КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт до лабораторних робіт № 1, 2

“Розподілені обчислення через MPI та OpenMP”

студент групи ТТП-31  
Скакун Олександр

КИЇВ-2019

**Постановка задачi**

Вибрати алгоритм для паралельних обчислень. Потім реалізувати його послідовно та паралельно за допомогою MPI та OpenMP. Вивести результати. Проаналізувати результати та скласти висновок.

В ході лабораторної роботи була обчислена експонента за допомогою ряду Тейлора. Для відчутного покращення у швидкості, я взяв n = 100000. При низьких значеннях n паралельні обчислення можуть давати гірший результат ніж послідовне.



**MPI**

|  |  |
| --- | --- |
| Число процесів | Час рішення |
| 4 | 6.37 |
| 3 | 7.14 |
| 2 | 9.43 |
| 1 | 17 |

**OpenMP**

|  |  |
| --- | --- |
| Число процесів | Час рішення |
| 4 | 3.37 |
| 3 | 4.32 |
| 2 | 6.66 |
| 1 | 8.93 |

**Висновок**

Отже, швидший результат був отриманий за допомогою технології OpenMP. Також, при збільшенні кількості процесів, мы отримуємо закономірне покращення в швидкості, яке можна відслідкувати у таблицях вище.

**Коди програм:**

**MPI**

#include "pch.h"

#include "mpi.h"

#include <iostream>

#include <windows.h>

using namespace std;

double Fact(int n)

{

if (n == 0)

return 1;

else

return n \* Fact(n - 1);

}

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

{

SetConsoleOutputCP(1251);

int n;

int myid;

int numprocs;

int i;

int rc;

long double drob, drobSum = 0, Result, sum;

double startwtime = 0.0;

double endwtime;

n = 100000;

if (rc = MPI\_Init(&argc, &argv))

{

cout << "Ошибка запуска, выполнение остановлено " << endl;

MPI\_Abort(MPI\_COMM\_WORLD, rc);

}

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

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

if (myid == 0)

{

startwtime = MPI\_Wtime();

}

MPI\_Bcast(&n, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

for (i = myid; i <= n; i += numprocs)

{

drob = 1 / Fact(i);

drobSum += drob;

}

MPI\_Reduce(&drobSum, &Result, 1, MPI\_LONG\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

cout.precision(20);

if (myid == 0)

{

cout << Result << endl;

endwtime = MPI\_Wtime();

cout << (endwtime - startwtime) \*100 << endl;

}

MPI\_Finalize();

return 0;

}

**OpenMP**

#include "pch.h"

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <omp.h>

int fact(int c);

double taylor(double x, int n);

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

double x = 0;

int n = 0;

int n\_max = 100000;

omp\_set\_num\_threads(4);

if (argc != 3) {

printf("Input three parametrs!\n");

return 1;

}

sscanf\_s(argv[1], "%lf", &x);

//sscanf\_s(argv[2], "%d", &n);

if ((n > n\_max) || (n < 0)) {

printf("Invalid parameter n, enter n from 0..32\n");

return 1;

}

int i;

double sum = 0;

double t = omp\_get\_wtime();

#pragma omp parallel for private(i) reduction(+: sum)

for (i = 1; i <= n; i++) {

double taylor = pow(x, i) / fact(i);

sum += taylor;

}

double diff = omp\_get\_wtime() - t;

printf("Time: %lf\n", diff);

return 0;

}

int fact(int c) {

int fact = 1;

int i;

for (i = 1; i <= c; i++) {

fact \*= i;

}

return fact;

}