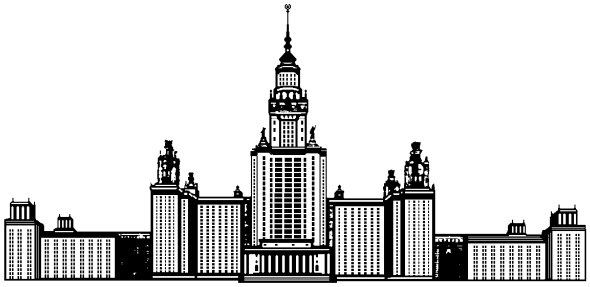

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра информационной безопасности

Мирпулатов Исломбек Пулат-угли

Численное интегрирование многомерных функций методом Монте-Карло

Суперкомпьютерное моделирование и технологии

Группа 608, Вариант 1

Москва, 2022

1 Постановка задачи

В качестве модельной задачи предлагается задача вычисления многомерного интеграла методом Монте-Карло. Программная реализация должна быть выполнена на языке C или C++ с использованием библиотеки параллельного программирования MPI.

Требуется исследовать масштабируемость параллельной MPI-программы на следующих параллельных вычислительных системах ВМК МГУ:

1. IBM Blue Gene/P,
2. IBM Polus

1.1 Математическая постановка задачи

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

Требуется вычислить определенный интеграл:

$$I = \iiint_G xy^2z^3 dxdydz$$

где область G ограничена поверхностями $z = xy, y = x, x = 1, z = 0$.

1.2 Численный метод решения задачи

Метод Монте-Карло для численного интегрирования.

Пусть область G ограниченна параллелепипедом: $\Pi : \begin{cases} 0 \leqslant x \leqslant 1 \\ 0 \leqslant y \leqslant 1 \\ 0 \leqslant z \leqslant 1 \end{cases}$

Рассмотрим функцию: $F(x, y, z) = \begin{cases} xy^2z^3, & (x, y, z) \in G \\ 0, & (x, y, z) \notin G \end{cases}$

Преобразуем искомый интеграл:

$$I = \iiint_G xy^2z^3 dxdydz = \iiint_\Pi F(x, y, z) dxdydz$$

Пусть $p_1 (x_1, y_1, z_1), p_2 (x_2, y_2, z_2), \ldots$ - 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. $|\Pi| = 1$

Вариант 1 подразумевает независимую генерацию точек MPI-процессами.

2 Решение

2.1 Аналитическое решение

$$I = \iiint_G xy^2z^3dxdydz$$

$$G = \{(x, y, z) : x \in [0, 1], y \in [0, x], z \in [0, xy]\}$$

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

$$= \int_0^1 dx \int_0^x \frac{xy^2z^4dz}{4}|_0^{xy}dy = \int_0^1 dx \int_0^x \frac{x^5y^6}{4}dy =$$

$$= \int_0^1 \frac{x^5y^7}{28}|_0^x dx = \int_0^1 \frac{x^{12}}{28}dx = \frac{x^{13}}{364}|_0^1 = \frac{1}{364}$$

2.2 Описание программной реализации

Реализуем адаптивный по точности алгоритм Монте-Карло численного интегрирования.

Программа на вход принимает требуемую точность EPS. Задаем кол-во процессов M.

На каждом процессе генерируется $\frac{N}{M}$ точек (x, y, z) из параллелепипеда (в моем случае куб $[0, 1]^3$). Для каждой точки проверяем попала ли в заданную область : $x \geq y$ и $xy \geq z$. Если да, то к локальной сумме добавляется значение xy^2z^3, в противном случае 0. Далее все локальные суммы через MPI_Allreduce объединяются в общую сумму. Эта сумма делится на количество точек N.

Проверяем, если разница между решением полученным аналитическим путем и нашим значением из предыдущего пункта не первышает требуемой точности, то останавливаемся. В обратном случае повторяем предыдущий пункт.

Оценка погрешности метода Монте-Карло:

$$EPS \approx O(\frac{1}{\sqrt{N}})$$

Отсюда следует, что можно вычислить примерное N:

$$N \approx \frac{1}{EPS^2}$$

Но так как это число получается слишком большое, было решено выбрать:

$$N \approx \frac{1000}{EPS}$$

Этого количества оказалось достаточно.

В результате работы программы выводится:

- Посчитанное приближённое значение интеграла.
- Ошибка посчитанного значения.
- Количество сгенерированных случайных точек.
- Время работы программы в секундах.

2.3 Программная реализация

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

double EXACT = 0.002747253;

int main(int argc, char **argv)
{
    int psize, prank, ierr;
    double result, eps;
    double x, y, z;


    eps = strtof(argv[1], NULL);

    ierr = MPI_Init(&argc, &argv);
    ierr = MPI_Comm_size(MPI_COMM_WORLD, &psize);
    ierr = MPI_Comm_rank(MPI_COMM_WORLD, &prank);

    srand(prank * 13);

    double time_elapsed = MPI_Wtime();

    double global_sum, local_sum = 0.0;
    long n = 1000 / (eps);
    long batch = n / psize;

    long i, it = 0;
    do
    {
        it++;
        for (i = 0; i < batch; i++)
        {
            x = (double) rand() / RAND_MAX;
            y = (double) rand() / RAND_MAX;
            z = (double) rand() / RAND_MAX;
            if ((x >= y) && (x * y >= z))
```

```
            {
                local_sum = local_sum + x * y * y * z * z * z;
            }
        }

        ierr = MPI_Allreduce(&local_sum, &global_sum, 1, MPI_DOUBLE, MPI_SUM,
                             MPI_COMM_WORLD);

        result = global_sum / (batch * psize * it);

    }
    while(eps < fabs(EXACT - result));


    time_elapsed = MPI_Wtime() - time_elapsed;

    if (prank == 0)
    {
        double error = fabs(EXACT - result);
        printf("result:\t%.10lf\n", result);
        printf("error:\t%.10lf\n", error);
        printf("iters:\t%ld\n", it * n);
        printf("time:\t%.10lf\n", time_elapsed);
    }

    ierr = MPI_Finalize();
    return 0;
}
```

2.4 Исследование масштабируемости программы на системах Blue Gene/P и Polus

Таблица 1. Таблица с результатами расчётов для системы Polus

Точность EPS	Число MPI процессов	Время работы программы	Ошибка	Ускорение
$3.0 \cdot 10^{-5}$	1	3.761	0.0000017742	1.00
$3.0 \cdot 10^{-5}$	4	0.952	0.0000005586	3.95
$3.0 \cdot 10^{-5}$	16	0.435	0.0000038100	8.65
$3.0 \cdot 10^{-5}$	32	0.344	0.0000046010	10.93
$5.0 \cdot 10^{-6}$	1	22.482	0.0000012201	1.00
$5.0 \cdot 10^{-6}$	4	5.718	0.0000007715	3.93
$5.0 \cdot 10^{-6}$	16	1.616	0.0000010205	13.91
$5.0 \cdot 10^{-6}$	32	0.875	0.0000025367	25.69
$1.5 \cdot 10^{-6}$	1	74.735	0.0000013346	1.00
$1.5 \cdot 10^{-6}$	4	19.122	0.0000009006	3.91
$1.5 \cdot 10^{-6}$	16	4.954	0.0000001053	15.08
$1.5 \cdot 10^{-6}$	32	2.777	0.0000002819	26.91

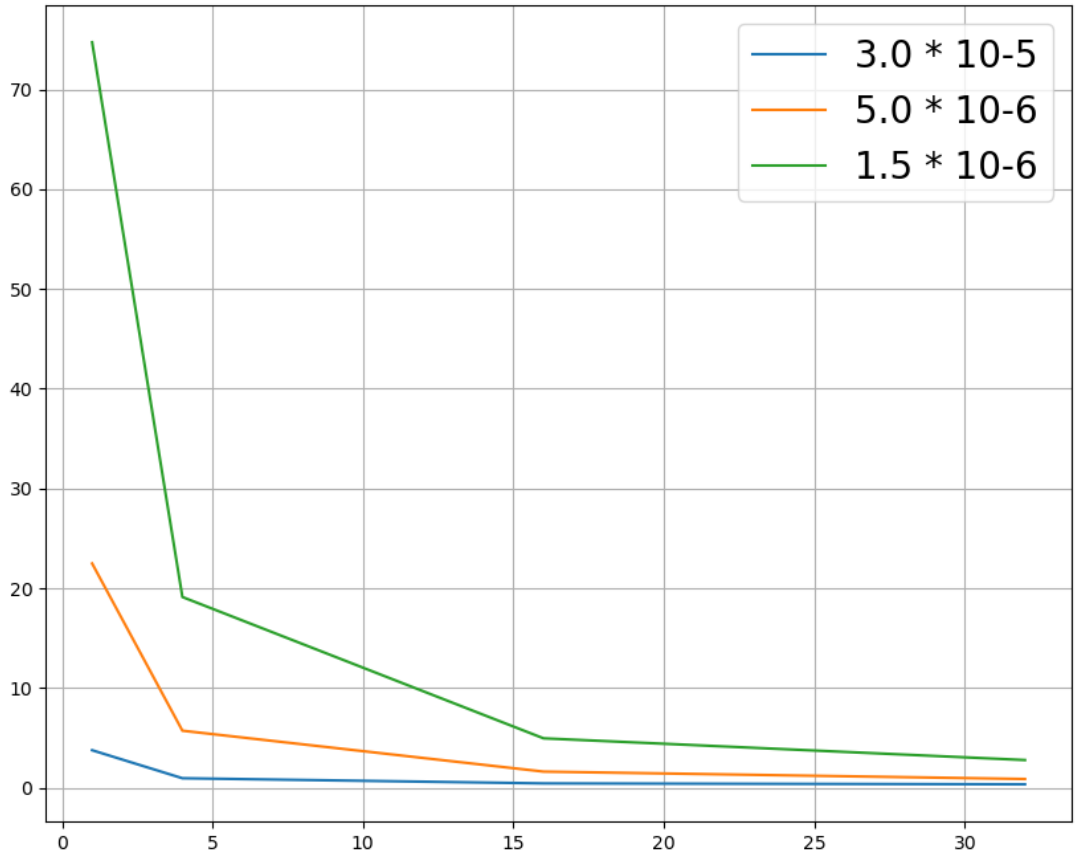

График 1. Зависимость времени работы от количества процессов