ECE574: Cluster Computing – Homework 5
OpenMP

Due: Friday 24 February 2023, 5:00pm

1. **Background**

   - In this homework we will take the sobel code from Homeworks #3 and #4 and parallelize it using OpenMP.
   - A helpful OpenMP tutorial can be found here:
     `https://hpc-tutorials.llnl.gov/openmp/`

2. **Setup**

   - For this assignment, log into the same Haswell-EP 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@weaver-lab.eece.maine.edu`
   - Download the code template from the webpage. You can do this directly via
     `wget http://web.eece.maine.edu/~vweaver/classes/ece574/ece574_hw5_code.tar.gz`
     to avoid the hassle of copying it back and forth.
   - Unpack the code
     `tar -xzvf ece574_hw5_code.tar.gz`
   - Run `make` to compile the code.
   - You may use your own code from a previous assignment as a basis for this assignment. (Alternately, the same poorly-optimized sample code from HW#4 is provided). It might make more sense to reuse your HW#3 code or the HW#4 coarse code as a basis rather than having to back out any optimizations from your HW#4 fine code. Just copy your un-parallelized code over `sobel_serial.c`, `sobel_coarse.c` and `sobel_fine.c`.

3. **Coarse-grained Parallelism (4 points)**

   Implement simple two-thread OpenMP parallelism where you run sobel_x and sobel_y in parallel, but it joins before doing the combine step serially.

   To do this, use the OpenMP Sections directives. Remember that OpenMP will automatically do a join at the end of a parallel section.

   - Edit the file `sobel_coarse.c`
   - Convert the code to use OpenMP.
   - You may need to add `#include <omp.h>`
   - Be sure to comment your code!
   - Compare the results generated to make sure they match the output given by previous homeworks.
   - Run your code using
     `sbatch time_coarse.sh`
     which will use the provided `space_station_hires.jpg`.
   - **Report in the README how much time it took for this code to run**

- Modify `time_coarse.sh` so it sets `OMP_NUM_THREADS=1` (for best results put this after the "time" but before the "./sobel_serial") and re-run and get the single-thread time
- **Report in the README the single-thread time as well as the speedup from having parallel code**

4. **Performance Measurement (1 point)**

   - Just like HW#4 use PAPI to measure the time various subcomponents take to run. Have your code print to the screen the wallclock time taken by:
     (a) `load_jpeg()`
     (b) overall sobelx/sobely
     (c) combine
     (d) `store_jpeg()`

5. **Fine-grained Threading (4 points)**

   For this part, update the code to do some sort of fine-grained parallelism. How you do it is up to you. The most straightforward way of doing this is using an OpenMP `for` directive. The easiest way to do this is to go into your `convolve` and `combine` routines and convert one of the for loops to be parallel.

   - For this exercise modify the `sobel_fine.c` file.
   - Some things to watch out for: remember to mark as private your various loop iterators and other variables (such as sums, etc.)
   - If you don't want to have to keep checking the image to be sure your code is working, an alternate is to use a checksum like `md5sum` to verify the output file matches. (the md5sum of the sobel output from space_station_hires.jpg is `7a17b02fe7e4e676b575f6f66ba4fa01`)
   - **Record in the README the total time (using time) as well as the PAPI timing measurements for 1, 2, 4, 8, 16, and 32 threads.**
     Change the thread count by modifying the `OMP_NUM_THREADS` value in `time_fine.sh` before running sbatch. Please don't hard-code the thread count into your program.
   - **Change the thread scheduler from static to dynamic. Does this change your performance in the 16-thread case?**

6. **Something cool (1 point)**

   Do something cool to further improve the performance of your code. It can be one of the following, or else you can try something of your own. Copy your code over to `sobel_cool.c` and edit that for this part.

   - Change another option in the OMP FOR directive (scheduler, loop collapse, simd, etc) and report how it changes the result in the 16-thread case.
   - See if you can work out a way to use an openmp-reduction in your code, and see if it helps performance.

7. **Submitting your work**

   - Be sure to edit the README to include your name, as well as the timing results and answers to questions.
   - Run `make submit` and it should create a file called `hw05_submit.tar.gz`.
   - e-mail the file to me by the homework deadline.