МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

**ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ. OPENMP. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ ПО МЕТОДУ ПРЯМОУГОЛЬНИКОВ ДЛЯ ДВУМЕРНОЙ ФУНКЦИИ.**

ЛАБОРАТОРНАЯ РАБОТА №2

студента 4 курса 451 группы

направления 09.03.04 — Программная инженерия

факультета КНиИТ

Голикова Артема Олеговича

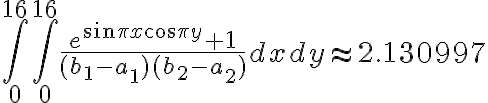
Проверил

канд. физ-мат. н., доцент \_\_\_\_\_\_\_\_\_\_\_\_ К. П. Савина

Саратов 2020

**Цель работы:** изучить использование потоков OpenMP (директивы #pragma omp paralel) и операции редукции, приводящих к уменьшению вычисления интегралов для двумерной функции на примере метода прямоугольников; сравнить время выполнения алгоритмов численного интегрирования с использованием разных режимов и разных способов распараллеливания.

**Ход работы:**



Код программы реализующий метод прямоугольников для последовательной реализации:

// Work02, последовательная версия

#include <time.h>

#include <iostream>

#define PI 3.1415926535897932384626433832795

using namespace std;

void integral(const double a1, const double b1, const double a2, const double b2,

const double h, double\* res)

{

int i, n, m;

double sum; // локальная переменная для подсчета интеграла

double x, y; // координата точки сетки

n = (int)((b1 - a1) / h); // количество точек сетки интегрирования

m = (int)((b2 - a2) / h);

sum = 0.0;

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

{

x = a1 + i \* h + h / 2.0;

for (int j = 0; j < m; j++)

{

y = a1 + j \* h + h / 2.0;

sum += (exp(sin(PI \* x) \* cos(PI \* y)) + 1) \* h \* h / (b1 - a1) / (b2 - a2);

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

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

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

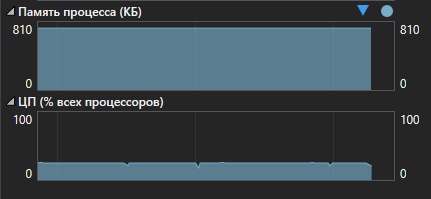
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

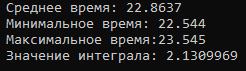
return 0;

}

Использование ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод прямоугольников для параллельной реализации (1-ый способ):

// Work02, параллельная версия 1-ый способ

#include <time.h>

#include <iostream>

#define PI 3.1415926535897932384626433832795

using namespace std;

void integral(const double a1, const double b1, const double a2, const double b2,

const double h, double\* res)

{

int i, n, m;

double sum; // локальная переменная для подсчета интеграла

double x, y; // координата точки сетки

n = (int)((b1 - a1) / h); // количество точек сетки интегрирования

m = (int)((b2 - a2) / h);

sum = 0.0;

#pragma omp parallel for private(x) reduction (+: sum)

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

{

x = a1 + i \* h + h / 2.0;

for (int j = 0; j < m; j++)

{

y = a1 + j \* h + h / 2.0;

sum += (exp(sin(PI \* x) \* cos(PI \* y)) + 1) \* h \* h / (b1 - a1) / (b2 - a2);

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

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

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

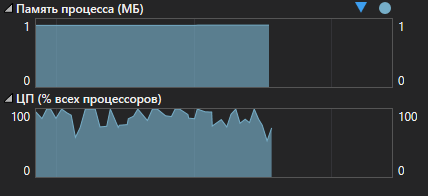
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

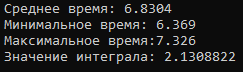
return 0;

}

Использование ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод прямоугольников для параллельной реализации (2-ой способ):

// Work02, параллельная версия 2-ой способ

#include <time.h>

#include <iostream>

#define PI 3.1415926535897932384626433832795

using namespace std;

void integral(const double a1, const double b1, const double a2, const double b2,

const double h, double\* res)

{

int i, n, m;

double sum; // локальная переменная для подсчета интеграла

double x, y; // координата точки сетки

n = (int)((b1 - a1) / h); // количество точек сетки интегрирования

m = (int)((b2 - a2) / h);

sum = 0.0;

#pragma omp parallel for private(x) reduction (+: sum)

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

{

x = a1 + i \* h + h / 2.0;

#pragma omp parallel for private(y)

for (int j = 0; j < m; j++)

{

y = a1 + j \* h + h / 2.0;

sum += (exp(sin(PI \* x) \* cos(PI \* y)) + 1) \* h \* h / (b1 - a1) / (b2 - a2);

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

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

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

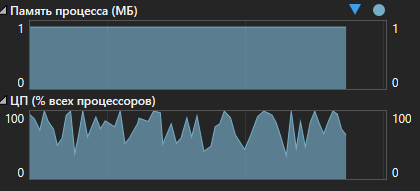
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

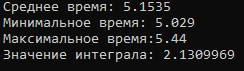
return 0;

}

Использования ЦП во время выполнения:



Результат выполнения:



Сравнение результатов:

|  |  |  |  |
| --- | --- | --- | --- |
| Последовательная  реализация | Параллельная  Реализация  (1-ый способ) | Параллельная  Реализация  (2-ой способ) | Ускорение  Посл./Пар. (1 способ)  Посл./Пар. (2 способ)  Пар.(1 сп.)/Пар.(2 сп.) |
| Мин. время: 22,544  Макс. время: 23,545  Сред. время: 22,8637 | Мин. время: 6,369  Макс. время: 7,326  Сред. время: 6,8304 | Мин. время: 5,029  Макс. время: 5,44  Сред. время: 5,1535 | 3,3473  4,4365  1,3254 |

Вычисления происходили на процессоре Intel Core i5-2310 (4 ядра, 4 потока). По результатам сравнения видно, что распараллеливание при вычислении интегралов для двумерной функции дает прирост по быстродействию в 3,3 - 4,4 раза, в зависимости от способа распараллеливания. Распараллеливание вторым способом дает прирост примерно 30% к быстродействию в сравнении с первым способом.

**Вывод:** изучил использование потоков OpenMP (директивы #pragma omp paralel) и операции редукции, приводящих к уменьшению вычисления интегралов для двумерной функции на примере метода прямоугольников; сравнил время выполнения алгоритмов численного интегрирования с использованием разных режимов и разных способов распараллеливания.