Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

ОТЧЕТ

по курсу «Параллельное программирование»

Выполнил

студент группы №932201

_____Д. А. Прокопьев

Проверил

старший преподаватель ММФ

_____В. И. Лаева

Томск-2024

Задание 6.

Используя OpenMP, реализовать программу для вычисления определенного интеграла от функции f(x)= $\frac{1.4+\sqrt{x+e^x}}{x+1}$ с точностью ε = 10^{-10} на отрезке [0; 5.1] с использованием метода правых треугольников. Провести параллельное вычисление интеграла на нескольких процессах и сравнить результаты.

Программа написана на языке C++ с использованием библиотеки OpenMP для параллельных вычислений. В программе используется функция f(x) для вычисления значения подынтегральной функции. Затем интеграл вычисляется методом правых треугольников на каждом процессе, результаты суммируются и выводятся на экран.

Приведём код написанной программы на языке C++:

```
#include <iostream>
#include <omp.h>
#include <cmath>
#include <iomanip>

using namespace std;

double f(double x) {
    // Подынтегральная функция
    return (1.4 + sqrt(x + pow(exp(1), x))) / (x + 0.7);
}

int main(int argc, char** argv) {
    double ans = 10.0735117837278;  // Ожидаемый результат интеграла
    const double a = 0, b = 5.1;
    const int n = 10000000;  // Общее количество подинтервалов
    double h = (b - a) / n;  // Шаг интегрирования
    double total_integral = 0.0;  // Общая сумма интеграла

    // Начало времени измерения
    double start_time = omp_get_wtime();

    // Распределение работы между потоками
    #pragma omp parallel for reduction(+:total_integral)
    for (int i = 0; i < n; ++i) {
        double x_mid = a + (i + 0.5) * h;  // Средняя точка подинтервала
        total_integral += f(x_mid);
    }

    total_integral *= h;  // Умножаем на шаг интегрирования

    // Вывод результатов
    cout << "Calculated Integral: " << setprecision(13) << total_integral <<
endl;
    cout << "Difference from expected: " << setprecision(13) << fabs(ans -
total_integral) << endl;
    cout << "Time taken: " << omp_get_wtime() - start_time << " seconds" <<
endl;

    return 0;
}
```

После выполнения программы на 1 процессе был получен результат:
Result: 10.07351178372
Time: 0.3930749893188
Вывод: Программа успешно реализует параллельное вычисление определенного интеграла с использованием MPI и метода правых треугольников. Проведенные вычисления показывают высокую точность и эффективность распараллеливания задачи.

Приведем результаты расчета программы для $n = 100000$:

```
Количество процессоров size =  1
integral  =   10.07351178372          time =0.3930749893188

Количество процессоров size =  2
integral  =   10.07351178372          time = 0.2035119533539

Количество процессоров size =  4
integral  =   10.07351178372          time =  0.1087310314178

Количество процессоров size =  5
integral  =   10.07351178372          time =0.1539080142975

Количество процессоров size =  10
integral  =   10.07351178372          time =0.1215319633484
```

Во всех запусках программа даёт верный результат вычисления интеграла.
Оценим ускорение $S_P = T_1 / T_p$ и эффективность $E_P = S_p / p$:

$$S_2 = \frac{T_1}{T_2} = \frac{0.39307}{0.20351} = 1.93 \qquad E_2 = \frac{S_2}{2} = \frac{1.93}{2} = 0.965$$

$$S_4 = \frac{T_1}{T_4} = \frac{0.39307}{0.10873} = 3.61 \qquad E_4 = \frac{S_4}{4} = \frac{3.61}{4} = 0.9025$$

$$S_5 = \frac{T_1}{T_5} = \frac{0.39307}{0.15390} = 2.55 \qquad E_5 = \frac{S_5}{5} = \frac{2.55}{5} = 0.51$$

$$S_{10} = \frac{T_1}{T_{10}} = \frac{0.39307}{0.12153} = 3.23 \qquad E_{10} = \frac{S_{10}}{10} = \frac{3.23}{10} = 0.323$$

Программа демонстрирует ускорение и эффективность при увеличении числа процессоров до определенного предела, после чего ускорение уменьшается. Например, при трех процессорах ускорение составляет примерно 3.61, что означает, что программа работает примерно в 3.61 раза быстрее, чем при одном процессоре. Однако при увеличении числа процессоров до 4 ускорение снижается до 2.55, а эффективность до 0.51. Это может быть связано с увеличением накладных расходов на управление процессами и коммуникацию между ними при большем числе процессоров. Но при увеличении числа до 10 ускорение растёт до 3.23, а эффективность до 0.323.