Mono And .NET

→ Miguel de Icaza
VP Developer Platform
Novell, Inc.
Agenda

- Introduction to Mono
- Mono’s Customizable CLR
- Mono’s C# Eval
- Assembly binary reshaping
- Turbo charging games and graphics
- Static Compilation
- Others
Mono 2.0

Just released!

- An open source .NET implementation:
  - A subset of .NET
  - Sponsored by Novell
  - ~120 non-affiliated contributors (1.2 -> 2.0)
- Direction driven by contributors
Compatibility

- Our goal is to have a compatible runtime to the CLR
  - ECMA specifications make it possible
  - Develop, build, debug on Visual Studio or Unix
  - Deploy on Linux, Mac OSX and embedded
APIs

Server
- ASP.NET
- Apache and FastCGI
- System.Data SQL Server

Client
- Gtk#
- Gdk#
- Cocoa#
- Windows.Forms
- Mono.Cairo
- Pango#

Third Party
- Postgres, MySQL
- SQLite, Oracle, Sybase
- Tao.Framework
- C5
- NDesk.DBus

Infrastructure
- Mono.Cecil
- Mono.ZeroConf
- Mono.Nat
- Mono.Addins
- Novell.Ldap
- Java/IKVM
- Mono.RelaxNG
- Mono.Fuse
- Mono.Torrent
- Mono.Nat
- Gecko# (Mozilla)
- Mono.Upnp
Mono's CLI Implementation

- We can offer a few bonuses
  - Take .NET where no .NET has gone before
  - Offering new forward-compatible features
  - Support special scenarios
Mono's CLI Implementation

- We can offer a few bonuses
  - Take .NET where no .NET has gone before
  - Offering new forward-compatible features
  - Support special scenarios
CLR for Everyone

- Microsoft has the CLR, CF and the CoreCLR
  - CoreCLR is a small version of CLR
  - CoreCLR used in Mesh and Silverlight
  - Compact Framework used in XNA
CLRs For Everyone

- Microsoft has the CLR, CF and the CoreCLR
  - CoreCLR is a small version of CLR
  - CoreCLR used in Mesh and Silverlight
  - Compact Framework used in XNA

- For everyone else, there is Mono
Mono Adaptability
From full framework to tailored framework

- Full framework is 100 megs (uncompressed)
- Minimal setup is 2 megs (uncompressed)
- Modular runtime can be shrunk/grown:
The Evolution of a Compiler
(or, C# 5 today)
Mono's C# 3.0 Compiler

- C# compiler written in C#
  - Originally, a project to learn to write C# code
Mono's C# 3.0 Compiler

- C# compiler written in C#
  - Originally, a project to learn to write C# code
- First bootstrap (2001)
  - 17 seconds to bootstrap, 10,000 lines
  - csc compiled it in a second
Mono's C# 3.0 Compiler

- C# compiler written in C#
  - Originally, a project to learn to write C# code

- First bootstrap (2001)
  - 17 seconds to bootstrap 10,000 lines
  - csc compiled it in a second

- Speed is no longer a problem
  - Today 82,000 lines in 2.2 seconds
  - 1.6x slower than csc
using System;
using Mono.CSharp;

class MyFirstCSharpInterpreter {
    static void Main (string [] args) {
        object r = Evaluator.Evaluate (args [0]);
        Console.WriteLine (r);
    }
}
Mono.CSharp – Applications

- Read-Eval-Print-Loop (repl)
- Script applications with C#
- Rapid prototyping in target language
- Automation

- Would be cool to have this on every app!
The csharp Command

- Python and Ruby have interactive shells
- Read-Eval-Print Loop
- Expressions and Statements:

```csharp
> 1;
1;
> "Hello, World".IndexOf (",");
5;
> 1 +
> > 2;
3
> var a = Console.ReadLine ();
```
LINQ From The Command Line

$ csharp
Mono C# Shell, type “help;” for help

Enter statements below.
csharp> using System.IO;
csharp> var last_week = DateTime.Now - TimeSpan.FromDays (7);
csharp> from f in Directory.GetFiles (“/etc”)
    >   let fi = new FileInfo (f)
    >   where fi.LastWriteTime < last_week
    >   select f;
{ “/etc/adjtime”, “/etc/asound.state”,
    “/etc/ld.so.cache”, “/etc/mtab”,
    “/etc/printcap”, “/etc/resolv.conf” }
csharp>
csharp> LoadLibrary ("System.Xml.Linq");
csharp> using System.Xml.Linq;
csharp> var xml = new XElement("CompilerSources",
    > from f in Directory.GetFiles ("/cvs/mcs/mcs")
    > let fi = new FileInfo (f)
    > orderby fi.Length
    > select new XElement ("file",
        > new XAttribute ("name", f),
        > new XAttribute ("size", fi.Length)));

csharp> xml;
<CompilerSources>
    <file name="/cvs/mcs/mcs/mcs.exe.config" size="395" />
    <file name="/cvs/mcs/mcs/gmcs.exe.config" size="464" />
    <file name="/cvs/mcs/mcs/OPTIMIZE" size="498" />
    <file name="/cvs/mcs/mcs/lambda.todo" size="658" />
[...]
</CompilerSources>
GUI Shell
C# eval hosted in a GUI

• Replace base class, with GUI base class:
GUI Shell
C# eval hosted in a GUI

- Replace base class, with GUI base class:

```
Mono C# Shell, type 'help;' for help

Enter statements or expressions below.

```csharp```
var logo = Image.FromFile("/tmp/moon.jpg");
logo.GetType();
System.Drawing.Bitmap
logo;
```csharp```
GUI Shell, Quick Plot Method

- Plot (Func<double,double>);
GUI Shell, Quick Plot Method

- Plot (Func<double,double>);
Reshaping The API
Turning the compiler into a library

Complete C# Compiler

Everything public

Mono.CSharp

Minimal API
Reshaping The API
Avoid manual work, reusing Mono.Cecil and Mono.Linker

Complete C# Compiler
- Everything public
  - gmc.exe
    - References
      - Mono.CompilerServices.SymbolWriter
      - Mono.CSharp
        - AbstractPropertyEventMethod
        - Accessor
        - Accessors
        - AddressOp
        - AnonymousExpression
        - AnonymousMethodBody
        - AnonymousMethodExpression
        - AnonymousMethodStatement
        - AnonymousTypeClass
        - AnonymousTypeDeclaration
        - AnonymousTypeParameter
        - ArrayParametersCollection
        - Argument
        - ArgumentAccess
        - ArgumentParameter
        - Argument
        - ArrayAccess
        - ArrayCreation
        - ArrayIndexCast
        - ArrayInit
        - Array
        - As
        - AssemblyClass

Mono Linker
- Uses Mono.Cecil
- Reshapes code
- Desired API
- Removes or hides

link.xml

Mono.CSharp
- Minimal API
  - Mono.CSharp.dll
    - References
      - Mono.CSharp
        - CompiledMethod
        - Evaluator
        - InteractiveBase
        - Report
Mono Linker Use Cases

- Shrinking Assemblies
  - Shipping only what is required
  - Simplify deployment

- Create your own Compact Framework
  - What you need from the superset

- .NET 3.5 to Silverlight
  - We reshape our assemblies.
  - Minimal hand-editing/tuning.
Beyond the CLR: Innovating on a Solid Foundation

Virtual machines are fascinating

- Great innovations are possible
  - Build on an existing large ecosystem
  - Instrument, expand, innovate
  - Special code generation
Beyond the CLR: Innovating on a Solid Foundation

Virtual machines are fascinating

- Great innovations are possible
  - Build on an existing large ecosystem
  - Instrument, expand, innovate
  - Special code generation

- VM potential limited by vendor realities
  - Provider scarcity
  - Shipping dates
  - Staffing
  - Product Management
  - Feature prioritization
Injecting Code Into A Live Process
The Mono.Attach.VirtualMachine API

- On the root AppDomain, on a new thread
Consoles for everyone!
Turbocharging Games

Fast, Productive, Safe.
Pick all three.
Game Software Components

- **Display**
  - Rendering
  - Shading
  - Scene
  - Animation
  - Geometry
  - GUI

- **Simulation**
  - Physics
  - Collision
  - Particles
  - Terrain

- **Game Logic**
  - World rules
  - Enemy AI
  - User control
  - Camera
  - Behavior

- **Support**
  - Audio
  - Input
  - Networking
The Problem
Games are real-time programs

- 30 to 60 frames per second (0.016 seconds)

Input
- User control
- Network events

AI
- Scripted, slow
- React to change
- Update scene

Updates
- Render Graphics
- Play audio
Problem: Scripting Is A Bottleneck
Gaming's Achilles' Heel

Display
- Rendering
- Shading
- Scene
- Animation
- Geometry
- GUI

Simulation
- Physics
- Collision
- Particles
- Terrain

Game Logic
- World rules
- Enemy AI
- User control
- Camera
- Behavior

Support
- Audio
- Input
- Networking
Language Choices

- **Productivity**
  - Easy
  - Difficult

- **Performance**
  - Slow
  - Fast

- **Languages**
  - Fully Dynamic
  - C#/Java
  - C/C++
  - Assembly Language
Mono in Gaming Today
Moving from scripting to static/compiled

- Mono’s CLR is ideal for embedding
Mono in Gaming Today
Moving from scripting to static/compiled

- Mono’s CLR is ideal for embedding

- Some examples
  - SecondLife: Switched from LSL to Mono
    - 50x to 300x performance increase
Mono in Gaming Today
Moving from scripting to static/compiled

- Mono’s CLR is ideal for embedding

- Some examples
  - SecondLife: Switched from LSL to Mono
    - 50x to 300x performance increase
  - Unity3D: Powers Cartoon Network’s FusionFall
    - Uses C#, UnityScript and Boo
    - UnityScript is a strongly typed Javascript
Demo
Managed Code In Gaming

Improving developer productivity while maintaining program speed

- Game AI
  - Traditional
    - Scripted
      - Slow/easy
  - Compiled
    - Fast/Hard

- Game Engines
  - Compiled
    - Fast/Hard

- Graphics Engine
  - Compiled
    - Fast/Hard
Managed Code In Gaming
Improving developer productivity while maintaining program speed

Game AI
- Traditional: Scripted (Slow/easy)
- Improved: Managed (Fast/Easy)

Game Engines
- Traditional: Compiled (Fast/Hard)
- Improved: Compiled (Fast/Hard)

Graphics Engine
- Traditional: Compiled (Fast/Hard)
- Improved: Compiled (Fast/Hard)
Managed Code In Gaming
Improving developer productivity while maintaining program speed

Game AI
- Traditional: Scripted, Slow/easy
- Improved: Managed, Fast/Easy
- Future: Managed, Fast/Easy

Game Engines
- Traditional: Compiled, Fast/Hard
- Improved: Compiled, Fast/Hard
- Future: Managed, Fast/Easy

Graphics Engine
- Traditional: Compiled, Fast/Hard
- Improved: Compiled, Fast/Hard
- Future: Compiled, Fast/Hard
3D Floating Point Vector Operations
At the core of gaming engines

- Exploring an innocent looking loop in C#:

```csharp
void UpdatePos(Vector3f[] points, ref Vector3f delta)
{
    for (int i = 0; i < points.Length; i++)
        points[i] += delta;
}
```

![Mathematical diagram showing vector addition](image)

```csharp
Vector3f static operator + (Vector3f a, Vector3f b)
{
    return new Vector3f (a.x+b.x, a.y+b.y, a.z+b.z);
}
```
The code that does the addition

```
.method private static hidebysig
    {
        // Method begins at RVA 0x2144
        // Code size 50 (0x32)
        .maxstack 4
        .locals init (int32 V_0)
        IL_0000:  ldc.i4.0
        IL_0001:  stloc.0
        IL_0002:  br IL_0028
        IL_0007:  ldarg.0
        IL_0008:  ldloc.0
        IL_000e:  dup
        IL_0014:  ldarg.1
        IL_0024:  ldloc.0
        IL_0025:  ldc.i4.1
        IL_0026:  add
        IL_0027:  stloc.0
        IL_0028:  ldloc.0
        IL_0029:  ldarg.0
        IL_002a:  ldlen
        IL_002b:  conv.i4
        IL_002c:  blt IL_0007
        IL_0031:  ret
    } // end of method X::UpdatePos
```
The IL implementation

```csharp
// method line 24
.method public static hidebysig specialname
default valuetype Mono.Simd.Vector4f op_Addition (valuetype Mono.Simd.Vector4f v1, valuetype Mono.Simd.Vector4f v2) cil managed
{
    // Method begins at RVA 0x24ac
    // Code size 69 (0x45)
    .maxstack 7
    .locals init (valuetype Mono.Simd.Vector4f V_0)
    IL_0000: ldloca.s 0
    IL_0002: ldarga.s 0
    IL_0004: ldfld float32 Mono.Simd.Vector4f::x
    IL_0009: ldarga.s 1
    IL_000b: ldfld float32 Mono.Simd.Vector4f::x
    IL_0010: add
    IL_0011: ldarga.s 0
    IL_0013: ldfld float32 Mono.Simd.Vector4f::y
    IL_0018: ldarga.s 1
    IL_001a: ldfld float32 Mono.Simd.Vector4f::y
    IL_001f: add
    IL_0020: ldarga.s 0
    IL_0022: ldfld float32 Mono.Simd.Vector4f::z
    IL_0027: ldarga.s 1
    IL_0029: ldfld float32 Mono.Simd.Vector4f::z
    IL_002e: add
    IL_002f: ldarga.s 0
    IL_0031: ldfld float32 Mono.Simd.Vector4f::w
    IL_0036: ldarga.s 1
    IL_0038: ldfld float32 Mono.Simd.Vector4f::w
    IL_003d: add
    IL_003e: call instance void valuetype Mono.Simd.Vector4f::.ctor(float32, float32, float32, float32)
    IL_0043: ldloc.0
    IL_0044: ret
} // end of method Vector4f::op_Addition
```
UpdatePos in x86 code

Generated assembly code

00000000 <X_UpdatePos>

<table>
<thead>
<tr>
<th>Line</th>
<th>opcode</th>
<th>Machine Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>69:</td>
<td>89 0c 24</td>
<td>mov %ecx,(%esp)</td>
<td></td>
</tr>
<tr>
<td>6c:</td>
<td>8b 4d d8</td>
<td>mov -0x28(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>6f:</td>
<td>89 4c 24 04</td>
<td>mov %ecx,0x4(%esp)</td>
<td></td>
</tr>
<tr>
<td>73:</td>
<td>8b 4d dc</td>
<td>mov -0x24(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>76:</td>
<td>89 4c 24 08</td>
<td>mov %ecx,0x8(%esp)</td>
<td></td>
</tr>
<tr>
<td>7a:</td>
<td>8b 4d e0</td>
<td>mov -0x20(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>7d:</td>
<td>8b 4c 24 0c</td>
<td>mov %ecx,0xc(%esp)</td>
<td></td>
</tr>
<tr>
<td>81:</td>
<td>83 ec 10</td>
<td>sub $0x10,%esp</td>
<td></td>
</tr>
<tr>
<td>84:</td>
<td>8b 4d c4</td>
<td>mov -0x3c(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>87:</td>
<td>8b 4c 24</td>
<td>mov %ecx,(%esp)</td>
<td></td>
</tr>
<tr>
<td>8a:</td>
<td>8b 4d c8</td>
<td>mov -0x38(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>91:</td>
<td>89 4c 24 08</td>
<td>mov %ecx,0x8(%esp)</td>
<td></td>
</tr>
<tr>
<td>94:</td>
<td>89 4c 24 08</td>
<td>mov %ecx,0x8(%esp)</td>
<td></td>
</tr>
<tr>
<td>98:</td>
<td>8b 4d d0</td>
<td>mov -0x30(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>9b:</td>
<td>8b 4c 24 0c</td>
<td>mov %ecx,0xc(%esp)</td>
<td></td>
</tr>
<tr>
<td>9f:</td>
<td>50</td>
<td>push %eax</td>
<td></td>
</tr>
<tr>
<td>a0:</td>
<td>e8 43 00 00 00</td>
<td>call op_Addition</td>
<td></td>
</tr>
<tr>
<td>a5:</td>
<td>83 c4 20</td>
<td>add $0x20,%esp</td>
<td></td>
</tr>
<tr>
<td>a8:</td>
<td>8b 45 bc</td>
<td>mov %eax,-0x44(%ebp)</td>
<td></td>
</tr>
<tr>
<td>ab:</td>
<td>8b 4d e4</td>
<td>mov -0x1c(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>ae:</td>
<td>89 08</td>
<td>mov %ecx,(%eax)</td>
<td></td>
</tr>
<tr>
<td>b0:</td>
<td>8b 4d e8</td>
<td>mov -0x18(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>b3:</td>
<td>89 48 04</td>
<td>mov %ecx,0x4(%eax)</td>
<td></td>
</tr>
<tr>
<td>b6:</td>
<td>8b 4d ec</td>
<td>mov -0x14(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>b9:</td>
<td>89 48 08</td>
<td>mov %ecx,0x8(%eax)</td>
<td></td>
</tr>
<tr>
<td>bc:</td>
<td>8b 4d f0</td>
<td>mov -0x10(%ebp),%ecx</td>
<td></td>
</tr>
<tr>
<td>bf:</td>
<td>89 48 0c</td>
<td>mov %ecx,0xc(%eax)</td>
<td></td>
</tr>
<tr>
<td>c2:</td>
<td>43</td>
<td>inc %ebx</td>
<td></td>
</tr>
<tr>
<td>c3:</td>
<td>8b 46 0c</td>
<td>mov %edx,0x0(%edi)</td>
<td></td>
</tr>
<tr>
<td>c6:</td>
<td>3b d8</td>
<td>cmp %eax,%ebx</td>
<td></td>
</tr>
<tr>
<td>c8:</td>
<td>8f 8c 4a ff ff ff</td>
<td>jle 18 &lt;X_UpdatePos+0x18&gt;</td>
<td></td>
</tr>
<tr>
<td>ce:</td>
<td>8d 65 f4</td>
<td>lea -0xc(%ebp),%esp</td>
<td></td>
</tr>
<tr>
<td>d1:</td>
<td>5e</td>
<td>pop %esi</td>
<td></td>
</tr>
<tr>
<td>d2:</td>
<td>5f</td>
<td>pop %edi</td>
<td></td>
</tr>
<tr>
<td>d3:</td>
<td>5b</td>
<td>pop %ebx</td>
<td></td>
</tr>
<tr>
<td>d4:</td>
<td>c9</td>
<td>leave</td>
<td></td>
</tr>
<tr>
<td>d5:</td>
<td>c3</td>
<td>ret</td>
<td></td>
</tr>
<tr>
<td>d6:</td>
<td>8b 4d d4</td>
<td>mov -0x2c(%ebp),%ecx</td>
<td></td>
</tr>
</tbody>
</table>
Mono.SIMD: Mapping To Native Instructions
SIMD aware runtime

- Object-oriented APIs for Vector processing
  - Vector4f, Vector4i, Vector2d, Vector16b, etc
  - Mapped to hardware operations

C#
- pos += delta

IL

x86
- movups (%eax),%xmm0
- movups (%edi),%xmm1
- addps %xmm1,%xmm0
- movups %xmm0,(%eax)

Detect SIMD use
UpdatePos With Mono's SIMD

00000000 <X_UpdatePos>:

0: 55                      push %ebp
1: 8b ec                   mov %esp,%ebp
3: 53                      push %ebx
4: 57                      push %edi
5: 56                      push %esi
6: 83 ec 04                sub $0x4,%esp
9: 8b 75 08                mov 0x8(%ebp),%esi
c: 8b 7d 0c                mov 0xc(%ebp),%edi
f: 33 db                   xor %ebx,%ebx
11: eb 29                   jmp 3c <X_UpdatePos+0x3c>
13: 8d 64 24 00            lea 0x0(%esp),%esp
17: 90                      nop
18: 39 5e 0c                cmp %ebx,0xc(%esi)
1b: 0f 86 2a 00 00 00       jbe 4b <X_UpdatePos+0x4b>
21: 8b cb                   mov %ebx,%ecx
23: c1 e1 04                shl $0x4,%ecx
26: 8b c6                   mov %esi,%eax
28: 03 c1                   add %ecx,%eax
2a: 05 10 00 00 00          add $0x10,%eax
2f: 0f 10 00 00 movups (%eax),%xmm0
32: 0f 10 0f 0f movups (%edi),%xmm1
35: 0f 58 c1                addps %xmm1,%xmm0
38: 0f 11 00 movups %xmm0,(%eax)
3b: 43                      inc %ebx
3c: 8b 46 0c                mov 0xc(%esi),%eax
3f: 3b d8                   cmp %eax,%ebx
41: 7c d5                   jl 18 <X_UpdatePos+0x18>
43: 8d 65 f4                lea -0xc(%ebp),%esp
46: 5e                      pop %esi
47: 5f                      pop %edi
48: 5b                      pop %ebx
49: c9                      leave
4a: c3                      ret
SIMD Operations Mix
Developer created tests

Matrix Mix

- Mono SIMD (aligned): 5.9 seconds
- Java -server: 18 seconds
- Regular FP: 19.5 seconds
Physics simulations, no optimizations

Mono.SIMD: Speedups

Based on the C++ simulation code at sharp-gamedev.blogspot.com/2008/09/updated-c-version.html
Ahead Of Time Compilation
Batch compilation of CIL/.NET code

- Ahead of Time compilation (AOT):
  - “ngen” in the .NET world
  - Precompiled IL code to native code

- Visible effects
  - Saves on startup time
  - Decreases footprint across multiple processes
  - Produces slower code

- Not complete
  - Can handle most of the JIT generated code
  - A few bits are not AOTed
Full Ahead Of Time Compilation
Entirely static compilation of CIL/.NET code

- Some devices disable on the fly codegen:
  - iPhone OS 2.x, XBox360
- Full AOT: Does AOT for the missing bits
Demo – Mono on iPhone.
Other Topics

Much more

- Mono Continuations.
  - Like Stackless-Python
  - Cooperative multi-threading
  - Avoids concurrency bugs
  - Concurrency achieved with processes

- Supercomputing Mono
  - 64 bit arrays
Learning More About Mono

- [http://www.mono-project.com](http://www.mono-project.com)
- Getting Started
  - [http://www.mono-project.com/Start](http://www.mono-project.com/Start)
- Community blogs
  - [http://www.go-mono.com/monologue](http://www.go-mono.com/monologue)
- Miguel’s blog
  - [http://tirania.org/blog](http://tirania.org/blog)