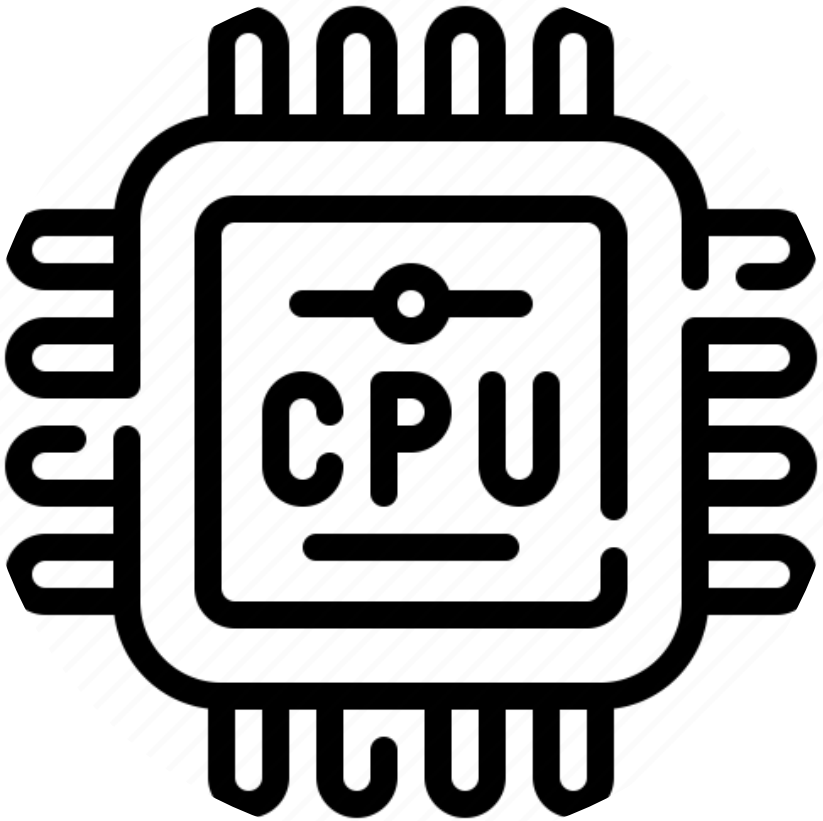
XOLUTION
Excellence in IT

Asynchronous and parallel programming

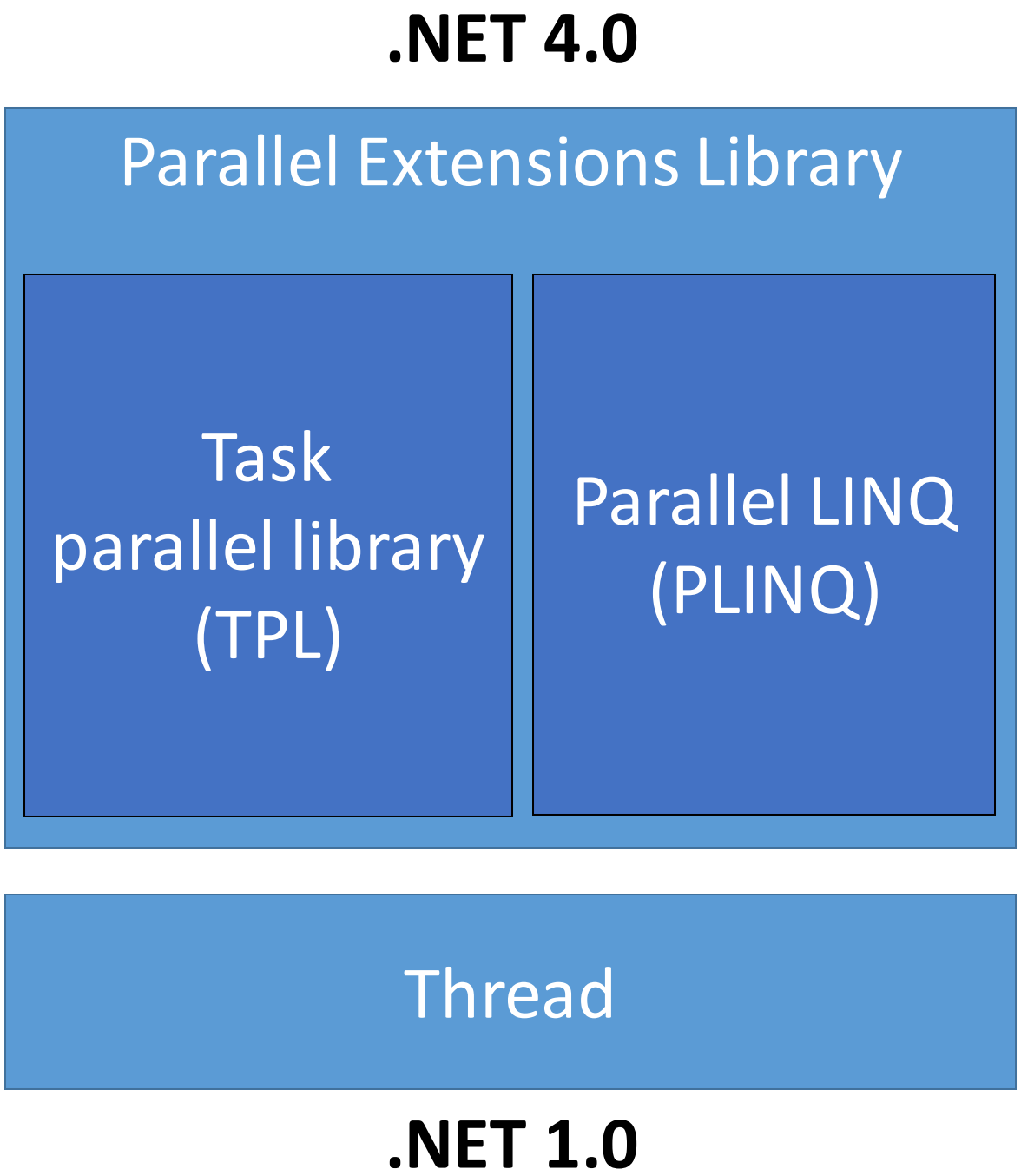
Ing. Tomáš Andrek

Basics

- **Core**: is an individual processor within a CPU.
- **Thread**: is a single sequential flow of control within a program.
- **Asynchronous programming**: non-blocking, without waiting
- **Parallel programming**: run multiple things at the same time
- **Synchronization**: is the concurrent execution of two or more threads that share critical resources.



Async and parallel in .NET



Async / await
(C# 5.0 and .NET Framework 4.5)

- **Async / await**: asynchronous? – YES, parallel? - MAYBE
- **Syntax sugar**: Older techniques: callbacks, Background worker, etc.
- **Await?** We can await every function which returns Task
- **Async?** A function which uses await must be marked as Async
- **Task?** If a function is marked as Async it returns a task

How it works?

StartButton_Click event handler

```
async Task<int> AccessTheWebAsync()
{
    HttpClient client = new HttpClient();

    Task<string> getStringTask = client.GetStringAsync("http://msdn.microsoft.com");

    DoIndependentWork();

    string urlContents = await getStringTask;

    return urlContents.Length;
}

void DoIndependentWork()
{
    resultsTextBox.Text += "Working . . . . . .\r\n";
}
```

Task<string> HttpClient.GetStringAsync(string url))

1 2 3 4 5 6 7 8

Normal processing

Yielding control to caller at an await

Resuming a suspended process

Examples

Example 1 – The difference between synchronous and asynchronous

Example 2 – Real asynchronous and effective

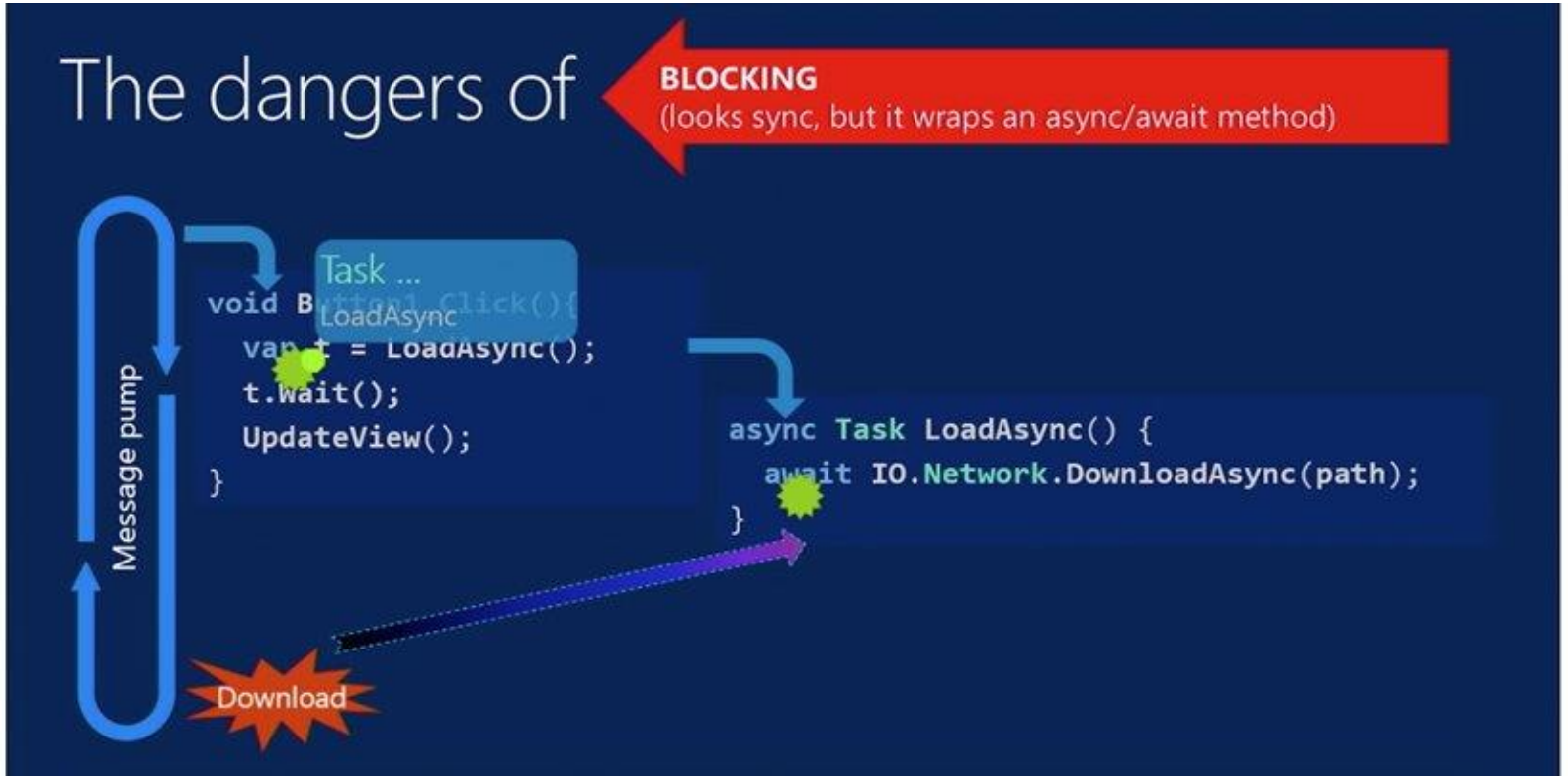
Example 3 – Checking if everything has been done. Task.WhenAll(

Example 4 – Checking if anything has been done. .Any()

XOLUTION
Excellence in IT

Example 5

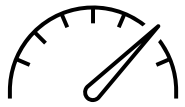
Async / Await – bad practises



1. - DO NOT block asynchronous method by calling .Result or .Wait
2. - DO NOT use type **void** together with **async** (exceptions, state machine)
3. - DO NOT wrap synchronous code into asynchronous

Example 6

Async / Await – When and Where ???



- From the bottom to the top
- **When?** Do I really have an asyc. operation? Vs. Want to run in parallel
- **When?** Performance can be higher or lower!
- **Where?** Desktop apps -> UIThread
- **Where?** Web apps -> go to the thread pool if idle and be used for sth. else
  - Async != faster. In fact, async is *slower*.
  - Utilize resources *more efficiently*
  - Depends on Backend

XOLUTION
Excellence in IT

Async / Await Various Techniques

Example 10 – Task continuation

Example 11 – Task cancellation

Example 12 — Progress reporting

XOLUTION
Excellence in IT

ConfigureAwait(false) - What's that ???

- Defines *"**Synchronization context**"* -> contains Culture, HttpContext
- Synchronization context ends at the end of a method
- Default value: Standard .NET = true
  .NET Core = true
  ASP.NET Core = false (no synch. context!!!)
- Despite ASP.NET Core always use Configure.Await - .NET Standard
- For libraries use **false,** On app level use **true** if necesssary, otherwise **false**

Parallelism and thread synchronization

Single thread	Multiple threads
• Lock • Monitor • Mutex	• Semaphore • Reader/Writer lock

Example 20b - Real example

Example 21 - Deadlocks

Example 20a - Lock and atomic operations

Parallel extensions and TPL

- **Parallel.Invoke**
  Executes an array of Action delegates in parallel, and then waits for them to complete

- **Parallel.For**
  Parallel equivalent of a C# for loop

- **Parallel.ForEach**
  Parallel equivalent of a C# foreach loop

Example 22 - Parallel.ForEach

Sources

- **Examples - source**
  - https://github.com/tomasandrek/async-and-parallel-programming-presentation
- **Docker**
  - https://www.docker.com/
- **.NET Core**
  - https://dotnet.microsoft.com/download
- **Visual Studio Code**
  - https://code.visualstudio.com/
- **Linux – Debian :-)**
  - https://www.debian.org/

XOLUTION
Excellence in IT

Thank you

Ing. Tomáš Andrek