SI 486L Lab 9: Another Slice of Pi

This lab is due before the next lab period. Source files should be zipped or tar'd into a single file and e-mailed (subject should contain the phrase "lab 9") as an attachment to the instructor. One of the files should be a makefile with a default target that builds an executable called "piomp" . A README file is always a welcome addition. ***Hand in a hard copy of this sheet with recorded values and answers.***

Introduction

Remember our first MPI program? It used Monte-Carlo techniques to approximate pi. We're going to revisit that program (`/courses/albing/486L/pi2.c`), make some timings, and convert it to use OpenMP.

Task #1

Your objective is to take the MPI program to approximate the value of pi and run it a few times with the time command to get an avg. execution time:

```
$ time mpirun -n 8 ./pi2 50000000
50000000 times on 8 ranks the value of pi is 3. 141626730000000034

real 0m1.772s          ← elapsed (wallclock) time
user 0m13.964s         ← cpu time, user space
sys  0m0.041s          ← cpu time, kernel space
```

Record avg. wallclock times for different iteration counts: 100000, 5000000, 50000000:

1 rank :	.010	.088	.864
4 ranks :	.011	.173	1.217
8 ranks :	.080	.328	2.207
16 ranks :	.192	.700	4.156

Task #2

Convert this code to use OMP instead of MPI. This may involve some significant "gutting" of the code. Test it to be sure you're getting reasonable results (in the neighborhood of 3.141).

Again using the time command get some avg. execution times; choose a scheduling discipline to get the best run times. Optimize your code to get it as fast as you can. Can you approach MPI speeds?

Record avg. times for different iteration counts: 100000, 5000000, 50000000:

1 thread :	.010	.148	1.406
4 threads :	.064	2.693	27.095
8 threads :	.058	2.967	24.848
16 threads :	.060	2.886	26.954

Task #3

Recompile this code so as not to use OpenMP, but run as a serial program. This may involve some minor code changes – specifically the addition of a `#ifdef` to see if OpenMP is being used. Run this program a few times to get an average run time for iteration counts: 100000, 5000000, 5000000.1

thread :	.006	.142	1.367