



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного автономного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
**(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №6

«Распараллеливание алгоритмов задач, решаемых сеточными методами»

ДИСЦИПЛИНА: «Параллельные процессы в информационных системах»

Выполнил: студент гр. ИУК4-31М

(Сафонов Н.С.)

(Подпись)

(Ф.И.О.)

Проверил:

(Корнюшин Ю.П.)

(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2025

Цель: формирование практических навыков распараллеливания алгоритмов задач, решаемых сеточными методами в MPI на примере решения задачи Пуассона в трехмерной области.

Задачи

1. Освоить способы декомпозиции данных, связанных с разными способами распределения данных по компьютерам.
2. Построить параллельный алгоритм решения задачи Пуассона методом Зейделя в MPI.
3. Сравнить временные характеристики алгоритма решения задачи Пуассона на одном и нескольких компьютерах.

Задание

Вариант 6

Задание 1

Внимательно изучить пример данной лабораторной работы.
Скомпилировать и запустить на 1-м процессоре.

Задание 2

Разработать параллельный алгоритм, написать и отладить параллельную программу решения задачи Пуассона методом Зейделя в MPI.

Рекомендация: параллельный алгоритм реализовать на «линейке» компьютеров.

Задание 3

Разработанный и отлаженный параллельный алгоритм проверить, например, для функции $f = X^2 + Y^2 + Z^2$.

Задание 4

В качестве параллельной программы взять вами созданную параллельную программу. В качестве последовательной программы, взять последовательную программу из примера предыдущего пункта. Обе программы запустить и засечь время счета для одного и того же пространства размером согласно варианту (таблица 1).

Вариант	Размерность	Кол-во компьютеров
6	[35 x 35 x 35]	1, 2, 5

Результат выполнения работы

Задание 1

```
PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_6> mpiexec -n 1 .\bin\task_1.exe 20 20 20
0
in = 20, jn = 20, kn = 20
Iter = 351, Eps = 1.000000e-05, Time = 0.047538 ms
Max differ = 3.452231e-04 at point (9, 9, 9)
PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_6>
```

Рисунок 1 – Результат выполнения примера 1

Листинг

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <sys/time.h>

#define a 1

double Fresh(double x, double y, double z) {
    return x + y + z;
}

double Ro(double x, double y, double z) {
    return -a * (x + y + z);
}

int main(int argc, char** argv) {
    if (argc < 4) {
        fprintf(stderr, "Usage: %s <in> <jn> <kn>\n", argv[0]);
        return 1;
    }

    int in = atoi(argv[1]);
    int jn = atoi(argv[2]);
    int kn = atoi(argv[3]);

    if (in <= 0 || jn <= 0 || kn <= 0) {
        fprintf(stderr, "Error: all dimensions must be positive integers.\n");
        return 1;
    }

    double *F_data = (double*)calloc((in + 1) * (jn + 1) * (kn + 1),
sizeof(double));
    if (!F_data) {
        perror("Failed to allocate F");
        return 1;
    }

#define F(i, j, k) F_data[(i)*(jn+1)*(kn+1) + (j)*(kn+1) + (k)]
```

```

double X = 2.0, Y = 2.0, Z = 2.0;
double e = 1e-5;

double hx = X / in;
double hy = Y / jn;
double hz = Z / kn;

double owx = hx * hx;
double owy = hy * hy;
double owz = hz * hz;

double c = 2.0 / owx + 2.0 / owy + 2.0 / owz + a;

for (int i = 0; i <= in; i++) {
    for (int j = 0; j <= jn; j++) {
        for (int k = 0; k <= kn; k++) {
            if (i == 0 || i == in || j == 0 || j == jn || k == 0 || k == kn)
{
                F(i, j, k) = Fresh(i * hx, j * hy, k * hz);
            } else {
                F(i, j, k) = 0.0;
            }
        }
    }
}

struct timeval tv1, tv2;
gettimeofday(&tv1, NULL);

int it = 0;
int converged = 0;

while (!converged) {
    converged = 1;
    for (int i = 1; i < in; i++) {
        for (int j = 1; j < jn; j++) {
            for (int k = 1; k < kn; k++) {
                double F_old = F(i, j, k);
                double Fi = (F(i+1, j, k) + F(i-1, j, k)) / owx;
                double Fj = (F(i, j+1, k) + F(i, j-1, k)) / owy;
                double Fk = (F(i, j, k+1) + F(i, j, k-1)) / owz;
                F(i, j, k) = (Fi + Fj + Fk - Ro(i * hx, j * hy, k * hz)) /
c;

                if (fabs(F(i, j, k) - F_old) > e) {
                    converged = 0;
                }
            }
        }
    }
    it++;
}

gettimeofday(&tv2, NULL);
double elapsed_sec = (tv2.tv_sec - tv1.tv_sec) + (tv2.tv_usec - tv1.tv_usec) /
1000000.0;

double max_diff = 0.0;
int mi = 0, mj = 0, mk = 0;

for (int i = 1; i < in; i++) {
    for (int j = 1; j < jn; j++) {
        for (int k = 1; k < kn; k++) {
            double exact = Fresh(i * hx, j * hy, k * hz);
            double diff = fabs(F(i, j, k) - exact);

```

```

        if (diff > max_diff) {
            max_diff = diff;
            mi = i; mj = j; mk = k;
        }
    }
}

printf("\nin = %d, jn = %d, kn = %d\n", in, jn, kn);
printf("Iter = %d, Eps = %e, Time = %.6f ms\n", it, e, elapsed_sec);
printf("Max differ = %e at point (%d, %d, %d)\n", max_diff, mi, mj, mk);

free(F_data);
return 0;
}

```

Задания 2 и 3

```

PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_5> mpiexec -n 8 .\bin\task_2.exe 8
Solution x:
x[0] = 2.000000
x[1] = 1.000000
x[2] = 1.000000
x[3] = 1.000000
x[4] = 1.000000
x[5] = 1.000000
x[6] = 1.000000
x[7] = 1.000000
Total execution time: 0.003925 seconds

```

Рисунок 2 – Результат выполнения заданий 2 и 3

Листинг

```

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

#define X 2.0
#define Y 2.0
#define Z 2.0
#define EPS 1e-5

double Fresh(double x, double y, double z) {
    return x*x + y*y + z*z;
}

double Ro(double x, double y, double z) {
    return 6.0;
}

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (argc < 4) {
        if (rank == 0) {
            fprintf(stderr, "Usage: %s <in> <jn> <kn>\n", argv[0]);
        }
        MPI_Finalize();
        return 1;
    }

    if (rank == 0) {
        FILE *in = fopen(argv[1], "r");
        FILE *jn = fopen(argv[2], "w");
        FILE *kn = fopen(argv[3], "w");
        if (!in || !jn || !kn) {
            perror("Error opening files");
            MPI_Abort(MPI_COMM_WORLD, 1);
        }
    }
}
```

```

}

int in = atoi(argv[1]);
int jn = atoi(argv[2]);
int kn = atoi(argv[3]);

if (in <= 0 || jn <= 0 || kn <= 0) {
    if (rank == 0) {
        fprintf(stderr, "Error: all dimensions must be positive
integers.\n");
    }
    MPI_Finalize();
    return 1;
}

double hx = X / in;
double hy = Y / jn;
double hz = Z / kn;
double owx = hx * hx;
double owy = hy * hy;
double owz = hz * hz;
double c = 2.0/owx + 2.0/owy + 2.0/owz;

int local_i = in / size;
int remainder = in % size;
if (rank < remainder) local_i++;
int i_start = (in / size) * rank + (rank < remainder ? rank : remainder);
int i_end = i_start + local_i - 1;

double (*F)[jn+1][kn+1] = malloc((local_i + 2) * sizeof(*F));
double (*F_old)[jn+1][kn+1] = malloc((local_i + 2) * sizeof(*F));

for (int i = 0; i < local_i + 2; i++)
    for (int j = 0; j <= jn; j++)
        for (int k = 0; k <= kn; k++)
            F[i][j][k] = 0.0;

for (int i = 0; i < local_i + 2; i++) {
    int i_global = i_start + i - 1;
    for (int j = 0; j <= jn; j++) {
        for (int k = 0; k <= kn; k++) {
            double x = (i_global >= 0 && i_global <= in) ? i_global * hx :
0.0;
            double y = j * hy;
            double z = k * hz;

            if (i_global == 0 || i_global == in || j == 0 || j == jn || k ==
0 || k == kn) {
                F[i][j][k] = Fresh(x, y, z);
            }
        }
    }
}

if (rank == 0) {
    for (int j = 0; j <= jn; j++)
        for (int k = 0; k <= kn; k++)
            F[0][j][k] = Fresh(0.0, j*hy, k*hz);
}
if (rank == size - 1) {
    for (int j = 0; j <= jn; j++)
        for (int k = 0; k <= kn; k++)
            F[local_i+1][j][k] = Fresh(X, j*hy, k*hz);
}

double start_time = MPI_Wtime();

```

```

double global_max;
int iter = 0;

do {
    if (rank > 0) {
        MPI_Recv(F[0], (jn+1)*(kn+1), MPI_DOUBLE, rank-1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
    }
    if (rank < size - 1) {
        MPI_Send(F[local_i], (jn+1)*(kn+1), MPI_DOUBLE, rank+1, 0,
MPI_COMM_WORLD);
    }

    for (int i = 1; i <= local_i; i++)
        for (int j = 0; j <= jn; j++)
            for (int k = 0; k <= kn; k++)
                F_old[i][j][k] = F[i][j][k];

    double local_max = 0.0;
    for (int i = 1; i <= local_i; i++) {
        int i_global = i_start + i - 1;
        double x = i_global * hx;
        for (int j = 1; j < jn; j++) {
            double y = j * hy;
            for (int k = 1; k < kn; k++) {
                double z = k * hz;
                double Fi = (F[i+1][j][k] + F[i-1][j][k]) / owx;
                double Fj = (F[i][j+1][k] + F[i][j-1][k]) / owy;
                double Fk = (F[i][j][k+1] + F[i][j][k-1]) / owz;
                F[i][j][k] = (Fi + Fj + Fk - Ro(x, y, z)) / c;

                double diff = fabs(F[i][j][k] - F_old[i][j][k]);
                if (diff > local_max) local_max = diff;
            }
        }
    }

    MPI_Allreduce(&local_max, &global_max, 1, MPI_DOUBLE, MPI_MAX,
MPI_COMM_WORLD);
    iter++;
    MPI_Barrier(MPI_COMM_WORLD);
} while (global_max > EPS);

double end_time = MPI_Wtime();
double total_time = end_time - start_time;

if (rank == 0) {
    printf(
        "Grid: %dx%dx%d, Processes: %d, Iterations: %d, Eps: %e, Time: %.6f
s\n",
        in, jn, kn, size, iter, EPS, total_time
    );
}

free(F);
free(F_old);
MPI_Finalize();
return 0;
}

```

Задание 4

```
PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_6> mpiexec -n 1 .\bin\task_1.exe 35 35 35
in = 35, jn = 35, kn = 35
Iter = 949, Eps = 1.000000e-05, Time = 0.736447 ms
Max differ = 1.085878e-03 at point (17, 17, 17)
PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_6> mpiexec -n 1 .\bin\task_2.exe 35 35 35
Grid: 35x35x35, Processes: 1, Iterations: 1101, Eps: 1.000000e-05, Time: 0.941090 s
PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_6> mpiexec -n 2 .\bin\task_2.exe 35 35 35
Grid: 35x35x35, Processes: 2, Iterations: 625, Eps: 1.000000e-05, Time: 0.286106 s
PS E:\Dev\bmstu-magistracy\term_3\parallel_computing\labs\lab_6> mpiexec -n 5 .\bin\task_2.exe 35 35 35
Grid: 35x35x35, Processes: 5, Iterations: 199, Eps: 1.000000e-05, Time: 0.065222 s
```

Рисунок 3 – Результат выполнения задания 4

Вывод: в ходе выполнения лабораторной работы были сформированы практические навыки распараллеливания алгоритмов задач, решаемых сеточными методами в MPI на примере решения задачи Пуассона в трехмерной области.