

What is this Parallel Computing Thing?

Shelley Knuth

shelley.knuth@colorado.edu

www.rc.colorado.edu

Questions? #RC_BasicSC

Link to survey on this topic: <http://tinyurl.com/rcpresurvey>

Slides:

https://github.com/ResearchComputing/Final_Tutorials/tree/master/Basics_Supercomputing

Outline

- Serial vs. Parallel processing
- Shared vs. Distributed Memory
- OpenMP vs. MPI
- Matlab
- Overhead

What Is Parallelism?

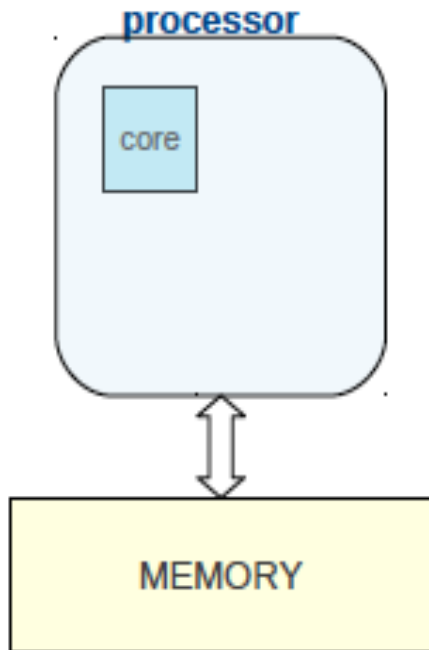
- What is parallelism?
 - Idea where many instructions are carried out simultaneously across a computing system
 - Can divide a large problem up into many smaller problems
 - The idea of splitting up mowing the lawn with your spouse
 - Or of you and your spouse mowing your lawn and your neighbor's lawn
 - Potentially faster, more efficient

Why Parallelize?

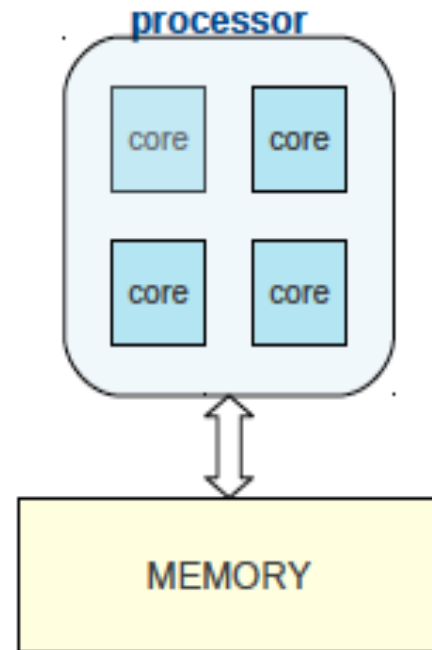
- Single core too slow for solving the problem in a “reasonable” time
 - “Reasonable” time: overnight, over lunch, duration of a PhD thesis
- Memory requirements
 - Larger problem
 - More physics
 - More particles

Basic Architecture

Older processor had only one
cpu core to execute instructions



Modern processors have 4 or more
independent cpu cores to execute instructions



Source: http://people.math.umass.edu/~johnston/PHI_WG_2014/OpenMPSlides_tamu_sc.pdf

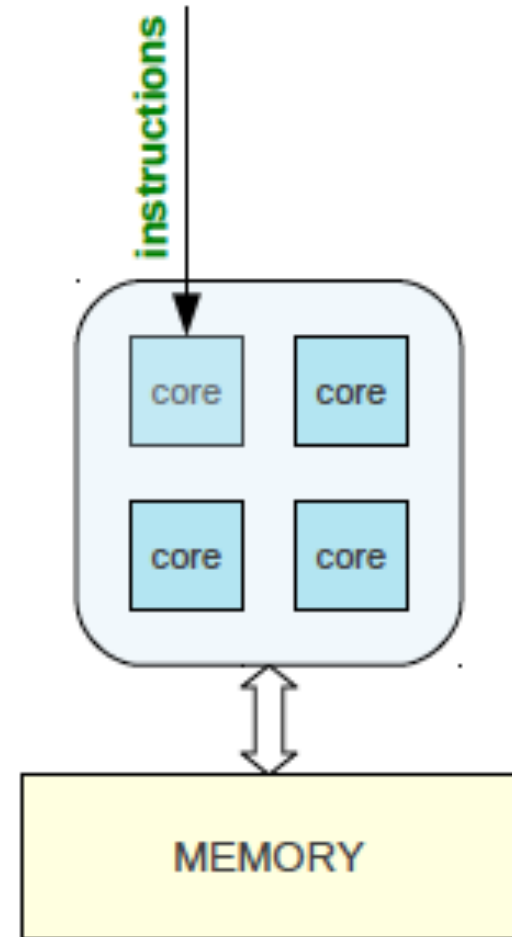
Serial Processing – Thought Experiment

- Jigsaw puzzle analogy**
- Have a 1000 piece jigsaw puzzle
 - You can do it yourself, maybe it will take 1 hour to do
 - Serial processing
- Maybe you have three friends sitting nearby willing to help, but you won't let them
 - Wasted resources

**from Henry Neeman, OSCER, “Supercomputing in Plain English”

Serial Processing

- Instructions are executed on one core
- The other cores sit idle
- If a task is running, Task 2 waits for Task 1 to complete, etc.
- Wasting resources
- Want to instead parallelize and use all cores



Source: http://people.math.umass.edu/~johnston/PHI_WG_2014/OpenMPSlides_tamu_sc.pdf

Shared Memory Parallel Processing – Thought Experiment

- Jigsaw puzzle analogy**
 - Let's say you decide to let one of your friends, Stacey, join you
 - Stacey and you sit at a table and each work on half the puzzle
 - In theory you reduce the puzzle time completion by half
 - However, other time sinks
 - Reaching for the same puzzle pieces
 - Resource contention
 - Communicating about puzzle interfaces
 - Might take 35 minutes instead of 30

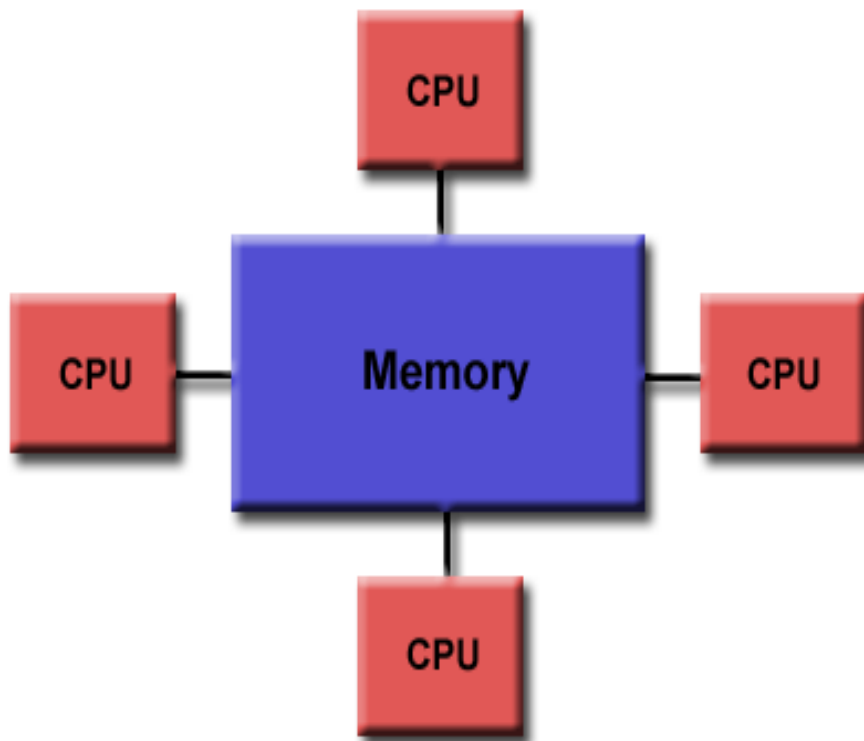
**from Henry Neeman, OSCER, "Supercomputing in Plain English"

Shared Memory Parallel Processing – Thought Experiment

- Jigsaw puzzle analogy**
 - Now you let your other two friends, Fred and Jim, join in
 - Now conceivably could finish in $\frac{1}{4}$ the time (15 minutes)
 - But there's even more contention for resources
 - More communication
 - Slows down the process even more (maybe takes 23 minutes to complete instead)
 - Too many people slows down the process too much to make it worthwhile
 - Eventually have a “diminishing return”

**from Henry Neeman, OSCER, “Supercomputing in Plain English”

Shared-memory Model

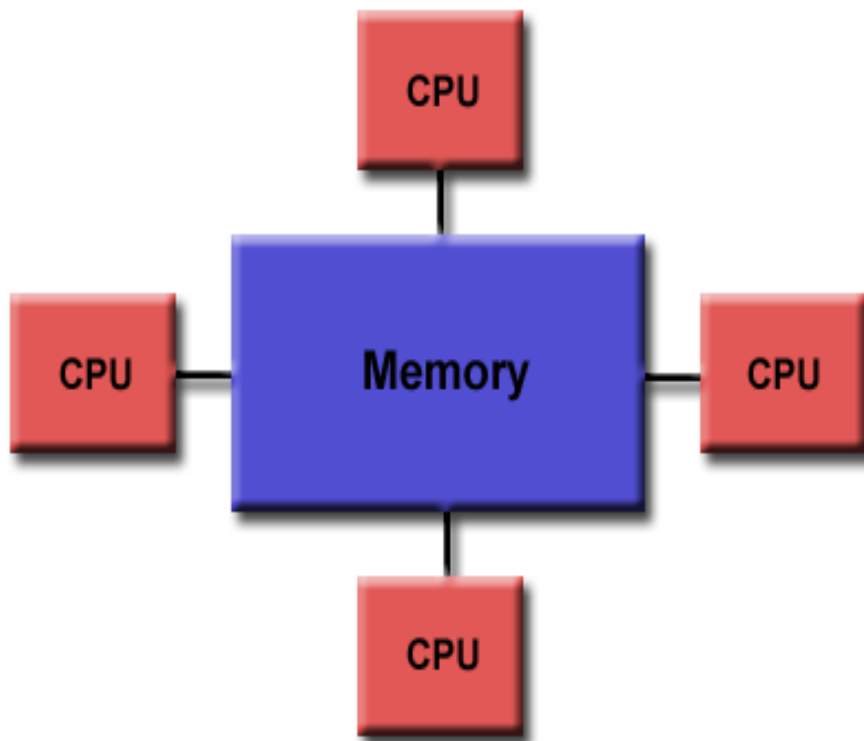


The concept is that all processors can access all memory available

Multiple processors can perform tasks on their own but share the same memory

Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

Shared-memory Model

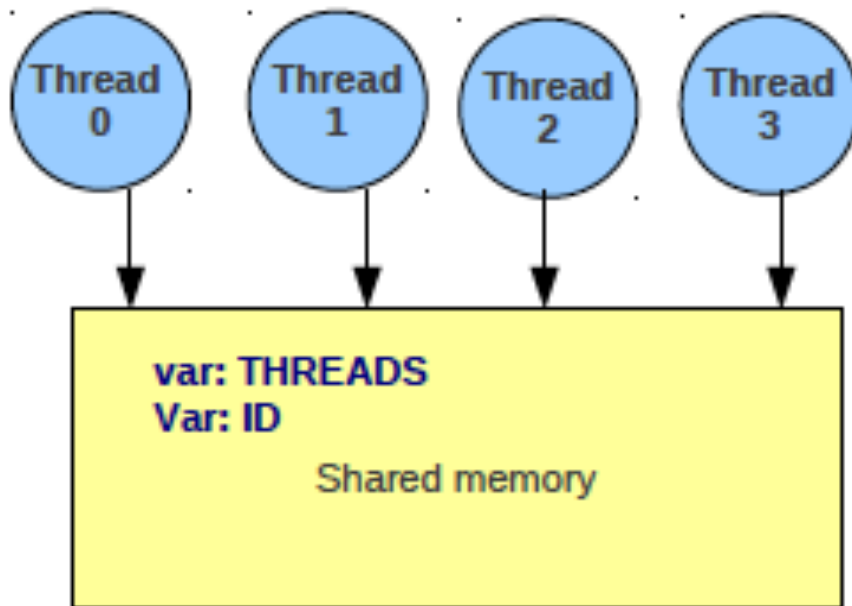


Advantage: data sharing is fast and uniform

Disadvantage: adding more processors can cause performance issues when accessing the same shared memory resource

Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

Shared-memory Model



A thread is a block of code with one entry and one exit that is abstract and is mapped onto a physical core. Multiple threads can be mapped onto one core.

Threads communicate by depositing contents in shared memory area

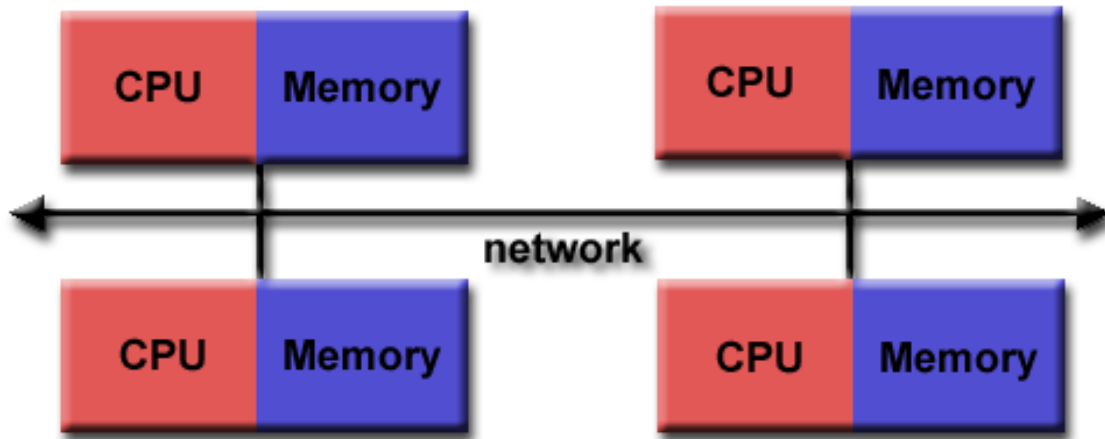
Source: http://people.math.umass.edu/~johnston/PHI_WG_2014/OpenMPSlides_tamu_sc.pdf

Distributed Memory Parallel Processing – Thought Experiment

- Jigsaw puzzle analogy**
 - Now we have two tables with one person at each table doing the puzzle
 - We split the puzzle equally between tables
 - Each person works completely independently
 - But to communicate costs more
 - How do you work out connecting the puzzle?
 - Can you really divide up the puzzle evenly?

**from Henry Neeman, OSCER, “Supercomputing in Plain English”

Distributed-memory Model

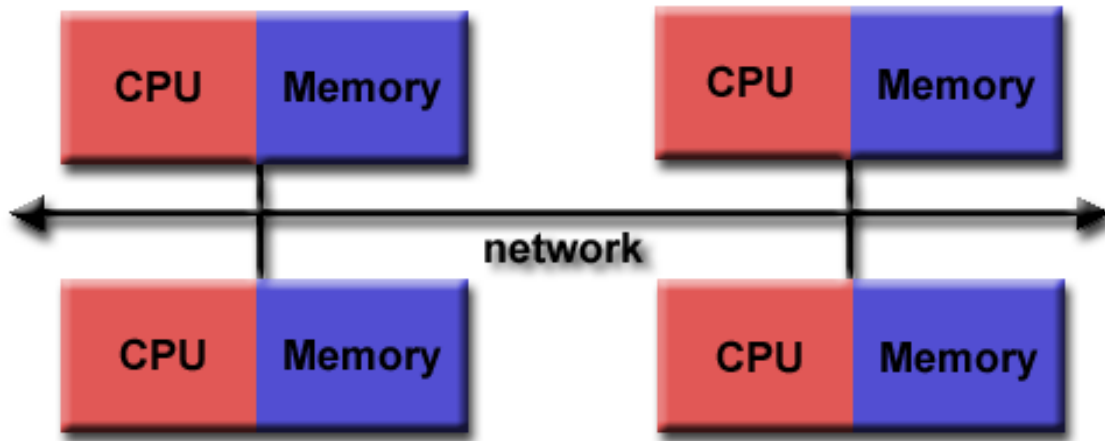


Distributed memory requires a communication network to connect memory

Processors have own memory and don't map globally

Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

Distributed-memory Model



Programmers explicitly define how processors access other processor's memory

Advantage: scalable memory

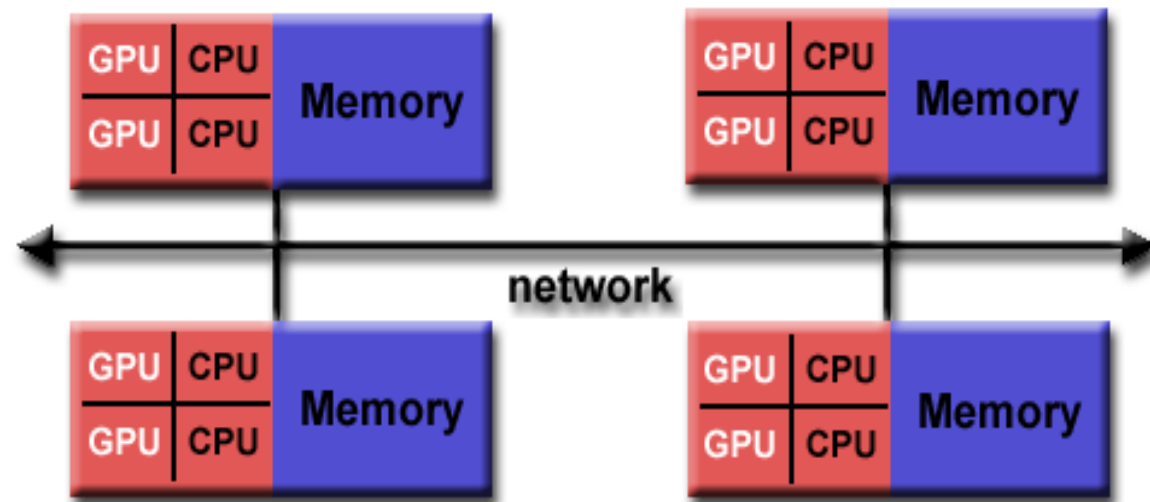
Disadvantage: need to know parallel programming!

Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

Distributed-Shared Memory

Most large and fast computers now

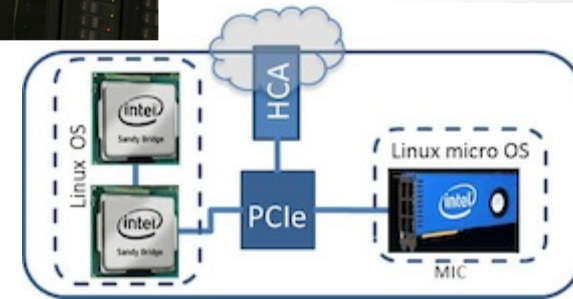
Shared memory machines connected to other shared memory machines



Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

Programming to Use Parallelism

- Parallelism across processors/threads
 - OpenMP
- Parallelism across multiple nodes - MPI



www.scan.co.uk

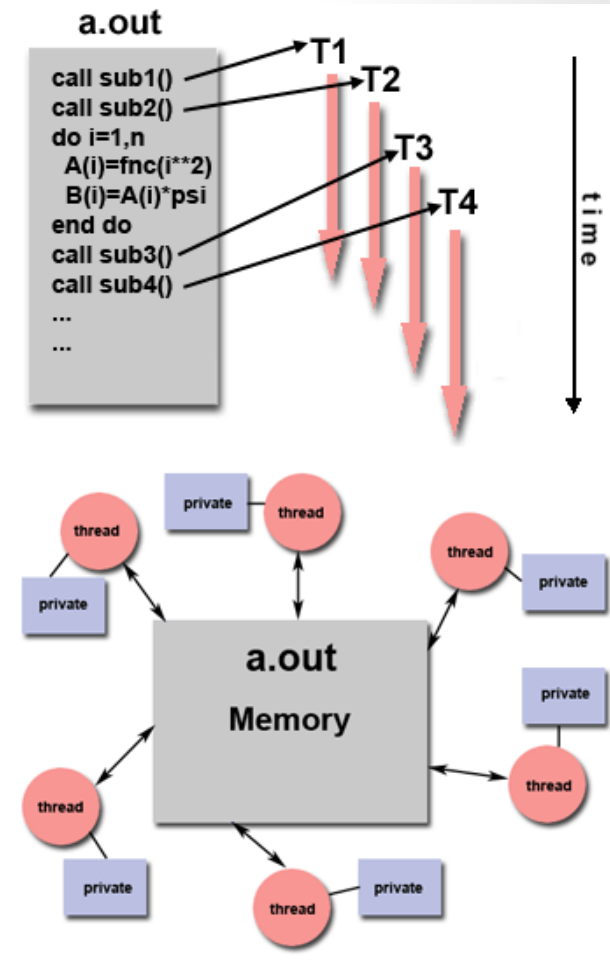
OpenMP

- OpenMP: An application programming interface (API) for parallel programming on multiprocessors
- Uses shared memory
- OpenMP is used through compiler directives embedded in Fortran, C, or C++ code
- Directs multi-threaded, shared memory parallelism
- Can do a lot with only a handful of commands
- Intended to be easy to use

Source: <https://computing.llnl.gov/tutorials/openMP/#Introduction>

Multi-Threaded, Shared Memory Parallelism

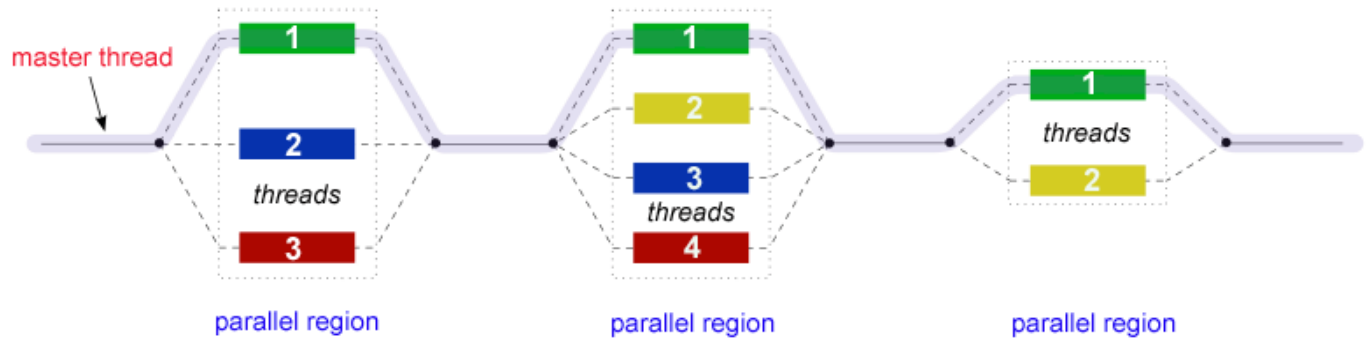
- Have a main program that loads all needed resources
- Main program does many things, including run subroutines
 - Threads that can be run concurrently
 - Share the same resources from the main program, but also has local data
 - Threads communicate through global memory
 - Must ensure multiple threads don't update concurrently
 - Where OpenMP and programmers come in



Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

OpenMP – Fork/Join

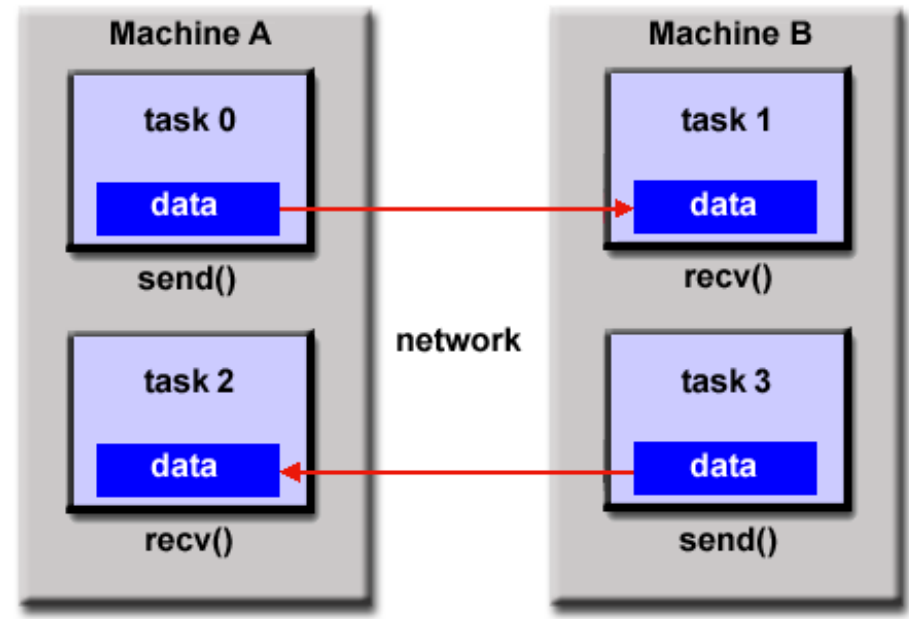
- OpenMP programs start with a single thread (master)
- Then Master creates a team of parallel “worker” threads (FORK)
- Statements in block are executed in parallel by every thread
- At end, all threads synchronize and join master thread



Source: <https://computing.llnl.gov/tutorials/openMP/#Introduction>

Distributed Memory

- Tasks use own local memory when computing
- Can reside on one or many machines
- Communicate by sending and receiving messages
- Have an inter-connect, such as Infiniband



http://www.ee.ryerson.ca/~courses/ee8218/mpi_openmp.pdf

MPI

- MPI is a library specification for message passing
Based on consensus of many organizations
 - Provides widely used standard for writing message passing programs
- Operates on a distributed model
- Exchange data through communication between tasks – send and receive data
- MPI can get complicated
- Programmers must explicitly implement parallelism using MPI constructs

<https://computing.llnl.gov/tutorials/mpi/>

MPI or OpenMP?

- OpenMP
 - Don't understand parallel programming
 - Only need to run on one node
 - Just want to speed up application
 - Program is not complicated
- MPI
 - Multiple nodes
 - Running out of memory and need to use more nodes

Parallel Processing Musts

- Need to be able to break the problem up into parts that can work independently of each other
- Need to also be able to be worked on simultaneously
 - Can't have the results from one CPU depend on another at each time step

Parallel Overhead

- Should you convert your serial code to parallel?
- Usually do it to speed up
- But need to consider things like overhead
- Overhead because of
 - Startup time
 - Synchronizations
 - Communication
 - Overhead by libraries, compilers
 - Termination time

https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

Questions?

- Email rc-help@colorado.edu
- Twitter: CUBoulderRC
- Link to survey on this topic:
<http://tinyurl.com/curc-survey16>
- Slides:
https://github.com/ResearchComputing/Final_Tutorials/tree/master/Basics_Supercomputing