Parallel Programming- Algorithms and Technique

TASK 1

```
using System;
using System.Diagnostics;
using System.Threading;
using System.Threading.Tasks;

namespace Task1
{
    class Program
    {
        static void Main(string[] args)
        {
            long col1 = 100, row1 = 1000;
            long col2 = 1000, row2 = 100;
            long[,] M1 = new long[row1, col1];
            long[,] M2 = new long[row2, col2];
            Random rdn = new Random();
            Console.WriteLine("Generating matrices");
            for (int i = 0; i < row1; i++)
            {
                for (int j = 0; j < col1; j++)
                {
                    M1[i, j] = rdn.Next();
                }
            }
            for (int i = 0; i < row2; i++)
            {
                for (int j = 0; j < col2; j++)
                {
                    M2[i, j] = rdn.Next();
                }
            }
            Stopwatch sw = new Stopwatch();

            Console.WriteLine("sequential");
            sw.Start();
            SerialAlgorithm(M1, M2);
            sw.Stop();
            Console.WriteLine($"sequential matrix multiplication Time = {sw.ElapsedMilliseconds} ms");

            Console.WriteLine("Parallel");
            sw.Restart();
            ParallelAlgorithm(M1, M2);
            sw.Stop();
            Console.WriteLine($"Parallel matrix multiplication Time = {sw.ElapsedMilliseconds} ms");
        }
        public static long[,] SerialAlgorithm(long[,] matrix1, long[,] matrix2)
        {
            long matAcol = matrix1.GetLength(1);
            long matArow = matrix1.GetLength(0);
            long matBcol = matrix2.GetLength(1);
            int matBrow = matrix2.GetLength(0);
            long[,] Product = new long[matArow, matBcol];


            if (matAcol == matBrow)
            {

                for (int i = 0; i < matArow; i++)
                {
                    for (int j = 0; j < matBcol; j++)
                    {
                        long sum = 0;
                        for (int t = 0; t < matBrow; t++)
                        {
                            sum += (matrix1[i, t] * matrix2[t, j]);
                        }
                        Product[i, j] = sum;
                    }
                }
                return Product;
            }
            throw new Exception("This matrices can not be multiplied");

        }
        public static long[,] ParallelAlgorithm(long[,] matrix1, long[,] matrix2)
        {

            long matAcol = matrix1.GetLength(1);
            long matArow = matrix1.GetLength(0);
            long matBcol = matrix2.GetLength(1);
            int matBrow = matrix2.GetLength(0);
            long[,] ParallelProduct = new long[matArow, matBcol];
            if (matAcol == matBrow)
            {
                Parallel.For(0, matArow, i =>
                {
                    for (int j = 0; j < matBcol; j++)
                    {
                        long sum = 0;
                        for (int t = 0; t < matBrow; t++)
                        {
                            sum += (matrix1[i, t] * matrix2[t, j]);
                        }
                        ParallelProduct[i, j] = sum;
                    }
                });
                return ParallelProduct;
            }
            throw new Exception("This matrices can not be multiplied");
        }
    }
}
```

```
Microsoft Visual Studio Debug Console
Generating matrices
sequential
sequential matrix multiplication Time = 2325 ms
Parallel
Parallel matrix multiplication Time = 842 ms
```

Speedup = time(sequential)/time(parallel)

=> 2325ms / 842ms = 2,76128266

Efficiency = (Speedup/number of threads) *100

=> (2,76128266/4) *100 = 69,03%

```
using System;
using System.Diagnostics;
using System.Threading;
using System.Threading.Tasks;

namespace Task1
{
    class Program
    {
        static void Main(string[] args)
        {
            long col1 = 100, row1 = 10000;
            long col2 = 10000, row2 = 100;
            long[,] M1 = new long[row1, col1];
            long[,] M2 = new long[row2, col2];
            Random rdn = new Random();
            Console.WriteLine("Generating matrices");
            for (int i = 0; i < row1; i++)
            {
                for (int j = 0; j < col1; j++)
                {
                    M1[i, j] = rdn.Next();
                }
            }
            for (int i = 0; i < row2; i++)
            {
                for (int j = 0; j < col2; j++)
                {
                    M2[i, j] = rdn.Next();
                }
            }
            Stopwatch sw = new Stopwatch();

            Console.WriteLine("sequential");
            sw.Start();
            SerialAlgorithm(M1, M2);
            sw.Stop();
            Console.WriteLine($"sequential matrix multiplication Time = {sw.ElapsedMilliseconds} ms");

            Console.WriteLine("Parallel");
            sw.Restart();
            ParallelAlgorithm(M1, M2);
            sw.Stop();
            Console.WriteLine($"Parallel matrix multiplication Time = {sw.ElapsedMilliseconds} ms");
        }
        public static long[,] SerialAlgorithm(long[,] matrix1, long[,] matrix2)
        {
            long matAcol = matrix1.GetLength(1);
            long matArow = matrix1.GetLength(0);
            long matBcol = matrix2.GetLength(1);
            int matBrow = matrix2.GetLength(0);
            long[,] Product = new long[matArow, matBcol];


            if (matAcol == matBrow)
            {

                for (int i = 0; i < matArow; i++)
                {
                    for (int j = 0; j < matBcol; j++)
                    {
                        long sum = 0;
                        for (int t = 0; t < matBrow; t++)
                        {
                            sum += (matrix1[i, t] * matrix2[t, j]);
                        }
                        Product[i, j] = sum;
                    }
                }
                return Product;
            }
            throw new Exception("This matrices can not be multiplied");

        }
        public static long[,] ParallelAlgorithm(long[,] matrix1, long[,] matrix2)
        {

            long matAcol = matrix1.GetLength(1);
            long matArow = matrix1.GetLength(0);
            long matBcol = matrix2.GetLength(1);
            int matBrow = matrix2.GetLength(0);
            long[,] ParallelProduct = new long[matArow, matBcol];
            if (matAcol == matBrow)
            {
                Parallel.For(0, matArow, i =>
                {
                    for (int j = 0; j < matBcol; j++)
                    {
                        long sum = 0;
                        for (int t = 0; t < matBrow; t++)
                        {
                            sum += (matrix1[i, t] * matrix2[t, j]);
                        }
                        ParallelProduct[i, j] = sum;
                    }
                });
                return ParallelProduct;
            }
            throw new Exception("This matrices can not be multiplied");
        }
    }
}
```

```
Microsoft Visual Studio Debug Console
Generating matrices
sequential
sequential matrix multiplication Time = 143153 ms
Parallel
Parallel matrix multiplication Time = 77438 ms
```

Speedup = time(sequential)/time(parallel)

=> 143153ms / 77438ms = 1,84861438

Efficiency = (Speedup/number of threads) *100

=> (1,84861438/4) *100 = 46,21%

```
Class Program
{
    0 references
    static void Main(string[] args)
    {
        long col1 = 10000, row1 = 1000;
        long col2 = 1000, row2 = 10000;
        long[,] M1 = new long[row1, col1];
        long[,] M2 = new long[row2, col2];
        Random rdn = new Random();
        Console.WriteLine(value: "Generating matrices");
        for (int i = 0; i < row1; i++)
        {
            for (int j = 0; j < col1; j++)
            {
                M1[i, j] = rdn.Next();
            }
        }
        for (int i = 0; i < row2; i++)
        {
            for (int j = 0; j < col2; j++)
            {
                M2[i, j] = rdn.Next();
            }
        }
        Stopwatch sw = new Stopwatch();

        Console.WriteLine(value: "sequential");
        sw.Start();
        SerialAlgorithm(M1, M2);
        sw.Stop();
        Console.WriteLine(value: $"sequential matrix multiplication Time = {sw.ElapsedMilliseconds} ms");

        Console.WriteLine(value: "Parallel");
        sw.Restart();
        ParallelAlgorithm(M1, M2);
        sw.Stop();
        Console.WriteLine(value: $"Parallel matrix multiplication Time = {sw.ElapsedMilliseconds} ms");
    }
    1 reference
    public static long[,] SerialAlgorithm(long[,] matrix1, long[,] matrix2)
    {
        long matAcol = matrix1.GetLength(dimension: 1);
        long matArow = matrix1.GetLength(dimension: 0);
        long matBcol = matrix2.GetLength(dimension: 1);
        int matBrow = matrix2.GetLength(dimension: 0);
        long[,] Product = new long[matArow, matBcol];


        if (matAcol == matBrow)
        {

            for (int i = 0; i < matArow; i++)
            {
                for (int j = 0; j < matBcol; j++)
                {
                    long sum = 0;
                    for (int t = 0; t < matBrow; t++)
                    {
                        sum += (matrix1[i, t] * matrix2[t, j]);
                    }
                    Product[i, j] = sum;
                }
            }
            return Product;
        }
        throw new Exception(message: "This matrices can not be multiplied");

    }
    1 reference
    public static long[,] ParallelAlgorithm(long[,] matrix1, long[,] matrix2)
    {

        long matAcol = matrix1.GetLength(dimension: 1);
        long matArow = matrix1.GetLength(dimension: 0);
        long matBcol = matrix2.GetLength(dimension: 1);
        int matBrow = matrix2.GetLength(dimension: 0);
        long[,] ParallelProduct = new long[matArow, matBcol];
        if (matAcol == matBrow)
        {
            Parallel.For(fromInclusive: 0, matArow, i =>
            {
                for (int j = 0; j < matBcol; j++)
                {
                    long sum = 0;
                    for (int t = 0; t < matBrow; t++)
                    {
                        sum += (matrix1[i, t] * matrix2[t, j]);
                    }
                    ParallelProduct[i, j] = sum;
                }
            });
            return ParallelProduct;
        }
        throw new Exception(message: "This matrices can not be multiplied");
    }
}
}
```

```
Microsoft Visual Studio Debug Console
Generating matrices
sequential
sequential matrix multiplication Time = 247131 ms
Parallel
Parallel matrix multiplication Time = 149022 ms
```

Speedup = time(sequential)/time(parallel)

=> 247131ms / 149022ms = 1,65835246

Efficiency = (Speedup/number of threads) *100

=> (1,65835246 / 4) * 100 = 41,45 %