Worksheet – Parallelism

Lernziele:

Sie können die Konzepte der Parallelen Programmierung in C# korrekt anwenden.

Aufgabe 1. Parallelisierung mit Threads

a) Sie finden auf dem Active Directory das Projekt „WS-ParallelismStud.zip“. Es implementiert die folgende sequentielle Implementierung einer Primzahl-Berechnung.

```
private static int[] CalculatePrimesSequential(int range)
{
    var results = new List<int>();
    for (var number = 3; number < range; number++)
    {
        var foundPrime = true;
        for (var divisor = 2; divisor * divisor <= number; divisor++)
            if (number % divisor == 0)
                foundPrime = false;

        if (foundPrime)
            results.Add(number);
    }

    return results.ToArray();
}
```

Führen Sie das Programm aus und tragen Sie die gemessenen Ausführungszeiten für die verschiedenen Zahlenbereiche in die Tabelle am Ende ein. Was stellen Sie fest?

b) Parallelisieren Sie diese Methode von Hand mit mehreren Threads. Implementieren Sie in dazu die neue Methode `CalculatePrimesDemo.CalculatePrimesThreads`. Teilen Sie dabei den Algorithmus in mehrere Blöcke auf, so dass Sie die Arbeit auf alle CPU Cores verteilen können.
Die Resultate können Sie dabei intern zunächst statt in einer `List` in einem `ConcurrentBag` zusammentragen werden. Verwenden Sie **`Thread.Join()`** oder eine **`Barrier`** um auf alle Threads zu warten.

Führen Sie das Programm aus und notieren Sie die benötigte Zeit für die verschiedenen Wertebereiche.

Hinweis: Es geht bei dieser Aufgabe nicht darum, den Algorithmus zu optimieren.

Aufgabe 2. Parallelisierung mit Parallel und Task

Parallelisieren Sie die Methode aus Aufgabe 1 nun mithilfe von `Parallel.For` und/oder `Task`. Implementieren Sie in dazu die neue Methode `CalculatePrimesDemo.CalculatePrimesParallelFor`. Messen Sie wieder die Zeiten, die für die Berechnung von verschiedenen Wertebereichen benötigt wird. Was stellen Sie fest?

Aufgabe 3. Parallelisierung mit PLINQ

Formulieren Sie die Methode aus Aufgabe 1 nun als LINQ-Statement und parallelisieren Sie diese anschliessend mit PLINQ. Implementieren Sie in dazu die neue Methode `CalculatePrimesDemo.CalculatePrimesLinq` und `CalculatePrimesDemo.CalculatePrimesPLinq`. Vergleichen Sie die Performance und die Resultate der LINQ-Version mit der PLINQ-Variante. Was stellen Sie fest?

Gemessene Zeiten in [ms]

Wertebereich	Sequentiell	Threading	Parallel.For	LINQ	PLINQ
[3..20]					
[3..200]					
[3..2'000]					
[3..20'000]					
[3..200'000]					
[3..2'000'000]					