Mohammed Ahmed 1001655176

Threads Lab Report

Part 1

1. Problem Statement and Evaluation Metrics

Given two character strings, I had to write a threaded program to find the number of substrings, in string s1, that is exactly the same as s2.

2. Choice of threading library

I chose to utilize the pthread library. The reason I chose to use this library is I have used this library in the past with other projects. Additionally, pthread is already set up on my virtual machine, so it was easier for me to work with this threading library.

3. Design of Experiment

I made several modifications to the substring.c program in order to conduct this experiment. I made it so that my program would break up the string into different parts in order to split up the work, and the number of parts it was broken into depended on the number of threads specified in the #define.

In order to conduct this experiment, I tested my program with 1, 2 and 4 threads to try to see a trend between the time it took to execute my program and the number of threads. I wanted to make sure that I collected accurate data, so I ran my program 10 times for each thread amount and averaged the times.

In order to compile and run my program, this are the commands I ran

```
gcc part1.c -lpthread

./a.out tests/hamlet.txt
```

4. Experimental Results

This is the data I collected for my program running 1, 2, and 4 threads

One Thread	
Trial	Elapsed Time (in milliseconds)
1	0.963
2	0.954
3	1.197
4	0.946
5	0.995
6	0.978
7	0.991
8	0.996
9	1.001
10	0.997
Average Elapsed Time	1.0018

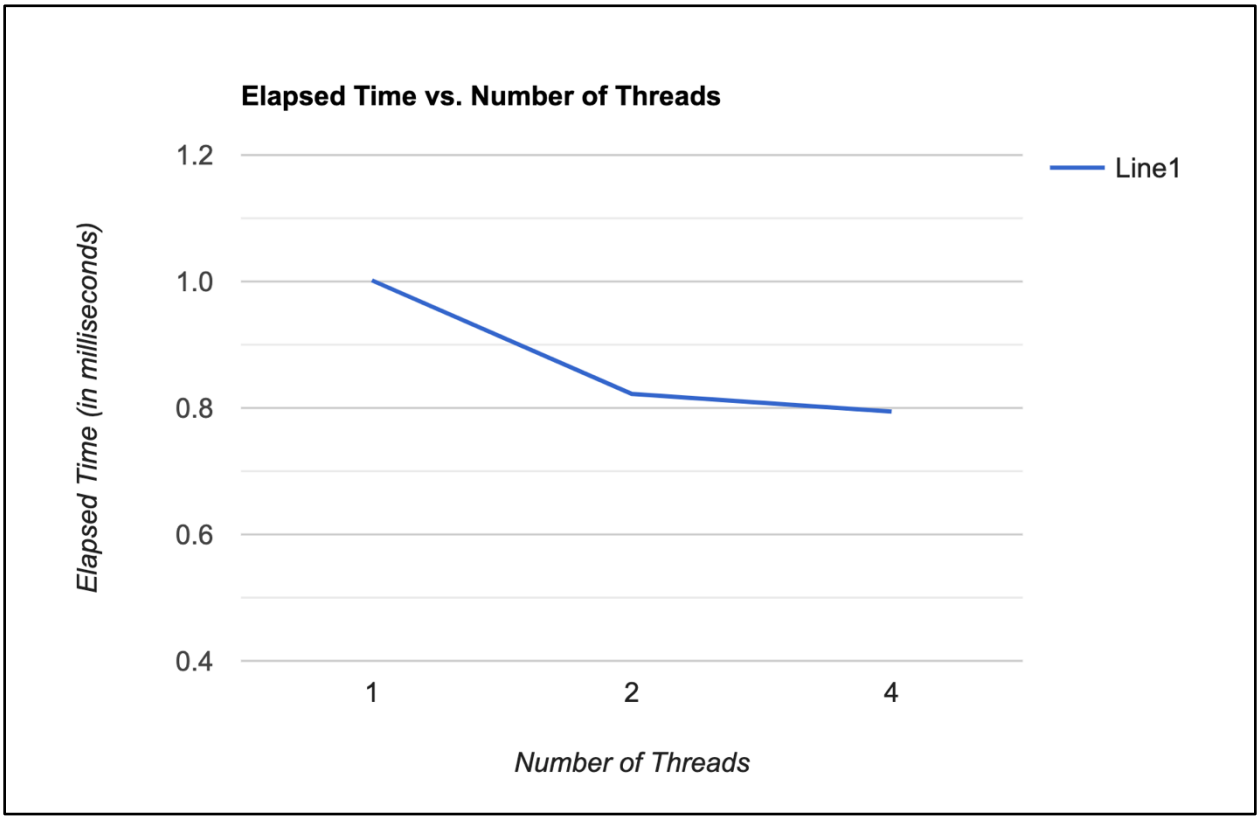
Two Threads	
Trial	Elapsed Time (in milliseconds)
1	0.784
2	0.806
3	0.783
4	0.815
5	0.87
6	0.85
7	0.83
8	0.841
9	0.841
10	0.802
Average Elapsed Time	0.8222

Four Threads	
Trial	Elapsed Time (in milliseconds)
1	0.791
2	0.804
3	0.795
4	0.743
5	0.813
6	0.783
7	0.792
8	0.871
9	0.784
10	0.766
Average Elapsed Time	0.7942

5. Analysis and Graphing of Experimental Results

Based on the data I collected, there appears to be a correlation between the number of threads I had in my program, and the time it took for my program to run. The more threads that I added in my program, the elapsed time reduced. There is a big increase in performance when I switched from one thread to two threads, and this really shows how multithreading a program can increase performance. The time reduced again switching from two to four threads, but the change was not as drastic.

Below is a graph showing this relationship.



Mohammed Ahmed 1001655176

Part 2

1. Problem Statement and Evaluation Metrics

The problem is using conditional variables, I needed to implement the producer-consumer algorithm using multithreading. I had to utilize two threads, one for the producer and one for the 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.

2. Choice of threading library

I chose to utilize the pthread library. The reason I chose to use this library is I have used this library in the past with other projects. Additionally, pthread is already set up on my virtual machine, so it was easier for me to work with this threading library.

3. Design of Experiment

I created two functions in order to implement the producer-consumer algorithms. My producer function read from the file, and stored the letters into a circular queue I implemented with an array. My consumer function prints from the buffer, outputting the message in the same order as the file. In order to prevent jumbled letters, I utilized two semaphores letters_added and letters_consumed, to control the flow of the program and ensure that the message would be printed correctly.

In order to compile and run my program, this are the commands I ran

```
gcc frys1.c -lpthread

./a.out
```

4. Experimental Results

The message printed successfully on to the screen.

5. Analysis and Graphing of Experimental Results

N/A