МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Звіт із
Розпаралелення додавання/віднімання матриць Лабораторна робота №2
з курсу «Паралельні та розподілені обчислення»

Виконав:
Гуменюк Станіслав
Група Пмі-33с

Оцінка ___
Перевірив:
Пасічник
Т.В.

2024

Завдання:

Напишіть програми обчислення множення двох матриць (послідовний та паралельний алгоритми). Порахуйте час роботи кожної з програм, обчисліть прискорення та ефективність роботи паралельного алгоритму.

В матрицях розмірності (n,m) (m,l) робіть змінними, щоб легко змінювати величину матриці.

Кількість потоків k - також змінна величина. Програма повинна показувати час при послідовному способі виконання програми, а також при розпаралеленні на k потоків.

Програмна реалізація:

Програма написана на пакеті .NET з використанням мови C#.
В меню є вибір функціоналу

```
-> 1. Multiply two random matrices with one thread
   2. Multiply two large random matrices with one thread
   3. Multiply two random matrices with multiple threads
   4. Multiply two large random matrices with multiple threads
   5. Calculate difference
   6. Best efficiency
   Exit
```

1. Множення послідовно для перевірки правильності виконання операції

```
Enter the size of the matrix:
n: 5
m: 3
p: 4
Matrix A:
8 1 1
1 5 8
4 5 1
3 6 1
3 2 5
Matrix B:
3 5 8 2
9 5 0 7
9 4 7 0
Matrix C:
42 49 71 23
120 62 64 37
66 49 39 43
72 49 31 48
72 45 59 20
Press any key to exit...
```

2. Множення великих матриць послідовно для засікання часу

```
Enter the size of the matrix:
n: 1000
m: 600
p: 800
Time elapsed with one thread: 4263ms
Press any key to exit...
```

3. Множення паралельно для перевірки правильності виконання операції

```
Enter the size of the matrix:
n: 5
m: 3
p: 4
Threads k: 3
Matrix A:
6 6 2
8 7 9
0 7 0
1 0 0
4 3 7
Matrix B:
3 5 6 3
8 7 9 9
6 3 5 8
Matrix C:
78 78 100 88
134 116 156 159
56 49 63 63
3 5 6 3
78 62 86 95
Press any key to exit...
```

4. Множення великих матриць паралельно для засікання часу

```
Enter the size of the matrix:
n: 1000
m: 600
p: 800
Threads k: 12
Time elapsed with 12 threads: 951ms
Press any key to exit...
```

5. Розрахунок прискорення множення матриць

```
Enter the size of the matrix:
n: 1000
m: 600
p: 800
Threads k: 12
Time elapsed with one thread: 4544ms
Time elapsed with 12 threads: 955ms
Difference: 4.76
Press any key to exit...
```

6. Найкраще прискорення для різної кількості потоків (2-16)

```
Enter the size of the matrices:
n: 1000
m: 600
p: 800
Calculating efficiency with 2 threads...
Efficiency: 1.65
Calculating efficiency with 3 threads...
Efficiency: 2.36
Calculating efficiency with 4 threads...
Efficiency: 2.85
Calculating efficiency with 5 threads...
Efficiency: 3.63
Calculating efficiency with 6 threads...
Efficiency: 4.11
Calculating efficiency with 7 threads...
Efficiency: 4.09
Calculating efficiency with 8 threads...
Efficiency: 4.34
Calculating efficiency with 9 threads...
Efficiency: 4.01
Calculating efficiency with 10 threads...
Efficiency: 4.45
Calculating efficiency with 11 threads...
Efficiency: 4.39
Calculating efficiency with 12 threads...
Efficiency: 4.41
Calculating efficiency with 13 threads...
Efficiency: 4.44
Calculating efficiency with 14 threads...
Efficiency: 4.32
Calculating efficiency with 15 threads...
Efficiency: 4.35
Calculating efficiency with 16 threads...
Efficiency: 4.45
Best efficiency: 4.45 with 10 threads
Press any key to exit...
```

Код множення матриць послідовно:

```
static int[,] MultiplyMatrices(int[,] A, int[,] B)
{
    int n = A.GetLength(0);
    int m = A.GetLength(1);
    int p = B.GetLength(1);
    int[,] C = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            C[i, j] = 0;
            for (int k = 0; k < m; k++)
            {
                C[i, j] += A[i, k] * B[k, j];
            }
        }
    }
    return C;
}
```

Код множення матриць паралельно з використанням класу Thread

```
static int[,] MultiplyMatrices(int[,] A, int[,] B, int k)
{
    int n = A.GetLength(0);
    int m = A.GetLength(1);
    int p = B.GetLength(1);
    int[,] C = new int[n, p];


    // create threads
    Thread[] threads = new Thread[k];

    int completedThreads = 0;
    for (int i = 0; i < k; i++)
    {
        threads[i] = new Thread((object obj) =>
        {
            for (int j = (int)obj; j < p; j += k)
            {
                for (int l = 0; l < n; l++)
                {
                    C[l, j] = 0;
                    for (int o = 0; o < m; o++)
                    {
                        C[l, j] += A[l, o] * B[o, j];
                    }
                }
            }
            Interlocked.Increment(ref completedThreads);
        });
        threads[i].Start(i);
    }
    // wait for all threads to finish
    while (completedThreads < k) {
        Thread.Sleep(1);
    }

    return C;
}
```

Висновок:
Обрахунок добутку двох матриць є ресурсозатратним процесом для великих розмірів, тому паралельне обчислення допомагає скоротити час на виконання таких операцій. В залежності від потужності комп'ютера, можна використовувати різну кількість потоків. Для мого 16-ти поточного процесору, враховуючи що система забирає декілька потоків собі, найефективнішою конфігурацією виявилась 8-14 потоків