Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

ОТЧЕТ

по дисциплине «Параллельное программирования»
Лабораторная работа №4

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии
Направленность (профиль) «Искусственный интеллект и разработка программных продуктов»

Руководитель работы
старший преподаватель ММФ
_____В.И. Лаева
подпись
«_____» _____ 20 ___ г.

Выполнил
студент группы № 932204
_____ М.В. Бондарев
подпись
«_____» _____ 20 ___ г.

Томск – 2024

Задание

Написать MPI-программу вычисления определенного интеграла, используя обобщенную квадратурную формулу Ньютона(«3/8»):

$$\int_a^b f(x)dx = \frac{b-a}{8n}\left[f(a)+f(b)+\sum_{i=1}^{3n-1}\begin{Bmatrix}2*f(a+i*h) \text{ если i кратно 3}\\ 3*f(a+i*h) \text{ если i не кратно 3}\end{Bmatrix}\right]$$

$$h = \frac{b-a}{2*n}$$

Обеспечить равномерную загрузку всех процессорных элементов, участвующих в работе программы. Вычислить ускорение и эффективность программы.

Интеграл

$$\int_{2.5}^{5} \frac{e^{-tan(0.8*x)}}{1.35+cos(x)}dx$$

Листинг программы

```c
#include <stdio.h>
#include <mpi.h>
#include <math.h>

double f(double x) {
    return exp(-tan(0.8 * x)) / (1.35 + cos(x));
}

double integrate(double a, double b, int n) {
    double h = (b - a) / (3 * n);
    double sum = 0.0;
    for (int i = 1; i < ((3 * n) - 1); i++) {
        double x = a + i * h;
        if (i % 3 == 0) {sum += 2 * f(x);}
        else {sum += 3 * f(x);}
    }
    return ((b - a)/(8 * n))*(f(a) + f(b) + sum);
}
```

```c
int main(int argc, char** argv) {
    int rank, size;
    double a = 2.5, b = 5;
    int n = 100000000;
    double result, local_result;
    double start_time, end_time;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int local_n = n / size;

        MPI_Barrier(MPI_COMM_WORLD);
    start_time = MPI_Wtime();

    local_result = integrate(
    a + (rank * (b - a) / size),
    a + ((rank + 1) * (b - a) / size),
    local_n);

    end_time = MPI_Wtime();
        MPI_Barrier(MPI_COMM_WORLD);

    MPI_Reduce(&local_result, &result, 1, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Size: %d \n", size);
        printf("Result: %.10f \n", result);
        printf("Time: %f seconds\n", end_time - start_time);
    }

    MPI_Finalize();

    return 0;
}
```

Описание программы

В коде программы заданы две функции:

- **f(double x)**
- **integrate(double a, double b, int n)**

f(x) вычисляет значение функции в точке **x**.

Функция **integrate(a, b, n)** - вычисляет значение интеграла с помощью обобщенной квадратурной формулы Ньютона(3/8).

Функция **integrate** вызывается на каждом из процессов, где каждому процессу задается свой интервал по формуле:

$$a = a + \frac{rank * (b - a)}{size}$$

$$b = a + \frac{(rank + 1) * (b - a)}{size}$$

Благодаря чему каждый процессор получит свой интервал, следовательно будет выполняться условие равномерно распределенной нагрузки.

Также число n вычисляется в зависимости от количества процессов

$$local\_n = \frac{n}{size}$$

Тестирование программы

Программа была запущена на 1, 2, 4, 8, 16 процессах, результат выполнения представлен ниже:

```
Size: 1
Result: 9.2877267831
Time: 28.668 seconds

Size: 2
Result: 9.2877267626
Time: 14.368 seconds

Size: 4
Result: 9.2877266998
Time: 7.163 seconds

Size: 8
Result: 9.2877265688
Time: 3.586 seconds

Size: 16
Result: 9.2877262959
Time: 1.797 seconds
```

Производительность программы

Во всех запусках программа даёт верный результат вычисления интеграла.

Оценим ускорение: $S_p = \dfrac{T_1}{T_p}$ и эффективность: $E_p = \dfrac{S_p}{p}$

$$S_2 = \frac{T_1}{T_2} = \frac{28.668}{14.368} = 1.995 \qquad E_2 = \frac{S_2}{2} = \frac{1.995}{2} = 0,997$$

$$S_4 = \frac{T_1}{T_4} = \frac{28.668}{7.163} = 3.932 \qquad E_4 = \frac{S_4}{4} = \frac{3.932}{4} = 0.983$$

$$S_8 = \frac{T_1}{T_8} = \frac{28.668}{3.586} = 7.994 \qquad E_8 = \frac{S_8}{8} = \frac{7.994}{8} = 0.999$$

$$S_{16} = \frac{T_1}{T_{16}} = \frac{28.668}{1.797} = 15.953 \qquad E_{16} = \frac{S_{16}}{16} = \frac{15.953}{16} = 0.997$$

Хорошее ускорение наблюдается при любом числе параллельных процессов и эффективность параллельного алгоритма очень высокая.