C1544657
CM3103
03/12/2017

High Performance Computing

Coursework 1

Programming with OpenMP

This report consists of 5 sections. These are:

1) Hardware and Software environment
2) Description of the timing experiments
3) Graphs
4) Discussion of Results
5) Conclusion

Hardware:

Processor: Intel Core i7 -4790 CPU @3.60GHz Quad Core, 8 threads, 8MB Smart Cache
GPU: GeForce GTX 960, 2048MB Total Memory, 128-bit Memory Interface
RAM: 16GB(2x8GB)

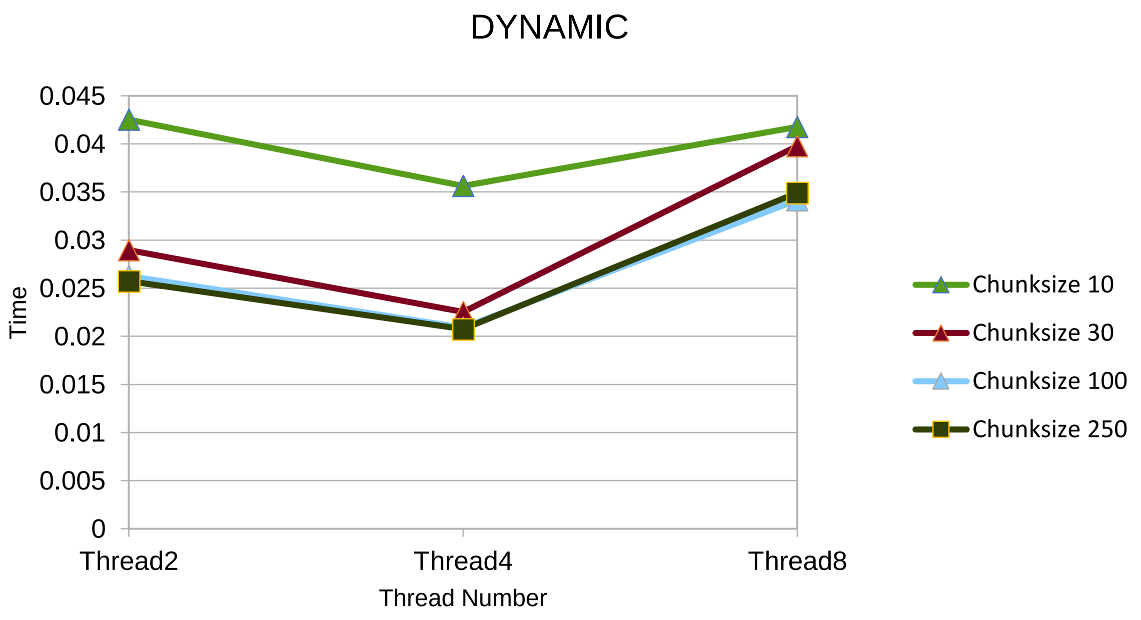
Software:

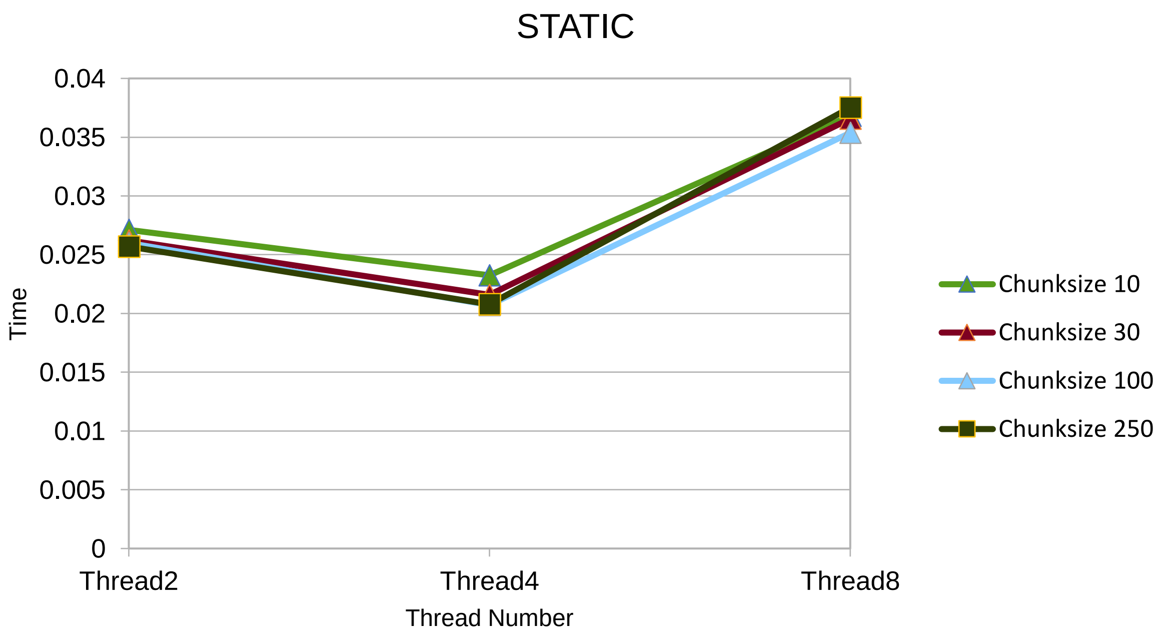
OS: Linux-x86_64
Editor: Sublime text
Programming Language: C
Software: OpenMP

Description of the timing experiments:

Firstly, I run the code provided to us 10 times to find the average and the standard deviation. The next step was to parallelize the code to take measurements of different numbers of chunksize values and different numbers of Threads. The numbers of Chunksizes I chose were 10,30,100,250. I chose these values to have both very small chunksizes and large chunksizes, to be able to experiment and make some observations to reach a conclussion on how much chunksize affects the parallel program. The Thread numbers used were 2,4,8. Timings for Dynamic and Static scheduling were recorded.
Below you will find the graphs that represent the average timings for different numbers of chunksizes and threads, both for static and dynamic scheduling.

C1544657
CM3103
03/12/2017





C1544657
CM3103
03/12/2017

Discussion of results:

The first observation I made from my experiments was that the higher the number of threads did not necessarily mean less time of execution. For our particular case, the program does not require a lot of processing power to run efficiently, as it is not so demanding. From the graphs we can see that the number of threads with the lowest time of execution is 4 threads. The runtimes for 2 threads are slightly higher than the ones of 4 threads and the runtimes for 8 threads are significantly higher than the ones of 4 threads. This means that using 4 threads the program performs at its fastest. This can be seen with all values for chunksizes, confirming my hypothesis. For dynamic scheduling we can see a significant difference in using a small chunksize, as using chunksize 10 results in a much slower execution and using chunksize 30 results in a slower execution compared to the faster 100 and 250 chunksizes. In static scheduling we can't spot a large variation in time using different chunksizes.

Conclusion:

Before this project I had minimal experience with the C programming language. This assignment has helped me develop my C programming skills and understand why parallel computing is needed in our society. I have gained the skill to know when a program needs to be modified to run in parallel to achieve better run time and how to perform this. My most important conclusion is that the higher the number of threads does not necessarily mean we're getting the best performance.