.NET 6.0:
What's new in JIT compiler

@EgorBo
Senior Software Engineer at Microsoft

Profile-Guided Optimization (PGO)

- Re-orders basic blocks (keeps hot ones together)
- De-virtualizes virtual calls via GDV (Guarded Devirtualization)
- Inliner relies on PGO data and can be very aggressive for hot paths
- Loop Clonning, Inlined Casts, etc. aren't applied in cold blocks

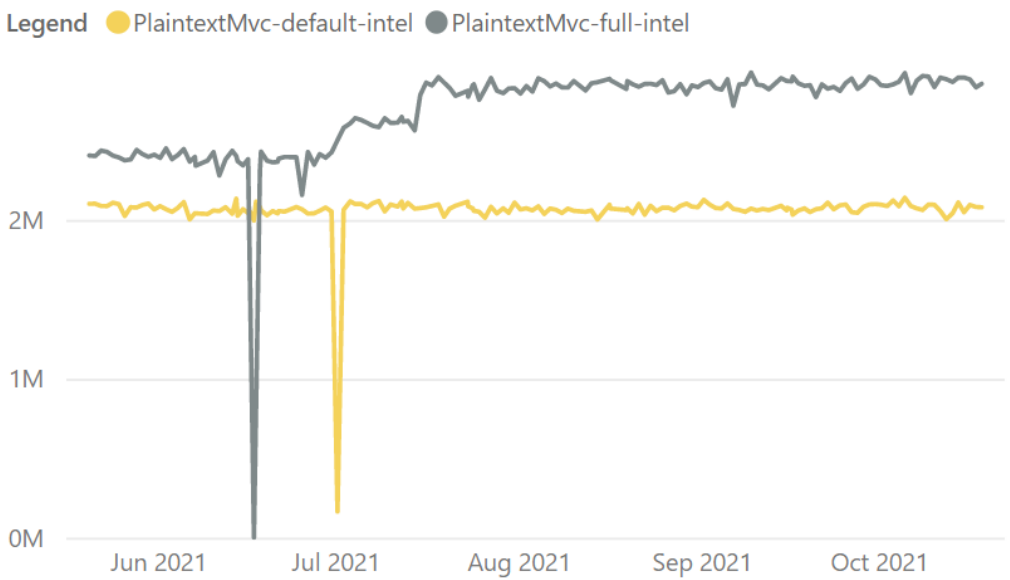
PGO: Instrumentation

```
static void DoWork(IDisposable d)
{
    if (d == null)
    {
        Console.WriteLine(":(");
    }
    else
    {
        d.Dispose();
    }
}
```

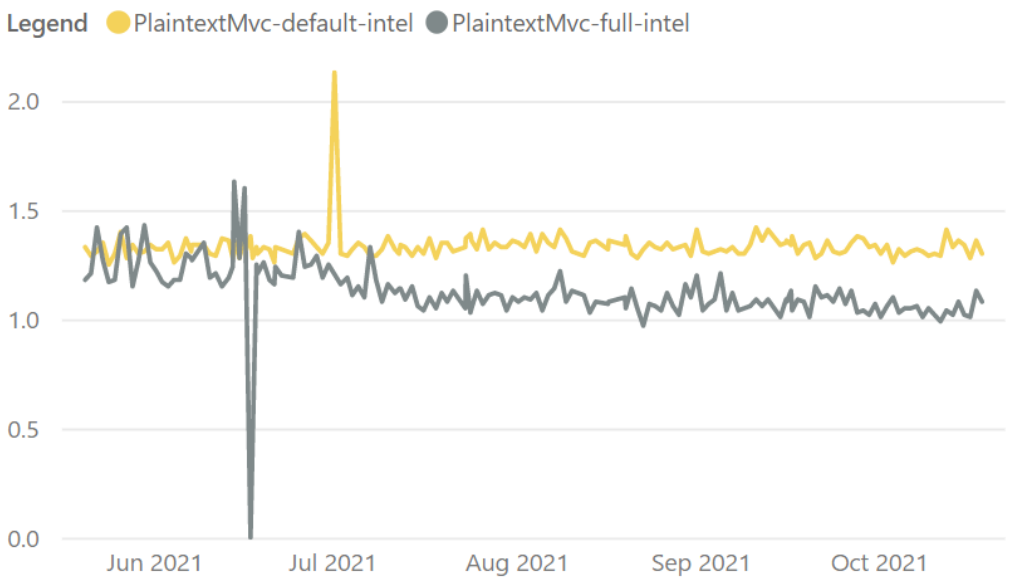
PGO: Instrumentation

```
static void DoWork(IDisposable d)
{
    if (d == null)
    {
        __DoWork_ILOffset2_counter++;
        Console.WriteLine(":(");
    }
    else
    {
        ___DoWork_ILOffset4_counter++;
        ___DoWork_ILOffset4_ClassProfile(d);
        d.Dispose();
    }
}
```

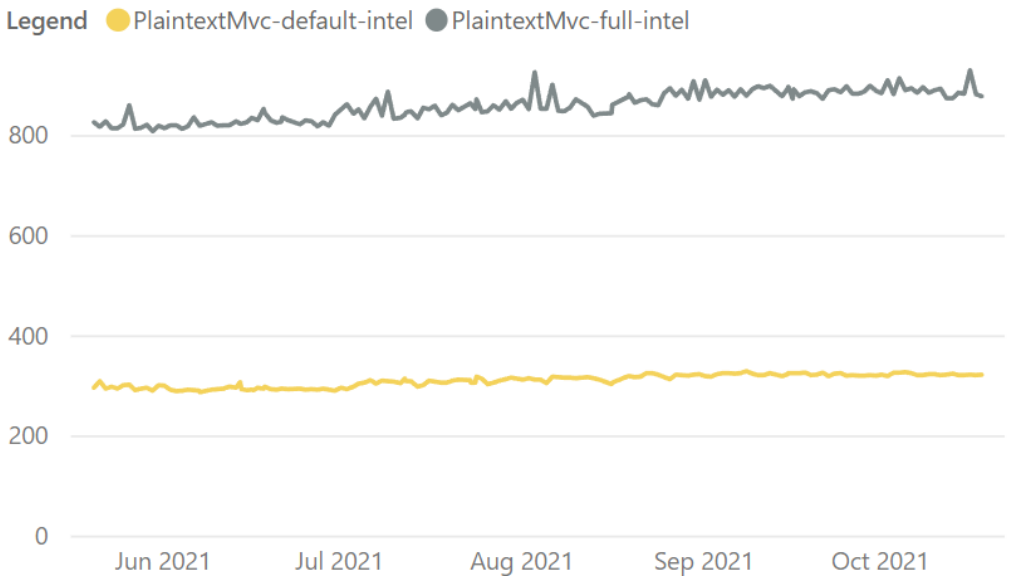
Requests Per Second



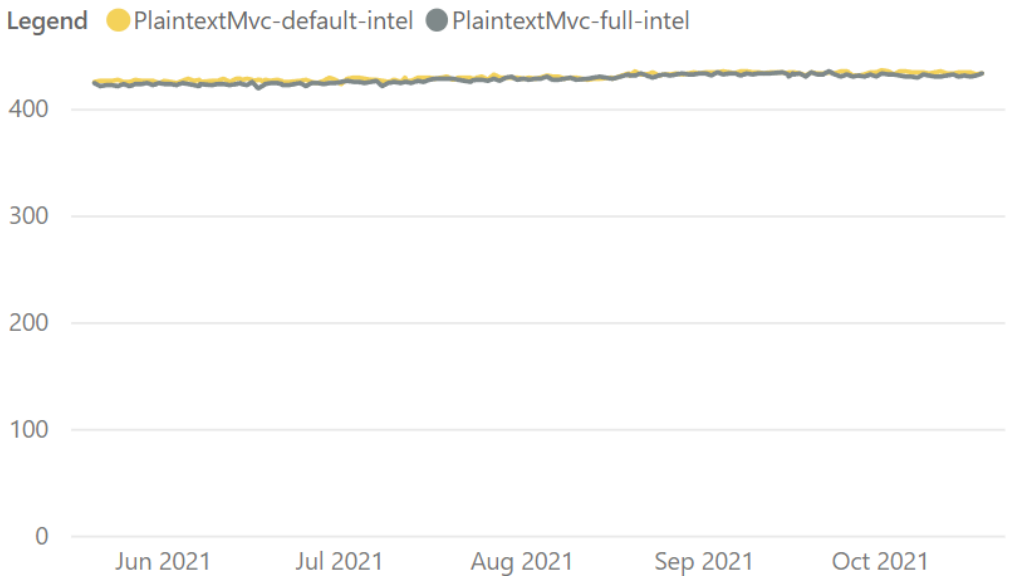
Latency - Average (ms)



Time to First Response (ms)



Memory (MB)



How to use Dynamic PGO

- **Dynamic PGO**
  - DOTNET_TC_QuickJitForLoops=1
  - DOTNET_TieredPGO=1

- **Dynamic PGO without prejitted code (we call it 'FullPGO')**
  - DOTNET_ReadyToRun=0
  - DOTNET_TC_QuickJitForLoops=1
  - DOTNET_TieredPGO=1

PGO: devirtualization

```
bool WriteAndFlush(Stream stream)
{
    stream.WriteByte(1);
    stream.Flush();
    return true;
}
```

Codegen for default mode

```
bool WriteAndFlush(Stream stream)
{
    stream.WriteByte(1);
    stream.Flush();
    return true;
}
```

```
; Method Program:WriteAndFlush(System.IO.Stream):bool:this
G_M46608_IG01:                ;; offset=0000H
       57                     push     rdi
       56                     push     rsi
       4883EC28               sub      rsp, 40
       488BF2                 mov      rsi, rdx
                            ;; bbWeight=1    PerfScore 2.50

G_M46608_IG02:                ;; offset=0009H
       488BCE                 mov      rcx, rsi
       BA01000000             mov      edx, 1
       488B3E                 mov      rdi, qword ptr [rsi]
       488B4770               mov      rax, qword ptr [rdi+112]
       FF10                   call     qword ptr [rax]System.IO.Stream:WriteByte(ubyte):this
       488BCE                 mov      rcx, rsi
       488B4758               mov      rax, qword ptr [rdi+88]
       FF5030                 call     qword ptr [rax+48]System.IO.Stream:Flush():this
       B801000000             mov      eax, 1
                            ;; bbWeight=1    PerfScore 13.00

G_M46608_IG03:                ;; offset=0029H
       4883C428               add      rsp, 40
       5E                     pop      rsi
       5F                     pop      rdi
       C3                     ret
                            ;; bbWeight=1    PerfScore 2.25
; Total bytes of code: 48
```

Tier0 with instrumentation

```
bool WriteAndFlush(Stream stream)
{
    stream.WriteByte(1);
    stream.Flush();
    return true;
}
```

```
mov     rcx, gword ptr [rbp+18H]
mov     gword ptr [rbp-08H], rcx
mov     rcx, gword ptr [rbp-08H]
mov     rdx, 0x7FFE86F8C270
call    CORINFO_HELP_CLASSPROFILE32
mov     rcx, gword ptr [rbp-08H]
mov     gword ptr [rbp-18H], rcx
mov     rcx, gword ptr [rbp-18H]
mov     edx, 1
mov     rax, gword ptr [rbp-18H]
mov     rax, qword ptr [rax]
mov     rax, qword ptr [rax+112]
call    qword ptr [rax]System.IO.Stream:WriteByte(ubyte):this
```

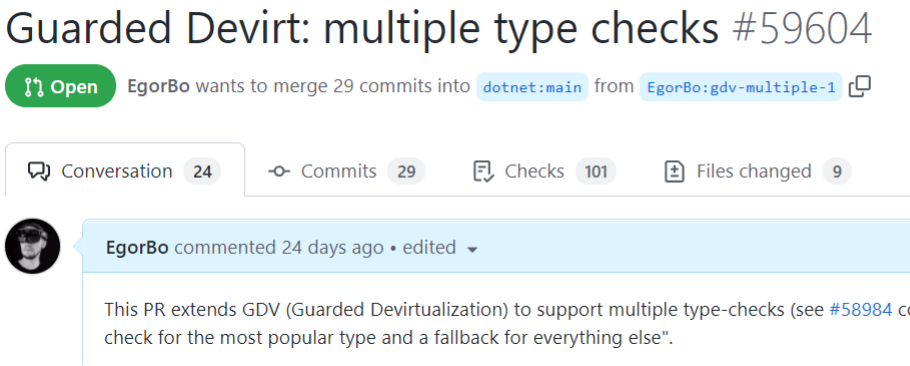
Guarded devirtualization (GDV)

```
bool WriteAndFlush(Stream stream)
{
    if (stream is MemoryStream)
        ((MemoryStream)stream).WriteByte(1); // can be inlined now!
    else
        stream.WriteByte(1);

    if (stream is MemoryStream)
        ((MemoryStream)stream).Flush(); // can be inlined now!
    else
        stream.Flush();
    return true;
}
```

GDV: Chaining

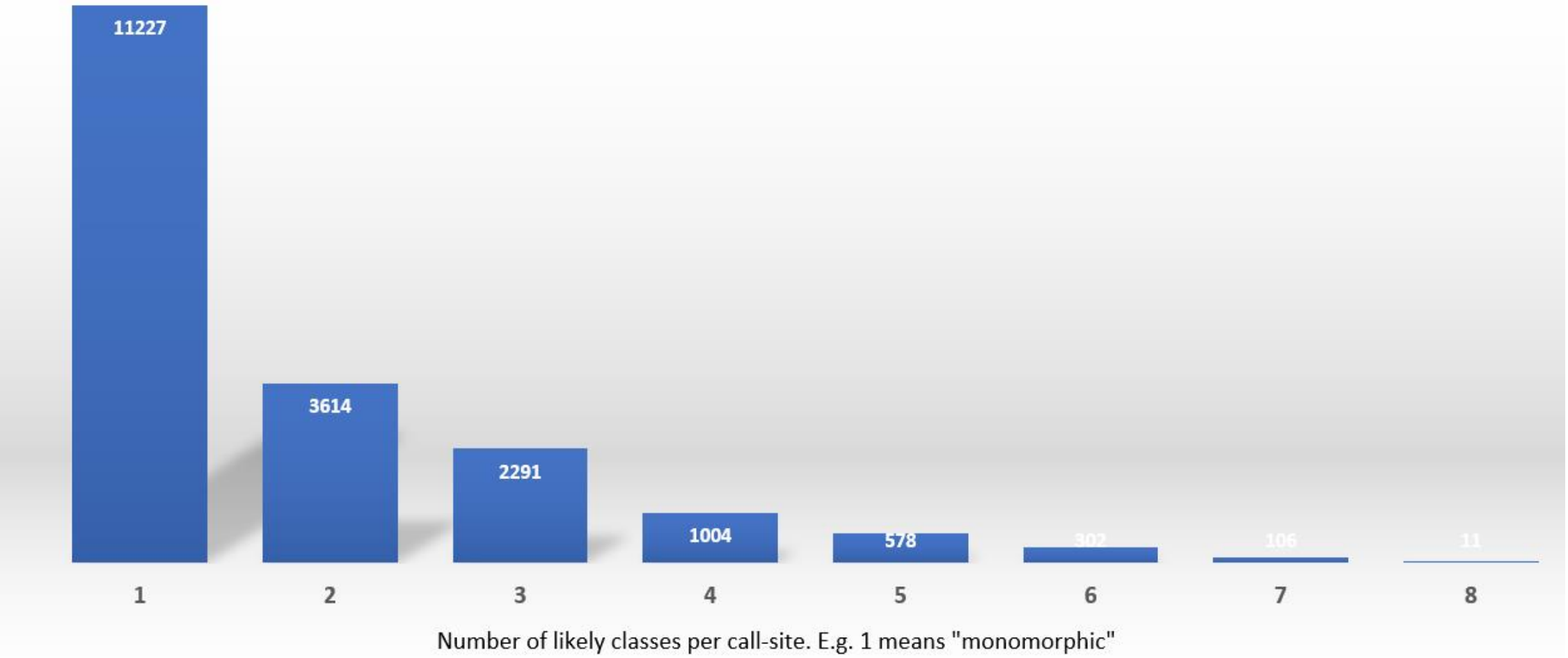
```
void WriteAndFlush(Stream stream)
{
    if (stream is MemoryStream)
    {
        ((MemoryStream)stream).WriteByte(1); // can be inlined now!
        ((MemoryStream)stream).Flush();      // can be inlined now!
    }
    else
    {
        stream.WriteByte(1); // indirect call (fallback)
        stream.Flush();      // indirect call (fallback)
    }
    return true;
}
```

Next steps: multiple guesses




```
bool WriteAndFlush(Stream stream)
{
    if (stream is MemoryStream)
        ((MemoryStream)stream).WriteByte(1); // can be inlined now!
    else if (stream is FileStream)
         ((FileStream)stream).WriteByte(1);  // can be inlined now!
    else
        stream.WriteByte(1);

    if (stream is MemoryStream)
        ((MemoryStream)stream).Flush(); // can be inlined now!
    else if (stream is FileStream)
        ((FileStream)stream).Flush();   // can be inlined now!
    else
        stream.Flush();
    return true;
}
```

Virtual calls in AvaloniaILSpy app



Live-coding:
PGO & GDV

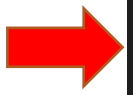
Improvements in Inliner

- Precise(r) IL scan
- Recognizes intrinsics in call opcodes (only if promoted to tier1 naturally)
- 25 new observations/heuristics
- Relies on PGO data if it’s around
- Extended IL and BB limits

```
TimeSpan TimeoutSpan => TimeSpan.FromSeconds(5);
```

.NET 5.0:

```
; Method Program:GetTimeoutSpan():System.TimeSpan:this
G_M11456_IG01:              ;; offset=0000H
       C5F877               vzeroupper
                         ;; bbWeight=1    PerfScore 1.00

G_M11456_IG02:              ;; offset=0003H
       C5FB10050D000000     vmovsd   xmm0, qword ptr [reloc @RWD00]
                         ;; bbWeight=1    PerfScore 3.00

G_M11456_IG03:              ;; offset=000BH
       E9D894FDFF           jmp      System.TimeSpan:IntervalFromDoubleTicks(double):System.TimeSpan
                         ;; bbWeight=1    PerfScore 2.00
RWD00   dq  4187D78400000000h   ;     50000000
; Total bytes of code: 16
```



```
private static TimeSpan IntervalFromDoubleTicks(double ticks)
{
    if ((ticks > long.MaxValue) || (ticks < long.MinValue) || double.IsNaN(ticks
        ThrowHelper.ThrowOverflowException_TimeSpanTooLong();
    if (ticks == long.MaxValue)
        return TimeSpan.MaxValue;
    return new TimeSpan((long)ticks);
}
```

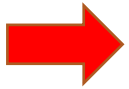
.NET 6.0:

```
; Method Program:GetTimeoutSpan():System.TimeSpan:this
G_M11456_IG01:              ;; offset=0000H
                         ;; bbWeight=1    PerfScore 0.00

G_M11456_IG02:              ;; offset=0000H
       B880F0FA02           mov      eax, 0x2FAF080
                         ;; bbWeight=1    PerfScore 0.25

G_M11456_IG03:              ;; offset=0005H
       C3                   ret
                         ;; bbWeight=1    PerfScore 1.00
; Total bytes of code: 6
```

```
bool IsEgor(string str) => str == "Egor";

bool Test() => IsEgor("Egor");
```

.NET 5.0:

```
; Method Program:Test():bool:this
G_M57035_IG01:                ;; offset=0000H
                        ;; bbWeight=1    PerfScore 0.00

G_M57035_IG02:                ;; offset=0000H
       48B98854CABD35020000 mov      rcx, 0x235BDCA5488      ; "Egor"
       488B11               mov      rdx, gword ptr [rcx]
       488BCA               mov      rcx, rdx
                        ;; bbWeight=1    PerfScore 2.50

G_M57035_IG03:                ;; offset=0010H
       E9F35DFDFF           jmp      System.String:Equals(System.String,System.String):bool
                        ;; bbWeight=1    PerfScore 2.00
; Total bytes of code: 21
```



```
public static bool Equals(string? a, string? b)
{
    if (object.ReferenceEquals(a, b))
    {
        return true;
    }

    if (a is null || b is null || a.Length != b.Length)
    {
        return false;
    }

    return EqualsHelper(a, b);
}
```

.NET 6.0:

```
; Method Program:Test():bool:this
G_M57035_IG01:                ;; offset=0000H
                        ;; bbWeight=1    PerfScore 0.00

G_M57035_IG02:                ;; offset=0000H
       B801000000           mov      eax, 1
                        ;; bbWeight=1    PerfScore 0.25

G_M57035_IG03:                ;; offset=0005H
       C3                   ret
                        ;; bbWeight=1    PerfScore 1.00
; Total bytes of code: 6
```

```
int Callee<T>(IDisposable disp, uint x)
{
    disp.Dispose();

    if (typeof(T) == typeof(int))
        return 1;
    if (typeof(T) == typeof(uint))
        return 2;
    if (typeof(T) == typeof(double))
        return 3;

    if ((x * 10) == BitOperations.PopCount(x))
    {
        return 5;
    }

    return (int)Math.Pow((int)x, 2);
}



void Caller(Program p) => Callee<string>(p, 42);
```

```
ldtoken !!T
call System.Type::GetTypeFromHandle
ldtoken [System.Private.CoreLib]System.Int32
call System.Type::GetTypeFromHandle
call System.Type::op_Equality
brfalse.s IL_001e
```

```
int Callee<T>(IDisposable disp, uint x)
{
    disp.Dispose();

    if (typeof(T) == typeof(int))
        return 1;
    if (typeof(T) == typeof(uint))
        return 2;
    if (typeof(T) == typeof(double))
        return 3;

    if ((x * 10) == BitOperations.PopCount(x))
    {
        return 5;
    }

    return (int)Math.Pow((int)x, 2);
}



void Caller(Program p) => Callee<string>(p, 42);
```

```
int Callee<T>(IDisposable disp, uint x)
{
    disp.Dispose();

    if (typeof(T) == typeof(int))
        return 1;
    if (typeof(T) == typeof(uint))
        return 2;
    if (typeof(T) == typeof(double))
        return 3;

    if ((x * 10) == BitOperations.PopCount(x))
    {
        return 5;
    }

    return (int)Math.Pow((int)x, 2);
}



void Caller(Program p) => Callee<string>(p, 42);
```

```
int Callee<T>(IDisposable disp, uint x)
{
    disp.Dispose();

    if (typeof(T) == typeof(int))
        return 1;
    if (typeof(T) == typeof(uint))
        return 2;
    if (typeof(T) == typeof(double))
        return 3;

    if ((x * 10) == BitOperations.PopCount(x))
    {
        return 5;
    }

    return (int)Math.Pow((int)x, 2);
}



void Caller(Program p) => Callee<string>(p, 42);
```

```
int Callee<T>(IDisposable disp, uint x)
{
    disp.Dispose();

    if (typeof(T) == typeof(int))
        return 1;
    if (typeof(T) == typeof(uint))
        return 2;
    if (typeof(T) == typeof(double))
        return 3;

    if ((x * 10) == BitOperations.PopCount(x))
    {
        return 5;
    }

    return (int)Math.Pow((int)x, 2);
}
```

```
1662 Inline candidate is generic and caller is not.  Multiplier increased to 2.
1663 Inline candidate has 4 foldable branches.  Multiplier increased to 9.
1664 Inline has 11 foldable intrinsics.  Multiplier increased to 21.
1665 Inline has 1 foldable binary expressions.  Multiplier increased to 23.
1666 Inline has 3 foldable unary expressions.  Multiplier increased to 26.
1667 Inline candidate has an arg that feeds a constant test.  Multiplier increased to 27.
1668 Inline candidate callsite is boring.  Multiplier increased to 28.3.
1669 calleeNativeSizeEstimate=1948
1670 callsiteNativeSizeEstimate=115
1671 benefit multiplier=28.3
1672 threshold=3254
1673 Native estimate for function size is within threshold for inlining 194.8 <= 325.4 (multiplier = 28.3)
```

```
void Caller(Program p) => Callee<string>(p, 42);
```

platform plaintext

Jit mode	Base RPS	PR RPS	diff	Base mean.lat	PR mean.lat	diff	Base P50	PR P50	diff	Base P75	PR P75	diff	Base P90	PR P90	diff	Base P99	PR P99	diff	Base first.req	PR first.req	diff	Base startup	PR startup	diff	Base work.set	PR work.set	diff	Base priv.mem	PR priv.mem	diff	Base cpu	PR cpu	diff
default	11151852	11519171	3.29%	1.38	1.22	-11.59%	0.75	0.72	-4.11%	1.15	1.09	-5.22%	1.98	1.88	-5.05%	15.64	14.94	-4.48%	59.00	61.00	3.39%	144.00	142.00	-1.39%	120.00	122.00	1.67%	702.00	705.00	0.43%	99.00	99.00	0.00%
default	11223432	11581214	3.19%	1.41	1.23	-12.77%	0.75	0.72	-3.21%	1.15	1.10	-4.35%	2.11	1.92	-9.00%	15.64	16.65	6.46%	59.00	61.00	3.39%	139.00	144.00	3.60%	120.00	121.00	0.83%	702.00	703.00	0.14%	99.00	99.00	0.00%
fullpgo	11423514	11757517	2.92%	1.47	1.35	-8.16%	0.74	0.71	-3.38%	1.16	1.10	-5.17%	2.44	2.30	-5.74%	22.12	30.57	38.20%	74.00	73.00	-1.35%	469.00	469.00	0.00%	118.00	118.00	0.00%	707.00	707.00	0.00%	99.00	99.00	0.00%
fullpgo	11428691	11856905	3.75%	1.31	1.33	1.53%	0.73	0.70	-4.39%	1.11	1.07	-3.60%	1.97	2.15	9.14%	14.68	15.79	7.56%	72.00	73.00	1.39%	465.00	472.00	1.51%	119.00	119.00	0.00%	707.00	707.00	0.00%	99.00	99.00	0.00%

platform caching

Jit mode	Base RPS	PR RPS	diff	Base mean.lat	PR mean.lat	diff	Base P50	PR P50	diff	Base P75	PR P75	diff	Base P90	PR P90	diff	Base P99	PR P99	diff	Base first.req	PR first.req	diff	Base startup	PR startup	diff	Base work.set	PR work.set	diff	Base priv.mem	PR priv.mem	diff	Base cpu	PR cpu	diff
default	259027	265956	2.68%	1.10	1.07	-2.73%	0.97	0.94	-3.09%	1.07	1.04	-2.80%	1.20	1.17	-2.50%	6.35	6.16	-2.99%	64.00	65.00	1.56%	1481.00	1431.00	-3.38%	427.00	429.00	0.47%	1550.00	1551.00	0.06%	98.00	98.00	0.00%
default	258040	264816	2.63%	1.12	1.07	-4.46%	0.97	0.94	-3.09%	1.08	1.05	-2.78%	1.22	1.18	-3.28%	7.24	5.72	-20.99%	64.00	65.00	1.56%	1589.00	1603.00	0.88%	426.00	428.00	0.47%	998.00	1551.00	55.41%	98.00	97.00	-1.02%
fullpgo	267517	285183	6.60%	1.07	1.02	-4.67%	0.94	0.88	-6.38%	1.03	0.97	-5.83%	1.16	1.09	-6.03%	6.58	6.86	4.26%	77.00	79.00	2.60%	2141.00	2095.00	-2.15%	424.00	423.00	-0.24%	1555.00	1554.00	-0.06%	98.00	97.00	-1.02%
fullpgo	265961	286785	7.83%	1.09	1.02	-6.42%	0.94	0.87	-7.45%	1.04	0.97	-6.73%	1.17	1.09	-6.84%	7.18	6.92	-3.62%	77.00	77.00	0.00%	2172.00	1966.00	-9.48%	424.00	424.00	0.00%	1556.00	1555.00	-0.06%	98.00	97.00	-1.02%

database fortunes_ef

Jit mode	Base RPS	PR RPS	diff	Base mean.lat	PR mean.lat	diff	Base P50	PR P50	diff	Base P75	PR P75	diff	Base P90	PR P90	diff	Base P99	PR P99	diff	Base first.req	PR first.req	diff	Base startup	PR startup	diff	Base work.set	PR work.set	diff	Base priv.mem	PR priv.mem	diff	Base cpu	PR cpu	diff
default	269570	276845	2.70%	1.04	1.00	-3.85%	0.84	0.82	-2.26%	1.23	1.20	-2.44%	1.60	1.55	-3.13%	4.91	4.47	-8.96%	972.00	1010.00	3.91%	191.00	192.00	0.52%	467.00	469.00	0.43%	1584.00	1588.00	0.25%	96.00	97.00	1.04%
default	265263	265324	0.02%	1.05	1.06	0.95%	0.86	0.85	-1.16%	1.26	1.25	-0.79%	1.62	1.64	1.23%	4.56	4.97	8.99%	972.00	1002.00	3.09%	191.00	191.00	0.00%	465.00	472.00	1.51%	1593.00	1589.00	-0.25%	97.00	97.00	0.00%
fullpgo	281182	309680	10.13%	1.00	0.91	-9.00%	0.81	0.73	-9.77%	1.18	1.06	-10.17%	1.53	1.38	-9.80%	4.44	4.38	-1.35%	1085.00	1082.00	-0.28%	546.00	547.00	0.18%	487.00	462.00	-5.13%	1589.00	1591.00	0.13%	96.00	96.00	0.00%
fullpgo	286530	310580	8.39%	0.98	0.91	-7.14%	0.79	0.73	-8.19%	1.16	1.06	-8.62%	1.49	1.37	-8.05%	4.66	4.33	-7.08%	1090.00	1086.00	-0.37%	550.00	557.00	1.27%	482.00	467.00	-3.11%	1592.00	1589.00	-0.19%	96.00	96.00	0.00%

database single_query_dapper

Jit mode	Base RPS	PR RPS	diff	Base mean.lat	PR mean.lat	diff	Base P50	PR P50	diff	Base P75	PR P75	diff	Base P90	PR P90	diff	Base P99	PR P99	diff	Base first.req	PR first.req	diff	Base startup	PR startup	diff	Base work.set	PR work.set	diff	Base priv.mem	PR priv.mem	diff	Base cpu	PR cpu	diff
default	359137	369681	2.94%	0.78	0.75	-3.66%	0.67	0.65	-2.97%	0.83	0.81	-2.65%	1.01	0.99	-1.98%	3.63	3.37	-7.16%	490.00	510.00	4.08%	190.00	193.00	1.58%	450.00	454.00	0.89%	1563.00	1570.00	0.45%	97.00	96.00	-1.03%
default	353405	366763	3.78%	0.79	0.75	-4.75%	0.69	0.66	-3.35%	0.84	0.81	-3.45%	1.03	0.98	-4.85%	3.40	3.16	-7.06%	504.00	508.00	0.79%	196.00	191.00	-2.55%	455.00	456.00	0.22%	1562.00	1571.00	0.58%	96.00	96.00	0.00%
fullpgo	384779	407757	5.97%	0.74	0.70	-4.56%	0.63	0.59	-5.57%	0.77	0.73	-5.57%	0.95	0.90	-5.26%	3.63	3.50	-3.58%	596.00	591.00	-0.84%	549.00	540.00	-1.64%	450.00	450.00	0.00%	1568.00	1574.00	0.38%	96.00	96.00	0.00%
fullpgo	381099	408060	7.07%	0.73	0.70	-4.64%	0.63	0.60	-6.00%	0.78	0.73	-6.56%	0.96	0.88	-8.33%	3.42	3.53	3.22%	590.00	614.00	4.07%	553.00	542.00	-1.99%	452.00	447.00	-1.11%	1574.00	1569.00	-0.32%	96.00	96.00	0.00%

orchard about-postgresql

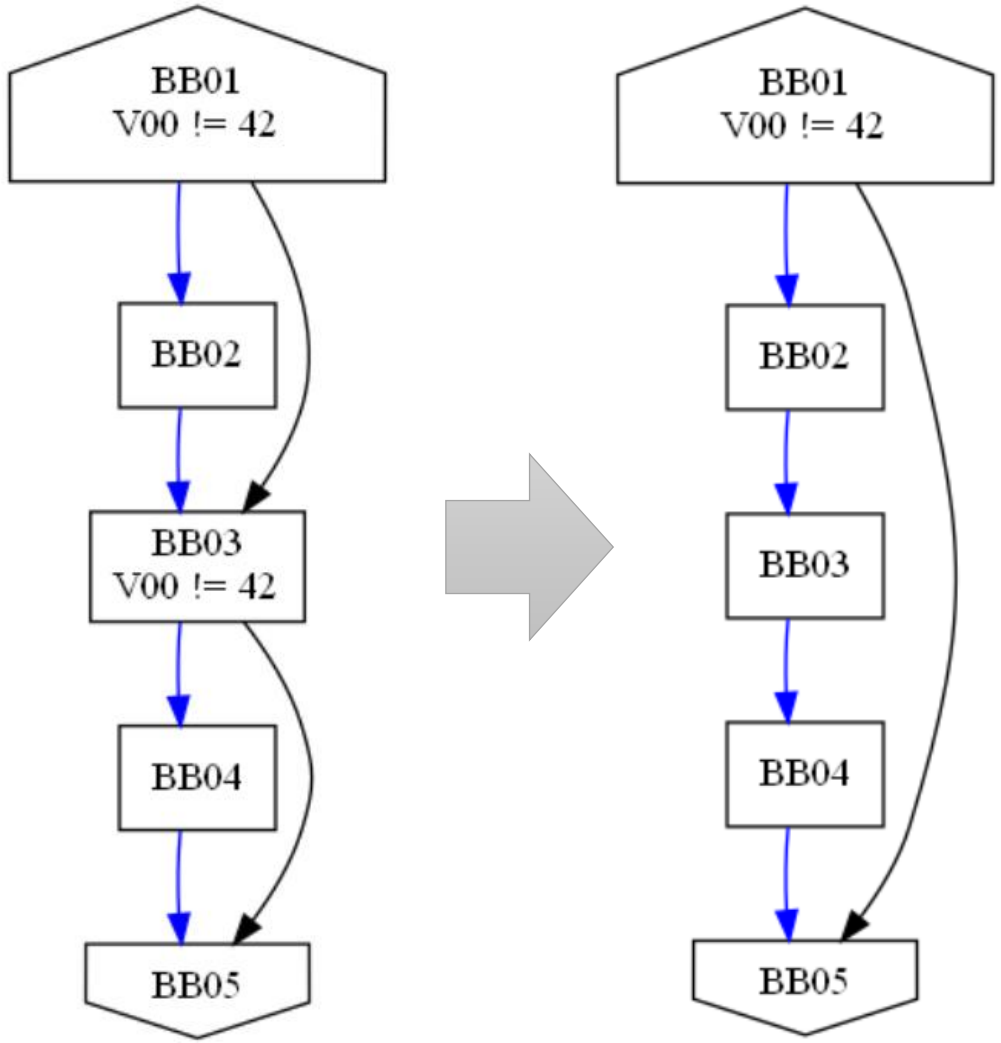
Jit mode	Base RPS	PR RPS	diff	Base mean.lat	PR mean.lat	diff	Base P50	PR P50	diff	Base P75	PR P75	diff	Base P90	PR P90	diff	Base P99	PR P99	diff	Base first.req	PR first.req	diff	Base startup	PR startup	diff	Base work.set	PR work.set	diff	Base priv.mem	PR priv.mem	diff	Base cpu	PR cpu	diff
default	165961	166337	0.23%	190.77	190.34	-0.23%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	4907.00	4926.00	0.39%	414.00	503.00	21.50%	537.00	537.00	0.00%	1620.00	1068.00	-34.07%	96.00	97.00	1.04%
default	153628	169263	10.18%	206.24	187.01	-9.32%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	4843.00	4909.00	1.36%	410.00	421.00	2.68%	537.00	542.00	0.93%	1618.00	1624.00	0.37%	98.00	96.00	-2.04%
fullpgo	172758	188230	8.96%	183.18	167.97	-8.30%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	5479.00	5623.00	2.63%	875.00	876.00	0.11%	540.00	539.00	-0.19%	1634.00	1635.00	0.06%	98.00	99.00	1.02%
fullpgo	173012	185251	7.07%	182.92	170.69	-6.69%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	5599.00	5491.00	-1.93%	862.00	878.00	1.86%	538.00	539.00	0.19%	1634.00	1634.00	0.00%	99.00	99.00	0.00%

proxy proxy-yarp

Jit mode	Base RPS	PR RPS	diff	Base mean.lat	PR mean.lat	diff	Base P50	PR P50	diff	Base P75	PR P75	diff	Base P90	PR P90	diff	Base P99	PR P99	diff	Base first.req	PR first.req	diff	Base startup	PR startup	diff	Base work.set	PR work.set	diff	Base priv.mem	PR priv.mem	diff	Base cpu	PR cpu	diff
default	229833	242880	5.68%	1111.41	1052.13	-5.33%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	185.00	188.00	1.62%	245.00	246.00	0.41%	446.00	451.00	1.12%	1577.00	1581.00	0.25%	95.00	96.00	1.05%
default	230446	242516	5.24%	1109.63	1054.88	-4.93%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	178.00	191.00	7.30%	250.00	247.00	-1.20%	450.00	451.00	0.22%	1580.00	1582.00	0.13%	96.00	96.00	0.00%
fullpgo	249879	268153	7.31%	1022.92	952.98	-6.84%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	321.00	325.00	1.25%	686.00	730.00	6.41%	445.00	450.00	1.12%	1587.00	1601.00	0.88%	96.00	94.00	-2.08%
fullpgo	248507	265782	6.95%	1028.80	962.94	-6.40%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	-1.00	-1.00	0.00%	317.00	322.00	1.58%	684.00	707.00	3.36%	448.00	449.00	0.22%	1588.00	1601.00	0.82%	96.00	96.00	0.00%

“Jump-Threading”

```
int DoWork(int x)
{
    int z = 0;

    if (x == 42)
        z++;

    if (x == 42)
        z++;

    return z;
}
```



CSE for floats and SIMD types

```
void DoWork()
{
    for (int i = 0; i < 1000; i++)
    {
        Console.WriteLine(3.14);
        Console.WriteLine(3.14);

        Console.WriteLine(
            Avx2.Add(
                Vector256.Create(2),
                Vector256.Create(3)).ToScalar());
    }
}
```

```
; Method Program:DoWork():this
G_M56739_IG01:              ;; offset=0000H
       57                   push     rdi
       56                   push     rsi
       4883EC38             sub      rsp, 56
       C5F877               vzeroupper
       C5F829742420         vmovaps  qword ptr [rsp+20H], xmm6
                          ;; bbWeight=1    PerfScore 5.25

G_M56739_IG02:              ;; offset=000FH
       33F6                 xor      esi, esi
       C5FB1035A7000000     vmovsd   xmm6, qword ptr [reloc @RWD64]
       C5FD10055F000000     vmovupd  ymm0, ymmword ptr[reloc @RWD00]
       C5FDFE0577000000     vpaddd   ymm0, ymm0, ymmword ptr[reloc @RWD32]
       C5F97EC7             vmovd    edi, xmm0
                          ;; bbWeight=1    PerfScore 11.25

G_M56739_IG03:              ;; offset=002DH
       C5F828C6             vmovaps  xmm0, xmm6
       E8F2F8FFFF           call     System.Console:WriteLine(double)
       C5F828C6             vmovaps  xmm0, xmm6
       E8E9F8FFFF           call     System.Console:WriteLine(double)
       8BCF                 mov      ecx, edi
       E8F2F8FFFF           call     System.Console:WriteLine(int)
       FFC6                 inc      esi
       81FEE8030000         cmp      esi, 0x3E8
       7CDD                 jl       SHORT G_M56739_IG03
                          ;; bbWeight=4    PerfScore 21.00

G_M56739_IG04:              ;; offset=0050H
       C5F828742420         vmovaps  xmm6, qword ptr [rsp+20H]
       C5F877               vzeroupper
       4883C438             add      rsp, 56
       5E                   pop      rsi
       5F                   pop      rdi
       C3                   ret
                          ;; bbWeight=1    PerfScore 7.25
RWD00   dq  0000000200000002h, 0000000200000002h, 0000000200000002h, 0000000200000002h
RWD32   dq  0000000300000003h, 0000000300000003h, 0000000300000003h, 0000000300000003h
RWD64   dq  40091EB851EB851Fh   ;         3.14
; Total bytes of code: 96
```

Arm64 Parity with x64

Use xmm for stack prolog zeroing rather than rep stos #32538

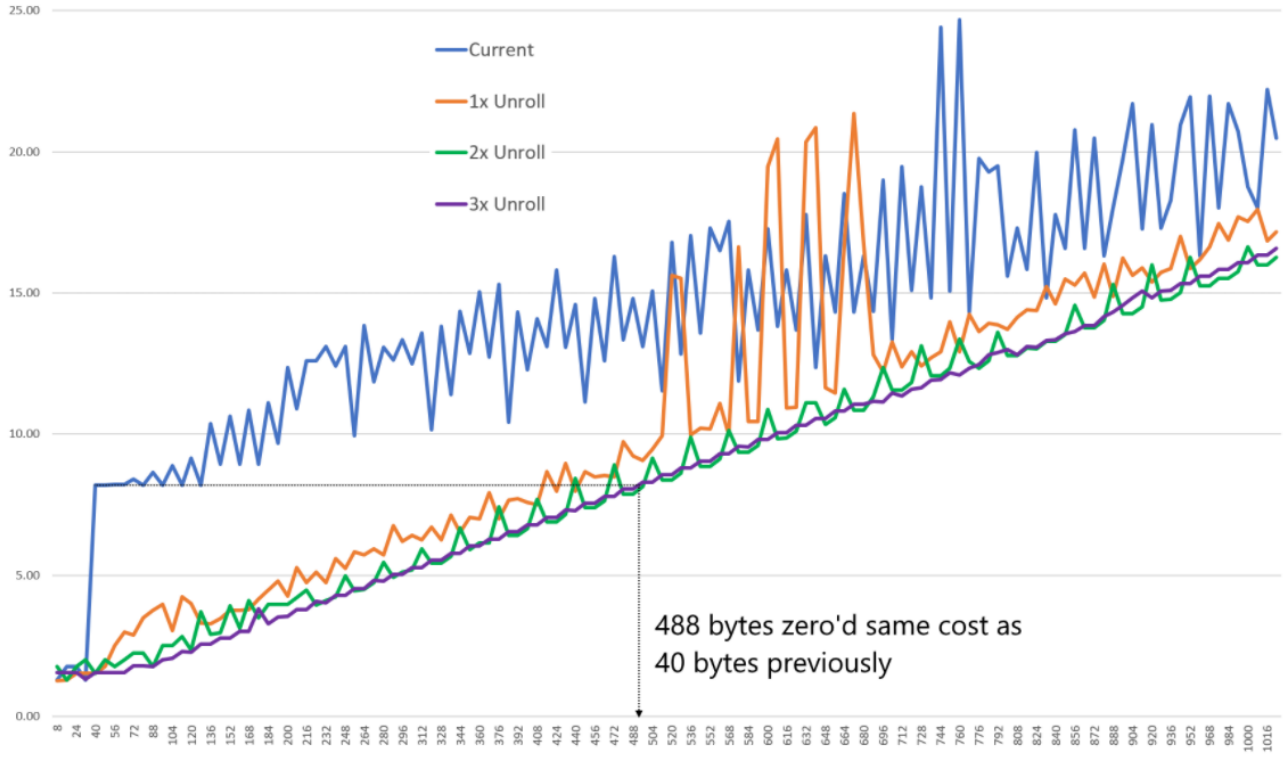
Merged AndyAyersMS merged 17 commits into `dotnet:master` from `benaadams:zero-init-xmm` on Mar 5, 2020

Conversation 217 Commits 17 Checks 149 Files changed 8

benaadams commented on Feb 19, 2020 • edited Member

Use `xmm` registers to clear prolog blocks rather than `rep stosd`.

`rep stos` does have a shallower growth rate so it will eventually overtake; however unlikely in the size range for the stack clearance:



Arm64 Parity with x64

[Arm64] Use SIMD register to zero init frame #46609

Merged **echesakovMSFT** merged 14 commits into `dotnet:master` from `echesakovMSFT:Arm64-Use-SIMDReg-ZeroInitFrame`

Conversation 24 | Commits 14 | Checks 87 | Files changed 14

echesakovMSFT commented on Jan 6 • edited

Below are the results of running benchmarks as in #32538

- baseline
- diff where `DC ZVA` instruction is prohibited
- diff where `DC ZVA` instruction is permitted for memory regions greater than or equal to 256 bytes

Arm64 Parity with x64

Arm64: Use zero register wherever possible #52269

Merged **kunalspathak** merged 1 commit into `dotnet:main` from `kunalspathak:arm64_zr` on May 6

JIT: Optimize redundant memory barriers on arm/arm64 #60219

Merged **EgorBo** merged 6 commits into `dotnet:main` from `EgorBo:redundant-mem-barriers` 6 days ago

RyuJIT: Remove redundant memory barrier for XAdd and XChg on arm #45970

Merged **sandreenko** merged 3 commits into `dotnet:master` from `EgorBo:jit-arm-redundant-dmb` on Dec 29, 2020

[RyuJIT] Implement Interlocked.And and Interlocked.Or for arm64-v8.1 #46253

Merged **EgorBo** merged 14 commits into `dotnet:master` from `EgorBo:intrinsify-interlocked-and-or` on Feb 10

Eliminate intermediate casts to double on ARM64 #53748

Merged **sandreenko** merged 2 commits into `dotnet:main` from `SingleAccretion:Port-AArch32-Opt-To-AArch64` on Jun 7

Updating Matrix4x4 to accelerate several functions using Arm.AdvSimd #40054

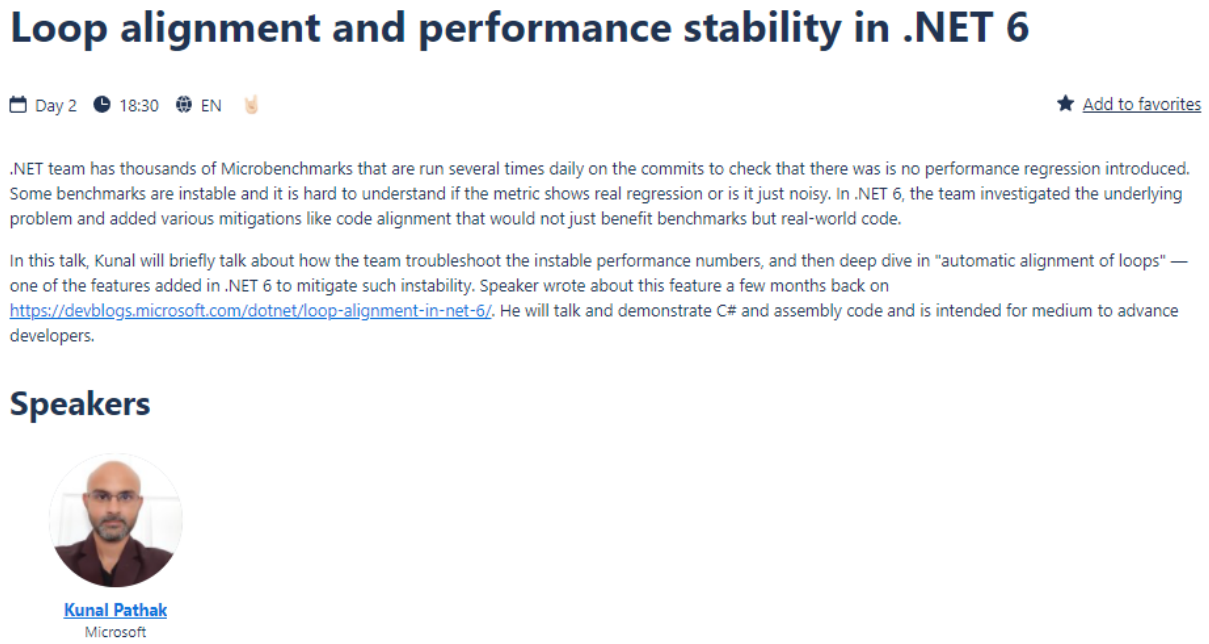
Merged **tannergooding** merged 4 commits into `dotnet:master` from `tannergooding:matrix4x4` on Aug 11, 2020

[Arm64] Treat Math/MathF.FusedMultiplyAdd as intrinsics #40124

Merged **echesakovMSFT** merged 3 commits into `dotnet:master` from `echesakovMSFT:Runtime_40078` on Aug 7, 2020

Not covered in this talk

- Loop Optimizations
- SIMD-related & Peephole Optimizations
- Register Allocator improvements
- Loop Alignment
- Keep structs in registers
- Bug-fixes



Loop alignment and performance stability in .NET 6

Day 2 18:30 EN

Add to favorites

.NET team has thousands of Microbenchmarks that are run several times daily on the commits to check that there was is no performance regression introduced. Some benchmarks are instable and it is hard to understand if the metric shows real regression or is it just noisy. In .NET 6, the team investigated the underlying problem and added various mitigations like code alignment that would not just benefit benchmarks but real-world code.

In this talk, Kunal will briefly talk about how the team troubleshoot the instable performance numbers, and then deep dive in "automatic alignment of loops" — one of the features added in .NET 6 to mitigate such instability. Speaker wrote about this feature a few months back on https://devblogs.microsoft.com/dotnet/loop-alignment-in-net-6/. He will talk and demonstrate C# and assembly code and is intended for medium to advance developers.

Speakers

Kunal Pathak
Microsoft

CLR JIT Team: we're hiring!

yabah@microsoft.com

@EgorBo
Senior Software Engineer at Microsoft