Homework 3 - Simple MPI Programs

Objective

Write two simple MPI programs

Run on Janus

Assignment

1. Programming assignment 3.7.1 from Pacheco's PPMPI textbook: Ring "Hello World" (p. 52) - 5 points

Write a program in which process *i* sends a greeting to process (*i* + 1) % *np*. The highest process should wrap around and send a greeting back to process zero.

You may get the program running on any computer -- such as the one you configured with MPI last week. Please run the program for several processor counts, including *np*=1, 2, and 32.

Answer the questions from the textbook:

1. Should the program send first and then receive, or receive and then send? Does it matter?
2. What happens when the program is run on one processor? (If it breaks, fix it!)

2. Programming assignment 4.7.2 from Pacheco's PPMPI textbook: Simpson's Rule (p. 64) - 10 points

Write a serial program and then a parallel program to integrate an arbitrary function using Simpson's rule.

- The integration interval should be passed as a command line argument. For example,

  ```
  ./simpsons -lb=-3.0 -ub=10.5
  ```

  is the format for specifying the lower and upper bounds.

- Your programs should support a command line argument like -n, fortran namelist, or declaration that specifies the number of intervals. You may introduce restrictions such that the number of intervals divides into the number of processors conveniently, but your program should handle error conditions gracefully.

- Your programs should support a command line argument like -verbose, fortran namelist, or declaration that makes the program print the intervals being used by each processor. For example, the program could print something like [1 4 2 4] [2 4 2 4 2] [4 2 4 1], or a list of x values and coefficients. *Your assignment of coefficients may be different depending on your distribution scheme, but make sure that you're dividing the intervals properly so that you're parallelizing a single instance of Simpson's rule, not just running* *multiple versions of Simpson's rule. In particular, no x value should evaluated more than once.*

- To test your Simpsons integration you will create the following three functions (F1, F2, F3). You'll need to add in a command line parameter –fx to specify which function you wish to test. NOTE: Let the value of π be restricted to 3.1415926

  (e.g. ./simpsons –fx=F1 –lb=-3.1415926 –ub=3.1415926)

  Note feel free to add additional functions for your own testing but you only be tested and graded for functions F1-F3.

  F1 - Create a function named F1 which solves for the following equation:

  $$F1(x) = \int_{-\pi}^{\pi} \cos(x^2) + sin(x^3)dx$$

  F2 - Fresnel integrals may be used to calculate the intensity of light near a sharp straight edge as partially defined by;

  $$F2(x) = \int_{o}^{b} \cos\left(\frac{\pi x^2}{2}\right) dx$$

  Create a function named F2 which solves for $C(x)$ where b = 5.

  F3 – Use the following elliptical integral to determine the period of a simple pendulum.

  $$F3(\theta) = \int_{a}^{b} \frac{d\theta}{\sqrt{1 - x^2 sin^2\theta}}$$

  Let *x* = 0.5. Use the following intervals for (a,b);

  (-π,0) , (0,π),   (-π, π)

- *Summary: the command line arguments you should have are: -lb, -ub, -n, and -verbose (optional). For example:*

  ```
  ./simpsons -lb=-3.0 -ub=10.5 -n=10 –fx=F1, or
  ./simpsons –fx=F3 -lb=-3.0 -ub=10.5 -n=10 -verbose
  ```

Thoroughly debug and test your program:

- To demonstrate that the program is generating coefficients correctly, run it using a low processor count and low interval count and submit a copy of the interval generation using the -verbose option.
- For larger quantities of intervals, compare the results of the serial and parallel implementations to a known function.

Assignment Submission

Report and Questions

For both programs, write a brief comment (just a paragraph) describing the program, how you tested it, why you believe the implementation is correct, and answers to any questions. Append annotated output that clearly shows that your implementation is correct.

Then, generate a formatted code print-out using enscript (or something similar). For example:

```
enscript -G2r --highlight -o file.ps file.c  eps2pdf ...
```

Print out your write-up and code and bring them to lab on the due date.

Online Deliverables

Create a tar file "username-hw3.tar.gz" file of your homework directory, including all code files and makefiles, and submit it using the d2l by the due date. Use your username in the filename and do not include any executable or object files. Create the .tar.gz file so that it unpacks from the directory without making subdirectories. For example:

```
cd ~/hpsc/hw01 tar cvzf ../username-hw01.tar.gz .
```

Submit your assignment to the d2l before the beginning of the next lab session.