Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по курсу
«Суперкомпьютерное моделирование и технологии»

«Численное интегрирование многомерных функций методом Монте-Карло»

Вариант 6

Студент 608 группы
И. А. Кулешов

Москва, 2022

1 Постановка задачи и методы решения

1.1 Математическая постановка задачи

Функция $f(x, y, z)$ — непрерывна в ограниченной замкнутой области $G \subset R^3$. Требуется вычислить определённый интеграл (вариант 6):

$$I = \iiint\limits_G \sin(x^2 + z^2) \cdot \ y \ dx\, dy\, dz,$$

где область $G = \{(x, y, z) : x^2 + y^2 + z^2 \leqslant 1, x > 0, y > 0, z > 0\}$.

1.2 Вычисление точного значения

$$I = \iiint\limits_G \sin(x^2 + z^2) \cdot \ y \ dx\, dy\, dz = \Big\{x = r\cos(\phi), y = y, z = r\sin(\phi),\ dx\, dy\, dz \ = r\, d\phi\, dr\, dy\Big\}$$
$$= \int_0^{\frac{\pi}{2}} d\phi \int_0^1 r\sin(r^2)\, dr \int_0^{\sqrt{1-r^2}} y\, dy = \int_0^{\frac{\pi}{2}} d\phi \int_0^1 r\ \sin(r^2)\frac{1-r^2}{2}\, dr = \Big\{\gamma = r^2, dr = \frac{d\gamma}{2r}\Big\}$$
$$= \frac{\pi}{8}\int_0^1 (1-\gamma)\sin(\gamma) = \frac{\pi}{8}\Big(1 - \sin(1)\Big) \approx 0.06225419868$$

1.3 Численный метод решения

Пусть область G ограниченна параллелепипедом: $\Pi = \begin{cases} a_1 \leqslant x \leqslant b_1 \\ a_2 \leqslant y \leqslant b_2 \\ a_3 \leqslant z \leqslant b_3 \end{cases}$

Рассмотрим функцию $F(x, y, z) = \begin{cases} f(x, y, z) & (x, y, z) \in G, \\ 0 & (x, y, z) \notin G, \end{cases}$

Преобразуем искомый интеграл:

$$I = \iiint\limits_G f(x, y, z)\ dx\, dy\, dz = \iiint\limits_\Pi F(x, y, z)\ dx\, dy\, dz$$

Пусть $p_1(x_1, y_1, z_1)$, $p_2(x_2, y_2, z_2)$, ... — случайные точки, равномерно распределённые в Π. Возьмём n таких случайных точек. В качестве приближённого значения интеграла предлагается использовать выражение:

$$I \approx |\Pi| \cdot \frac{1}{n}\sum_{i=1}^{n} F(p_i),$$

где $|\Pi|$ - объём параллелепипеда Π. $|\Pi| = (b_1 - a_1)(b_2 - a_2)(b_3 - a_3) = (1-0)(1-0)(1-0) = 1$

Вариант 6 предлагает метод распараллеливания независимой генерацией точек MPI- процессами для решения задачи.

2 Алгоритм решения

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

int main(int argc, char *argv[]) {


    const double pi = 4.0 * atan(1.0);
    // 0.06225419868
    double exact_integral_value = pi / 8.0 * (1 - sin(1.0));
    double P_region_volume = 1.0 * 1.0 * 1.0;

    double integral_approx;
    double x_rand, y_rand, z_rand;
    double func_sum;
    int n_dots_per_iter = 100;
    int n_dots = 0;
    double cur_iteration_sum;
    int iterations = 0;
    double total_sum = 0.0;
    double error;

    double eps;
    double max_duration;

    int num_procs, my_rank, root = 0;

    if (argc > 1 && argv[1] != NULL) {
        sscanf(argv[1], "%lf", &eps);
    } else {
        fprintf(stderr,"eps is missing!\n");
        return 1;
    }

    int seed_bias = 0;
    if (argc > 2 && argv[2] != NULL) {
        sscanf(argv[2], "%d", &seed_bias);
    } else {
        fprintf(stderr,"seed_bias is missing!\n");
        return 2;
    }

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
```

```
    int my_seed = seed_bias * 100 + my_rank;
    srand(my_seed);

    double start = MPI_Wtime();

    do {
        iterations += 1;
        func_sum = 0;
        for (size_t i = 0; i < n_dots_per_iter; ++i) {
            x_rand = (double) rand() / RAND_MAX;
            y_rand = (double) rand() / RAND_MAX;
            z_rand = (double) rand() / RAND_MAX;
            if ((x_rand*x_rand + y_rand*y_rand + z_rand*z_rand) > 1) {
                func_sum += 0;
            } else {
                func_sum += sin(x_rand*x_rand + z_rand*z_rand)*y_rand;
            }
        }

        MPI_Reduce(&func_sum, &cur_iteration_sum, 1,
                   MPI_DOUBLE, MPI_SUM, root, MPI_COMM_WORLD);

        if (my_rank == root) {
            n_dots += n_dots_per_iter;
            total_sum += (1.0 / num_procs) * cur_iteration_sum;
            integral_approx = P_region_volume * (1.0/n_dots)*total_sum;
            error = fabs(integral_approx - exact_integral_value);
        }
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Bcast(&error, 1, MPI_DOUBLE, root, MPI_COMM_WORLD);
    } while (error > eps);


    double cur_rank_duration = MPI_Wtime() - start;

    MPI_Reduce(&cur_rank_duration, &max_duration, 1,
               MPI_DOUBLE, MPI_MAX, root, MPI_COMM_WORLD);
    MPI_Finalize();
    if (my_rank == root) {
        printf("num_procs: %d, eps: %.10f, time: %.10f sec,"
               " integral approx value: %.10f, error: %.10f ,"
               " dots generated: %d, seed_bias: %d\n",
               num_procs, eps, max_duration, integral_approx,
               error, num_procs * n_dots, seed_bias);
    }
    return 0;
}
```

3 Результатами расчётов для системы Polus

Для уменьшения фактора влияния псевдослучайности на результат были проведены 4 серии измерений с разными параметрами seed функции srand().
После этого было произведено медианнное усреднение, дающее устойчивость к возможным выбросам, вызванных неудачной генерацией точек.

3.1 Таблица усредненных результатов

Точность	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	1	0.0024093	1.000	0.00001947
	4	0.0040887	1.697	0.00002311
	16	0.0027297	1.133	0.00000924
	32	0.0023540	0.977	0.00002012
$5.0 \cdot 10^{-6}$	1	0.0045879	1.000	0.00000317
	4	0.0171443	3.737	0.00000394
	16	0.0040081	0.874	0.00000298
	32	0.0041637	0.908	0.00000196
$1.5 \cdot 10^{-6}$	1	0.0066012	1.000	0.00000106
	4	0.0179275	2.716	0.00000070
	16	0.0103402	1.566	0.00000095
	32	0.0059100	0.895	0.00000061

3.2 Графики усредненных результатов ускорения

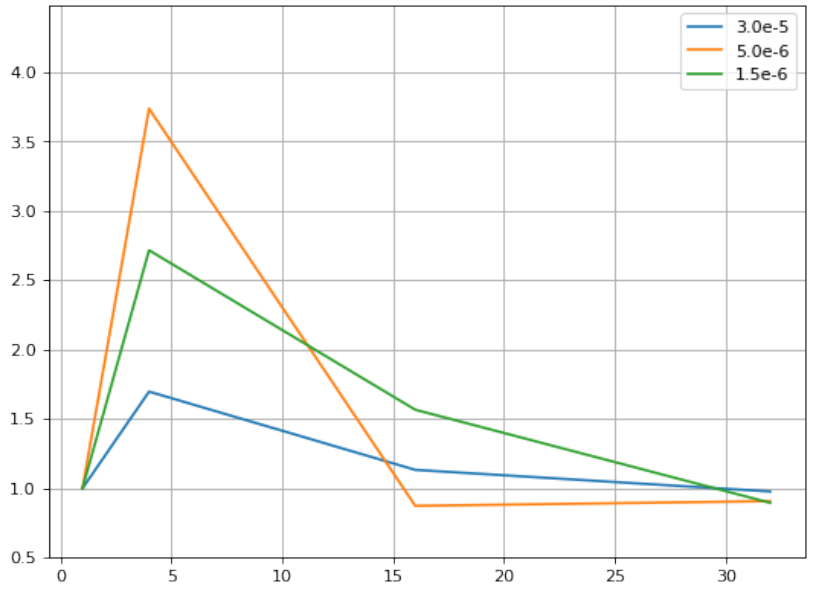

Рис. 1: График зависимости ускорения от числа процессов на Polus