Introduction to Parallel Programming with MPI

by

Prasad Maddumage
Research Computing Center
Florida State University
October 31, 2019
Outline

• Part 1
  – Basics of parallel programming
  – Basics of MPI
  – Point to point communication
  – Blocking vs. non-blocking calls
  – Collective communication

• Part 2
  – Parallel programming strategies
Parallel Programming Strategies

- Client-server
- Data Parallelism
- Task Parallelism
- Pipeline
Client-Server

- Server (master) decides what clients (slaves) do
- **Embarrassingly parallel**: The problem can easily be broken into roughly equal amounts of work per process and has very little communication (low communication overhead)
- Has near linear speedup and easy to program
- Eg: Monte Carlo methods - widely used to simulate a physical phenomena or calculate an integral
  - Randomly generate large number of samples (**realizations**) of a phenomenon/equation and take the average over all samples
  - Simulation stops when the average value converges
Client-Server example
Calculating Pi


$\text{cp -a /gpfs/home/prasad/temp/MPIworkshop}$
Data Parallelism

- Each process does **exactly same operations** on a **unique subset of data**
- Problems involve calculus: solving differential equations etc.
  - Numerically solving these equations over a large domain is very common
  - Data parallelism can be applied to parallelize this type of problems
  - Eg: CFD, Heat transfer, Weather prediction, etc.
Task Parallelism

- Each process does **different operations** on **exactly same set of data**
- Task parallelism is a widely used technique
- **N body problem**
  - N objects interacting with each other via forces: stars under gravity, molecules under electrostatic force etc.
  - Send properties of each object to all processes and let each process find the total force on a subset of particles
  - After each time step, use MPI_Allreduce
  - Applications: Cosmology, structural biology, machine learning
Task Parallelism: N body Problem

- Each particle exerts a force on each other
- N particles → N(N – 1)/2 forces to evaluate → O(N^2)
- Brute force method
  - Find all the forces on each particle and find new position and velocity from the total force
  - Send all position data to all processes and let each find the force on a subset of particles
  - Do an all to all reduction, MPI_ALLGATHER, at the end of each time step
Pipeline Parallelism

- Each process does its work, passes its set of data to next process and receives next set of data from previous process.
- All processes are connected to form a data pipeline.
- Every process executes the same tasks and results are passed to the “next” process and more data is received from the “previous” process.
- Workers can be connected in a circular (closed) loop or linear (open).
- Matrix multiplication can be done in a pipeline.
Pipeline: N-Body Simulation

Each worker is assigned a block of bodies

Each worker finds forces on its bodies

Each worker finds velocities of its bodies

This continues for a given time step