PSC II : the parallel part.

Distributed memory computing

OpenMP "shared memory parallel"

Parallel data structures & algorithms

For grid-based PDE discretisations.

- Linear algebra
  - Solving $Ax = b$
  - Multigrid
    - Hierarchies of grids
    - ⇒ Fast solver for $Ax=b$ given some properties of A.
- Discretised operators
  - $\partial_t u - \nabla^2 u = f$
  - -!-
- Some parallel performance

- Live coding as demonstration
- "Blackboard" slides
- Read and talk about some papers.

Introduce in this course two software libraries' C libraries

- MPI : message passing interface
- PETSc : portable extensible toolkit for scientific computation

Programming in python.
→ mpi4py & petsc4py.

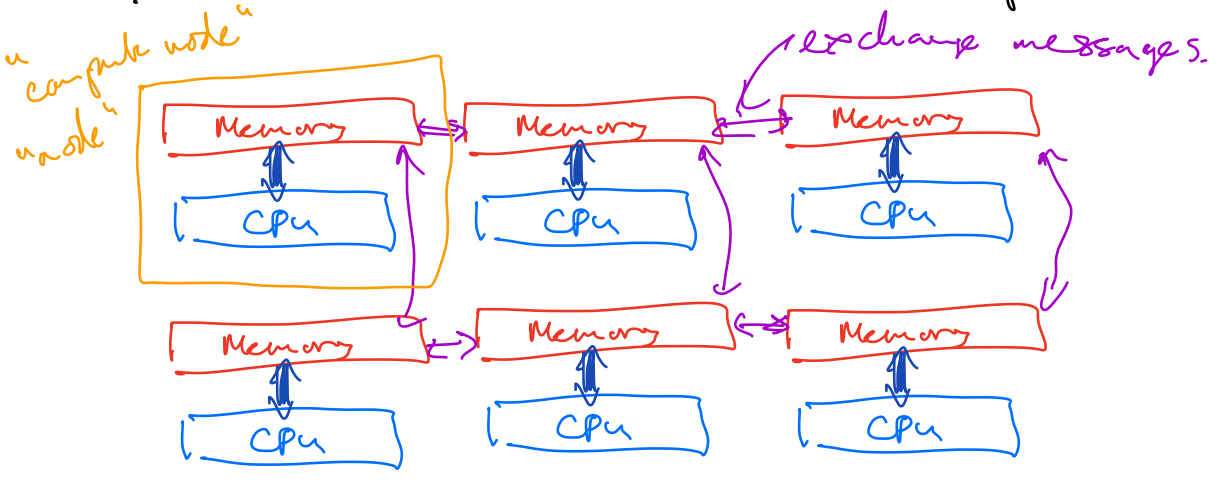
Parallel computing: lots of work building data structures: "kind of boring"
→ want to use them to solve some problems.

Why distributed memory?

Physics gets in the way if we want
> ~128 cores in a single computer.
$O(£5000)$.
Limited in size of problem you can solve by: compute capacity
available RAM.
What size of problem might we want to solve?
→ Want better aircraft design
Jumbo ~ 100m long 60m high
100m wide.
$6 \times 10^6 m^3$.
Jet planes fly fast, → airflow is turbulent.
- need mm resolution to capture this
$10^9 mm^3$ in $1 m^3$
~ 10^{15} grid boxes for my plane at 1mm resolution.

~ 4 Petabytes of data with 1 double per box.

⟹ Need more than one computer.



MPI is a library for programming such systems.

Challenges we want to address:

1. Each node can only see a part of the whole picture at any one time.
   → Exploit structure in the problem to distribute it.

2. We want solution algorithms that are at worst $O(N \log N)$ in the size of the problem. (In terms of resolution, for example).
   → Goal: keep runtime fixed when increasing problem size by adding more compute.

For PDEs it turns out there is structured sparsity in the problem that allows us to address both of these. (In many cases).

$\Longrightarrow$ MPI codes "hello world".