

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт

З комп'ютерного практикуму № 7 з дисципліни
«Технології паралельних обчислень»

Виконала студентка ІП-13 Замковий Дмитро Володимирович
(шифр, прізвище, ім'я, по батькові)

Перевірів ст.викл. Дифучин Антон Юрійович
(прізвище, ім'я, по батькові)

Оцінка _____

Дата оцінювання _____

Київ 2024

Лабораторна робота 7

Завдання

1. Ознайомитись з методами колективного обміну повідомленнями типу «один-до-багатьох», «багато-до-одного», «багато-до-багатьох» (див. лекцію та документацію стандарту MPI).
2. Реалізувати алгоритм паралельного множення матриць з використанням розподілених обчислень в MPI з використанням методів колективного обміну повідомленнями. 40 балів.
3. Дослідити ефективність розподіленого обчислення алгоритму множення матриць при збільшенні розміру матриць та при збільшенні кількості вузлів, на яких здійснюється запуск програми. Порівняйте ефективність алгоритму при використанні методів обміну повідомленнями «один-до-одного», «один-до-багатьох», «багато-до-одного», «багато-до-багатьох». 60 балів.

Лістинги програми

Файл task2\Main.java

```
import mpi.*;
import static java.lang.System.exit;

class Main {
    private static final int MASTER = 0;
    private static final int SIZE = 20;

    public static void main(String[] args) {
        double[][] a = new double[SIZE][SIZE];
        double[][] b = new double[SIZE][SIZE];
        double[][] c = new double[SIZE][SIZE];

        MPI.Init(args);
        int size = MPI.COMM_WORLD.Size();
        int rank = MPI.COMM_WORLD.Rank();

        int nCalc = SIZE / size;

        if (size < 2) {
            System.out.println("Need at least two MPI tasks. Quitting...");
            MPI.COMM_WORLD.Abort(1);
            exit(1);
        }

        if (SIZE % size != 0) {
            System.out.println("Number of MPI tasks must be multiple of SIZE. Quitting...");
            MPI.COMM_WORLD.Abort(1);
            exit(1);
        }

        if (rank == MASTER) {
            System.out.format("mpi_mm has started with %d tasks.\n", size);

            for (int i = 0; i < SIZE; i++) {
                for (int j = 0; j < SIZE; j++) {
                    a[i][j] = 10;
                }
            }
        }
    }
}
```

```

    }
    }
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            b[i][j] = 10;
        }
    }

    Main.printMatrix(a, "Matrix A");
    Main.printMatrix(b, "Matrix B");
}

int len = nCalc * SIZE;
int offset = rank * nCalc * SIZE;
double[] aRecv = new double[len];
double[] arrA = Main.transform(a);
MPI.COMM_WORLD.Scatter(arrA, 0, len, MPI.DOUBLE, aRecv, 0, len,
MPI.DOUBLE, MASTER);
a = Main.transform(aRecv, offset, len);

double[] bRecv = Main.transform(b);
MPI.COMM_WORLD.Bcast(bRecv, 0, SIZE * SIZE, MPI.DOUBLE, MASTER);
b = Main.transform(bRecv);

for (int i = nCalc * rank; i < nCalc * (rank + 1); i++) {
    for (int j = 0; j < SIZE; j++) {
        for (int k = 0; k < SIZE; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}

double[] cRecv = new double[SIZE * SIZE];
double[] arrC = Main.transform(c, offset, len);
MPI.COMM_WORLD.Gather(arrC, 0, len, MPI.DOUBLE, cRecv, 0, len,
MPI.DOUBLE, MASTER);
c = Main.transform(cRecv);

if (rank == MASTER)
    Main.printMatrix(c, "Matrix C");

MPI.Finalize();
}

private static void printMatrix(double[][] matrix, String name) {
    System.out.print(" \n\n\n\n");
    System.out.println(name);
    System.out.println("*****");
    for (double[] row : matrix) {
        System.out.println();
        for (double elem : row) {
            System.out.format("%10.2f ", elem);
        }
    }
    System.out.println("\n*****");
}

/**
 * Transform 2D array to 1D array
 *
 * @param matrix 2D array
 * @return 1D array
 */
private static double[] transform(double[][] matrix) {

```

```

        double[] tmp = new double[SIZE * SIZE];
        for (int i = 0; i < SIZE; i++) {
            System.arraycopy(matrix[i], 0, tmp, i * SIZE, SIZE);
        }
        return tmp;
    }

    /**
     * Transform 2D array to 1D array with offset
     *
     * @param matrix 2D array
     * @param offset offset
     * @param len length
     * @return 1D array
     */
    private static double[] transform(double[][] matrix, int offset, int len) {
        double[] tmp = new double[len];
        System.arraycopy(Main.transform(matrix), offset, tmp, 0, len);
        return tmp;
    }

    /**
     * Transform 1D array to 2D array
     *
     * @param array 1D array
     * @return 2D array
     */
    private static double[][] transform(double[] array) {
        double[][] tmp = new double[SIZE][SIZE];
        for (int i = 0; i < SIZE; i++) {
            System.arraycopy(array, i * SIZE, tmp[i], 0, SIZE);
        }
        return tmp;
    }

    /**
     * Transform 1D array to 2D array with offset
     *
     * @param array 1D array
     * @param offset offset
     * @return 2D array
     */
    private static double[][] transform(double[] array, int offset, int len) {
        double[][] tmp = new double[SIZE][SIZE];
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                int index = i * SIZE + j;
                if (index >= offset && index < offset + len) {
                    tmp[i][j] = array[i * SIZE + j - offset];
                }
            }
        }
        return tmp;
    }
}

```

Файл task3\Main.java

```

import mpi.*;
import static java.lang.System.exit;

class Main {
    private static final int MASTER = 0;

```

```

private static final int SIZE = 20;

public static void main(String[] args) {
    double[][] a = new double[SIZE][SIZE];
    double[][] b = new double[SIZE][SIZE];
    double[][] c = new double[SIZE][SIZE];

    MPI.Init(args);
    int size = MPI.COMM_WORLD.Size();
    int rank = MPI.COMM_WORLD.Rank();

    int nCalc = SIZE / size;

    if (size < 2) {
        System.out.println("Need at least two MPI tasks. Quitting...");
        MPI.COMM_WORLD.Abort(1);
        exit(1);
    }

    if (SIZE % size != 0) {
        System.out.println("Number of MPI tasks must be multiple of SIZE.
Quitting...");
        MPI.COMM_WORLD.Abort(1);
        exit(1);
    }

    if (rank == MASTER) {
        System.out.format("mpi_mm has started with %d tasks.%n", size);

        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                a[i][j] = 10;
            }
        }
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                b[i][j] = 10;
            }
        }

        Main.printMatrix(a, "Matrix A");
        Main.printMatrix(b, "Matrix B");
    }

    int len = nCalc * SIZE;
    int offset = rank * nCalc * SIZE;
    double[] aRecv = new double[len];
    double[] arrA = Main.transform(a);
    MPI.COMM_WORLD.Scatter(arrA, 0, len, MPI.DOUBLE, aRecv, 0, len,
MPI.DOUBLE, MASTER);
    a = Main.transform(aRecv, offset, len);

    double[] bRecv = Main.transform(b);
    MPI.COMM_WORLD.Bcast(bRecv, 0, SIZE * SIZE, MPI.DOUBLE, MASTER);
    b = Main.transform(bRecv);

    for (int i = nCalc * rank; i < nCalc * (rank + 1); i++) {
        for (int j = 0; j < SIZE; j++) {
            for (int k = 0; k < SIZE; k++) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
}

```

```

        double[] cRecv = new double[SIZE * SIZE];
        double[] arrC = Main.transform(c, offset, len);
        MPI.COMM_WORLD.Allgather(arrC, 0, len, MPI.DOUBLE, cRecv, 0, len,
MPI.DOUBLE);
        c = Main.transform(cRecv);

        if (rank == MASTER)
            Main.printMatrix(c, "Matrix C");

        MPI.Finalize();
    }

    private static void printMatrix(double[][] matrix, String name) {
        System.out.print(" \n\n\n\n");
        System.out.println(name);
        System.out.println("*****");
        for (double[] row : matrix) {
            System.out.println();
            for (double elem : row) {
                System.out.format("%10.2f ", elem);
            }
        }
        System.out.println("\n*****");
    }

    /**
     * Transform 2D array to 1D array
     *
     * @param matrix 2D array
     * @return 1D array
     */
    private static double[] transform(double[][] matrix) {
        double[] tmp = new double[SIZE * SIZE];
        for (int i = 0; i < SIZE; i++) {
            System.arraycopy(matrix[i], 0, tmp, i * SIZE, SIZE);
        }
        return tmp;
    }

    /**
     * Transform 2D array to 1D array with offset
     *
     * @param matrix 2D array
     * @param offset offset
     * @param len length
     * @return 1D array
     */
    private static double[] transform(double[][] matrix, int offset, int len) {
        double[] tmp = new double[len];
        System.arraycopy(Main.transform(matrix), offset, tmp, 0, len);
        return tmp;
    }

    /**
     * Transform 1D array to 2D array
     *
     * @param array 1D array
     * @return 2D array
     */
    private static double[][] transform(double[] array) {
        double[][] tmp = new double[SIZE][SIZE];
        for (int i = 0; i < SIZE; i++) {
            System.arraycopy(array, i * SIZE, tmp[i], 0, SIZE);
        }
    }

```


[illegible]

Різний розмір матриці (при 10 вузлах)

15000
10000
5000
0
-5000

100 500 1000 1500 2000

один-до-одного, ms
багато-до-багатьох, ms
Лінійна (один-до-одного, ms)
Лінійна (багато-до-одного, ms)

багато-до-одного, ms
Лінійна (багато-до-багатьох, ms)

Різна кількість вузлів (при матриці розмірністю 2000)

40000
35000
30000
25000
20000
15000
10000
5000
0

2 5 10 16 20

один-до-одного, ms
багато-до-багатьох, ms
Лінійна (один-до-одного, ms)
Лінійна (багато-до-одного, ms)

багато-до-одного, ms
Лінійна (багато-до-багатьох, ms)

Висновки

В ході даної лабораторної роботи було розроблено та досліджено методи колективного обміну повідомленнями типу «один-до-багатьох», «багато-до-одного», «багато-до-багатьох». Також був проведений аналіз і порівняння методів з методом «один-до-одного». Було виявлено, що метод «один-до-одного» незначно, але програє іншим методам колективного спілкування.