Распространённые ошибки оценки производительности .NET-приложений

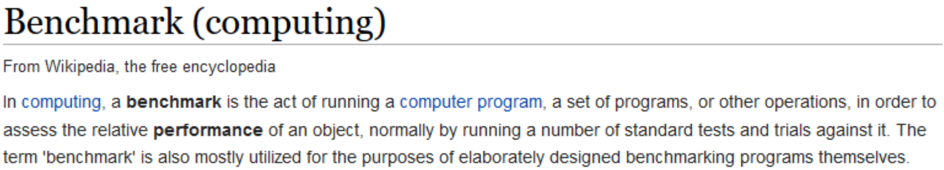
Андрей Акиньшин, JetBrains

SPB .NET Meetup #6

Часть 1

Теория

Поговорим про бенчмарки



Benchmark (computing)

From Wikipedia, the free encyclopedia

In computing, a **benchmark** is the act of running a computer program, a set of programs, or other operations, in order to assess the relative **performance** of an object, normally by running a number of standard tests and trials against it. The term 'benchmark' is also mostly utilized for the purposes of elaborately designed benchmarking programs themselves.

©*Wikipedia*

Теория: Зачем люди делают бенчмарки?

1. **Ради холивора**: Node.js – Но Java... – Node.js!
2. **Ради маркетинга**: проверить, что мы вкладываемся в установленные критерии
3. **Ради инжиниринга**: изолировать и зафиксировать перформансный феномен, чтобы была референсная точка для улучшения
4. **Ради науки**: понять, какой моделью описывается система, и на основании этой модели предсказать будущее поведение

©*Aleksey Shipilëv*

План работ

1. Поставить задачу
2. Выбрать метрики
3. Выбрать инструмент
4. Провести эксперимент
5. Выполнить анализ, сделать выводы

Окружение

- C#-компилятор: версия старого csc? Roslyn?
- Версия CLR: CLR2? CLR4? CoreCLR? Mono?
- Версия ОС: Windows? Linux? MacOS? FreeBSD?
- Версия JIT: x86? x64? RyuJIT?
- Версия GC: MS (какой CLR?)? Mono (Boehm/Sgen)?
- Компиляция: JIT? NGen? .NET Native?
- Железо: ???
- ...

Запуск бенчмарка

- Release build
- Без дебаггера
- Выключите другие приложения
- Используйте максимальную производительность

DateTime vs Stopwatch

```
var start = DateTime.Now;
Foo();
var finish = DateTime.Now;
Console.WriteLine((finish - start).Milliseconds);
```

vs

```
var sw = Stopwatch.StartNew();
Foo();
sw.Stop();
Console.WriteLine(sw.ElapsedMilliseconds);
```

DateTime vs Stopwatch

```
var start = DateTime.Now;
Foo();
var finish = DateTime.Now;
Console.WriteLine((finish - start).Milliseconds);
```

vs

```
var sw = Stopwatch.StartNew();
Foo();
sw.Stop();
Console.WriteLine(sw.ElapsedMilliseconds);
```

Возможные значения (Windows 10, MS.NET, Core i7):

	Granularity	Latency
DateTime	1 000 000 ns*	30–40 ns
Stopwatch	370–466 ns	14–18 ns

Итерации

Плохо:

```
// Granularity(Stopwatch) = 466 ns
// Latency(Stopwatch) = 18 ns
var sw = Stopwatch.StartNew();
Foo(); // 100 ns
sw.Stop();
Console.WriteLine(sw.ElapsedMilliseconds);
```

Итерации

Плохо:

```
// Granularity(Stopwatch) = 466 ns
// Latency(Stopwatch) = 18 ns
var sw = Stopwatch.StartNew();
Foo(); // 100 ns
sw.Stop();
Console.WriteLine(sw.ElapsedMilliseconds);
```

Лучше:

```
var sw = Stopwatch.StartNew();
for (int i = 0; i < N; i++) // (N * 100 + eps) ns
    Foo();
sw.Stop();
var total = sw.ElapsedTicks / Stopwatch.Frequency;
Console.WriteLine(total / N);
```

Прогрев

Запустим бенчмарк несколько раз:

```
int[] x = new int[128 * 1024 * 1024];
for (int iter = 0; iter < 5; iter++)
{
    var sw = Stopwatch.StartNew();
    for (int i = 0; i < x.Length; i += 16)
        x[i]++;
    sw.Stop();
    Console.WriteLine(sw.ElapsedMilliseconds);
}
```

Прогрев

Запустим бенчмарк несколько раз:

```
int[] x = new int[128 * 1024 * 1024];
for (int iter = 0; iter < 5; iter++)
{
    var sw = Stopwatch.StartNew();
    for (int i = 0; i < x.Length; i += 16)
        x[i]++;
    sw.Stop();
    Console.WriteLine(sw.ElapsedMilliseconds);
}
```

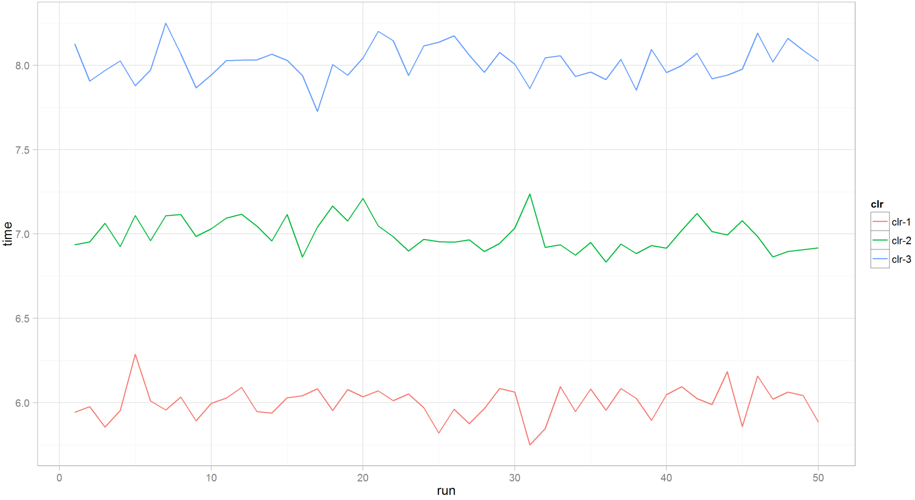
Результат:

```
176
81
62
62
62
```

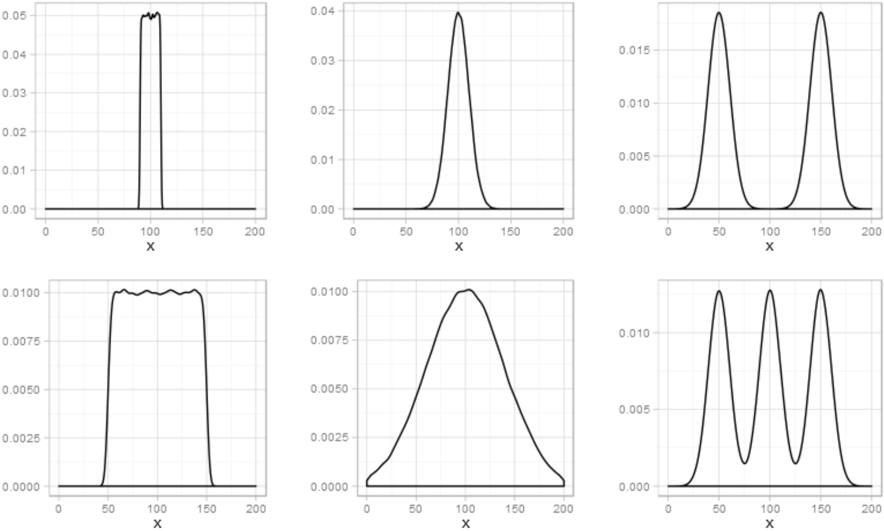
Несколько запусков бенчмарка

```
Run 01 : 529.8674 ns/op
Run 02 : 532.7541 ns/op
Run 03 : 558.7448 ns/op
Run 04 : 555.6647 ns/op
Run 05 : 539.6401 ns/op
Run 06 : 539.3494 ns/op
Run 07 : 564.3222 ns/op
Run 08 : 551.9544 ns/op
Run 09 : 550.1608 ns/op
Run 10 : 533.0634 ns/op
```

Несколько запусков CLR



Статистика


0.05
0.04
0.03
0.02
0.01
0.00
0
50
100
150
200
x
0.04
0.03
0.02
0.01
0.00
0
50
100
150
200
x
0.015
0.010
0.005
0.000
0
50
100
150
200
x
0.0100
0.0075
0.0050
0.0025
0.0000
0
50
100
150
200
x
0.0100
0.0075
0.0050
0.0025
0.0000
0
50
100
150
200
x
0.010
0.005
0.000
0
50
100
150
200
x

Накладные расходы

```
var sw = Stopwatch.StartNew();
int x = 0;
for (int i = 0; i < N; i++) // overhead
    x++; // target operation
sw.Stop();
```

Изоляция бенчмарков

Плохо:

```
Measure1();
Measure2();
```

Изоляция бенчмарков

Плохо:

```
Measure1();
Measure2();
```

Вспомним про:

- Interface method dispatch
- Garbage collector

Борьба с оптимизациями

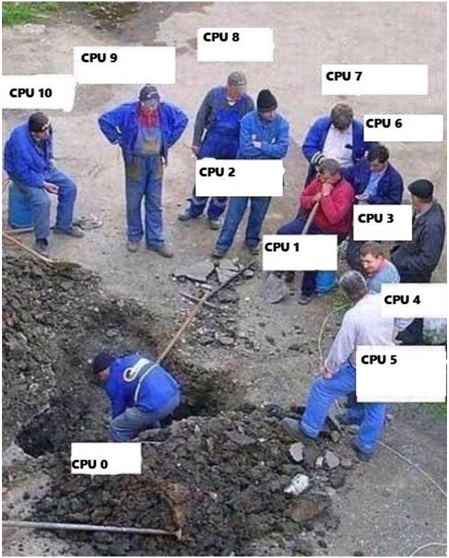
- Dead code elimination
- Inlining
- Constant folding
- Instruction Level Parallelism
- Branch prediction
- ...

Организация доступа к памяти

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access	120 ns	6 min
Solid-state disk I/O	50-150 μs	2-6 days
Rotational disk I/O	1-10 ms	1-12 months
Internet: SF to NYC	40 ms	4 years
Internet: SF to UK	81 ms	8 years
Internet: SF to Australia	183 ms	19 years
OS virtualization reboot	4 s	423 years
SCSI command time-out	30 s	3000 years
Hardware virtualization reboot	40 s	4000 years
Physical system reboot	5 m	32 millenia

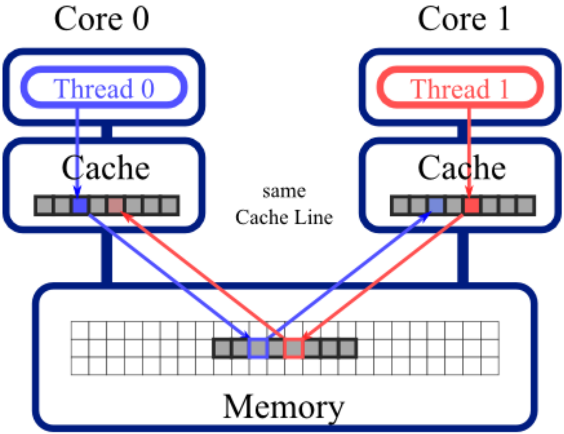
© Systems Performance: Enterprise and the Cloud

Processor affinity



Многопоточные бенчмарки сложны

False sharing:



© http://colfaxresearch.com/optimization-techniques-for-the-intel-mic-architecture-part-3-of-3-false-sharing-and-padding/

False sharing в действии

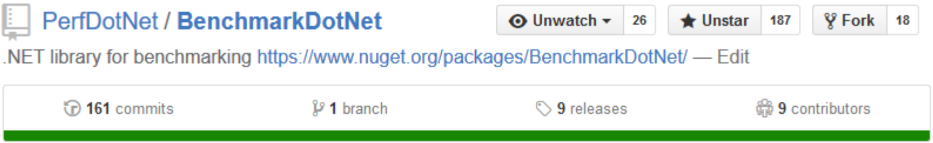
```
private static int[] x = new int[1024];

private void Inc(int p)
{
  for (int i = 0; i < 10000001; i++)
      x[p]++;
}

private void Run(int step)
{
  var sw = Stopwatch.StartNew();
  Task.WaitAll(
      Task.Factory.StartNew(() => Inc(0 * step)),
      Task.Factory.StartNew(() => Inc(1 * step)),
      Task.Factory.StartNew(() => Inc(2 * step)),
      Task.Factory.StartNew(() => Inc(3 * step)));
  Console.WriteLine(sw.ElapsedMilliseconds);
}
```

False sharing в действии

```
private static int[] x = new int[1024];

private void Inc(int p)
{
  for (int i = 0; i < 10000001; i++)
      x[p]++;
}

private void Run(int step)
{
  var sw = Stopwatch.StartNew();
  Task.WaitAll(
      Task.Factory.StartNew(() => Inc(0 * step)),
      Task.Factory.StartNew(() => Inc(1 * step)),
      Task.Factory.StartNew(() => Inc(2 * step)),
      Task.Factory.StartNew(() => Inc(3 * step)));
  Console.WriteLine(sw.ElapsedMilliseconds);
}
```

Run(1)	Run(256)
∼400	∼150

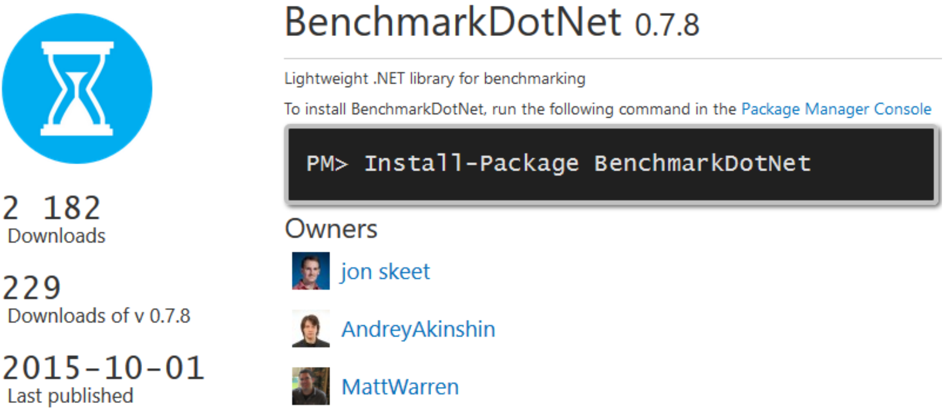
BenchmarkDotNet



PerfDotNet / **BenchmarkDotNet**

.NET library for benchmarking https://www.nuget.org/packages/BenchmarkDotNet/ — Edit

161 commits   **1** branch   **9** releases   **9** contributors



BenchmarkDotNet 0.7.8

2 182
Downloads

229
Downloads of v 0.7.8

2015-10-01
Last published

Lightweight .NET library for benchmarking

To install BenchmarkDotNet, run the following command in the Package Manager Console

```
PM> Install-Package BenchmarkDotNet
```

Owners

jon skeet

AndreyAkinshin

MattWarren

BenchmarkDotNet

v0.7.8:

- Создание отдельного проекта для каждого бенчмарка
- Запуск под разными окружениями
- Прогрев, многократный запуск, статистики
- Анализ накладных расходов
- И много чего ещё...

BenchmarkDotNet

v0.7.8:

- Создание отдельного проекта для каждого бенчмарка
- Запуск под разными окружениями
- Прогрев, многократный запуск, статистики
- Анализ накладных расходов
- И много чего ещё...

В следующих сериях:

- Просмотр IL и ASM
- Графики
- Поддержка CoreCLR/.NET Native
- Многопоточные бенчмарки

Практика

Сумма элементов массива

```
const int N = 1024;
int[,] a = new int[N, N];
```

```
[Benchmark]
public double SumIj()
{
  var sum = 0;
  for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
      sum += a[i, j];
  return sum;
}
```

```
[Benchmark]
public double SumJi()
{
  var sum = 0;
  for (int j = 0; j < N; j++)
    for (int i = 0; i < N; i++)
      sum += a[i, j];
  return sum;
}
```

Сумма элементов массива

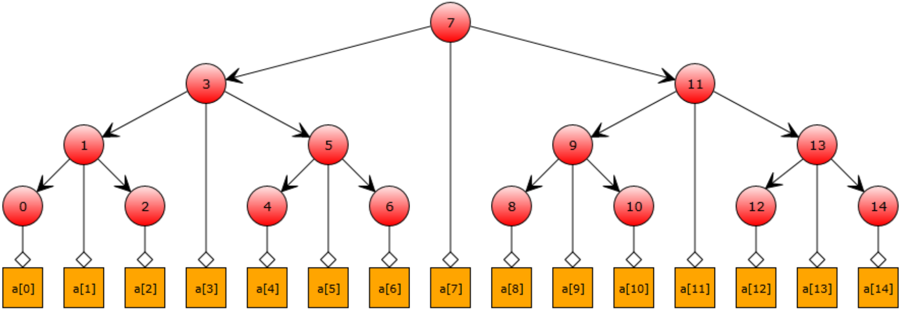
```
const int N = 1024;
int[,] a = new int[N, N];
```

```
[Benchmark]
public double SumIj()
{
  var sum = 0;
  for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
      sum += a[i, j];
  return sum;
}
```

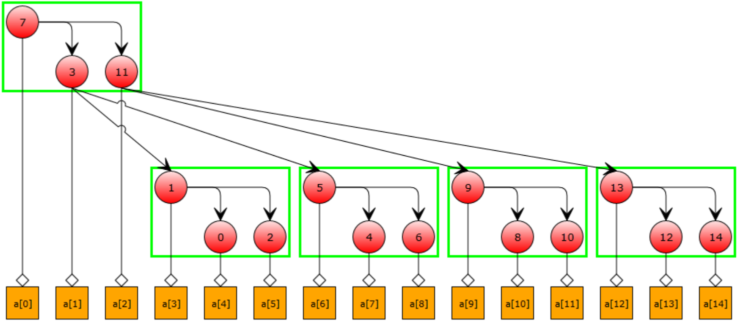
```
[Benchmark]
public double SumJi()
{
  var sum = 0;
  for (int j = 0; j < N; j++)
    for (int i = 0; i < N; i++)
      sum += a[i, j];
  return sum;
}
```

	SumIj	SumJi
LegacyJIT-x86	1 попугай	3.5 попугая

Binary Search



Cache-Conscious Binary Search



Branch prediction

```
const int N = 32767;
int[] sorted, unsorted; // random numbers [0..255]
private static int Sum(int[] data)
{
  int sum = 0;
  for (int i = 0; i < N; i++)
    if (data[i] >= 128)
      sum += data[i];
  return sum;
}
```

```
[Benchmark]
public int Sorted()
{
  return Sum(sorted);
}
```

```
[Benchmark]
public int Unsorted()
{
  return Sum(unsorted);
}
```

Branch prediction

```
const int N = 32767;
int[] sorted, unsorted; // random numbers [0..255]
private static int Sum(int[] data)
{
  int sum = 0;
  for (int i = 0; i < N; i++)
    if (data[i] >= 128)
      sum += data[i];
  return sum;
}
```

```
[Benchmark]
public int Sorted()
{
  return Sum(sorted);
}
```

```
[Benchmark]
public int Unsorted()
{
  return Sum(unsorted);
}
```

	Sorted	Unsorted
LegacyJIT-x86	1 попугай	7.4 попугая

Интерфейсы

```
private interface IFoo {
  double Inc(double x);
}
private class Foo1 : IFoo {
  public double Inc(double x) =>
    x + 1;
}
private class Foo2 : IFoo {
  public double Inc(double x) =>
  x + 1;
}
private double Run(IFoo foo) {
  double sum = 0;
  for (int i = 0; i < 1001; i++)
    sum += foo.Inc(0);
  return sum;
}
```

```
[Benchmark]
public double Run11() {
  var bar1 = new Foo1();
  var bar2 = new Foo1();
  return Run(bar1) + Run(bar2);
}

[Benchmark]
public double Run12() {
  var bar1 = new Foo1();
  var bar2 = new Foo2();
  return Run(bar1) + Run(bar2);
}
```

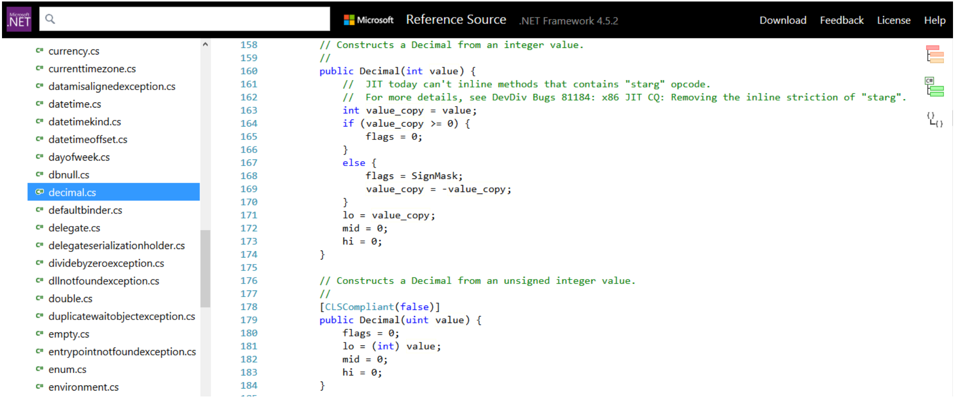
Интерфейсы

```
private interface IFoo {
  double Inc(double x);
}
private class Foo1 : IFoo {
  public double Inc(double x) =>
    x + 1;
}
private class Foo2 : IFoo {
  public double Inc(double x) =>
  x + 1;
}
private double Run(IFoo foo) {
  double sum = 0;
  for (int i = 0; i < 1001; i++)
    sum += foo.Inc(0);
  return sum;
}
```

```
[Benchmark]
public double Run11() {
  var bar1 = new Foo1();
  var bar2 = new Foo1();
  return Run(bar1) + Run(bar2);
}

[Benchmark]
public double Run12() {
  var bar1 = new Foo1();
  var bar2 = new Foo2();
  return Run(bar1) + Run(bar2);
}
```

	Run11	Run12
LegacyJIT-x64	1 попугай	1.25 попугая

История



- currency.cs
- currenttimezone.cs
- datamisalignedexception.cs
- datetime.cs
- datetimekind.cs
- datetimeoffset.cs
- dayofweek.cs
- dbnull.cs
- decimal.cs
- defaultbinder.cs
- delegate.cs
- delegateserializationholder.cs
- dividebyzeroexception.cs
- dllnotfoundexception.cs
- double.cs
- duplicatewaitobjectexception.cs
- empty.cs
- entrypointnotfoundexception.cs
- enum.cs
- environment.cs

```
// Constructs a Decimal from an integer value.
//
public Decimal(int value) {
    //  JIT today can't inline methods that contains "starg" opcode.
    //  For more details, see DevDiv Bugs 81184: x86 JIT CQ: Removing the inline striction of "starg".
    int value_copy = value;
    if (value_copy >= 0) {
        flags = 0;
    }
    else {
        flags = SignMask;
        value_copy = -value_copy;
    }
    lo = value_copy;
    mid = 0;
    hi = 0;
}

// Constructs a Decimal from an unsigned integer value.
//
[CLSCompliant(false)]
public Decimal(uint value) {
    flags = 0;
    lo = (int) value;
    mid = 0;
    hi = 0;
}
```

Inlining

```
// mscorlib/system/decimal.cs,158
// Constructs a Decimal from an integer value.
public Decimal(int value) {
  // JIT today can't inline methods that contains "starg"
  // opcode. For more details, see DevDiv Bugs 81184:
  // x86 JIT CQ: Removing the inline striction of "starg".
  int value_copy = value;
  if (value_copy >= 0) {
    flags = 0;
  }
  else {
    flags = SignMask;
    value_copy = -value_copy;
  }
  lo = value_copy;
  mid = 0;
  hi = 0;
}
```

Inlining

```
[Benchmark]
int Calc() => WithoutStarg(0x11) + WithStarg(0x12);

int WithoutStarg(int value) => value;

int WithStarg(int value)
{
  if (value < 0)
    value = -value;
  return value;
}
```

Inlining

```
[Benchmark]
int Calc() => WithoutStarg(0x11) + WithStarg(0x12);

int WithoutStarg(int value) => value;

int WithStarg(int value)
{
  if (value < 0)
    value = -value;
  return value;
}
```

LegacyJIT-x86	LegacyJIT-x64	RyuJIT-x64
1 попугай	0 попугаев	1 попугай

Как же так?

LegacyJIT-x64

```
; LegacyJIT-x64
mov             ecx,23h
ret
```

Как же так?

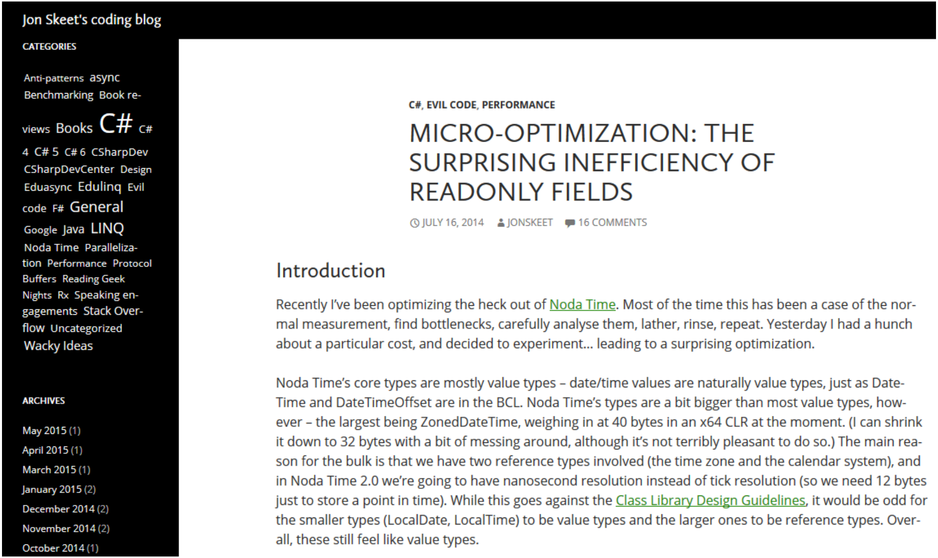
LegacyJIT-x64

```
; LegacyJIT-x64
mov          ecx,23h
ret
```

RyuJIT-x64

```
// Inline expansion aborted due to opcode
// [06] OP_starg.s in method
// Program:WithStarg(int):int:this
```

История



Jon Skeet's coding blog

CATEGORIES

Anti-patterns async Benchmarking Book reviews Books C# C# 4 C# 5 C# 6 CSharpDev CSharpDevCenter Design Eduasync Eduling Evil code F# General Google Java LINQ Noda Time Parallelization Performance Protocol Buffers Reading Geek Nights Rx Speaking engagements Stack Overflow Uncategorized Wacky Ideas

ARCHIVES

May 2015 (1)
April 2015 (1)
March 2015 (1)
January 2015 (2)
December 2014 (2)
November 2014 (2)
October 2014 (1)

C#, EVIL CODE, PERFORMANCE

MICRO-OPTIMIZATION: THE SURPRISING INEFFICIENCY OF READONLY FIELDS

JULY 16, 2014 JONSKEET 16 COMMENTS

Introduction

Recently I've been optimizing the heck out of Noda Time. Most of the time this has been a case of the normal measurement, find bottlenecks, carefully analyse them, lather, rinse, repeat. Yesterday I had a hunch about a particular cost, and decided to experiment… leading to a surprising optimization.

Noda Time's core types are mostly value types – date/time values are naturally value types, just as DateTime and DateTimeOffset are in the BCL. Noda Time's types are a bit bigger than most value types, however – the largest being ZonedDateTime, weighing in at 40 bytes in an x64 CLR at the moment. (I can shrink it down to 32 bytes with a bit of messing around, although it's not terribly pleasant to do so.) The main reason for the bulk is that we have two reference types involved (the time zone and the calendar system), and in Noda Time 2.0 we're going to have nanosecond resolution instead of tick resolution (so we need 12 bytes just to store a point in time). While this goes against the Class Library Design Guidelines, it would be odd for the smaller types (LocalDate, LocalTime) to be value types and the larger ones to be reference types. Overall, these still feel like value types.

Поговорим про Readonly fields

```
public struct Int256
{
    private readonly long bits0, bits1, bits2, bits3;
    public Int256(long bits0, long bits1, long bits2, long bits3)
    {
        this.bits0 = bits0; this.bits1 = bits1;
        this.bits2 = bits2; this.bits3 = bits3;
    }
    public long Bits0 => bits0; public long Bits1 => bits1;
    public long Bits2 => bits2; public long Bits3 => bits3;
}
private Int256 a = new Int256(1L, 5L, 10L, 100L);
private readonly Int256 b = new Int256(1L, 5L, 10L, 100L);
[Benchmark] public long GetValue() =>
    a.Bits0 + a.Bits1 + a.Bits2 + a.Bits3;
[Benchmark] public long GetReadOnlyValue() =>
    b.Bits0 + b.Bits1 + b.Bits2 + b.Bits3;
```

Поговорим про Readonly fields

```
public struct Int256
{
    private readonly long bits0, bits1, bits2, bits3;
    public Int256(long bits0, long bits1, long bits2, long bits3)
    {
        this.bits0 = bits0; this.bits1 = bits1;
        this.bits2 = bits2; this.bits3 = bits3;
    }
    public long Bits0 => bits0; public long Bits1 => bits1;
    public long Bits2 => bits2; public long Bits3 => bits3;
}
private Int256 a = new Int256(1L, 5L, 10L, 100L);
private readonly Int256 b = new Int256(1L, 5L, 10L, 100L);
[Benchmark] public long GetValue() =>
    a.Bits0 + a.Bits1 + a.Bits2 + a.Bits3;
[Benchmark] public long GetReadOnlyValue() =>
    b.Bits0 + b.Bits1 + b.Bits2 + b.Bits3;
```

	LegacyJIT-x64	RyuJIT-x64
GetValue	1 попугай	1 попугай
GetReadOnlyValue	6.2 попугая	7.6 попугая

Как же так?

```
; GetValue
IL_0000: ldarg.0
IL_0001: ldflda valuetype Program::a
IL_0006: call instance int64 Int256::get_Bits0()

; GetReadOnlyValue
IL_0000: ldarg.0
IL_0001: ldfld valuetype Program::b
IL_0006: stloc.0
IL_0007: ldloca.s 0
IL_0009: call instance int64 Int256::get_Bits0()
```

См. также: Jon Skeet, Micro-optimization: the surprising inefficiency of readonly fields

Поговорим про SIMD

```
private struct MyVector
{
    public float X, Y, Z, W;
    public MyVector(float x, float y, float z, float w)
    {
        X = x; Y = y; Z = z; W = w;
    }
    [MethodImpl(MethodImplOptions.AggressiveInlining)]
    public static MyVector operator *(MyVector left, MyVector right)
    {
        return new MyVector(left.X * right.X, left.Y * right.Y,
                            left.Z * right.Z, left.W * right.W);
    }
}
private Vector4    vector1,   vector2,   vector3;
private MyVector myVector1, myVector2, myVector3;
[Benchmark] public void MyMul() => myVector3 = myVector1 * myVector2;
[Benchmark] public void BclMul() => vector3 = vector1 * vector2;
```

Поговорим про SIMD

```
private struct MyVector
{
    public float X, Y, Z, W;
    public MyVector(float x, float y, float z, float w)
    {
        X = x; Y = y; Z = z; W = w;
    }
    [MethodImpl(MethodImplOptions.AggressiveInlining)]
    public static MyVector operator *(MyVector left, MyVector right)
    {
        return new MyVector(left.X * right.X, left.Y * right.Y,
                            left.Z * right.Z, left.W * right.W);
    }
}
private Vector4    vector1,   vector2,   vector3;
private MyVector myVector1, myVector2, myVector3;
[Benchmark] public void MyMul() => myVector3 = myVector1 * myVector2;
[Benchmark] public void BclMul() => vector3 = vector1 * vector2;
```

	LegacyJIT-x64	RyuJIT-x64
MyMul	34 попугая	5 попугаев
BclMul	34 попугая	1 попугай

Как же так?

```
; LegacyJIT-x64

; MyMul, BclMul
; ...
movss     xmm3,dword ptr [rsp+40h]
mulss     xmm3,dword ptr [rsp+30h]
movss     xmm2,dword ptr [rsp+44h]
mulss     xmm2,dword ptr [rsp+34h]
movss     xmm1,dword ptr [rsp+48h]
mulss     xmm1,dword ptr [rsp+38h]
movss     xmm0,dword ptr [rsp+4Ch]
mulss     xmm0,dword ptr [rsp+3Ch]
xor       eax,eax
mov       qword ptr [rsp],rax
mov       qword ptr [rsp+8],rax
lea       rax,[rsp]
movss     dword ptr [rax],xmm3
movss     dword ptr [rax+4],xmm2
movss     dword ptr [rax+8],xmm1
movss     dword ptr [rax+0Ch],xmm0
; ...
```

```
; RyuJIT-x64

; MyMul
; ...
vmulss    xmm0,xmm0,xmm4
vmulss    xmm1,xmm1,xmm5
vmulss    xmm2,xmm2,xmm6
vmulss    xmm3,xmm3,xmm7
; ...

; BclMul
vmovupd   xmm0,xmmword ptr [rcx+8]
vmovupd   xmm1,xmmword ptr [rcx+18h]
vmulps    xmm0,xmm0,xmm1
vmovupd   xmmword ptr [rcx+28h],xmm0
```

Думаем про ASM

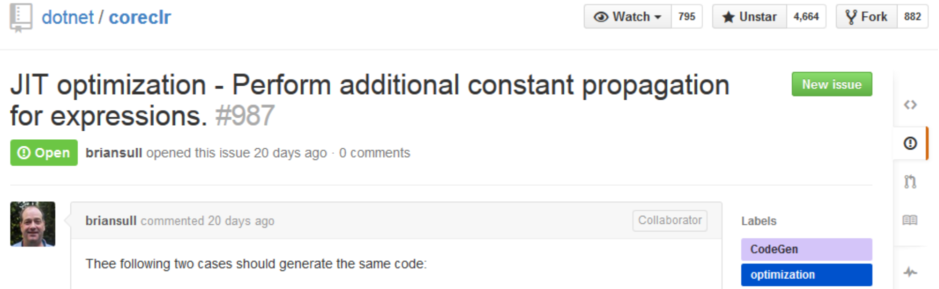
```
double x = /* ... */;
double a = x + 1;
double b = x * 2;
double c = Math.Sqrt(x);
```

Думаем про ASM

```
double x = /* ... */;
double a = x + 1;
double b = x * 2;
double c = Math.Sqrt(x);
```

	LegacyJIT-x86	LegacyJIT-x64	RyuJIT-x64
	(x87 FPU)	*(SSE2)*	*(AVX)*
x + 1	faddp	addsd	vaddsd
x * 2	fmul	mulsd	vmulsd
Sqrt(X)	fsqrt	sqrtsd	vsqrtsd

История



dotnet / coreclr

JIT optimization - Perform additional constant propagation for expressions. #987

briansull opened this issue 20 days ago · 0 comments

briansull commented 20 days ago

Thee following two cases should generate the same code:

Задачка

```
double Sqrt13() =>
    Math.Sqrt(1) + Math.Sqrt(2) + Math.Sqrt(3) + /* ... */
    + Math.Sqrt(13);
```

VS

```
double Sqrt14() =>
    Math.Sqrt(1) + Math.Sqrt(2) + Math.Sqrt(3) + /* ... */
    + Math.Sqrt(13) + Math.Sqrt(14);
```

Задачка

```
double Sqrt13() =>
    Math.Sqrt(1) + Math.Sqrt(2) + Math.Sqrt(3) + /* ... */
    + Math.Sqrt(13);
```

VS

```
double Sqrt14() =>
    Math.Sqrt(1) + Math.Sqrt(2) + Math.Sqrt(3) + /* ... */
    + Math.Sqrt(13) + Math.Sqrt(14);
```

	RyuJIT-x64[1]
Sqrt13	40 попугаев
Sqrt14	1 попугай

[1] RyuJIT RC

Как же так?

RyuJIT-x64, Sqrt13

```
vsqrtsd      xmm0,xmm0,mmword ptr [7FF94F9E4D28h]
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D30h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D38h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D40h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D48h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D50h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D58h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D60h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D68h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D70h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D78h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D80h]
vaddsd       xmm0,xmm0,xmm1
vsqrtsd      xmm1,xmm0,mmword ptr [7FF94F9E4D88h]
vaddsd       xmm0,xmm0,xmm1
ret
```

Как же так?

RyuJIT-x64, Sqrt14

```
vmovsd      xmm0,qword ptr [7FF94F9C4C80h]
ret
```

Как же так?

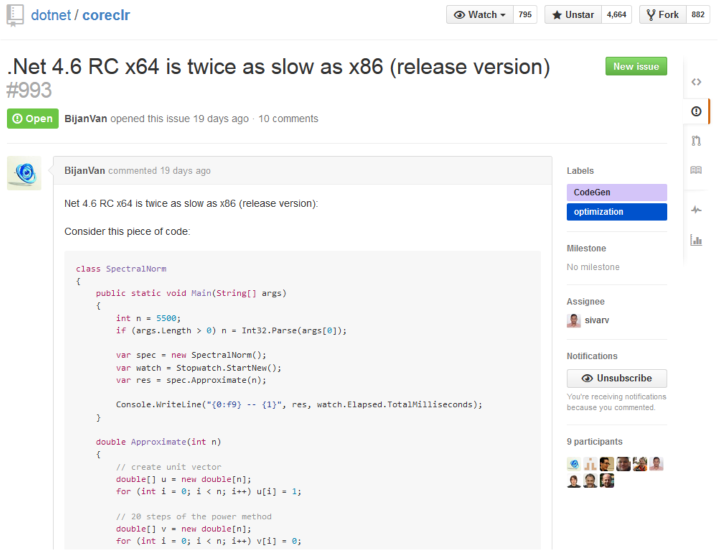
Большое дерево выражения

```
*  stmtExpr  void  (top level) (IL 0x000...  ???)
|     /--*  mathFN    double sqrt
|     |  \--*  dconst    double 13.000000000000000
|  /--*  +         double
|  |  |  /--*  mathFN    double sqrt
|  |  |  |  \--*  dconst    double 12.000000000000000
|  |  \--*  +         double
|  |     |  /--*  mathFN    double sqrt
|  |     |  |  \--*  dconst    double 11.000000000000000
|  |     \--*  +         double
|  |        |  /--*  mathFN    double sqrt
|  |        |  |  \--*  dconst    double 10.000000000000000
|  |        \--*  +         double
|  |           |  /--*  mathFN    double sqrt
|  |           |  |  \--*  dconst    double 9.0000000000000000
|  |           \--*  +         double
|  |              |  /--*  mathFN    double sqrt
|  |              |  |  \--*  dconst    double 8.0000000000000000
|  |              \--*  +         double
|  |                 |  /--*  mathFN    double sqrt
|  |                 |  |  \--*  dconst    double 7.0000000000000000
|  |                 \--*  +         double
|  |                    |  /--*  mathFN    double sqrt
|  |                    |  |  \--*  dconst    double 6.0000000000000000
|  |                    \--*  +         double
|  |                       |  /--*  mathFN    double sqrt
|  |                       |  |  \--*  dconst    double 5.0000000000000000
// ...
```

Constant folding в действии

```
N001 [000001]   dconst    1.0000000000000000 => $c0 {DblCns[1.000000]}
N002 [000002]   mathFN    => $c0 {DblCns[1.000000]}
N003 [000003]   dconst    2.0000000000000000 => $c1 {DblCns[2.000000]}
N004 [000004]   mathFN    => $c2 {DblCns[1.414214]}
N005 [000005]   +         => $c3 {DblCns[2.414214]}
N006 [000006]   dconst    3.0000000000000000 => $c4 {DblCns[3.000000]}
N007 [000007]   mathFN    => $c5 {DblCns[1.732051]}
N008 [000008]   +         => $c6 {DblCns[4.146264]}
N009 [000009]   dconst    4.0000000000000000 => $c7 {DblCns[4.000000]}
N010 [000010]   mathFN    => $c1 {DblCns[2.000000]}
N011 [000011]   +         => $c8 {DblCns[6.146264]}
N012 [000012]   dconst    5.0000000000000000 => $c9 {DblCns[5.000000]}
N013 [000013]   mathFN    => $ca {DblCns[2.236068]}
N014 [000014]   +         => $cb {DblCns[8.382332]}
N015 [000015]   dconst    6.0000000000000000 => $cc {DblCns[6.000000]}
N016 [000016]   mathFN    => $cd {DblCns[2.449490]}
N017 [000017]   +         => $ce {DblCns[10.831822]}
N018 [000018]   dconst    7.0000000000000000 => $cf {DblCns[7.000000]}
N019 [000019]   mathFN    => $d0 {DblCns[2.645751]}
N020 [000020]   +         => $d1 {DblCns[13.477573]}
...
```

История



dotnet / coreclr

.Net 4.6 RC x64 is twice as slow as x86 (release version) #993

BijanVan opened this issue 19 days ago · 10 comments

BijanVan commented 19 days ago

Net 4.6 RC x64 is twice as slow as x86 (release version):

Consider this piece of code:

```
class SpectralNorm
{
    public static void Main(String[] args)
    {
        int n = 5500;
        if (args.Length > 0) n = Int32.Parse(args[0]);

        var spec = new SpectralNorm();
        var watch = Stopwatch.StartNew();
        var res = spec.Approximate(n);

        Console.WriteLine("{0:f9} -- {1}", res, watch.Elapsed.TotalMilliseconds);
    }

    double Approximate(int n)
    {
        // create unit vector
        double[] u = new double[n];
        for (int i = 0; i < n; i++) u[i] = 1;

        // 20 steps of the power method
        double[] v = new double[n];
        for (int i = 0; i < n; i++) v[i] = 0;
```

Labels

CodeGen

optimization

Milestone

No milestone

Assignee

sivarv

Notifications

You're receiving notifications because you commented.

9 participants

Задачка

```
private double[] x = new double[11];

[Benchmark]
public double Calc()
{
    double sum = 0.0;
    for (int i = 1; i < x.Length; i++)
        sum += 1.0 / (i * i) * x[i];
    return sum;
}
```

Задачка

```
private double[] x = new double[11];

[Benchmark]
public double Calc()
{
    double sum = 0.0;
    for (int i = 1; i < x.Length; i++)
        sum += 1.0 / (i * i) * x[i];
    return sum;
}
```

	LegacyJIT-x64	RyuJIT-x64[1]
Calc	1 попугай	2 попугая

[1] RyuJIT RC

Как же так?

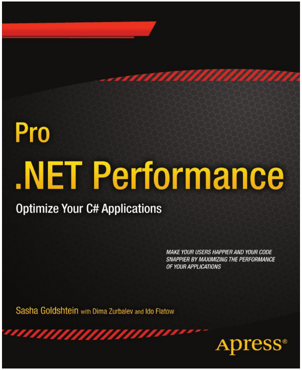
```
; LegacyJIT-x64
; eax = i
mov         eax,r8d
; eax = i*i
imul        eax,r8d
; xmm0=i*i
cvtsi2sd    xmm0,eax
; xmm1=1
movsd       xmm1,
    mmword ptr [7FF9141145E0h]
; xmm1=1/(i*i)
divsd       xmm1,xmm0


; xmm1=1/(i*i)*x[i]
mulsd       xmm1,
    mmword ptr [rdx+r9+10h]
; xmm1 = sum + 1/(i*i)*x[i]
addsd       xmm1,xmm2
; sum = sum + 1/(i*i)*x[i]
movapd      xmm2,xmm1
```

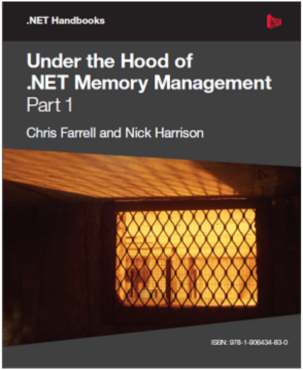
```
; RyuJIT-x64
; r8d = i
mov         r8d,eax
; r8d = i*i
imul        r8d,eax
; xmm1=i*i
vcvtsi2sd   xmm1,xmm1,r8d
; xmm2=1
vmovsd      xmm2,
    qword ptr [7FF9140E4398h]
; xmm2=1/(i*i)
vdivsd      xmm2,xmm2,xmm1
mov         r8,rdx
movsxd      r9,eax
; xmm1 = 1/(i*i)
vmovaps     xmm1,xmm2
; xmm1 = 1/(i*i)*x[i]
vmulsd      xmm1,xmm1,
    mmword ptr [r8+r9*8+10h]
; sum += 1/(i*i)*x[i]
vaddsd      xmm0,xmm0,xmm1
```

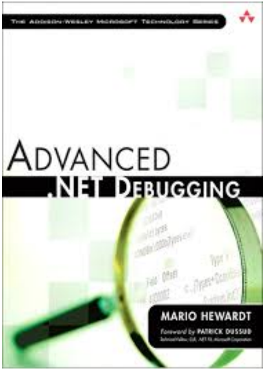
См. также: https://github.com/dotnet/coreclr/issues/993

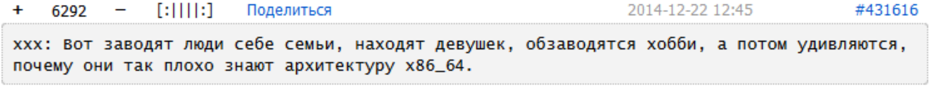
Методическая литература

Для успешных микрооптимизаций нужно очень много знать:


Pro
.NET Performance
Optimize Your C# Applications
MAKE YOUR USERS HAPPIER AND YOUR CODE SNAPPIER BY MAXIMIZING THE PERFORMANCE OF YOUR APPLICATIONS
Sasha Goldshtein with Dima Zurbalev and Ido Flatow
Apress®


.NET Handbooks
Under the Hood of .NET Memory Management
Part 1
Chris Farrell and Nick Harrison
ISBN: 978-1-906434-83-0


The Addison-Wesley Microsoft Technology Series
Advanced .NET Debugging
Mario Hewardt
Foreword by Patrick Dussud



xxx: Вот заводят люди себе семьи, находят девушек, обзаводятся хобби, а потом удивляются, почему они так плохо знают архитектуру x86_64.

Вопросы?

Андрей Акиньшин
http://aakinshin.net
https://github.com/AndreyAkinshin
https://twitter.com/andrey_akinshin
andrey.akinshin@gmail.com