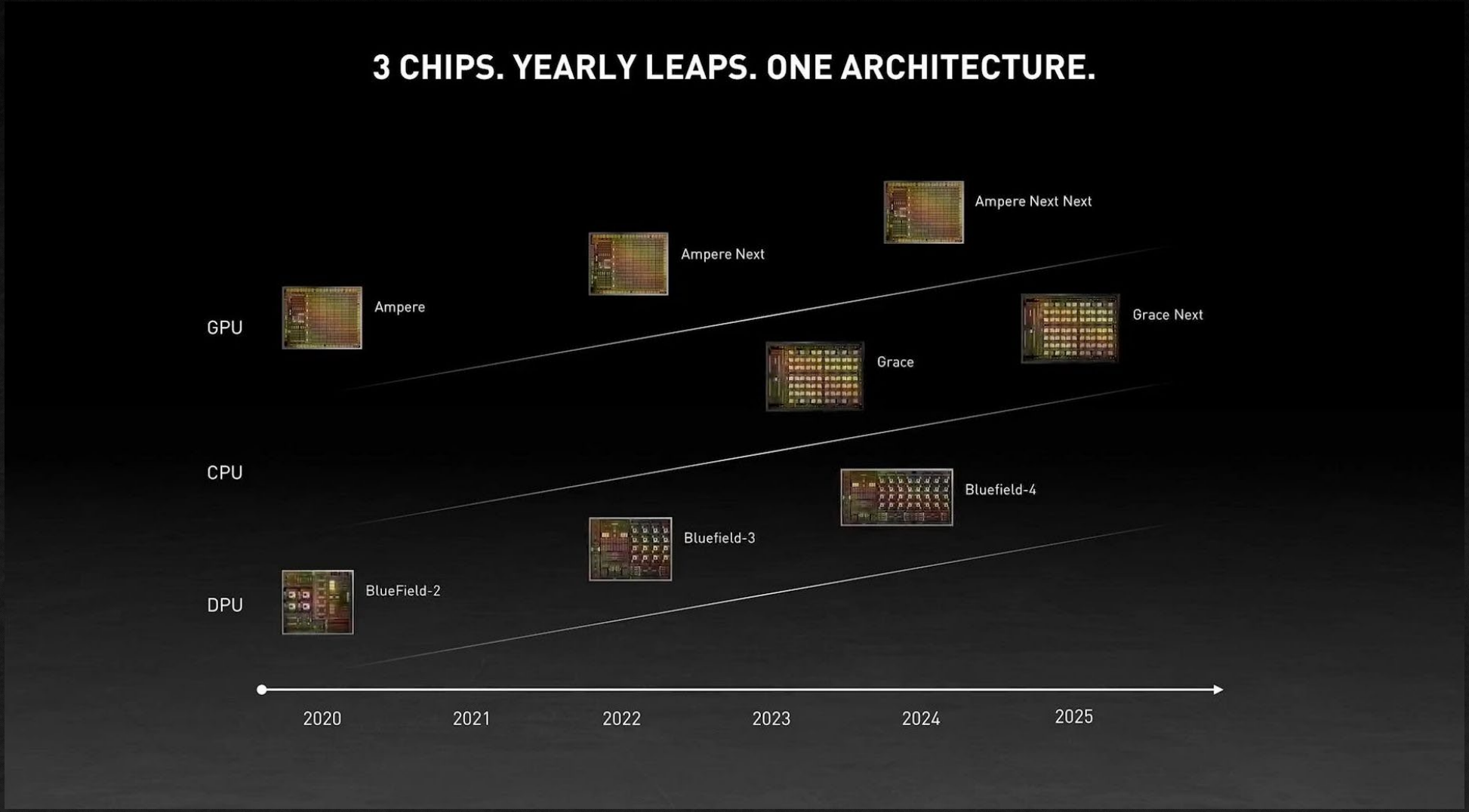
Multi-core architecture

- All CPUs have multiple cores, ranging from 2 to 128
  - Workhorse of scientific computing
- Graphics cards (GPUs) can be used for scientific computing, which have up to 16k lightweight cores
  - Very slow individually and have little cache memory
- There are also “data processing units” (DPUs), which is an nVIDIA term. They are specialized for machine learning and matrix operations.
  - ARM based (cell phone CPUs, Apple M1/2 CPUs)

2880-core GPU



nVIDIA Roadmap

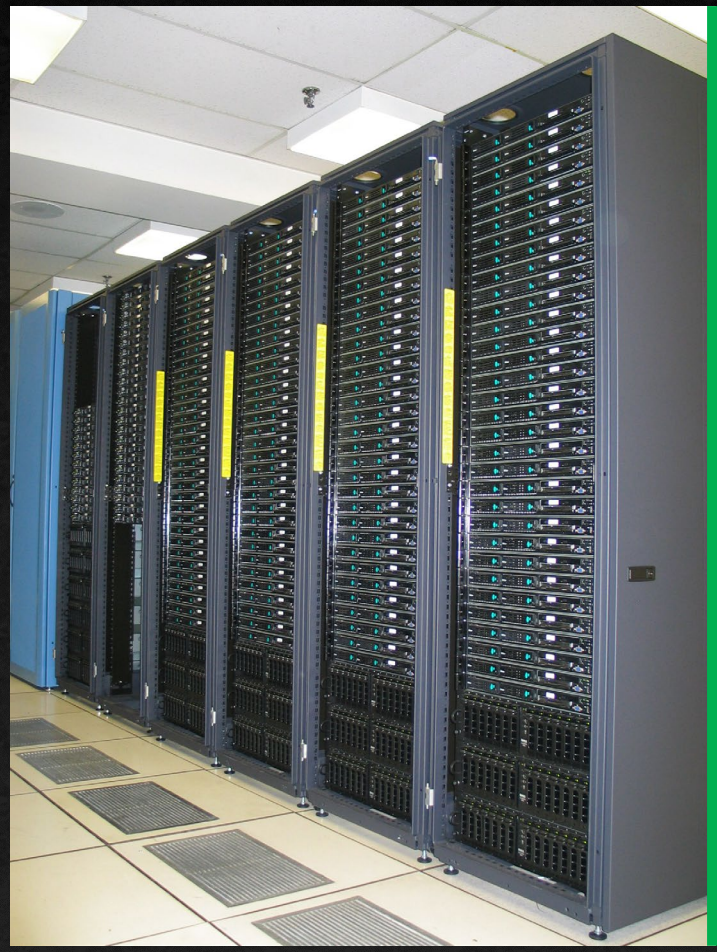


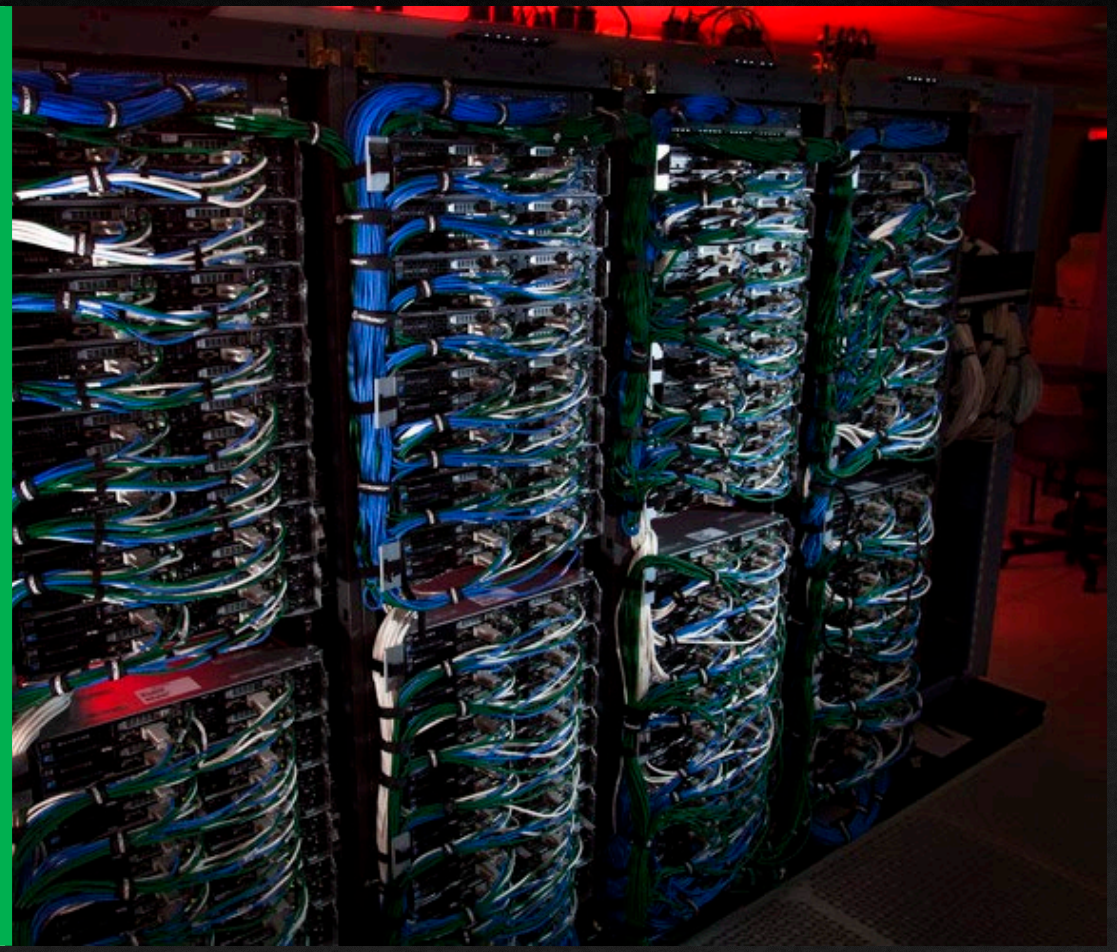
Why not GPUs, all the time?

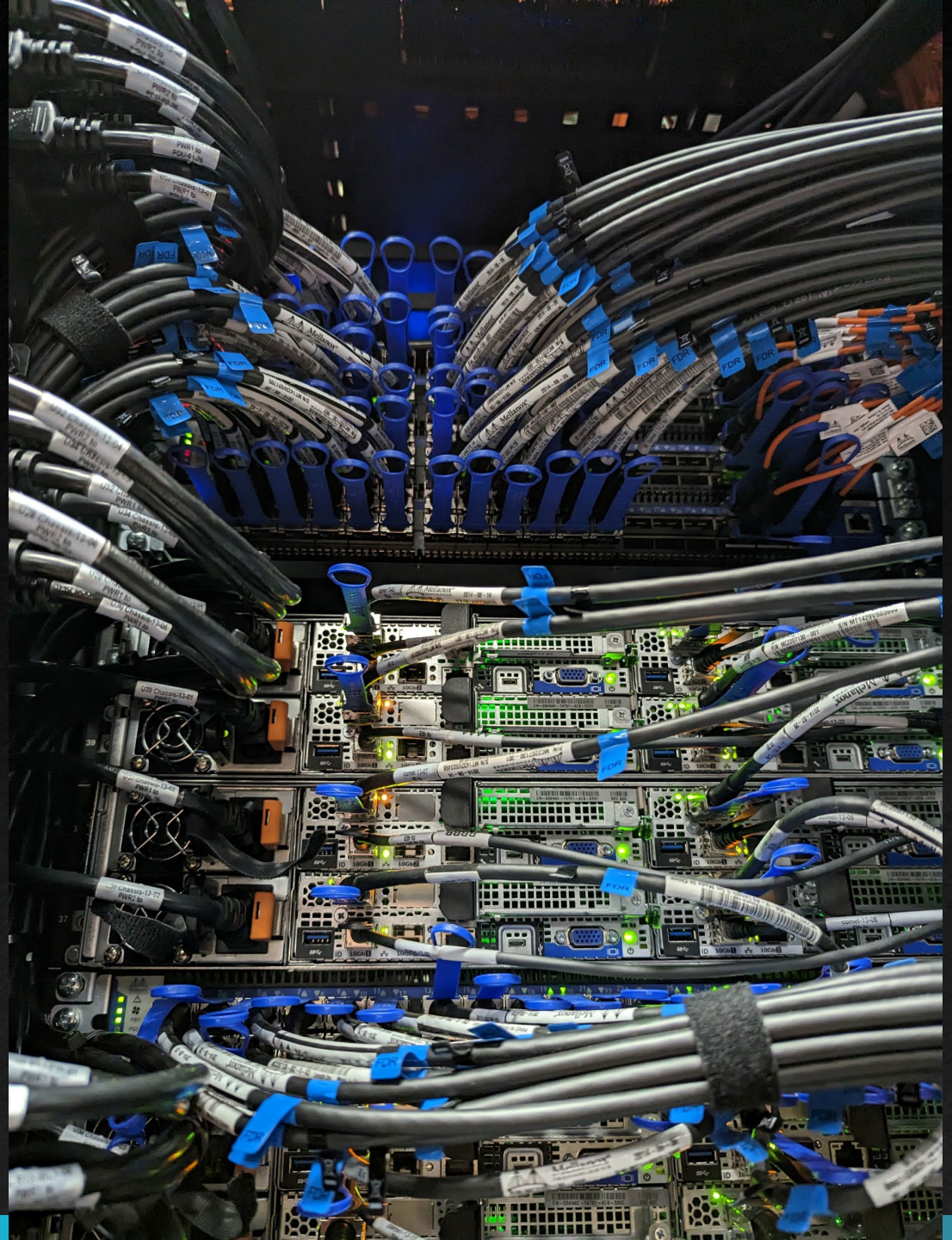
- Limited amount of memory for each core
- Difficult to program (for now)
  - Need to micro-manage memory movement, parallelism, using programming language CUDA (C-based)
- When properly done, it can speed up calculations up to 50x
  - In most cases, it's about a 10x speedup
- Same speed-up expectations for other many-core CPU/DPU chips, but it's easier to program. Doesn't require a rewrite/refactor of the code

Compute clusters

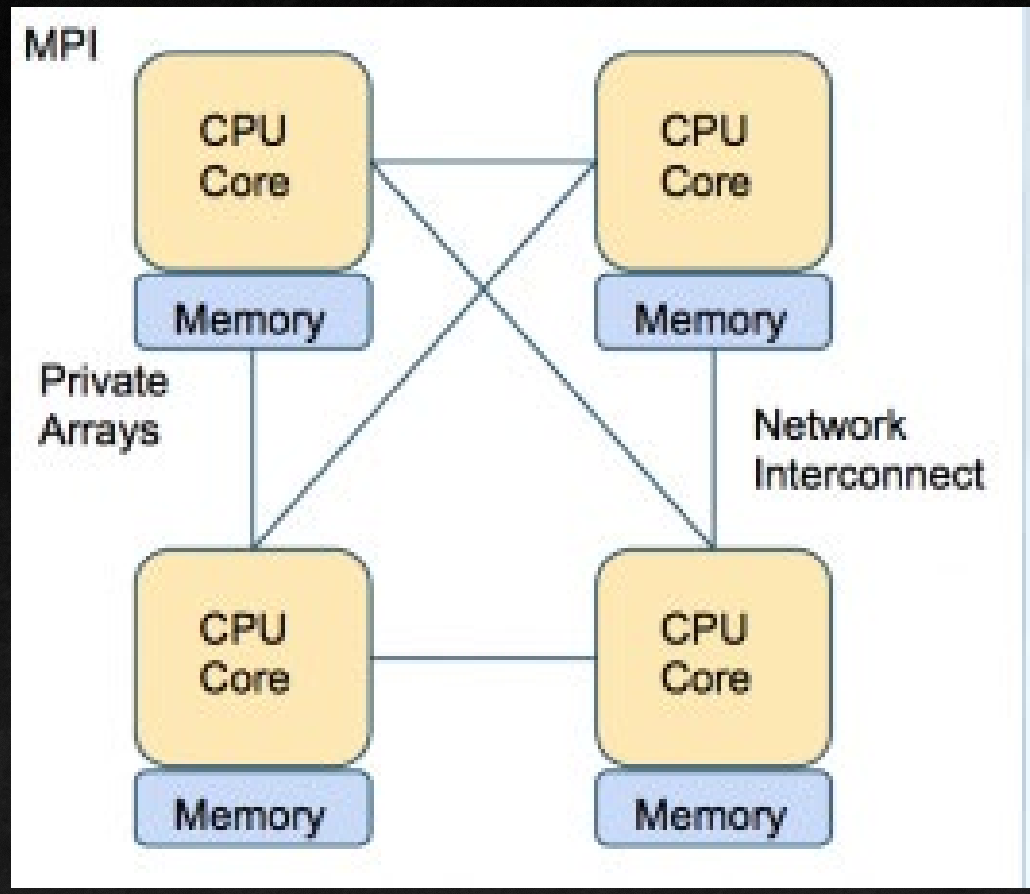
Need to communicate between nodes

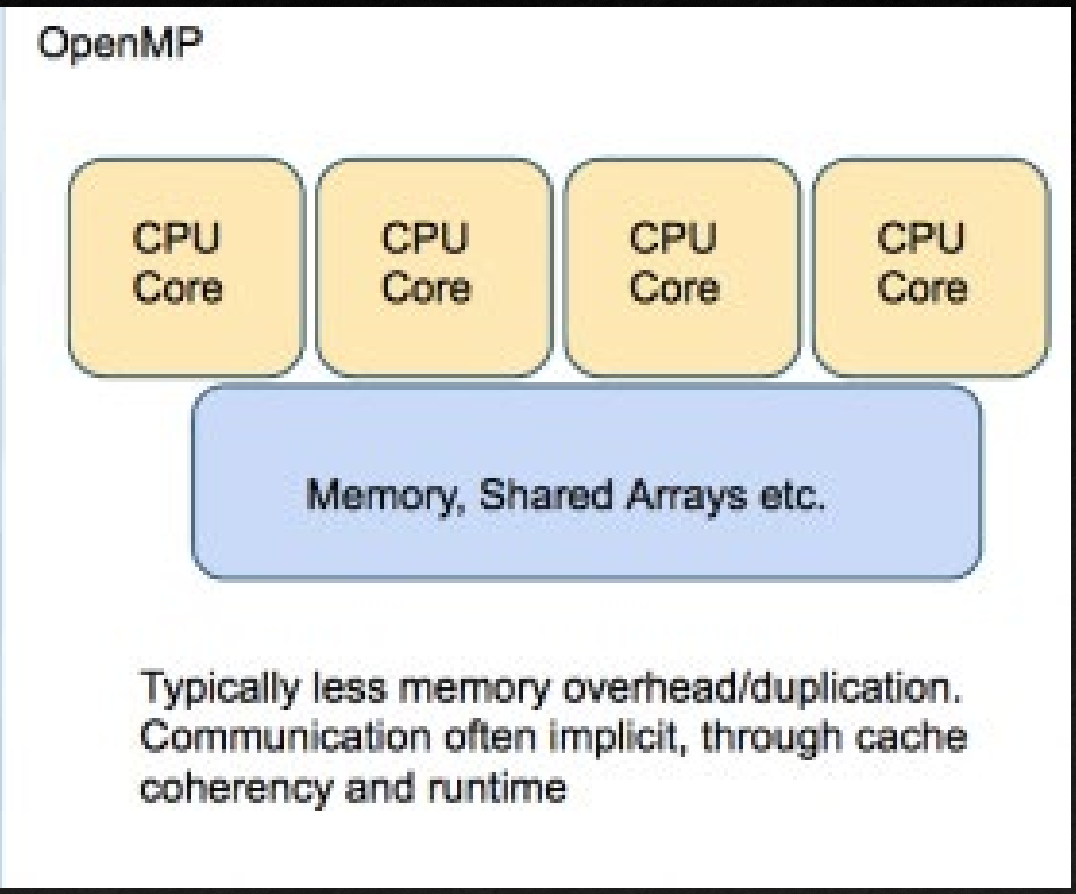






Parallel programming models

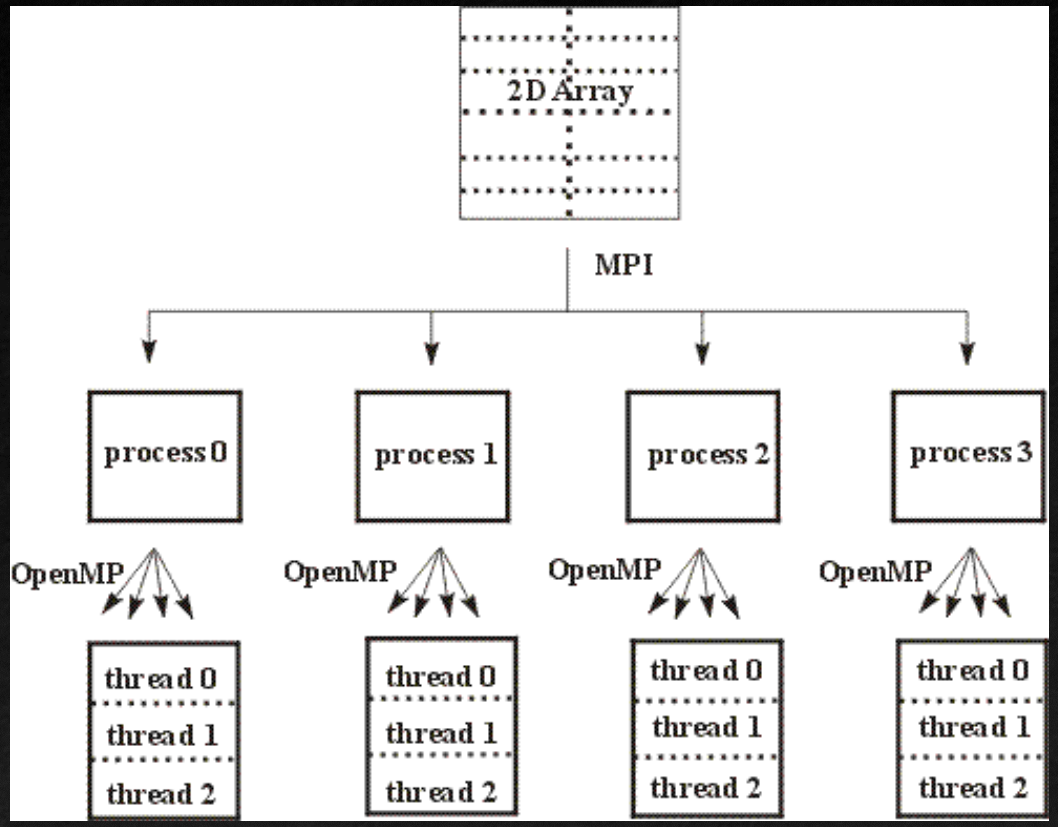




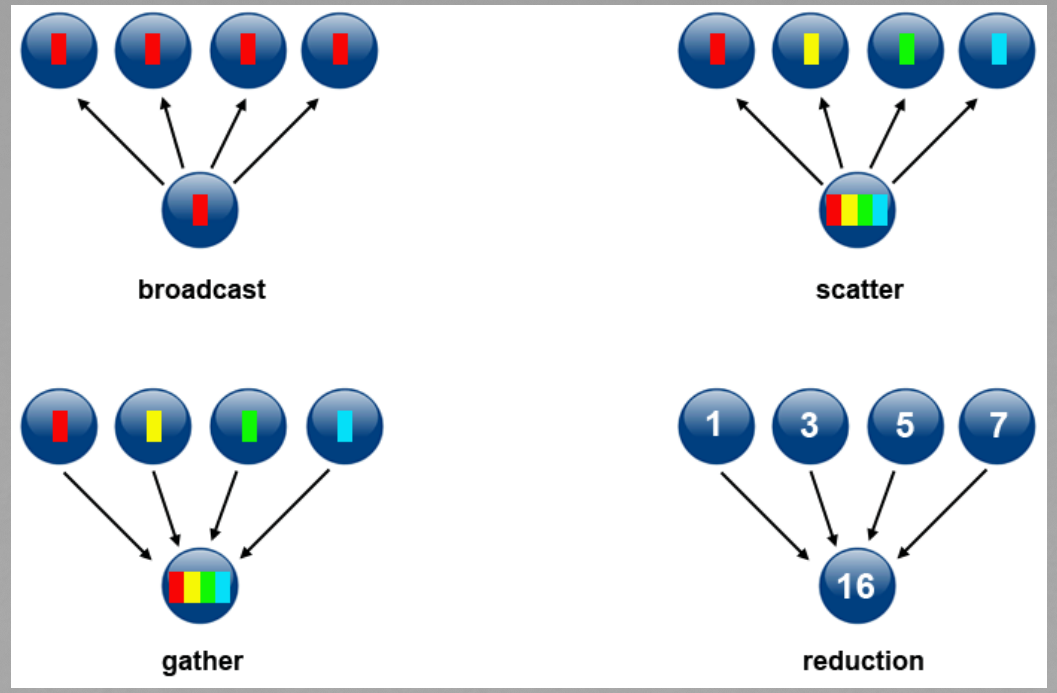
Distributed memory

Shared memory

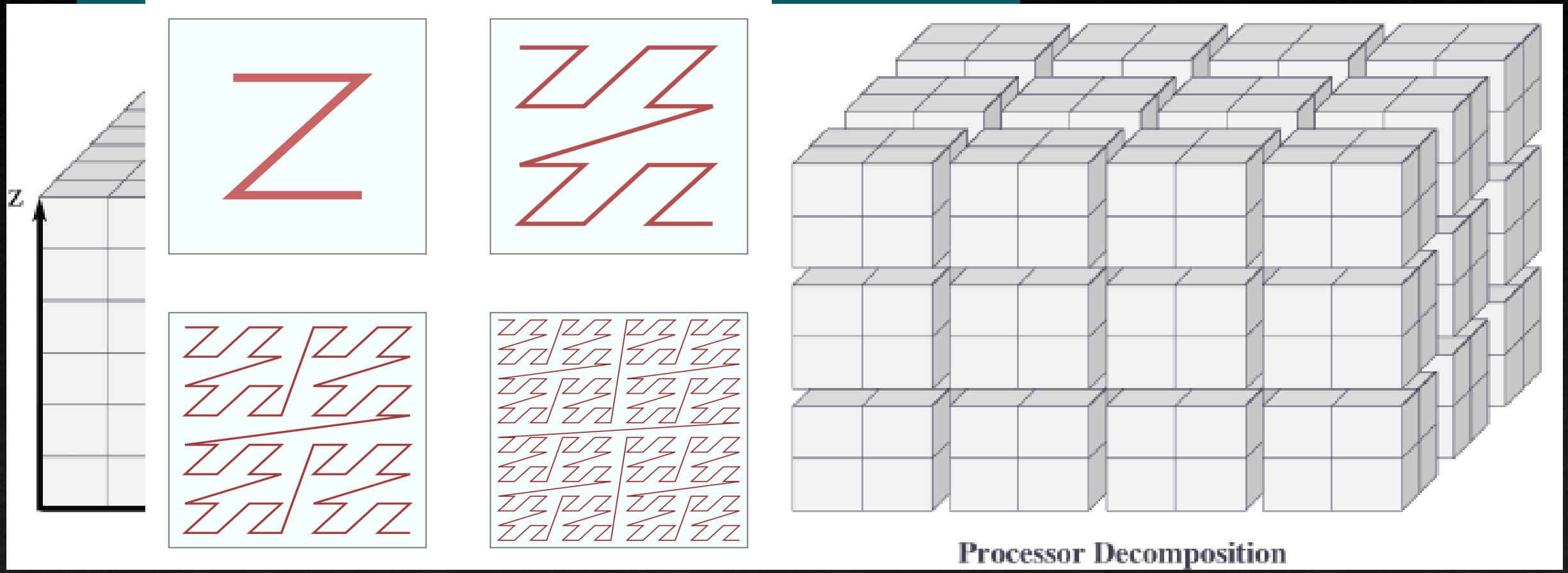
Parallel programming models



Can combine the two

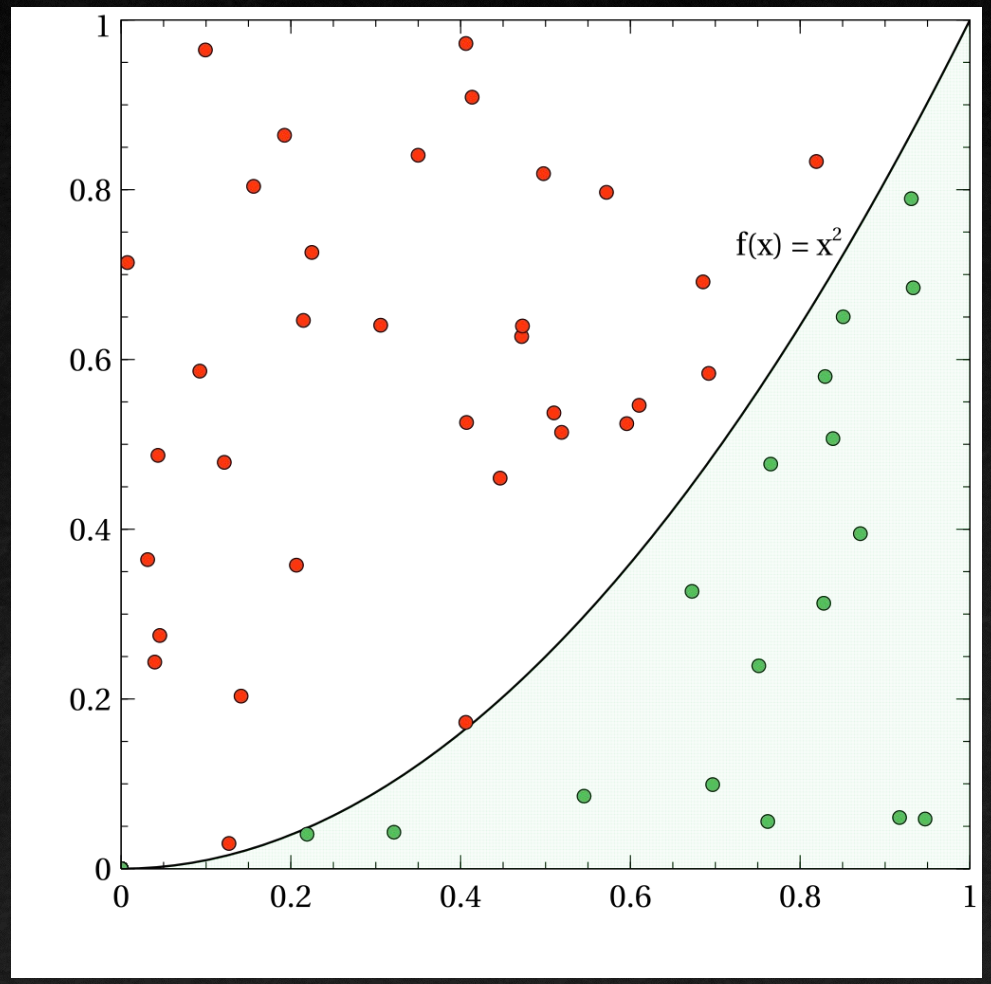


Sharing the computation



If the domain is spatial, we can split it into N parts,
each being calculated by a single core.

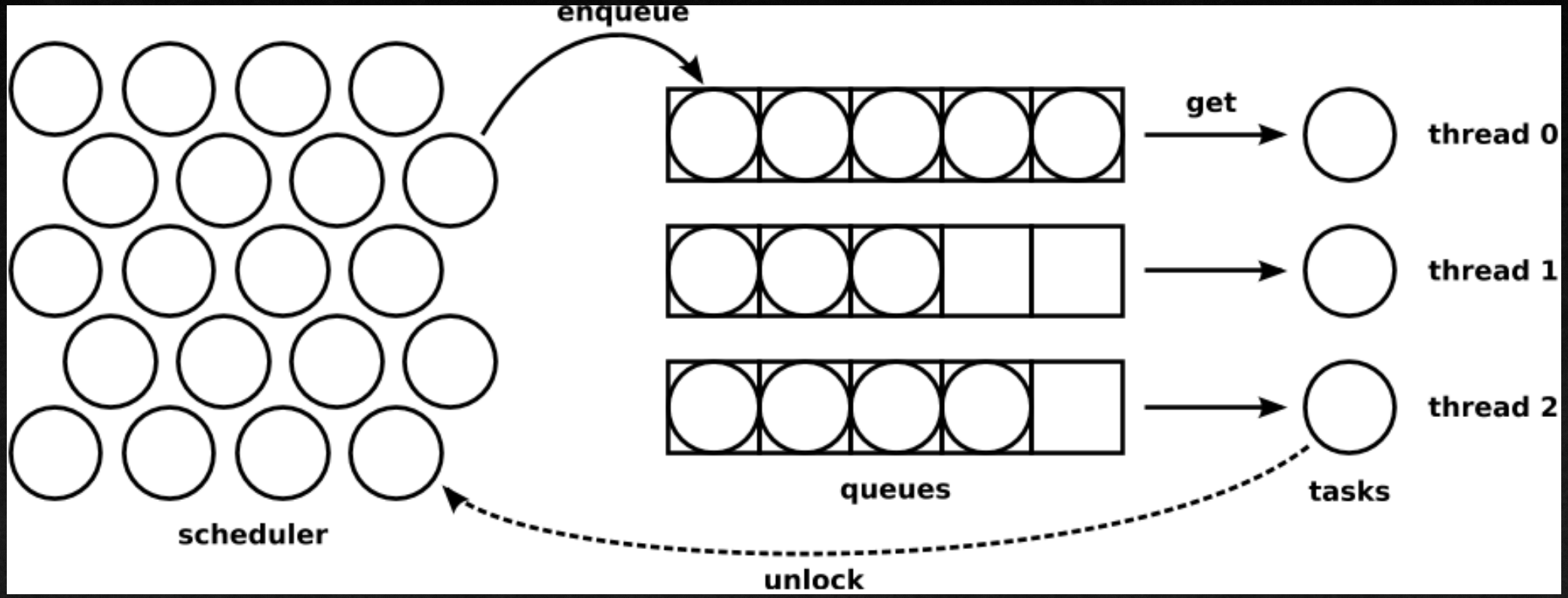
Sharing the computation



For Monte Carlo integration, each random number is independent and can be computed without communication until the very end

Sharing the computation

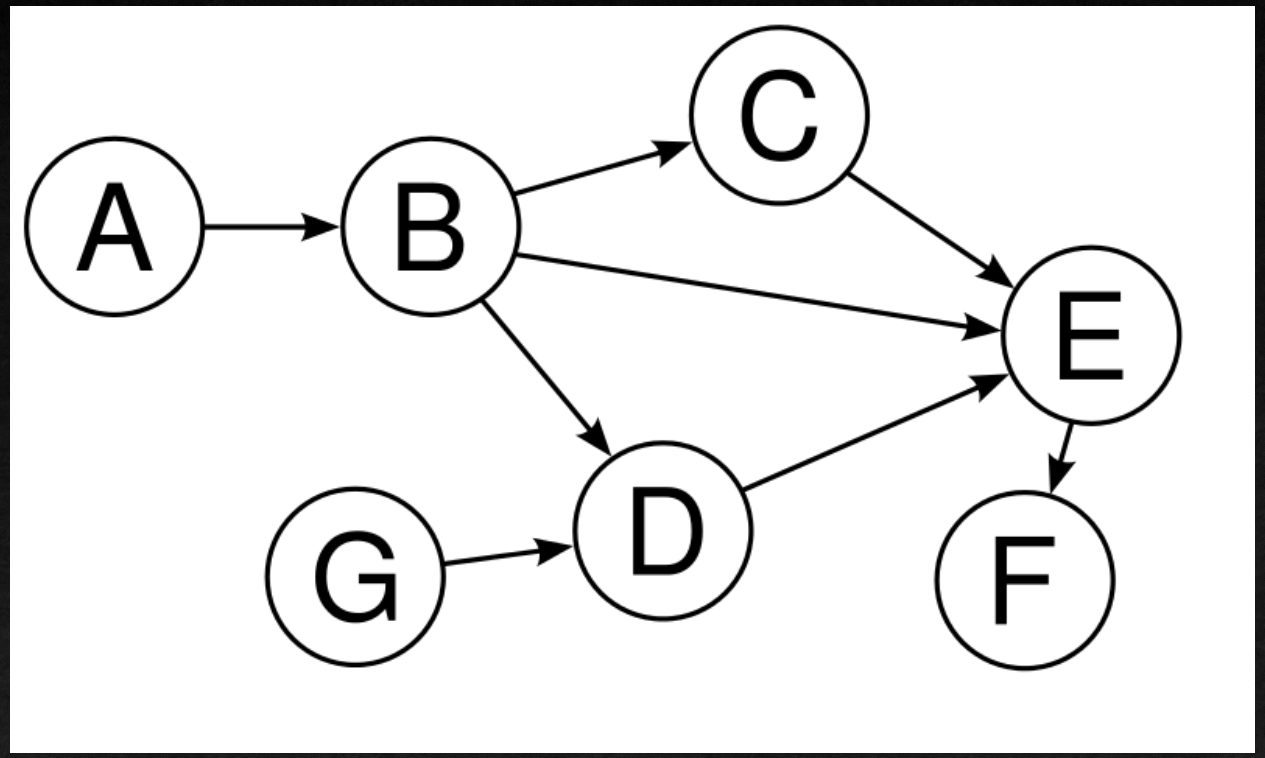
Task-based



Each task has data and dependencies associated with it

Sharing the computation

Task-based

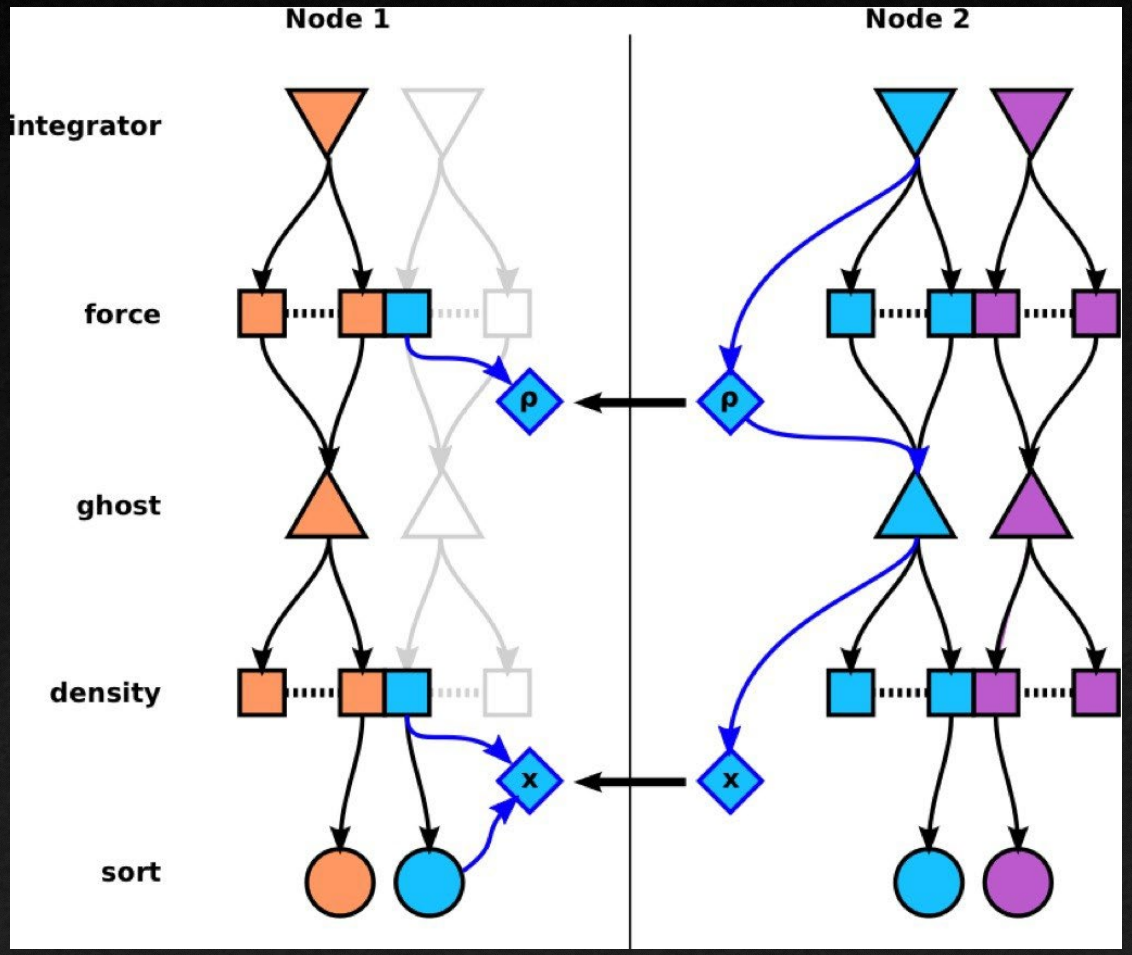


Each task has data and dependencies associated with it

Can be represented with a directed acyclic graph (DAG)

Sharing the computation

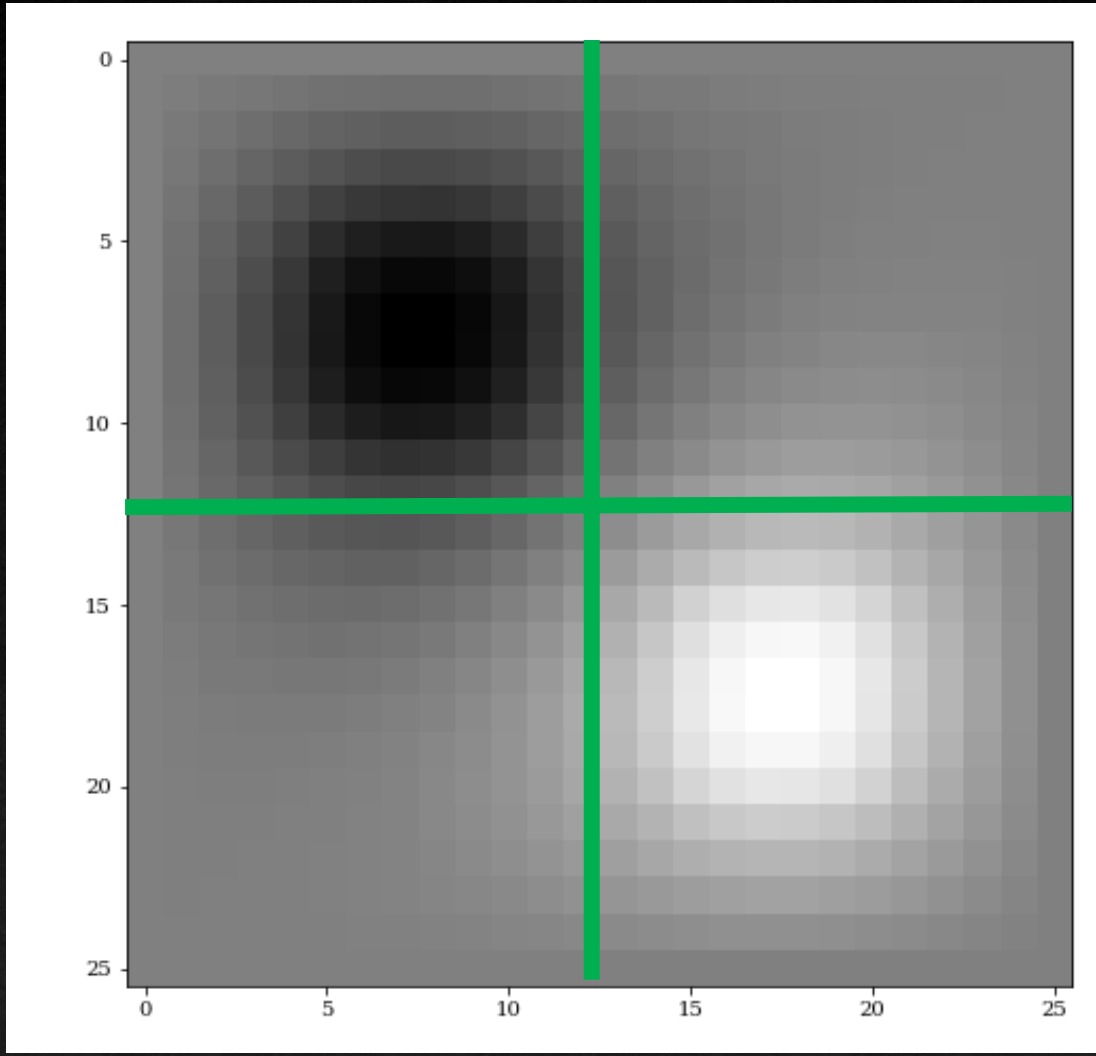
Task-based



Each task has data and dependencies associated with it

Can be represented with a directed acyclic graph (DAG)

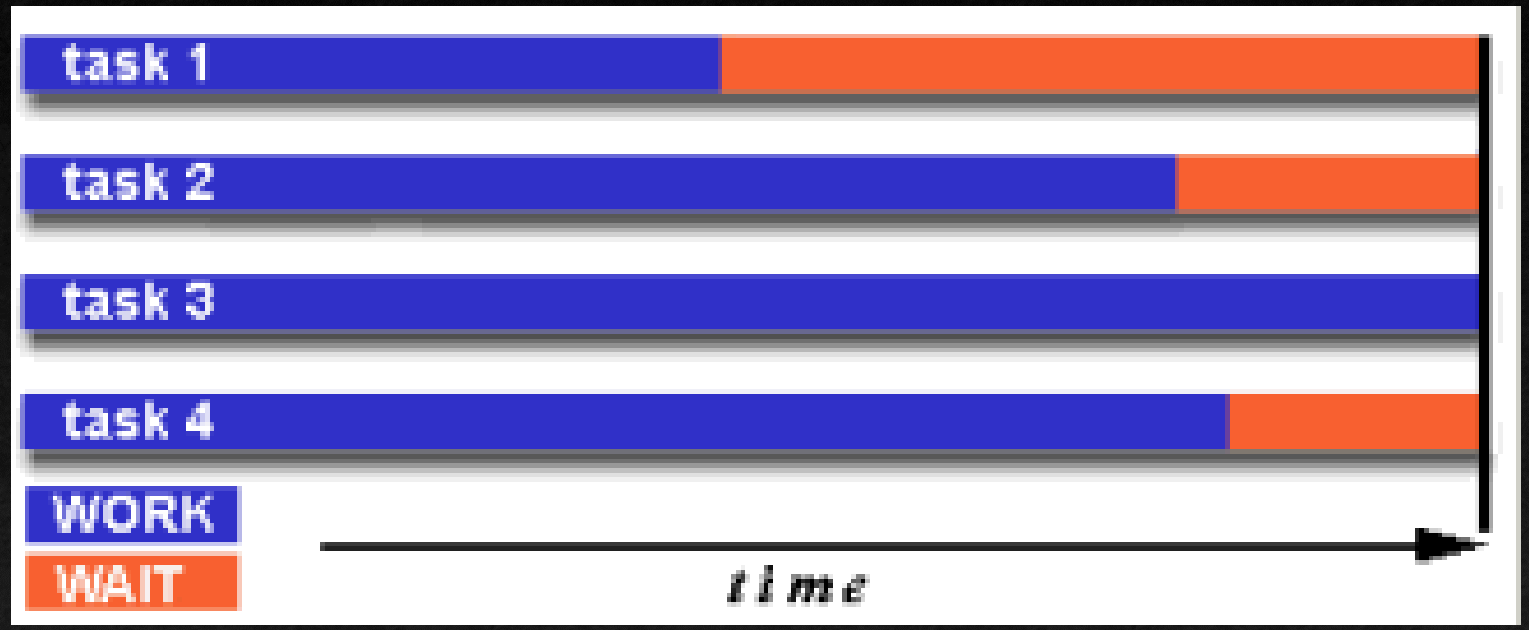
Sharing the computation



For spatially distributed calculations, the difficult lies in sharing the boundary conditions between partitions

In 3D, this is proportional to the surface-to-volume ratio of the partitions

Synchronization & Load balancing



Suppose that Task 3 takes the longest.

Then every core must wait (in the simplest parallel scheme) until it's finished to receive the boundary conditions to proceed to the next timestep.

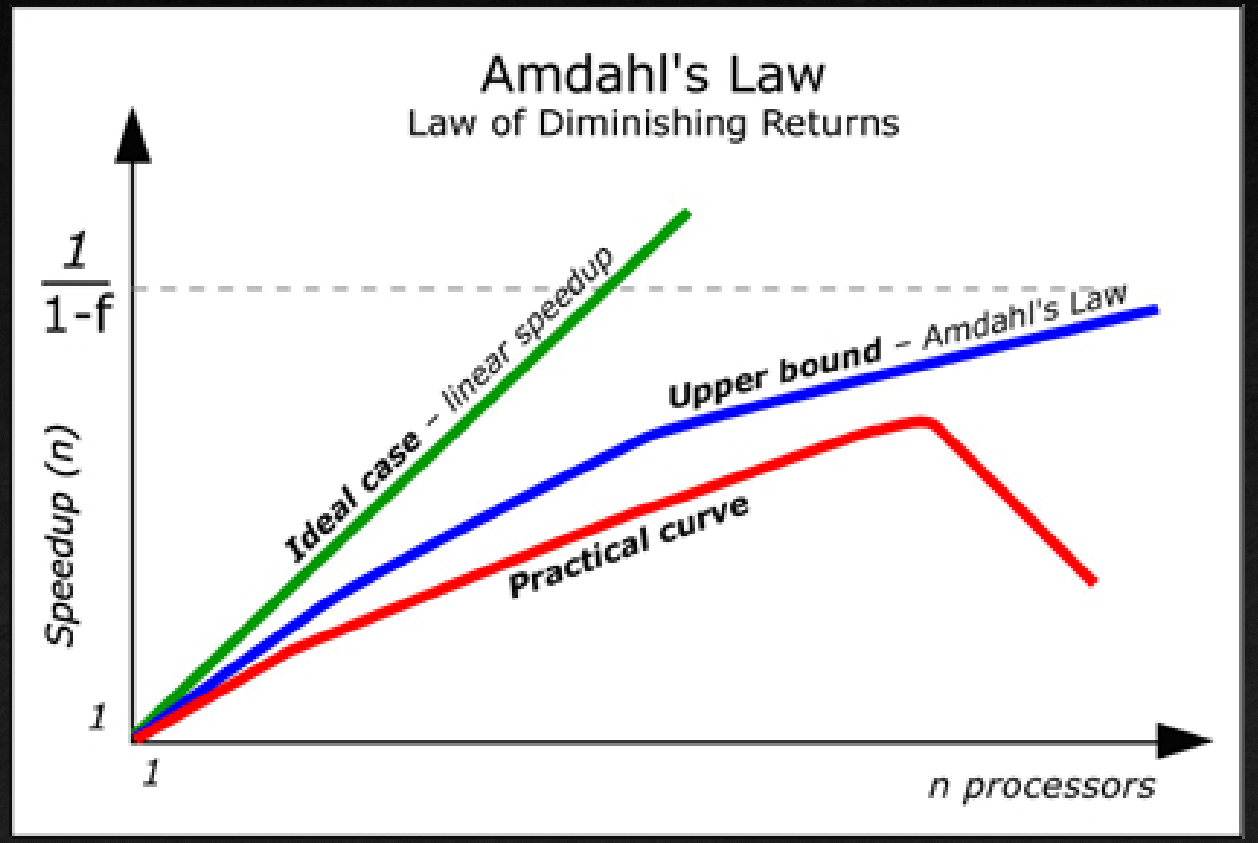
Parallel Efficiency: Amdahl's Law

- Consider a code that isn't completely load balanced and has a parallel efficiency P
- (1-P) is the amount of time waiting or the non-parallelized fraction
- This code is run on N cores
- The speed-up will be

$$S(N) = \frac{1}{(1-P) - \frac{P}{N}}$$

$$\lim_{n\to\infty} S(N) = \frac{1}{1-P}$$

- Doesn't include parallel overhead (communication between nodes)



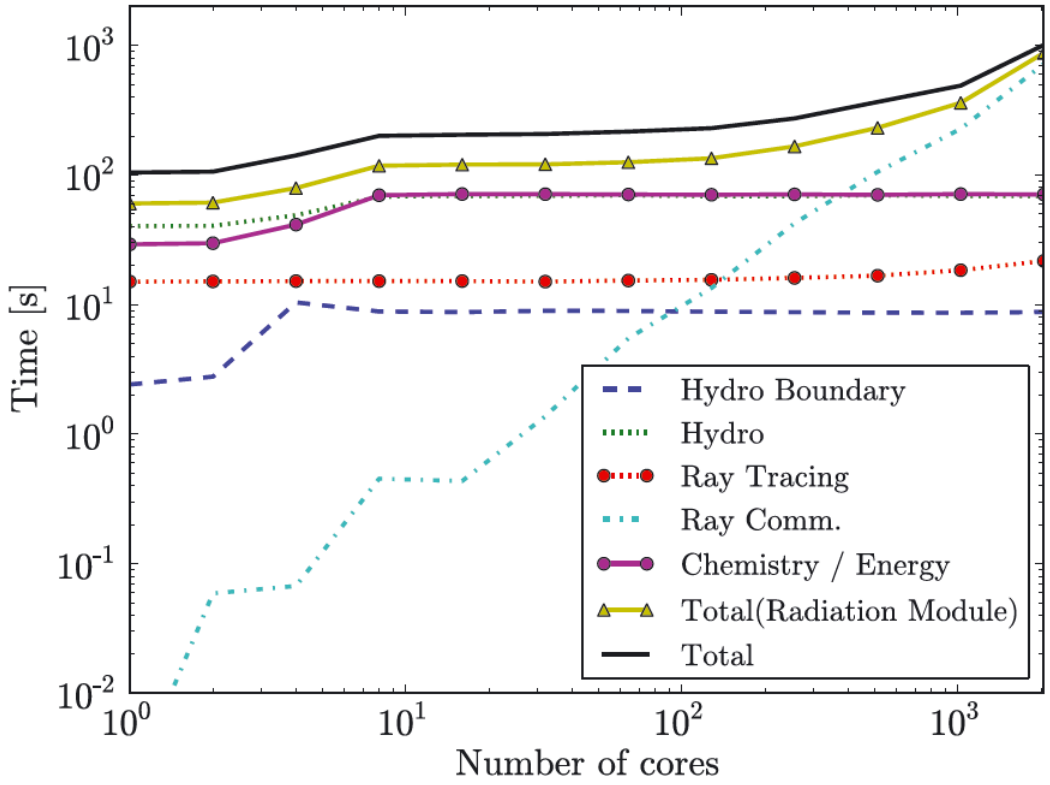


Figure 51. Weak-scaling test with one 64^3 block per process. Each block has one source and is set up similar to the radiation hydrodynamics Test 5. Above eight processes, all parts of the code exhibit good weak scaling except for the interprocessor ray communication. The radiation module timing includes the ray tracing, communication, chemistry and energy solver, and all other overheads associated with the radiation transport. The cache locality of the data causes the decrease in performance from one to eight processes.

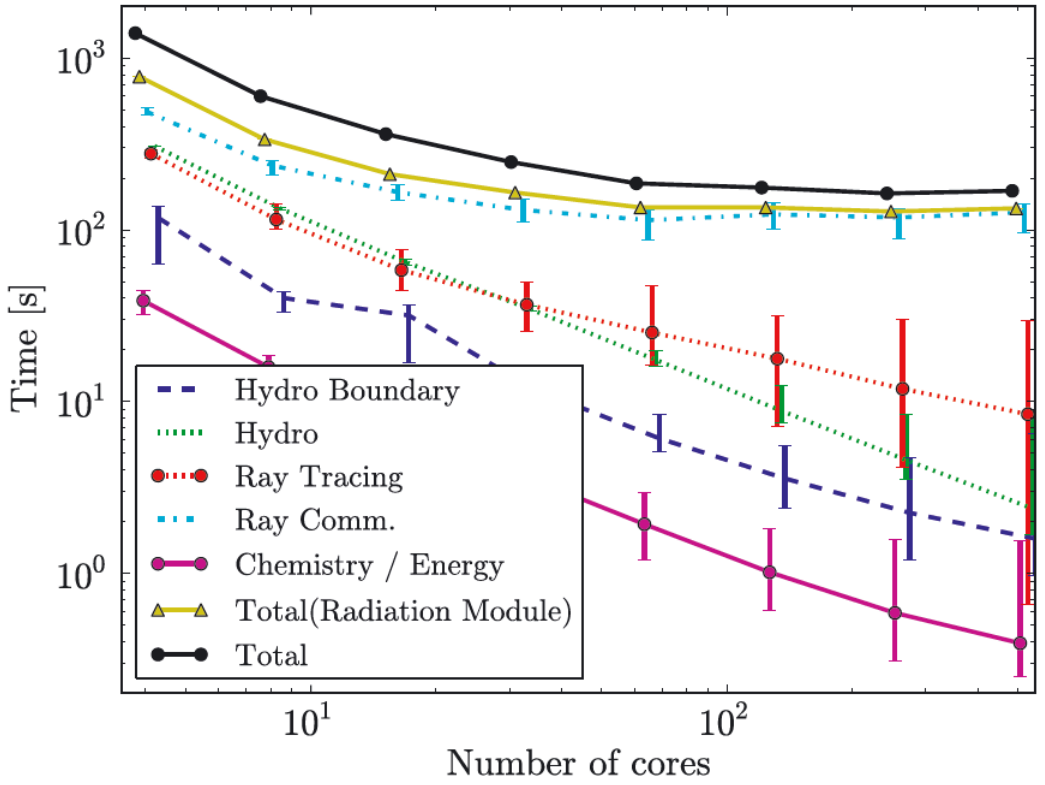


Figure 52. Strong-scaling test with a 256^3 AMR calculation of cosmological reionization with $\sim 392^3$ zones. The error bars represent the minimum and maximum time spent on a single core and measures load balancing. Each point has been slightly offset to make the error bars distinguishable. The hydrodynamics and non-equilibrium chemistry solvers scale well. The communication of rays does not scale well in this problem and limits the performance, whereas the ray tracing scales relatively well.