ECE574: Cluster Computing – Homework 4
Pthreads

Due: Friday 9 October 2015, 5:00pm

1. Background

   - In this homework we will take the sobel code from HW#3 and parallelize it using pthreads.

2. Setup

   - For this assignment, log into the same Haswell machine we used in previous homeworks. As a reminder, use the username handed out in class and ssh in like this
     `ssh -p 2131 username@vincent-weaver-2.umelst.maine.edu`

   - Download the code template from the webpage. You can do this directly via
     `wget http://web.eece.maine.edu/~vweaver/classes/ece574_2015f/ece574_hw4_code.tar.gz`
     to avoid the hassle of copying it back and forth.
   - Decompress the code
     `tar -xzvf ece574_hw4_code.tar.gz`
   - Run `make` to compile the code.
   - You may use your own code from HW#3 as a basis for this assignment. (Alternately some really poorly-optimized sample code is provided). To use your code just copy your `sobel.c` file from HW#3 over top of the `sobel_coarse.c` file in the HW#4 directory.

3. **Coding (7 points)**

   Implement simple two-thread parallelism where you run sobel_x and sobel_y in parallel, but then join and do the combine step serially.

   - Edit the file `sobel_coarse.c`
   - Convert the code to use pthreads.
   - You may need to add `#include <pthread.h>`
   - Modify generic_convolve to be of `void *` type and take one `void *` argument. You will have to create a `struct` to hold the values you want to pass in and do some casting back and forth from the void pointer. This is some tricky C coding, so the provided `sobel_coarse.c` example shows you how to do this.
   - Create one thread for each convolve operation using `pthread_create()`
   - Once both threads are running, have the main thread wait for them using `pthread_join()`
   - Be sure to comment your code!
   - Compare the results generated to make sure they match the output given by your HW#3 code.
   - Run your code using
     `sbatch time_sobel.sh`
     Which will use the provided `IMG_1733.JPG`
     Report how long it takes to run compared the the time taken by your single-threaded HW#3 code.

4. **Instrument with PAPI (2 points)**

   I had some trouble getting inherited perf events to work on this code. So instead we will use a different feature of PAPI, which is gathering timing values.

   - You can comment out the code that creates the eventset and starts/stops it, we won't be needing that.
   - With PAPI you can gather a current timestamp with microseconds granularity via `PAPI_get_real_usec()`.
   - To measure how long a routine is, just measure the timestamp before and after, then subtract. The value is a 64-bit one, so make sure you assign it to a value of type `long long` and print it using the `"%lld"` option in `printf()`.
   - Have your code measure the total convolution time, the combine time, and the `load_jpeg()` and `store_jpeg()` times and print the results to the screen.

5. **Something Cool (1 point)**

   This is complicated and I made it worth not many points so do not waste too much time on it unless you want to.

   - Instead of doing simple 2-thread parallelism, parallelize the entire code base at a fine-grained level.
   - Copy your `sobel_coarse.c` file over `sobel_fine.c` and then modify the `sobel_fine.c` file.
   - A straightforward way of doing this (but not the only way) is to create 8 threads, run sobel_x in parallel, join then create 8 threads, run sobel_y in parallel, join, then create 8 threads, run combine in parallel, join, finish.
   - You might want to start out doing the above using just 1 thread first, and the results should be similar to your timing results for your previous sobel_fine
   - Record the total time (using time) as well as the PAPI timing measurements for 1, 2, 4, and 8 threads.

6. Submitting your work.

   - Be sure to edit the README to include your name, as well as the timing results, and any notes you want to add about your something cool.
   - Run `make submit` and it should create a file called `hw04_submit.tar.gz`. E-mail this file to me.
   - e-mail the file to me by the homework deadline.