Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

**(ПНИПУ)**

Факультет: Электротехнический

Кафедра: Информационные технологии и автоматизированные системы.

# Направление подготовки: Разработка программно-информационных систем.

**Лабораторная работа № 5**

Технология MPI. Решение СЛАУ методом Гаусса.

Выполнил студент гр. РИС-18-1б

Гилязов Р.А.

(Фамилия, имя, отчество)

###### Проверил:

Старший преподаватель кафедры ИТАС, Щапов В.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, Ф.И.О. руководителя от кафедры)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

**Пермь, 2020**

**Постановка задачи:**

1. Написать программу, которая будет решать СЛАУ методом Гаусса.
2. Используя технологию MPI, распараллелить решение.

**Решение:**

#include <mpi.h>

#include <random>

#include <iostream>

void init\_matrix(float\* pMatrix, int Size) {

std::default\_random\_engine generator;

std::normal\_distribution<double> randomize(-10, 100);

for (int i = 0; i < Size; i++) {

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

pMatrix[i \* Size + j] = randomize(generator);

}

}

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

int size\_matrix = 1000;

double begin;

double end;

int rank;

int count\_process;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &count\_process);

int current\_num\_row = size\_matrix / count\_process;

float\* matrix = new float[size\_matrix \* size\_matrix];

init\_matrix(matrix, size\_matrix);

float\* current\_matrix = new float[size\_matrix \* current\_num\_row];

MPI\_Scatter(matrix, size\_matrix \* current\_num\_row, MPI\_FLOAT, current\_matrix,

size\_matrix \* current\_num\_row, MPI\_FLOAT, 0, MPI\_COMM\_WORLD);

float\* rows = new float[size\_matrix \* current\_num\_row];

if (rank == 0) {

begin = MPI\_Wtime();

}

for (int i = 0; i < (rank \* current\_num\_row); i++) {

MPI\_Bcast(rows, size\_matrix, MPI\_FLOAT, i / current\_num\_row, MPI\_COMM\_WORLD);

for (int j = 0; j < current\_num\_row; j++) {

for (int k = i + 1; k < size\_matrix; k++) {

current\_matrix[j \* size\_matrix + k] -= current\_matrix[j \* size\_matrix + i] \* rows[k];

}

current\_matrix[j \* size\_matrix + i] = 0;

}

}

int pivot\_colum;

for (int i = 0; i < current\_num\_row; i++) {

pivot\_colum = rank \* current\_num\_row + i;

for (int j = pivot\_colum; j < size\_matrix; j++) {

current\_matrix[i \* size\_matrix + j] /= current\_matrix[i \* size\_matrix + pivot\_colum];

}

current\_matrix[i \* size\_matrix + pivot\_colum] = 1;

memcpy(rows, current\_matrix + (i \* size\_matrix), size\_matrix \* sizeof(float));

MPI\_Bcast(rows, size\_matrix, MPI\_FLOAT, rank, MPI\_COMM\_WORLD);

for (int j = i + 1; j < current\_num\_row; j++) {

for (int k = pivot\_colum + 1; k < size\_matrix; k++) {

current\_matrix[j \* size\_matrix + k] -= current\_matrix[j \* size\_matrix + i] \* rows[k];

}

current\_matrix[j \* size\_matrix + pivot\_colum] = 0;

}

}

for (int i = rank \* current\_num\_row + 1; i < size\_matrix; i++) {

MPI\_Bcast(rows, size\_matrix, MPI\_FLOAT, i / current\_num\_row, MPI\_COMM\_WORLD);

}

MPI\_Barrier(MPI\_COMM\_WORLD);

if (rank == 0) {

end = MPI\_Wtime() - begin;

}

MPI\_Gather(current\_matrix, size\_matrix \* current\_num\_row, MPI\_FLOAT, matrix,

size\_matrix \* current\_num\_row, MPI\_FLOAT, 0, MPI\_COMM\_WORLD);

if (rank == 0) {

std::cout << "Size = " << size\_matrix << "\nTime: " << end << "second" << std::endl;

}

MPI\_Finalize();

delete[] matrix;

delete[] current\_matrix;

delete[] rows;

}

**Тесты**

Таблица 1 – Замеры времени для разного количества процессов

|  |  |  |  |
| --- | --- | --- | --- |
| Количество переменных | 1 ядро, сек | 4 ядра, сек | 6 ядер, сек |
| 100 | 0.0007341 | 0.0004767 | 0.0008727 |
| 1000 | 0.18064 | 0.0707087 | 0.0609431 |
| 5000 | 29.4611 | 16.0513 | 14.2805 |