

Міністерство освіти і науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 4

з курсу “ Паралельне програмування-2. Програмування для паралельних систем”

на тему:

“ Бібліотека МРІ”

Виконав: студент ІІI курсу групи ІП – 42

Водотієць Денис Ігорович

Київ – 2017

**Технічне завдання**

Структура ПКС:



Математична задача: MA = d \* MB + (B \* C) \* (MO \* MK).

Мова та бібліотека програмування: C++, MPI.

**Виконання**

Етап 1. Розробка паралельного математичного алгоритму

1. ai = BH \* CH, i = 1..P;
2. a = a + ai, i = 1..P;

3. MAH = d \* MBH + a \* (MOH \* MK);

Де H = N / P, де P – кількість процесів,

B*H* - H елементів вектора B,

C*H* - H елементів вектора C,

MA*H* - H рядків матриці МA,

MB*H* - H рядків матриці МB,

MO*H* - H рядків матриці МO,

ЗР: a, d, MK.

Етап 2. Розробка алгоритму потоків

Потік Т1:

1. Ввести MK, d;
2. Відправити MK, d в Т2;
3. Отримати MB, C, MO, B з Т2;
4. Рахувати: a1 = CH \* BH;
5. Отримати a2 з Т2;
6. Рахувати a = a1 + a2;
7. Відправити а в Т2;
8. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
9. Отримати MА6H з Т2;
10. Вивести MА.

Потік Т2:

1. Отримати MB, C, MO, B з Т3;
2. Отримати MK, d з Т1;
3. Відправити MK, d, MB, C, MO, B в T5;
4. Відправити MK, d в Т3;
5. Рахувати: a2 = CH \* BH;
6. Отримати a3 з Т3 та a5 з Т5;
7. Рахувати a2 = a2 + a3 + a5;
8. Відправити а2 в Т1;
9. Отримати a з Т1;
10. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
11. Отримати MАH з Т5 та MA4H з Т3;
12. Відправити MA6H в Т1.

Потік Т3:

1. Отримати MB, C з Т4;
2. Отримати MO, B з Т6;
3. Відправити MB, C, MO, B в Т2;
4. Отримати MK, d з Т2;
5. Відправити MK, d, MO, B в Т4;
6. Відправити MK, d, MB, C в Т6;
7. Рахувати: a3 = CH \* BH;
8. Отримати a4 з Т4 та a6 з Т6;
9. Рахувати a3 = a3 + a4 + a6;
10. Відправити а3 в Т2;
11. Отримати a з Т2;
12. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
13. Отримати MАH з Т4 та MA2H з Т6;
14. Відправити MA4H в Т2.

Потік Т4:

1. Ввести MB, C.
2. Відправити MB, C в Т3;
3. Отримати MK, d, MO, B з Т3;
4. Рахувати: a4 = CH \* BH;
5. Відправити а4 в Т3;
6. Отримати a з Т3;
7. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
8. Відправити MAH в Т3.

Потік Т5:

1. Отримати MK, d, MB, C, MO, B з Т2;
2. Рахувати: a5 = CH \* BH;
3. Відправити а5 в Т2;
4. Отримати a з Т2;
5. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
6. Відправити MAH в Т2.

Потік Т6:

1. Отримати MK, d, MB, C з Т3;
2. Отримати MO, B з Т7;
3. Відправити MO, B в Т3;
4. Відправити MK, d, MB, C в Т7;
5. Рахувати: a6 = CH \* BH;
6. Отримати a7 з Т7;
7. Рахувати: a6 = a6 + a7;
8. Відправити а6 в Т3;
9. Отримати a з Т3;
10. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
11. Отримати MAH з Т7;
12. Відправити MA2H в Т3.

Потік Т7:

1. Ввести MO, B;
2. Відправити MO, B в Т6;
3. Отримати MK, d, MB, C з Т6;
4. Рахувати: a7 = CH \* BH;
5. Відправити а7 в Т6;
6. Отримати a з Т6;
7. Рахувати: MAH = d \* MBH + a \* (MOH \* MK);
8. Відправити MAH в Т6.

Етап 3. Розробка схеми взаємодії потоків



Етап 4. Розробка програми

#include <iostream>

#include <limits.h>

#include "mpi.h"

/////////////////////////////////////////////////////////////

///// Lab 4. MPI /////

///// MA = d \* MB + (B \* C) \* (MO \* MK) /////

///// Denys Vodotiiets /////

///// IP-42 /////

/////////////////////////////////////////////////////////////

//#pragma comment(linker, "/STACK:1024")

using namespace std;

const int N = 100;

const int T1 = 0;

const int T2 = 1;

const int T3 = 2;

const int T4 = 3;

const int T5 = 4;

const int T6 = 5;

const int T7 = 6;

void inputMatrix(int matrix[N][N]);

void outputMatrix(int matrix[N][N]);

void inputVector(int vector[N]);

void ckeckSize(int r, int s);

void sendMatrixPart(int matrix[N][N], int start, int end, int dest, int tag);

void recvMatrixPart(int matrix[N][N], int start, int end, int source, int tag, MPI\_Status status);

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

{

MPI\_Init(&argc, &argv);

int rank, size;

int C[N], B[N];

int MA[N][N], MK[N][N], MB[N][N], MO[N][N];

int a, d;

MPI\_Status status;

int msgTag = 0;

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

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

ckeckSize(rank, size);

int P = 7;

int H = N / P;

//input T1

if (rank == T1)

{

inputMatrix(MK);

d = 1;

//send to T2

MPI\_Send(MK, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&d, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

//receive from T2

MPI\_Recv(MB, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(C, N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MO, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(B, N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

}

//input T2

else if (rank == T2)

{

//receive from T1

MPI\_Recv(MK, N \* N, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&d, 1, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD, &status);

//send to T5

MPI\_Send(MK, N \* N, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&d, 1, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

//send to T3

MPI\_Send(MK, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&d, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

//receive from T3

MPI\_Recv(MB, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(C, N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MO, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(B, N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

//send to T5

MPI\_Send(MB, N \* N, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(C, N, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(MO, N \* N, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(B, N, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

//send to T1

MPI\_Send(MB, N \* N, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(C, N, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(MO, N \* N, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(B, N, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD);

}

//input T3

else if (rank == T3)

{

//receive from T2

MPI\_Recv(MK, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&d, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

//send to T4

MPI\_Send(MK, N \* N, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&d, 1, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD);

//send to T6

MPI\_Send(MK, N \* N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&d, 1, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

//receive from T4

MPI\_Recv(MB, N \* N, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(C, N, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD, &status);

//send to T2

MPI\_Send(MB, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(C, N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

//send to T6

MPI\_Send(MB, N \* N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(C, N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

//receive from T6

MPI\_Recv(MO, N \* N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(B, N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

//send to T2

MPI\_Send(MO, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(B, N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

//send to T4

MPI\_Send(MO, N \* N, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(B, N, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD);

}

//input T4

else if (rank == T4)

{

inputMatrix(MB);

inputVector(C);

//send to T3

MPI\_Send(MB, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(C, N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

//receive from T3

MPI\_Recv(MK, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&d, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MO, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(B, N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

}

//input T5

else if (rank == T5)

{

//receive from T2

MPI\_Recv(MK, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&d, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MB, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(C, N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MO, N \* N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(B, N, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

}

//input T6

else if (rank == T6)

{

//receive from T7

MPI\_Recv(MO, N \* N, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(B, N, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD, &status);

//send to T3

MPI\_Send(MO, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(B, N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

//receive from T3

MPI\_Recv(MK, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&d, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MB, N \* N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(C, N, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

//send to T7

MPI\_Send(MK, N \* N, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&d, 1, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(MB, N \* N, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(C, N, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD);

}

else if (rank == T7)

{

inputMatrix(MO);

inputVector(B);

//send to T6

MPI\_Send(MO, N \* N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(B, N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

//receive from T6

MPI\_Recv(MK, N \* N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&d, 1, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MB, N \* N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(C, N, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

}

int startIndex = rank \* H;

int endIndex = (rank + 1) \* H;

if (rank == P - 1)

{

endIndex = N;

}

//calc B \* C

int a\_i = 0;

for (int i = startIndex; i < endIndex; i++)

{

a\_i += B[i] \* C[i];

}

if (rank == T1)

{

int a2;

MPI\_Recv(&a2, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

a = a\_i + a2;

MPI\_Send(&a, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

}

else if (rank == T2)

{

int a3, a5;

MPI\_Recv(&a3, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&a5, 1, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD, &status);

a\_i += a3 + a5;

MPI\_Send(&a\_i, 1, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD);

MPI\_Recv(&a, 1, MPI\_INT, T1, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Send(&a, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&a, 1, MPI\_INT, T5, msgTag, MPI\_COMM\_WORLD);

}

else if (rank == T3)

{

int a4, a6;

MPI\_Recv(&a4, 1, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&a6, 1, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

a\_i += a4 + a6;

MPI\_Send(&a\_i, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

MPI\_Recv(&a, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Send(&a, 1, MPI\_INT, T4, msgTag, MPI\_COMM\_WORLD);

MPI\_Send(&a, 1, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

}

else if (rank == T4)

{

MPI\_Send(&a\_i, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

MPI\_Recv(&a, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

}

else if (rank == T5)

{

MPI\_Send(&a\_i, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD);

MPI\_Recv(&a, 1, MPI\_INT, T2, msgTag, MPI\_COMM\_WORLD, &status);

}

else if (rank == T6)

{

int a7;

MPI\_Recv(&a7, 1, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD, &status);

a\_i += a7;

MPI\_Send(&a\_i, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD);

MPI\_Recv(&a, 1, MPI\_INT, T3, msgTag, MPI\_COMM\_WORLD, &status);

MPI\_Send(&a, 1, MPI\_INT, T7, msgTag, MPI\_COMM\_WORLD);

}

else if (rank == T7)

{

MPI\_Send(&a\_i, 1, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD);

MPI\_Recv(&a, 1, MPI\_INT, T6, msgTag, MPI\_COMM\_WORLD, &status);

}

//Calc MA\_H

for (int i = startIndex; i < endIndex; i++)

{

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

{

int sum = 0;

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

{

sum += MO[i][k] \* MK[k][j];

}

MA[i][j] = MB[i][j] \* d + a \* sum;

}

}

if (rank == T1)

{

recvMatrixPart(MA, H, N, T2, msgTag, status);

cout << "Result MA ";

if (N > 10)

{

cout << "is too large";

}

else

{

cout << ":" << endl;

outputMatrix(MA);

}

}

else if (rank == T2)

{

recvMatrixPart(MA, T3 \* H, N, T3, msgTag, status);

recvMatrixPart(MA, T5 \* H, T6 \* H, T5, msgTag, status);

sendMatrixPart(MA, T2 \* H, N, T1, msgTag);

}

else if (rank == T3)

{

recvMatrixPart(MA, T6 \* H, N, T6, msgTag, status);

recvMatrixPart(MA, T4 \* H, T5 \* H, T4, msgTag, status);

sendMatrixPart(MA, T3 \* H, N, T2, msgTag);

}

else if (rank == T4)

{

sendMatrixPart(MA, T4 \* H, N, T3, msgTag);

}

else if (rank == T5)

{

sendMatrixPart(MA, T5 \* H, T6 \* H, T2, msgTag);

}

else if (rank == T6)

{

recvMatrixPart(MA, T7 \* H, N, T7, msgTag, status);

sendMatrixPart(MA, T6 \* H, N, T3, msgTag);

}

else if (rank == T7)

{

sendMatrixPart(MA, T7 \* H, N, T6, msgTag);

}

MPI\_Finalize();

return 0;

}

void inputMatrix(int matrix[N][N])

{

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

{

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

{

matrix[i][j] = 1;

}

}

}

void outputMatrix(int matrix[N][N])

{

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

{

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

{

printf("%10d ", matrix[i][j]);

}

cout << endl;

}

cout << endl;

}

void inputVector(int vector[N])

{

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

{

vector[i] = 1;

}

}

void ckeckSize(int rank, int size)

{

if (size < 7 && N < 7)

{

if (rank == 0)

{

cout << "For correct program work the count of threads must be more than 7." <<

"Please make sure that you input correct data!" << endl <<

"Your threads' size is " << size << endl;

}

MPI\_Finalize();

exit(-1);

}

}

void sendMatrixPart(int matrix[N][N], int start, int end, int dest, int tag)

{

for (int i = start; i < end; i++)

{

MPI\_Send(matrix[i], N, MPI\_INT, dest, tag, MPI\_COMM\_WORLD);

}

}

void recvMatrixPart(int matrix[N][N], int start, int end, int source, int tag, MPI\_Status status)

{

for (int i = start; i < end; i++)

{

MPI\_Recv(matrix[i], N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, &status);

}

}