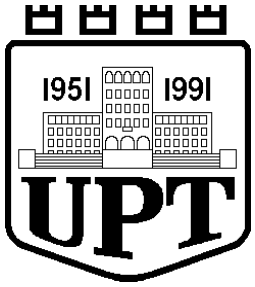

UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË DHE INFORMACIONIT
DEPARTAMENTI I INXHINIERISË INFORMATIKE

Punë Laboratori nr. 2

Tema: Përdorimi I funksioneve të librarisë mpi.h

Lënda: Sisteme Të Shpërndara

Grupi: III-B

Punoi:
Piro Gjikdhima

Pranoi:
MSc.Megi Tartari

Ushtrimi 1

Implementoni ne gjuhen C nje program qe realizon shumezimin e 2 matricave A[n][n] dhe B[n][n], dhe rezultatin e ruan ne matricen C[n][n]. Perdorni funksionet e MPI per te realizuar ekzekutimin me procese paralele.

KODI

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

void populateMatrix(int rows, int cols, int matrix[rows][cols]);
void multiplyMatrix(int rows_A, int cols_A, int rows_B, int cols_B, int
A[rows_A][cols_A], int B[rows_B][cols_B], int C[cols_A][cols_B]);

int main(int argc, char *argv[]) {
    srand(102021);

    int rank, size, rows_A, cols_A, rows_B, cols_B;

    rows_A = atoi(argv[1]);
    cols_A = atoi(argv[2]);
    rows_B = atoi(argv[3]);
    cols_B = atoi(argv[4]);

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

    if (argc != 5) {
        if (rank == 0) {
            printf("\033[1;34mPerdorimi: %s <rows_A> <cols_A> <rows_B>
<cols_B>\033[0m\n", argv[0]);
        }
        MPI_Abort(MPI_COMM_WORLD, 1);
    }


    if (cols_A != rows_B) {
        printf("\033[1;31mNuk mund të bëhet shumëzimi i matricave! Kolonat e A (%d)
duhet të përputhen me rreshtat e B (%d).\033[0m\n", cols_A, rows_B);
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    MPI_Bcast(&rows_A, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&cols_A, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&rows_B, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&cols_B, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

```
    if (rows_A % size != 0) {
        if (rank == 0) {
            printf("\033[1;31mNumri i rreshtave në Matricën A duhet të jetë i
plotpjestueshem nga numri i proceseve.\033[0m\n");
        }
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    int rows_per_process = rows_A / size;
    int A[rows_A][cols_A], B[rows_B][cols_B], C[rows_A][cols_B];
    int local_A[rows_per_process][cols_A];
    int local_C[rows_per_process][cols_B];


    if (rank == 0) {
        populateMatrix(rows_A, cols_A, A);
        populateMatrix(rows_B, cols_B, B);
    }

    MPI_Bcast(B, rows_B * cols_B, MPI_INT, 0, MPI_COMM_WORLD);

    MPI_Scatter(A, rows_per_process * cols_A, MPI_INT, local_A, rows_per_process *
cols_A, MPI_INT, 0, MPI_COMM_WORLD);

    multiplyMatrix(rows_per_process, cols_A, rows_B, cols_B, local_A, B, local_C);

    MPI_Gather(local_C, rows_per_process * cols_B, MPI_INT, C, rows_per_process *
cols_B, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Matricë rezultati C:\n");
        for (int i = 0; i < rows_A; i++) {
            for (int j = 0; j < cols_B; j++) {
                printf("%d ", C[i][j]);
            }
            printf("\n");
        }
    }

    MPI_Finalize();
    return 0;
}

void populateMatrix(int rows, int cols, int matrix[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = rand() % 10;
        }
    }
}
```

```
void multiplyMatrix(int rows_A, int cols_A, int rows_B, int cols_B, int
A[rows_A][cols_A], int B[rows_B][cols_B], int C[cols_A][cols_B]){
    for (int i = 0; i < rows_A; i++) {
        for (int j = 0; j < cols_B; j++) {
            C[i][j] = 0;
            for (int k = 0; k < cols_A; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
```

Rezultate

Kodi i mësipërm përdor funksionet e librarisë mpi.h per te bere paralelizimin e kërkesës se dhene. Llogjika është e tille:

1. Fillimisht marrim prej konsoles numrin e rreshtave dhe shtyllave per secilën matrice.
2. Para se të vazhdojmë me paralelizimin, programi kontrollon që përputhshmëria e dimensioneve të matricave të jetë e duhur për shumëzim.Pra a janë kolonat e A = me rreshtat e B.
3. Pastaj kontrollohet nëse numri i rreshtave te matricës A është i plotpjesëtueshëm me numrin e proceseve.Nese jo , do te thotë nuk mundemi ti ndajmë ne menyre te barabarte midis proceseve rreshtat e A dhe kemi gabim. Dalim prej cdo procesi.
4. Nëse nuk ka problem, bëjmë broadcast cdo rresht e shtylle tek cdo proces, gjithashtu dërgojmë dhe matricën B.Nderkohe kemi krijuar matrica lokale A dhe C qe do te perdoren si buffer prej cdo procesi.
5. Programi përdor MPI_Scatter për të shpërndarë rreshtat e Matricës A midis proceseve.
6. Çdo proces kryen shumëzimin lokal të pjesës së tij të matricës A me matricën B.
7. Pasi secili proces ka përfunduar shumëzimin, MPI_Gather përdoret për të mbledhur rezultatet nga të gjitha proceset dhe për t'i bashkuar ato në Matricën C të procesit kryesor.
8. Në fund, procesi kryesor (rank 0) printon matricën C, që është rezultati i shumëzimit të matricave A dhe B.

Ky program përdor MPI për të realizuar paralelizimin e shumëzimit të matricave. Ai ndan punën midis proceseve të ndryshëm për të realizuar shumëzimin dhe për të optimizuar kohën e ekzekutimit. Pas përfundimit të operacionit, rezultati i shumëzimit të matricave është mbledhur dhe shfaqur nga procesi kryesor.

```
┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 4 ./usht1 4 4 4 4
Matricë rezultati C:
81 46 21 44
100 66 110 78
36 30 28 20
88 52 63 61

┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 2 ./usht1 4 4 4 4
Matricë rezultati C:
81 46 21 44
100 66 110 78
36 30 28 20
88 52 63 61

┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 3 ./usht1 3 5 5 4
Matricë rezultati C:
35 87 48 28
81 160 87 84
56 84 44 55

┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 3 ./usht1 6 5 5 4
Matricë rezultati C:
41 63 33 43
98 127 72 101
66 43 42 59
111 131 101 125
63 78 85 52
114 122 54 80
```

Ushtrimi 2

Implementoni ne gjuhen C nje program qe gjen shumatoren e te gjithe elementeve te nje matrice A. Perdorni funksionet e MPI per te realizuar ekzekutimin me procese paralele.

Kodi

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

void populateMatrix(int rows, int cols, int matrix[rows][cols]);

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

    srand(102021);

    int rank, size, rows, cols;

    rows = atoi(argv[1]);
    cols = atoi(argv[2]);

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

    if (argc != 3) {
        if (rank == 0) {
            printf("\033[1;34mPerdorimi: %s <rows> <cols>\033[0m\n", argv[0]);
        }
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    MPI_Bcast(&rows, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&cols, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rows % size != 0 && cols % size != 0) {
        if (rank == 0) {
            printf("\033[1;31mNumri i rreshtave: (%d) ose i shtyllave: (%d) duhet
plotpjestueshem nga numri i proceseve: (%d).\033[0m\n",rows, cols, size);
        }
        MPI_Abort(MPI_COMM_WORLD, 1);
    }
    else if (cols % size == 0)
    {
        int temp = rows;
        rows = cols;
        cols = temp;
    }
```

```
    int rows_per_process = rows / size;
    int A[rows][cols];
    int local_A[rows_per_process][cols];


    if (rank == 0) {
        populateMatrix(rows, cols, A);
    }

    MPI_Scatter(A, rows_per_process * cols, MPI_INT, local_A, rows_per_process *
cols, MPI_INT, 0, MPI_COMM_WORLD);

    long local_sum = 0;
    for (int i = 0; i < rows_per_process; i++) {
        for (int j = 0; j < cols; j++) {
            local_sum += local_A[i][j];
        }
    }
    printf("Procesi %d llogariti shumen: %ld\n", rank, local_sum);


    long global_sum = 0;
    MPI_Reduce(&local_sum, &global_sum, 1, MPI_LONG, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Shuma e matrices: %ld\n", global_sum);
    }

    MPI_Finalize();
    return 0;
}

void populateMatrix(int rows, int cols, int matrix[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = (rand() % 100) + 1;
        }
    }
}
```

Konkluzione

Kodi i mësipërm përdor funksionet e librarisë mpi.h per te bere paralelizimin e kërkesës se dhene. Llogjika është e tille:

1. **Inicializimi i MPI dhe marrja e të dhënave**:
    - Nga komanda argv, merren dimensionet e matrices (rows dhe cols).
    - Inicializohen proceset dhe identifikohen rank dhe size (numri i proceseve).
2. **Kontrolli i argumenteve**:
    - Nëse nuk jepen saktësisht 2 argumente (përveç emrit të programit), procesi kryesor (rank 0) printon një mesazh gabimi dhe abortohet ekzekutimi.
3. **Broadcast i dimensioneve**:
    - Procesi kryesor dërgon dimensionet rows dhe cols te të gjitha proceset përmes MPI_Bcast.
4. **Kontrolli i plotpjesëtueshmërisë**:
    - Kontrollohet nëse rows ose cols janë të plotpjesëtueshëm me numrin e proceseve (size).
    - **Nëse cols % size == 0**, kodi ndërron vendet e rows dhe cols.
5. **Përgatitja e matricës dhe shpërndarja e rreshtave**:
    - Procesi kryesor gjeneron matricën A me vlera të rastësishme duke përdorur populateMatrix().
    - Përdoret MPI_Scatter për të shpërndarë nga rows_per_process rreshta (ose kolona, në varësi të kushtit të mësipërm) në çdo proces.
6. **Llogaritja lokale e shumës**:
    - Çdo proces llogarit shumën e elementeve të pjesës së tij të matricës (local_A).
    - Rezultati lokal ruhet në variablin local_sum.
7. **Mbledhja e shumave globale**:
    - Përdoret MPI_Reduce për të mbledhur të gjitha local_sum në një global_sum te procesi kryesor.
8. **Shfaqja e rezultatit**:
    - Procesi kryesor (rank 0) printon shumën totale të elementeve të matricës.

```
┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 4 ./usht2 4 10
Procesi 0 llogariti shumen: 424
Shuma e matrices: 1878
Procesi 1 llogariti shumen: 488
Procesi 2 llogariti shumen: 555
Procesi 3 llogariti shumen: 411

┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 4 ./usht2 10 4
Procesi 0 llogariti shumen: 424
Shuma e matrices: 1878
Procesi 1 llogariti shumen: 488
Procesi 2 llogariti shumen: 555
Procesi 3 llogariti shumen: 411

┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 5 ./usht2 10 4
Procesi 0 llogariti shumen: 298
Procesi 1 llogariti shumen: 449
Procesi 2 llogariti shumen: 397
Procesi 3 llogariti shumen: 391
Procesi 4 llogariti shumen: 343
Shuma e matrices: 1878

┌──(kali㉿kali)-[~/Desktop/SistemeShpernadara/Lab/Lab2]
└─$ mpirun -np 5 ./usht2 100 4
Procesi 0 llogariti shumen: 3684
Procesi 2 llogariti shumen: 4118
Procesi 4 llogariti shumen: 4127
Procesi 1 llogariti shumen: 4057
Procesi 3 llogariti shumen: 3580
Shuma e matrices: 19566
```