**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**



**Лабораторна робота №4**

з дисципліни

«Технології розподілених систем та паралельних обчислень»

**Виконав:**

студент групи КН-308

Матвіїв Микола

**Викладач:**

Мочурад Л.І.

Львів – 2020 р.

**Варіант 13**

**Мета:**

Ознайомитися із засобами для організації паралельного виконання програми, що надаються технологією MPI.

**Завдання:**

Розробити програму, яка повинна реалізувати обчислення значення функції f(x) на відрізку [1, N+1] з кроком h = N/k , де N - номер варіанта;

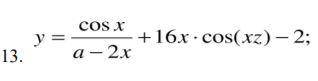
A = N

B = N\*2

K = N/2

Z = N^2

N = 13;



**Код програми:**

#include <stdio.h>

#include "mpi.h"

#include <iostream>

#include <ctime>

#include <iomanip>

#include <fstream>

#include <math.h>

#define N 13

#define k 6.5

using namespace std;

double step = double(N) / double(k);

double f(double x) {

int a = N;

int z = N \* N;

return (cos(x) / (a - 2 \* x)) + 16 \* x \* cos(x \* z) - 2;

}

void seq\_method()

{

double ts1, ts2;

ts1 = MPI\_Wtime();

double y;

for (double x = 1; x <= N + 1; x += step) {

y = f(x);

cout << "x = " << x << " y = " << y << endl;

//printf("x = %.3f y = %.3f\n", x, y);

}

ts2 = MPI\_Wtime();

double elapsed\_secs = double(ts2 - ts1) / CLOCKS\_PER\_SEC;

cout << setprecision(10) << "Elapsed time (Serial) is " << elapsed\_secs << endl << endl;

}

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

double t1, t2;

double x = 1;

double\* tx = &x;

MPI\_Status Status;

int ProcNum, ProcRank;

MPI\_Init(&argc, &argv);

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

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

MPI\_Barrier(MPI\_COMM\_WORLD);

if (!ProcRank) {

printf("Performing serial computation on cpu %d\n", ProcRank);

seq\_method();

t1 = MPI\_Wtime();

}

while (\*tx < N + 1) {

if (ProcRank == 0)

{

{

MPI\_Send(tx, 1, MPI\_DOUBLE, 1, 1, MPI\_COMM\_WORLD);

\*tx += step;

MPI\_Send(tx, 1, MPI\_DOUBLE, 2, 1, MPI\_COMM\_WORLD);

\*tx += step;

MPI\_Send(tx, 1, MPI\_DOUBLE, 3, 1, MPI\_COMM\_WORLD);

\*tx += step;

}

}

else {

MPI\_Recv(tx, 1, MPI\_DOUBLE, MPI\_ANY\_SOURCE, 1, MPI\_COMM\_WORLD, &Status);

cout << "Process #" << ProcRank << " x = " << \*tx << " y = " << f(\*tx) << endl;

//printf("#%d thread: x = %.3f y = %.3f\n", ProcRank, \*tx, f(\*tx));

}

}

if (!ProcRank) {

t2 = MPI\_Wtime();

double elapsed\_secs1 = double(t2 - t1) / CLOCKS\_PER\_SEC;

//printf("Elapsed time is %.10f\n\n", elapsed\_secs);

cout << setprecision(10) << "Elapsed time (MPI) is " << elapsed\_secs1;

}

MPI\_Finalize();}

**Результати виконання:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Крок | Час розрахунку на 1 ядрі у секундах | Час розрахунку на 2 ядрах у секундах | Час розрахунку на 4 ядрах у секундах | Час розрахунку на 8 ядрах у секундах |
| 13 | 0,0185316 | 0,0113824 | 0,0064595 | 0,0048913 |
| 1300 | 0,0350632‬ | 0,0227648 |  | 0,0097826 |
| 130000 | 0,0585948 | 0,0455296 |  | 0,0189652 |
| 13000000 |  | 0,0682944 | 0,0327572 | 0,0049913 |

Таблиця 1. Час виконання програми на різній кількості ядер

**Рис.1.** Залежність часу від кількості ядр до довжини відрізка(синій колір – 1 ядро, помаранчевий – 2 ядра, сірий – 4 ядра, жовтий – 8 ядр)

**Висновок**

На лабораторінй роботі я ознайомився із організацією паралельного виконання програми за допомогою технології MPI.

Порівняв час виконання в залежесності від кількості ядер