CSE 415: Introduction to Parallel Computing
Spring 2018, Homework 3
Due 11:59 pm, March 2, 2018

*This homework is 15% of your **homework grade**.*

***Important note:** Please use a word processing software (e.g., MS Word, Mac Pages, Latex, etc.) to type your homework. Follow the submission instructions at the end to turn in an electronic copy of your work.*

1) **[15 points] Shared Memory Architectures**
   Compare NUMA and UMA architectures in terms of their designs and performance characteristics.

2) **[25 points] Multistage Interconnect**
   Draw the multistage baseline interconnect for 16 processors using 4 stages.

3) **[60 pts] Estimation of the value of π**

The integral $\int_0^1 \sqrt{1-x^2}$ evaluates to $\pi/4$ in exact arithmetic. In this assignment, you will numerically approximate this integral using the trapezoidal rule. The trapezoidal rule for *N+1* sample points is as follows:

$$\int_a^b f(x)\,dx \approx h[\frac{1}{2}f_0 + f_1 + \cdots + f_{N-1} + \frac{1}{2}f_N] = \frac{h}{2}f_0 + h\sum_{i=1}^{N-1} f_i + \frac{h}{2}f_N.$$

Here, x_0 = a (starting point), x_N = b (ending point), h = $\frac{b-a}{N}$ is the distance between each sample.

Below is a pseudocode that implements the trapezoidal rule:

```
a = 0
b = 1
h = (b-a)/N
total = h/2(f(a)+f(b))
for i = 1 … N-1
      x = a + i * h
      total += f(x) * h
pi = total*4
```

The function f above calculates $\sqrt{1-x^2}$ for a given x. You can either implement it as a function, or simply calculate it within the for loop to avoid function call overheads (for better performance). Clearly, as the number of samples *N* increases, we get a better approximation, but at increased computational cost.

i) [25 pts *for correctness*] Using the skeleton code provided, first implement a sequential code to estimate π. Then create two OpenMP parallel versions of your code. Your first OpenMP parallel version must use atomic updates (at each iteration), and the second version must use reduction to resolve the race conditions among threads in estimating the number π.

ii) [35 pts *for performance and interpretation*] On the **intel16** cluster, plot the speedup and efficiency curves for your OpenMP parallel versions using 1, 2, 4, 8, 14, 20 and 28 threads for different problem sizes, e.g., 1000, 100000, 10 million, and 1 billion iterations. Analyze how do the two different OpenMP versions compare against each other with respect to problem size, i.e. how do their relative speedup and efficiencies compare? Explain your observations.

Instructions:

- **Accessing the git repo.** You can pull the skeleton codes from the git repo. Please refer to *"Homework Instructions"* under the *"Reference Material"* section on D2L.

- **Discussions.** For your questions about the homework, please use the Slack group for the class so that answers by the TA, myself (or one of your classmates) can be seen by others.

- **Compilation and execution.** You can use any compiler with OpenMP support. The default compiler environment at HPCC is GNU, and you need to use the –fopenmp flag to compile OpenMP programs properly. Remember to set the OMP_NUM_THREADS environment variable, when necessary.

- **Measuring your execution time properly.** The omp_get_wtime() command will allow you to measure the timing for a particular part of your program (see the skeleton codes). *Make sure to collect at least 3 measurements and take their averages while reporting a performance data point.*

- **Executing your jobs.** On the dev-nodes there will be several other programs running simultaneously, and your measurements will not be accurate. After you make sure that your program is bug-free and executes correctly on the dev-nodes, the way to get good performance data for different programs and various input sizes is to use the interactive or batch execution modes. *Note that jobs may wait in the queue to be executed for a few hours on a busy day, thus plan accordingly and do not wait for the last day of the assignment.*

    i) **Interactive queue.** Suggested options for starting an interactive queue on the **intel16** cluster is as follows:

    ```
    qsub -I -l nodes=1:ppn=28,walltime=00:30:00,feature=intel16 -N myjob
    ```

    The options above will allow exclusive access to a node for 30 minutes. If you ask for a long job, your job may get delayed. Note that default memory per job is 750 MBs, which should be plenty for the problems in this assignment. But if you will need more memory, you need to specify it in the job script.

    ii) **Batch job script.** Sometimes getting access to a node interactively may take very long. In that case, we recommend you to create a job script with the above options, and submit it to the queue (this may still take a couple hours, but at least you do not have to sit in front of the computer). Note that you can execute several runs of your programs with different input values in the same job script – this way you can avoid submitting and tracking several jobs. An example job script is provided in the file *job_script_pi.qsub*.

- **Submission.** Your submission will include:
    - A pdf file named "HW3*_yourMSUNetID*.pdf", which contains your answers to the non-implementation questions, and report and interpretation of performance results. *If your*

*assignment is written using a document processing software such as word, please export the document to PDF format for your homework submission. Do not submit the *.doc file.*

- All source files used to generate the reported performance results. Make sure to use the exact files names listed below:
  - sequential.c
  - atomic.c
  - reduction.c

  These are the default names in the git repository –do not change the directory structure, file names, or form of the function declarations.

To submit your work, please follow the directions given in the "Homework Instructions" under the "Reference Material" section on D2L. Make sure to strictly follow these instructions; otherwise you may not receive proper credit.