Concurrent Programming

CS105 Programming Languages
Supplement:
MPI: Message Passing Interface
Outline

- Domain and Functional decomposition
- Data-parallel vs. Message Passing
- Parallel programming issues
- Message Passing model
- MPI
- Hello World
- Communication
- Compiling and running MPI
Problem Decomposition

- First step in parallel programming: decompose the problem
  - Domain decomposition: factor the data
  - Functional decomposition: ...
Domain Decomposition

- *Data parallelism*, data divided into units of similar size and mapped to processors
- Advantage: single flow of control; simple structure
- Example: finite differencing algorithm
Functional Decomposition

- Data parallelism undesirable when pieces of data require different amounts of computation
- *Task parallelism* divides problem into tasks which are assigned to processors as they are idle
- Employs client-server model
Parallel Programming Issues

- Load balancing
- Minimizing communication
  - Computation time,
  - Idle time
  - Communication time (latency and bandwidth)
- Overlapping communication and computation
  - Non-blocking communication interleaved with computation
  - Difficult to achieve but beneficial
Message Passing Model

- Parallel program consists of processes with purely local data
  - No direct access to data of another process
- Sharing data between processes explicit sending and receiving
- The model is extremely general
  - Covers any type of parallel computation
  - Implemented on variety of hardware platforms
  - Superior performance (from greater control of data and control flow)
What is MPI

- **Message Passing Interface (MPI)**
  - Library of functions used in C or Fortran to manage and communicate between processes
- Developed by the MPI Forum
- Implementations of standard available on many platforms
Why Use MPI

- Primary goals:
  - Source code portability
  - Efficient implementations across platforms
- Not part of MPI standard
  - Launching/killing programs (platform-and implementation dependent)
  - Dynamic process management (changing number of processors)
  - Parallel IO
  - Debugging
MPI Features

- Processes communicate via four categories of library calls:
  - Initialize, manage, terminate communication
  - Pairwise communication
  - Groupwise communication
  - Creation of arbitrary data types
#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();

}

First MPI Program
Communication

- Point-to-point communications
  - Explicit send and receive required
- Blocking and non-blocking (returns upon completion or immediately)
- Collective communication
  - Broadcast
  - Gather and scatter
  - Reduction operations
Compiling and Running MPI Program

- Compile program (with mpi library)
  - Mpicc -lmpi sourcefile
- Run program on $n$ processors using wrapper
  - Mpirun -np $n$ compiledprogram