***Протокол***

*Лабораторна робота №8*

*Виконав: студент гр. ІО – 72*

*Паскалов П.Г.*

*ПРО*

*MB, MZ*

Структура ПОС

*5*

*4*

*MA, MС*

*3*

*2*

*1*

***Математична операція:***

MA=MB(MC\*MZ)

Мова: С++(MPI)

***Етап 1.*** *Розробка паралельного алгоритму*

MAH=MB(MC\*MZH)

***Етап 2.*** *Розробка алгоритмів процесорів****.***

***Task1***

1. Прийняти від Т2 MB, MZH
2. Прийняти від Т2 МС
3. Обчислити: MAH=MB(MC\*MZH)
4. Передати MAH в Т2

Task2

1. Ввести МB, MZ
2. Передати в T1 MB, MZH
3. Передати в T3 MB, MZ3H
4. Прийняти МCвід Т3
5. Передати МCв Т1
6. Обчислити: MAH=MB(MC\*MZH)
7. Прийняти МАН від Т1
8. Передати МА2Н в Т3

Task3

1. Прийняти МB, МZ3H від Т2
2. Передати МB, МZ2H в Т4
3. Прийняти від Т4 МC
4. Передати в Т2 MC
5. Обчислити: MAH=MB(MC\*MZH)
6. Прийняти МА2Н від Т2
7. Передати МА3Н  в Т4

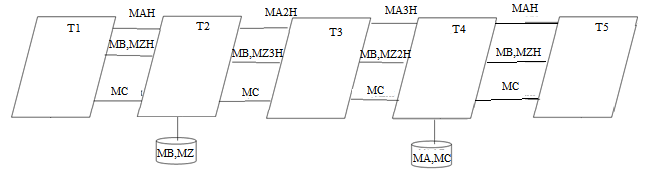
Task4

1. Ввести MC
2. Прийняти МB, МZ2H від Т3
3. Передати МB, МZH в Т5
4. Передати в Т5 MC
5. Передати в Т3 MC
6. Обчислити: MAH=MB(MC\*MZH)
7. Прийняти МА3H від Т3
8. Прийняти МАН від Т5
9. Вивести МА

Task5

1. Прийняти від Т4 MB, MZH
2. Прийняти від Т4 МС
3. Обчислити: MAH=MB(MC\*MZH)
4. Передати MAH в Т4

***Stage 3.*** *Devising interaction processes.*



***Stage 4. Listing***

#include "mpi.h"

#include "math.h"

#include <stdio.h>

#include <tchar.h>

#include "stdio.h"

#include "windows.h"

#include <iostream>

#pragma comment(lib, "mpich.lib")

using namespace std;

#define N 10

#define P 5

const int H = N / P;

void initalisingM(int \*t)

{

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

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

t[i+N\*j] = 0;

}

void InputMatrix(int\* matr)

{

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

matr[i] = 1;

}

void OutputMatrix(int \*matr)

{

int count = N;

for(int i =0 ; i < count\*count; i++)

{

printf("%d ", matr[i]);

if(((i+1) % count) == 0 && (i != 0))

printf("\n");

}

printf("\n");

}

void Calc(int \*MA, int \*MB, int \*MC, int \*MZ)

{

int \*MA1 = new int[N\*N];

initalisingM(MA1);

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

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

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

MA1[j+N\*i] += MZ[k+N\*i] \*MC[j+N\*k];

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

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

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

MA[j+N\*i] += MA1[j+N\*i] \*MB[j+N\*k];

}

int main(int args, char \* argv[])

{

int id;

int p;

MPI\_Status status;

MPI\_Init(&args,&argv);

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

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

switch(id)

{

case 0:

{

cout<<" Task"<<id<<" start"<<endl;

int \*MA = new int[N\*N];

int \*MC = new int[N\*N];

int \*MB = new int[N\*N];

int \*MZ = new int[N\*N];

initalisingM(MA);

initalisingM(MC);

initalisingM(MZ);

initalisingM(MB);

MPI\_Recv(MB,N\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MZ,H\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MC,N\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD,&status);

Calc(MA, MB, MC, MZ);

MPI\_Send((void\*) MA,H\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD);

cout<<" Task"<<id<<" finish"<<endl;

}

break;

case 1:

{

cout<<" Task"<<id<<" start"<<endl;

int \*MA = new int[N\*N];

int \*MA1 = new int[N\*N];

int \*MC = new int[N\*N];

int \*MB = new int[N\*N];

int \*MZ = new int[N\*N];

int \*MZ\_O = new int[N\*N];

initalisingM(MA);

initalisingM(MA1);

initalisingM(MC);

initalisingM(MZ);

initalisingM(MB);

InputMatrix(MB);

InputMatrix(MZ);

int k = 0;

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

{

MZ\_O[k] = MZ[i];

k++;

}

MPI\_Send((void\*) MB,N\*N,MPI\_INT,0,0,MPI\_COMM\_WORLD);

MPI\_Send((void\*) MZ\_O,H\*N,MPI\_INT,0,0,MPI\_COMM\_WORLD);

k = 0;

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

{

MZ\_O[k] = MZ[i];

k++;

}

MPI\_Send((void\*) MB,N\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD);

MPI\_Send((void\*) MZ\_O,3\*H\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD);

MPI\_Recv(MC,N\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD,&status);

MPI\_Send((void\*) MC,N\*N,MPI\_INT,0,0,MPI\_COMM\_WORLD);

Calc(MA, MB, MC, MZ);

MPI\_Recv(MA1,H\*N,MPI\_INT,0,0,MPI\_COMM\_WORLD,&status);

k = 0;

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

{

MA[i] = MA1[k];

k++;

}

MPI\_Send((void\*) MA,2\*H\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD);

cout<<" Task"<<id<<" finish"<<endl;

}

break;

case 2:

{

cout<<" Task"<<id<<" start"<<endl;

int \*MA = new int[N\*N];

int \*MA1 = new int[N\*N];

int \*MC = new int[N\*N];

int \*MB = new int[N\*N];

int \*MZ = new int[N\*N];

int \*MZ\_O = new int[N\*N];

initalisingM(MA);

initalisingM(MA1);

initalisingM(MC);

initalisingM(MZ);

initalisingM(MB);

MPI\_Recv(MB,N\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MZ,3\*H\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD,&status);

int k = 0;

for(int i = H\*N ; i < 3\*H\*N ; i++)

{

MZ\_O[k] = MZ[i];

k++;

}

MPI\_Send((void\*) MB,N\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD);

MPI\_Send((void\*) MZ\_O,2\*H\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD);

MPI\_Recv(MC, N\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD,&status);

MPI\_Send((void\*) MC,N\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD);

Calc(MA, MB, MC, MZ);

MPI\_Recv(MA1,2\*H\*N,MPI\_INT,1,0,MPI\_COMM\_WORLD,&status);

k = 0;

for(int i = H\*N ; i < 3\*H\*N ; i++)

{

MA[i] = MA1[k];

k++;

}

MPI\_Send((void\*) MA,3\*H\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD);

cout<<" Task"<<id<<" finish"<<endl;

}

break;

case 3:

{

cout<<" Task"<<id<<" start"<<endl;

int \*MA = new int[N\*N];

int \*MA1 = new int[N\*N];

int \*MC = new int[N\*N];

int \*MB = new int[N\*N];

int \*MZ = new int[N\*N];

int \*MZ\_O = new int[N\*N];

initalisingM(MA);

initalisingM(MA1);

initalisingM(MC);

initalisingM(MZ);

initalisingM(MB);

InputMatrix(MC);

MPI\_Recv(MB,N\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MZ,2\*H\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD,&status);

MPI\_Send((void\*) MC,N\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD);

int k = 0;

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

{

MZ\_O[k] = MZ[i];

k++;

}

MPI\_Send((void\*) MB,N\*N,MPI\_INT,4,0,MPI\_COMM\_WORLD);

MPI\_Send((void\*) MZ\_O,H\*N,MPI\_INT,4,0,MPI\_COMM\_WORLD);

MPI\_Send((void\*) MC,N\*N,MPI\_INT,4,0,MPI\_COMM\_WORLD);

Calc(MA, MB, MC, MZ);

MPI\_Recv(MA1,3\*H\*N,MPI\_INT,2,0,MPI\_COMM\_WORLD,&status);

k = 0;

for(int i = H\*N ; i < 4\*H\*N ; i++)

{

MA[i] = MA1[k];

k++;

}

MPI\_Recv(MA1,H\*N,MPI\_INT,4,0,MPI\_COMM\_WORLD,&status);

k = 0;

for(int i = 4\*H\*N ; i < 5\*H\*N ; i++)

{

MA[i] = MA1[k];

k++;

}

OutputMatrix(MA);

cout<<" Task"<<id<<" finish"<<endl;

}

break;

case 4:

{

cout<<" Task"<<id<<" start"<<endl;

int \*MA = new int[N\*N];

int \*MC = new int[N\*N];

int \*MB = new int[N\*N];

int \*MZ = new int[N\*N];

initalisingM(MA);

initalisingM(MC);

initalisingM(MZ);

initalisingM(MB);

MPI\_Recv(MB,N\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MZ,H\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MC, N\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD,&status);

Calc(MA, MB, MC, MZ);

MPI\_Send((void\*) MA,H\*N,MPI\_INT,3,0,MPI\_COMM\_WORLD);

cout<<" Task"<<id<<" finish"<<endl;

}

break;

}

MPI\_Finalize();

return 0;

}