EE/CSCI 451
Spring 2019
Programming Homework 3
Assigned: Feb 15, 2019
Due: March 2, 2019, before 11:59 pm
Total Points: 50

1 Login to HPC

- The host is: hpc-login**3**.usc.edu
- Username and password are the same as your email account
- **Do not** run your program in the login node. After login, use the 'srun' command to run your program on a remote node. For example:
  srun -n1 **-c**8 ./run

2 Estimating π [25 points]

In this problem, you will use OpenMP directives to parallelize a serial program. The serial program is given in 'p1_serial.c'. You need use OpenMP to parallelize the loop between Line 28 and Line 32. To compile the program, type: gcc -O3 -fopenmp -o run p1_serial.c

1. Use 1, 2 and 4 threads and the **DO/for** directive to parallelize the loop. Name this version as 'p1_omp.c'.
2. Report the execution time of your code.

Hint: you may need the **REDUCTION** data attribute [1].

3 QuickSort [25 points]

In this problem, you need implement the quick sort algorithm [2] to sort an array in ascending order. Quick sort is a divide and conquer algorithm. The algorithm first divides a large array into two smaller sub-arrays: the low elements and the high elements; then recursively sorts the sub-arrays. The steps are:

- Pick an element, called a pivot, from the array.
- Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.
- Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

The serial program is given in 'p2_serial.c'. You need use Pthread to parallelize . To compile the program, type: gcc -O3 -lpthrad -o run p2_serial.c

1. Pthread version: randomly pick up an element of m, $m[rand()\%size]$, as the pivot, reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position, say f. **Run 2 threads in parallel**, one calling Quicksort$(m, 0, f-1)$ and the other calling Quicksort$(m, f, size-1)$. Name this program as 'p2_pthread.c' and report its execution time. **Note that the Quicksort function is the same as the one used in serial version.**
2. (**Optional!**) OpenMP version: you may need the OpenMP **section** clause.

4 Submission Instructions

- Two .c files: 'p1_omp.c' and 'p2_pthread.c'. Make sure your program is runnable. (15+15 pts)
- Report. Write clearly how to compile and run your code. Screenshot of the execution time and performance on hpc. (20 pts)

You may discuss the algorithms. However, the programs have to be written individually. Submit the code and the report via ee451spring2019@gmail.com. Please make sure to include your name, student ID and the homework number in the PDF, and name your PDF file lastname_firstname_pa#.

References

[1] "OpenMP Reduction,"
`http://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-reduction.html`

[2] "Quicksort,"
`http://en.wikipedia.org/wiki/Quicksort`