Hamza Alwani
1001638905

Threads

Part A

Problem:

Given two string, S1 and S2. Count all the occurrences of S2 in S1. The search has to be done in concurrently. The experiment is set to see how much time it takes to execute the same program and same string but with different number of threads.

Thread Library:

For the thread library I chose to use C++Threads. I chose this library since the program was written in C++. Some of the benefits of this library is that it allows multithreading and running functions parallelly. Along with its high level of functionality and its powerful library it also extremely easy to use and to write.

Design of the Experiment and the Program:

The program begins by reading and setting up s1 and s2 once set up done the Clock starts.

Then the program divides the string s1 into #NUMOFthreads. Then each string is sent into its own thread along with an extra buffer, so each word is fully checked. Once a thread is over it joins back into the main program once all the threads join the timer is stopped. The time and total occurrences are printed. To make the experiment more precise and accurate the program was run 1000 times with 1,2,3,4, and 5 threads. To keep the experiment fair all the threads used the “shakespear” file as inputs;

Hamza Alwani
1001638905

You need: g++. If you don't have g++: $ *sudo apt install g++*

To build: $ *g++ thread.cpp -o thread -lpthread*

To run $ *./thread "yourfile"*

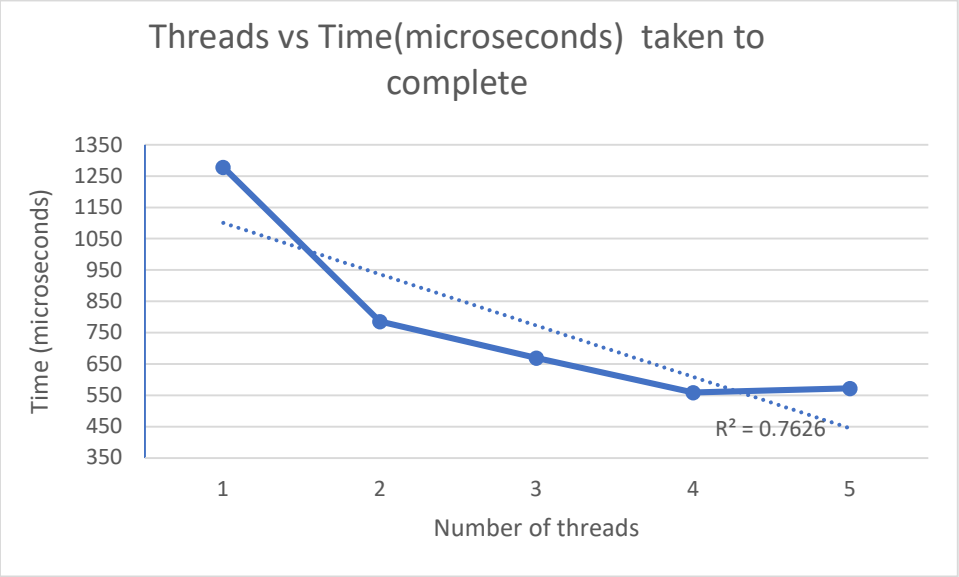
Results:

Threads	Time (microseconds)
1	1278.77
2	785.864
3	668.992
4	559.345
5	571.689

*see reulsuts.xlsx for full data.

The results clearly show that there is a huge difference in time as you go from 1 thread (1278.77 microseconds) to 2 threads(785.864 microseconds) and each thread after that also has a significant difference. Up until 5 threads which is slightly slower than 4 threads.

Graph Analyze:



Hamza Alwani
1001638905

The graph shows that there is a liner regression. As threads increase the time to execute decreases. Till a specific point after which it starts to increase again. IN my experiment it seems that point is around 5 threads. This may be due to the fact that the Virtual machine was allocated 5 processors and anything above that would have to wait for a thread to finish therefore not be parallel. Therefore, if there were more points in the graph I believe that the graph would have a quadratic shape where it bottoms out at 4 threads and goes up to infinity as threads go to infinity.

Hamza Alwani
1001638905

Part B

Problem:

Use condition variables to implement the producer-consumer algorithm. Assume two threads: one producer and one consumer. The producer reads characters one by one from a string stored in a file named “message.txt”, then writes sequentially these characters into a circular queue. Meanwhile, the consumer reads sequentially from the queue and prints them in the same order. Assume a buffer (queue) size of 5 characters. Write a threaded program using condition variables.

Thread Library:

For the thread library I chose to use C++Threads and mutex and conditional variable. I chose C++ threads library since the program was written in C++.

Mutex library was chosen since it allows threads to run mutually exclusively.

Finally, conditional variable library was used since the program needed to have a conditional variable.

All these libraries offer high level of functionality and powerful execution all while being extremely easy to use and to write.

Hamza Alwani
1001638905

Design of the Experiment and the Program:

The program begins by creating two Threads, consumer and producer. The producer thread is run n times. N being the number of chars in the file. Whereas consumer is run while *bool keepgoing* is true. On the last char producer changes *keepgoing* to false therefore both the consumer and producer end and join into the main program.

Producer:

1. Gets char from file
2. Pushes char onto circular queue

Consumer:

1. Gets char from que
2. Pops char from queue using global index and Prints char

You need: g++. If you don't have g++: $ *sudo apt install g++*

To build: $ *g++ con_pro.cpp -o con_pro -lpthread*

To run $ *./ con_pro.cpp*