**PARALLEL AND DISTRIBUTED COMPUTING LAB**

**REPORT**

**NAME:** S Shyam Sundaram

**REG NO:** 19BCE1560

**PROGRAMMING ENVIRONMENT:** MPI

**PROBLEM:** MPI

**DATE:** 20th October, 2021

**HARDWARE CONFIGURATION:**

|  |  |  |  |
| --- | --- | --- | --- |
| CPU NAME | | : | Intel core i5 – 1035G1 @ 1.00 Ghz |
| Number of Sockets: | | : | 1 |
| Cores per Socket | | : | 4 |
| Threads per core | | : | 1 |
| L1 | Cache size | : | 320KB |
| L2 | Cache size | : | 2MB |
| L3 | Cache size (Shared): | | 6MB |
| RAM | | : | 8 GB |

**QUESTION 1**

Write an MPI program that performs matrix multiplication.

**CODE**

#include <mpi.h>

#include <stdio.h>

#define N 12 //can change this

int a[N][N],b[N][N],c[N][N];

MPI\_Status status;

int main(int argc, char\*\*argv)

{

MPI\_Init(&argc,&argv);

int id,numprocs;

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&id);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&numprocs);

if(id==0) //master

{

printf("Name: Shyam Sundaram\nReg num: 19BCE1560\n\n");

int co=1;

for(int i=0;i<N;++i)

for(int j=0;j<N;++j)

{

a[i][j]=co++;

b[i][j]=j+1;

}

int rows=N/(numprocs-1),off=0;

for(int i=1;i<numprocs;++i)

{

MPI\_Send(&off,1,MPI\_INT,i,1,MPI\_COMM\_WORLD);

MPI\_Send(&rows,1,MPI\_INT,i,1,MPI\_COMM\_WORLD);

MPI\_Send(&a[off],rows\*N,MPI\_INT,i,1,MPI\_COMM\_WORLD);

MPI\_Send(&b,N\*N,MPI\_INT,i,1,MPI\_COMM\_WORLD);

off=off+rows;

}

for(int i=1;i<numprocs;++i)

{

MPI\_Recv(&off,1,MPI\_INT,i,2,MPI\_COMM\_WORLD,&status);

MPI\_Recv(&rows, 1, MPI\_INT, i, 2, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&c[off], rows\*N, MPI\_DOUBLE, i, 2, MPI\_COMM\_WORLD, &status);

}

printf("Answer is: \n");

for (int i=0; i<N; i++)

{

for (int j=0; j<N; j++)

printf("%d ", c[i][j]);

printf ("\n");

}

}

else //workers

{

int off,rows;

MPI\_Recv(&off, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&rows, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&a, rows\*N, MPI\_DOUBLE, 0, 1, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&b, N\*N, MPI\_DOUBLE, 0, 1, MPI\_COMM\_WORLD, &status);

printf("%d rows: %d\n",id,rows);

/\* Matrix multiplication \*/

for (int k=0; k<N; k++)

for (int i=0; i<rows; i++)

{

c[i][k] = 0;

for (int j=0; j<N; j++)

c[i][k] = c[i][k] + a[i][j] \* b[j][k];

}

MPI\_Send(&off, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

MPI\_Send(&rows, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

MPI\_Send(&c, rows\*N, MPI\_DOUBLE, 0, 2, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

}

**COMMANDS**

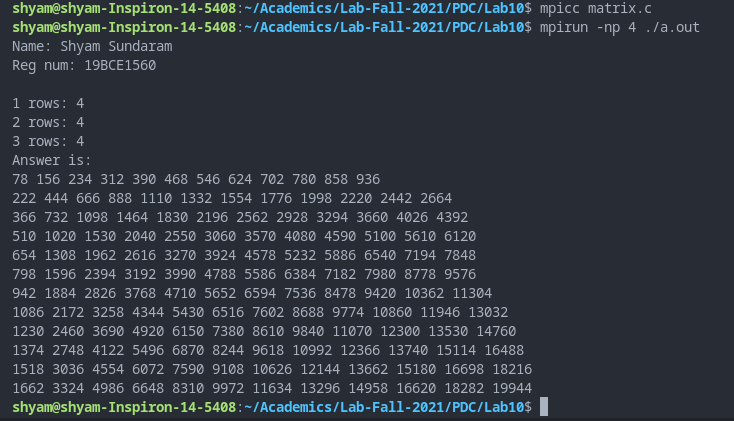
mpicc matrix.c

mpirun -np 4 ./a.out

**OUTPUT**

****

With N=6



With N=12

**OBSERVATION**

Each worker worked on multiplying the respective rows they received from matrix ‘a’ and multiplied with matrix ‘b’ and returned the corresponding resultant row to the master.

**CONCLUSION**

We have found product of matrices using MPI in C.

**QUESTION 2**

Write an MPI program that counts the number of primes in a given range.

**CODE**

#include <mpi.h>

#include <stdio.h>

MPI\_Status status;

int isPrime(int n)

{

for(int i=2;i\*i<=n;++i)

{

if(n%i==0)

return -1;

}

return 1;

}

int main(int argc, char\*\* argv)

{

MPI\_Init(&argc,&argv);

int id,numprocs;

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&id);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&numprocs);

if(id==0) //master

{

int s=1,e=1000; //s is the start of range and e is the end. They are inclusive.

int num=e-s+1;

int pernum=num/(numprocs-1);

int off=s;

int count=0,ret;

int leftover=num%(numprocs-1);

for(int i=1;i<numprocs;++i)

{

MPI\_Send(&pernum,1,MPI\_INT,i,1,MPI\_COMM\_WORLD);

MPI\_Send(&off,1,MPI\_INT,i,1,MPI\_COMM\_WORLD);

off=off+pernum;

}

for(int i=1;i<numprocs;++i)

{

MPI\_Recv(&ret,1,MPI\_INT,i,2,MPI\_COMM\_WORLD,&status);

count+=ret;

}

if(leftover>0)

{

for(int i=off;i<off+leftover;++i)

{

if(i!=1 && isPrime(i)==1)

count++;

}

}

printf("Name: Shyam Sundaram\nReg num: 19BCE1560\n\n");

printf("Total primes: %d\n",count);

}

else

{

int n,s,count=0;

MPI\_Recv(&n,1,MPI\_INT,0,1,MPI\_COMM\_WORLD,&status);

MPI\_Recv(&s,1,MPI\_INT,0,1,MPI\_COMM\_WORLD,&status);

for(int i=s;i<s+n;++i)

{

//printf("ID %d: %d %d\n ",id,i,isPrime(i));

if(i!=1 && isPrime(i)==1)

count++;

}

MPI\_Send(&count, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

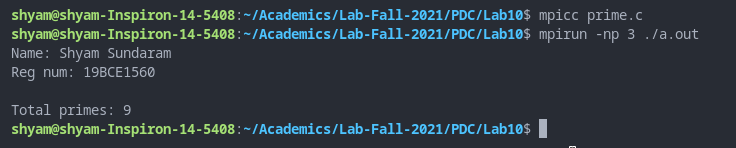
}

**COMMANDS**

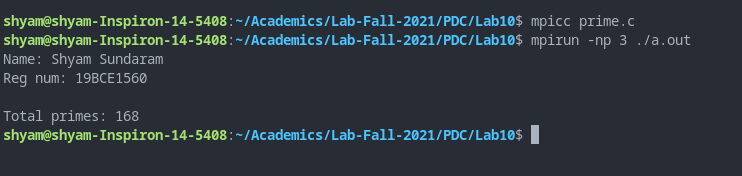
mpicc prime.c

mpirun -np 4 ./a.out

**OUTPUT**

****

Between 2 and 23



Between 1 and 1000

**OBSERVATION**

Each worker worked in a specific sub-range given to them and returned number of primes to the master. The master in turns sums the counts and also counts the left-over primes.

**CONCLUSION**

We have found total number of primes in a given range using MPI in C.