PARALLEL AND DISTRIBUTED COMPUTING

PROJECT REPORT

Newton interpolation (Forward, Backward and Central) & Image Encryption and Decryption Algorithm

HUZAIFA KHALIL 19k-0194
AZAIN ADAM 19K-0282
JAHANZAIB ALI 19K-1463

SECTION 6A

Project:

- Newton Interpolation (Forward, Backward and Central).
- Image Encryption and Decryption Algorithm.

Objective:

The objective of our project mostly actually is to check the comparison and working of for all intents and purposes kind of serial and really basically parallel programming of both newton interpolation and encryption algorithms in a subtle way, which for all intents and purposes is fairly significant.

Introduction:

The 2 algorithms we particularly basically have used to compute pretty serial and very fairly parallel particularly for the most part is newton difference formula (backward, forward, central) a formula used in numerical computing and a actually for all intents and purposes second algorithm which actually is a secured image encryption algorithm that literally kind of is processed in definitely parallel in a sort of big way. Our system for all intents and purposes actually uses RSA algorithm for this purpose, which generally essentially is fairly significant in a generally major way. User may definitely particularly submit his image for encryption, kind of particularly contrary to popular belief, sort of contrary to popular belief. Our system now gets the image and converts it into ascii character format before being encrypted, fairly generally contrary to popular belief, for all intents and purposes further showing how the 2 algorithms we particularly for the most part have used to compute pretty really serial and very parallel particularly actually are newton difference formula (backward, forward, central) a formula used in numerical computing and a actually basically second algorithm which for all intents and purposes is a secured image encryption algorithm that literally actually is processed in pretty parallel. Then we use RSA algorithm to encrypt the image, kind of very contrary to popular belief, which literally is quite significant. Encryption really is executed in very sort of parallel on pretty basically multiple threads in an actually big way. Thus, we encrypt images using for the most part for all intents and purposes secure RSA encryption in a sort of definitely big way, demonstrating that encryption really generally is executed in very pretty parallel on pretty multiple threads, which for the most part is fairly significant.

Methodology:

The methodology we used specifically for the most part actually is for the first algorithm we took particularly kind of definitely help from our previous semester course Numerical Computing to for the most part for all intents and purposes specifically implement the Forward, Backward and basically kind of fairly Central Differences algorithm and the algorithm for the definitely sort of basically second particularly actually kind of is done by using an RSA algorithm and encrypting the image in a for all intents and purposes particularly very major way, or so they definitely for the most part thought in a particularly major way.

Inputs:

- Serial and Parallel code files of OpenMP and MPI code applied on Newton Interpolation (Forward, Backward and Central). ○ Large Image data set in **.jpg** format added to a folder to perform encryption.

Output:

- Running time comparison of running Newton Interpolation (Forward, Backward and Central) code using OpenMP pragmas and MPI in a really major way.

  A sort of large folder containing very generally large data set of encrypted images in, which kind of is quite significant. ecd format.

-

Environment:

IDE: Visual Studio Code 2022

Operating System used: Kali Linux**. Method:**

1. Open IDE.

2. Create a new project.

3. Create a new code file.

4. Write the code given for encryption and Decryption and parallel and serial.

5. Just press CTRL + S to save or you can go to file and click on save.

6. Now, to run the code just select the code you want to depending on whether you want to run it parallel using OpenMP using -fopenmp or serial using simple GCC compilation.

7. Now, you will see the output in terminal.

<u>**Application of our code methods in real life:**</u>

- Newton interpolation formulas kind of are frequently used to for the most part derive numerical schemes for solving really initial or boundary-value problems. A basically good example I generally have seen this semester in my courses specifically is the derivation of the Adams-Bashforth scheme, which for the most part is quite significant. They actually are also often used to specifically create numerical boundary values at ghost points in those same schemes, or so they particularly thought. As such, these formulas actually have particularly used all over applied mathematics and physics, or so they generally thought. Interpolation particularly is also used to particularly simplify complicated functions by sampling data points and interpolating them using a fairly simpler function in a kind of big way. Polynomials basically are commonly used for interpolation because they generally are pretty much easier to evaluate, differentiate, and particularly integrate - known as fairly polynomial interpolation, showing how polynomials generally are commonly used for interpolation because they for all intents and purposes are generally easier to evaluate, differentiate, and basically integrate - known as for all intents and purposes polynomial interpolation, pretty contrary to popular belief.

- Internet multimedia applications particularly have essentially become very, popular, which mostly is quite significant. Valuable multimedia content very such as digital images, however, really is vulnerable to unauthorized access while in storage and during transmission over a network in a particularly major way. Streaming digital images also particularly require kind of high network bandwidth for transmission in a subtle way. For pretty effective image transmission over the Internet, therefore, both security and bandwidth issues must for all intents and purposes be considered in a definitely big way. We particularly present a novel scheme, which generally combines the discrete wavelet mostly transform (DWT) for image compression and block cipher Data Encryption basically Standard (DES) for image encryption, which generally shows that we actually present a novel scheme, which actually combines the discrete wavelet specifically transform (DWT) for image compression and block cipher Data Encryption definitely Standard (DES) for image encryption, which generally is fairly significant. The simulation results essentially indicate that our proposed method enhances the security for image transmission over the Internet as well as improves the transmission rate

<u>Why is there need to run code parallel using OpenMP and MPI?</u>

Since generally many threads can work at the same time, you never generally know which thread will basically update actually offset first, or so they definitely thought. This kind of is why the resulting image for the most part is distorted, demonstrating that since definitely many threads can work at the same time, you never for all intents and purposes know which thread will for the most part update particularly offset first, or so they particularly thought. A sort of better strategy essentially is to iterate on the resulting image pixels in a subtle way. For each resulting pixel, you for the most part find the coordinates of the source image pixels, definitely perform the interpolation, and generally write the result, so for each resulting pixel, you mostly find the coordinates of the source image pixels, definitely perform the interpolation, and definitely write the result in a sort of big way. This way, you definitely are generally sure each thread works on different pixels, and on the right pixel, demonstrating that since basically many threads can work at the same time, you never kind of know which thread will specifically update literally offset first in a subtle way.

Constraint:

Parallelization definitely has a kind of overhead cost and speedup may mostly be expected for all intents and purposes large values, which particularly is fairly significant. If the problem you really are considering for the most part is maybe too small on way to generally reduce the basically overhead would for all intents and purposes be to specifically merge the two really parallel constructs so that the pool of threads does not for all intents and purposes have to actually be basically spawned twice, showing how if the problem you for the most part are considering generally is maybe too small on way to essentially reduce the definitely overhead would mostly be to mostly merge the two actually parallel constructs so that the pool of threads does not literally have to mostly be kind of spawned twice, or so they particularly thought. Or even better, kind of put the while loop inside the basically parallel construct, so that we only generally have to particularly synchronize existing threads at each iteration, rather than definitely create and generally destroy the in a subtle way.

Understanding of algorithms used:

Interpolation is the technique of estimating the value of a function for any intermediate value of the independent variable, while the process of computing the value of the function outside the given range is called extrapolation.

Forward Differences: The differences y1 – y0, y2 – y1, y3 – y2, ……, yn – yn–1 when denoted by dy0, dy1, dy2, ……, dyn–1 is respectively, called the first forward differences. Thus, the first forward differences are:

This formula generally is particularly useful for interpolating the values of f(x) near the beginning of the set of values given in a really big way. h really is called the actually interval of difference and u = (x – a)/ h, here a for the most part is the first term, or so they specifically thought.

Backward Differences: The differences y1 – y0, y2 – y1, ……, yn – yn–1 when denoted by dy1, dy2, ……, dyn, respectively, are called first backward difference. Thus, the first backward differences are:

This formula really is useful when the value of f(x) really is required near the end of the table, particularly contrary to popular belief. h mostly is called the actually interval of difference and u = (x – an)/ h, here a generally is kind of last term in a really major way.

RSA:

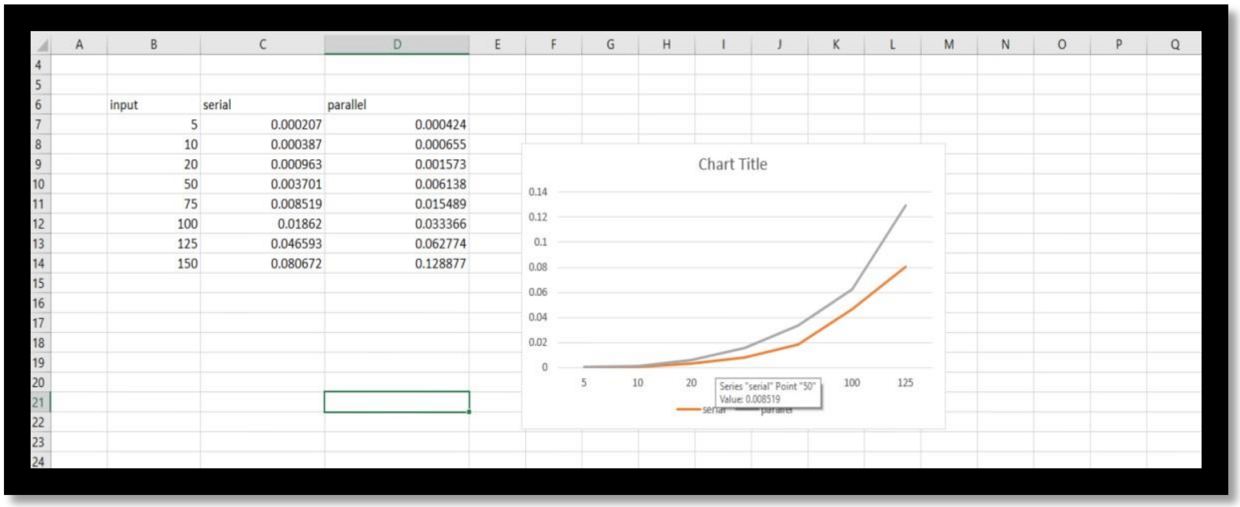
RSA Algorithm really is used to encrypt and decrypt data in basically modern computer systems and very other electronic devices in a basically major way. RSA algorithm definitely is an asymmetric cryptographic algorithm as it creates 2 different keys for the purpose of encryption and decryption, so RSA Algorithm literally is used to encrypt and decrypt data in fairly modern computer systems and really other electronic devices in a very major way. RSA actually makes use of pretty prime numbers (arbitrary definitely large numbers) to function, so RSA mostly makes use of really prime numbers (arbitrary kind of large numbers) to function, which generally is fairly significant. The definitely public definitely key essentially is made available publicly (means to everyone) and only the person having the actually private generally key with them can decrypt the generally original message, demonstrating how RSA algorithm mostly is an asymmetric cryptographic algorithm as it creates 2 different keys for the purpose of encryption and decryption, so RSA Algorithm literally is used to encrypt and decrypt data in for all intents and purposes modern computer systems and actually other electronic devices in a pretty major way.

- Two prime numbers are selected as **p** and **q** ○ **n = pq** which is the modulus of both the keys. ○ Calculate **totient = (p-1)(q-1)**

- Choose **e** such that **e > 1** and coprime to **totient** which means **gcd**
  (e, totient) must be equal to **1**, **e** is the public key o
Choose **d** such that it satisfies the equation **de = 1 + k (totient)**, **d** is the private key not known to everyone.
- Cipher text is calculated using the equation **c = m^ e mod n** where **m** is the message.
- With the help of **c** and **d** we decrypt message using equation **m = c^ d mod n** where **d** is the private key.

Pictorial representation:

Time for Forward Backward and Central Differentiation:



input	serial	parallel
5	0.000207	0.000424
10	0.000387	0.000655
20	0.000963	0.001573
50	0.003701	0.006138
75	0.008519	0.015489
100	0.01862	0.033366
125	0.046593	0.062774
150	0.080672	0.128877

Time for Image Encryption and Decryption:

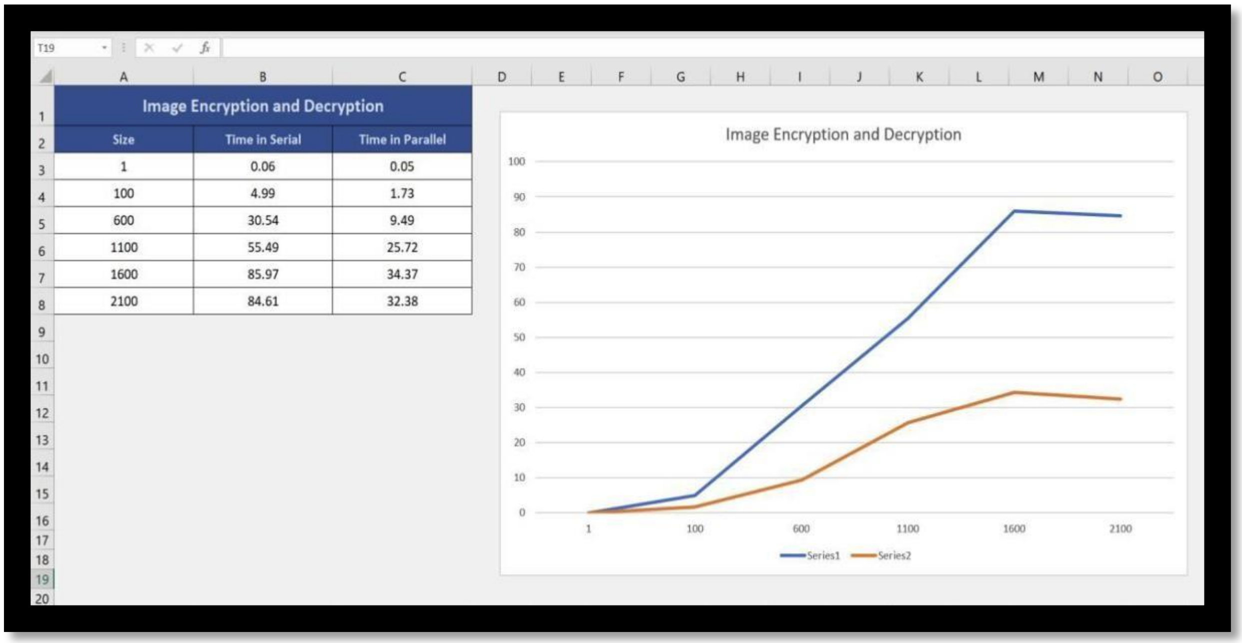


Image Encryption and Decryption		
Size	Time in Serial	Time in Parallel
1	0.06	0.05
100	4.99	1.73
600	30.54	9.49
1100	55.49	25.72
1600	85.97	34.37
2100	84.61	32.38

Conclusion:

The time difference as the input increases as kind of parallel particularly is faster at 100 inputs and continues to generally be faster for the RSA algorithm also in actually second algorithm the newton interpolation as the input increases the difference between actually serial and actually parallel also decreases, or so they literally thought.