CMOR 421/521:

Distributed Parallelism

M	W	F	
■	■	■	1
⬛	■	■	2
■	■	■	3
■	■	■	4
■	■	⬛	5
■	■	🟨	6
			7
▨	▨	▨	8
			9
⬛	⬛	⬛	10
			11
			12
			13
			14
			15

Topics

- Why do we need communication?
- Have we seen communication before?
  - Other instances of communication in technology
- MPI and its implementations
  - Using MPI

Why Do We Need Communication?

- **Reason 1:** We want more workers!
  - More workers -> more parallelization
  - Most modern CPUs have multiple cores; **what if we want to parallelize across multiple CPUs?**
    - They may or may not not have any shared memory
- **Why not just put more cores on a single CPU?**
  - People (i.e. chip manufacturers) are doing this
  - Actually the motivation behind GPUs
  - **Not all problems lend themselves to huge amounts of cores**
  - **Caching:** the further from the chip, the slower the memory access. *Shared memory can degrade performance*.

Why Do We Need Communication?

- **Reason 2:** We want more capable workers!
  - Some problems require so much memory that we NEED to distribute the memory across multiple CPUs just so the problem can be represented
  - **More cores won’t fix this**
- **Why not just put more memory on a single CPU?**
  - People (i.e. chip manufacturers) are doing this
  - There are monster CPUs with huge amounts of memory
  - **Not all problems need huge amounts of memory**
  - **Caching:** the larger a cache, the slower the memory access

Why Do We Need Communication?

- **Bigger and badder CPUs aren’t (always) the answer**
  - A better CPU will be optimal for some problems
  - Optimization is often wrt a given problem
  - Not all problems look the same; optimal for who/what?
  - **Optimization can hurt generalization**
- Being able to parallelize without the assumption of shared memory allows us to parallelize across multiple CPUs/GPUs
- It’s also backwards compatible: it can work with shared memory too

Most supercomputers are distributed

Have We Seen Communication Before?

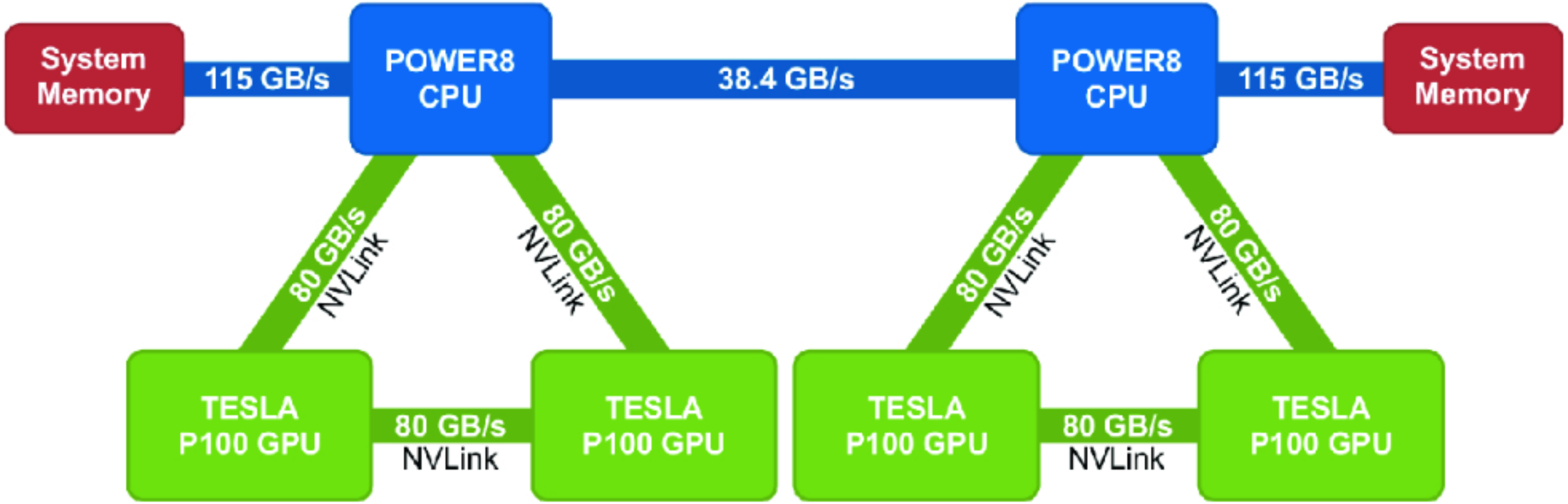
- **Yes! Everyday**
- **Example 1: The Internet**
  - The internet is just file sharing
  - The web address is actually a file address
  - That file lives on another computer somewhere
  - What if we want to access that file 24/7? What if they power their computer off?
    - Dedicated computer for “serving” files, aka “servers”

Have We Seen Communication Before?

- **The internet and HPC have some things in common**
- **Before WiFi…**
  - Computers were connected to the internet via ethernet cables and the like
  - Supercomputers often connect multiple CPUs via ethernet cables
  - There are more advanced connections that have evolved specifically for supercomputing/clusters

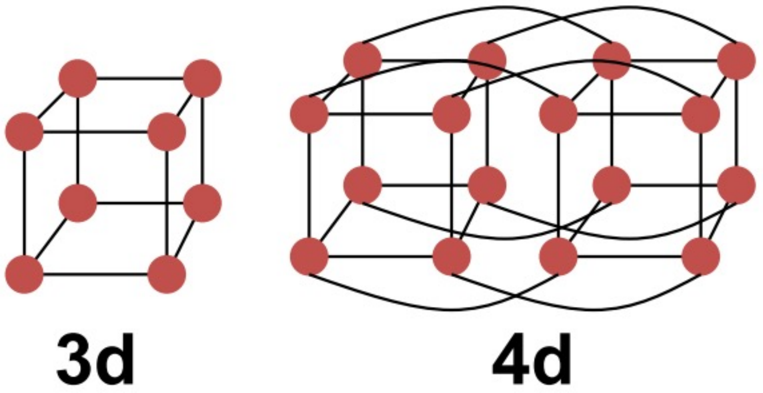
Have We Seen Communication Before?

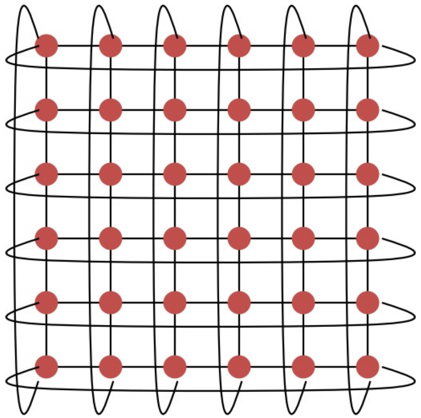
- **Example:** An Nvidia Power8 GPU Node

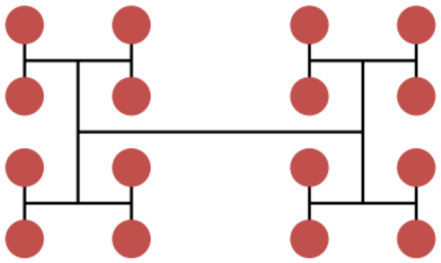


Some examples of network *topologies*

- Line, ring, tree, star, mesh, torus, hypercube
  - Ring is surprisingly common and efficient, especially for a small number of distributed nodes
- More complex examples: butterfly, dragonfly
- Diameter and “bisection bandwidth”







Course Overview:

- Pre-parallelism
- Non-communicating Parallelism
- Communicating Parallelism <- We are here
- Multi-Node Parallelism

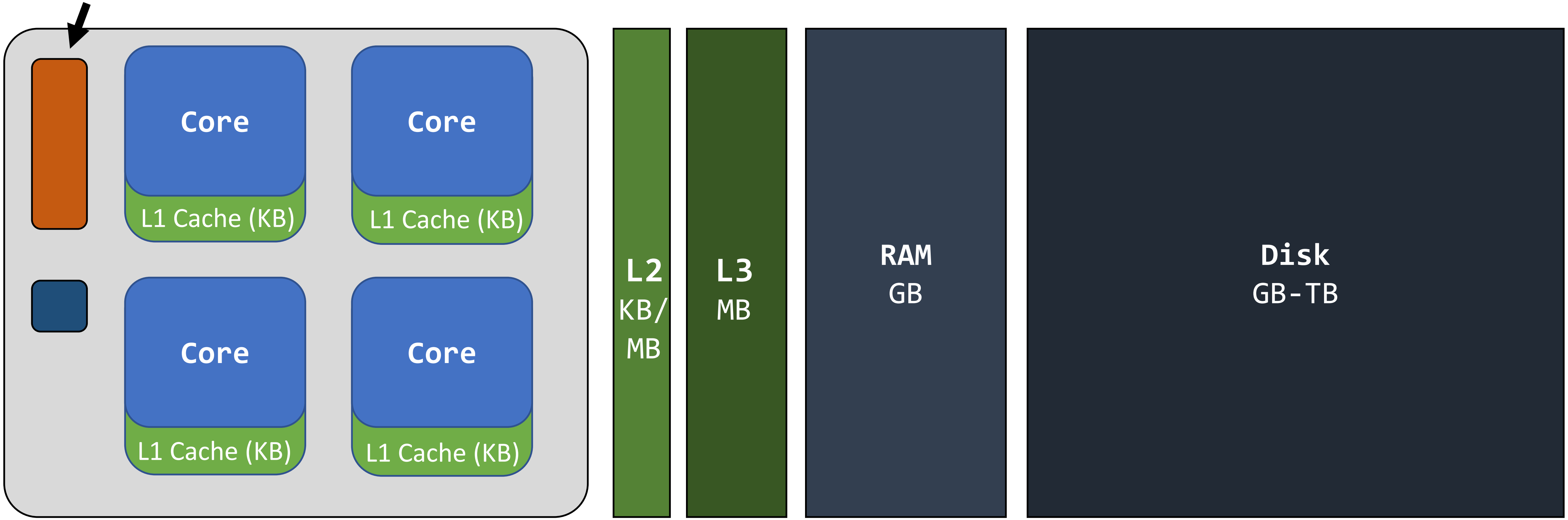
Communication on a Single CPU?

Why bother on single CPUs?

- It’s a different paradigm
- It’s a more powerful paradigm: it can extend to systems without shared memory
- “a well-used system is never idle…”
- There’s more “workers” on a CPUs that just cores

Computer Architecture

Memory unit: this moves memory to and from caches, RAM, and disk



A Specialized Worker is Still a Worker

Multiple workers mean we can parallelize work

- Parallel work is often split into two kinds:
  - Computation (what we’ve looked at so far)
  - Communication
- The two can be overlapped, i.e. run in parallel!
- A parallel paradigm that handles communication explicitly can overlap the two
  - Sorry OpenMP :(

What is MPI?

- **MPI = Message Passing Interface**
  - Similar to how the internet has standards (rules) for serving files (providing information), HPC needs a standard for communication
- MPI is actually just a standard
  - Think of IEEE, ISO, ASTM, ASME, etc
- There are multiple implementations of that standard; we will use OpenMPI (not OpenMP)
  - OpenMP: Open (source) Multi-Processing
  - OpenMPI: Open (source) Message Passing Interface
  - MPICH is another implementation of MPI

What Carries Over?

Communicating parallelism entails both computation and communication

- We’ve been looking at parallelizing computation up until this point
- Everything we’ve done up til now, we’ll keep doing:
  - Domain decomposition
  - Thread mapping (except now we’ll call it process mapping)
- But now we’ll also add communication and remove the assumption of shared memory

How do people usually expose parallelism?

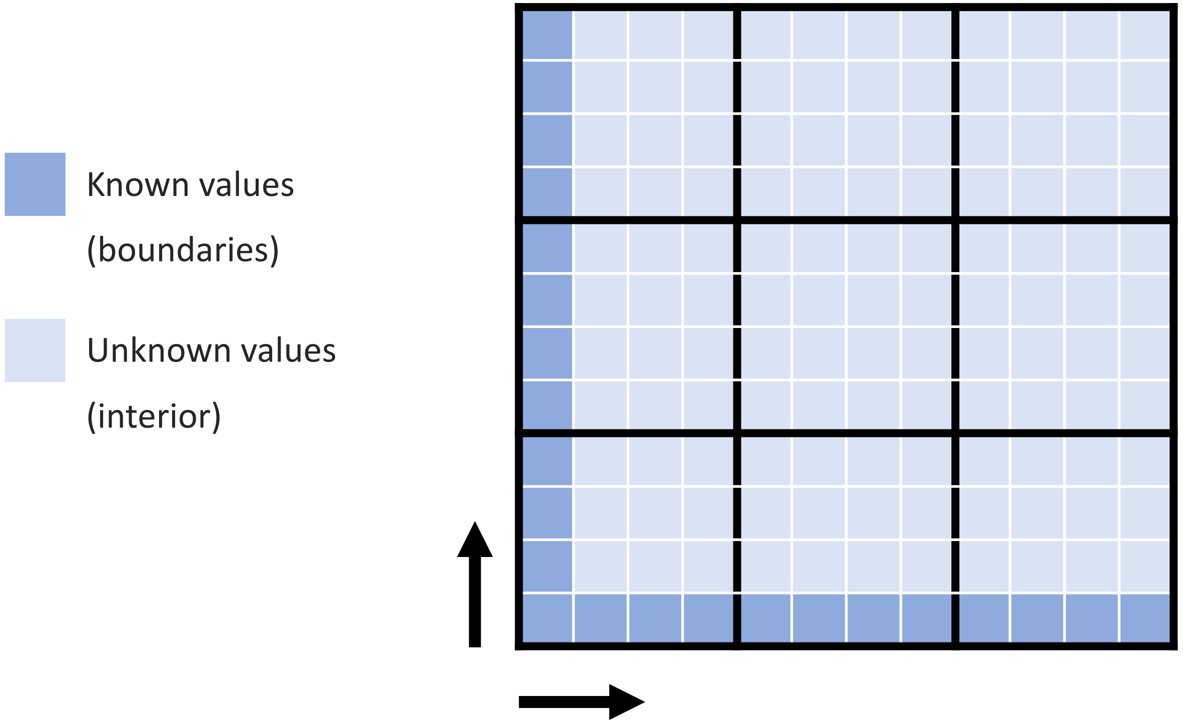
Some examples

- Dense matrix-matrix multiplication
- Large systems of ODEs
  - Explicit and implicit time-stepping
- Grid or stencil-based methods (finite difference, finite element methods, etc)

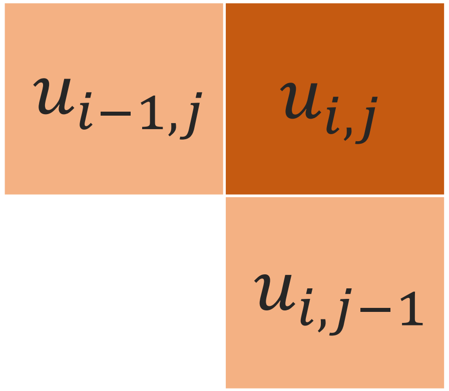
Old Friends: The FD BVP

- How might this problem look different now?

Domain



FD stencil



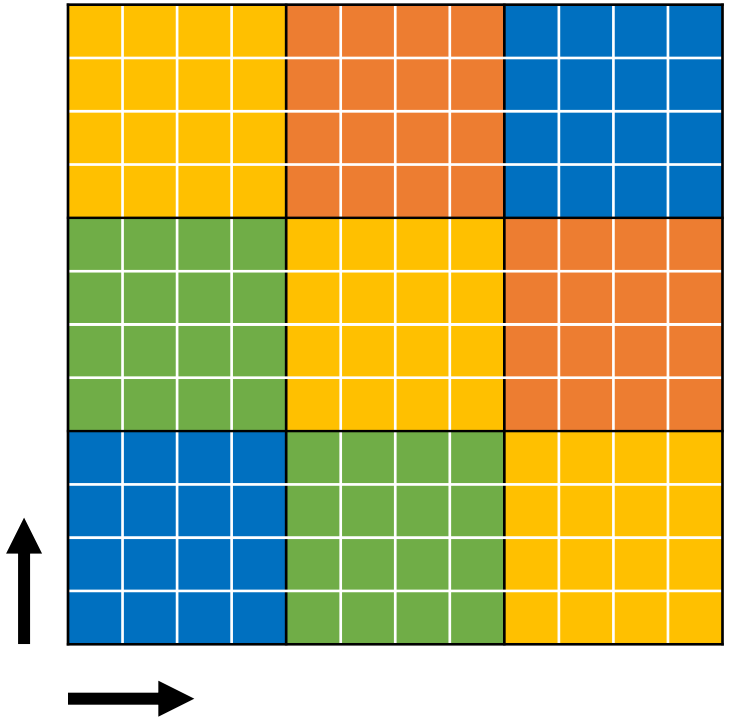
The FD BVP with Communication

The order dependency is exactly the same

- We have to parallelize using a wavefront scheme

But no, there is no shared memory…

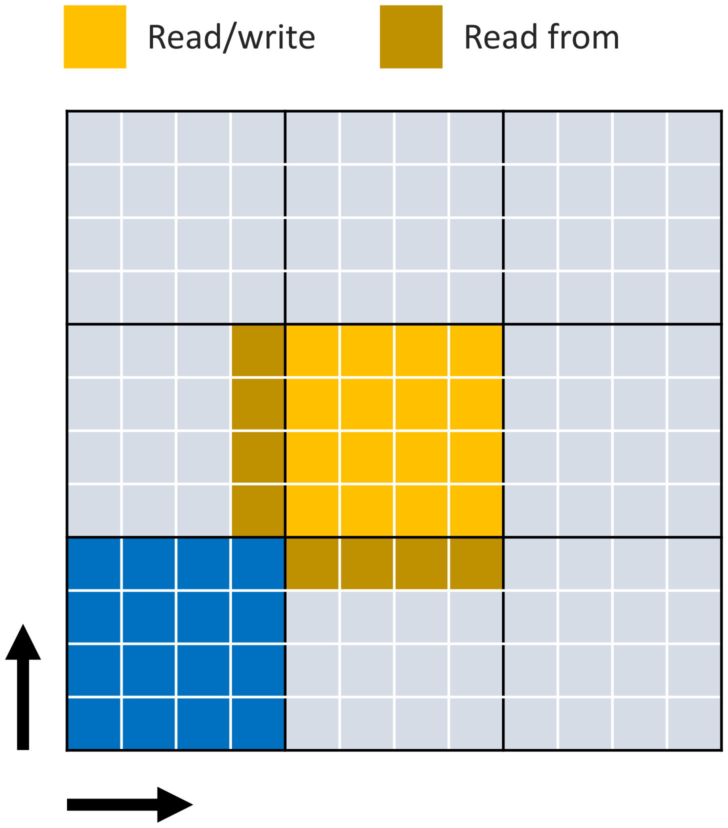
- The blocks tell us what values we assign to, not the values we need (i.e. the values we read from)
- We're assuming there is no shared memory; how do blocks get this data?



The FD BVP with Communication

Blocks now have to send and receive information between each parallel step

- There is a “halo” of values around the block that are needed for computation
  - These are also called ghost values
- Those values are computed by different blocks (or maybe not at all)
- The blue block has no halo/ghost values since it is on the boundary
- **We have more work to do now…**



Using MPI

OpenMP

```
#include <omp.h>

int main() {
   #pragma omp parallel …
   {
      int nT, ID;
      nt = omp_get_num_threads();
      ID = omp_get_thread_num();
   }
   return 0;
}
```

```
> g++ -o prog –fopenmp <files>
> export OMP_NUM_THREADS=4
> ./prog
```

MPI

```
#include <mpi.h>

int main() {
    MPI_Init(NULL, NULL);
    int nR, rank;
    MPI_Comm_size(MPI_COMM_WORLD, &nR);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Finalize();
    return 0;
}
```

```
> mpic++ -o prog <files>
> mpirun -nR ./prog
```

New Syntax and Terminology!

- MPI has a stronger C “flavor” than OpenMP
- You do a lot of pass by pointer

OpenMP	MPI
Thread ID	Rank
Thread	Process
Thread Team	Groups/Communicators

Everything You Can do in OpenMP…

… Can be done in MPI:

- From OpenMP:
  - Parallel regions
  - Parallel-for
  - Tasks
  - Sections
- From the user:
  - Load-balancing
  - Thread mapping

Installing

- OpenMPI vs MPICH
  - Two different implementations of an MPI *standard*
  - OpenMPI’s goal: support the most common MPI functions
  - MPICH’s goal: implement everything in the MPI standard
  - More details: https://stackoverflow.com/a/25493270
- On Mac, “brew update; brew install open-mpi”
- mpicc, mpic++, etc – these are all wrappers