**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**



**Лабораторна робота**

з дисципліни

«Технології розподілених систем та паралельних обчислень»

**Виконав:**

студент групи КН-308

Матвіїв Микола

**Викладач:**

Мочурад Л.І.

Львів – 2020 р.

**Лабораторна робота №1**

**Тема: Основні конструкції OpenMP. Модель даних**

**Мета:** Ознайомитися з технологією OpenMP та набути практичних навиків

її використання

**Варіант №13**

**Завдання.** Створити програму яка повинна реалізувати матрично-векторне множення, використовуючи вхідні дані відповідно до завдання (n – розмірність квадратної матриці). Варіанти 1-10 виконують розбиття матриці по горизонтальних смужках, 11-20 – по вертикальних.

****

Обробити паралельним та послідовним способами множення матриці на вектор відповідно до свого варіанту. Вивести результат виконання алгоритму паралельним та послідовним способами. Визначити час, який був затрачений на виконання програми для обох способів множення матриці на вектор. Розроблену програму виконати почергово на 1, 2, 4 та 8 ядерному процесорі. Побудувати графік залежності часу обчислення від кількості ядер. Навести оцінки паралельного прискорення (parallel speedup) та паралельної ефективності (parallel efficiency).

**Програмний код**

#include <iostream>

#include <omp.h>

#include <math.h>

#define n 5000

using namespace std;

double par, seq;

float Matrix[n][n], Vector[n], C0[n][n], C1[n][n], C2[n][n], C4[n][n];

void forward\_method(float Matrix[n][n], float Vector[n], float C[n][n])

{

double ts1, ts2;

float element = 0;

float element2 = 0;

float Answer[n];

ts1 = omp\_get\_wtime();

for (int i = 0; i < n; i++) {

element = 0;

for (int j = 0; j < n; j++) {

element += Matrix[j][i] \* Vector[i];

++ C[j][i] = element;

}

}

for (int q = 0; q < n; q++) {

element2 = 0;

for (int w = 0; w < n; w++) {

element2 += C[q][w];

}

Answer[q] = element2;

}

ts2 = omp\_get\_wtime();

seq = ts2 - ts1;

printf("Forward Method time = %.15f\n\n", seq);

}

void parallel(float Matrix[n][n], float Vector[n], float C[n][n], int thr\_num)

{

double t1, t2;

float element = 0;

float element2 = 0;

float Answer[n];

int i, j;

omp\_set\_dynamic(0);

omp\_set\_num\_threads(thr\_num);

#pragma omp parallel private(element, j)

{

#pragma omp for

for (i = 0; i < n; i++) {

element = 0;

for (j = 0; j < n; j++) {

element += Matrix[j][i] \* Vector[i];

C[j][i] = element;

}

}

for (int q = 0; q < n; q++) {

element2 = 0;

for (int w = 0; w < n; w++) {

element2 += C[q][w];

}

Answer[q] = element2;

}

}

t1 = omp\_get\_wtime();

omp\_set\_dynamic(0);

omp\_set\_num\_threads(thr\_num);

#pragma omp parallel private(element, j)

{

#pragma omp for

for (i = 0; i < n; i++) {

element = 0;

for (j = 0; j < n; j++) {

element += Matrix[j][i] \* Vector[i];

C[j][i] = element;

}

}

for (int q = 0; q < n; q++) {

element2 = 0;

for (int w = 0; w < n; w++) {

element2 += C[q][w];

}

Answer[q] = element2;

}

}

t2 = omp\_get\_wtime();

par = t2 - t1;

printf("Parallel %d time = %.15f\n", thr\_num, par);

printf("Speedup = %.3f\n", seq / par);

printf("Efficiency = %.3f\n\n", seq / (par \* thr\_num));

}

int main()

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

Matrix[i][j] = (float)((j+1)\*sin(i+1));

}

Vector[i] = (float)((i+1)\*sin(i+1));

}

cout << "n = " << n << endl << endl;

forward\_method(Matrix, Vector, C0);

parallel(Matrix, Vector, C1, 1);

parallel(Matrix, Vector, C2, 2);

parallel(Matrix, Vector, C4, 4);

cin.ignore();

return 0;

}

**Результати роботи програми**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Розмір**  **матриці** | **Час виконання послідовного алгоритму (сек.)** |  | | **Паралельний алгоритм** | | | | | | | |
| **1 потік** | | | **2 потоки** | | **4 потоки** | | **8 потоків** | |
| Час,  (сек.) | Прискорення | | Час,  (сек.) | Прискорення | Час ,  (сек.) | Прискорення | Час,  (сек.) | Прискорення |
| 1 000 | 0.915325610 | 0,916515610 | 1.231 | | 0,61101 | 1.555 | 0,381882 | 2,413 | 0,241188 | 3,834 |
| 2 000 | 1.314466555 | 1,213516555 | 1.273 | | 0,809011 | 1.529 | 0,505632 | 2,517 | 0,319346 | 3,845 |
| 3 000 | 2.9841323651 | 2,16515123651 | 1.351 | | 1,443434 | 1.463 | 0,902146 | 2,384 | 0,569777 | 3,659 |
| 4 000 | 2.9621616515 | 2,9956165152 | 1.134 | | 1,997078 | 1.474 | 1,248174 | 2,469 | 0,78832 | 3,891 |
| 5 000 | 3.6841165103 | 3,4561165103 | 1.160 | | 2,304078 | 1.536 | 1,440049 | 2,446 | 0,909504 | 3,801 |
| 6 000 | 4.9984915645 | 4,45616155645 | 1.198 | | 2,970774 | 1.580 | 1,856734 | 2,356 | 1,172674 | 3,795 |
| 7 000 | 5.786513121 | 5,64816513121 | 1.211 | | 3,765443 | 1.508 | 2,353402 | 2,124 | 1,486359 | 3,848 |
| 8 000 | 6.6142005165 | 6,122005165 | 1.241 | | 4,081337 | 1.505 | 2,550835 | 2,364 | 1,611054 | 3,795 |
| 9 000 | 7.6510469604 | 7,650999784469604 | 1.137 | | 5,100667 | 1.492 | 3,187917 | 2,684 | 2,013421 | 3,841 |

*Таблиця 1. Час обчислення та оцінка прискорення в залежності від кількості потоків та розміру матриці*

*Графік 1. Залежність часу обчислення від розміру матриці при послідовному алгоритмі та при розпаралеленні з використанням різної кількості потоків*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Розмір матриці** | **2 потоки** | | **4 потоки** | | **8 потоків** | |
| **Прискорення** | **Паралельна ефективність** | **Прискорення** | **Паралельна ефективність** | **Прискорення** | **Паралельна ефективність** |
| 1 000 | 1.555 | 1.231 | 2,413 | 2,09931 | 3,834 | 3,33558 |
| 2 000 | 1.529 | 1.224 | 2,517 | 2,18979 | 3,845 | 3,34515 |
| 3 000 | 1.463 | 1.118 | 2,384 | 2,07408 | 3,659 | 3,18333 |
| 4 000 | 1.474 | 1.119 | 2,469 | 2,14803 | 3,891 | 3,38517 |
| 5 000 | 1.536 | 1.228 | 2,446 | 2,12802 | 3,801 | 3,30687 |
| 6 000 | 1.580 | 1.234 | 2,356 | 2,04972 | 3,795 | 3,30165 |
| 7 000 | 1.508 | 1.221 | 2,124 | 1,84788 | 3,848 | 3,34776 |
| 8 000 | 1.505 | 1.221 | 2,364 | 2,05668 | 3,795 | 3,30165 |
| 9 000 | 1.492 | 1.220 | 2,684 | 2,33508 | 3,841 | 3,34167 |

*Таблиця 2. Оцінка прискорення та паралельної ефективності на обчислювальній машині з 4-х ядерним процесором*

*Графік 2. Залежність ефективності обчислення від розміру матриці та кількості потоків*

*Графік 3. Залежність прискорення обчислення від розміру матриці та кількості потоків*

Оцінка паралельного прискорення: , де m – кількість ядер.

Паралельна ефективність вираховувалась за формулою:

**Висновок**

На лабораторній роботі я ознайомився з технологією OpenMP і набув практичних навичок використання. Також я навчився визначати паралельну ефективність, паралельне прискорення. Також було досліджено залежності часу обчислення