2017 年春季 – 并行与分布式计算导论

ASSIGNMENT 2

Assigned: 05/04/2017, Due: 05/19/2017

Instructor: Guojie Luo (gluo@pku.edu.cn)

Problem 1 – MPI_Send & MPI_Recv

The objective of this exercise is to investigate the amount of time required for message passing between two processes, i.e. an MPI communication timing test is performed.

In this exercise different size messages are sent back and forth between two processes a number of times. Timings are made for each message before it is sent and after it has been received. The difference is computed to obtain the actual communication time. Finally, the average communication time and the bandwidth are calculated and output to the screen.

For example, one can run this code on two nodes (one process on each node) passing messages of length 1, 100, 10,000, and 1,000,000 and record the results in a table like the one below.

Length	Communication Time (μSec)	Communication Bandwidth (Megabit/Sec)
1	0.000001	73.403187
100	0.000002	3240.799903
10,000	0.000046	13947.887101
1,000,000	0.003633	17616.862072

Problem 2 – Matrix-Vector Multiplication by MPI

Task: Parallelize Matrix-Vector Multiplication by MPI using the checkerboard block decomposition, and submit the parallelized source code and the report.

Instructions: Reuse the gen.c provided last time to generate random matrices and vectors, or write your own generator.

Problem 3 – Iso-efficiency and Scalability

Assume that a problem of size n requires $O(n^2)$ memory usage and $O(n^3)$ computation in sequential implementation. We have two MPI implementations using p processing elements. The first implementation has the computational complexity of $O(n^3/p)$ and communication complexity of $O(n^2)$. The second implementation has the computational complexity of $O(n^3/p)$ and communication complexity of $O(pn\sqrt{n})$. And we assume the communication do not overlap with the computation in the MPI implementations.

1. Please compute the iso-efficiency relations for both implementations;
2. Please compute the scalability function of both implementations;
3. Which implementation is more scalable?