CS 418, Spring 2008
Assignment 3: Parallel Programming Using the Message Passing Model

Assigned: Tuesday, Feb. 26
Due: Thursday., Mar. 6, noon

This assignment deals with parallel programming using the message passing model. The aim of this assignment is to understand the trade-offs involved in this model, and compare them with what we have learned about the shared memory model for writing parallel programs.

1 Policy and Logistics

Please work in groups of 2 people to solve the problems for this assignment. (Hand in one assignment per group.) There will be both electronic and hard copy hand-ins, as described below. Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory. In the following, *HOMEDIR* refers to the directory:

`/afs/cs.cmu.edu/academic/class/15418-s08/public`

and *ASSTDIR* refers to the subdirectory *HOMEDIR*`/asst/asst3`.

2 Written Assignments

Please answer problem 2.7 (parts c and f only) in the textbook.

3 Programming Assignment: WSP with Message Passing

Now that you have solved the Wandering Salesman Problem using the shared-memory programming model, please solve the same problem using the message-passing programming model. In particular, write your code using the Message Passing Interface (MPI). (You can also use MPICH if you prefer, which is basically the same thing.)

Your report should include the following items:

1. A brief (approximately one page) description of how your program works. Describe the general program flow and all significant data structures. Compare and contrast with the design you used for the shared-memory version.
2. The solution to the problem given in `ASSTDIR/input/distances`. This file contains a 16 city problem.
3. Execution time and speedup (both total and computation) for 1, 2, 4, 8, 16, 24, and 32 processors on the NCSA `cobalt` machine, and 1, 2, 4, and 8 processors on the PSC `rachel` machine. (If you can get even more processors, that is great.)
4. Discuss the results you expected and explain the reasons for any non-ideal behavior you observe. In particular, if you don't get perfect speedup, explain why.

5. Compare the performance of your message-passing version of WSP on `cobalt` with your shared-memory version of the same application on the same machine. (Include graphs to illustrate the difference in performance.) Discuss any differences in performance, why you think the two versions behave differently, etc.

6. Discuss any interesting differences in performance that you observe between running your message-passing code on `cobalt` and running it on `rachel`. Why does the same code perform differently (if it does)? Can you achieve good performance on both machines using the same source code?

If the execution time for your program takes more than a few minutes on `cobalt`, double-check your program and algorithm. Once again, make sure your programs run on a uniprocessor before trying to run them in batch. Also, debug your programs using smaller numbers of cities (perhaps `ASSTDIR/input/dist4`) and small numbers of processors before trying larger runs.

4 Using MPI

A small introduction and tutorial is being handed out with this assignment; it can also be obtained from `ASSTDIR/mpi_tutorial.pdf`. This tutorial gives only a very rudimentary overview to all the things that MPI allows you to do. To learn more about MPI, please consult one or both of the following web sites:

`http://webct.ncsa.uiuc.edu:8900/public/MPI/`
This NCSA online course entitled "Introduction to MPI" is free. Register online at the above URL, with any easy-to-remember login and password (these need not be the same as your login and password for the NCSA machines). Chapters 2-8 cover everything you will need for this assignment. You can log into your account any number of times, so the course material can also be used as an online reference.

`https://computing.llnl.gov/tutorials/mpi/`
This is another good online tutorial for MPI.

5 Hand In

Electronic submission:

Your solution to the WSP. Do this by naming your file *last*-`wsp_mp.c`, where *last* is the last name of one of your group members, and copying this file to the directory

`/afs/cs.cmu.edu/academic/class/15418-s08/public/asst/asst3/handin`

Include as comments near the beginning of this file the identities of all members of your group. Also remember to put comments in your code.

Hard-copy submission:

1. Answers to the questions in Section 2.
2. Answers to the questions in Section 3.
3. A listing of your code.