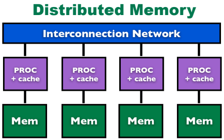
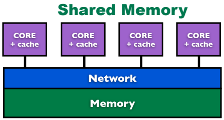
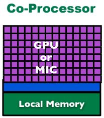
Contents

1. Introduction
   - Quick Overview of Parallel Hardware
   - A Concise Introduction to Parallelism
   - R and Parallelism

Three Basic Flavors of Hardware



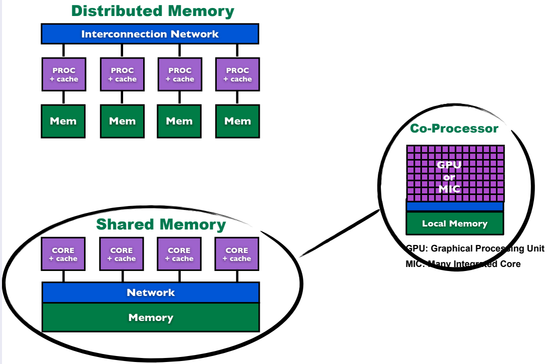




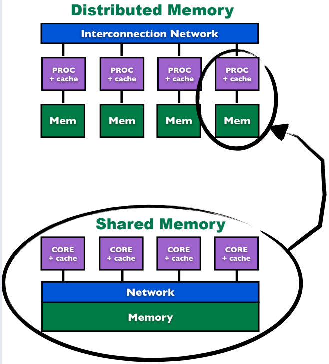
GPU: Graphical Processing Unit

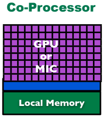
MIC: Many Integrated Core

Your Laptop or Desktop



A Server or Cluster

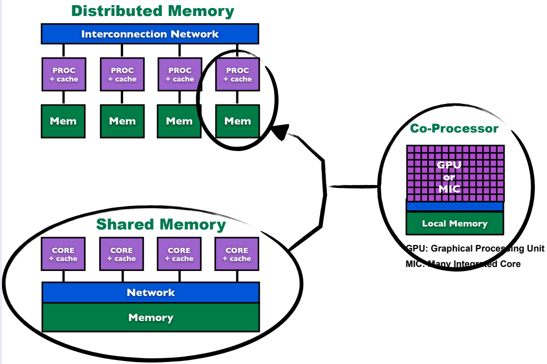




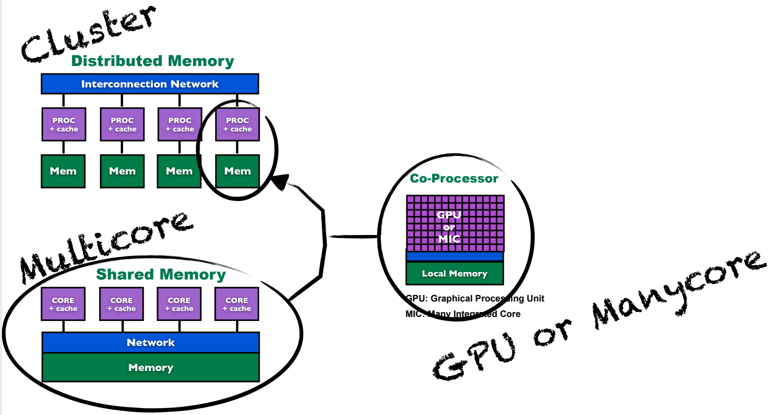
GPU: Graphical Processing Unit

MIC: Many Integrated Core

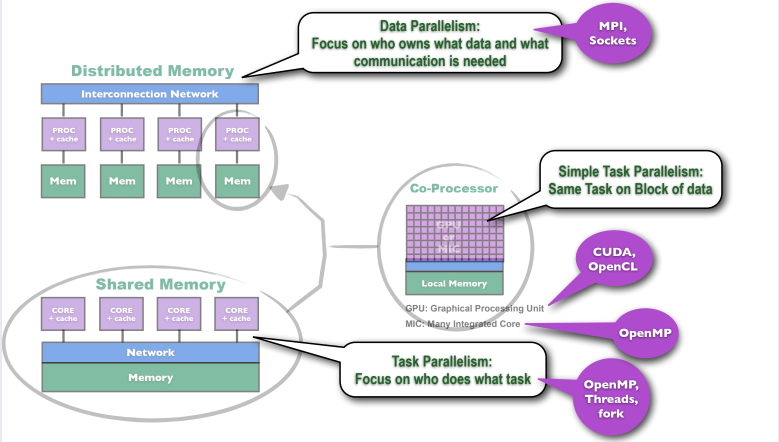
Server to Supercomputer



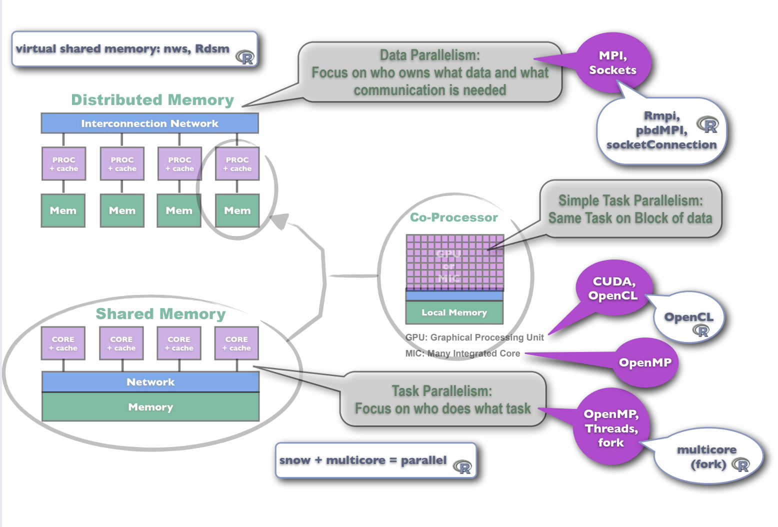
Knowing the Right Words



“Native” Programming Models and Tools



R Interfaces to Native Tools



virtual shared memory: nws, Rdsm

Data Parallelism:
Focus on who owns what data and what communication is needed

MPI, Sockets

Rmpi, pbdMPI, socketConnection

Simple Task Parallelism:
Same Task on Block of data

GPU: Graphical Processing Unit
MIC: Many Integrated Core

CUDA, OpenCL

OpenCL

OpenMP

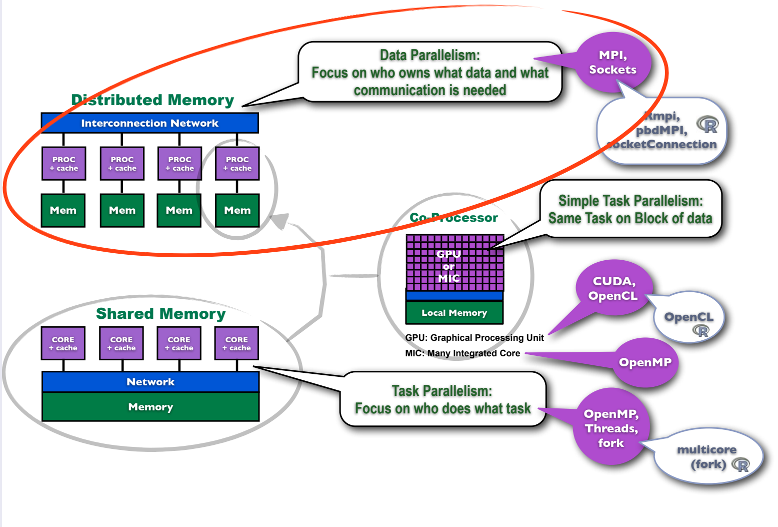
Task Parallelism:
Focus on who does what task

OpenMP, Threads, fork

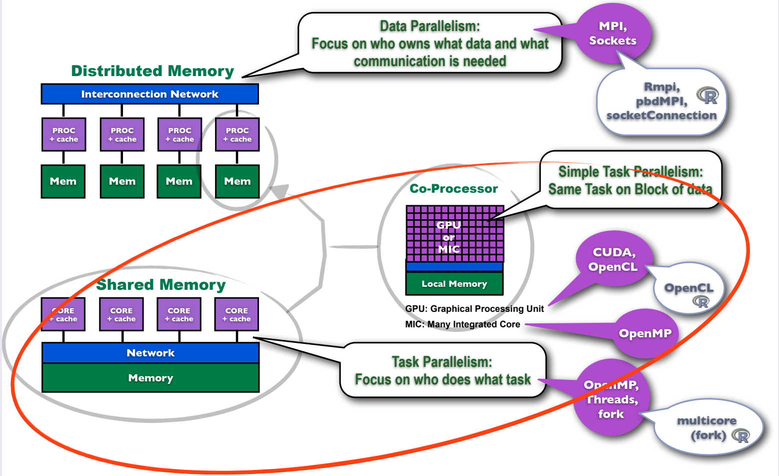
multicore (fork)

snow + multicore = parallel

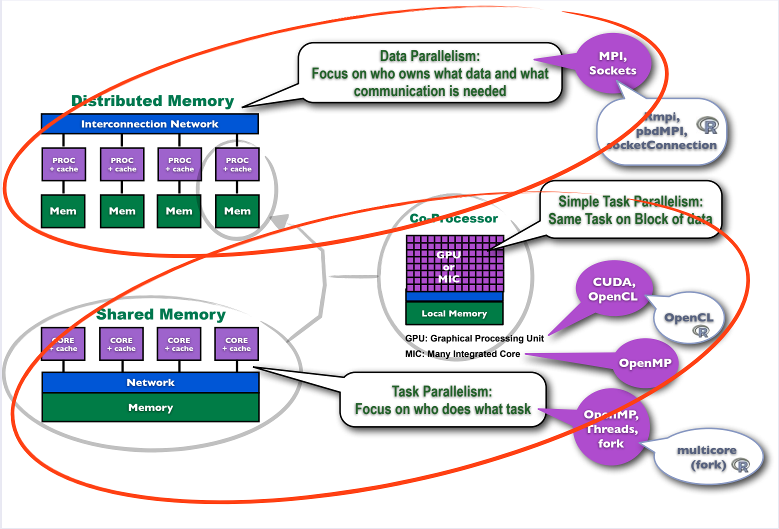
30+ Years of Parallel Computing Research



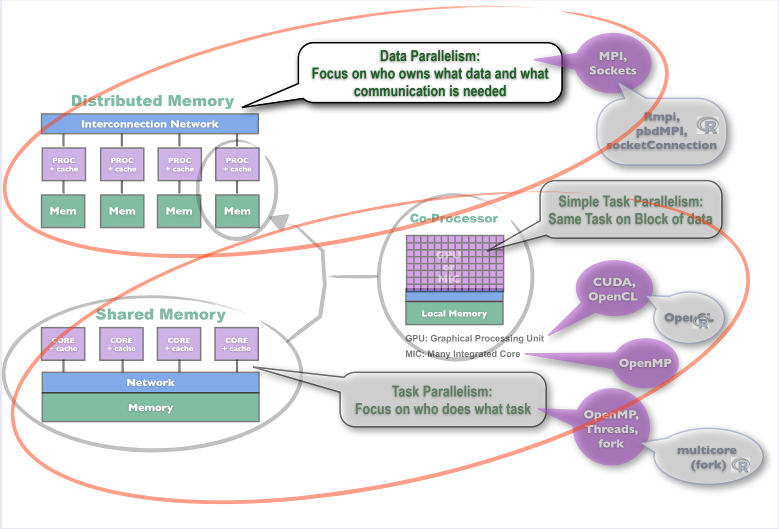
Last 10 years of Advances



Putting It All Together Challenge



pbdR Focus on Data Parallelism



Data Parallelism:
Focus on who owns what data and what communication is needed

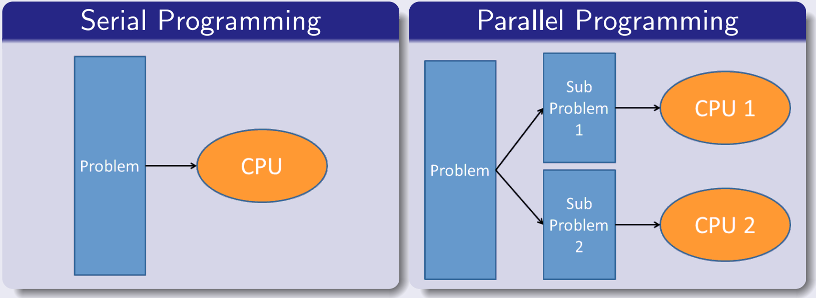
Simple Task Parallelism:
Same Task on Block of data

Task Parallelism:
Focus on who does what task

What is Parallelism?

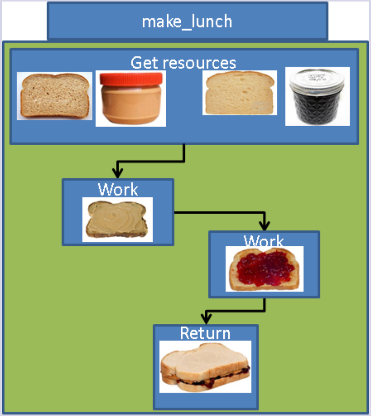
- Doing more than one thing at a time.
- The simultaneous use of multiple compute resources to solve a computational problem.

Parallelism

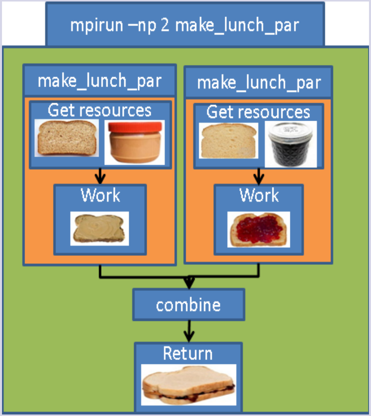


Parallelism

Serial Programming



Parallel Programming



Kinds of Parallelism

- *Data Parallelism*: Data is distributed
- *Task Parallelism*: Tasks are distributed

(This is a gross oversimplification)

pbdR Paradigms: Data Parallelism

Data parallelism:

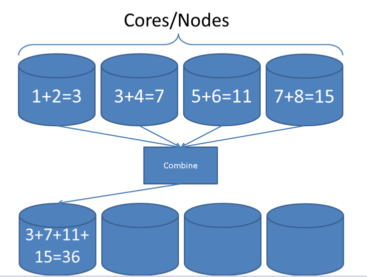
- No one processor/node owns all the data.
- Processors own local pieces of a (conceptually) larger, global object

Task parallelism:

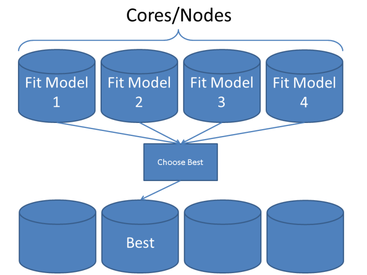
- Often involves different tasks to the same data.

Data vs Task Parallelism

Data Parallelism



Task Parallelism



Parallel Programming Vocabulary: Difficulty in Parallelism

1. *Implicit parallelism*: Parallel details hidden from user
2. *Explicit parallelism*: Some assembly required...
3. *Embarrassingly Parallel*: Also called *loosely coupled*. Obvious how to make parallel; lots of independence in computations.
4. *Tightly Coupled*: Opposite of embarrassingly parallel; lots of dependence in computations.

Speedup

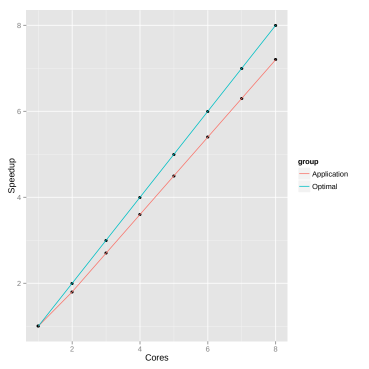
- *Wallclock Time*: Time of the clock on the wall from start to finish
- *Speedup*: unitless measure of improvement; more is better.

$$S_{n_1,n_2} = \frac{\text{Run time for } n_1 \text{ cores}}{\text{Run time for } n_2 \text{ cores}}$$

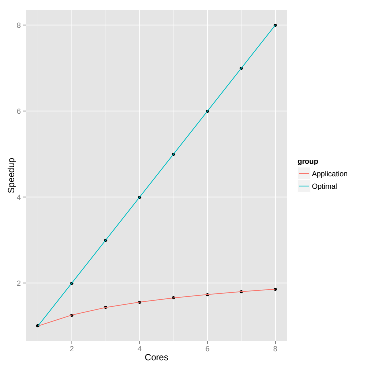
  - n_1 is often taken to be 1
  - In this case, comparing parallel algorithm to serial algorithm

Speedup

Good Speedup



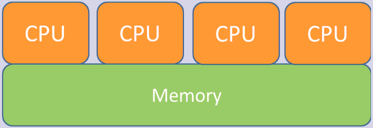
Bad Speedup



Recall: Shared and Distributed Memory Machines

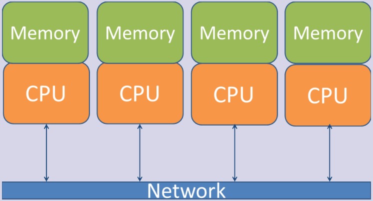
Shared Memory

Direct access to read/change memory (one node)



Distributed

No direct access to read/change memory (many nodes); requires communication



Shared and Distributed Memory Machines

Shared Memory Machines

Thousands of cores

Nautilus, University of Tennessee
1024 cores
4 TB RAM

Distributed Memory Machines

Hundreds of thousands of cores

Kraken, University of Tennessee
112,896 cores
147 TB RAM

R and Parallelism

What about R?

Problems with Serial R

1. Slow.
2. If you don't know what you're doing, it's *really* slow.
3. Performance improvements usually for small machines.
4. Very ram intensive.

Why We Need Parallelism

1. Saves compute time.
2. Data size is skyrocketing.
3. Necessary for many problems.
4. Its necessity is coming.
5. *It's really cool.*

Recall: Parallel R Packages

Shared Memory

1. **foreach**
2. **parallel**
3. **snow**
4. **multicore**

Distributed

1. **Rmpi**
2. R+Hadoop
3. **pbdR**

(and others. . .)

R and Parallelism

The solution to many of R's problems is parallelism. However . . .

What we have

1. Mostly serial.
2. Mostly not distributed
3. Data parallelism mostly explicit

What we want

1. Mostly parallel.
2. Mostly distributed.
3. Mostly implicit.

R and Parallelism

Likewise, the HPC community is looking for high-level languages for data. . .