**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**



**Лабораторна робота**

з дисципліни

«Технології розподілених систем та паралельних обчислень»

**Виконав:**

студент групи КН-308

Гецянин Дмитро

**Викладач:**

Мочурад Л.І.

Львів – 2020р.

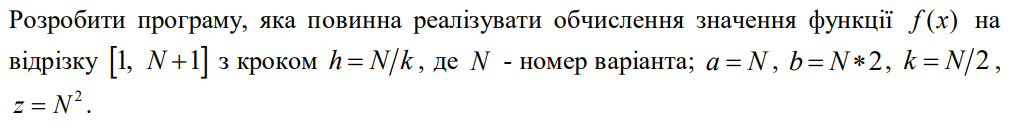
**Лабораторна робота №4**

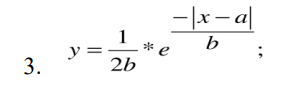
**Тема: Знайомство з технологією паралельного програмування MPI.**

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

**Варіант №3**

**Завдання.**





**Програмний код**

# include <cstdlib>

# include <iostream>

# include <iomanip>

# include <cmath>

# include <ctime>

# include <mpi.h>

**using** **namespace** std;

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

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

{

**int** id;

**int** ierr;

**int** n;

**int** q\_part;

**int** p;

cout << "Input n" << endl;

cin >> n;

**int** a = n;

**int** b = n \* **2**;

**double** q;

**double** wtime\_diff;

**double** wtime\_end;

**double** wtime\_start;

**double** x;

//

// Initialize MPI.

//

ierr = MPI\_Init ( &argc, &argv );

**if** ( ierr != **0** )

{

cout << "**\n**";

cout << "BUFFON - Fatal error!**\n**";

cout << " MPI\_Init returned ierr = " << ierr << "**\n**";

exit ( **1** );

}

//

// Get the number of processes.

//

ierr = MPI\_Comm\_size ( MPI\_COMM\_WORLD, &p );

//

// Determine this processes's rank.

//

ierr = MPI\_Comm\_rank ( MPI\_COMM\_WORLD, &id );

//

// Record the starting time.

//

**if** ( id == **0** )

{

wtime\_start = MPI\_Wtime ( );

}

//

// Broadcast the number of intervals to the other processes.

//

ierr = MPI\_Bcast ( &n, **1**, MPI\_INT, master, MPI\_COMM\_WORLD );

**double** k = n / **2**;

**double** h = n / k;

**for** ( x = **1**; x <= n + **1**; x += h)

{

**double** param = -(fabs(x - a)) / b;

**double** y = exp(param) / **2** \* b;

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

}

ierr = MPI\_Reduce ( &q\_part, &q, **1**, MPI\_DOUBLE, MPI\_SUM, master,

MPI\_COMM\_WORLD );

**if** ( id == **0** )

{

wtime\_end = MPI\_Wtime ( );

wtime\_diff = wtime\_end - wtime\_start;

cout << "**\n**";

cout << " Wall clock elapsed seconds = " << wtime\_diff << "**\n**";

}

//

// Terminate MPI.

//

MPI\_Finalize ( );

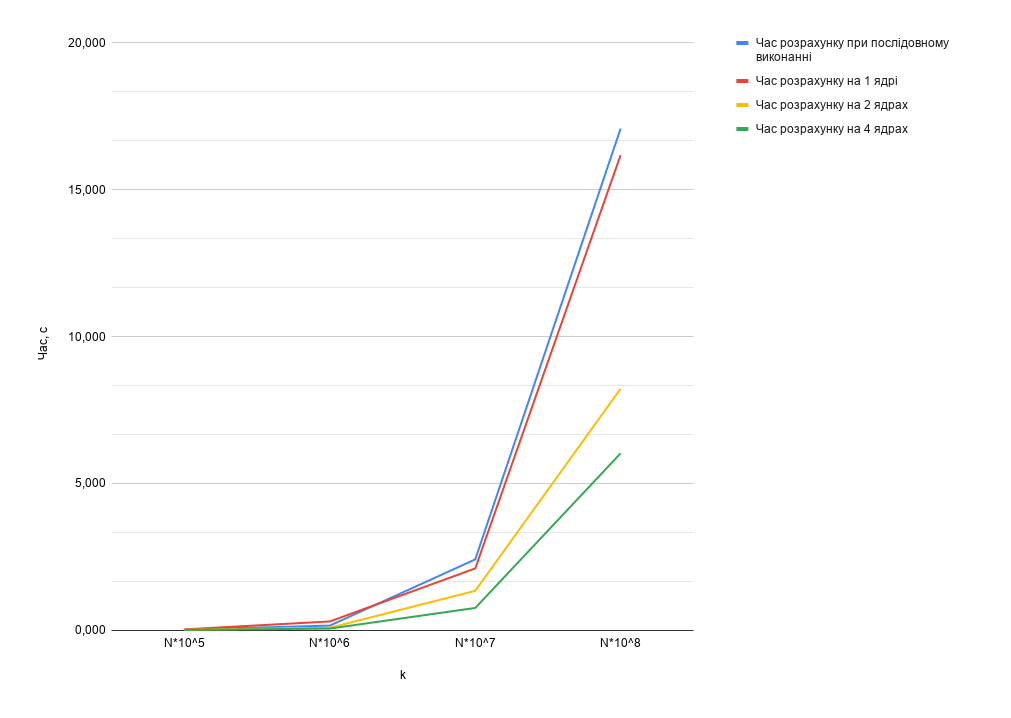
**return** **0**;

}

**Результати виконання програмного коду**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **k** | **Час розрахунку при послідовному виконанні** | **Час розрахунку на 1 ядрі** | **Час розрахунку на 2 ядрах** | **Час розрахунку на 4 ядрах** |
| N\*10^5 | 0,025 | 0,027 | 0,012 | 0,008 |
| N\*10^6 | 0,155 | 0,297 | 0,081 | 0,054 |
| N\*10^7 | 2,410 | 2,103 | 1,340 | 0,756 |
| N\*10^8 | 17,083 | 16,180 | 8,220 | 6,022 |

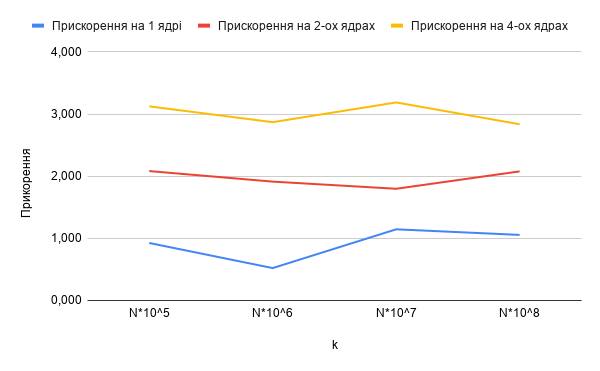
Таблиця 1. Результати виконання при виконанні на різних кількостях ядер та при різних параметрах k.



Графік 1. Залежність часу обчислення від кількості ядер та відповідного параметру k

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Прискорення на 1 ядрі** | **Прискорення на 2-ох ядрах** | **Прискорення на 4-ох ядрах** |
| N\*10^5 | 0,926 | 2,083 | 3,125 |
| N\*10^6 | 0,522 | 1,914 | 2,870 |
| N\*10^7 | 1,146 | 1,799 | 3,188 |
| N\*10^8 | 1,056 | 2,078 | 2,837 |

Таблиця 2. Результати прискорення



Графік 2. Результати прискорення

**Висновок.** Виконуючи лабораторну роботу, я ознайомився із засобами для організації паралельного виконання програми, що надаються технологією MPI.