Slides to Accompany *Programming Languages and Methodologies*

R. J. Schalkoff

Chapter 15, Part 2: MPI

Parallel (Imperative) Programming Language Extensions

Fundamental to any consideration of parallel (declarative) programming languages are three entities:

1. A means to distinguish or delineate a process or code segment and denote the process as PARALLEL, SERIAL or some combination;
2. A means to indicate a block which should be processed in parallel; and
3. A means to allow synchronization or communication between processes.

A Process

For our purposes, a process is an instance of a program running in a computer, and synonymous with the term *task*.

- A process has an associated set of data used to keep track of the process.
- A process can initiate a subprocess, termed a *child* process. The initiating process is referred to as the *parent* process.
- Processes can exchange information or synchronize their operation through several methods of interprocess communication (IPC).
- To allow programming with multiple (concurrent) processes, a technique is needed to spawn multiple parallel processes.

MPI and Beowulf Clusters

- A Beowulf computing cluster is a low-cost, scalable supercomputer using common, off the shelf (COTS) components.
- Most Beowulf PCs currently use a UNIX-like (e.g., linux) OS and thus leverage existing low-cost computing techniques and tools (e.g., linux, GNU tools, MPI, etc.).
- A Beowulf programmer may implement 'conceptual architectures' such as SI(SP)MD, MIMD and tree structured architectures via control of interprocess communications.

MPI

MPI stands for the Message Passing Interface.

- From a programming viewpoint, MPI is a common API for parallel programming.
- There are a number of freely available MPI implementations for heterogeneous networks of workstations and symmetric multiprocessors, based upon both Unix (linux) and Windows NT.
- MPI facilitates processes communication and synchronization using a library of functions.
- MPI allows the overlap of interprocess communication and computation.
- The basic communication primitive in MPI is the transmittal

of data between a pair of processes, i.e. 'point to point communication'.

- The typical application developed using MPI embodies an extension of the SIMD type, denoted the Single Program, Multiple Data (SPMD) computational model. The program is distributed (using `mpirun`) to each the np processes, and is executed on each.

Process Rank: 'Who Am I'?

- Each process is identified by a process rank.
- Process ranks are integers and are returned by a call to a communicator using `MPI_Comm_rank( )`.
- *Knowledge of process rank for each process provides a way for the behavior of the distributed program to be tailored to each process.* In addition, interprocess communication is facilitated.

MPI 'Hello World' Example

```
#include <stdio.h>
#include "mpi.h"

main(int argc, char** argv) {
        int p; /* Rank of process */
        int n; /* Number of processes */
        int source; /* Rank of sender */
        int dest; /* Rank of receiver */
        int tag = 50; /* Tag for messages */
  char message[100]; // storage for message
        MPI_Status status; /* Return status for receive */


        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &p);
        MPI_Comm_size(MPI_COMM_WORLD, &n);

        if (p != 0) {
```

```
            sprintf(message, "Hello World from process %d",
                                    p);
            dest = 0;
            MPI_Send(message, strlen(message)+1, MPI_CHAR, dest,
                                    tag, MPI_COMM_WORLD);
}
        else    {  // p == 0
            for (source = 1; source < n; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, tag,
                                    MPI_COMM_WORLD, &status);
            printf("%s\n", message);
}
}
        MPI_Finalize();
}       /* main */
```

Sample Results: A 16-process version of the program was run:

```
$gcc hello-mpi.c -lmpi -o hello-mpi
$mpirun -np16  hello-mpi
```

with results as shown below:

```
Hello World from process 1
Hello World from process 2
Hello World from process 3
Hello World from process 4
Hello World from process 5
Hello World from process 6
Hello World from process 7
Hello World from process 8
Hello World from process 9
Hello World from process 10
Hello World from process 11
Hello World from process 12
Hello World from process 13
Hello World from process 14
Hello World from process 15
```