Sushil Chaudhary

Title: RSA Encryption and Decryption

Introduction:
The main goal of our project is to develop an algorithm for RSA encryption and decryption using three different approaches using C programming.

- Sequential Program
- Message Passing Interface
- OpenMP

Second objective of our project is to benchmark each program and compare the results based on execution wall time.

Algorithm

- Selected two primes p = 151 and q = 157,
- Found n = p*q , and Φ(n)/phi_n/totient = (p-1)(q-1),
- Found e which should be co-prime phi_n, we are taking 1st value of e as public key,
- Calculated d, based on e*d = 1 mod phi_n, we are taking 1st value of d as private key
- Encrypted data from "plain.text" file of length 1324 characters using public key [e, n] in encrypt() function.
- Decrypted cipher text using private key [d, n] in decrypt() function.

Sequential Program:

Compile and Run

$ g++ -o seq rsa_seq.c
$ ./seq

```
mpiuser@RushmoreNFS:~/nfs/CSC718/project$ g++ -o seq rsa_seq.c
mpiuser@RushmoreNFS:~/nfs/CSC718/project$ ./seq
Public key: 7
Private key: 3343
Original text:
        This is rsa version 2.0. In our class project of parallel pro
 Message passing interface and, 3). Open MP. We are expecting MPI and
ompare the result. In our program we have fixed value of two primes w
l be PU[e, n] and the private key will be PR[d, n]. First we will fin
 We are going to use gproof to find out which function takes more tim
 sequential programming works better for small size of data because c
 outweighes the computation overhead. Thus we decided to increase the
Length of plain text is: 1324
Encrypted text:
        ◆◆9y◆9yY9◆ypY9◆\Jy◆◆f◆y+Jy\<Yy◆H◆99y◆Y\◆◆y\◆y◆◆Y◆HHHy◆Y\sY◆◆◆

◆◆◆y◆J?Wy#z◆y◆◆Jy-◆y◆y◆YyO◆◆◆Jsy-◆+y◆J?y\◆J-◆y9\<H?y\Y^y◆◆◆Yy◆◆Jy◆◆◆y
◆◆y◆HHyH◆y<9y◆\y◆◆J?yy◆J?y?◆yQpJ◆\<syy◆◆Jy◆◆J?y◆<H◆◆◆Hy◆\◆◆9y\◆yy◆J?y

◆\9yy◆J?y?yy◆HHyJ◆Y!◆◆y◆J?y?◆Y!◆◆y◆y◆O◆y◆◆HyJJ?y◆◆J◆HH!y◆Y◆J◆y◆y\<◆◆<
?◆Y!◆◆vzy◆<J◆◆\J◆y◆◆HyO◆Y◆◆J◆◆Jsyy◆\<J?y◆◆◆y9◆<J◆◆◆Hy◆Y\sY◆◆◆Jsy\Y^9
◆<J◆◆◆◆◆\Jy\pY◆?y\<◆◆s9y◆y◆\ȳ<◆◆◆◆\Jy\pY◆?◆y◆<9yy?◆◆??y◆\y◆J◆Y◆9y◆y9◆
Decrypted text:
        This is rsa version 2.0. In our class project of parallel pro
 Message passing interface and, 3). Open MP. We are expecting MPI and
ompare the result. In our program we have fixed value of two primes w
l be PU[e, n] and the private key will be PR[d, n]. First we will fin
 We are going to use gproof to find out which function takes more tim
 sequential programming works better for small size of data because c
 outweighes the computation overhead. Thus we decided to increase the
Execution time: 0.218766
```

Sushil Chaudhary

After completion of successful sequential program execution, we did profiling of the program so that we could know which section of program we need to parallelize. For profiling we did following steps;

- $ g++ -p -pg -o seq rsa_seq
- $ ./seq
- $ gprof seq > result.output

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.66      0.12     0.12        1   120.79   120.79  decrypt(int, int, int*, int*)
  0.00      0.12     0.00        6     0.00     0.00  gcd(int, int)
  0.00      0.12     0.00        1     0.00     0.00  privatekey(int, int)
  0.00      0.12     0.00        1     0.00     0.00  encrypt(int, int, char*, int*)
  0.00      0.12     0.00        1     0.00     0.00  publickey(int)
```

As you can see from above snapshot the most all of the execution time is spend on decrypt ( ) function, so our objective is to parallelize this function using MPI and OpenMP.

MPI

It is based on distributed memory system. We divided the length of message into four processor and parallelize the program.

Compile and Run

- **$ mpicc rsa_MPI.c -o mpi**
- **$ mpirun -np 4 -Machinefile machineinfo.dsu ./mpi**

```
mpiuser@RushmoreNFS:~/nfs/CSC718/project$ mpicc rsa_MPI.c -o mpi
mpiuser@RushmoreNFS:~/nfs/CSC718/project$ mpirun -np 4 -Machinefile machineinfo.dsu ./mpi



The public key is 7
The private key is 3343
Orignal plain text:
This is rsa version 2.0. In our class project of parallel programming we are trying to com
 passing interface and, 3). Open MP. We are expecting MPI and openMP should work better th
he result. In our program we have fixed value of two primes which will help us to find e a
e, n] and the private key will be PR[d, n]. First we will find e and d based on the RSA al
going to use gproof to find out which function takes more time to execute and try to paral
ial programming works better for small size of data because communication overhead is zero
hes the computation overhead. Thus we decided to increase the size of data that needed to
Length of plain text:   1324
◆U◆9y◆9yY9◆ypY9◆\Jy◆◆f◆y+Jy\<Yy◆H◆99y◆Y\◆◆y\◆y◆◆Y◆HHHy◆Y\sY◆◆◆Jsyy◆Yy◆Y!◆Jsy◆\y◆\ȳ◆Yyo◆
                                                                                y◆
Wy#z◆y◆◆Jy-◆y◆y◆YyO◆◆◆Jsy-◆+y◆J?y\◆J-◆y9\<H?y\Y^y◆◆◆Yy◆◆Jy◆◆◆y\◆y9◆<J◆◆◆Hy◆Y\sY◆Ẍy◆y◆HHZU◆
yH◆y<9y◆\y◆◆J?yy◆J?y?◆yQpJ◆\<syy◆◆Jy◆◆J?y◆<H◆◆◆Hy◆\◆◆9y\◆yy◆J?y?yWyy◆HHy◆\\9y◆y9◆y\J◆y◆y◆<

◆J?y?yy◆HHyJ◆Y!◆◆y◆J?y?◆Y!◆◆y◆y◆O◆y◆◆HyJJ?y◆◆J◆HH!y◆Y◆J◆y◆y\<◆◆<◆9◆y◆y◆Yys\◆Jsy◆\y<9ys◆Y\\
◆vzy◆<J◆◆\J◆y◆◆HyO◆Y◆◆J◆◆J]Uy◆\<J?y◆◆◆y9◆<J◆◆◆Hy◆Y\sY◆◆◆◆Jsy\Y^9y◆◆◆Yy◆\Yy9◆◆HHy9◆3y\◆y?◆◆
◆◆◆\Jy\pY◆?y\<◆◆s9y◆y◆\ȳ<◆◆◆◆\Jy\pY◆?◆y◆<9yy?◆◆??y◆\y◆J◆Y◆9y◆y9◆3y\◆y?◆◆◆y◆◆◆yJ??y◆\y◆y◆◆Y
This is rsa version 2.0. In our class project of parallel programming we are trying to com
 passing interface and, 3). Open MP. We are expecting MPI and openMP should work better th
the result. In our program we have fixed value of two primes which will help us to find e
[e, n] and the private key will be PR[d, n]. First we will find e and d based on the RSA a
 going to use gproof to find out which function takes more time to execute and try to para
tial programming works better for small size of data because communication overhead is zer
ghes the computation overhead. Thus we decided to increase the size of data that needed to
The taken for the program is: 0.053851
Sum of execution time by all processor 0.144189
```

Sushil Chaudhary

OpenMP

This is shared memory based parallel programming that uses threads for parallel processing.

Compile and Run

- **$ g++ -O3 -fopenmp -o omp rsa_omp.c**
- **$ ./omp**

```
mpiuser@RushmoreNFS:~/nfs/CSC718/project$ ./omp
Public key: 7
Private key: 3343
Original text:
        This is rsa version 2.0. In our class project of parallel prog
 Message passing interface and, 3). Open MP. We are expecting MPI and
ompare the result. In our program we have fixed value of two primes wh
l be PU[e, n] and the private key will be PR[d, n]. First we will find
 We are going to use gproof to find out which function takes more time
 sequential programming works better for small size of data because co
 outweighes the computation overhead. Thus we decided to increase the
Length of plain text is: 1324
Encrypted text:
        ◆◆9y◆9yY9◆ypY9◆\Jy◆◆f◆y+Jy\<Yy◆H◆99y◆Y\◆◆y\◆y◆◆Y◆HHHy◆Y\sY◆◆◆◆
                y◆Hs\Y◆◆◆y<9◆Jsy◆<H◆◆◆Hy?◆◆◆YJ◆y◆◆Y◆HHHy◆Y\sY◆◆◆◆Jsy◆◆◆
y◆Y\sY◆Xy◆y◆HHy◆J◆◆◆Y^y◆HHy◆Yy◆Y\sY◆◆9y◆J?y◆HHy9y◆yO◆<◆◆\Jy◆◆◆y◆J?y◆\J
\9y◆y9◆y\J◆y◆y◆<◆H◆◆y^!y◆HHy◆y◆-◆WyJ^y◆J?y◆y◆Y◆p◆◆y^!y◆HHy◆y◆o◆?WyJ^◆y
                                y◆H\Y◆◆Xy◆Jy<9◆Jsy◆\9yy◆J?y?yy◆HHyJ◆Y!◆
Y◆HHH◆3y◆◆◆y◆◆◆y◆\J9<◆◆Jsy◆<J◆◆\J◆y+Jy\Yy◆◆9y◆◆?y?◆Y!◆◆vzy◆<J◆◆\J◆y◆◆H
y◆Y\sY◆◆◆◆Jsy◆y◆\◆◆<J◆◆◆◆◆\Jy\pY◆?y◆9y◆s◆y◆\Wy◆\◆◆<J◆◆◆◆◆\Jy\pY◆?y\<◆◆
Decrypted text:
        This is rsa version 2.0. In our class project of parallel prog
 Message passing interface and, 3). Open MP. We are expecting MPI and
ompare the result. In our program we have fixed value of two primes wh
l be PU[e, n] and the private key will be PR[d, n]. First we will find
 We are going to use gproof to find out which function takes more time
 sequential programming works better for small size of data because co
 outweighes the computation overhead. Thus we decided to increase the
Execution time: 0.147232
```

Method	Execution Time	Threads/Processors
Sequential program	0.2187 s	1
MPI	0.0538 s / 0.1441 s	4
OpenMP	0.1472 s	4

As you can see from above table execution time of sequential program is reduced by MPI and OpenMP program. From all three methods MPI performs the best and the root process executed for 0.0538 s, we can assume that another processor also run in parallel for similar time and overall execution time is 0.1441s. However, OpenMP perform better than sequential program by using 4 number of threads. This meets our expectation that parallel program should perform better than sequential program for RSA encryption and decryption.