CIS 5930-04 Parallel Computing
Spring 2007

Assignment 1

Due: 20 Feb 2007

Problem: In this assignment, you will parallelize a program to compute the value of π using Simpson's Rule (exercise 4.12 in the text) in a few different ways, and evaluate the performance of you implementations.

Parallelization: Create parallel versions of the above program as given below.

1. Using MPI (using MPI_Reduce, in particular).
2. Using OpenMP.
3. With hybrid MPI-OpenMP.
4. Using MPI, but with your own implementation of MPI_Reduce (which should work for the operations needed in this example). This implementation should use MPI Send and Recv to collect the data on the root, and have the root perform the arithmetic operations.
5. Using MPI, but with your own implementation of MPI_Reduce (which should work for the operations needed in this example). This implementation should use MPI Send and Recv with the tree-structured communication discussed in class.

Files: You should create a directory called *hw1* on the p690 at NCSA. This directory should contain the following files:

1. *mpi-simpson.c*, implementing the code for #1 above.
2. *omp-simpson.c*, implementing code for #2 above.
3. *mpiomp-simpson.c*, implementing code for #3 above.
4. *reduce2.c*, implementing MPI_Reduce for #4 above. This will be compiled with mpi-simpson.c to create the executable for #4 above.
5. *reduce3.c*, implementing MPI_Reduce for #5 above. This will be compiled with mpi-simpson.c to create the executable for #5 above.
6. *Makefile*: The makefile for this assignment. Typing make should create five executables, *simpson1*, *simpson2*, *simpson3*, *simpson4*, and *simpson5*, corresponding to each of the above parallelization techniques. Each executable will take an integer argument, which specifies the number of points (*n* in fig 4.9) at which to perform the evaluation. It outputs the estimate of π.

Performance evaluation: Evaluate the performance of each implementation as a function of the number of evaluation points and number of threads/processors, and compare them. Your result should include speedup and efficiency curves, conclusions about the relative performance of different methods, and an explanation for the observed differences. If the user wishes to perform this computation with *n* evaluation points, and has at most *P* processors

available, then you should also provide a rule of the thumb for choosing the best method and optimal number of processors.

Turning in your assignment: All your C code files should have a time stamp before midnight, 20 Feb 2007. Please have read and execute permissions unset until at least midnight 20 Feb 2007. Read permission should be set on by noon 21 Feb 2007. Please turn in a hardcopy of your performance result by noon 21 Feb 2007, by either placing it in my mailbox or slipping it under my office door.

Grading: You will be graded on the following (i) correctness of your codes, (ii) performance of your codes, especially relative to those of others in the class, and (iii) the quality of your performance evaluation document (both, presentation, and technical quality).