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

ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ. WORK11

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 11 3
1.1 Условие задачи 3
1.2 Последовательная реализация 3
1.3 Параллельная реализация (по внешнему циклу) 4
1.4 Параллельная реализация (по внутреннему циклу) 6
2 Результат работы 9
3 Характеристики компьютера 10

1 Work 11

1.1 Условие задачи

Аналогично работе с OMP выполните следующее задание через MPI.

Модифицируйте разработанную ранее программу по методу прямоугольников для численного интегрирования тестовой функции:

$$\int_0^{16}\int_0^{16} \frac{e^{\sin\pi x\cos\pi y}+1}{(b_1-a_1)(b_2-a_2)}dx\,dy \approx 2.130997$$

Сравните время численного интегрирования для последовательной и параллельных реализаций и параллельных реализаций между собой.

1.2 Последовательная реализация

```
#include <iostream>
#include <cmath>
#include <mpi.h>

using namespace std;
#define PI 3.14159265358979323846264338327795

double func(double x, double y, const double a1, const double b1, const double a2, const
↪  double b2)
{
        return ((exp(sin(PI * x) * cos(PI * y)) + 1) / ((b1 - a1) * (b2 - a2)));
}
void integral(const double a1, const double b1, const double a2, const double b2,
        const double h1, const double h2, double* res) {
        int i, j, N, M;
        double sum; // локальная переменная для подсчета интеграла
        double x, y; // координата точки сетки

        N = (int)((b1 - a1) / h1); // количество точек сетки интегрирования
        M = (int)((b2 - a2) / h2);

        sum = 0.0;

        for (i = 1; i < N; i++) {
                x = a1 + i * h1 + h1 / 2;
                for (j = 1; j < M; j++) {
                        y = a2 + j * h2 + h2 / 2;
                        sum += h1 * h2 * func(x, y, a1, b1, a2, b2);
                }
```

```
        }

        *res = sum;
}

double experiment(double* res) {
        double stime, ftime; // время начала и конца расчета
        double a = 0.0; // левая граница интегрирования
        double b = 16.0; // правая граница интегрирования
        double h1 = 0.01; // шаг интегрирования
        double h2 = 0.01; // шаг интегрирования
        stime = clock();
        integral(a, b, a, b, h1, h2, res); // вызов функции интегрирования
        ftime = clock();
        return (ftime - stime) / CLOCKS_PER_SEC;
}

int main(int* argc, char** argv) {
        int i; // переменная цикла
        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
        avg_time = experiment(&res);

        // вывод результатов эксперимента
        cout << "execution time : " << avg_time / numbExp << endl;
        cout.precision(8);
        cout << "integral value : " << res << endl;
        return 0;
}
```

1.3 Параллельная реализация (по внешнему циклу)

```
#include <iostream>
#include <cmath>
#include <mpi.h>

using namespace std;
#define PI 3.14159265358979323846264338327795
```

```
double func(double x, double y, const double a1, const double b1, const double a2, const
↪  double b2)
{
        return ((exp(sin(PI * x) * cos(PI * y)) + 1) / ((b1 - a1) * (b2 - a2)));
}
void integral(const double a1, const double b1, const double a2, const double b2,
        const double h1, const double h2, double* res) {
        int i, j, N, M;
        double sum; // локальная переменная для подсчета интеграла
        double x, y; // координата точки сетки

        N = (int)((b1 - a1) / h1); // количество точек сетки интегрирования
        M = (int)((b2 - a2) / h2);

        sum = 0.0;
        int commsize;
        int rank;
        double Result;
        MPI_Init(NULL, NULL);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        MPI_Comm_size(MPI_COMM_WORLD, &commsize);
        MPI_Bcast(&N, 1, MPI_INT, 0, MPI_COMM_WORLD);

        for (i = 1; i < N; i++) {
                x = a1 + i * h1 + h1 / 2;
                for (j = 1; j < M; j++) {
                        y = a2 + j * h2 + h2 / 2;
                        sum += h1 * h2 * func(x, y, a1, b1, a2, b2);
                }
        }

        MPI_Reduce(&sum, &Result, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

        MPI_Finalize();

        *res = sum;
}

double experiment(double* res) {
        double stime, ftime; // время начала и конца расчета
        double a = 0.0; // левая граница интегрирования
        double b = 16.0; // правая граница интегрирования
        double h1 = 0.01; // шаг интегрирования
        double h2 = 0.01; // шаг интегрирования
        stime = clock();
        integral(a, b, a, b, h1, h2, res); // вызов функции интегрирования
        ftime = clock();
        return (ftime - stime) / CLOCKS_PER_SEC;
```

```
}

int main(int* argc, char** argv) {
        int i; // переменная цикла
        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
        avg_time = experiment(&res);

        // вывод результатов эксперимента
        cout << "execution time : " << avg_time / numbExp << endl;
        cout.precision(8);
        cout << "integral value : " << res << endl;
        return 0;
}
```

1.4 Параллельная реализация (по внутреннему циклу)

```
#include <iostream>
#include <cmath>
#include <mpi.h>

using namespace std;
#define PI 3.14159265358979323846264338327795

double func(double x, double y, const double a1, const double b1, const double a2, const
↪  double b2)
{
        return ((exp(sin(PI * x) * cos(PI * y)) + 1) / ((b1 - a1) * (b2 - a2)));
}
void integral(const double a1, const double b1, const double a2, const double b2,
        const double h1, const double h2, double* res) {
        int i, j, N, M;
        double sum; // локальная переменная для подсчета интеграла
        double x, y; // координата точки сетки

        N = (int)((b1 - a1) / h1); // количество точек сетки интегрирования
        M = (int)((b2 - a2) / h2);

        sum = 0.0;
```

```
        int commsize;
        int rank;
        double Result;
        MPI_Init(NULL, NULL);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        MPI_Comm_size(MPI_COMM_WORLD, &commsize);
        MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);

        for (i = 1; i < N; i++) {
                x = a1 + i * h1 + h1 / 2;
                for (j = 1; j < M; j++) {
                        y = a2 + j * h2 + h2 / 2;
                        sum += h1 * h2 * func(x, y, a1, b1, a2, b2);
                }
        }

        MPI_Reduce(&sum, &Result, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

        MPI_Finalize();

        *res = sum;
}

double experiment(double* res) {
        double stime, ftime; // время начала и конца расчета
        double a = 0.0; // левая граница интегрирования
        double b = 16.0; // правая граница интегрирования
        double h1 = 0.01; // шаг интегрирования
        double h2 = 0.01; // шаг интегрирования
        stime = clock();
        integral(a, b, a, b, h1, h2, res); // вызов функции интегрирования
        ftime = clock();
        return (ftime - stime) / CLOCKS_PER_SEC;
}

int main(int* argc, char** argv) {
        int i; // переменная цикла
        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
```

```
        avg_time = experiment(&res);

        // вывод результатов эксперимента
        cout << "execution time : " << avg_time / numbExp << endl;
        cout.precision(8);
        cout << "integral value : " << res << endl;
        return 0;
}
```

2 Результат работы

Распараллеливание по внешнему циклу дает ускорение в 1.52 раза.

Распараллеливание по внутреннему циклу дает ускорение 1.3 раза.

Таким образом, наиболее эффективным вариантом распараллеливания функции, вычисляющей двойной интеграл, является вариант с распараллеливанием внешнего цикла.

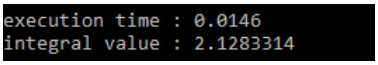

Рисунок 1 – Результат последовательной реализации

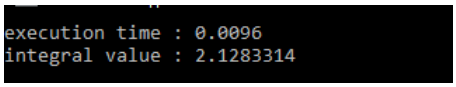

Рисунок 2 – Результат параллельной реализации (по внешнему циклу)

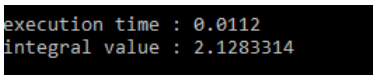

Рисунок 3 – Результат параллельной реализации (по внутреннему циклу)

3 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