An introduction to MPI

Presented by
Jerry Perez

Overview

- What is MPI?
- The basics of parallel computer architectures.
- The Message Passing Model
- MPI examples

What is MPI?

Message Passing Interface, (MPI), is a standard library of subroutines (Fortran) or function calls (C) that can be used to implement a message passing program.

What does MPI do?

MPI allows the coordination of a program running as multiple processes in a distributed memory environment, yet is flexible enough to also be used in a shared memory system.

What types of computers use MPI?

MPI programs can be used and compiled on a wide variety of parallel computers such as the IBM SP2, the Silicon Graphics Origin 2000, or a cluster of workstations such as a Beowulf cluster over a network.

Message Passing Fundamentals

As a programmer, you may find that you need to solve ever larger, more memory intensive problems, or simply solve problems with greater speed than is possible on a serial computer.

You can turn to parallel programming and parallel computers to satisfy these needs. Using parallel programming methods on parallel computers gives you access to greater memory and Central Processing Unit (CPU) resources not available on serial computers. Hence, you are able to solve large problems that may not have been possible otherwise, as well as solve problems more quickly.

Message Passing Fundamentals

One of the basic methods of programming for parallel computing is the use of message passing libraries. These libraries manage transfer of data between instances of a parallel program running (usually) on multiple processors in a parallel computing architecture.

The Message Passing Model

- MPI is intended as a standard implementation of the "message passing" model of parallel computing.
- A parallel computation consists of a number of *processes*, each working on some local data. Each process has purely local variables, and there is no mechanism for any process to *directly* access the memory of another.
- Sharing of data between processes takes place by message passing, that is, by explicitly sending and receiving data between processes.

SPMD

MPI uses the SPMD
(single program multiple data) paradigm.

- The executable is replicated across n nodes.
- Each replication exchanges data with the other. Each replication does computation on the exchange data in a multi threaded loop.

Goals of MPI

The primary goals addressed by MPI are to:

1. Provide source code portability. MPI programs should compile and run as-is on any platform.
2. Allow efficient implementations across a range of architectures.

MPI also offers

1. A great deal of functionality, including a number of different types of communication, special routines for common "collective" operations, and the ability to handle user-defined data types and topologies.
2. Support for heterogeneous parallel architectures.

Basic Features of Message Passing Programs

Message passing programs consist of multiple instances of a serial program that communicate by library calls. These calls may be roughly divided into four classes:

1. Calls used to initialize, manage, and finally terminate communications.
2. Calls used to communicate between pairs of processors.
3. Calls that perform communications operations among groups of processors.
4. Calls used to create arbitrary data types.

A First Program: Hello World!

C

```
#include <stdio.h>
#include <mpi.h>
void main (int argc, char
   *argv[ ]) {
int err;
err = MPI_Init (&argc, &argv);
printf ("Hello world!\n");
err = MPI_Finalize();
}
```

Fortran

```
PROGRAM hello
INCLUDE 'mpif.h'
INTEGER err
CALL MPI_INIT(err)
PRINT *, "Hello world!“
CALL MPI_FINALIZE(err)
END
```

Where to go for more info

http://webct.ncsa.uiuc.edu:8900/public/MPI/
Register for a free online MPI class.

http://www.netlib.org/utk/papers/mpi-book/node1.html
THE Complete MPI reference.

Or read Parallel Programming with MPI by Peter S. Pacheco: ISBN#1-55860-339-5