Национальный Технический Университет Украины

Киевский Политехнический Институт

Факультет Информатики и Вычислительной Техники

Кафедра Вычислительной Техники

Лабораторная Робота №8

По предмету: «Параллельные и Распределенные Вычисления»

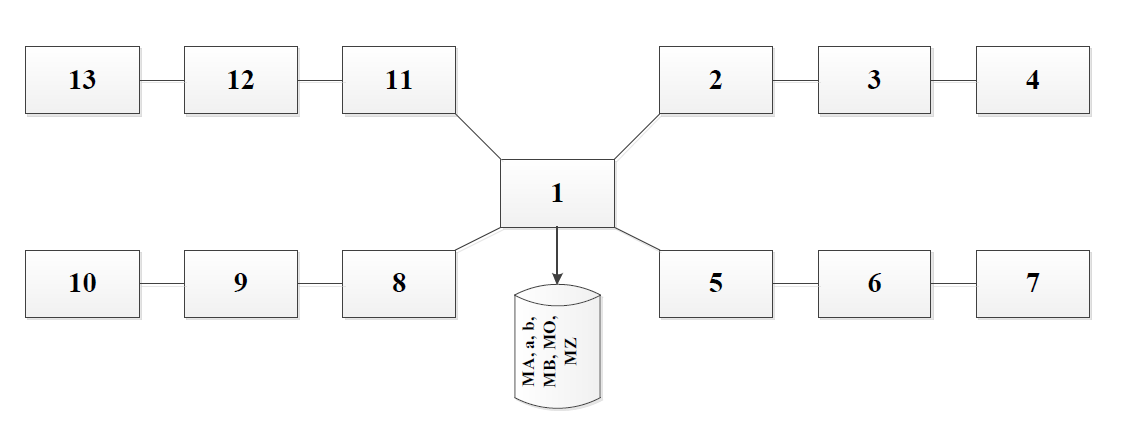
Выполнил: студент III-го курса

Группы ИВ-91

Визгин Антон Дмитриевич

Киев 2012г.

**Задание:** разработать параллельный алгоритм, алгоритмы процессов, схему их взаимодействия и программу для вычисления математической задачи.



**Этап 1. Разработка параллельного математического алгоритма.**

Математическая задача: MA = a \* MB + b \* MO \* MZ

Средства программирования: C++, библиотека MPI

Общие ресурсы: a, b, MO

**Этап 2. Разработка алгоритмов задач.**

**T1**

1. Ввод alpha, beta, MB, MO, MZ
2. Передача alpha, beta, MB[H..4H], МO[H..4H], MZ в задачу Т2
3. Передача alpha, beta, MB[4H..7H], МO[4H..7H], MZ в задачу Т5
4. Передача alpha, beta, MB[7H..10H], МO[7H..10H], MZ в задачу Т8
5. Передача alpha, MB[10H..13H], МC[10H..13H], MZ в задачу Т11
6. Счёт MAH= alpha \* MBH+ beta \* MO \* MZH
7. Принять МА[H..4H] от задачи Т2
8. Принять МА[3H..5H] от задачи Т5
9. Принять МА[5H..8H] от задачи Т8
10. Принять МА[8H..13H] от задачи Т11
11. Вывод МА

**T2**

1. Принять alpha, beta, MB[H..4H], МO[H..4H], MZ от задачи Т1
2. Передача alpha, beta, MB[2H..4H], МO[2H..4H], MZ в задачу Т3
3. Счёт MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[2H..4H] от задачи Т3
5. Передача MA[H..4H] в задачу Т1

**T3**

1. Принять alpha, beta, MB[2H..4H], МO[2H..4H], MZ от задачи Т2
2. Передача alpha, beta, MB[3H..4H], МO[3H..4H], MZ в задачу Т4
3. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[3H..4H] от задачи Т4
5. Передача MA[2H..4H] в задачу Т2

**Т4**

1. Принять alpha, beta, MB[3H..4H], МO[3H..4H], MZ от задачи Т3
2. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
3. Передача MA[3H..4H] в задачу Т3

**Т5**

1. Принять alpha, beta, MB[4H..7H], МO[4H..7H], MZ от задачи Т1
2. Передача alpha, beta, MB[5H..7H], МO[5H..7H], MZ в задачу Т6
3. Счёт MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[5H..7H] от задачи Т6
5. Передача MA[4H..7H] в задачу Т1

**T6**

1. Принять alpha, beta, MB[5H..7H], МO[5H..7H], MZ от задачи Т5
2. Передача alpha, beta, MB[6H..7H], МO[6H..7H], MZ в задачу Т7
3. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[6H..7H] от задачи Т7
5. Передача MA[5H..7H] в задачу Т5

**Т7**

1. Принять alpha, beta, MB[6H..7H], МO[6H..7H], MZ от задачи Т6
2. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
3. Передача MA[6H..7H] в задачу Т6

**Т8**

1. Принять alpha, beta, MB[7H..10H], МO[7H..10H], MZ от задачи Т1
2. Передача alpha, beta, MB[8H..10H], МO[8H..10H], MZ в задачу Т9
3. Счёт MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[8H..10H] от задачи Т9
5. Передача MA[7H..10H] в задачу Т1

**T9**

1. Принять alpha, beta, MB[8H..10H], МO[8H..10H], MZ от задачи Т8
2. Передача alpha, beta, MB[9H..10H], МO[9H..10H], MZ в задачу Т10
3. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[9H..10H] от задачи Т10
5. Передача MA[8H..10H] в задачу Т8

**Т10**

1. Принять alpha, beta, MB[9H..10H], МO[9H..10H], MZ от задачи Т9
2. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
3. Передача MA[9H..10H] в задачу Т9

**Т11**

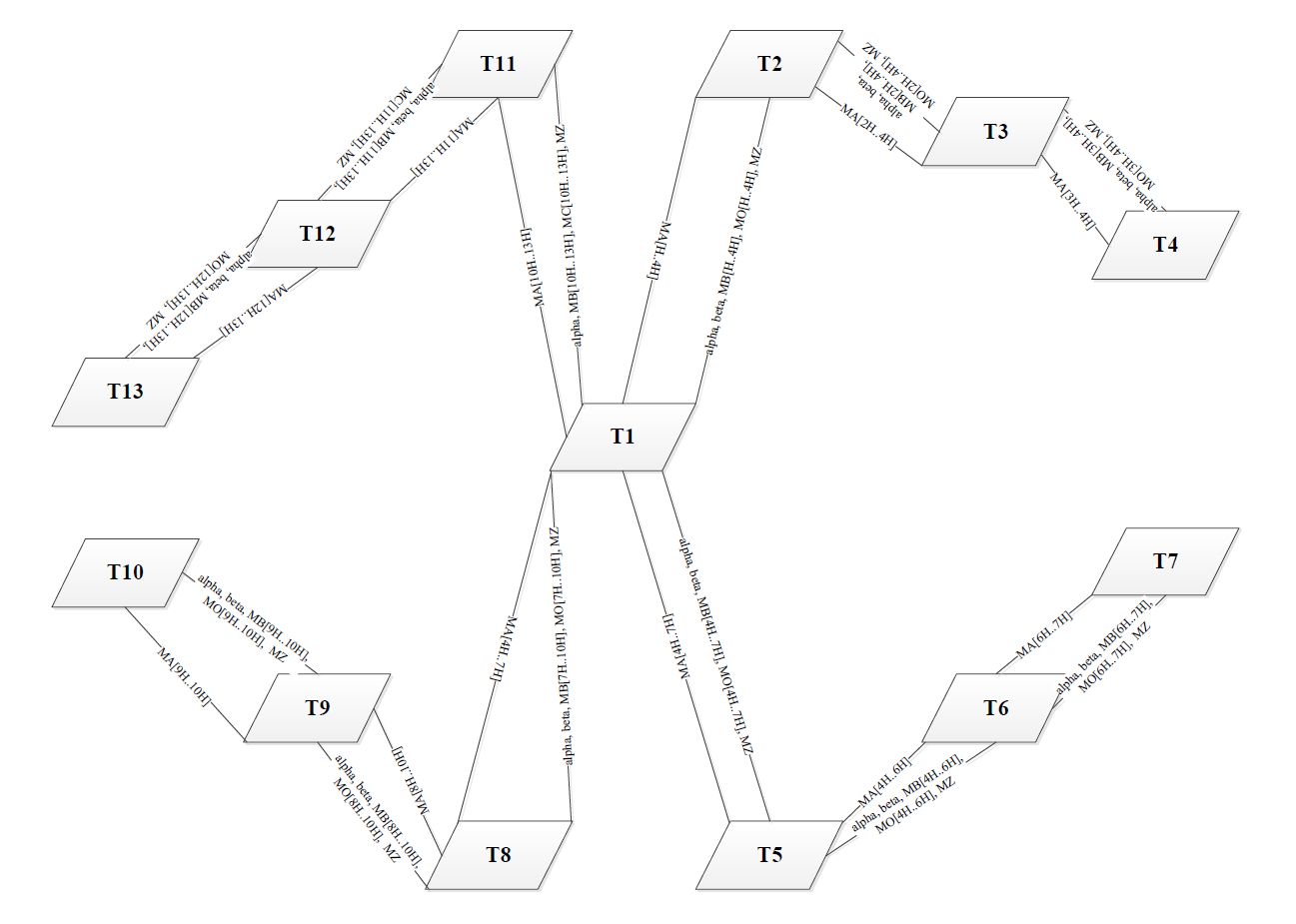
1. Принять alpha, beta, MB[10H..13H], МC[10H..13H], MZ от задачи Т1
2. Передача alpha, beta, MB[11H..13H], МC[11H..13H], MZ в задачу Т12
3. Счёт MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[11H..13H] от задачи Т12
5. Передача MA[10H..13H] в задачу Т1

**T12**

1. Принять alpha, beta, MB[11H..13H], МO[11H..13H], MZ от задачи Т11
2. Передача alpha, beta, MB[12H..13H], МO[12H..13H], MZ в задачу Т13
3. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
4. Принять MA[12H..13H] от задачи Т13
5. Передача MA[11H..13H] в задачу Т11

**Т13**

1. Принять alpha, beta, MB[12H..13H], МO[12H..13H], MZ от задачи Т12
2. Счёт MAH= MAH= alpha \* MBH+ beta \* MO \* MZH
3. Передача MA[12H..13H] в задачу Т12

**Этап 3. Разработка схемы взаимодействия задач**

**Этап 4. Разработка программы**

//---------------Курсова робота-----------------------------

//---------------Визгин Антон Дмитриевич--------------------

//---------------A = B \* MD + alpha \* C \* ( MO \* MT )-------

//---------------16.05.2012---------------------------------

#include <mpi.h>

#include <iostream>

using namespace std;

int N = 13;

int P = 13;

int H = N / P;

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

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

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

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

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

int alpha;

int beta;

MPI\_Status s;

static void inputMatrix(int\*\* matr, int temp)

{

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

{

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

{

matr[i][j] = temp;

}

}

}

static void calc()

{

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

{

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

{

MB[i][j] = alpha \* MB[i][j];

}

}

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

{

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

{

MO[i][j] = beta \* MO[i][j];

}

}

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

{

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

{

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

{

MO[i][j] += MO[i][k] \* MZ[k][j];

}

}

}

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

{

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

{

MA[i][j] = MB[i][j] + MO[i][j];

}

}

}

static void initMatrix(int\*\* matr)

{

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

{

matr[i] = new int [N];

}

}

static void send\_OR(int left, int right, int tag, int dest)

{

for (int i = left; i < right; i++)

{

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

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

}

}

static void recieve\_OR(int left, int right, int source, int tag)

{

for (int i = left; i < right; i++)

{

MPI\_Recv(MB[i], N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Recv(MO[i], N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

}

static void send\_ab(int tag, int dest)

{

MPI\_Send(&alpha, 1, MPI\_INT, dest, tag, MPI\_COMM\_WORLD);

MPI\_Send(&beta, 1, MPI\_INT, dest, tag, MPI\_COMM\_WORLD);

}

static void recieve\_ab(int source, int tag)

{

MPI\_Recv(&alpha, N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Recv(&beta, N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

static void send\_matr(int dest, int tag)

{

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

{

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

}

}

static void recieve\_matr(int source, int tag)

{

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

{

MPI\_Recv(MZ[i], N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

}

static void send\_MA(int left, int right, int dest, int tag)

{

for (int i = left; i < right; i++)

{

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

}

}

static void recieve\_MA(int left, int right, int source, int tag)

{

for (int i = left; i < right; i++)

{

MPI\_Recv(MA[i], N, MPI\_INT, source, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

}

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

{

MPI\_Init(&argc, &argv);

int rank;

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

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

//---------------Program Started-----------------------------

std::cout << "Main Started" << std::endl;

std::cout << "T" << P << " Started" << std::endl;

initMatrix(MA);

initMatrix(MB);

initMatrix(MO);

initMatrix(MZ);

if (rank == 0)

{

std::cout << "T1 Inputing" << std::endl;

inputMatrix(MO,1);

inputMatrix(MB,1);

inputMatrix(MZ,1);

alpha = 1;

beta = 1;

send\_OR(H, 4 \* H, 1, 0);

send\_ab(1, 0);

send\_matr(1, 0);

send\_OR(4 \* H, 7 \* H, 4, 0);

send\_ab(4, 0);

send\_matr(4, 0);

send\_OR(7 \* H, 10 \* H, 7, 0);

send\_ab(7, 0);

send\_matr(7, 0);

send\_OR(10 \* H, 13 \* H, 10, 0);

send\_ab(10, 0);

send\_matr(10, 0);

calc();

recieve\_MA(H, 4 \* H, 1, 0);

recieve\_MA(3 \* H, 5 \* H, 4, 0);

recieve\_MA(5 \* H, 8 \* H, 7, 0);

recieve\_MA(8 \* H, 13 \* H, 10, 0);

if (N <= 20)

{

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

{

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

cout << " " << MA[i][j] << endl;

}

}

}

if (rank == 1)

{

recieve\_ab(0, 0);

recieve\_OR(H, 4 \* H, 0, 0);

recieve\_matr(0, 0);

send\_ab(2, 0);

send\_OR(2 \* H, 4 \* H, 2, 0);

send\_matr(2, 0);

calc();

recieve\_MA(2 \* H, 4 \* H, 2, 0);

send\_MA(H, 4 \* H, 0, 0);

}

if (rank == 2)

{

recieve\_ab(1, 0);

recieve\_OR(2 \* H, 4 \* H, 1, 0);

recieve\_matr(1, 0);

send\_ab(3, 0);

send\_OR(3 \* H, 4 \* H, 3, 0);

send\_matr(3, 0);

calc();

recieve\_MA(3 \* H, 4 \* H, 3, 0);

send\_MA(2 \* H, 4 \* H, 1, 0);

}

if (rank == 3)

{

recieve\_ab(2, 0);

recieve\_OR(3 \* H, 4 \* H, 2, 0);

recieve\_matr(2, 0);

calc();

send\_MA(3 \* H, 4 \* H, 2, 0);

}

if (rank == 4)

{

recieve\_ab(0, 0);

recieve\_OR(4 \* H, 7 \* H, 0, 0);

recieve\_matr(0, 0);

send\_ab(5, 0);

send\_OR(5 \* H, 7 \* H, 5, 0);

send\_matr(5, 0);

calc();

recieve\_MA(5 \* H, 7 \* H, 5, 0);

send\_MA(4 \* H, 7 \* H, 0, 0);

}

if (rank == 5)

{

recieve\_ab(4, 0);

recieve\_OR(5 \* H, 7 \* H, 4, 0);

recieve\_matr(4, 0);

send\_ab(6, 0);

send\_OR(6 \* H, 7 \* H, 6, 0);

send\_matr(6, 0);

calc();

recieve\_MA(6 \* H, 7 \* H, 6, 0);

send\_MA(5 \* H, 7 \* H, 4, 0);

}

if (rank == 6)

{

recieve\_ab(5, 0);

recieve\_OR(6 \* H, 7 \* H, 5, 0);

recieve\_matr(5, 0);

calc();

send\_MA(6 \* H, 7 \* H, 5, 0);

}

if (rank == 7)

{

recieve\_ab(0, 0);

recieve\_OR(7 \* H, 10 \* H, 0, 0);

recieve\_matr(0, 0);

send\_ab(8, 0);

send\_OR(8 \* H, 10 \* H, 8, 0);

send\_matr(8, 0);

calc();

recieve\_MA(8 \* H, 10 \* H, 8, 0);

send\_MA(7 \* H, 10 \* H, 0, 0);

}

if (rank == 8)

{

recieve\_ab(7, 0);

recieve\_OR(8 \* H, 10 \* H, 7, 0);

recieve\_matr(7, 0);

send\_ab(9, 0);

send\_OR(9 \* H, 10 \* H, 9, 0);

send\_matr(9, 0);

calc();

recieve\_MA(9 \* H, 10 \* H, 9, 0);

send\_MA(8 \* H, 10 \* H, 8, 0);

}

if (rank == 9)

{

recieve\_ab(8, 0);

recieve\_OR(9 \* H, 10 \* H, 8, 0);

recieve\_matr(8, 0);

calc();

send\_MA(9 \* H, 10 \* H, 8, 0);

}

if (rank == 10)

{

recieve\_ab(0, 0);

recieve\_OR(10 \* H, 13 \* H, 0, 0);

recieve\_matr(0, 0);

send\_ab(11, 0);

send\_OR(11 \* H, 13 \* H, 11, 0);

send\_matr(11, 0);

calc();

recieve\_MA(11 \* H, 13 \* H, 11, 0);

send\_MA(10 \* H, 13 \* H, 0, 0);

}

if (rank == 11)

{

recieve\_ab(10, 0);

recieve\_OR(11 \* H, 13 \* H, 10, 0);

recieve\_matr(10, 0);

send\_ab(12, 0);

send\_OR(12 \* H, 13 \* H, 12, 0);

send\_matr(12, 0);

calc();

recieve\_MA(12 \* H, 13 \* H, 12, 0);

send\_MA(11 \* H, 13 \* H, 10, 0);

}

if (rank == 12)

{

recieve\_ab(11, 0);

recieve\_OR(12 \* H, 13 \* H, 11, 0);

recieve\_matr(11, 0);

calc();

send\_MA(12 \* H, 13 \* H, 11, 0);

}

std::cout << "T " << P << " Ended" << std::endl;

//---------------MPI\_Finilize()-----------------------------

MPI\_Finalize();

std::cout << "Main Ended" << std::endl;

std::cin;

return 0;

}