Introduction to MPI

EAS 520
High Performance Scientific Computing

University of Massachusetts Dartmouth

Spring 2014

References

This presentation is almost an exact copy of Dartmouth College's Introduction to MPI tutorial. The link can be found in:

`http://www.dartmouth.edu/~rc/classes/intro_mpi/`

Changes from the original document are related to compilers and job submissions for UMass Dartmouth clusters.

Advantages of Parallel Programming

- Need to solve larger problems
  - more memory intensive
  - more computation
  - more data intensive

- Parallel programming provides
  - more CPU resources
  - more memory resources
  - solve problems that were not possible with serial program
  - solve problems more quickly

Parallel Computer Architectures

Two Basic Architectures

- Distributed Memory (ex. Compute cluster)
    - collection of nodes which have multiple cores
    - each node uses its own local memory
    - work together to solve a problem
    - communicate between nodes and cores via messages
    - nodes are networked together
- Shared Memory Computer
    - multiple cores
    - share a global memory space
    - cores can efficiently exchange/share data

Parallel Programming Models

- Directives-based parallel programming language
  - OpenMP (most widely used)
  - High Performance Fortran (HPF)
  - directives tell processor how to distribute data and work across the processors
  - directives appear as comments in the serial code
  - implemented on shared memory architectures

- Message Passing (MPI)
  - pass messages to send/receive data between processes
  - each process has its own local variables
  - can be used on either shared or distributed memory architectures
  - outgrowth of PVM software

Pros and Cons of MPI

- Pros of MPI
  - runs on either shared or distributed memory architectures
  - can be used on a wider range of problems than OpenMP
  - each process has its own local variables
  - distributed memory computers are less expensive than large shared memory computers
- Cons of MPI
  - requires more programming changes to go from serial to parallel version
  - can be harder to debug
  - performance is limited by the communication network between the nodes

Pros and Cons of OpenMP

- Pros of OpenMP
  - easier to program and debug than MPI
  - directives can be added incrementally - gradual parallelization
  - can still run the program as a serial code
  - serial code statements usually don't need modification
  - code is easier to understand and maybe more easily maintained
- Cons of OpenMP
  - can only be run in shared memory computers
  - requires a compiler that supports OpenMP
  - mostly used for loop parallelization

Parallel Programming Issues

- Goal is to reduce execution time
  - computation time
  - idle time - waiting for data from other processors
  - communication time - time the processors take to send and receive messages
- Load Balancing
  - divide the work equally among the available processors
- Minimizing Communication
  - reduce the number of messages passed
  - reduce amount of data passed in messages
- Where possible - overlap communication and computation
- Many problems scale well to only a limited number of processors

Problem Decomposition

Two kinds of decompositions:

- Domain decomposition
    - data divided into pieces of same size and mapped to different processors
    - processor works only on data assigned to it
    - communicates with other processors when necessary
    - examples of domain (data) decomposition
        - embarrassingly parallel applications (Monte Carlo simulations)
        - finite difference calculations
        - numerical integration
- Functional decomposition
    - used when pieces of data require different processing times
    - performance limited by the slowest process
    - program decomposed into a number of small tasks
    - tasks assigned to processors as they become available
    - implemented in a master/slave paradigm
    - examples of functional decomposition
        - surface reconstruction from a finite element mesh
        - searching images or data bases

What is MPI ?

- MPI stands for Message Passing Interface
- library of functions (C/C++) or subroutines (Fortran)
- History
    - Early message passing Argonne's P4 and Oak Ridge PVM in 1980s
    - MPI-1 completed in May 1994
    - MPI-2 completed in 1998
        - parallel I/O
        - C++/F90 bindings
        - dynamic process management
    - full MPI-2 implementations only recently
- MPI-2 features gradually added to MPI implementations

Differences between versions of MPI

- Examples of Different Implementations
  - MPICH - developed by Argonne Nationa Labs (freeware)
  - MPI/LAM - developed by Indiana, OSC, Notre Dame (freeware)
  - MPI/Pro - commerical product
  - Apple's X Grid
  - **OpenMPI** - MPI-2 compliant, thread safe
- Similiarities in Various Implementations
  - source code compatibility (except parallel I/O)
  - programs should compile and run as is
  - support for heterogeneous parallel architectures
    - clusters, groups of workstations, SMP computers, grids
- Difference in Various Implementations
  - commands for compiling and linking
  - how to launch an MPI program
  - parallel I/O (from MPI-2)
  - debugging
- Programming Approaches
  - SPMD - Single Program Multiple Data (same program on all processors)
  - MPMD- Multiple Program Multiple Data (different programs on different processors)