Московский государственный университет им. М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет по заданию №1

Разработка параллельной версии программы для вычисления определенного интеграла с использованием метода прямоугольников.

Савицкий Илья

321 группа

Москва – 2021

Постановка задачи

- Написать последовательный алгоритм подсчета интеграла методом прямоугольников
- Распараллелить его
- Запустить на суперкомпьютере с разными параметрами и сделать выводы о времени его выполнения

Текст параллельного алгоритма

```
#include <math.h>
#include <omp.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

typedef double (*math_func)(double);

double calculate_integral(double from, double to, int n, math_func func)
{
    int i;
    double h = (to - from) / (double)n;
    double sum = 0.0, x;
    #pragma omp parallel for reduction(+ : sum) private(x)
    for (i = 1; i <= n; ++i) {
        x = h * ((double)i - 0.5);
        sum += func(x);
    }
    return sum * h;
}

int main(int argc, char *argv[]) {
    double start, stop;
    start = omp_get_wtime();
    int n = strtol(argv[1], NULL, 10);
    double res = calculate_integral(0, 10000, n, sqrt);
    stop = omp_get_wtime();
    printf("%d,%lf,%d,%lf\n", omp_get_max_threads(), res, n,
stop-start);
    return 0;
}
```

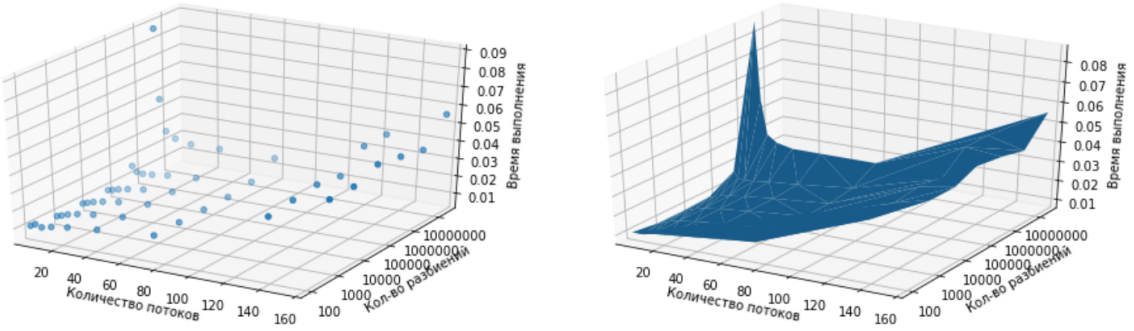
Сам алгоритм берет в себя интервал интегрирования, количество разбиение интервала и функцию, которую требуется проинтегрировать. После вычисления ширины одного прямоугольника, алгоритм проходится по интервалу этими прямоугольниками и складывает все их площади в переменную `sum`. Замер времени производится при помощи встроенной в `omp` функции `omp_get_wtime()`, после чего основная информация о выполнении программы выводится в поток вывода.

Замер времени работы программы

Для проверки работы алгоритма были выбраны следующие параметры:

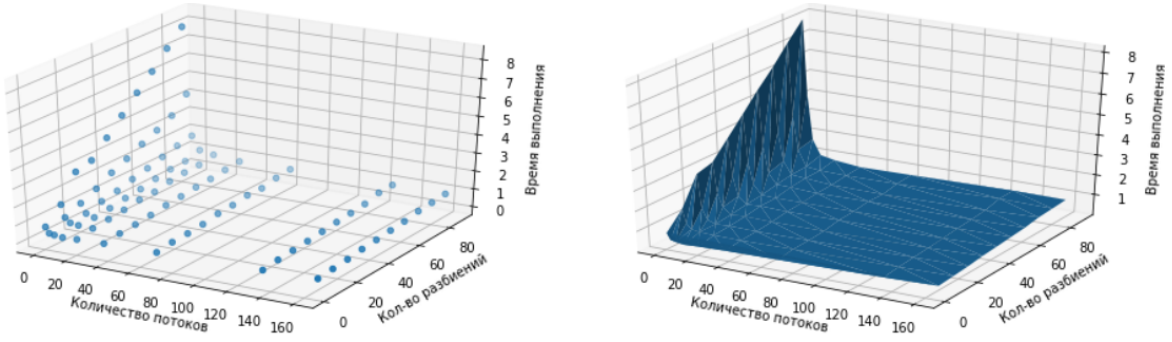
- Набор кол-ва тредов: 1, 2, 4, 8, 16, 32, 64, 128, 160(как максимально доступное)
- Два набора количеств разбиений:
  - поменьше, логарифмический: 100, 1000, 10000, 100000, 1000000, 10000000
  - побольше, линейный: 100000000, 200000000, 500000000, 300000000, 400000000, 600000000, 700000000, 800000000, 900000000, 1000000000
- Функция для которой вычисляется интеграл - `sqrt(x)`

Тогда для первого набора разбиений получается следующий график(шкала x логарифмическая):



Видно, что график выравнивается на на ~60 потоках, после чего время выполнения начинает только увеличиваться.

Однако, этот эффект пропадает на бóльшем наборе разбиений(к сожалению, из-за бага в matplotlib некорректно отображается кол-во разбиений, однако оно тут такое, как указано выше):



Как видно, на большом количестве данных не наблюдается роста на большом количестве процессов