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

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ, ФОРМУЛА СИМПСОНА, ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ И КРАТНЫХ ИНТЕГРАЛОВ

ОТЧЕТ О ПРАКТИКЕ

Студента 3 курса 311 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Аношкина Андрея Алексеевича

Проверил
Старший преподаватель ____________ М. С. Портенко

Саратов 2024

СОДЕРЖАНИЕ

1 Work 10

Задание

Аналогично работе с OMP выполните следующее задание через MPI.

Реализуйте параллельные алгоритмы, использующие метод прямоугольников и формулу Симпсона для подсчета интегралов. Точные значения интегралов указаны для проверки численных вычислений. В случае, если в верхнем пределе интегрирования указан знак бесконечности, то в расчете необходимо заменить его на 10^6. Сравните время численного интегрирования для последовательной и параллельной реализации. Какое ускорение выполнения программы предоставляет переход к многопоточной версии?

Вариант задания 2:

$$\int_0^\infty \frac{dx}{1+x^2} = \frac{\pi}{2}$$

Метод прямоугольников, формула Симпсона

Метод прямоугольников геометрически заключается в том, что интеграл приближенно представляется в виде суммы площадей элементарных прямоугольников.

Для случая деления отрезка интегрирования на равные части и вычисления функции в центре отрезков:

$$\begin{cases} J = \int_a^b f(x)\,dx \approx \sum_{i=0}^{N-1}(h \cdot f(x_i)) = h \cdot \sum_{i=0}^{N-1} f(x_i) \\ x_i = a + i \cdot h + \frac{h}{2} \end{cases} \tag{1}$$

где N — количество отрезков интегрирования, а $h = (b-a)/N$

Ступенчатая (stair-case) аппроксимация гладких изогнутых поверхностей, возникает в результате дискретизации модели прямоугольной сеткой. Для борьбы со ступенчатой аппроксимацией может использоваться, например, квадратурная формула Симпсона:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}[\,f(a) + f(b) + 4\sum_{k=1}^{N} f(a+(2k-1)h) + 2\sum_{k=1}^{N-1} f(a+2kh)] \tag{2}$$

где $h = \frac{b-a}{2N}, N \gg 1$.

Формула Симпсона геометрически заключается в том, что через три ординаты, отвечающие трем последовательным узлам сетки, проводится парабола и затем складываются получившиеся при этом площади элементарных криволинейных трапеций.

Реализация

Код решения приведен ниже:

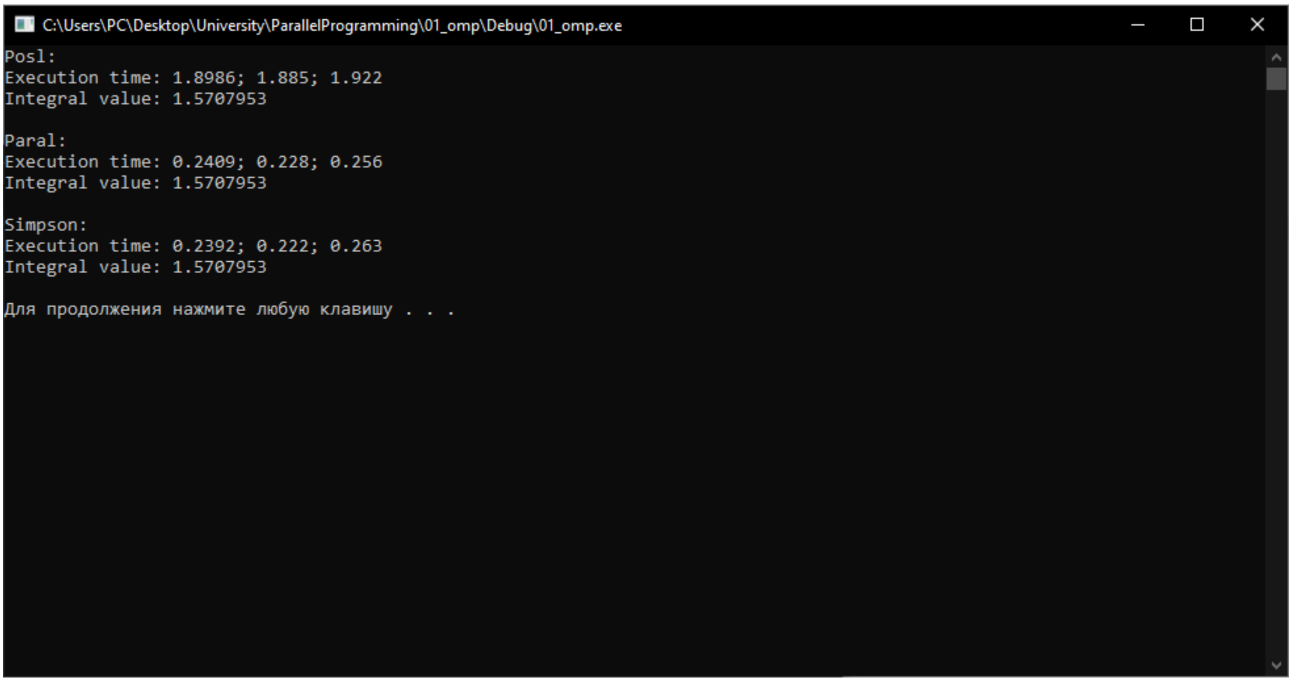
```
//MPI_Simpliest_c_Bind.cpp
#include "mpi.h"
#include <iostream>
#include <time.h>
#define PI 3.1415926535897932384626433832795

using namespace std;

int NProc, ProcId;
MPI_Status st;

double f1(double x) {
	return 1.0 / (1 + x * x);
}

void integral_posl(const double a, const double b, const double h, double* res) {
	double sum = 0;
	int n = (int)((b - a) / h);

	for (int i = 0; i < n; ++i) {
		double x = a + i * h + h / 2;
		sum += f1(x) * h;
	}

	*res = sum;

}

void integral_paral(const double a, const double b, const double h, double* res) {
	MPI_Barrier(MPI_COMM_WORLD);
	double sum = 0;
	int n = (int)((b - a) / h);

	for (int i = ProcId; i < n; i += NProc) {
		double x = a + i * h + h / 2;
		sum += f1(x) * h;
	}

	if (ProcId != 0)
		MPI_Send(&sum, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
```

```
    else {
        for (int i = 1; i < NProc; ++i) {
            double s;
            MPI_Recv(&s, 1, MPI_DOUBLE, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD,
            ↪  &st);
            sum += s;
        }

        *res = sum;
    }
}

void integral_Simpson(const double a, const double b, const double h, double* res) {
    MPI_Barrier(MPI_COMM_WORLD);
    double sum = f1(a) + f1(b);
    int n = (int)((b - a) / 2 / h);

    for (int i = ProcId + 1; i <= n; i += NProc) {
        double x = a + h * (2 * i - 1);
        sum += f1(x) * 4;
    }

    for (int i = ProcId + 1; i < n; i += NProc) {
        double x = a + 2 * i * h;
        sum += 2 * f1(x);
    }

    if (ProcId != 0)
        MPI_Send(&sum, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
    else {
        for (int i = 1; i < NProc; ++i) {
            double s;
            MPI_Recv(&s, 1, MPI_DOUBLE, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD,
            ↪  &st);
            sum += s;
        }

        *res = h / 3 * sum;
    }
}

double experiment(double* res, void f(const double a, const double b, const double h, double* res)) {
    double stime = 0, ftime = 0;
    double a = 0;
    double b = 1e+6;
    double h = 0.01;
    stime = clock();
    f(a, b, h, res);
    ftime = clock();

```

```
    return (ftime - stime) / CLOCKS_PER_SEC;
}

void calculate(void f(const double a, const double b, const double h, double* res)) {
    double avg_time = 0;
    double min_time = 0;
    double max_time = 0;
    double res = 0;
    int numbExp = 1;

    min_time = max_time = experiment(&res, f);
    avg_time = min_time / numbExp;
    for (int i = 0; i < numbExp - 1; ++i) {
        double time = experiment(&res, f);
        if (time > max_time)
            max_time = time;
        if (time < min_time)
            min_time = time;

        avg_time += time / numbExp;
    }

    if (ProcId == 0) {
        cout << "Execution time: " << avg_time << "; " << min_time << "; " << max_time << "\n";
        cout.precision(8);
        cout << "Integral value: " << res << "\n";
    }
}

int main() {

    bool calcPosl = true, calcParal = true, calcSimpson = true;

    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &NProc);
    MPI_Comm_rank(MPI_COMM_WORLD, &ProcId);

    // Последовательное вычисление

    if (calcPosl && ProcId == 0) {
        cout << "Posl:\n";
        calculate(integral_posl);
        cout << "\n";
    }

    // Параллельное вычисление

    if (calcParal) {
        if (ProcId == 0)
```

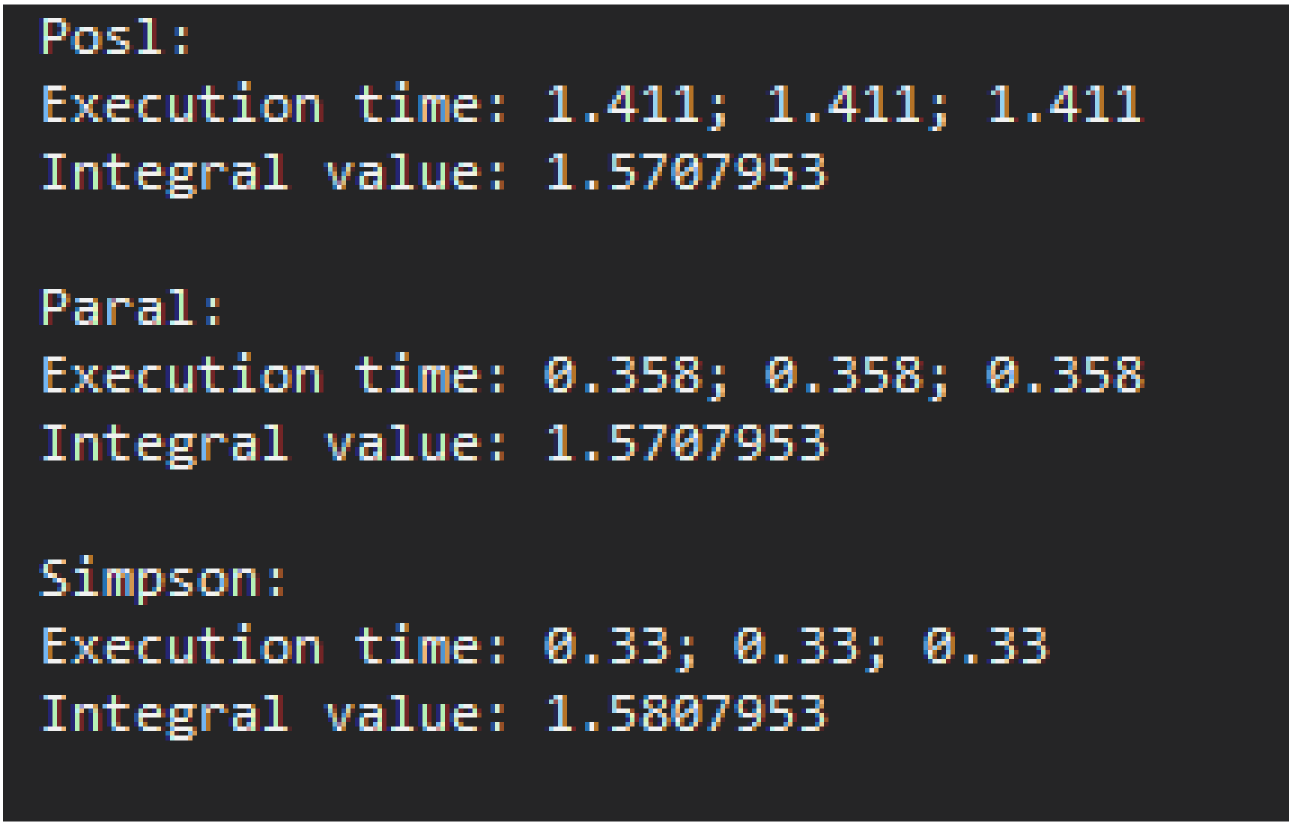
```
            cout << "Paral:\n";
        calculate(integral_paral);
        if (ProcId == 0)
            cout << "\n";
    }

    // Параллельное вычисление по формуле Симпсона

    if (calcSimpson) {
        if (ProcId == 0)
            cout << "Simpson:\n";
        calculate(integral_Simpson);
        if (ProcId == 0)
            cout << "\n";
    }

    MPI_Finalize();

    return 0;
}
```

Результаты работы



```
Posl:
Execution time: 1.8986; 1.885; 1.922
Integral value: 1.5707953

Paral:
Execution time: 0.2409; 0.228; 0.256
Integral value: 1.5707953

Simpson:
Execution time: 0.2392; 0.222; 0.263
Integral value: 1.5707953

Для продолжения нажмите любую клавишу . . .
```

Рисунок 1 – Work-1 (OpenMP)



```
Posl:
Execution time: 1.411; 1.411; 1.411
Integral value: 1.5707953

Paral:
Execution time: 0.358; 0.358; 0.358
Integral value: 1.5707953

Simpson:
Execution time: 0.33; 0.33; 0.33
Integral value: 1.5807953
```

Рисунок 2 – Work-10 (OpenMPI)

Результаты работы приложения с $OpenMPI$ оказались хуже, чем работа реализации с использованием $OpenMP$ в связи со временем, необходимым для пересылки сообщений между процессами. Однако в перспективе более сложных вычислений $OpenMPI$ гарантирует лучшую эффективность.

Характеристики устройства

Процессор: Intel(R) Core(TM) i5-10400F

Ядер: 6

Оперативная память: 16 Гб