A Quick Introduction to MPI

Joel C. Adams
Hardware: A Diverse Landscape

- Shared-memory systems
- Distributed-memory systems
- Heterogeneous systems
What is MPI?

• The *Message Passing Interface*

• An industry-standard library for message passing parallel computing in C, C++, Fortran, with 3rd party bindings for Java, Python, R, ...

• Designed by a large consortium:
  – 12 companies: *Cray, IBM, Intel, ...*
  – 11 national labs: *ANL, LANL, LLNL, ORNL, Sandia, ...*
  – representatives from 16 universities

• Useful on shared- or distributed-parallel systems
MPI Software: Multiprocessing

- Software *processes* run on each computer
- MPI lets these processes communicate by *sending-receiving messages* via the network.
- *Single Program Multiple Data (SPMD)* pattern
  - Each process runs the same program, but has different data values (e.g., a different process ID) as it runs
MPI Runtime

• To run an MPI program from the command line:

\[
\text{mpirun} \quad \text{–np} \quad N \quad \text{–hostfile} \quad \text{hostFile} \quad \text{.} / \text{program}
\]

Launch \( N \) processes
(each will get a unique rank)
Vary \( N \) to test scalability

Launch those \( N \) processes
on the computers listed in \text{hostFile}
(optional on many clusters)

Each process runs this same program
(SPMD pattern)
mpirun –np 1 ...

Head Node

Process 0

Compute Node 1

Network Switch

Compute Node 2

Compute Node 3
mpirun -np 4 -mapby node ...

Process 0

Process 1

Process 2

Process 3

Head Node

Compute Node 1

Compute Node 2

Compute Node 3
Parallel Problem Solving

Two common parallel algorithmic strategies:

• **Data decomposition**: process a dataset of size N by dividing the data among the P processes
  – Each process does N/P of the work, in parallel
  – Can scale well for large datasets

• **Task decomposition**: Divide a process into its functional steps (aka *tasks*); perform any independent tasks in parallel
  – Scalability bounded by the number of tasks
Data Decomposition (1 Process)

process 0
Data Decomposition (2 Processes)

Process 0

Process 1
Data Decomposition (4 Processes)

- Process 0
- Process 1
- Process 2
- Process 3
Task Decomposition

The independent steps in a sequential computation can be “parallelized”:

```c
int main() {
    x = f();
    y = g();
    z = h();
    w = x + y + z;
}
```
**Patternlets...**

are minimalist, scalable, and complete programs, each illustrating one or more parallel patterns:

- **Minimalist** to help students understand the pattern by eliminating non-essential details
- **Scalable** so that students can vary the number of processes and see the pattern’s behavior change
- **Complete** for flexible use:
  - Instructors can use them in a ‘live coding’ lecture
  - Students can explore them in a hands-on exercise, and use them as models for their own programs.
Exemplars...

are programs that use parallel patterns to solve a ‘real world’ problem.

Exemplars let students see how a pattern can be useful in a meaningful context.

A patternlet is useful for introducing students to a pattern; an exemplar is useful for helping students see how and why a pattern is relevant.