МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ. WORK15

ОТЧЕТ О ПРАКТИКЕ

студента 3 курса 311 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Вильцева Данила Денисовича

Проверил
Старший преподаватель _____ М. С. Портенко

Саратов 2023

СОДЕРЖАНИЕ

1 Work 15

1.1 Условие задачи

Аналогично работе с OMP выполните следующие задания через MPI.

Выполните разработку параллельного варианта для одного из итерационных методов:

3. верхней релаксации.

Для тестовой матрицы из нулей и единиц проведите вычислительные эксперименты, результаты занесите в таблицу 1.

Таблица 1. Время выполнения последовательного и параллельного итерационного алгоритмов решения систем линейных уравнений и ускорение

Номер теста	Порядок системы	Последовательный алгоритм	Параллельный алгоритм	
			Время	Ускорение
1	10			
2	100			
3	500			
4	1000			
5	1500			
6	2000			
7	2500			
8	3000			

Какой из алгоритмов Гаусса или итерационный обладает лучшими показателями ускорения? Заполните таблицу 2.

Таблица 2. Ускорение параллельных алгоритмов Гаусса и итерационного (вариант) решения систем линейных уравнений

Номер теста	Порядок системы	Ускорение алгоритма Гаусса	Ускорение итерационного алгоритма (вариант)
1	10		
2	100		
3	500		
4	1000		
5	1500		
6	2000		
7	2500		
8	3000		

1.2 Решение. Параллельная версия

```
#include <iostream>
#include <time.h>
#include <cmath>
#include <windows.h>
#include <cstdlib>
#include <mpi.h>


using namespace std;


int ProcNum;
int ProcRank;
int* pParallelPivotPos;  // The Number of pivot rows selected at the iterations
int* pProcPivotIter;     // The Iterations, at which the rows were pivots
int* pProcInd;
int* pProcNum;

// Function that converts numbers form LongInt type to
// double type
double LiToDouble(LARGE_INTEGER x) {
        double result = ((double)x.HighPart) * 4.294967296E9 +
                (double)((x).LowPart);
        return result;
}

// Function that gets the timestamp in seconds
double GetTime() {
        LARGE_INTEGER lpFrequency, lpPerfomanceCount;
        QueryPerformanceFrequency(&lpFrequency);
        QueryPerformanceCounter(&lpPerfomanceCount);
        return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
}

double* upper_relaxation_method(double** a, double* b, int n, double eps, double w, double*
↪  x, double* xn) {
        int i, j, k = 0;
        double norma;
        for (i = 0; i < n; i++)
        {
                xn[i] = 0;
                x[i] = xn[i];
        }
        do
        {
                k++;
                norma = 0;
```

```
                for (i = 0; i < n; i++)
                {
                        x[i] = b[i];
                        for (j = 0; j < n; j++)
                        {
                                if (i != j)
                                        x[i] = x[i] - a[i][j] * x[j];
                        }
                        x[i] /= a[i][i];

                        x[i] = w * x[i] + (1 - w) * xn[i];

                        if (fabs(x[i] - xn[i]) > norma)
                                norma = fabs(x[i] - xn[i]);
                        xn[i] = x[i];
                        MPI_Reduce(&norma, &xn[i], 1, MPI_DOUBLE, MPI_SUM, 0,
↪  MPI_COMM_WORLD);
                }
        } while (norma > eps);

        return x;

}

double experiment(double* res, double** a, double* b, int n, double eps, double w, double*
↪  x, double* xn)
{
        double stime, ftime; // время начала и конца расчета
        stime = GetTime();
        upper_relaxation_method(a, b, n, eps, w, x, xn); // вызов функции интегрирования
        ftime = GetTime();
        return (ftime - stime) / CLOCKS_PER_SEC;
}


int main()
{
        setlocale(LC_CTYPE, "RUSSIAN");
        int n;
        double eps;
        double w;

        cout << "Введите размерность матрицы N*N:";
        cin >> n;
        double** a = new double* [n];
        for (int i = 0; i < n; i++)
                a[i] = new double[n];
```

```
double* b = new double[n];
double* x = new double[n];
double* xn = new double[n];

for (int i = 0; i < n; i++)
{
        for (int j = 0; j < n; j++)
        {

                a[i][j] = rand() / double(1000);
        }
}

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

        b[i] = rand() / double(1000);
}

eps = 0.001;
w = 1.12;

double time; // время проведенного эксперимента
double res; // значение вычисленного интеграла
double min_time; // минимальное время работы
                                // реализации алгоритма
double max_time; // максимальное время работы
                                // реализации алгоритма
double avg_time; // среднее время работы
                                // реализации алгоритма
int numbExp = 10; // количество запусков программы

// первый запуск
MPI_Init(NULL, NULL);
MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
min_time = max_time = avg_time = experiment(&res, a, b, n, eps, w, x, xn);
// оставшиеся запуски
for (int i = 0; i < numbExp - 1; i++)
{
        time = experiment(&res, a, b, n, eps, w, x, xn);
        avg_time += time;
        if (max_time < time) max_time = time;
        if (min_time > time) min_time = time;
}
// вывод результатов эксперимента
```

```
        cout << "execution time : " << avg_time / numbExp << "; " <<
                min_time << "; " << max_time << endl;
        cout.precision(8);
        MPI_Finalize();
}
```

2 Результат работы

```
Введите размерность матрицы N*N:10
execution time : 1.8204e-07; 1.462e-07; 2.301e-07
```

```
Введите размерность матрицы N*N:100
execution time : 4.6529e-07; 3.749e-07; 8.724e-07
```

```
Введите размерность матрицы N*N:500
execution time : 1.52253e-06; 1.4353e-06; 1.7096e-06
```

```
Введите размерность матрицы N*N:1000
execution time : 5.01069e-06; 4.2747e-06; 5.681e-06
```

```
Введите размерность матрицы N*N:1500
execution time : 9.49917e-06; 9.1884e-06; 9.9701e-06
```

```
Введите размерность матрицы N*N:2000
execution time : 1.11124e-05; 1.06317e-05; 1.20982e-05
```

```
Введите размерность матрицы N*N:2500
execution time : 1.66678e-05; 1.62406e-05; 1.74302e-05
```

```
Введите размерность матрицы N*N:3000
execution time : 2.37601e-05; 2.25053e-05; 2.61351e-05
```

3 Результат работы. Таблицы

Номер теста	Порядок системы	Последовательный алгоритм	Параллельный алгоритм	
			Время	Ускорение
1	10	4.70599e-08	1.8204e-07	0.258
2	100	6.3934e-07	4.6529e-07	1.374067
3	500	3.39393e-06	1.52253e-06	2.229118
4	1000	3.96223e-05	5.01069e-06	7.90755
5	1500	3.32889e-05	9.49917e-06	3.50446
6	2000	2.66843e-05	1.11124e-05	2.40125
7	2500	4.23384e-05	1.66678e-05	2.54013
8	3000	5.99791e-05	2.37601e-05	2.24136

Номер теста	Порядок системы	Ускорение Гаусса	Ускорение верхней релаксации
1	10	0	0.258
2	100	2,65	1.374067
3	500	2,76	2.229118
4	1000	2,972222	7.90755
5	1500	2,806	3.50446
6	2000	3,5494	2.40125
7	2500	3,3397	2.54013
8	3000	3,465	2.24136

4 Характеристики компьютера

Характеристики устройства

Имя устройства	DESKTOP-MSS8D39
Процессор	Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz 3.20 GHz
Оперативная память	8,00 ГБ
Код устройства	E3BB953D-13B0-42A7-944B-1ED9FD0E C328
Код продукта	00330-80000-00000-AA153
Тип системы	64-разрядная операционная система, процессор x64