DotNet 2021

ONLINE TECH CONFERENCE

#DotNet2021

22nd June 2021

Introduction to native compilation in Dotnet

www.dotnet2021.com

DotNet 2021

ORGANIZATION

plain concepts

IN COOPERATION WITH

FUNDACIÓN GOMAESPUMA

"Educando con una sonrisa."

SPONSORS

Microsoft

intelequia

Devs DNA™

My Public Inbox®

#DotNet2021

Carlos Landeras

General Manager at Plain Concepts

Cloud and new technologies passionate.

Working with Microsoft technology since 2008.

Working at Plain Concepts since 2016.

Agenda

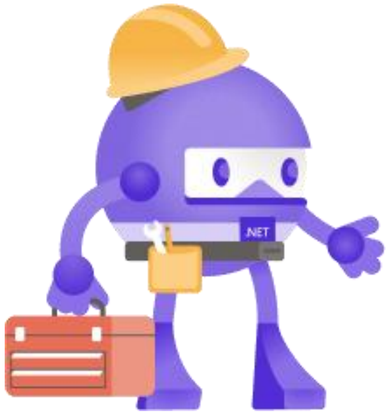
- An introduction to AOT (Ahead-of-time) compilation
- Scenarios
- JIT vs AOT (What are the differences)
- Startup time and performance
- Compiling with Native AOT (Demo)
- Cross-architecture compilation (Demo)
- Ahead of time compilation + Docker (Demo)
- Interop with Rust (Rust to C# and C# to Rust demos)

Introduction to native compilation

- Native AOT .NET Runtime can compile dotnet applications into a native single-file executable (improved startup time and performance)

- It can also produce standalone dynamic or static libraries that can be consumed by applications written in other programming languages. (Consume C# from C++, rust, etc).

- Cross compilation: There are dedicated nuget feeds to compile in Linux, macOS and Windows x64



Scenarios

- Copying a single file executable from one machine and run on another (of the same kind) without installing a .NET runtime.
- Creating and running a docker image that contains a single file executable (one file in addition to Ubuntu).
- Compiling C# libraries into dynamic or static libraries, so they can be consumed by other languages like C++ or rust without having to use COM or library wrappers.

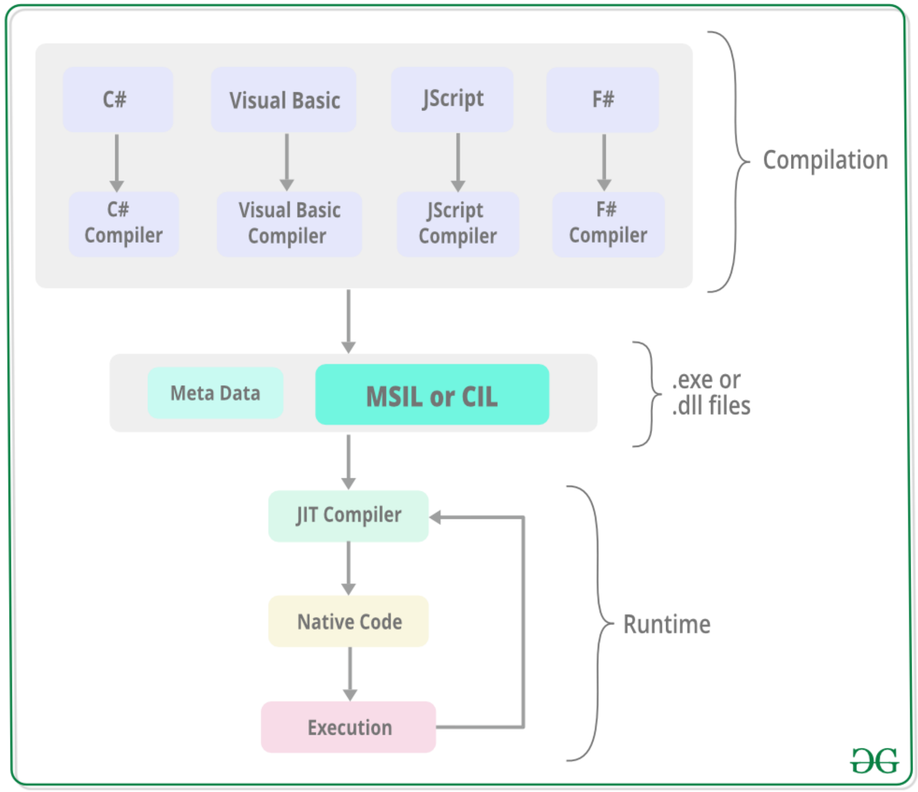
JIT vs AOT

JIT = Just in time compilation

- When the C# compiler compiles our source code it generates **IL** assemblies.

  Common **I**ntermediate **L**anguage (CIL) bytecode

- MSIL is a CPU-independant set of instructions that can be converted to native code.
- Our assemblies contents are split in two categories:
  - Code of our application
  - Medatada about our code (types, base classes, interfaces, methods, fields, etc).
- When we execute a dotnet application, the runtime converts MSIL code into native code



Source: geeksforgeeks

JIT benefits and drawbacks

Benefits	Drawbacks
Independant of the hardware or OS the code will run on	The Runtime needs a type loading step to compute the information necessary to execute the program. A lot of things need to happen before the runtime actually executes our first line of code.
Great version resiliency (high level intermediate language)	Data structures are not fully optimized like in other native compiled languages

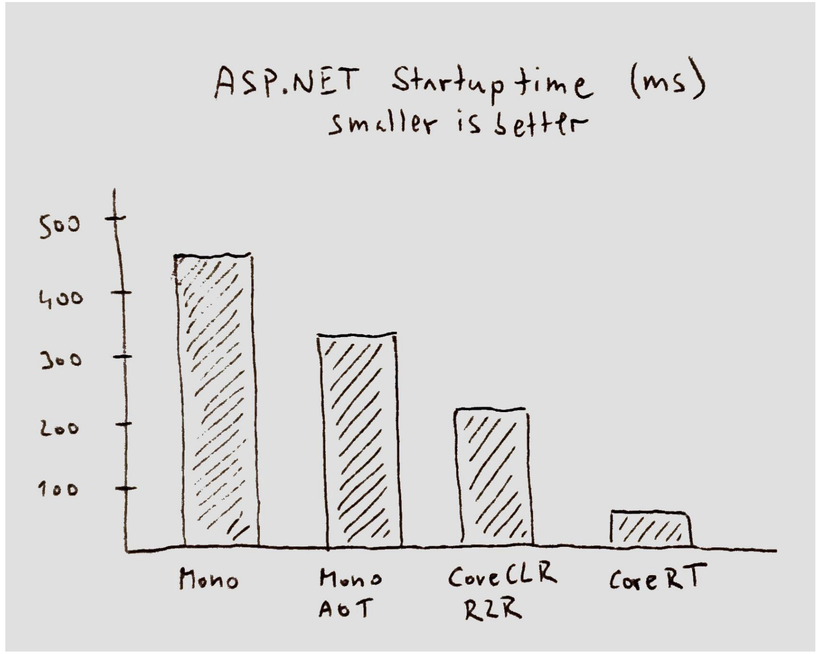
AOT

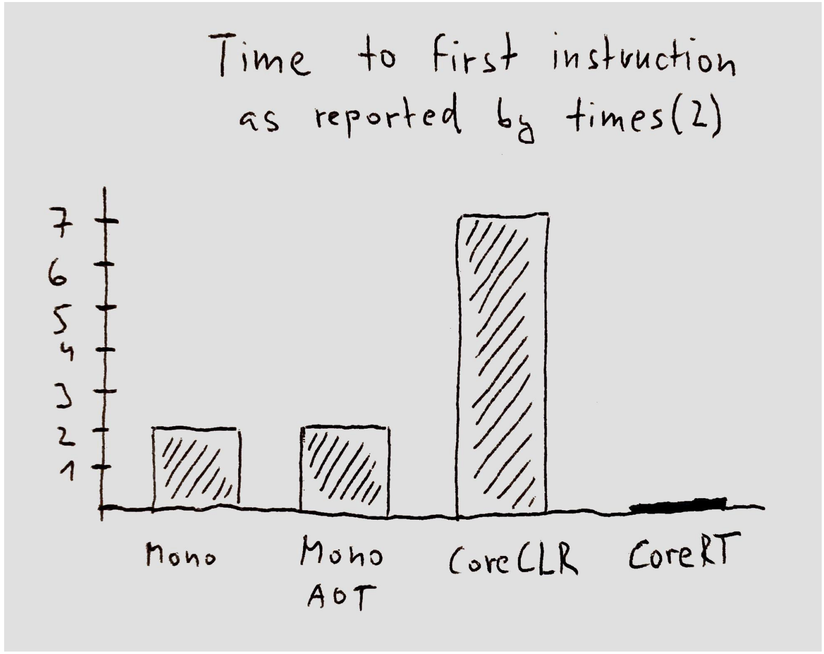
Ahead of time compilation

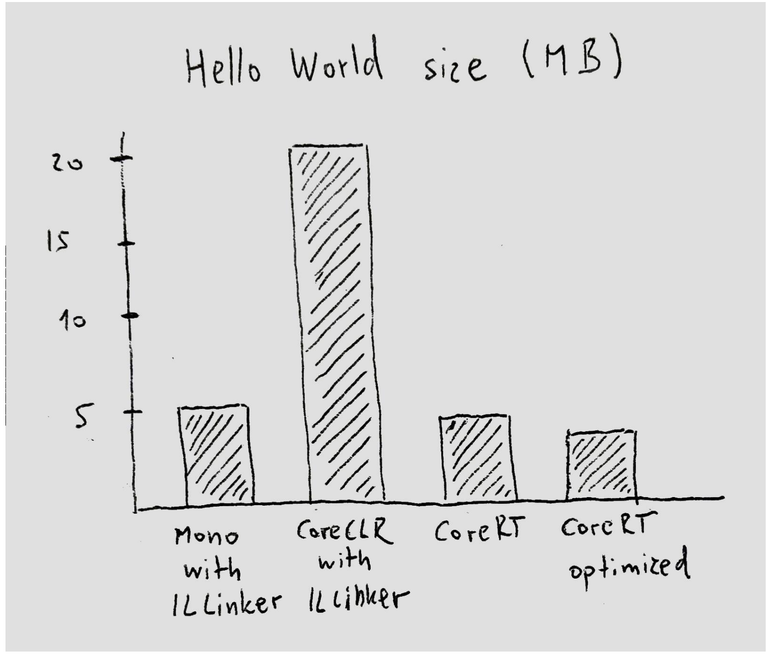
- AOT emits platform specific native code.
- We can run self-contained native executables without installing the .NET runtime in the operative system
- Note: Our program is as hard to decompile as a c++ native executable. Reflector no longer works :P

AOT benefits and drawbacks

Benefits	Drawbacks
Improved performance and reduced startup time as we do not have to execute all the preliminary JIT compiler steps before running our code.	No Support for Dynamic loading **Assembly.LoadFile** No Support for runtime code generation **System.Reflection.Emit**
Smaller compilation output sizes.	Reflection based assemblies can force us to declare Assembly directives to help the compiler find types that should be analyzed







Source: **@MStrehovsky**

Show me the code !

- Compiling with Native AOT
- Cross-platform compilation
- Ahead of time compilation + Docker (Demo)
- Interop with Rust (Rust to C# and C# to Rust demos)

DotNet 2021

ONLINE TECH CONFERENCE

www.dotnet2021.com

#DotNet2021

Thanks and ...
See you soon!

Thanks also to the sponsors. Without whom this would not have been posible.