جامعة الشام الخاصة

كلية الهندسة المعلوماتية

اسم المقرر: Parallel programming

Homework6 MPI

*Code:*

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "mpi.h"

#include <pthread.h>

#define N 100

double\* random\_numbers;

double\* square\_roots;

void\* compute\_square\_root(void\* arg) {

    int index = \*((int\*) arg);

    square\_roots[index] = sqrt(random\_numbers[index]);

    pthread\_exit(NULL);

}

int main(int argc, char\*\* argv) {

    int rank, size;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if (rank == 0) {

        random\_numbers = (double\*) malloc(N \* sizeof(double));

        for (int i = 0; i < N; i++) {

            random\_numbers[i] = rand() / (double) RAND\_MAX;

        }

    }

    double\* local\_random\_numbers = (double\*) malloc(N / size \* sizeof(double));

    MPI\_Scatter(random\_numbers, N / size, MPI\_DOUBLE, local\_random\_numbers, N / size, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

    square\_roots = (double\*) malloc(N / size \* sizeof(double));

    pthread\_t\* threads = (pthread\_t\*) malloc(N / size \* sizeof(pthread\_t));

    int\* indices = (int\*) malloc(N / size \* sizeof(int));

    for (int i = 0; i < N / size; i++) {

        indices[i] = i;

        pthread\_create(&threads[i], NULL, compute\_square\_root, (void\*) &indices[i]);

    }

    for (int i = 0; i < N / size; i++) {

        pthread\_join(threads[i], NULL);

    }

    double\* all\_square\_roots = NULL;

    if (rank == 0) {

        all\_square\_roots = (double\*) malloc(N \* sizeof(double));

    }

    MPI\_Gather(square\_roots, N / size, MPI\_DOUBLE, all\_square\_roots, N / size, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

    if (rank == 0) {

        double sum = 0.0;

        for (int i = 0; i < N; i++) {

            sum += all\_square\_roots[i];

        }

        double average = sum / N;

        printf("Average: %f\n", average);

        free(all\_square\_roots);

    }

    free(local\_random\_numbers);

    free(square\_roots);

    free(threads);

    free(indices);

    if (rank == 0) {

        free(random\_numbers);

    }

    MPI\_Finalize();

    return 0;

}

شرح الكود:

1. إنشاء أرقام عشوائية على العقدة الأولى:

- على العقدة الأولى نقوم بإنشاء مجموعة من الأرقام العشوائية باستخدام مولد أرقام عشوائية لنفترض أن لدينا N أرقام عشوائية.

2. إرسال الأرقام إلى جميع العقد (scatter):

- استخدم وظيفة MPI MPI\_Scatter لتوزيع الأرقام العشوائية المُنشأة من العقدة الأولى إلى جميع العقد الأخرى في المتواصلة MPI. ستتلقى كل عقدة جزءًا من الأرقام العشوائية.

3. حساب الجذر التربيعي على كل pthread لكل عقدة:

- على كل عقدة، نقوم بإنشاء pthread (خيط) لكل رقم عشوائي مستلم. سيقوم كل خيط بحساب الجذر التربيعي للرقم المُسند له.

4. حساب المتوسط للنتائج باستخدام MPI (reduce):

- بعد انتهاء جميع الخيوط من حساب الجذور التربيعية، نقوم بجمع النتائج من كل عقدة باستخدام وظيفة MPI MPI\_Gather أو MPI\_Allgather. ستجمع هذه الوظيفة النتائج من كل عقدة في مصفوفة على العقدة الأولى.

- على العقدة الأولى، نقوم بحساب المتوسط للنتائج عن طريق جمع جميع الجذور التربيعية المحسوبة وقسمتها على إجمالي عدد الأرقام العشوائية (N).