CS5473 - Project 2

Author: Enzo Durel
Professor: Dr. Chongle Pan

February 7, 2025

Contents

1 Problem 1 . . . . . **1**

2 Problem 3 . . . . . **2**

3 Problem 4 . . . . . **3**

List of Tables

1 Problem 1 Runtime 1
2 Problem 1 Speedup 1
3 Problem 1 Efficiency 1
4 Problem 3 Runtime 2
5 Problem 3 Speedup 2
6 Problem 3 Efficiency 2
7 Problem 4 Runtime 3
8 Problem 4 Speedup 3
9 Problem 4 Efficiency 3

Problem 1

$$Speedup = \frac{T_{serial}}{T_{parallel}}$$

$$Efficiency = \frac{Speedup}{p}$$

Table 1: Problem 1 Runtime

Runtime	Test 1	Test 2	Test 3	Test 4
1	3,47	7,14	14,27	28,85
2	1,92	3,91	7,93	15,89
4	1,26	2,97	5,89	8,91
8	1,07	2,13	4,15	8,16

Table 2: Problem 1 Speedup

Speedup	Test 1	Test 2	Test 3	Test 4
1	1	1	1	1
2	1,807291667	1,826086957	1,799495586	1,8156073
4	2,753968254	2,404040404	2,422750424	3,237934905
8	3,242990654	3,352112676	3,438554217	3,535539216

Table 3: Problem 1 Efficiency

Efficiency	Test 1	Test 2	Test 3	Test 4
1	1	1	1	1
2	0,9036458333	0,9130434783	0,8997477932	0,9078036501
4	0,6884920635	0,601010101	0,6056876061	0,8094837262
8	0,4053738318	0,4190140845	0,4298192771	0,441942402

The program is not strongly scalable due to communication overhead, memory bandwidth saturation, or false sharing. We see this because the Speedup does not reach the p value. However, the program runtime does not remain constant even if test sizes are bigger, so the program is only partially weakly scalable.

Problem 3

The program divides the input string size by the number of threads. Each thread will be assigned and encrypt only a portion of the input buffer. Because the threads will not access the same byte in memory, there is no race condition and each threads can work perfectly at the same time. It's using static scheduling.

Table 4: Problem 3 Runtime

Runtime	Test 1	Test 2	Test 3	Test 4
1	0,083	0,173	0,395	0,956
2	0,061	0,092	0,212	0,513
4	0,028	0,061	0,124	0,314
8	0,02	0,05	0,079	0,178

Table 5: Problem 3 Speedup

Speedup	Test 1	Test 2	Test 3	Test 4
1	1	1	1	1
2	1,360655738	1,880434783	1,863207547	1,863547758
4	2,964285714	2,836065574	3,185483871	3,044585987
8	4,15	3,46	5	5,370786517

Table 6: Problem 3 Efficiency

Efficiency	Test 1	Test 2	Test 3	Test 4
1	1	1	1	1
2	0,6803278689	0,9402173913	0,9316037736	0,9317738791
4	0,7410714286	0,7090163934	0,7963709677	0,7611464968
8	0,51875	0,4325	0,625	0,6713483146

I think the program is strong scalable because the speedup tends to approach linear ~p when the size of the problem increases. The Efficiency remains correct with big files as we can see in the Test 4. This program is not weakly scalable because the running time does not remain constant as the problem size grows.

Problem 4

Table 7: Problem 4 Runtime

Runtime	Test 1
1	22,58
2	11,69
4	6,67
8	4,25

Table 8: Problem 4 Speedup

Speedup	Test 1
1	1
2	1,931565441
4	3,385307346
8	5,312941176

Table 9: Problem 4 Efficiency

Efficiency	Test 1
1	1
2	0,9657827203
4	0,8463268366
8	0,6641176471