**Data vs. Task Parallelism**

Implement a C# application to **multiply two N by M matrices** using following three programming models:

* Data Parallelism (using Parallel Loops)
* Task Parallelism (using Implicit Task creation)
* Task Parallelism (using Explicit Task creation)

Compute the execution time of these three implementations using 3 sets of randomly generated matrices (of different sizes). Prepare a comparison table to compare the performance of these three techniques (using average completion time).

([Hint](https://csharp-station.com/using-the-stopwatch-class-to-log-time/#%3A~%3Atext%3DTo%20use%20it%2C%20you%20have%2CStop()%3B%20Console): Stopwatch class can be used to find execution time). Conclude your findings.)

Solution :

Data Parallelism

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PDC\_2

{

internal class FileName

{

static void Main(string[] args)

{

Console.WriteLine("Enter the dimensions for Matrix A (N M):");

string[] dimensions = Console.ReadLine().Split(' ');

int n = int.Parse(dimensions[0]);

int m = int.Parse(dimensions[1]);

Console.WriteLine("Enter the dimensions for Matrix B (N M):");

string[] dimensions1 = Console.ReadLine().Split(' ');

int n1 = int.Parse(dimensions1[0]);

int m1 = int.Parse(dimensions1[1]);

int[,] matrixA = new int[n, m];

int[,] matrixB = new int[n1, m1];

int[,] resultMatrix = new int[n, m1];

Console.WriteLine("Enter the elements of matrix A:");

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

{

string[] rowValues = Console.ReadLine().Split(' ');

for (int j = 0; j < m; j++)

{

matrixA[i, j] = int.Parse(rowValues[j]);

}

}

Console.WriteLine("Enter the elements of matrix B:");

for (int i = 0; i < n1; i++)

{

string[] rowValues = Console.ReadLine().Split(' ');

for (int j = 0; j < m1; j++)

{

matrixB[i, j] = int.Parse(rowValues[j]);

}

}

// Perform matrix multiplication using Data Parallelism

Parallel.For(0, n, i =>

{

Parallel.For(0, m1, j =>

{

Parallel.For(0, m, k =>

{

resultMatrix[i, j] += matrixA[i, k] \* matrixB[k, j];

});

});

});

Console.WriteLine("The result of matrix multiplication is:");

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

{

for (int j = 0; j < m1; j++)

{

Console.Write(resultMatrix[i, j] + " ");

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}

Task Parallelism Using Implicit Function

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PDC\_2

{

internal class FileName

{

static void Main(string[] args)

{

Console.WriteLine("Enter the dimensions for Matrix A (N M):");

string[] dimensions = Console.ReadLine().Split(' ');

int n = int.Parse(dimensions[0]);

int m = int.Parse(dimensions[1]);

Console.WriteLine("Enter the dimensions for Matrix B (N M):");

string[] dimensions1 = Console.ReadLine().Split(' ');

int n1 = int.Parse(dimensions1[0]);

int m1 = int.Parse(dimensions1[1]);

int[,] matrixA = new int[n, m];

int[,] matrixB = new int[n1, m1];

int[,] resultMatrix = new int[n, m1];

Console.WriteLine("Enter the elements of matrix A:");

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

{

string[] rowValues = Console.ReadLine().Split(' ');

for (int j = 0; j < m; j++)

{

matrixA[i, j] = int.Parse(rowValues[j]);

}

}

Console.WriteLine("Enter the elements of matrix B:");

for (int i = 0; i < n1; i++)

{

string[] rowValues = Console.ReadLine().Split(' ');

for (int j = 0; j < m1; j++)

{

matrixB[i, j] = int.Parse(rowValues[j]);

}

}

// Perform matrix multiplication using Task Parallelism (using Implicit Task creation)

Parallel.For(0, n, i =>

{

for (int j = 0; j < m1; j++)

{

for (int k = 0; k < m; k++)

{

resultMatrix[i, j] += matrixA[i, k] \* matrixB[k, j];

}

}

});

Console.WriteLine("The result of matrix multiplication is:");

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

{

for (int j = 0; j < m1; j++)

{

Console.Write(resultMatrix[i, j] + " ");

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}

Task Parallelism Using Explict Function

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PDC\_2

{

internal class FileName

{

static void Main(string[] args)

{

Console.WriteLine("Enter the dimensions for Matrix A (N M):");

string[] dimensions = Console.ReadLine().Split(' ');

int n = int.Parse(dimensions[0]);

int m = int.Parse(dimensions[1]);

Console.WriteLine("Enter the dimensions for Matrix B (N M):");

string[] dimensions1 = Console.ReadLine().Split(' ');

int n1 = int.Parse(dimensions1[0]);

int m1 = int.Parse(dimensions1[1]);

int[,] matrixA = new int[n, m];

int[,] matrixB = new int[n1, m1];

int[,] resultMatrix = new int[n, m1];

Console.WriteLine("Enter the elements of matrix A:");

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

{

string[] rowValues = Console.ReadLine().Split(' ');

for (int j = 0; j < m; j++)

{

matrixA[i, j] = int.Parse(rowValues[j]);

}

}

Console.WriteLine("Enter the elements of matrix B:");

for (int i = 0; i < n1; i++)

{

string[] rowValues = Console.ReadLine().Split(' ');

for (int j = 0; j < m1; j++)

{

matrixB[i, j] = int.Parse(rowValues[j]);

}

}

// Perform matrix multiplication using Task Parallelism (using Explicit Task creation)

List<Task> tasks = new List<Task>();

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

{

for (int j = 0; j < m1; j++)

{

int row = i;

int col = j;

tasks.Add(Task.Run(() =>

{

for (int k = 0; k < m; k++)

{

resultMatrix[row, col] += matrixA[row, k] \* matrixB[k, col];

}

}));

}

}

Task.WaitAll(tasks.ToArray());

Console.WriteLine("The result of matrix multiplication is:");

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

{

for (int j = 0; j < m1; j++)

{

Console.Write(resultMatrix[i, j] + " ");

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}