Паралельне програмування з використанням бібліотеки TPL та PLINQ

Лекція 6. Базові алгоритми паралельних обчислень

1. Паралельне множення матриць

Матриця в мовах програмування представляється двовимірним масивом. Результатом добутку двох матриць A і B є третя матриця C, елементи якої дорівнюють сумі добутків елементів у відповідному рядку першого множника і стовпця другого.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Кількість стовпців в матриці A повинна співпадати з кількістю рядків в матриці B. Якщо матриця A має розмірність $m \times n$, B — $n \times k$, то розмірність їх добутку $AB = C$ є $m \times k$.

Послідовна реалізація цього завдання може виглядати так:

http://msdn.microsoft.com/ru-ru/library/dd460713.aspx

```
static void MultiplyMatricesSequential(double[,] matA, double[,] matB,
                                       double[,] result)
        {
            int matACols = matA.GetLength(1);
            int matBCols = matB.GetLength(1);
            int matARows = matA.GetLength(0);

            for (int i = 0; i < matARows; i++)
            {
                for (int j = 0; j < matBCols; j++)
                {
                    for (int k = 0; k < matACols; k++)
                    {
                        result[i, j] += matA[i, k] * matB[k, j];
                    }
                }
            }
        }
```

Розпаралелювання полягає в заміні зовнішнього циклу For циклом Parallel.For:

```
 static void MultiplyMatricesParallel(double[,] matA, double[,] matB, double[,]
result)
        {
            int matACols = matA.GetLength(1);
            int matBCols = matB.GetLength(1);
            int matARows = matA.GetLength(0);

            // A basic matrix multiplication.
            // Parallelize the outer loop to partition the source array by rows.
```

```
            Parallel.For(0, matARows, i =>
            {
                for (int j = 0; j < matBCols; j++)
                {
                    // Use a temporary to improve parallel performance.
                    double temp = 0;
                    for (int k = 0; k < matACols; k++)
                    {
                        temp += matA[i, k] * matB[k, j];
                    }
                    result[i, j] = temp;
                }
            }); // Parallel.For
        }
```

В цих методах реалізовані лише методи множення. Самі матриці створюються і заповнюються в інших методах. Нижче наведено повний код програми:

Лістинг 6.1.

```
namespace MultiplyMatrices
{
    using System;
    using System.Collections.Generic;
    using System.Collections.Concurrent;
    using System.Diagnostics;
    using System.Linq;
    using System.Threading;
    using System.Threading.Tasks;

    class Program
    {
        #region Sequential_Loop
        static void MultiplyMatricesSequential(double[,] matA, double[,] matB,
                                               double[,] result)
        {
            int matACols = matA.GetLength(1);
            int matBCols = matB.GetLength(1);
            int matARows = matA.GetLength(0);

            for (int i = 0; i < matARows; i++)
            {
                for (int j = 0; j < matBCols; j++)
                {
                    for (int k = 0; k < matACols; k++)
                    {
                        result[i, j] += matA[i, k] * matB[k, j];
                    }
                }
            }
        }
        #endregion

        #region Parallel_Loop

        static void MultiplyMatricesParallel(double[,] matA, double[,] matB,
double[,] result)
        {
            int matACols = matA.GetLength(1);
            int matBCols = matB.GetLength(1);
            int matARows = matA.GetLength(0);
```

```
            // A basic matrix multiplication.
            // Parallelize the outer loop to partition the source array by rows.
            Parallel.For(0, matARows, i =>
            {
                for (int j = 0; j < matBCols; j++)
                {
                    // Use a temporary to improve parallel performance.
                    double temp = 0;
                    for (int k = 0; k < matACols; k++)
                    {
                        temp += matA[i, k] * matB[k, j];
                    }
                    result[i, j] = temp;
                }
            }); // Parallel.For
        }

        #endregion


        #region Main
        static void Main(string[] args)
        {
            // Set up matrices. Use small values to better view
            // result matrix. Increase the counts to see greater
            // speedup in the parallel loop vs. the sequential loop.
            int colCount = 180;
            int rowCount = 2000;
            int colCount2 = 270;
            double[,] m1 = InitializeMatrix(rowCount, colCount);
            double[,] m2 = InitializeMatrix(colCount, colCount2);
            double[,] result = new double[rowCount, colCount2];

            // First do the sequential version.
            Console.WriteLine("Executing sequential loop...");
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();

            MultiplyMatricesSequential(m1, m2, result);
            stopwatch.Stop();
            Console.WriteLine("Sequential loop time in milliseconds: {0}",
stopwatch.ElapsedMilliseconds);

            // For the skeptics.
            OfferToPrint(rowCount, colCount2, result);

            // Reset timer and results matrix.
            stopwatch.Reset();
            result = new double[rowCount, colCount2];

            // Do the parallel loop.
            Console.WriteLine("Executing parallel loop...");
            stopwatch.Start();
            MultiplyMatricesParallel(m1, m2, result);
            stopwatch.Stop();
            Console.WriteLine("Parallel loop time in milliseconds: {0}",
stopwatch.ElapsedMilliseconds);
            OfferToPrint(rowCount, colCount2, result);

            // Keep the console window open in debug mode.
```

```
            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }


        #endregion

        #region Helper_Methods

        static double[,] InitializeMatrix(int rows, int cols)
        {
            double[,] matrix = new double[rows, cols];

            Random r = new Random();
            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < cols; j++)
                {
                    matrix[i, j] = r.Next(100);
                }
            }
            return matrix;
        }

        private static void OfferToPrint(int rowCount, int colCount, double[,]
matrix)
        {
            Console.WriteLine("Computation complete. Print results? y/n");
            char c = Console.ReadKey().KeyChar;
            if (c == 'y' || c == 'Y')
            {
                Console.WindowWidth = 180;
                Console.WriteLine();
                for (int x = 0; x < rowCount; x++)
                {
                    Console.WriteLine("ROW {0}: ", x);
                    for (int y = 0; y < colCount; y++)
                    {
                        Console.Write("{0:#.##} ", matrix[x, y]);
                    }
                    Console.WriteLine();
                }

            }
        }

        #endregion
    }
}
```