Computer Science for the Physical Sciences

Week 9

Craig Rasmussen (Research Support Services, University of Oregon)

Computer Science Minor: *This Week*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures
- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix nm
  - Operating Systems
  - Automata Theory
  - Software Methodology I-II Debugging MPI Parallelism
  - Databases
  - Computational Science Numerical Differentiation
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning

China: “Fastest” Computer in the World
34 Petaflops

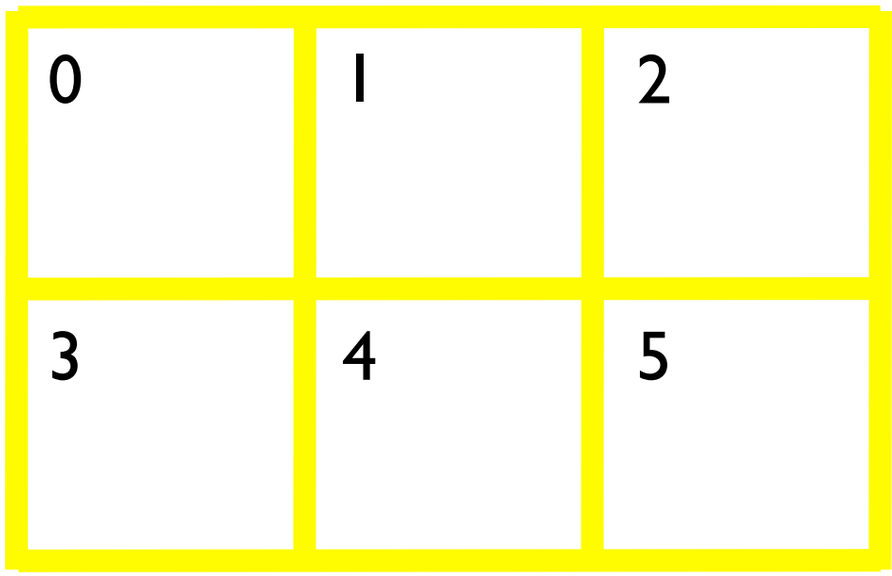
ACISS System - Compute Resources

- Basic node (17 total Teraflops)
  - 128 ProLiant SL390 G7
  - Two Intel X5650 2.66 GHz 6-core CPUs per node (1,536 total cores)
  - 72 GB DDR3 RAM per basic node

Communication is paramount!

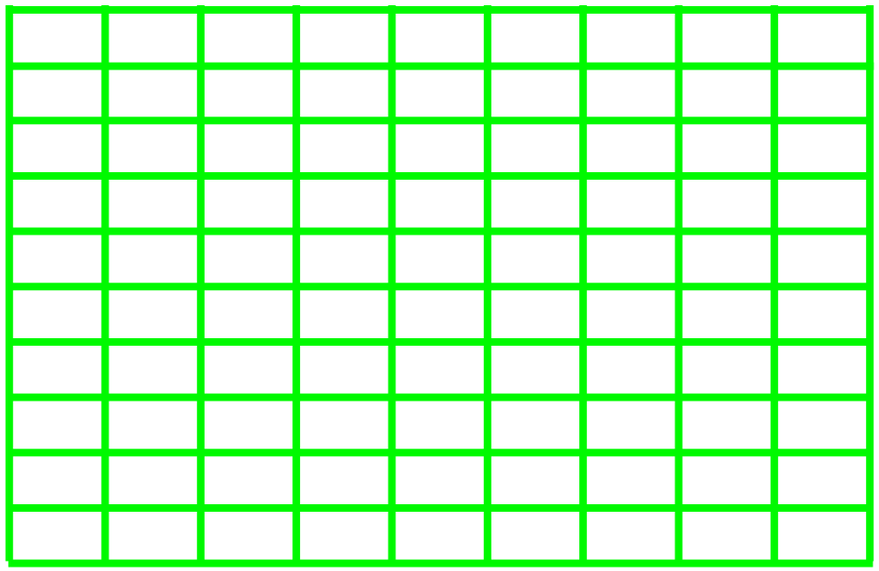
Processor Topology

- A set of P processes can be arranged in a virtual grid of Mp*Np = P processes
- 2 rows by 3 columns = 6 processes



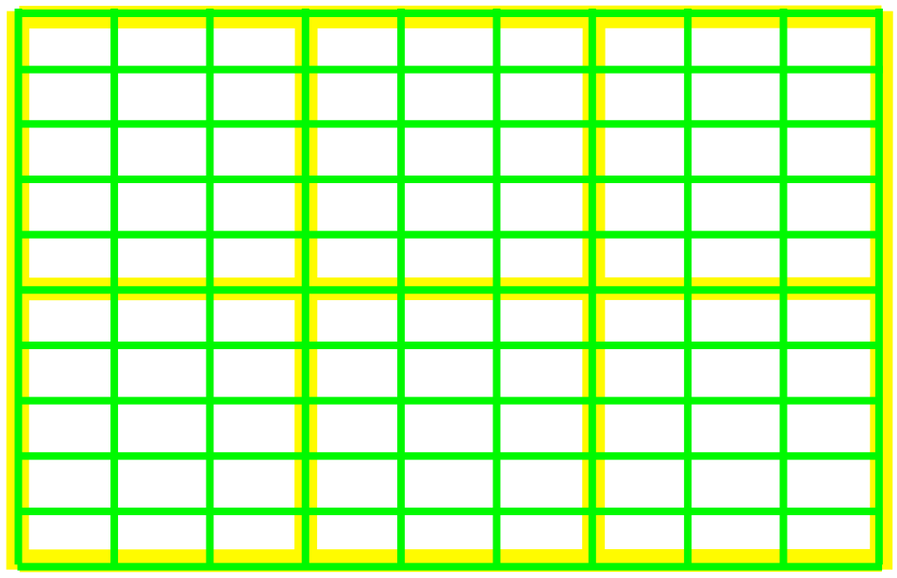
Data Grid Topology

- Cells of data can be arranged in a grid of M*N = L cells
- 10 rows by 9 columns = 90 cells



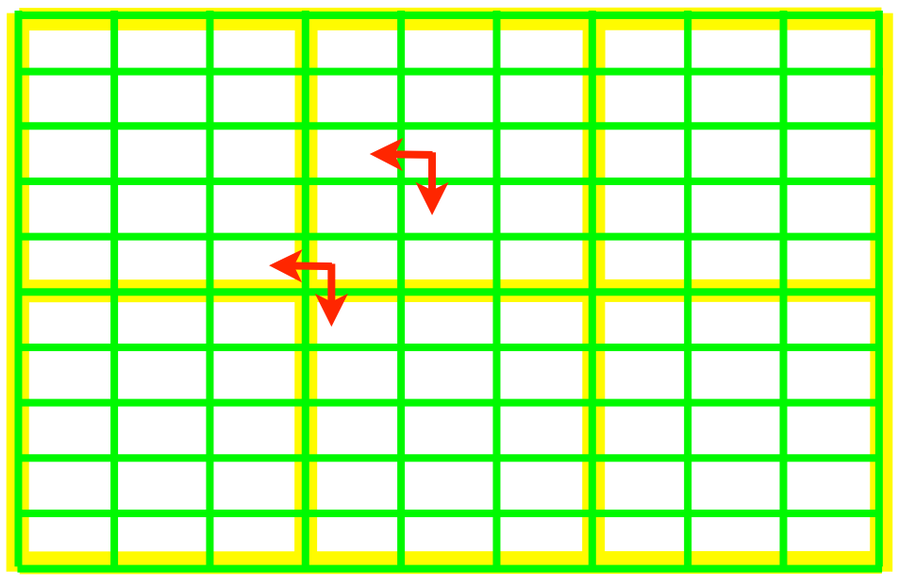
Data Parallelism (SPMD)

- Distribute a group of cells to a processor
- Scales well (up to 100 K + processors)
  - weak scaling: as cell count increases, use more processors
- 5 rows by 3 columns = 15 cells per processor



Data Parallelism: MPI Communication

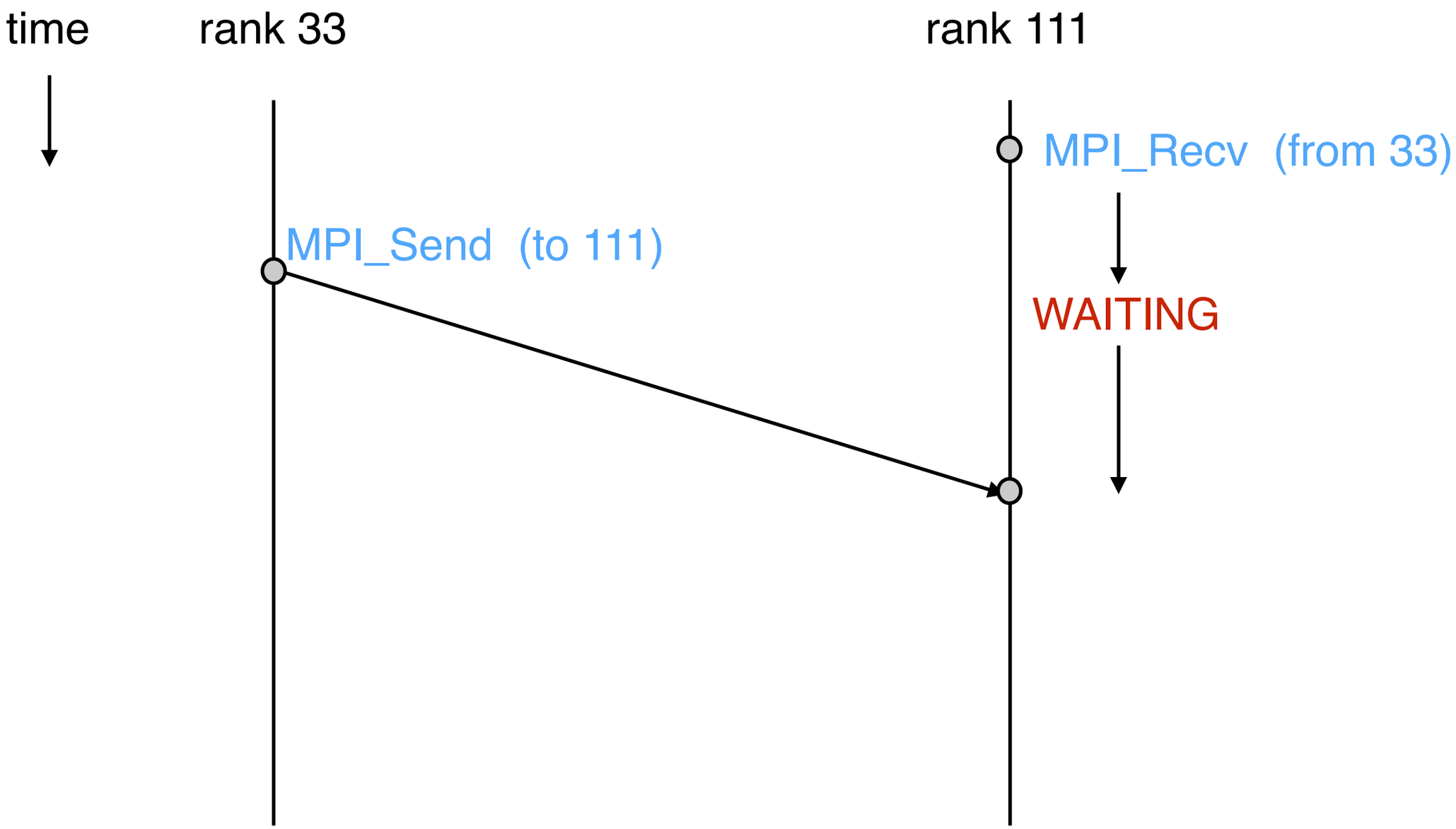
- Communication is usually between neighboring processors
  - communication costs scale as processor surface
  - computation done on volume (# of cells)
- Global communication often scales O(log(P))
  - calculate an average over all cells (e.g., average temperature)



Parallelism with MPI: *An overview*

- MPI is remarkably easy
- Six function MPI
  - MPI_Init # initialize the MPI library
  - MPI_Comm_size # how many processes in the communicator
  - MPI_Comm_rank # what is my process # in the communicator
  - MPI_Send # send a buffer to a process (blocking)
  - MPI_Recv # receive from a process (blocking)
  - MPI_Finalize # finished using MPI
- MPI_COMM_WORLD is the default communicator that includes all processes
  - subsets of MPI_COMM_WORLD cab be taken

Parallelism with MPI: *Matching send and recv*



Parallelism with MPI: *Extra functions*

- More MPI functions
  - MPI_Barrier # block until all processes reach this point
  - MPI_Reduce # sum reduction (eg) of all elements in an array
  - MPI_Broadcast # broadcast a buffer to everyone
  - MPI_Scatter # scatter sections of a buffer to everyone
  - MPI_Gather # gather sections of a buffer from everyone

Shell Command: *nm*

- nm - display name list (symbol table)
- Steps to build an executable from a source file (*.f90)
  - compile program -> $(FC) -c -I include_path hello.f90
  - link program -> $(FC) -o hello hello.o -L library_path -lsome_library
- hello.o is an object file and contains symbols (e.g., functions) and code
- hello is an executable file (created by linker from *.o and libraries)
- What happens if a symbol (function code) can’t be found by linker?
  - linker can’t create an executable if all dependencies aren’t satisfied
  - use nm to track down missing symbols

Debugging

- ADVICE: learn to use the debugger
- I make lots of errors in coding so I live in the debugger
- But sometimes with parallel programming printing is still a useful option for debugging