

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

Кафтанатій Богдан Сергійович

Київ – 2017

**Технічне завдання**

Структура ПКС:



Математична задача: A = max(Z) \* E + d \* B \* (MO \* MK).

Мова та бібліотека програмування: C++, MPI.

**Виконання**

Етап 1. Розробка паралельного математичного алгоритму

1. a*i* = max(Z*H*), i = 1..P;

2. a = max(a, a*i*);

3. A*H* = a \* E*H* + d \* B \* (MK*H* \* MO).

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

A*H* - H елементів вектора А,

MK*H* - H рядків матриці МO.

ОР: a, d, B, MO.

Етап 2. Розробка алгоритму потоків

Потік Т1:

1. Ввести Z, E;
2. Відправити Z, E в Т2 та Т4;
3. Отримати В, MK, d, MO з Т2;
4. Рахувати: a1 = max(Z*H*);
5. Рахувати: а = max(a, a*1*);
6. Рахувати: A*H* = a \* E*H* + d \* B \* (MK*H* \* MO);
7. Отримати А3H з Т4;
8. Отримати А2H з Т2;
9. Вивести А.

Потік Т2:

1. Отримати Z, E з Т1;
2. Відправити Z, E в Т3;
3. Отримати В, MK, d, MO з Т3;
4. Відправити В, MK, d, MO в Т1;
5. Рахувати: a2 = max(Z*H*);
6. Рахувати: а = max(a, a*2*);
7. Рахувати: A*H* = a \* E*H* + d \* B \* (MK*H* \* MO);
8. Отримати АH з Т3;
9. Відправити А2H в Т1;

Потік Т3:

1. Ввести B, MK;
2. Отримати Z, E з Т2;
3. Відправити В, MK в Т2 та Т6;
4. Отримати d, MO з Т6;
5. Відправити d, MO в Т2;
6. Рахувати: a3 = max(Z*H*);
7. Рахувати: а = max(a, a*3*);
8. Рахувати: A*H* = a \* E*H* + d \* B \* (MK*H* \* MO);
9. Відправити АH в Т2;

Потік Т4:

1. Отримати Z, E з Т1;
2. Відправити Z, E в Т5;
3. Отримати В, MK, d, MO з Т5;
4. Рахувати: a4 = max(Z*H*);
5. Рахувати: а = max(a, a*4*);
6. Рахувати: A*H* = a \* E*H* + d \* B \* (MK*H* \* MO);
7. Отримати А2Н з Т5;
8. Відправити А3H в Т1;

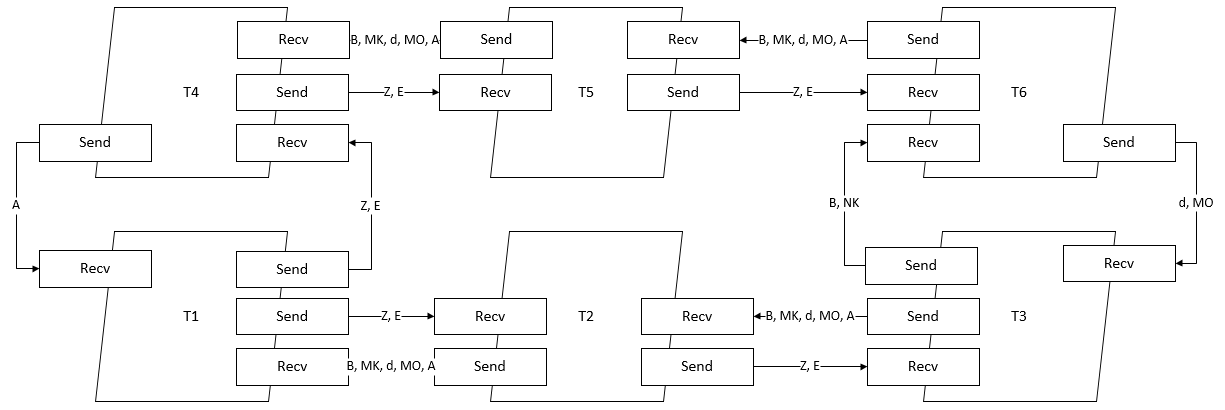
Потік Т5:

1. Отримати Z, E з Т3;
2. Відправити Z, E в Т5;
3. Отримати В, MK, d, MO з Т5;
4. Відправити В, MK, d, MO в Т3;
5. Рахувати: a5 = max(Z*H*);
6. Рахувати: а = max(a, a*5*);
7. Рахувати: A*H* = a \* E*H* + d \* B \* (MK*H* \* MO);
8. Отримати АH з Т6;
9. Відправити А2H в Т4;

Потік Т6:

1. Ввести d, MO;
2. Отримати Z, E з Т5;
3. Отримати В, MK з Т3;
4. Відправити В, MK в Т5;
5. Відправити d, MO в Т3 та Т5;
6. Рахувати: a6 = max(Z*H*);
7. Рахувати: а = max(a, a*6*);
8. Рахувати: A*H* = a \* E*H* + d \* B \* (MK*H* \* MO);
9. Відправити АH в Т5;

Етап 3. Розробка схеми взаємодії потоків



Етап 4. Розробка програми

#include <iostream>

#include <limits.h>

#include "mpi.h"

using namespace std;

const int N = 600;

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 inputVector(int vector[N])

{

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

{

vector[i] = 1;

}

}

void outputVector(int vector[N])

{

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

{

cout << vector[i] << " ";

}

cout << endl;

}

int maxElement(int vector[N], int from, int to)

{

int result = INT\_MIN;

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

{

if (vector[i] > result)

result = vector[i];

}

return result;

}

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

{

MPI\_Init(&argc, &argv);

int rank, size;

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

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

int P = size;

int H = N / P;

int A[N], B[N], E[N], Z[N];

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

int d, a;

MPI\_Status status;

if (rank == 0)

{

inputVector(Z);

inputVector(E);

MPI\_Send(Z + H, H \* 2, MPI\_INT, 1, 0, MPI\_COMM\_WORLD);

MPI\_Send(E + H, H \* 2, MPI\_INT, 1, 0, MPI\_COMM\_WORLD);

MPI\_Send(Z + (H \* 3), N - (H \* 3), MPI\_INT, 3, 0, MPI\_COMM\_WORLD);

MPI\_Send(E + (H \* 3), N - (H \* 3), MPI\_INT, 3, 0, MPI\_COMM\_WORLD);

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

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

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

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

}

else if (rank == 1)

{

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

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

MPI\_Send(Z + 2 \* H, H, MPI\_INT, 2, 0, MPI\_COMM\_WORLD);

MPI\_Send(E + 2 \* H, H, MPI\_INT, 2, 0, MPI\_COMM\_WORLD);

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

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

MPI\_Send(B, N, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

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

MPI\_Recv(&d, 1, MPI\_INT, 2, 0, MPI\_COMM\_WORLD, &status);

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

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

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

}

else if (rank == 2)

{

inputVector(B);

inputMatrix(MK);

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

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

MPI\_Send(B, N, MPI\_INT, 1, 0, MPI\_COMM\_WORLD);

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

MPI\_Send(B, N, MPI\_INT, 5, 0, MPI\_COMM\_WORLD);

MPI\_Send(MK, N \* N, MPI\_INT, 5, 0, MPI\_COMM\_WORLD);

MPI\_Recv(&d, 1, MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MO, N\*N, MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

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

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

}

else if (rank == 3) {

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

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

MPI\_Send(Z + 4 \* H, N - (H \* 4), MPI\_INT, 4, 0, MPI\_COMM\_WORLD);

MPI\_Send(E + 4 \* H, N - (H \* 4), MPI\_INT, 4, 0, MPI\_COMM\_WORLD);

MPI\_Recv(B, N, MPI\_INT, 4, 0, MPI\_COMM\_WORLD, &status);

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

MPI\_Recv(&d, 1, MPI\_INT, 4, 0, MPI\_COMM\_WORLD, &status);

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

}

else if (rank == 4)

{

MPI\_Recv(Z + 4 \* H, N - (H \* 4), MPI\_INT, 3, 0, MPI\_COMM\_WORLD, &status);

MPI\_Recv(E + 4 \* H, N - (H \* 4), MPI\_INT, 3, 0, MPI\_COMM\_WORLD, &status);

MPI\_Send(Z + 5 \* H, N - (H \* 5), MPI\_INT, 5, 0, MPI\_COMM\_WORLD);

MPI\_Send(E + 5 \* H, N - (H \* 5), MPI\_INT, 5, 0, MPI\_COMM\_WORLD);

MPI\_Recv(B, N, MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MK, N \* N, MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

MPI\_Send(B, N, MPI\_INT, 3, 0, MPI\_COMM\_WORLD);

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

MPI\_Recv(&d, 1, MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

MPI\_Recv(MO, N\*N, MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

MPI\_Send(&d, 1, MPI\_INT, 3, 0, MPI\_COMM\_WORLD);

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

}

else if (rank == 5)

{

d = 1;

inputMatrix(MO);

MPI\_Recv(Z + 5 \* H, N - (H \* 5), MPI\_INT, 4, 0, MPI\_COMM\_WORLD, &status);

MPI\_Recv(E + 5 \* H, N - (H \* 5), MPI\_INT, 4, 0, MPI\_COMM\_WORLD, &status);

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

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

MPI\_Send(B, N, MPI\_INT, 4, 0, MPI\_COMM\_WORLD);

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

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

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

MPI\_Send(&d, 1, MPI\_INT, 4, 0, MPI\_COMM\_WORLD);

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

}

int startIndex = rank\*H;

int endIndex = rank == P - 1 ? N : (rank + 1) \* H;

int a\_i = maxElement(Z, startIndex, endIndex);

MPI\_Allreduce(&a\_i, &a, 1, MPI\_INT, MPI\_MAX, MPI\_COMM\_WORLD);

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

{

int tempV[N];

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

{

int M = 0;

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

{

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

}

tempV[j] = M;

}

int resultNumber = 0;

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

{

resultNumber += tempV[i] \* B[i];

}

A[i] = a \* E[i] + d \* resultNumber;

}

if (rank == 5)

{

MPI\_Send(A + (H \* 5), N - (H \* 5), MPI\_INT, 4, 0, MPI\_COMM\_WORLD);

}

else if (rank == 4)

{

MPI\_Recv(A + (H \* 5), N - (H \* 5), MPI\_INT, 5, 0, MPI\_COMM\_WORLD, &status);

MPI\_Send(A + (H \* 4), N - (H \* 4), MPI\_INT, 3, 0, MPI\_COMM\_WORLD);

}

else if (rank == 3)

{

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

MPI\_Send(A + (H \* 3), N - (H \* 3), MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

else if (rank == 2)

{

MPI\_Send(A + (H \* 2), H, MPI\_INT, 1, 0, MPI\_COMM\_WORLD);

}

else if (rank == 1)

{

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

MPI\_Send(A + H, 2 \* H, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

else if (rank == 0)

{

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

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

}

//output

if (rank == 0)

{

cout << "Result A:" << endl;

if (N <= 15)

outputVector(A);

else

cout << "Vector is too large." << endl;

}

MPI\_Finalize();

return 0;

}