# Workshop: Task Parallel Library (TPL)

Einführung

Bei der TPL handelt es sich um eine Bibliothek für die Entwicklung multi-thread-fähiger Anwendungen. Sie steht seit Visual Studio 2010 und dem .NET Framework 4.0 zur Verfügung. Der große Vorteil für den Entwickler liegt darin, dass auf das Management jedes einzelnen Threads verzichtet werden kann. Stattdessen liegt der Fokus auf der Aufgabenstellung und der Definition von Tasks und deren Synchronisation. Durch die TPL erfolgt das Erstellen der Threads automatisch während der Laufzeit unter Berücksichtigung der optimalen Hardwareausnutzung. D.h. bei einem Vierkern-Prozessor würde der Task automatisch in vier Threads aufgeteilt und abgearbeitet, bei einem Achtkern-Prozessor dementsprechend automatisch in acht Threads usw. [1].

Grundsätzlich kann zwischen Datenparallelismus und Taskparallelismus unterschieden werden. Im folgenden Workshop wird auf diese beiden Konzepte in Zusammenhang mit der TPL näher eingegangen und jeweils mit einem oder mehreren Beispielen veranschaulicht.

Data Parallelism

Unter Datenparallelismus versteht man eine Anwendung, in denen der gleiche Vorgang gleichzeitig d.h. parallel auf einen Teilbereich der gesamten Datenmenge (z.B. Array) angewendet wird. Dadurch können mehrere Threads gleichzeitig auf verschiedene Datensegmente angewendet werden [2].

In der TPL wird dieser Anwendungsfall durch die Klasse *System.Threading.Tasks.Parallel* unterstützt. Diese Klasse stellt methodenbasierte parallele Implementierungen von Schleifen bereit. Durch Verwendung der TPL müssen keine Threads oder Arbeitsaufgaben in die Warteschlange eingereiht werden, die Schleifenlogik für eine parallele *for* oder *foreach*-Schleife bleibt weitgehend die gleiche wie bei einer sequenziellen Implementierung. Grundsätzlich sind auch keine Sperren erforderlich.

Um die Einfachheit der TPL zu veranschaulichen ist in *Abbildung 1* ein einfaches Beispiel einer sequenziellen *foreach*-Schleife und deren paralleler Implementierung abgebildet. Bei der parallelen-Schleife wird die Datenquelle (z.B. Array) partitioniert, sodass mehrere Teile der Datenquelle gleichzeitig bearbeitet werden können. Der Taskplaner arbeitet dabei im Hintergrund und partitioniert die Aufgabe mit Berücksichtigung der Systemressourcen und der Arbeitsauslastung [2]

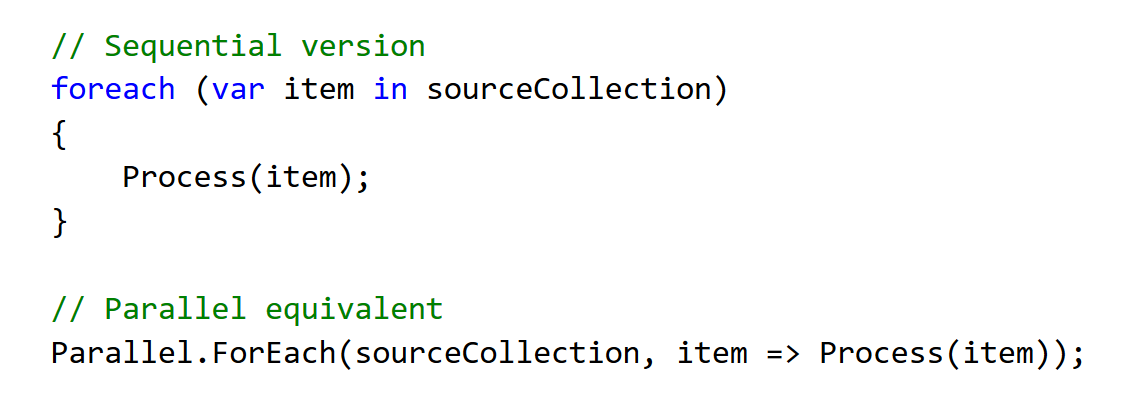


Abbildung 1 - Sequenzielle und parallele Implementierung (mit TPL) einer einfachen foreach-Schleife [2]

Weiterführende Informationen:

Die parallelen Implementierungen der *for* bzw. *foreach*-Methode verfügen zudem über Überlagerungen für:

* das Anhalten bzw. Unterbrechen der Schleifenausführung,
* die Zustandsüberwachung der Schleife in anderen Threads,
* das Beibehalten des lokalen Threadzustandes,
* das Abschließen lokaler Threadobjekte,
* die Steuerung des Parallelitätsgrades [2].

Die oben genannten Funktionalitäten werden in der TPL durch folgende Hilfstypen ermöglicht: *ParallelLoopState, ParallelOptions, ParallelLoopresult, CancellationToken* und *CancellationTokenSource* [2]*.*

Weitere Beispiele für die Implementierung von Datenparallelismus mit Hilfe der TPL sind unter folgendem Link abrufbar: <https://msdn.microsoft.com/de-DE/library/dd537608(v=vs.110).aspx> (aufgerufen am 18.11.2017)

Task Parallelism

Einführung (was ist es wofür einfaches Beispiel)

Example mit allen (aus Visual Studio Vorlage Programming)

Aufgabenstellung zum selber Lösen

Zum Schluss Beispiellösng vorzeigen

( DataFlow

Einführung (was ist es wofür einfaches Beispiel)

Example mit allen (aus Visual Studio Vorlage Programming)

Aufgabenstellung zum selber Lösen

Zum Schluss Beispiellösng vorzeigen

)

Mögliche Pitfalls in Data und Task Paralleisierung

# Datenfluss ()TPL Dataflow

Ist eine sehr wenig genutzte Library (nur 379 Questions on Stack Overflow stand 16.11.2017)

Für was verwende ich die Datenfluss Lib?

Besteht aus Verschiedenen Blöcken

Pufferblöcke

BufferBlock (wichtig)

BroadcastBlock

WriteOnceBlock

Ausführungsblöcken

ActionBlock (wichtigster im Ganzen)

TransformBlock

TransformManyBlock

Gruppierungsblöcke

BatchBlock

JainBlock

BatchedJoinBlock

kann wie Lego zusammengebaut werden

Zusammenfassung 2min

# Mögliche Pitfalls in Data und Task Paralleisierung

* Gehen Sie nicht davon aus, dass eine parallele Ausführung immer schneller ist.
* Vermeiden Sie es, in gemeinsam genutzte Speicherpositionen zu schreiben.
* Vermeiden Sie eine zu starke Parallelisierung.
* Vermeiden Sie den Aufruf nicht threadsicherer Methoden.
* Beschränken Sie Aufrufe auf threadsichere Methoden.
* Seien Sie vorsichtig, wenn Sie in Delegaten warten, die von Parallel.Invoke aufgerufen werden.
* Gehen Sie nicht davon aus, dass Iterationen von "ForEach", "For" und "ForAll" immer parallel ausgeführt werden.
* Vermeiden der Ausführung paralleler Schleifen im UI-Thread

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | A. Kühnel, Visual C# 2010, Bonn: Galileo Computing, 2010. |
| [2] | Microsoft, „https://docs.microsoft.com,“ Microsoft, 30 3 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl. [Zugriff am 19 11 2017]. |