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

Орлова Анна Олеговна, 614 группа

Вариант 1

Отчёт по заданию в рамках курса

«Суперкомпьютерное моделирование и технологии»

Численное интегрирование многомерных функций методом Монте-Карло

Москва
2022

1 Математическая постановка задачи и численный метод решения

Функция f(x, y, z) — непрерывна в ограниченой замкнутой области $G \subset \mathbb{R}^3$.

Требуется вычислить определённый интеграл:

$$I = \iiint_G xy^2z^3\,dx\,dy\,dz$$

,

где область $G = \{(x, y, z) : z = xy, y = x, x = 1, z = 0\}$

Пусть область G ограничена параллипипедом V:

$$\begin{cases} a_1 = 0 \le x \le 1 = b_1, \\ a_2 = 0 \le y \le 1 = b_2, \\ a_3 = 0 \le z \le 1 = b_3. \end{cases}$$

Объем параллипипеда V равен 1.

2 Нахождение точного значения интеграла аналитически

$$I = \int_0^1 dx \int_0^x dy \int_0^{xy} xy^2z^3\,dz = \int_0^1 x\,dx \int_0^x y^2\,dy \int_0^{xy} z^3\,dz = \frac{1}{4}\int_0^1 x\,dx \int_0^x y^2x^4y^4\,dy =$$

$$= \frac{1}{4}\int_0^1 x\,dx \int_0^x y^6x^4\,dy = \frac{1}{4}\int_0^1 x^5\,dx \int_0^x y^6\,dy = \frac{1}{4}\frac{1}{7}\int_0^1 x^5x^7\,dx = \frac{1}{28}\int_0^1 x^{12}\,dx =$$

$$= \frac{1}{28}\frac{1}{13} = \frac{1}{364} \approx 0.0027472527472527475$$

3 Описание программной реализации

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

double generate(x, y)
{
    return ((double)rand()/(double)(RAND_MAX)) * (y - x) + x;
}

int main(int argc, char *argv[])
{
    int size, rank;
    long N = 0, i = 0;
    double integral = 1.0/364.0, epsilon = strtod(argv[1], NULL);
    double a1 = 0.0, a2 = 1.0, b1 = 0.0, b2 = 1.0, c1 = 0.0, c2 = 1.0;
    int par = 1, root = 0;
    double x, y, z;
    double sum = 0.0, f_sum, func, int_appr = 0.0, error = 0.0, sum_2 = 0;
    double time_start, time_end, ttime, timee;
    int seed, M = 0; // seeds = {1, 13, 20, 614, 1300, 2022, 809, 5575, 5555}

    sscanf(argv[1], "%lf", &epsilon);
    sscanf(argv[2], "%d", &seed);
    sscanf(argv[3], "%d", &M);

    double buff[M];

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

    srand(seed + rank);

    time_start = MPI_Wtime();

    while (1)
    {
        N++;

        for(int j = 0; j < M; j++)
        {
            x = generate(a1, a2);
            y = generate(b1, b2);
            z = generate(c1, c2);

            if (a1 <= x <= a2 && b1 <= y <= b2 && c1 <= z <= c2 && x * y >= z && y <= x)
            {
                func = x*pow(y, 2)*pow(z, 3);
                i++;
            }
```

```
            else
            {
                func = 0.0;
            }

            buff[j] = func;
        }

        MPI_Reduce(&buff, &f_sum, M, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

        if (rank == root)
        {
            sum = sum + f_sum/((double)size);
            int_appr = par * sum/ ((double)N);
            error = fabs(integral - int_appr);
        }

        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Bcast(&error, 1, MPI_DOUBLE, root, MPI_COMM_WORLD);

        if (error < epsilon)
            {
                break;
            }
    }

    time_end = MPI_Wtime();
    timee = time_end - time_start;
    MPI_Reduce(&timee, &ttime, 1, MPI_DOUBLE, MPI_MAX, root, MPI_COMM_WORLD);

    MPI_Finalize();

    if (rank == root)
    {
        printf("Integral approximate = %f\n", int_appr);
        printf("Integral numerical = %f\n", integral);
        printf("Error:  %e   %.10f\n", error, error);
        printf("Epsilon:  %.10f\n", epsilon);
        printf("Seed:  %d\n", seed);
        printf("M size (buff):  %d\n", M);
        printf("Process number (all points) N*size:  %ld\n", M*N*size);
        printf("Max time:  %.8f\n", ttime);
        printf("Size: %d\n", size);
    }
    return 0;
}
```

4 Исследование масштабираемости программы

Точность ε	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	1	0.17324641	1	0.0000246969
	4	0.06495596	2.667	0.0000165031
	16	0.05870321	2.951	0.0000162982
	32	0.05654092	3.064	0.0000089105
$5.0 \cdot 10^{-6}$	1	0.17718153	1	0.0000012058
	4	0.06353680	2.789	0.0000006679
	16	0.05835385	3.036	0.0000036487
	32	0.03297861	5.374	0.0000037834
$1.5 \cdot 10^{-6}$	1	0.17705425	1	0.0000012058
	4	0.06126828	2.890	0.0000006679
	16	0.04609713	3.841	0.0000013381
	32	0.04144036	4.273	0.0000000727

Рис. 1: Значение seed = 5555

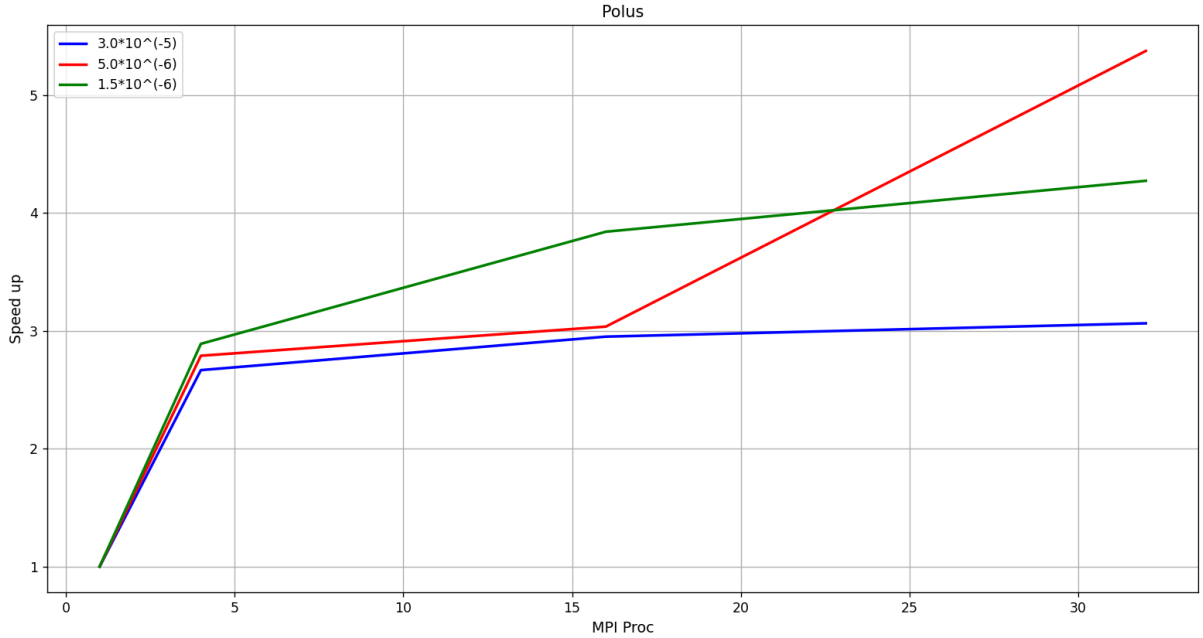

Рис. 2: Зависимость ускорения от числа MPI - процессов (значение seed = 5555)