Τρίτη εργαστηριακή εργασία στο μάθημα Εισαγωγή στον Παράλληλο Υπολογισμό

Ιάκωβος Μαστρογιαννόπουλος - cse242017102

2021-01-02

Contents

Abstract

Αυτή είναι η τρίτη εργαστηριακή εργασία στο μάθημα Εισαγωγή στον Παραλλήλο Υπολογισμού του 5ου εξαμήνου.

1 Κατανόηση

Το πρόβλημα που μας ζητείται να επιλύσουμε το παρακάτω ζητούμενο: έχουμε μια καρτησιανή τοπολογία MxN, και πρέπει με την χρήση της C και του Framework MPI να βρούμε το καρτησιανό γινόμενο. Μας δώθηκε η επιλόγη να επιλέξουμε μεταξύ `MPI_Send()` και `MPI_Recv()` ή με `MPI_Cart_sub()` και `MPI_Reduce()`. Επιλέχτηκε να υλοποιηθεί με το πρώτο τρόπο. Τα δεδομένα μπορεί να τα δίνει ο χρήστης τα δεδομένα από το πληκτρολόγιο ή από αρχείο.

2 Υλοποίηση

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

int main(int argc, char *argv[]){
    int rank, size, tag = 100;
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int **array;
    int *localArray;
    int m_sum, n_sum;

    FILE *file = fopen("input.txt", "r");
    if (file == NULL) {
        printf("The file couldn't be found! Exiting...\n");
        exit(-1);
    }

    if (rank == 0) {
        fscanf(file, "%d", &m_sum);
        fscanf(file, "%d", &n_sum);

        if (n_sum % size != 0) {
            printf("N must be integral multiple of P, thus exiting the program...\n");
            exit(1);
        }
    }

    MPI_Bcast(&m_sum, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&n_sum, 1, MPI_INT, 0, MPI_COMM_WORLD);

    array = (int **)malloc(sizeof(int *) * m_sum);
    for (int i = 0; i < m_sum; i++)
        array[i] = (int *)malloc(sizeof(int) * n_sum);

    if (rank == 0) {
        for (int i = 0; i < m_sum; i++)
            for (int j = 0; j < n_sum; j++) {
                fscanf(file, "%d", &array[i][j]);
```

```
                printf("array[%d][%d] = %d\n", i, j, array[i][j]);
            }
    }

    free(file);

    localArray = (int *)malloc(sizeof(int) * n_sum);
    for (int i = 0; i < m_sum; i++)
        MPI_Scatter(array[i], 1, MPI_INT, &localArray[i], 1,
    MPI_INT, 0, MPI_COMM_WORLD);

    int coreSum = localArray[0];
    for (int i = 1; i < m_sum; i++)
        coreSum += localArray[i];

    int totalSum = 0;
    if (rank < size - 1)
        MPI_Recv(&totalSum, 1, MPI_INT, rank + 1, tag,
    MPI_COMM_WORLD, &status);

    if (rank > 0) {
        totalSum += coreSum;
        MPI_Send(&totalSum, 1, MPI_INT, rank - 1, tag,
    MPI_COMM_WORLD);
    }

    else {
        printf("Total Sum = %d\n", totalSum + coreSum);
        fflush(stdout);
    }

    free(array);
    free(localArray);
    MPI_Finalize();

    return 0;
}
```