University of the West of England

PARALLEL COMPUTING
(Brute-force Attack)

LOGBOOK

Prepared by: 20034743
Course code: UFCFFL-15-M

May 13, 2021

Table of Contents

INTRODUCTION:

This loogbook is the part of coursework of the module UFCFFL-15-M UWE Parallel Computing. It contains two parts: development and documentation. In this document, a detailed progress of AES brute force attack is explained using it's development, testing and concluding phases. We use different ways to break the password but our main goal is to increase the speed of password cracking. For this purpose, we have been pre-supplied with salted ciphertext encoded in Base64 and it's plantext equivalent. We used two different ways to crack the password i.e. serial and parallel (OpenMP and OpenMPI). Our recommended language for this coursework is C/C++ and we are using Visual studio code for development palteform. This loogbook is created with maximum effort and time. It's individual work so it's difficult to mention work with timestamp and the work was summerized in logbook at the end.

LOGS OF PROGRESS:

At this stage, I was unaware of the C language as we haven't worked in it from last 4 years. I struggled a lot to become familer with the coding language. I took some courses to learn and recall it and then I started working. First step in this coursework development was, to get required results. Cracking the password through the serial coding was the base, as we have to first crack password then improvement of performance will be considered using OpenMP and OpenMPI.

In this step, our given consideration was to use five characters length with the character only from "a-z, A-Z, 0-9". For that I had to crack the password with the serial code first to get on track. I got a lot of problems to crack the password provided. It took a lot of time to break the password. Using the dictionary with sequence of 0-9, A-Z, a-z. I found these answers for the 4 different cipher texts i.e.

12345 time elpased is 20 seconds

Cipher Text:

"U2FsdGVkX19q3SzS6GhhzAKgK/YhFVTkM3RLVxxZ+nM6yXdfLZtvhyRR4oGohDotiifnR1iKyitSpiB
M3hng+eoFfGbtgCu3Zh9DwIhgfS5A+OTl5a4L7pRFG4yL432HsMGRC1hy1RNPSzA0U5YyWA==\n"

(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is 12345
Time elpased is 20 seconds

29Apr time elpased is 41 seconds

Cipher Text:

"U2FsdGVkX1/Y+mHv2oZdo5MLKEQWCATfc31jSGWXZ6D3gWuLdZYVUrRnGNecV+EdFsMYSWh
Eh1nsP9tMwpQaPeWMP3MZ6G0HCLVw+fjRjYY1Fi+lpuGKdjmZh0Loylw0gVo2SUxNigSvjnn3xAGHg
==\n"

(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is 29Apr
Time elpased is 41 seconds

Mar10 time elpased is 433 seconds

Cipher Text:

"U2FsdGVkX1/x92BdYvopo2z2ZE5u68vEA+00lPDdMF0rr7SGaWdB3+INMw3TWtKNsEI4SKIA0mf87d
j7/Q8KiJ2Wzh6MtdxKAfrjvueXod32tU7F35IdyMWCxJGQZcIey0/DLIW3SHqYhuTSP0GBBQ==\n"

((base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is Mar10
Time elpased is 433 seconds

Zest time elpased is 11 seconds

Cipher Text:

"U2FsdGVkX18IzeFxDZrMxL56zmCxpJTpMMCShpV02j9QRvgeAuvSc6V406zzfuwETgIxJXaqvFHMVu
FXfR+X6ZDFm2SClHRuI9C1yL+JRRRAUZS22BrE8y0XS0Zwhk5JZS3IBRuNSRNgELQ+Fimmsw==\n"

(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is Zest
Time elpased is 11 seconds

After using the dictionary with sequence of "UperCase → LoweCase → Numbers". I found these answers for the 4 different cipher texts i.e.

Zest time elpased is 8 seconds

Cipher Text:

"U2FsdGVkX18IzeFxDZrMxL56zmCxpJTpMMCShpV02j9QRvgeAuvSc6V406zzfuwETgIxJXaqvFHMVu
FXfR+X6ZDFm2SClHRuI9C1yL+JRRRAUZS22BrE8y0XS0Zwhk5JZS3IBRuNSRNgELQ+Fimmsw==\n"

(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is Zest
Time elpased is 8 seconds

METHOD OF PARALLELISATION:

Parallelism approach used in this coursework is SIMD "Single Instruction Multiple Data". It's from one of the best method to parallelism adopted in modern high-performance computing.This method has steam multiple data having sigle instruction, including most array processes. The brute force algorithm used in this coursework operates on a sequential order starting from variable initialisation, base64 decode and then Salt removal. Until this step we cann't implement parallelism. In next step, we move towards the password cracking. While working on this part we have sometimes got much time in password cracking, then parallalism is implemented using OPENMP and OPENMPI.

PERFORMANCE ANALYSIS:

After cracking the passwords through the serial code our main goal was to increase the performance of the Brute force attack because in my cases our system was taking longer time to crack the password. I implemented the parallelism and then I got slight decrease in the time of password cracking.

For the case of password 12345:

Using the Dictionary “UpperCase → LowerCase → Digits”. We got slight difference in our system. For this dicitionary the password come at the end of the dictionary and it look longer to break. We got different results as below:

1. *Serial Approach:*

```
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is 12345
Time elpased is 1171 seconds
```

2. *Parallel Approach:*

```
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is 12345
Time elpased is 1070 seconds
```

There is a decrease of almost 101 seconds with difficult dictionary.

After changing the dictionary to “Numbers → UpperCase → LowerCase”. We got the password more then 100 times faster because of easier and password related dictionary. Take a look:

3. *Serial Approach:*

```
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is 12345
Time elpased is 20 seconds
```

4. *Parallel Approach:*

```
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ gcc crackaes.c -lcrypto
(base) malikusmannawaz@malikusmannawaz:~/Desktop/UWE/4 PC/Codes$ ./a.out
Password is 12345
Time elpased is 21 seconds
```

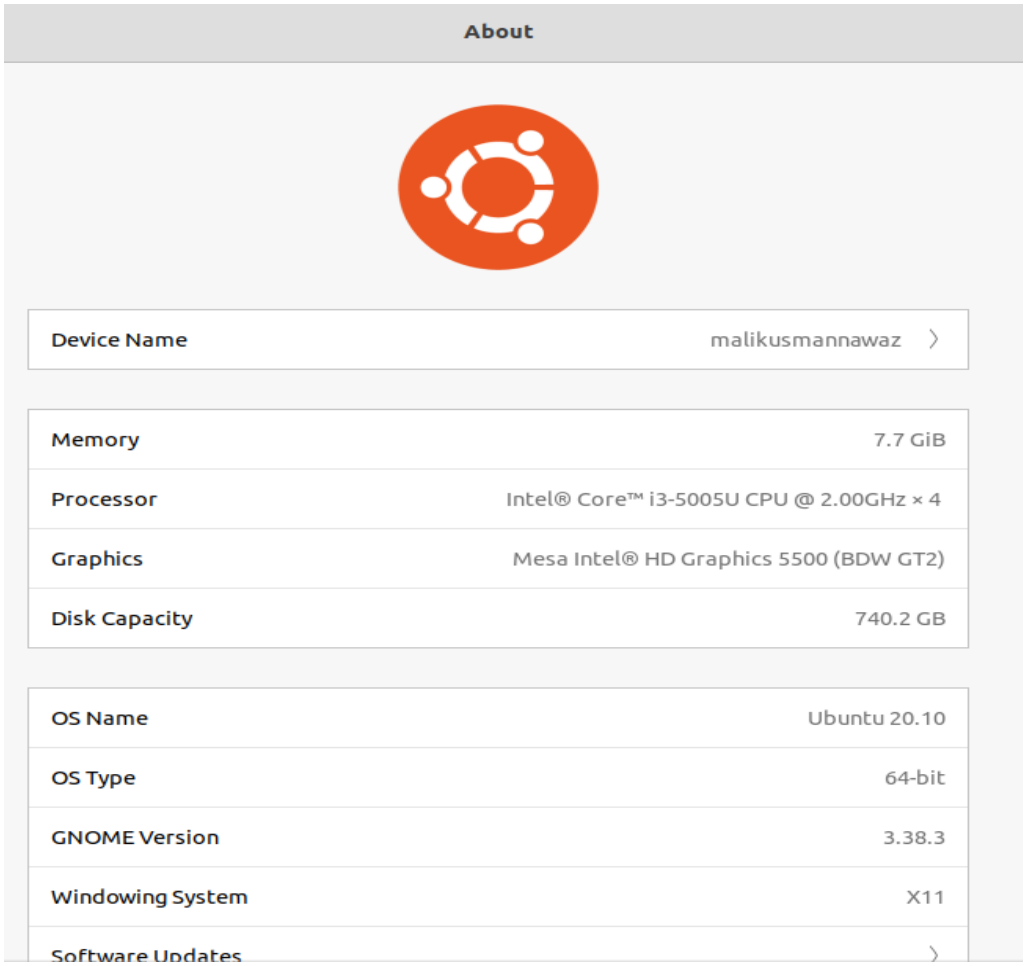
There is a increase of almost 1 seconds.

Implementing parallislism saved our code execution time due to many factors. Parallel programming is a broad concept. We can have many program running on the machine, parallalism mainly helps us to do multitasking. In programming we do parallelism to run code in different threads. When the code the run in different thread it's using different memory block and CPU time. It basically acting to the machine that it's diffferent process. While consuming more resources of machine, we get our desired computation in less time. In this coursework, we have made 5 different threads that are working in parallel to maximize the system resoucre usage and minimize the time consumed. In start we got the difficulties in get the desired outcomes in less time but after struggling hard and giving our maximum time. I got some decrease in the time consumption.

Mentioning one thing that, sometimes parallel comptuing takes longer time then the serial execution. So, it's not mandatory to get the best from the parallel computing all the time. Serial approach sometimes works fine.

CONCLUSIONS:

Due to the low system prossesing and other sepecifcation, it took a lot of time and resource to complete this coursework. I struggled hard due to many factors including COVID-19, no face to face interaction, communication gap and many others. I have performed my best on my own and the result I got were not even satisfactory for me. Due to system specs and other learning diffculities, I waited for the program to get the output for more then 1 hour and 30 mins . My main operating system is Ubuntu 20.10 and other detailed are shown in the image below.



Concluding all of this, I want to mention that for this course we can perfrom better if we had face to face classes and had access to the computer with higher specifications in university all the time during this course.

UWE Gitlab Links:

Serial Code:

OpenMP:

OpenMPI:

Readme:

REFERENCES:

- www.kaspersky.com. 2021. Brute Force Attacks: Password Protection | Kaspersky. [ONLINE] Available at: https://www.kaspersky.com/resource-center/definitions/brute-force-attack. [Accessed 20 April 2021].
- Using OpenMP with C — Research Computing University of Colorado Boulder documentation. 2021. Using OpenMP with C — Research Computing University of Colorado Boulder documentation. [ONLINE] Available at: https://curc.readthedocs.io/en/latest/programming/OpenMP-C.html. [Accessed 25 April 2021].
- Using OpenMP with C — Research Computing University of Colorado Boulder documentation. 2021. Using OpenMP with C — Research Computing University of Colorado Boulder documentation. [ONLINE] Available at: https://curc.readthedocs.io/en/latest/programming/OpenMP-C.html. [Accessed 1 May 2021].