CS 418, Spring 2011

Assignment 3: Parallel Branch-and-Bound for the Wandering Salesman Problem, Using the Message Passing Model

Assigned: Tuesday, Feb. 22
Due: Thursday, Mar. 3, 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 the *"Assignments and exam information"* web page in the class WWW directory. In the following, *HOMEDIR* refers to the directory:

`/afs/cs.cmu.edu/academic/class/15418-s11/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, solve the same problem using the message-passing programming model. In particular, write your program using the Message Passing Interface (MPI). (You can also use MPICH if you prefer, which is basically the same thing.)

NOTE: The salesman can start at any city. It is not necessary that he has to start at city 1.

Your report should include the following items:

1. A brief (roughly one or two pages) 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 17 city problem. (For debugging purposes, you may want to use some of the smaller input files included in the same directory. The city locations corresponding the distance files are provided in the 'city' files.)
3. Execution time and speedup (both total and computation) for 1, 2, 4, 8, 16, 24, and 32 processors on both `pople` and `blacklight`. (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 `blacklight` with your shared-memory version of the application on the same machine. (Include graphs to illustrate the difference in performance.) Discuss any differences in performance with possible reasons for such a behavior.

6. Compare your results on the two machines. If you see different behaviors on `pople` versus `blacklight`, please discuss what you think is the likely cause of the different behaviors. 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, double-check your program and algorithm. Make sure your programs run on a uniprocessor before trying to run them in parallel. Also, debug your programs using smaller numbers of cities (perhaps `HOMEDIR/asst/asst3/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 of the following websites:

`http://www.citutor.org`
: This NCSA online course entitled "Introduction to MPI" is free. Register online at the above URL, with any easy-to-remember login and password. 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/`
: Another tutorial on MPI. (If you take the NCSA course, this one is optional.)

5 Hand-in

Electronic submission:

Your solution to the WSP. Do this by naming your file *last*`-wsp.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-s11/public/asst/asst3/handin`

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

Hard-copy submission:

1. Answers to the questions in Section 2.

2. Answers to the questions listed in Section 3.

3. A listing of your code.