Shubhayu Shrestha 1001724804

Professor Trevor Bakker

CSE 3320-002: Operating Systems

17 March 2021

Assignment 2 - Threads

Part 1:

Problem Statement and Evaluation Metrics:

For part 1 I was given two character strings, and I was required to write a threaded program to find out the number of substrings, s2, in a main string, s1.

Choice of Threading Libraries:

My choice of threading libraries for this experiment was utilizing pthreads as this was the thread library I was most comfortable using and understood the most, after gaining experience from my 1325 Object Oriented Programming course. Furthermore, this library was already set up in the provided VM.

Design of the Experiment:

For this experiment, I wrote a program, utilizing the substring.c program provided in the class GitHub, that will check the main string from a file in which it will compare another string to the main string, iterating through the main string and finding the number of substrings in the file. The main goal of this experiment was to see if utilizing multiple threads would affect the elapsed time. By setting the number of threads as a define, it allowed me to change the number of threads utilized and split up the work amongst them. When collecting the data, I ran the program 50 times for each thread and collected the data and then averaged the times for comparison.

I compiled and ran my code utilizing the provided hamlet.txt program:

```
gcc part1.c -lpthread

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

Collected Data:

Data for One Thread:

Trial Number	Elapsed Time (in milliseconds)
1	1.07
2	1.033
3	1.023
4	0.967
5	0.956
6	1.02
7	1.054
8	0.994
9	1.157
10	1.128
11	1.047
12	1.033
13	1.05
14	1.067
15	1.056
16	0.921
17	1.023
18	1.502
19	1.009

20	1.028
21	0.997
22	0.997
23	0.991
24	1.25
25	1.05
26	0.976
27	0.937
28	0.945
29	1.025
30	1.041
31	1.028
32	1.13
33	1.017
34	1
35	1.038
36	1.039
37	1.034
38	0.995
39	0.998
40	1.113
41	1.036
42	1.178
43	1.003
44	1.116
45	1.015
46	1.085
47	1.347
48	1.049
49	1.07
50	1.047
Average:	**1.0537**

Data for Two Threads:

Trial Number	Elapsed Time (in milliseconds)
1	0.93
2	0.886
3	0.929
4	0.872
5	0.975
6	0.902
7	0.947
8	0.981
9	0.959
10	0.915
11	0.858
12	0.904
13	0.859
14	0.974
15	0.908
16	0.901
17	0.91
18	0.869
19	0.895
20	0.831
21	0.881
22	0.866
23	0.851
24	1.019
25	0.966
26	0.913
27	0.827
28	0.925
29	0.892
30	0.89
31	0.908

32	0.916
33	0.861
34	0.902
35	0.891
36	0.963
37	0.855
38	0.875
39	0.923
40	0.875
41	1.102
42	0.856
43	0.832
44	0.9
45	0.852
46	0.928
47	0.86
48	0.861
49	0.869
50	0.962
Average:	**0.90452**

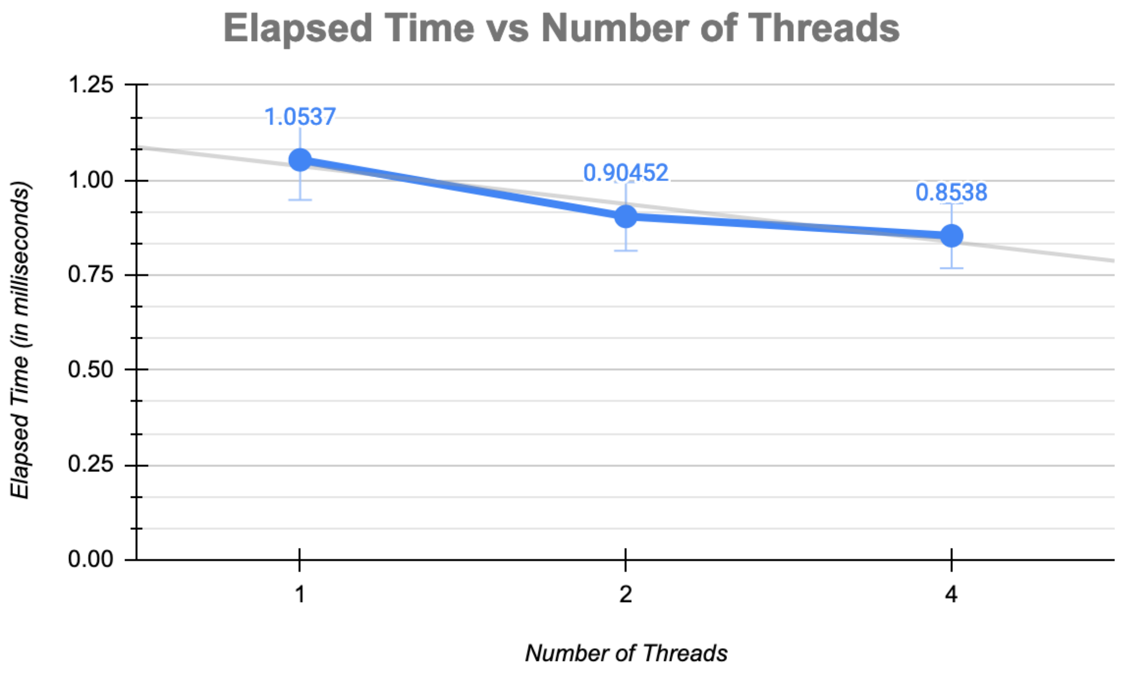
Data for Four Threads:

Trial Number	Elapsed Time (in milliseconds)
1	0.913
2	0.947
3	0.959
4	0.807
5	0.87
6	0.914
7	0.837
8	0.893
9	0.906
10	0.794
11	0.875
12	0.835
13	0.835
14	0.823
15	0.896
16	1.15
17	0.869
18	0.882
19	0.81
20	0.795
21	0.803
22	0.913
23	0.804
24	0.871
25	0.791
26	0.794
27	0.823
28	0.797
29	0.851
30	0.823
31	0.867

32	0.807
33	1.102
34	0.78
35	0.813
36	0.793
37	0.805
38	0.905
39	0.795
40	0.888
41	0.812
42	0.81
43	0.799
44	0.854
45	0.805
46	0.746
47	0.822
48	0.892
49	0.876
50	0.839
Average:	**0.8538**

Data Analysis:

When collecting the data, I separated the data collection to three parts, with each being a different number of threads: 1, 2, 4. After I completed 50 trials for each thread and calculated the average elapsed time for each thread for comparison and to find an overall trend. I found the average elapsed time for 1 thread to be 1.0537 milliseconds, for 2 threads it took 0.90452 milliseconds, and finally for 4 threads it took 0.84198 milliseconds. As shown, with the averages, there is a decrease in elapsed time, hence multithreading and searching for substrings utilizing more threads causes a decrease in elapsed time, as it splits the necessary work. This trend is supported by the graph as well, showing a decrease in elapsed time to complete when increasing the number of threads.



Part 2:

Problem Statement and Evaluation Metrics:

We needed to implement the producer-consumer algorithm, utilizing two threads, in order to read characters from a text file named "message.txt" and print the contents of the file to the user, utilizing a queue with the size of 5.

Choice of Threading Libraries:

My choice of threading libraries for this experiment was utilizing pthreads as this was the thread library I was most comfortable using and understood it the most, after gaining experience from my 1325 Object Oriented Programming course.

Design of the Experiment:

For this experiment, I utilized two semaphores, as this would help us coordinate the activity between multiple concurrently running threads. The two semaphores I utilized were for the producer and the consumer. The producer function was utilized to read from the message.txt file and store the contents of the file into a buffer with the size of 5, which was set as a define at the beginning of the program. The consumer function was utilized to read from the buffer and ultimately print out the contents of the file. The semaphores were utilized to make sure that there was no overlapping between the two processes.

I compiled and ran my code:

```
gcc part2.c -lpthread

./a.out
```

Collected Data:

N/A

Data Analysis:

N/A

Conclusion:

In this program, by utilizing the semaphores, I was able to discover how they are utilized to coordinate the different concurrent activities done by threads and how utilizing two semaphores can work together, reading a file and outputting the contents of the file.