**Task Parallelism. Работа с Task в C#**

Паралелната библиотека за задачи (Tasks) в C# се основава на концепцията за задача, която представлява асинхронна операция. В някои отношения една задача наподобява нишка, но на по-високо ниво на абстракция. Терминът паралелизъм на задачите се отнася до една или повече независими задачи, които се изпълняват едновременно. Задачите предоставят две основни предимства:

* По-ефективно и по-гъвкаво използване на системните ресурси.

Зад кулисите задачите са поставени на опашка в ThreadPool, който е подобрен с алгоритми, които определят и се приспособяват към броя на нишките и които осигуряват балансиране на натоварването, за да се увеличи производителността. Това прави задачите сравнително леки и можете да създадете много от тях, за да постигнете фин паралелизъм.

* Повече програмен контрол, отколкото е възможно с нишка или работен елемент.

Задачите и рамката, изградена около тях, осигуряват богат набор от приложни програмни интерфейси (API), които поддържат чакането, анулирането, продълженията, надеждното управление на изключенията, подробното състояние, персонализирания график и др.

ThreadPool - Предоставя набор от нишки, които могат да бъдат използвани за изпълнение на задачи, публикуване на работни елементи, обработване на асинхронни вход / изход, изчакване от името на други нишки и обработка на таймерите.

Начини за създаване и стартиране на нишки:

1. Създаване и стартитане по-късно

*// use an Action delegate and named method*

*Task task1 = new Task(new Action(printMessage));*

*// use an anonymous delegate*

*Task task2 = new Task(delegate { printMessage() });*

*// use a lambda expression and a named method*

*Task task3 = new Task(() => printMessage());*

*// use a lambda expression and an anonymous method*

*Task task4 = new Task(() => { printMessage() });*

*// Launch the tasks*

*t1.Start();*

*t2.Start();*

*t3.Start();*

*t4.Start();*

1. Създаване и стартиране едновременно(препоръчително);

*// Construct a started task*

*Task t2 = Task.Factory.StartNew(() => printMessage());*

*// Construct a started task using Task.Run.*

*Task t3 = Task.Run(() => printMessage());*

Допълнителна информация на:

<https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming>

<https://www.c-sharpcorner.com/article/task-and-thread-in-c-sharp/>

Задача:

1. Да се демонстрира работа със задачи в C# с прост пример.
2. Да се напише програма, която да раздели умножението на масив в толкова задачи, колкото са процесорите на компютъра.

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Program

{

static void Main()

{

TaskParallelismSimpleExample();

double[] array = new double[10000];

for (int i = 0; i < array.Length; i++)

array[i] = 1;

TaskParallelismArrayExample(array, 8.999);

Console.ReadLine();

}

static void MultiplyArray(double[] array, double multuplyDouble, int startIndex, int endIndex)

{

Console.WriteLine("MultiplyArray. Task={0}, Thread={1}, start={2}, end={3}",

Task.CurrentId,

Thread.CurrentThread.ManagedThreadId,startIndex,endIndex);

for (int i = startIndex; i < endIndex; i++)

{

array[i] = array[i] \* multuplyDouble;

}

}

static void TaskParallelismArrayExample(double[] array, double factor)

{

int degreeOfParallelism = Environment.ProcessorCount;

Task[] tasks = new Task[degreeOfParallelism];

for (int taskNumber = 0; taskNumber < degreeOfParallelism; taskNumber++)

{

// capturing taskNumber in lambda wouldn't work correctly

int taskNumberCopy = taskNumber;

tasks[taskNumber] = Task.Factory.StartNew(

() =>

MultiplyArray(array, 88.999, array.Length \* taskNumberCopy / degreeOfParallelism, array.Length \* (taskNumberCopy + 1) / degreeOfParallelism)

);

}

// Wait for the tasks to finish.

Task.WaitAll(tasks);

}

static void TaskParallelismSimpleExample()

{

// Construct a started task

Task t1 = Task.Factory.StartNew(

() =>

{

Console.WriteLine("TaskParallelismSimpleExample, Task={0}, Thread={1}", Task.CurrentId,

Thread.CurrentThread.ManagedThreadId);

});

Console.WriteLine("t1 has been launched. Task={0}, Thread={1}", Task.CurrentId, Thread.CurrentThread.ManagedThreadId);

// Wait for the task to finish.

t1.Wait();

//Construct a started task using Task.Run.

Task t2 = Task.Run(() =>

{

Console.WriteLine("t2 has been launched. Task={0}, Thread={1}", Task.CurrentId, Thread.CurrentThread.ManagedThreadId);

});

// Wait for the task to finish.

t2.Wait();

}

}

}