CS 221 C and Systems Programming

Assignment 10
Due at 11:59 pm on December 9, 2015

1 Pthreads Programming with Busy-Waiting (25 marks)

Implement a parallel program using `Pthreads` to compute the `minimum` and `maximum` values of a large array of double values. The program, called `findminmax_busy`, should take 2 command line arguments: the number of threads to use and the size of the array.

```
> findminmax_busy thread_count size
```

Your program should use the C random number generator with a specified seed to create an array of the specified size of double values, hence the results are reproducible. You can find the array generator code, provided by Dr. Peter Pacheco, on `Canvas` under November 30's class exercises. Your thread function should first calculate the portion of the array that the thread would compute its `maximum` and `minimum` values. Special attention should be made so that your code would work on array of any size regardless it is dividable by the number of thread or not. Next, use `busy-waiting` mechanism to execute the code in critical section.

Test your program by using the random number seed 1 (`srandom(1)`) for array generation and then run the program using thread count of 4 and array size of 1000000:

```
> findminmax_busy 4 1000000

min:  0.000006, max:  0.999997
```

Note that the provided random number generator normalizes double values to be between 0 and 1.

2 Pthreads Programming with Mutexes (25 marks)

The `Pthreads` program, called `findminmax_mutex`, is similar to that of the previous question, except `mutex`, instead of `busy-waiting`, is used to execute the code in the critical section. When running your code on the same command line arguments, it should produce the same results:

```
> findminmax_mutex 4 1000000

min:  0.000006, max:  0.999997
```

3 Pthreads Programming with Semaphores (25 marks)

The `Pthreads` program, called `findminmax_sem`, is similar to that of the previous question, except `semaphore`, instead of `mutex`, is used to execute the code in the critical section. When running your code on the same command line arguments, it should produce the same results:

```
> findminmax_sem 4 1000000

min:  0.000006, max:  0.999997
```

Note: Unnamed semaphores – which we're using – are not implemented on MacOS X. So you should develop your semaphore program on a Linux system

4 Performance Comparison (25 marks)

To be added, as I am discussing with Dr. Peter Pacheco of using penguin cluster to run experiments for data collections to conduct performance comparison.