Assignment 1: Sequential warmup

Xun Zhu

Question 1

The source code is included in the folder under the name `main.c`. Parameters are defined as C-preprocessor variables and algorithm (i-j, j-i, or tiled) and block size are chosen at the compile time.

Note that in the tiled version, I used four separate for-loops for the edge cases when N is not divisible by the block size, as opposed to having one for-loop and branching using if-statements, which would be slower.

With $N = 18,000$ and using the Intel Compiler (`icc`), the running time for i-j was about 2 seconds. However, with the array being of type double (8 bytes = 64 bits), this means the the memory allocated for either arrays would be

$$64 \times 18,000^2 \approx 4.83 \times 2^{32}\text{bits},$$

which necessitates the `-mcmodel=medium` parameter when compiled.

An interesting observation is that `icc` (14.0.2.144 Build 20140120) did not generate faster code than `gcc` (7.1.0). The binary created by the latter finished in 1.99 seconds.

I wrote a Python script for invoking the compiler with various parameter combinations, collecting the results, and aggregate the results into a CSV file.

I noticed that the first few runs take much longer than following runs, usually over 8 minutes or so. I suppose this is when the OS is recovering from the last task which potentially tapped into swap, but I'm not so sure. For example, here are the logs of the first few runs in a particular experiement, notice how the wall clock time dropped from hundreds of seconds to about two seconds:

algo	bs	wall_time	rep	l1_load_misses	llc_load_misses
ij	1	194.70775351	1	4719950260	168155701
ji	1	589.410447918	1	11118078659	317424545
tiled	512	212.40171788	1	5067632874	169008293
tiled	1	11.366876506	1	645914378	47143981

algo	bs	wall_time	rep	l1_load_misses	llc_load_misses
tiled	2	1.969204341	1	451714702	42381861
tiled	3	1.965555544	1	451693187	42362063
tiled	4	1.992775057	1	451775277	42408239
tiled	5	1.99024135	1	456029172	45395570

To compensate for that, I decided to do a couple of *warmup* runs before the real runs.

Question 2