Lab 4

Deeksha Juneja

2/16/2016

Introduction

In lab 4, we have to understand threads and extend the given nqueens.c to make it a multi-threaded program. The main thread functions that I used were create and join.

Part 1

The first thing we did was created a thread. We use a program that has two different threads that are created in addition to the main thread.

The expected output of running the code: The expected output for me was to see the execution of thread one, then thread two and then the main. Hence it was:

“I am thread 1”

“I am thread 2”

“I am thread 0”

But surprisingly:

The actual output of the code was “I am thread 0”. This is because the main thread did not wait for thread 1 and thread 2 to complete their task before returning from the program.

Part 2

In this section we address the problem of main thread not waiting for the other threads. We use join to make the main thread wait for the other two threads. We add the join statement before the main thread returns from the program.

Here, the output of the program was “I am thread 1”

“I am thread 2”

“I am thread 0”

This is the expected output of the program as well. So joining solves our problem.

Part 3

We also look at another program called parallel_merge and played around with it to check how long it takes the program to merge in different situations.

Part 4

Nqueens is a very interesting problem where each queen has to be placed such that it cant attack the other one. So no two queens can be in the same row, column or diagonal. The code provided implements the nqueen problem using a single thread and my task was to solve it by using multiple threads. The way I solved it was to make threads equal to the number of queens. From there on it was rather easy to get the code going. I am creating an array of threads where the size of the array is equal to the number of queens. I am doing this in the for loop in main. I have added another for loop to join all the threads.

No of Queens	Threaded	single-threaded
10	0:00.13	0:00.14
11	0:00.54	0:00.57
12	0:03.23	0:03.34
13	0:19.19	0.20.01
14	1:52.17	2:01.09
15	12:51.74	13:53.42

It is evident for a small number of threads, the threaded program is faster than the single threaded program. Threads require more CPU usage, need more memory resources etc. Due to this, when there is a large number of threads, the threads take a longer time. Moreover, in this particular solution, the more the number of queens, more computation needs to be done. As more computation needs to be done and the amount of computation goes up non-linearly, the work of a thread is also going up non-linearly and hence, the time also goes up non-linearly. The more the threads used, the more operating system resources get used and the performance decreases. Moreover, keeping track of all the threads also takes time. Moreover, we are measuring CPU-seconds, i.e. time spent by each CPU cumulatively. This measure won't go down with additional threads, because it is proportional to amount of work which is constant. On the other hand, this may go up with number of threads, as every new thread incurs some additional bookkeeping. Also, printing takes a long time as its an expensive task. Also, tasks that are done in sequence instead of in parallel determine the amount of time because the thread has to wait upon them.

If the task to be done is rather trivial, then it is not a good idea to use multi-threading because it will be more time consuming than a single thread. In such a case, a single thread will give much more efficient and faster results. This is because the overhead of creating a thread is going to be much more time expensive. Also if we have operations which can only be done sequentially and not in parallel, then it would make more sense to use a single thread.

Conclusion

In the end, I learnt quite a lot about threads and solving nqueens for pretty interesting. Before this lab, I did not know how to use threads and I got to learn a lot. It was interesting to create multiple threads for multiple queens. The actually code that I wrote was extremely short but thinking it through was interesting. I learnt how to use create and join function effectively. This program also increased my knowledge of C programming.