Поговорим о различных версиях .NET

Андрей Акиньшин, Энтерра

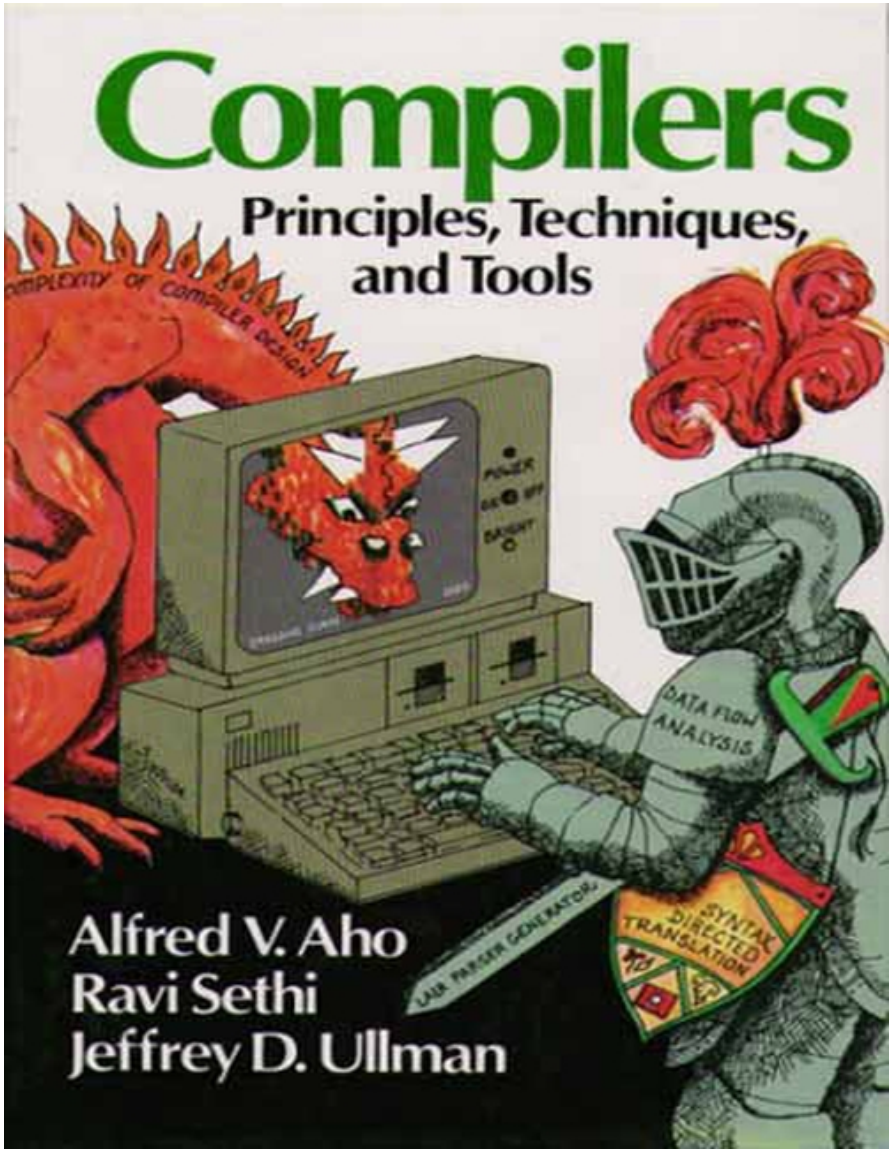
.NEXT 2014 Moscow

О чём будем разговаривать?

О разных версиях MS.NET и Mono, а именно:

1. Compilers
2. BCL
3. GC
4. JIT
5. CLR Internal

Поговорим о компиляторах



Свёртка констант

```
Console.WriteLine(checked(int.MinValue * -1));
Console.WriteLine(checked(int.MinValue / -1));
Console.WriteLine(unchecked(int.MinValue * -1));
Console.WriteLine(unchecked(int.MinValue / -1));
```

Свёртка констант

```
Console.WriteLine(checked(int.MinValue * -1));
Console.WriteLine(checked(int.MinValue / -1));
Console.WriteLine(unchecked(int.MinValue * -1));
Console.WriteLine(unchecked(int.MinValue / -1));
```

		MS.NET	Mono
checked	*	Error CS0220	Error CS0220
	/	Error CS0220	Error CS0220
unchecked	*	-2147483648	-2147483648
	/	-2147483648	Error CS0220

Кодировки

```
[Description(Value)]
public class Program
{
  public const string Value = "X\ud800Y";
  public static void Main()
  {
    var description = GetDescription(typeof(Program));
    Dump("Attribute", description.Description);
    Dump("UTF8->String", Encoding.UTF8.GetString(
      new byte[] { 0x58, 0xED, 0xA0, 0x80, 0x59 }));
    Dump("String->UTF-8", Encoding.UTF8.GetBytes(Value));
  }
}
```

Кодировки

```
[Description(Value)]
public class Program
{
  public const string Value = "X\ud800Y";
  public static void Main()
  {
    var description = GetDescription(typeof(Program));
    Dump("Attribute", description.Description);
    Dump("UTF8->String", Encoding.UTF8.GetString(
      new byte[] { 0x58, 0xED, 0xA0, 0x80, 0x59 }));
    Dump("String->UTF-8", Encoding.UTF8.GetBytes(Value));
  }
}
```

	MS.NET	Mono
IL	58 ed a0 80 59	58 59 bf bd 00
Attribute	0058 0059 fffd fffd 0000	null
UTF8->String	0058 fffd fffd 0059	0058 fffd fffd fffd 0059
String->UTF-8	58 ef bf bd 59	58 59 bf bd 00

Замыкания

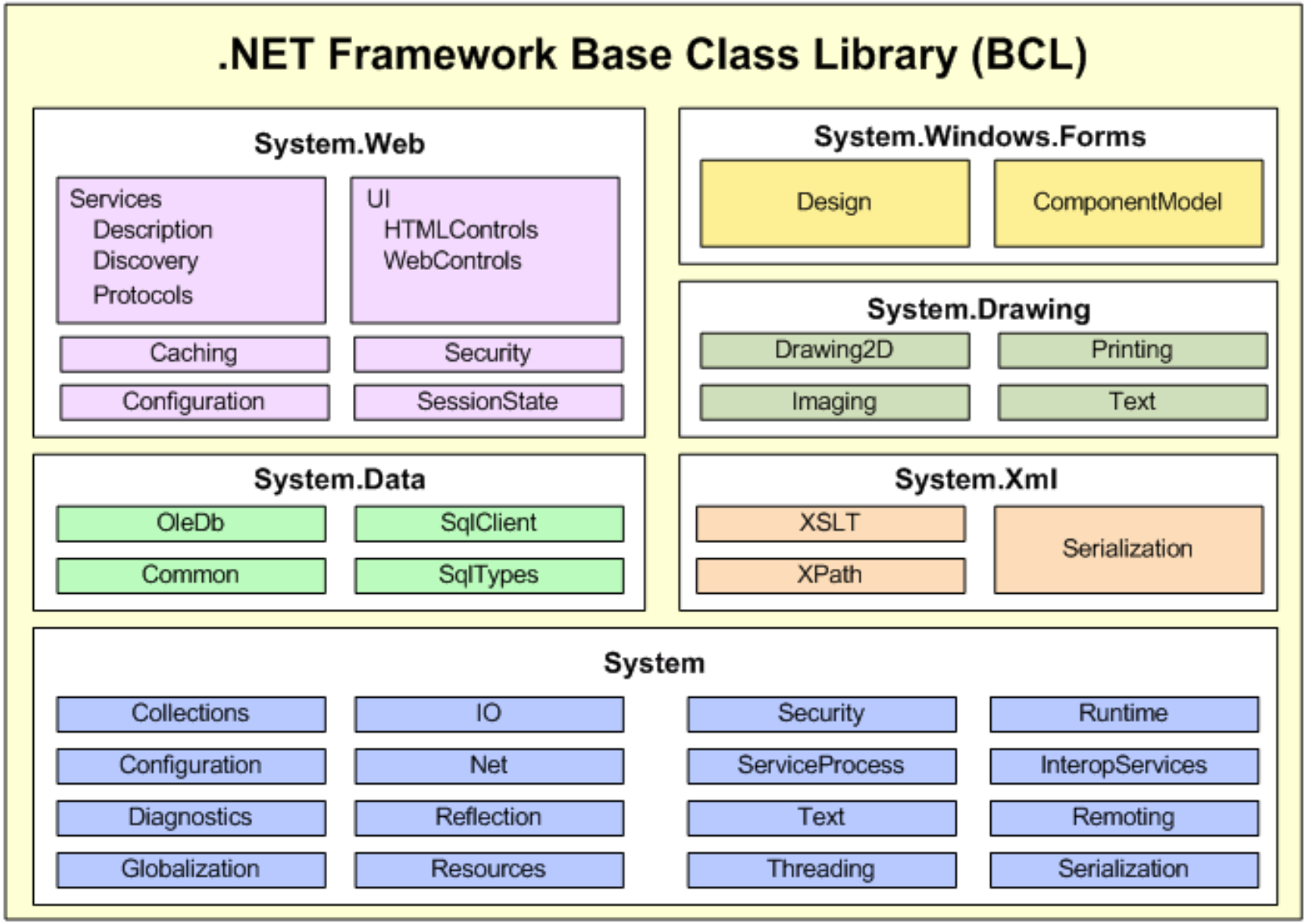
```
var numbers = new int[] { 1, 2, 3 };
var actions = new List<Action>();
foreach (var number in numbers)
    actions.Add(() => Console.WriteLine(number));
foreach (var action in actions)
    action();
```

Замыкания

```
var numbers = new int[] { 1, 2, 3 };
var actions = new List<Action>();
foreach (var number in numbers)
    actions.Add(() => Console.WriteLine(number));
foreach (var action in actions)
    action();
```

Compiler	Command line	Output
MS.NET (C#3, 3.5.30729.7903)	v3.5\csc.exe	3 3 3
MS.NET (C#4, 4.0.30319.1)	v4.0.30319\csc.exe	3 3 3
MS.NET (C#5, 4.0.30319.33440)	v4.0.30319\csc.exe	1 2 3
MS.NET (C#5, 4.0.30319.33440)	v4.0.30319\csc.exe /langversion:4	1 2 3
Mono 2.4.4	gmcs	3 3 3
Mono 3.10	mcs	1 2 3
Mono 3.10	mcs -langversion:4	1 2 3

Поговорим о базовых классах



ThreadPool.GetMaxThreads

```
int workerThreads, completionPortThreads;
ThreadPool.GetMaxThreads(
  out workerThreads, out completionPortThreads);
```

ThreadPool.GetMaxThreads

```
int workerThreads, completionPortThreads;
ThreadPool.GetMaxThreads(
  out workerThreads, out completionPortThreads);
```

Зависит от: runtime, hardware, OS.
Возможные значения:

	workerThreads
MS.NET 2.0	25
MS.NET 3.5	250
MS.NET 4.0 x86	1023
MS.NET 4.0 x64	32768
Mono 2.4.4, Ubuntu 14.04 Server	35
Mono 3.10, Ubuntu 14.04 Server	100
Mono 3.10, Kubuntu 14.04	400
Mono 3.3, Windows 8.1	800

ValueType.GetHashCode

```
var a1 = new KeyValuePair<int, int>(1, 2);
var a2 = new KeyValuePair<int, int>(1, 3);
Console.WriteLine(
  a1.GetHashCode() != a2.GetHashCode());

var b1 = new KeyValuePair<int, string>(1, "x");
var b2 = new KeyValuePair<int, string>(1, "y");
Console.WriteLine(
  b1.GetHashCode() != b2.GetHashCode());
```

ValueType.GetHashCode

```
var a1 = new KeyValuePair<int, int>(1, 2);
var a2 = new KeyValuePair<int, int>(1, 3);
Console.WriteLine(
  a1.GetHashCode() != a2.GetHashCode());

var b1 = new KeyValuePair<int, string>(1, "x");
var b2 = new KeyValuePair<int, string>(1, "y");
Console.WriteLine(
  b1.GetHashCode() != b2.GetHashCode());
```

	<int, int>	<int, string>
MS.NET	True	False
Mono	True	True

Uri.AbsoluteUri

```
var uri = new Uri("http://x/%2F/y.?%3D%3F");
Console.WriteLine("AbsoluteUri: " +
                  uri.AbsoluteUri);
Console.WriteLine("ToString(): " +
                  uri.ToString());
```

Uri.AbsoluteUri

```
var uri = new Uri("http://x/%2F/y.?%3D%3F");
Console.WriteLine("AbsoluteUri: " +
                  uri.AbsoluteUri);
Console.WriteLine("ToString(): " +
                  uri.ToString());
```

	AbsoluteUri				ToString()			
	.	%2F	%3D	%3F	.	%2F	%3D	%3F
MS.NET 4.0	∅	/	%3D	%3F	∅	/	=	?
MS.NET 4.0 Fix	∅	%2F	%3D	%3F	∅	/	=	?
MS.NET 4.5	.	%2F	%3D	%3F	.	%2F	%3D	%3F
Mono 3.2.8	.	/	%3D	%3F	.	/	=	%3F
Mono 3.10	.	%2F	%3D	%3F	.	%2F	%3D	%3F

TypeBuilder.CreateType

```
private interface IFoo {}
void Main()
{
  var typeBuilder = moduleBuilder.DefineType(
    "Foo", TypeAttributes.Public,
    typeof(object), new[] { typeof(IFoo) });
  typeBuilder.CreateType();
}
```

TypeBuilder.CreateType

```
private interface IFoo {}
void Main()
{
  var typeBuilder = moduleBuilder.DefineType(
    "Foo", TypeAttributes.Public,
    typeof(object), new[] { typeof(IFoo) });
  typeBuilder.CreateType();
}
```

	Output
MS.NET	TypeLoadException
Mono 3.8	OK

Mono Bug 22059 (August 13, 2014)
Fixed in 68e5cc3 (August 18, 2014)

List.ForEach

```
var list = new List<int> { 1, 2 };
list.ForEach(i =>
{
    if (i == 1)
        list.Add(3);
    Console.WriteLine(i);
});
```

List.ForEach

```
var list = new List<int> { 1, 2 };
list.ForEach(i =>
{
    if (i == 1)
        list.Add(3);
    Console.WriteLine(i);
});
```

	Output
MS.NET 4.0	1 2 3
MS.NET 4.5	1 InvalidOperationException
Mono 3.10	1 2 3

Mono Bug 24775 (November 24, 2014)
Fixed in 5517c56 (November 25, 2014)

Поговорим о сборщике мусора

Сколько весит объект?

```
const int ObjectCount = 10000000;
var array = new object[ObjectCount];
var before = GC.GetTotalMemory(true);
for (int i = 0; i < ObjectCount; i++)
  array[i] = new object();
var after = GC.GetTotalMemory(true);
var objectSize = (after - before) * 1.0 /
                 ObjectCount;
GC.KeepAlive(array);
```

Сколько весит объект?

```
const int ObjectCount = 10000000;
var array = new object[ObjectCount];
var before = GC.GetTotalMemory(true);
for (int i = 0; i < ObjectCount; i++)
  array[i] = new object();
var after = GC.GetTotalMemory(true);
var objectSize = (after - before) * 1.0 /
                 ObjectCount;
GC.KeepAlive(array);
```

	MS.NET x86	MS.NET x64
object	12	24

Размер объекта в Mono-x64

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0
Boehm	100	0

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0
Boehm	100	0
Boehm	500	8.192

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0
Boehm	100	0
Boehm	500	8.192
Boehm	600	13.653

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0
Boehm	100	0
Boehm	500	8.192
Boehm	600	13.653
Boehm	1000	12.288

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0
Boehm	100	0
Boehm	500	8.192
Boehm	600	13.653
Boehm	1000	12.288
Boehm	10000000	16

Размер объекта в Mono-x64

GC	ObjectCount	Size
Boehm	1	0
Boehm	10	0
Boehm	100	0
Boehm	500	8.192
Boehm	600	13.653
Boehm	1000	12.288
Boehm	10000000	16
Sgen	1	16

Размер long-массива в Mono-x64-Sgen

Размер long-массива в Mono-x64-Sgen

n	Size
0	40

Размер long-массива в Mono-x64-Sgen

n	Size
0	40
1	40

Размер long-массива в Mono-x64-Sgen

n	Size
0	40
1	40
2	56

Размер long-массива в Mono-x64-Sgen

n	Size
0	40
1	40
2	56
3	56

Размер long-массива в Mono-x64-Sgen

n	Size
0	40
1	40
2	56
3	56
4	88

Размер long-массива в Mono-x64-Sgen

n	Size
0	40
1	40
2	56
3	56
4	88
5	88
6	88
7	88

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size
0	40	11	120
1	40		
2	56		
3	56		
4	88		
5	88		
6	88		
7	88		
8	120		
9	120		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size
0	40	11	120
1	40	12	176
2	56	18	176
3	56		
4	88		
5	88		
6	88		
7	88		
8	120		
9	120		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size
0	40	11	120
1	40	12	176
2	56	18	176
3	56	19	248
4	88	27	248
5	88		
6	88		
7	88		
8	120		
9	120		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size
0	40	11	120
1	40	12	176
2	56	18	176
3	56	19	248
4	88	27	248
5	88	28	352
6	88	40	352
7	88		
8	120		
9	120		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size
0	40	11	120
1	40	12	176
2	56	18	176
3	56	19	248
4	88	27	248
5	88	28	352
6	88	40	352
7	88	41	504
8	120	59	504
9	120		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size
0	40	11	120	84	704
1	40	12	176		
2	56	18	176		
3	56	19	248		
4	88	27	248		
5	88	28	352		
6	88	40	352		
7	88	41	504		
8	120	59	504		
9	120	60	704		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size
0	40	11	120	84	704
1	40	12	176	85	1016
2	56	18	176	123	1016
3	56	19	248		
4	88	27	248		
5	88	28	352		
6	88	40	352		
7	88	41	504		
8	120	59	504		
9	120	60	704		

Размер long-массива в Mono-x64-Sgen

n	Size
0	40
1	40
2	56
3	56
4	88
5	88
6	88
7	88
8	120
9	120

n	Size
11	120
12	176
18	176
19	248
27	248
28	352
40	352
41	504
59	504
60	704

n	Size
84	704
85	1016
123	1016
124	1360
166	1360

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size
0	40	11	120	84	704
1	40	12	176	85	1016
2	56	18	176	123	1016
3	56	19	248	124	1360
4	88	27	248	166	1360
5	88	28	352	167	2040
6	88	40	352	251	2040
7	88	41	504		
8	120	59	504		
9	120	60	704		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size
0	40	11	120	84	704
1	40	12	176	85	1016
2	56	18	176	123	1016
3	56	19	248	124	1360
4	88	27	248	166	1360
5	88	28	352	167	2040
6	88	40	352	251	2040
7	88	41	504	252	2728
8	120	59	504	337	2728
9	120	60	704		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size	n	Size
0	40	11	120	84	704	507	4088
1	40	12	176	85	1016		
2	56	18	176	123	1016		
3	56	19	248	124	1360		
4	88	27	248	166	1360		
5	88	28	352	167	2040		
6	88	40	352	251	2040		
7	88	41	504	252	2728		
8	120	59	504	337	2728		
9	120	60	704	338	4088		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size	n	Size
0	40	11	120	84	704	507	4088
1	40	12	176	85	1016	509	5456
2	56	18	176	123	1016	678	5456
3	56	19	248	124	1360		
4	88	27	248	166	1360		
5	88	28	352	167	2040		
6	88	40	352	251	2040		
7	88	41	504	252	2728		
8	120	59	504	337	2728		
9	120	60	704	338	4088		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size	n	Size
0	40	11	120	84	704	507	4088
1	40	12	176	85	1016	509	5456
2	56	18	176	123	1016	678	5456
3	56	19	248	124	1360	679	8000
4	88	27	248	166	1360	996	8000
5	88	28	352	167	2040		
6	88	40	352	251	2040		
7	88	41	504	252	2728		
8	120	59	504	337	2728		
9	120	60	704	338	4088		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size	n	Size
0	40	11	120	84	704	507	4088
1	40	12	176	85	1016	509	5456
2	56	18	176	123	1016	678	5456
3	56	19	248	124	1360	679	8000
4	88	27	248	166	1360	996	8000
5	88	28	352	167	2040	997	8008
6	88	40	352	251	2040		
7	88	41	504	252	2728		
8	120	59	504	337	2728		
9	120	60	704	338	4088		

Размер long-массива в Mono-x64-Sgen

n	Size	n	Size	n	Size	n	Size
0	40	11	120	84	704	507	4088
1	40	12	176	85	1016	509	5456
2	56	18	176	123	1016	678	5456
3	56	19	248	124	1360	679	8000
4	88	27	248	166	1360	996	8000
5	88	28	352	167	2040	997	8008
6	88	40	352	251	2040	998	8016
7	88	41	504	252	2728	999	8024
8	120	59	504	337	2728	1000	8032
9	120	60	704	338	4088	1001	8040

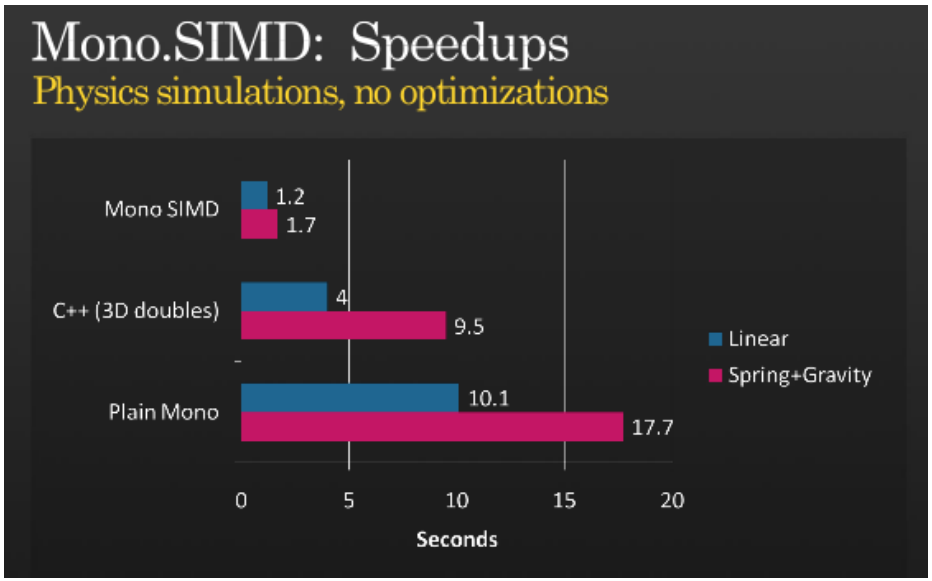
Исходники SGen

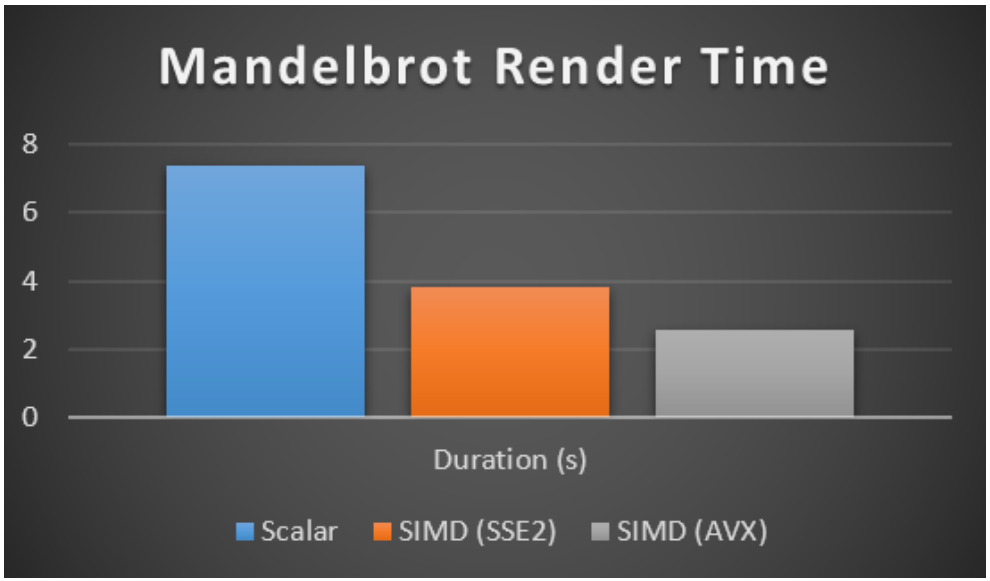
```
// mono/mono/metadata/sgen-internal.c
/* keep each size a multiple of ALLOC_ALIGN */
#if SIZEOF_VOID_P == 4
static const int allocator_sizes [] = {
      8,   16,   24,   32,   40,   48,   64,   80,
     96,  128,  160,  192,  224,  248,  296,  320,
    384,  448,  504,  528,  584,  680,  816, 1088,
   1360, 2044, 2336, 2728, 3272, 4092, 5456, 8188 };
#else
static const int allocator_sizes [] = {
      8,   16,   24,   32,   40,   48,   64,   80,
     96,  128,  160,  192,  224,  248,  320,  328,
    384,  448,  528,  584,  680,  816, 1016, 1088,
   1360, 2040, 2336, 2728, 3272, 4088, 5456, 8184 };
#endif

// mono/mono/metadata/sgen-conf.h
#define SGEN_MAX_SMALL_OBJ_SIZE 8000
```

Поговорим о JIT-компиляторе

Как дела с SIMD?

- P/Invoke
- Mono.SIMD
- Microsoft.Bcl.Simd





Размотка циклов

```
public int Sum997()
{
  int sum = 0;
  for (int i = 0; i < 997; i++)
    sum += a[i];
  return sum;
}
public int Sum1000()
{
  int sum = 0;
  for (int i = 0; i < 1000; i++)
    sum += a[i];
  return sum;
}
```

Размотка циклов

```
public int Sum997()
{
  int sum = 0;
  for (int i = 0; i < 997; i++)
    sum += a[i];
  return sum;
}
public int Sum1000()
{
  int sum = 0;
  for (int i = 0; i < 1000; i++)
    sum += a[i];
  return sum;
}
```

```
// Аналог Sum1000()
// с размоткой цикла (loop unrolling)
public int Sum1000Unrolled()
{
  int sum = 0;
  for (int i = 0; i < 1000; i += 4)
  {
    sum += a[i];
    sum += a[i+1];
    sum += a[i+2];
    sum += a[i+3];
  }
  return sum;
}
```

Размотка циклов

NonStatic / Static

```
00 sub  rsp,28h
04 mov  r8,rcx
07 xor  ecx,ecx
09 mov  edx,ecx                     ; sum = 0
0b nop  dword ptr [rax+rax]
10 xor  r10d,r10d                   ; i = 0
13 mov  r9,qword ptr [r8+8]         ; r9 = &nonStaticField
17 mov  rax,qword ptr [r9+8]        ; rax = nonStaticField.Length
1b mov  r11d,3E4h                   ; r11 = 996
21 cmp  r11,rax                     ; if r11 >= nonStaticField.Length then
24 jae  000000000000008A            ; throw IndexOutOfRangeException
26 mov  r11d,3E5h                   ; r11 = 997
2c cmp  r11,rax                     ; if r11 >= nonStaticField.Length then
2f jae  000000000000008A            ; throw IndexOutOfRangeException
31 mov  r11d,3E6h                   ; r11 = 998
37 cmp  r11,rax                     ; if r11 >= nonStaticField.Length then
3a jae  000000000000008A            ; throw IndexOutOfRangeException
3c mov  r11d,3E7h                   ; r11 = 999
42 cmp  r11,rax                     ; if r11 >= nonStaticField.Length then
45 jae  000000000000008A            ; throw IndexOutOfRangeException
47 nop  word ptr [rax+rax+00000000h]
50 mov  eax,dword ptr [r9+r10+10h]  ; eax = nonStaticField[i]
55 add  edx,eax                     ; sum += eax
57 mov  eax,dword ptr [r9+r10+14h]  ; eax = nonStaticField[i+1]
5c add  edx,eax                     ; sum += eax
5e mov  eax,dword ptr [r9+r10+18h]  ; eax = nonStaticField[i+2]
63 add  edx,eax                     ; sum += eax
65 mov  eax,dword ptr [r9+r10+1Ch]  ; eax = nonStaticField[i+3]
6a add  edx,eax                     ; sum += eax
6c add  r10,10h                     ; i += 4
70 cmp  r10,0FA0h                   ; if i < 1000 then
77 jl   0000000000000050            ; loop by i
79 inc  ecx                         ; iteration++
7b cmp  ecx,0F4240h                 ; if iteration < 1000000  then
81 jl   0000000000000010            ; loop by iteration
83 mov  eax,edx                     ; eax = sum (Result)
85 add  rsp,28h
89 ret
8a call 000000005FA4AE14            ; IndexOutOfRangeException
8f nop
```

```
00 sub  rsp,28h
04 xor  ecx,ecx                     ; iteration = 0
06 mov  edx,ecx                     ; sum = 0
08 nop  dword ptr [rax+rax+00000000h]
10 xor  r8d,r8d                     ; i = 0
13 mov  r9,12D756F0h                ; r9 = staticField
1d mov  r9,qword ptr [r9]           ; r9 = &staticField
20 mov  r10,qword ptr [r9+8]        ; r10 = staticField.Length
24 cmp  r8,r10                      ; if r8 >= staticField.Length then
27 jae  0000000000000080            ; throw IndexOutOfRangeException
29 mov  eax,dword ptr [r9+r8*4+10h] ; eax = staticField[i]
2e add  edx,eax                     ; sum += eax
30 lea  rax,[r8+1]                  ; rax = i+1
34 cmp  rax,r10                     ; if rax >= staticField.Length then
37 jae  0000000000000080            ; throw IndexOutOfRangeException
39 mov  eax,dword ptr [r9+rax*4+10h] ; eax = staticField[i+1]
3e add  edx,eax                     ; sum += eax
40 lea  rax,[r8+2]                  ; rax = i+2
44 cmp  rax,r10                     ; if rax >= staticField.Length then
47 jae  0000000000000080            ; throw IndexOutOfRangeException
49 mov  eax,dword ptr [r9+rax*4+10h] ; eax = staticField[i+2]
4e add  edx,eax                     ; sum += eax
50 lea  rax,[r8+3]                  ; rax = i+3
54 cmp  rax,r10                     ; if rax >= staticField.Length then
57 jae  0000000000000080            ; throw IndexOutOfRangeException
59 mov  eax,dword ptr [r9+rax*4+10h] ; eax = staticField[i+3]
5e add  edx,eax                     ; sum += eax
60 add  r8,4                        ; i += 4
64 cmp  r8,3E8h                     ; if i < 1000 then
6b jl   0000000000000013            ; loop by i
6d inc  ecx                         ; iteration++
6f cmp  ecx,0F4240h                 ; if iteration < 1000000 then
75 jl   0000000000000010            ; loop by iteration
77 mov  eax,edx                     ; eax = sum (result)
79 add  rsp,28h
7d ret
7e xchg ax,ax
80 call 000000005FA49F64            ; IndexOutOfRangeException
85 nop
```

Размотка циклов

Различия крупным планом:

```
; NonStaticRun-x64.asm
; eax = nonStaticField[i]
mov  eax, dword ptr [r9+r10+10h]
; i += 4
add  r10, 10h

; StaticRun-x64.asm
; eax = staticField[i]
mov  eax, dword ptr [r9+r8*4+10h]
; i += 4
add  r8, 4
```

Кто быстрее?

```
// Целевой интерфейс
interface IFoo
{
  int Inc(int x);
}
void Run(IFoo foo)
{
  for (int i = 0; i < 1000000; i++)
    foo.Inc(0);
}
// Запускаем Run()
// с двумя имлементациями IFoo
Run(new FastFoo());
Run(new SlowFoo());
```

```
class FastFoo : IFoo
{
  public int Inc(int x)
  {
    return x + 1;
  }
}
class SlowFoo : IFoo
{
  public int Inc(int x)
  {
    return 1 + x;
  }
}
```

Кто быстрее?

```
// Целевой интерфейс
interface IFoo
{
  int Inc(int x);
}
void Run(IFoo foo)
{
  for (int i = 0; i < 1000000; i++)
    foo.Inc(0);
}
// Запускаем Run()
// с двумя имлементациями IFoo
Run(new FastFoo());
Run(new SlowFoo());
```

```
class FastFoo : IFoo
{
  public int Inc(int x)
  {
    return x + 1;
  }
}
class SlowFoo : IFoo
{
  public int Inc(int x)
  {
    return 1 + x;
  }
}
```

$$\text{MS.NET:}\quad \text{Time(Fast)} \ll \text{Time(Slow)}$$
$$\text{Mono:}\quad \text{Time(Fast)} \approx \text{Time(Slow)}$$

Поговорим о внутренностях рантайма

Осторожно, сложный пример!

Внутреннее устройство массивов

```
var a = new object[1][]; // Address: 0x012410
a[0] = new object[1];    // Address: 0x012424

// Memory dump (a)                              //
// 0x01240C  000000  SyncBlockIndex (a)         //
// 0x012410  1731d4  object[][] MethodTable // a
// 0x012414  000001  a.Length                   //
// 0x012418  854d7a  object[] TypeHandle        // !!!
// 0x01241C  012424  Address of a[0]            //
//                                              //
// Memory dump (a[0])                           //
// 0x012420  000000  SyncBlockIndex (a[0])      //
// 0x012424  bfab98  object[] MethodTable       // a[0]
// 0x012428  000001  a[0].Length                //
// 0x01242C  c4b060  Address of a[0][0]         //
```

Исходники MS CLI 2.0

```
// sscli20\clr\src\vm\typehandle.h

// A TypeHandle is the FUNDAMENTAL concept of type identity in the CLR.
// That is two types are equal if and only if their type handles
// are equal.  A TypeHandle, is a pointer sized struture that encodes
// everything you need to know to figure out what kind of type you are
// actually dealing with.

// At the present time a TypeHandle can point at two possible things
//
//  1) A MethodTable  (Intrinsics, Classes, Value Types
//                     and their instantiations)
//  2) A TypeDesc     (all other cases: arrays, byrefs, pointer types,
//                     function pointers, generic type variables)
//
// or with IL stubs, a third thing:
//
//  3) A MethodTable for a native value type.
```

Исходники MS CLI 2.0

```
class ArrayBase : public Object
{
    // ...
    // What comes after this conceptually is:
    // TypeHandle elementType;
    // ...

// sscli20\clr\src\vm\object.h
FORCEINLINE BOOL IsUnsharedMT() const {
    LEAF_CONTRACT;
    STATIC_CONTRACT_SO_TOLERANT;
    return((m_asTAddr & 2) == 0);
}

FORCEINLINE BOOL IsTypeDesc() const  {
    WRAPPER_CONTRACT;
    return(!IsUnsharedMT());
}

// MethodTable: 0, 1, 4, 5, 8, 9, C, D
// TypeDesc   : 2, 3, 6, 7, A, B, E, F
```

MethodTable или TypeDesc?

```
var types = new[] {
  typeof(int),   typeof(object),  typeof(Stream),
  typeof(int[]), typeof(int[][]), typeof(object[]) };
foreach (var type in types)
{
  bool isTypeDesc = (((int)type.TypeHandle.Value)&2) > 0;
  Console.WriteLine("{0}: {1}",
    type.Name, isTypeDesc ? "TypeDesc" : "MethodTable");
}
```

MethodTable или TypeDesc?

```
var types = new[] {
  typeof(int),   typeof(object),  typeof(Stream),
  typeof(int[]), typeof(int[][]), typeof(object[]) };
foreach (var type in types)
{
  bool isTypeDesc = (((int)type.TypeHandle.Value)&2) > 0;
  Console.WriteLine("{0}: {1}",
    type.Name, isTypeDesc ? "TypeDesc" : "MethodTable");
}
```

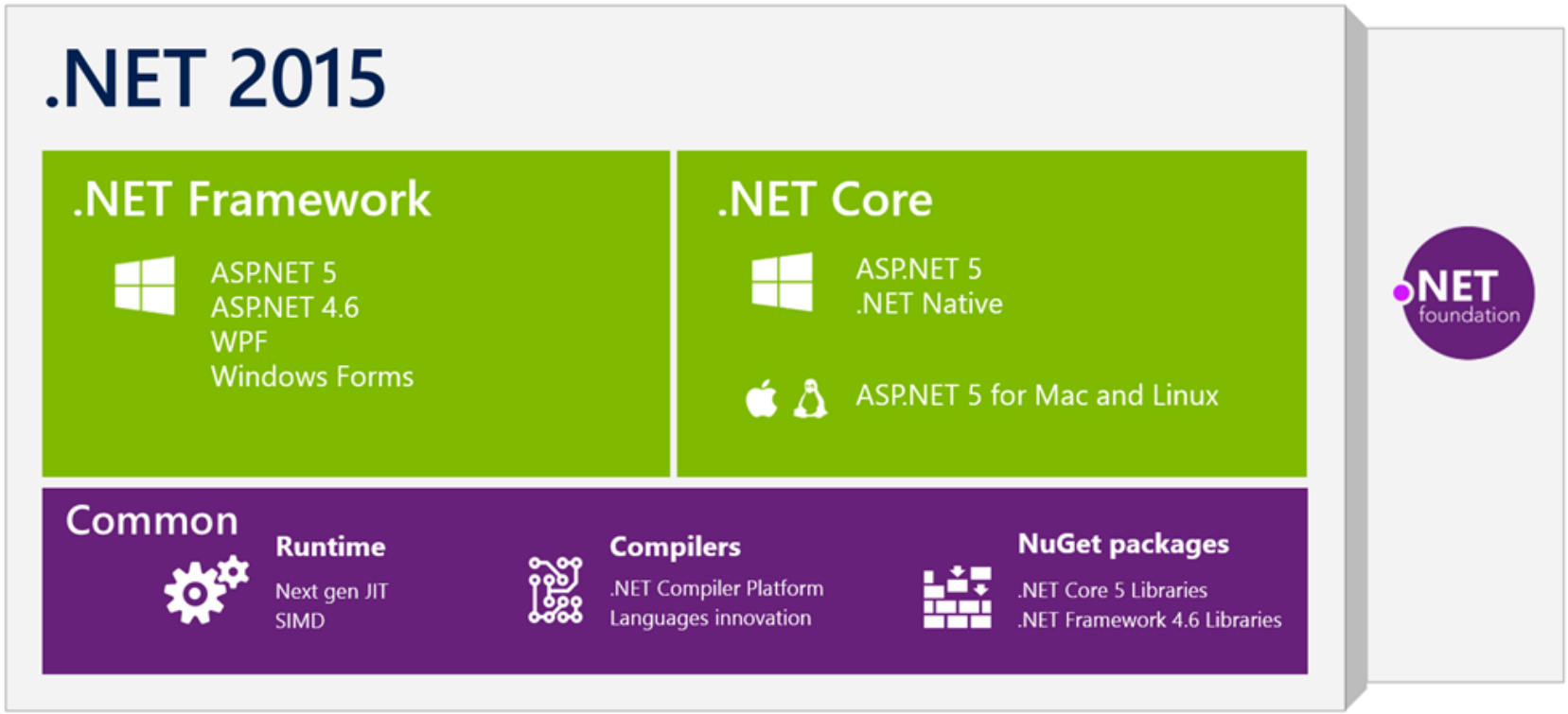
	MS.NET	Mono
Int32	MethodTable	MethodTable
Object	MethodTable	MethodTable
Stream	MethodTable	MethodTable
Int32[]	TypeDesc	MethodTable
Int32[][]	TypeDesc	MethodTable
Object[]	TypeDesc	MethodTable

Чёрная магия

```
public class OneLong { public long X; }
public class TwoInt { public int Y1, Y2; }
public unsafe IntPtr GetAddress(object obj)
{
  var typedReference = __makeref(obj);
  return *(IntPtr*)(&typedReference);
}
public unsafe T Convert<T>(IntPtr address)
{
  var fakeInstance = default(T);
  var typedReference = __makeref(fakeInstance);
  *(IntPtr*)(&typedReference) = address;
  return __refvalue(typedReference, T);
}
public void Run()
{
  var oneLong = new OneLong { X = 1 + (2L << 32) };
  var twoInt = Convert<TwoInt>(GetAddress(oneLong));
  Console.WriteLine(twoInt.Y1 + " " + twoInt.Y2); // 1 2
  oneLong.X = 3 + (4L << 32);
  Console.WriteLine(twoInt.Y1 + " " + twoInt.Y2); // 3 4
}
```

Хорошие времена для .NET-чиков

The open .NET ecosystem



Напутствие

Думайте про различия в рантаймах!

Вопросы?

Андрей Акиньшин, Энтерра
http://aakinshin.ru
andrey.akinshin@gmail.com