Microsoft

.NET Runtime on mobile:
Journey with NativeAOT on iOS

Ivan Povazan

.NET runtime(s)

Question

- How many .NET runtimes does Microsoft ship?

Supported platforms

- Desktop (Windows, Linux, MacOS, …) – **CoreCLR**
- Mobile (Android, iOS, …), WebAssembly, … – **Mono**

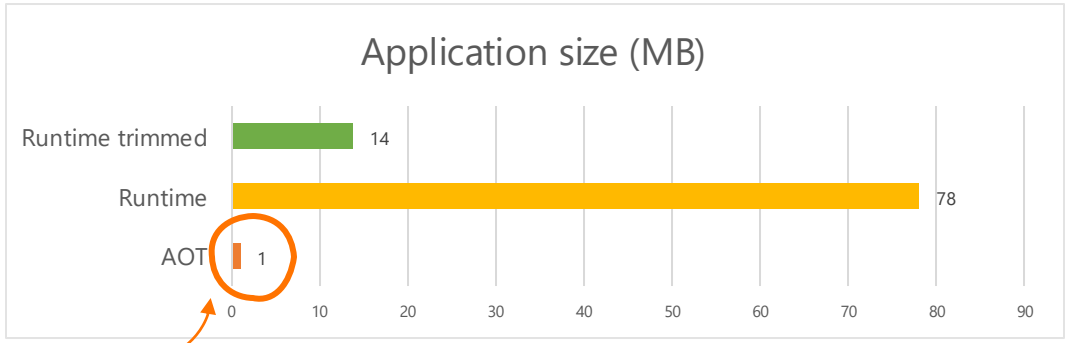
Runtimes and execution engines (performance++)

- **CoreCLR** (RyuJIT, R2R, dynamic PGO)
- **Mono** (miniJIT, interpreter, AOT, LLVM AOT)

NativeAOT

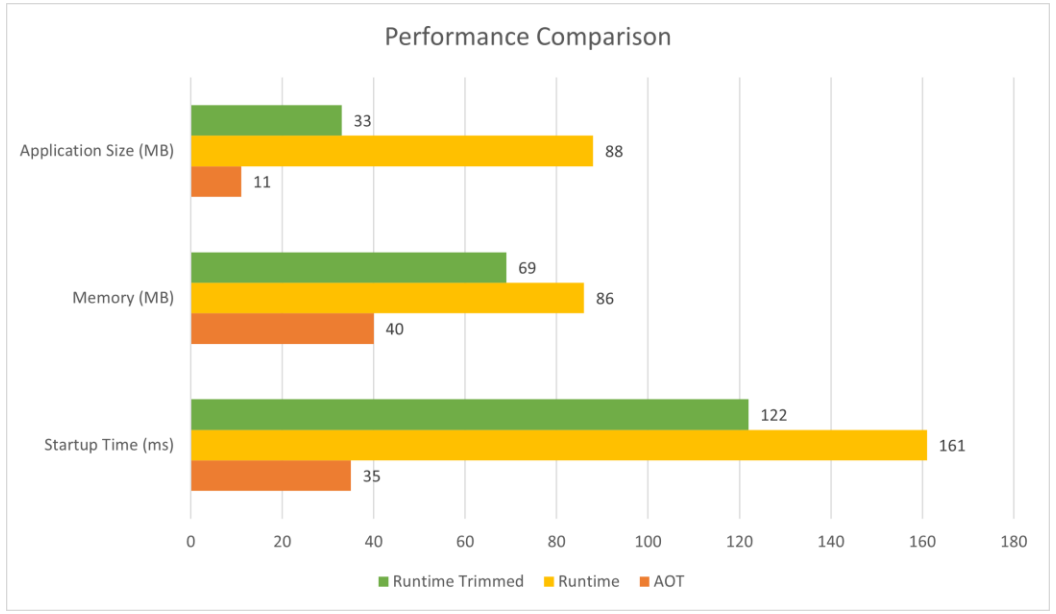
Desktop performance: CoreCLR vs NativeAOT

Console



Yes, this is 1MB NativeAOT app

ASP.NET



- ✅ Application size – a fraction of the full app
- ✅ Less memory use – up to 2x less memory usage
- ✅ Startup time – 4-5x faster start up

NativeAOT magic

- Runtime – stripped-down version of CoreCLR runtime
- AOT compiler (built-in trimmer, RyuJIT compiler)
- Base class and native libraries

Characteristics:

- Ahead-of-time compilation and metadata transformation
- **No managed assemblies**
- **Full trimming and full program optimization**
  - Aggressive removal of unreferenced code
  - Entry points: Main and **UnmanagedCallersOnly** attribute
- No dynamic code
- No unconstrained reflection
- Pure native binaries (no managed debugging)
- Programs must be AOT and trim-compatible (0 warnings)
- **No trimmer extensions**

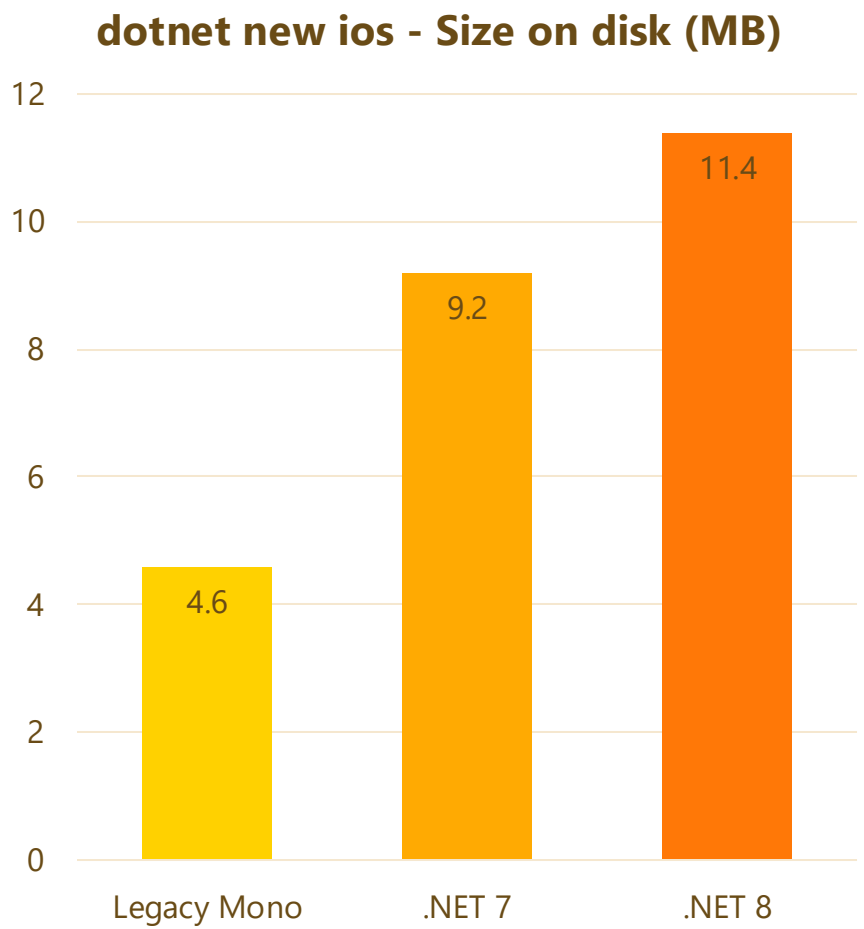
iOS performance and Mono

iOS requirements

- Hardened runtime – no JIT
- Applications: **small** + **fast**

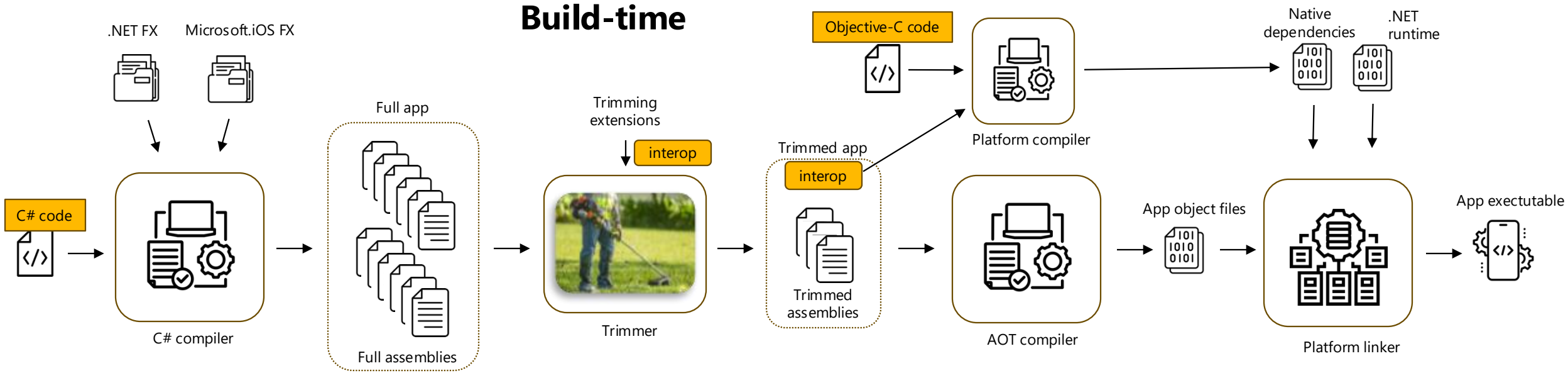
Mono

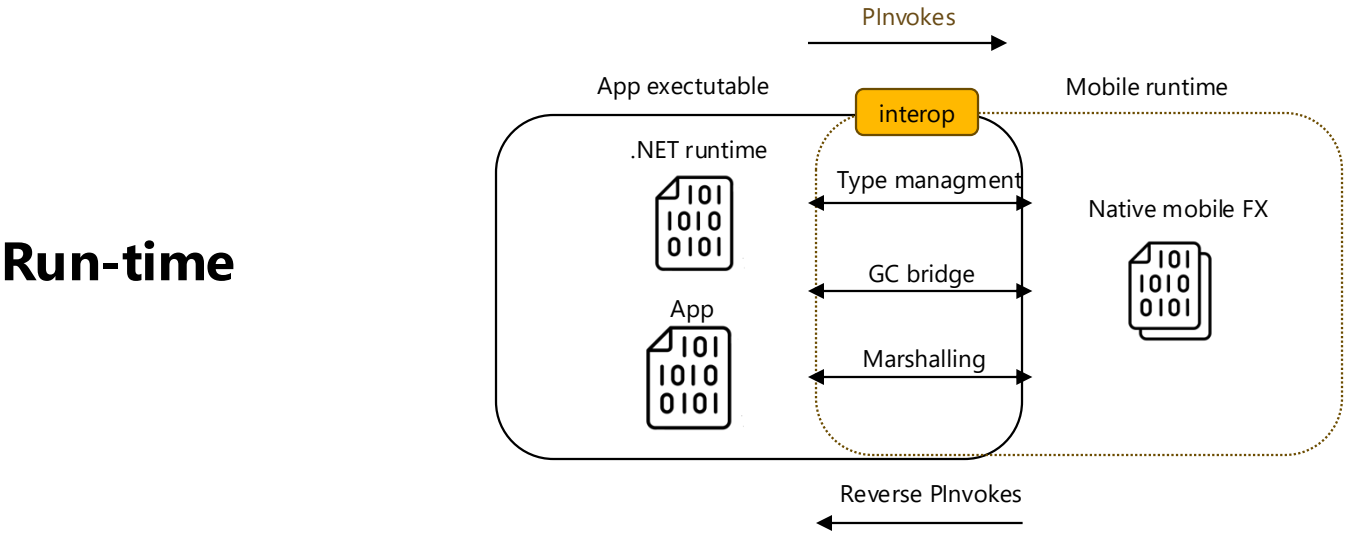
- AOT compiler + trimmer (trimming extensions)
- No full program optimization
- Generics – large and slow fallbacks
- Depends on assembly metadata
- Versatile – interpreter (dynamic code)
- Struggle with performance ☹



Can NativeAOT help?

iOS App model and Mono





```
using Foundation;
...
[Register ("MyNSObject")]
public class MyNSObject : NSObject
{
    public override string Description
        => $".NET Meetup is awesome: {base.Description}";

    [Export ("answer")]
    public int Answer { get; } = 42;

    [DllImport("__Internal")]
    extern public static void callObjectiveC(IntPtr handle);
}

bject myNSObject = new ();
MyNSObject.callObjectiveC (myNSObject.Handle);
```

ToString() == Description

Expose a public property

interop

C# code

```
#import <Foundation/Foundation.h>

@interface MyNSObject : NSObject
-(int) answer;
@end

void callObjectiveC (MyNSObject* mynsobj)
{
    NSObject *nsobj = [[NSObject alloc] init];

    NSLog (@"nsobj:   %@", [nsobj description]);
    NSLog (@"mynsobj: %@", [mynsobj description]);
    NSLog (@"answer:  %d", [mynsobj answer]);
}
```

Objective-C code

```
@implementation MyNSObject {
    ...
    -(int) answer
    {
        static MonoMethod *managed_method = NULL;
        return native_to_managed_trampoline_5 (self, _cmd, &managed_method, 0x702);
    }
    ...

static int native_to_managed_trampoline_5 (
    id self, SEL _cmd, MonoMethod **managed_method_ptr, uint32_t token_ref)
{
    // get method from metadata token, marshalling, exception handling
    GCHandle refl_method_handle = xamarin_get_method_from_token (token_ref, &excp_gchandle);
    ...
    managed_method = xamarin_get_reflection_method_method (refl_method);
    ...
    retval = mono_runtime_invoke (managed_method, mthis, arg_ptrs, &excp);
}
```

```
default 01:58:28.806338+0100    Playground  nsobj:   <NSObject: 0x3032e7140>
default 01:58:28.806566+0100    Playground  mynsobj: .NET Meetup is awesome: <MyNSObject: 0x3030d3540>
default 01:58:28.806770+0100    Playground  answer:  42
```

C# code

```
using Foundation;
...
[Register ("MyNSObject")]
public class MyNSObject : NSObject
{
    public override string Description
        => $".NET Meetup is awesome: {base.Description}";

    [Export ("answer")]
    public int Answer { get; } = 42;

    [DllImport("__Internal")]
    extern public static void callObjectiveC(IntPtr handle);
}
...
MyNSObject myNSObject = new ();
MyNSObject.callObjectiveC (myNSObject.Handle);
```

Not referenced from managed!

Objective-C code

```
#import <Foundation/Foundation.h>

@interface MyNSObject : NSObject
-(int) answer;
@end

void callObjectiveC (MyNSObject* mynsobj)
{
    NSObject *nsobj = [[NSObject alloc] init];

    NSLog (@"nsobj:   %@", [nsobj description]);
    NSLog (@"mynsobj: %@", [mynsobj description]);
    NSLog (@"answer:  %d", [mynsobj answer]);
}
```

interop

```
@implementation MyNSObject {
    ...
    -(int) answer
    {
        static MonoMethod *managed_method = NULL;
        return native_to_managed_trampoline_5 (self, _cmd, &managed_method, 0x702);
    }
    ...

static int native_to_managed_trampoline_5 (
    id self, SEL _cmd, MonoMethod **managed_method_ptr, uint32_t token_ref)
{
    // get method from metadata token, marshalling, exception handling
    GCHandle refl_method_handle = xamarin_get_method_from_token (token_ref, &excp_gchandle);
    ...
    managed_method = xamarin_get_reflection_method_method (refl_method);
    ...
    retval = mono_runtime_invoke (managed_method, mthis, arg_ptrs, &excp);
}
```

Metadata token reference!

NativeAOT:

- aggressive trimming
- no source metadata

C# code

```
using Foundation;
...
[Register ("MyNSObject")]
public class MyNSObject : NSObject
{
    public override string Description
        => $".NET Meetup is awesome: {base.Description}";

    [Export ("answer")]
    public int Answer { get; } = 42;

    [DllImport("__Internal")]
    extern public static void callObjectiveC(IntPtr handle);
}
...
MyNSObject myNSObject = new ();
MyNSObject.callObjectiveC (myNSObject.Handle);
```

Objective-C code

```
#import <Foundation/Foundation.h>

@interface MyNSObject : NSObject
-(int) answer;
@end

void callObjectiveC (MyNSObject* mynsobj)
{
    NSObject *nsobj = [[NSObject alloc] init];

    NSLog (@"nsobj:   %@", [nsobj description]);
    NSLog (@"mynsobj: %@", [mynsobj description]);
    NSLog (@"answer:  %d", [mynsobj answer]);
}
```

NativeAOT:
- UCO entry points

interop - managed

```
// Playground.MyNSObject.__Registrar_Callbacks__
internal sealed class __Registrar_Callbacks__
{
    ...
    [UnmanagedCallersOnly(EntryPoint = "callback_6_Playground_MyNSObject_get_Answer")]
    public unsafe static int callback_6_Playground_MyNSObject_get_Answer(
        IntPtr pobj, IntPtr sel, IntPtr* exception_gchandle)
    {
        // marshalling, exception handling
        try
        {
            MyNSObject nSObject = Runtime.GetNSObject<MyNSObject>(pobj, sel, ... )
            return nSObject.Answer;
        }
        catch (Exception ex)
        {
            *exception_gchandle = Runtime.AllocGCHandle(ex);
        }
        return default(int);
    }
}
```

interop - native

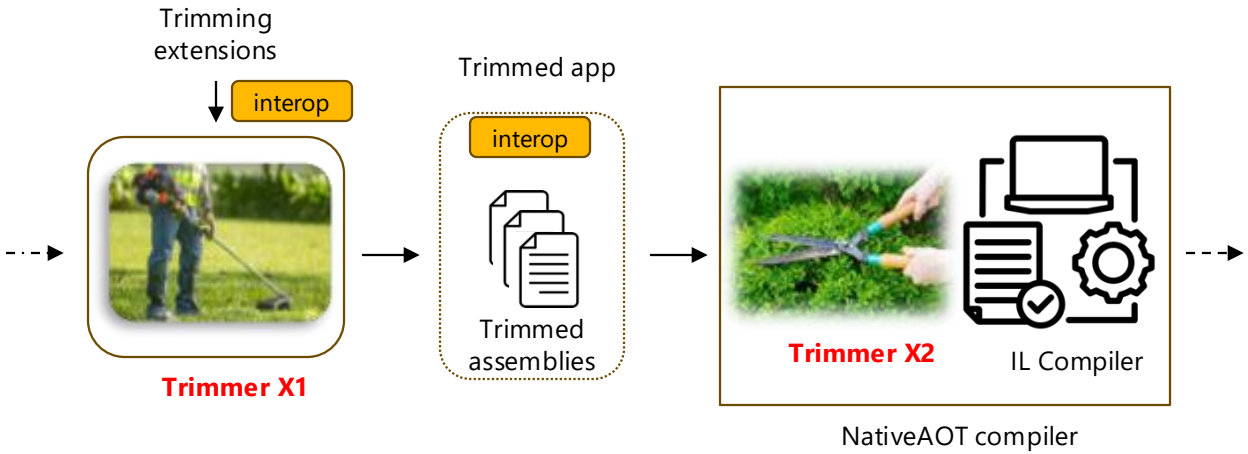
```
@implementation MyNSObject {
    ...
    int callback_6_Playground_MyNSObject_get_Answer (
      id self, SEL sel, GCHandle* excp_gchandle);

    -(int) answer
    {
        GCHandle exception_gchandle = INVALID_GCHANDLE;
        int rv = { 0 };
        rv = callback_6_Playground_MyNSObject_get_Answer (self, _cmd, &excp_gchandle);
        xamarin_process_managed_exception_gchandle (excp_gchandle);
        return rv;
    }
```

Improved Microsoft.iOS FX and SDK

- New interop:
  - Generates metadata tokens during build
  - Removed various use of reflection (startup ++)
  - Exposed UCO methods accessed directly (startup ++)
    - MacOS 3-6x faster, MacCatalyst 30-50% faster
  - Runtime-agnostic

- Not ideal solution
  - Running trimmer twice
  - Goal: trimmable interop

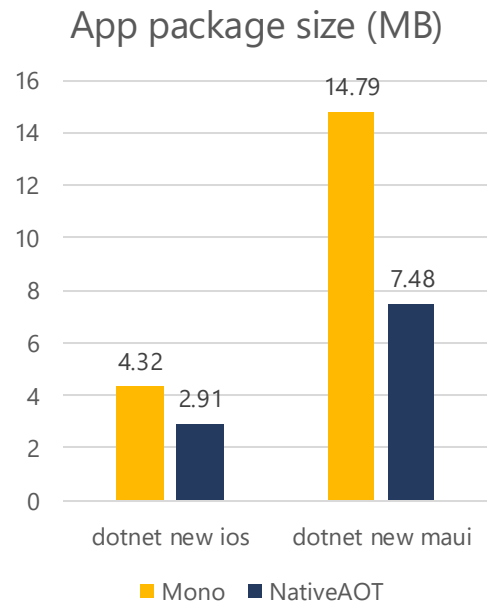
- Trimmable framework
  - **Mono** TrimMode=Full



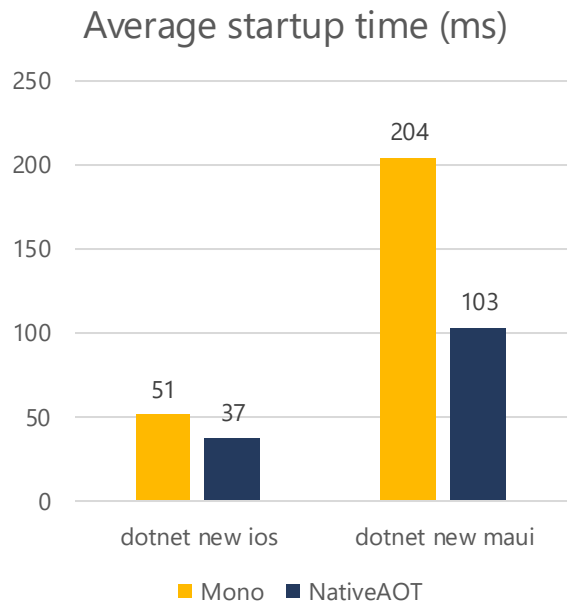
.NET 8 achievements

- Microsoft.iOS framework trim-compatible
- TrimMode=full
- MAUI iOS applications – **build warnings**

Up to **2x smaller** apps



Up to **2x faster** startup time



MAUI app size overview

Application .ipa size (MB)	Mono TrimMode=partial	NativeAOT TrimMode=partial	Diff	Runs?	NativeAOT TrimMode=full	Diff	Runs?
WeatherTwentyOne	16.6	11.9	-28.4%	✅	7.9	-52.3%	✅
Calculator	14.9	10.8	-27.4%	✅	7.4	-50.5%	✅
PointOfSale	24.3	18.49	-24.0%	✅	13.7	-43.7%	❌
SpaceXHistory	20.3	15.18	-25.2%	✅	11.1	-45.4%	❌
Podcasts	24.46	16.8	-31.1%	TBD	TBD	TBD	TBD

MAUI FX and NativeAOT

- MAUI framework compatibility
  - AOT/trim analyzers -> 0 warnings
  - Template app **69 trim/AOT warnings** in .NET 8
- Considerations
  - Solving AOT/trim-compatibility in our codebase
  - Detecting features that can't work with NativeAOT
  - Don't break existing apps
- Approaches
  - Trimmer annotations + refactor
  - Feature switches (disable unsupported features)
  - Introduce new APIs
- Problematic areas
  - Custom DI containers
  - XAML parsing at runtime
  - Data bindings
  - Implicit operators - type conversions

Fixing warnings in MAUI FX

- Code refactoring
  - Trimmer annotations
    - DynamicallyAccessedMember* attributes
    - Enables safe use of Reflection
  - Unsafe patterns
    - Misuse of MakeGenericType
    - Iterating over implemented interfaces
  - Custom DI containers (fixed)
    - Handler service
    - ImageSource service
- Incompatibilities:
  - XAML runtime parsing
    - Pre-compile all XAML
    - Remove LoadFromXaml
  - Data bindings
    - Use typed-compiled bindings
  - Others
    - Implicit operators
      - Implement TypeConverters
    - Navigation via QueryPropertyAttribute
      - Implement IQueryAttributable
    - HybridWebView control, …

Data bindings

- Classic bindings
  - Slow
  - No type safety
  - Resolution via reflection
  - Not trim-compatible

XAML

```
<StackLayout BindingContext="{StaticResource pageViewModel}" x:Name="stack">
    <Label Text="{Binding Customer.Name}" x:Name="customerNameLabel" />
</StackLayout>
```

C#

```
stack.BindingContext = (PageViewModel)pageViewModel;
customerNameLabel.SetBinding(Label.TextProperty, "Customer.Name");
```

- Typed bindings
  - No proper C# API
- New API – Source generators
  - Speed
  - Developer experience
  - Intellisense

XAML

```
<StackLayout BindingContext="{StaticResource pageViewModel}" x:Name="stack">
    <Label Text="{Binding Customer.Name}" x:DataType="local:PageViewModel"/>
</StackLayout>
```

C#

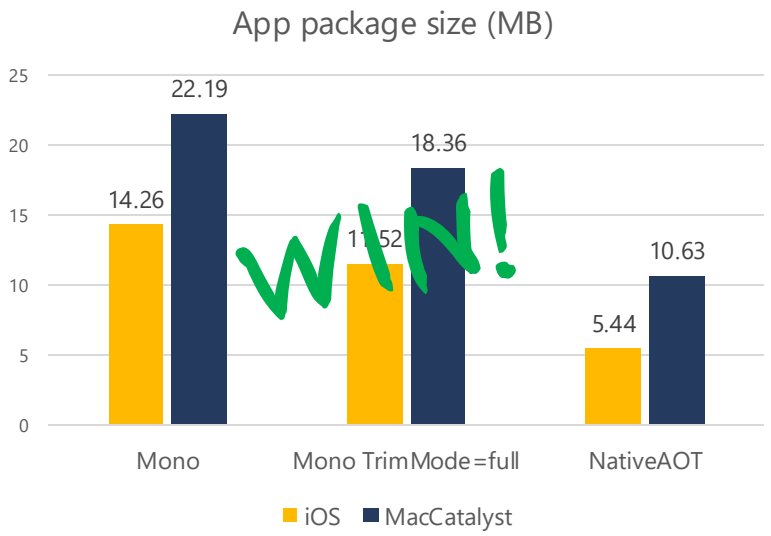
```
customerNameLabel.SetBinding(Label.TextProperty, (PageViewModel vm) => vm.Customer.Name);
// turns into ...

var binding = new TypedBinding<PageViewModel, string>(
    getter: vm => (vm.Customer.Name, true),
    setter: (vm, value) =>
    {
        if (vm?.Customer is not null)
        {
            vm.Customer.Name = value;
        }
    },
    handlers: new Tuple<Func<PageViewModel, object?>, string>[]
    {
        new(vm => vm, "Customer"),
        new(vm => vm.Customer, "Name"),
    });

customerNameLabel.SetBinding(Label.TextProperty, binding);
```

.NET 9 achievements

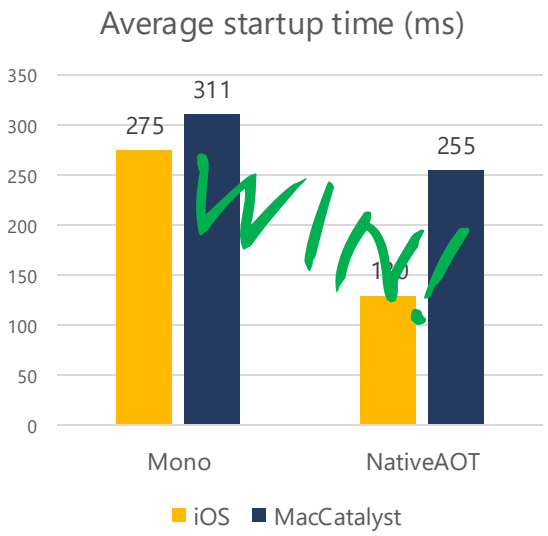
- MAUI framework trim-compatible for **iOS** and **MacCatalyst**
- TrimMode=Full – new default
- MAUI iOS app – **0 warnings**

Up to **3x smaller** apps

*Mono **-20% smaller**

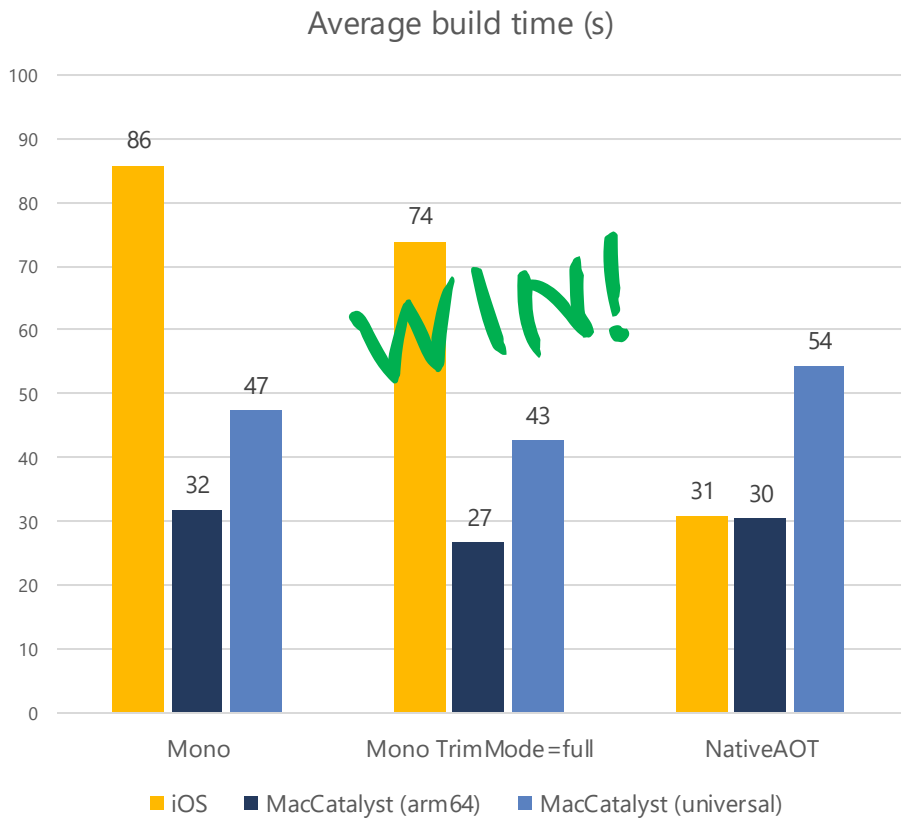


Up to **2x faster** startup time



Up to **3x faster** build time on **iOS**

Comparable build times on **MacCatalyst**



Key takeaways

- Great perfomance with MAUI apps! 

NativeAOT is an advanced feature

- Code adaptations likely required
- Not for every app
  - Limitations
  - AOT, trim-incompatible dependencies (NuGets) won't work
- No managed debugging (**Mono** ++)
- No dynamic code support (**Mono** ++)
- Limited runtime diagnostics (**Mono** ++)

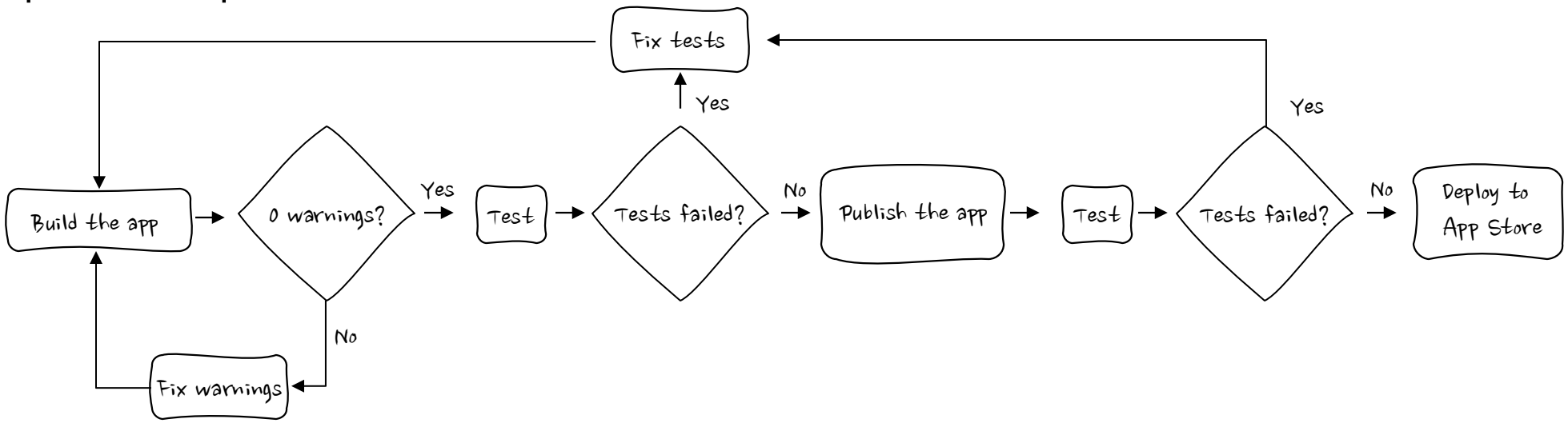
- AOT, trim-compatible apps (**Mono** ++)
- Source-generators to the rescue
  - System.Text.Json serializer
  - Typed bindings
  - Regular expressions, …

How to try it out?

- Enabling NativeAOT

```
<PropertyGroup>
  <!-- enable trimming and AOT analyzers on all platforms -->
  <IsAotCompatible>true</IsAotCompatible>

  <!-- select platforms to use with NativeAOT -->
  <PublishAot Condition="$([MSBuild]::GetTargetPlatformIdentifier('$(TargetFramework)')) == 'ios'">true</PublishAot>
  <PublishAot Condition="$([MSBuild]::GetTargetPlatformIdentifier('$(TargetFramework)')) == 'maccatalyst'">true</PublishAot>
</PropertyGroup>
```

- Development loop



Reach out https://aka.ms/dotnet-discord #apple

Thank you!

- References:
  - https://learn.microsoft.com/en-us/dotnet/maui/deployment/nativeaot?view=net-maui-9.0
  - https://learn.microsoft.com/en-us/dotnet/maui/deployment/trimming?view=net-maui-9.0
  - https://learn.microsoft.com/en-us/dotnet/core/deploying/native-aot/
  - https://learn.microsoft.com/en-us/dotnet/core/deploying/trimming/prepare-libraries-for-trimming