What is new in Mono? July 2013

miguel@gnome.org

Agenda

- Today:
  - What is new in Mono-land.
  - Major Work-in-Progress Projects

- Tomorrow:
  - Brainstorming session
  - Future of Mono and .NET

MONO 3.2

Mono 3.2

- Core .NET 4.5 support
  - Base Class Libraries
  - C# 5
- Available today:
  - Desktop
  - Server
  - Mobile
- 2.10.x is no longer supported

What is new in Mono 3.2

- Vastly improved SGen GC
  - Now the default!
- Micro-optimizations
  - Mobile-driven
- Consume PCL binaries
- Shipping OSS Microsoft frameworks
- New Mono frameworks

SGen Garbage Collector

- Our Precise, generational collector

- Now the default for Mono
  - “mono” now runs mono –gc=sgen
  - libmono is Boehm
  - libmono-sgen is Sgen

- Sgen is now faster on almost every workload
- MonoDevelop/Xamarin Studio
  - Been using it as a default since February

What is new?

- Major collectors:
  - Copy collector (old, reference code)
  - Mark and Sweep
- Minor collectors:
  - Now with support for cementing
- Help game developers love GC
- dtrace probes

Mark and Sweep

- Dynamic heap by default
  - Can specify fixed heap if desired
- Allocates blocks of memory to bucket sizes
  - Think of the spirit to “slab” allocation
  - Support changes in workloads with evacuation
- Concurrent Marking, Lazy Sweeping

Disabling Major Collector

New API on Mono.Runtime (mscorlib.dll)

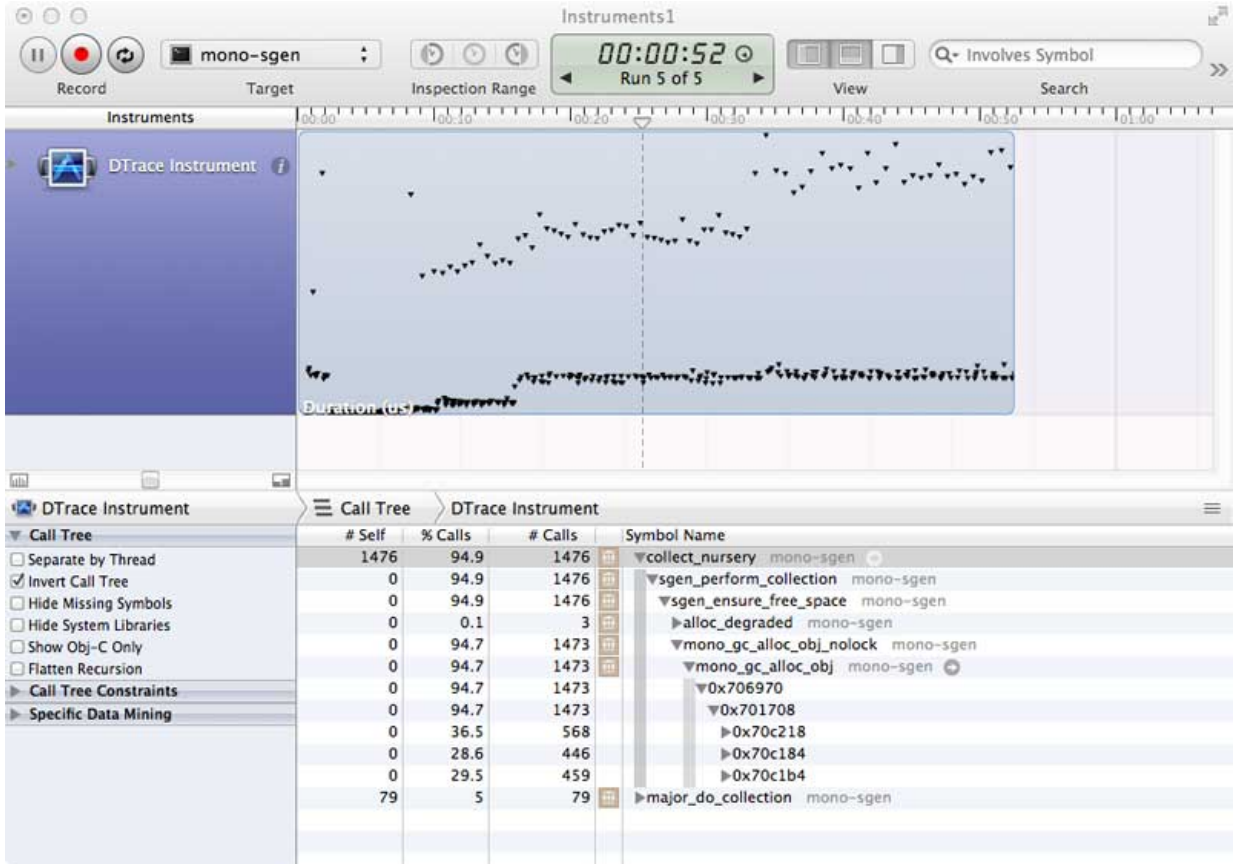
```
if (!Mono.Runtime.SetGCAllowSynchronousMajor(false))
    Console.WriteLine("Sorry, the GC won't cooperate.");

// your low-latency code here

Mono.Runtime.SetGCAllowSynchronousMajor(true);
```

- Exposed on Mobile.
- Use System.Reflection to find it on desktop editions.

dtrace probes on MacOS



Major collections and minor collections pause times visualized

http://schani.wordpress.com/2012/11/02/sgen-and-dtrace/

STATIC BUILD IMPROVEMENTS

Static Compilation

- For systems that don’t support JITing
  - Apple’s iOS devices
  - Consoles (PlayStation, Xbox)
  - Security: when not shipping IL

- Limited generics support
  - Static analysis limitations
  - Some dynamisms in .NET idioms
    - LINQ and Value Types

“Attempting to JIT compile method”

```
System.ExecutionEngineException: Attempting to JIT
compile method Foo[]:Bar`1
```

“Attempting to JIT compile method”

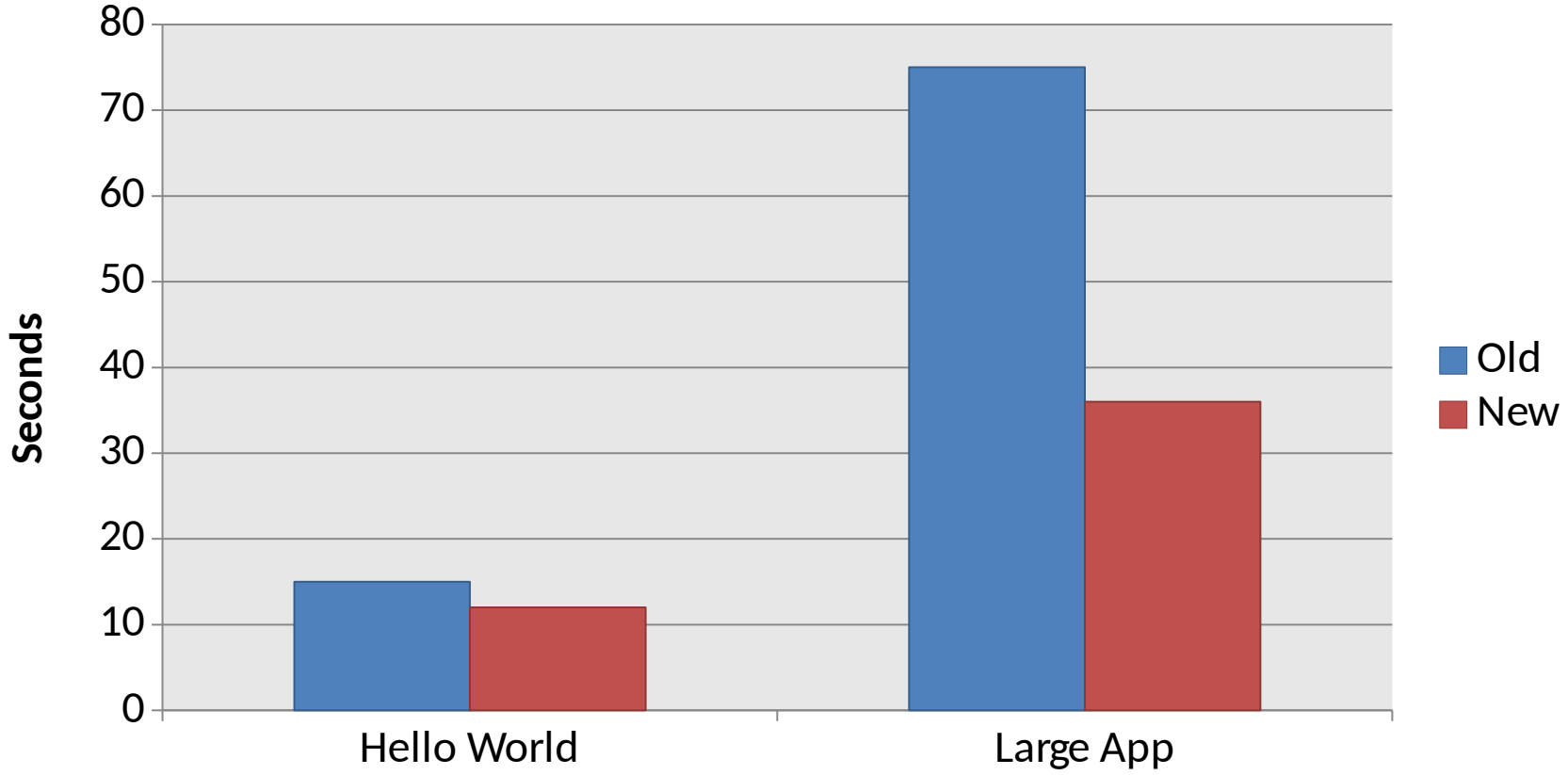
System.ExecutionEngineException: Attempting to JIT compile method Foo[]:Bar`1

FIXED!

iOS Development Improvements

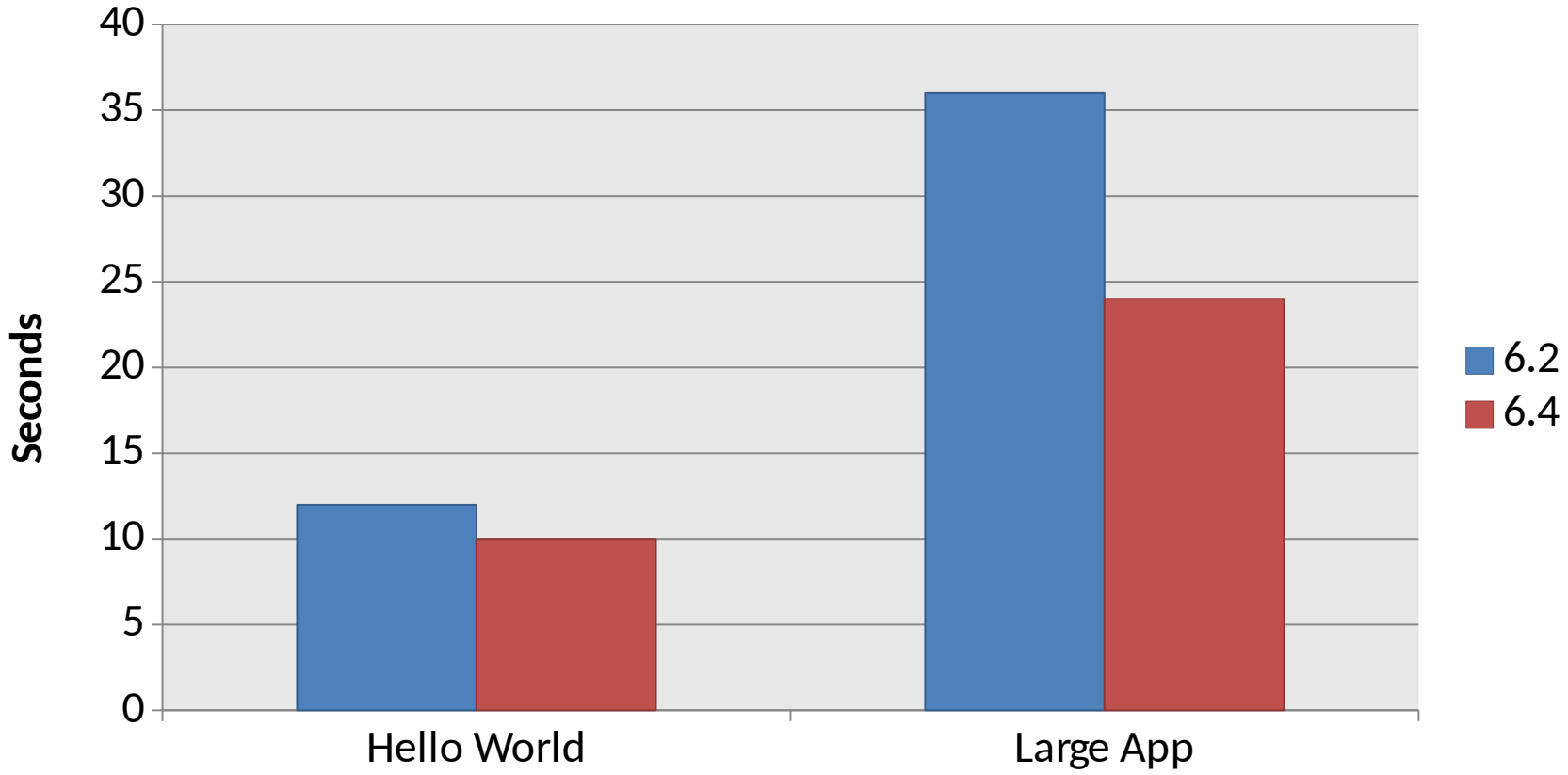
- Improve development cycle
- Incremental Builds
  - Disable linker (compile everything)
  - Cache resulting native code aggressively
- Incremental Upload
  - Only uploads changes
  - Main executable, libraries or resources

Incremental Deployment (With Build Caching)



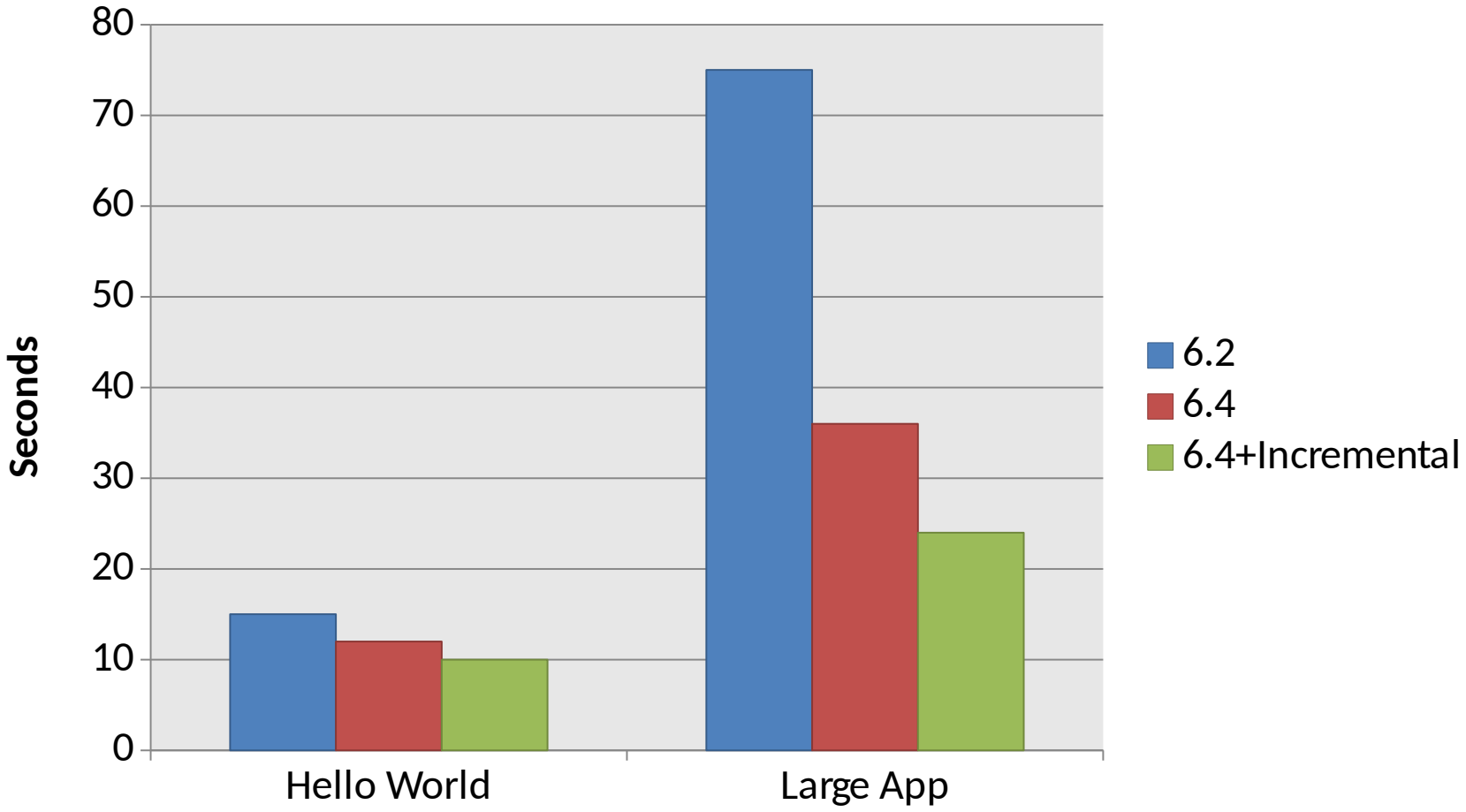
Deployment times can now be as short as one second.

Incremental Deployment (With Build Caching)



Deployment times can now be as short as one second.

Build Speed Improvements



MICROSOFT OPEN SOURCE CODE

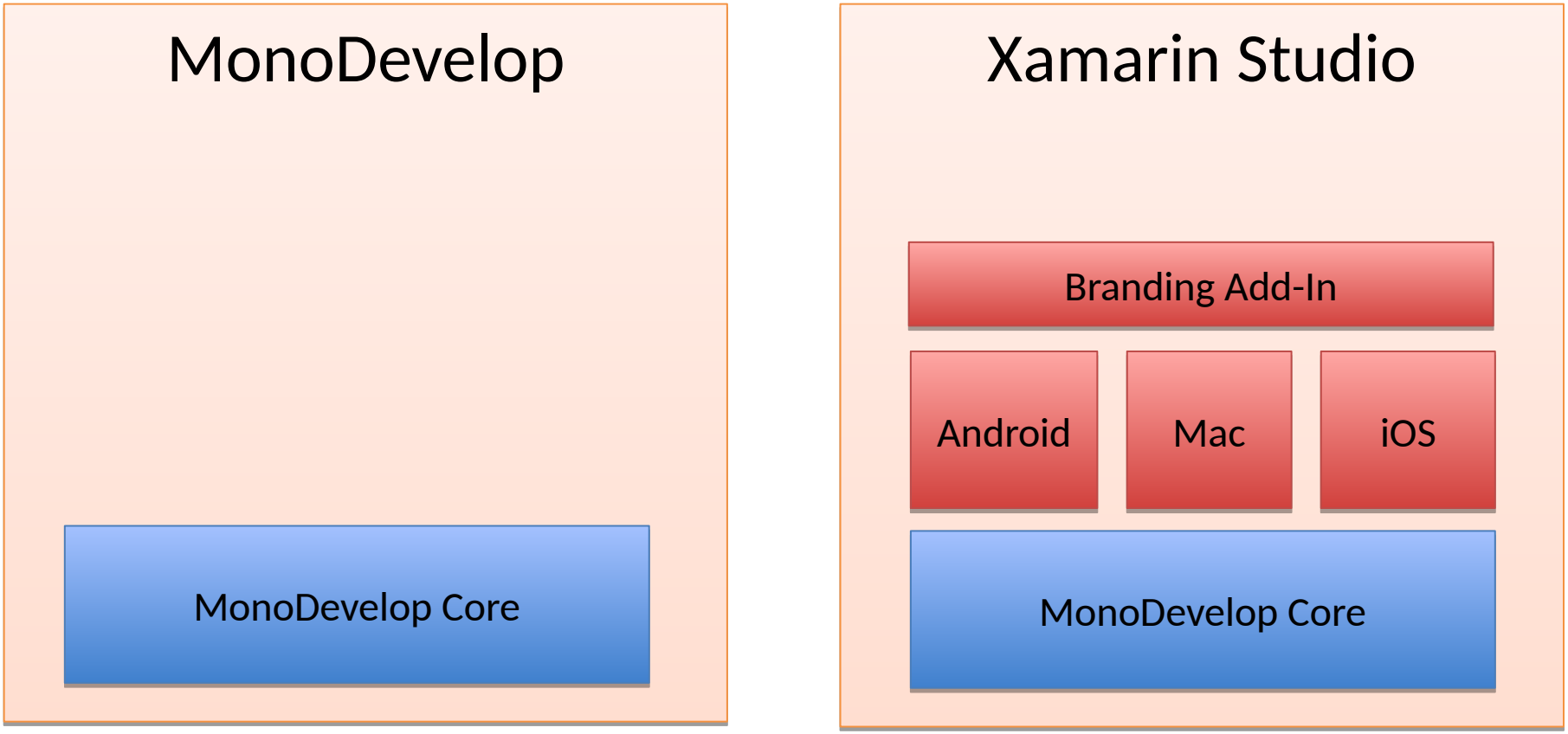
Now Bundled with Mono

- Reactive Extensions
  - Possible on iOS with new code gen changes

- F# everywhere

- Razor
- Entity Framework
- ASP.NET WebStack

MonoDevelop / Xamarin Studio



Open source

Commercial

CROSS PLATFORM FRAMEWORKS

Cocos2D XNA

- XNA port of the popular Cocos2D API
  - Over 4,000 games built with this API
  - Well documented, well known

- MonoGame brings it everywhere
  - iOS, Android, Windows Phone (7 and 8)
  - Mac
  - Windows (GL, DX, , Windows Store)
  - Xbox360, Ouya
  - Play Station Mobile (PS Vita + Sony Androids)

Using Cocos2D XNA Today

- Available as:
  - NuGet Packages
  - Templates for VS and Xamarin Studio

- Source code:
  - http://github.com/mono/cocos2d-xna

- Getting Started:
  - http://docs.xamarin.com/guides/cross-platform/cocos2d_xna

Angry Ninjas - Full Open Source Game

https://github.com/xamarin/AngryNinjas

Cocos2D Crash Course

- Mike Bluestein’s talk tomorrow

Xamarin.Mobile

- Base class library for mobile services
- Mike Bluestein's talk

Xamarin.Auth

- Clients for OAuth 1 and OAuth 2
  - Includes variations
- Stores user credentials
- Support non-standard auth schemes
- Cross Platform

Xamarin.Auth

```
using Xamarin.Auth;

var auth = new OAuth2Authenticator(
  clientId: "App ID from https://developers.facebook.com/apps",
  scope: "",
  authorizeUrl: new Uri("https://m.facebook.com/dialog/oauth/"),
  redirectUrl: new Uri("http://www.facebook.com/connect/login_success.html"));

auth.Completed += (sender, eventArgs) => {
  DismissViewController(true, null);
  if (eventArgs.IsAuthenticated) {
    //Use eventArgs.Account to do wonderful things
  }
}
PresentViewController(auth.GetUI(), true, null);
//1. Create the service
```

Xamarin.Social

- Posts statuses, links, images/media to social networks
  - Access social network APIs using authenticated requests.
  - Automatically and securely store user credentials using Xamarin.Auth.
  - Cross Platform

- Extensible, currently has support for:
  - App.net
  - Facebook
  - Flickr
  - Pinterest
  - Twitter

Using Xamarin.Social

```
var facebook = new FacebookService {
  ClientId = MyFacebook_AppId
  RedirectUrl = new System.Uri(MyRedirectUrl)
};

//2. Create an item to share
var item = new Item { Text = "Xamarin.Social is the bomb" };
item.Links.Add (new Uri("http://github.com/xamarin/xamarin.social"));

//3. Present the UI on iOS
var shareController = facebook.GetShareUI(item, result => {
  //result lets you know if the user shared the item or canceled
  DismissViewController(true, null);
});
PresentViewController(shareController, true, null);
```

Now all Open Source

- All frameworks:
  - Cocos2D XNA
    - http://github.com/mono/cocos2d-xna
  - Xamarin.Auth
    - http://github.com/xamarin/Xamarin.Auth
  - Xamarin.Mobile
    - http://github.com/xamarin/Xamarin.Mobile
  - Xamarin.Social
    - http://github.com/xamarin/Xamarin.Social

- Taking patches!

WORK IN PROGRESS

PLAYSCRIPT

PlayScript

- Started by Zynga
  - Xamarin working to integrate into Mono
  - Rescuing Flash Developers!

- ActionScript look-alike language
  - Superset of ActionScript
  - With C# 5 features
  - Optional strongly-typed
    - Encouraged for performance

PlayScript Compiler

- Dual Compiler
  - playsc foo.play
    - Compiles PlayScript to foo.exe
  - playsc bar.cs
    - Compiles C# to bar.exe
  - playsc foo.play bar.cs
    - Compiles PlayScript foo.play and C# bar.cs
    - Into foo.exe

- Allows blending C# and PlayScript code in one assembly

PlayScript Compiler

- Based on Mono's C# compiler
  - Altered to accept PlayScript language

- Side effects:
  - Compiler as a service support for PlayScript
  - Full Xamarin Studio integration
    - code completion
    - Docs
    - Project support

PlayScript Libraries

- pscorlib
  - Provides core library for PlayScript/ActionScript
- Flash Stage3D
  - GPU accelerated framework
  - Used by gaming frameworks

Ported Frameworks

- Starling Framework
- 2D Game Engine
- Powers Angry Birds

- Away3D
- 3D Game Engine
- Physics

PlayScript today

- Compiles existing large ActionScript projects
  - Some missing language features
  - Actively working to improve language
  - Merging into Mono (playscript branch)
  - Everywhere Mono runs

- On GitHub:
  - Project: http://github.com/playscript

C++ INTEROP

Evolving Cxxi

- Couple of years ago Cxxi project was launched
- Used GCC-XML to parse C++ definitions
- Runtime creation of C# to C++ bridges
  - Reflection.Emit
  - Pluggable VTable ABIs

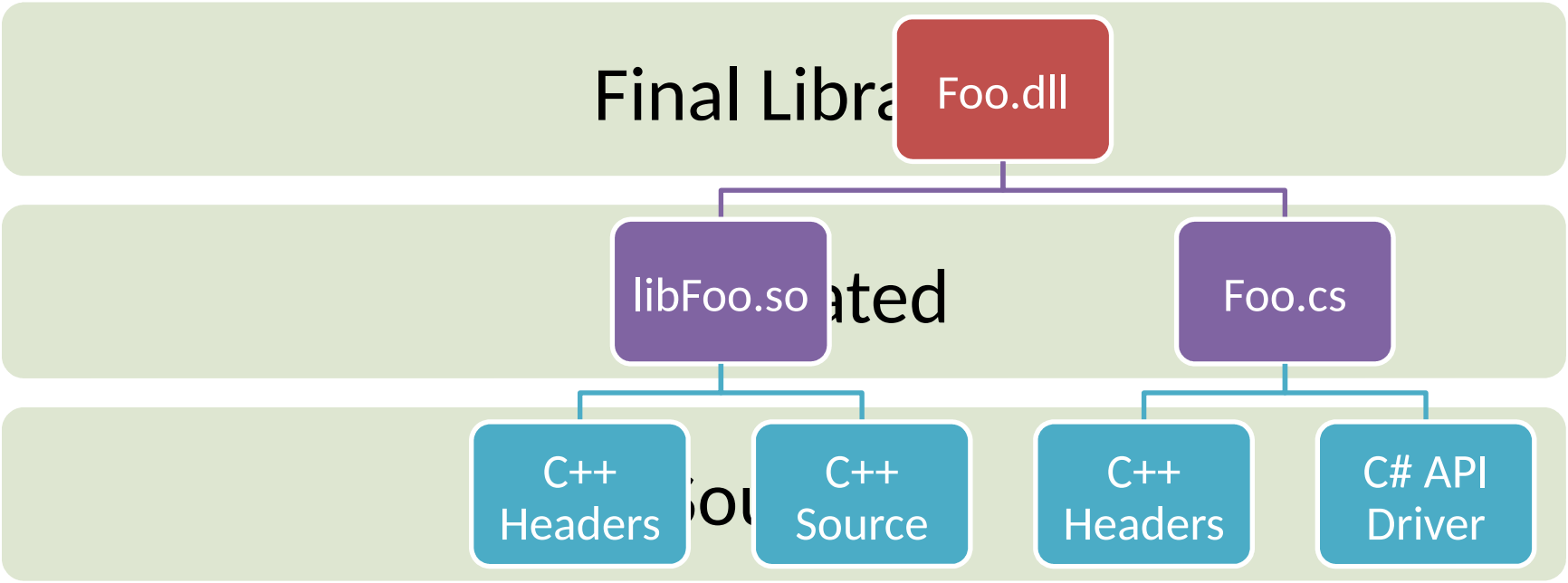
CppSharp

- New C++ binding technology from Mono

- Produces C# libraries to consume C++ code
  - Use Clang for higher fidelity
  - Static compilation (to support iOS, Consoles)
  - Customize binding for .NET consistency
  - Direct calls to C++ (no intermediate glue)
  - Imports C++ Doxygen docs into C# docs

http://github.com/mono/CppSharp

Generating a binding



```
public class Sample : ILibrary
{
  public Sample(LanguageGeneratorKind kind) : base("Sample", kind) {}

      public override void SetupPasses(Driver driver, PassBuilder passes)
  {
    passes.RenameDeclsUpperCase(RenameTargets.Any);
    passes.FunctionToInstanceMethod();
  }

  public override void Preprocess(Driver driver, Library lib)
  {
    lib.SetClassAsValueType("Foo");
    lib.SetNameOfFunction("FooAdd", "FooCalc");
    lib.IgnoreClassField("Foo", "b");
  }

  static class Program
  {
    public static void Main(string[] args)
    {
      ConsoleDriver.Run(new Sample(LanguageGeneratorKind.CPlusPlusCLI));
      ConsoleDriver.Run(new Sample(LanguageGeneratorKind.CSharp));
    }
  }
```

NATIVE INTEROP

Assembly in C#

- Allow developers to inline assembly code
  - Micro-optimizations
  - Optimize a code path
  - Use arch-specific features

- You can not get more native than that.

Assembly in C#

- Allow developers to inline assembly code
  - Micro-optimizations
  - Optimize a code path
  - Use arch-specific features

Example

```
unsafe intClear(IntPtr buffer, int count)
{
    asm.BindIntPtr(0, buffer);
    asm.BindInt(1, count);
    asm.Emitx86(@"
      push %ecx
     push %edi
     mov %eax,%edi
     mov %ebx,%ecx
     xor %eax,%eax
       rep stosb
       mov $1,%eax
     pop %edi
     pop %ecx
   ");
    return asm.FetchInt(0);
}
```

In case there are any doubts

```
unsafe int Clear(IntPtr buffer, int count)
{
    asm.BindIntPtr(0, buffer);
    asm.BindInt(1, count);
    asm.EmitX86(@"
      push %ecx
     push %edi
     mov %eax,%edi
     mov %ebx,%ecx
     xor %eax,%eax
       rep stosb
       mov $1,%eax
     pop %edi
     pop %ecx
   ");
    return asm.FetchInt(0);
}
```

Example

```
unsafe intClear(IntPtr buffer, int count)
{
        asm.BindIntPtr(0, buffer);
    asm.BindInt(1, count);
    asm.Emitx86(@"
                push %ecx
     push %edi
     mov %eax,%edi
     mov %ebx,%ecx
     xor %eax,%eax
            rep stosb
       mov $1,%eax
   ");
    return asm.FetchInt(0);
}
```

- Bind Parameters
- Let Mono know which parameters you will pass

Example

```
unsafe int Clear(IntPtr buffer, int count)
{
        asm.BindIntPtr(0, buffer);
    asm.BindInt(1, count);
    asm.EmitX86(@"
                push %ecx
     push %edi
     mov %eax, %edi
     mov %ebx, %ecx
     xor %eax, %eax
            rep stosb
       mov $1, %eax
   ");
    return asm.FetchInt(0);
}
```

- Bind Parameters
- Let Mono know which parameters you will pass

- Assembly code as a string
- Assembled by LLVM
- Mono JIT Calling conventions

Example

```
unsafe int Clear(IntPtr buffer, int count)
{
        asm.BindIntPtr(0, buffer);
    asm.BindInt(1, count);
    asm.EmitX86(@"
                push %ecx
     push %edi
     mov %eax,%edi
     mov %ebx,%ecx
     xor %eax,%eax
            rep stosb
       mov $1,%eax
   ");
    return asm.FetchInt(0);
}
```

- Bind Parameters
- Let Mono know which parameters you will pass

- Assembly code as a string
- Assembled by LLVM
- Mono JIT Calling conventions

- Fetch result value

Example

```
unsafe int Clear(IntPtr buffer, int count)
{
        asm.BindIntPtr(0, buffer);
    asm.BindInt(1, count);
    asm.EmitX86(@"
                push %ecx
     push %edi
     mov %eax,%edi
     mov %ebx,%ecx
     xor %eax,%eax
            rep stosb
       mov $1,%eax
   ");
    return asm.FetchInt(0);
}
```

- Bind Parameters
- Let Mono know which parameters you will pass

- Assembly code as a string
- Assembled by LLVM
- Mono JIT Calling conventions

- Fetch result value

Passing parameters, extracting

- Input parameters:
  - asm.BindXXX (int incomingParameter, XXX value);
- Emit assembly code:
  - asm.Emit (string code)
  - Assembled with LLVM assembler
  - Injected into resulting assembly
  - Follow Mono native rules for preserving register usage
- Output parameters:
  - asm.FetchXXX (int resultParamter)

How it works

- Platform.Native.dll
  - Contains stub methods
  - Allows same source to build on .NET and Mono

- Follow our previous efforts:
  - Like SIMD
  - On Mono, the runtime recognizes some calls

EmitXxx ignored for other platforms

```
//Highly optimized native way of getting a native one!
unsafe int GetOneConstant()
{
    asm.EmitX86("mov $1,%eax");
    asm.EmitArm("mov r0,#1");
    asm.EmitX64("mov $1,%rax");
    return asm.FetchInt(0);
}
```

For more complex code, use asm.Arch:

```
unsafe int GetOneConstant()
{
    if (asm.Arch == Arch.x86) {
            asm.EmitX86("mov $1,%eax");
     return asm.FetchInt(0);
   } else
     return 1;
}
```

WHAT IS NEXT?

Future for Mono

- Community feedback on features needed
- Brainstorming session tomorrow!
  - Where should Mono and .NET go?
  - Bring your favorite feature request