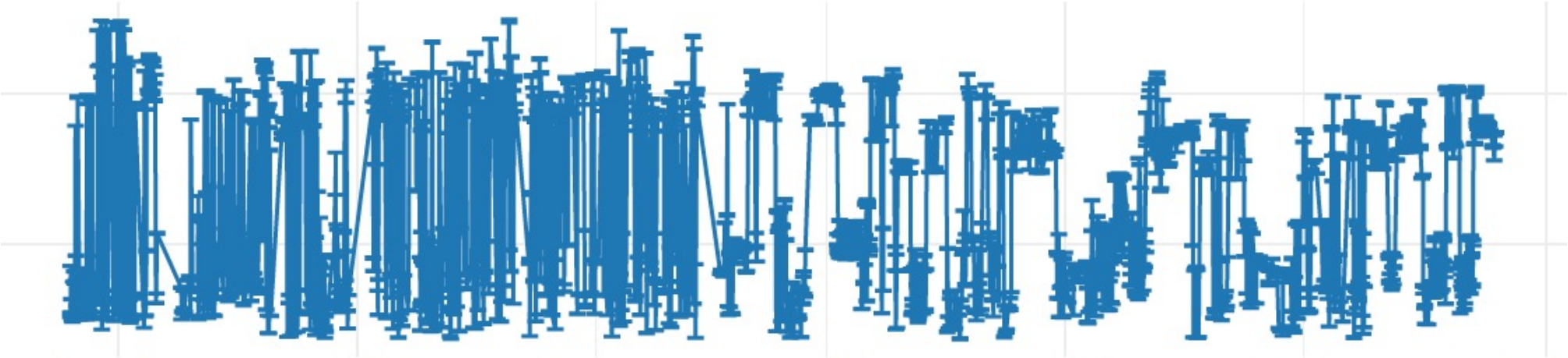
Performance Stability in .NET 6

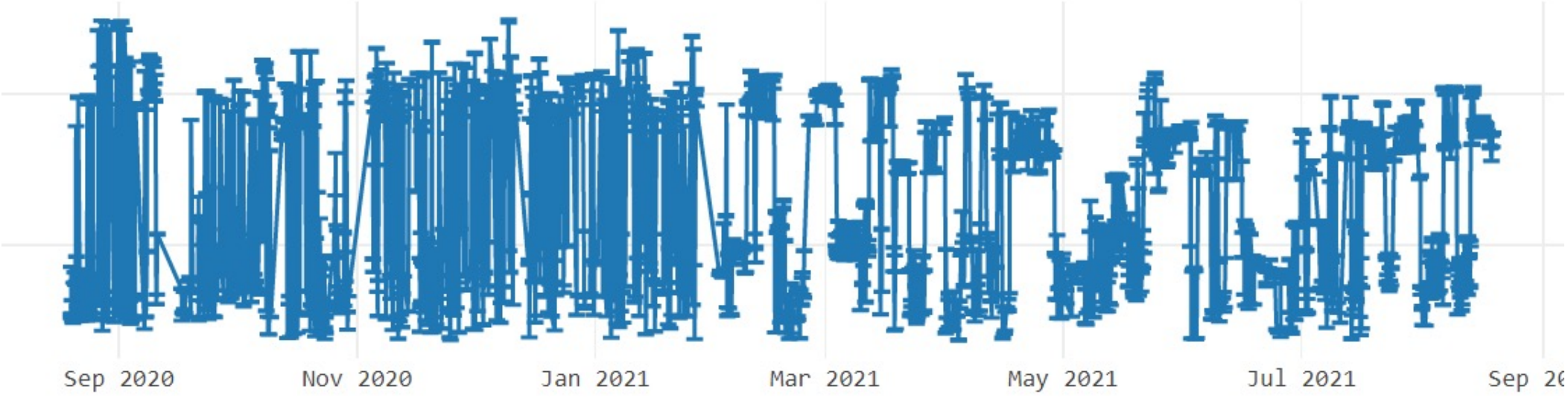


About me

- In Microsoft since 2009
- Loves Compiler domain and performance investigations
- Part of JIT team of .NET Runtime
- Prior member of Javascript Engine “Chakra” team
- Collaborator of nodejs/node and nodejs/node-chakracore
- https://kunalspathak.github.io/
- @KuXMLP

Microsoft

Imagine…



Agenda

- Triaging perf issues
- Deep dive on Code alignment
- Data alignment
- Other tooling improvements

Triaging perf issues

Process

- 1000+ libraries and runtime micro benchmarks
- Windows x86/x64/arm64, Ubuntu x64/arm64
- Ran 10+ times a day on batched commits
- Results aggregated and stored in a database
- Offline analysis of result to identify regressions/improvements
- Issues are filed in dotnet/perf-autofiling-issues
- v-team triage issues once in a week
  - Verify if it is a real issue
  - Narrow down the change that caused it
- Real issues are transferred to dotnet/runtime

https://github.com/dotnet/performance/tree/main/src/benchmarks/micro

Auto-filed issues

- [Perf] Changes at 8/15/2021 6:10:26 PM refs/heads/main Regression Windows 10.0.19042 x64
  #600 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/14/2021 7:50:40 AM refs/heads/main Regression Windows 10.0.19042 x64
  #599 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/14/2021 3:02:07 AM refs/heads/main Regression Windows 10.0.19042 x64
  #598 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/13/2021 7:08:12 PM refs/heads/main Regression Windows 10.0.19042 x64
  #597 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/13/2021 12:19:46 PM refs/heads/main Regression Windows 10.0.19042 x64
  #596 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/11/2021 11:06:12 PM refs/heads/main Regression Windows 10.0.19042 x64
  #594 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/12/2021 10:09:01 PM refs/heads/main Regression Windows 10.0.19042 x64
  #595 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/11/2021 5:11:43 PM refs/heads/main Regression Windows 10.0.19042 x64
  #593 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/11/2021 4:09:47 AM refs/heads/main Regression Windows 10.0.19042 x64
  #592 opened 2 days ago by performanceautofiler bot

- [Perf] Changes at 8/16/2021 4:34:59 AM Improvement refs/heads/main Windows 10.0.19042 x64
  #612 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/15/2021 9:17:00 PM Improvement refs/heads/main Windows 10.0.19042 x64
  #611 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/14/2021 7:50:40 AM Improvement refs/heads/main Windows 10.0.19042 x64
  #610 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/13/2021 12:19:46 PM Improvement refs/heads/main Windows 10.0.19042 x64
  #609 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/12/2021 10:09:01 PM Improvement refs/heads/main Windows 10.0.19042 x64
  #608 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/12/2021 5:58:49 AM Improvement refs/heads/main Windows 10.0.19042 x64
  #607 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/12/2021 1:10:32 AM Improvement refs/heads/main Windows 10.0.19042 x64
  #606 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/11/2021 5:11:43 PM Improvement refs/heads/main Windows 10.0.19042 x64
  #605 opened 2 days ago by performanceautofiler bot
- [Perf] Changes at 8/11/2021 2:15:18 AM Improvement refs/heads/main Windows 10.0.19042 x64
  #604 opened 2 days ago by performanceautofiler bot

https://github.com/dotnet/perf-autofiling-issues/issues

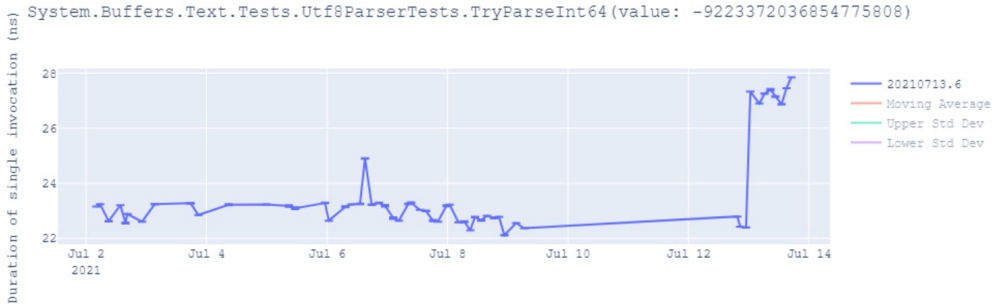
Easy to triage issues

Run Information

Architecture	x64
OS	Windows 10.0.18362
Baseline	f280419a07a9445e1c6724e5717b3da7bdc5be7d
Compare	7b19ccefccb4d116a64bf09c9bb1db3dd1df35e8
Diff	Diff

Regressions in System.Buffers.Text.Tests.Utf8ParserTests

Benchmark	Baseline	Test	Test/Base	Test Quality	Edge Detector	Baseline IR	Compare IR	IR Ratio	Baseline ETL
TryParseInt64 - Duration of single invocation	23.23 ns	27.85 ns	1.20	0.02	False				



https://github.com/dotnet/runtime/issues/56020

Run Information

Architecture	x64
OS	ubuntu 18.04
Baseline	290430eedbfe0e690c8dd119bba6f5a95f2bef53
Compare	47e82c1f625428c02eb6b31d7a7400c53a8c24b9
Diff	Diff

Improvemnts in System.Tests.Perf_Guid

Benchmark	Baseline	Test	Test/Base	Test Quality	Edge Detector	Baseline IR	Compare IR	IR Ratio	Baseline ETL	Co
ctor_str - Duration of single invocation	61.05 ns	30.38 ns	0.50	0.02	True					
Parse - Duration of single invocation	60.29 ns	29.79 ns	0.49	0.03	False					
ParseExactD - Duration of single invocation	55.28 ns	25.21 ns	0.46	0.02	True					

https://github.com/dotnet/perf-autofiling-issues/issues/517

Demo

[Perf] Changes at 8/12/2021 3:35:05 AM #627

New issue

Open performanceautofiler bot opened this issue yesterday · 0 comments

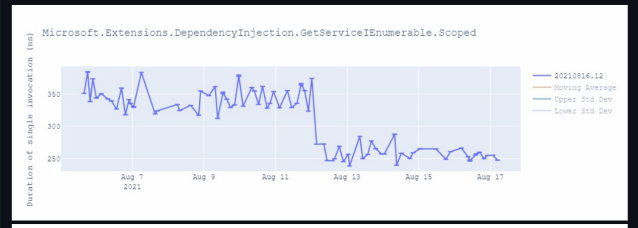
performanceautofiler bot commented yesterday

Run Information

Architecture	arm64
OS	ubuntu 18.04
Baseline	08123228999d8790c12cf83b4f922ba4de789668
Compare	60f1105f6acaa5cd98b4c16fcec1328d3935b90e
Diff	Diff

Improvemnts in Microsoft.Extensions.DependencyInjection.GetServiceIEnumerable

Benchmark	Baseline	Test	Test/Base	Test Quality	Edge Detector	Baseline IR	Compare IR	IR Ratio	Baseline ETL	Cc
Scoped - Duration of single invocation	332.37 ns	247.45 ns	0.74	0.48	False					
Transient - Duration of single invocation	15.70 µs	433.17 ns	0.03	0.51	True					



Assignees
No one assigned

Labels
arm64 Improvement refs/heads/main ubuntu 18.04

Projects
None yet

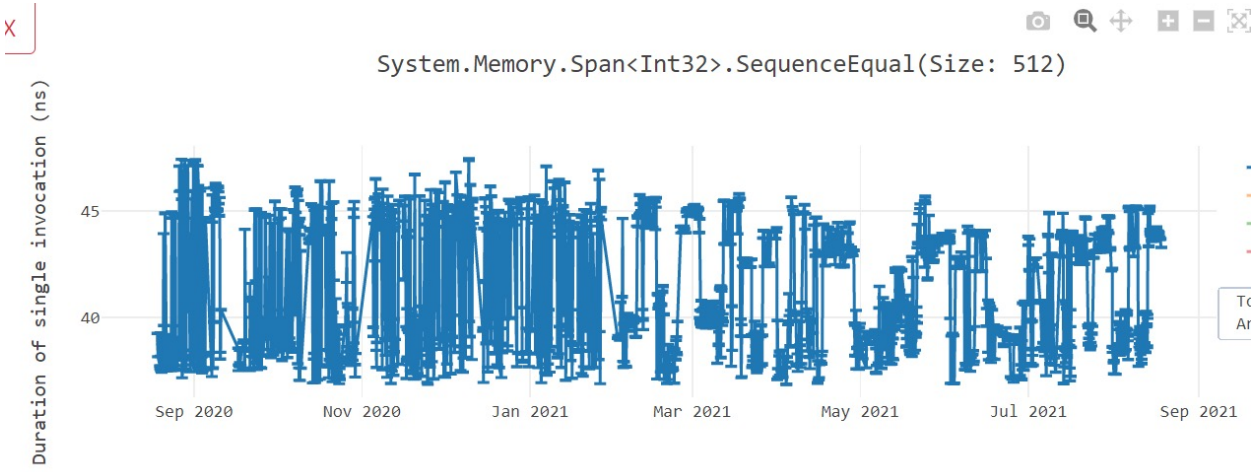
Milestone
No milestone

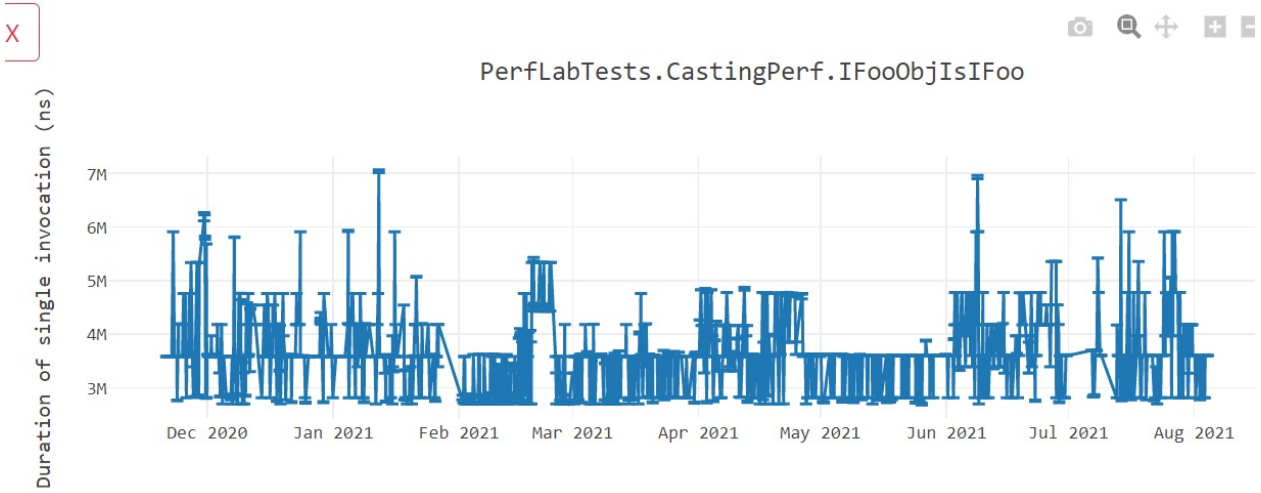
Linked pull requests
Successfully merging a pull request may close this issue.
None yet

Notifications Customize
Subscribe
You're not receiving notifications from this thread.

0 participants

Difficult to triage





https://pvscmdupload.blob.core.windows.net/reports/allTestHistory/refs/heads/main_x64_Windows%2010.0.18362/System.Memory.Span(Int32).SequenceEqual(Size%3a%20512).html
https://pvscmdupload.blob.core.windows.net/reports/allTestHistory/refs/heads/main_arm64_ubuntu%2018.04/PerfLabTests.CastingPerf.IFooObjIsIFoo.html

Improvements in .NET 6

- Replaced https://github.com/DrewScoggins/performance-2 with https://github.com/dotnet/perf-autofiling-issues
- Statistical analysis improvements by Drew
- Microsoft Edge team's regression analyzer
- ML analyzer

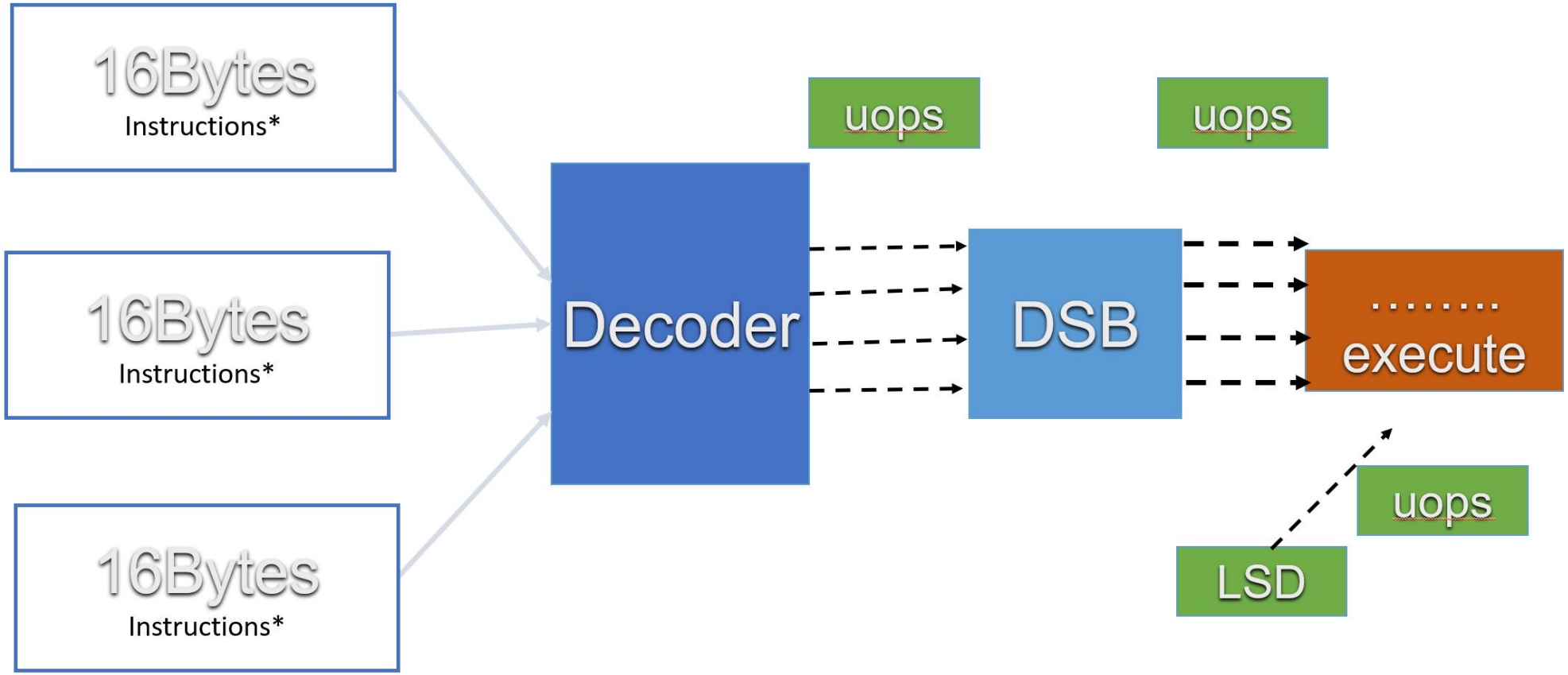
TODOs for .NET 7

- Eliminate false positives and noisy issues
- Integrate ML/Edge analyzer to flag real issues
- Re-bucketize the issues
  - All benchmarks affected by given commit range has a single issue
- Ambitious: Auto pilot mode

Code alignment

Assembly code on the way…

Computer Architecture 101



https://www.youtube.com/watch?v=IX16gcX4vDQ&ab_channel=LLVM

Alignment

- CISC (Intel/AMD)

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00007ff9a59feb80																
00007ff9a59feb90															X	X
00007ff9a59febA0	1	1	1	1	1	2	2	2	3	3	3	4	4	4	4	
00007ff9a59febB0																

- RISC (Arm)

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00007ff9a59feb80																
00007ff9a59feb90																
00007ff9a59febA0																
00007ff9a59febB0																

- Synonyms: Padding, NOPs, align instructions
- 16B: (address % 16) == 0

Method alignment in .NET 5

- Only for Windows x86/x64, and Ubuntu x64
- Methods having loops starts at 32B boundary
- Hot Ngen code starts at 16B boundary
- Smaller (< 16 bytes) JITed methods starts at 16B boundary
- Else:
  - Methods on x86 starts at 4B boundary
  - Methods on x64 starts at 8B boundary

https://github.com/dotnet/runtime/pull/2249
https://github.com/dotnet/runtime/pull/42909

Loop alignment in .NET 6

- Identify hot inner most loop(s)
- Add NOP instructions to align the loop code

```
 00007ffd`10d1b4c8        41C1EB1F            shr      r11d, 31
 00007ffd`10d1b4cc        4585D3              test     r10d, r11d
 00007ffd`10d1b4cf        7429                je       SHORT G_M58758_IG05
 00007ffd`10d1b4d1                            align    [15 bytes]
; ............................... 32B boundary ...............................

G_M58758_IG03:
 00007ffd`10d1b4e0        4D63D1              movsxd   r10, r9d
 00007ffd`10d1b4e3        4203449110          add      eax, dword ptr [rcx+4*r10+16]
 00007ffd`10d1b4e8        4503C1              add      r8d, r9d
 00007ffd`10d1b4eb        412BC0              sub      eax, r8d
 00007ffd`10d1b4ee        03C2                add      eax, edx
 00007ffd`10d1b4f0        41FFC1              inc      r9d
 00007ffd`10d1b4f3        443BCA              cmp      r9d, edx
 00007ffd`10d1b4f6        7CE8                jl       SHORT G_M58758_IG03

G_M58758_IG04:
 00007ffd`10d1b4f8        EB1E                jmp      SHORT G_M58758_IG06

G_M58758_IG05:
 00007ffd`10d1b4fa        443B4908            cmp      r9d, dword ptr [rcx+8]
 00007ffd`10d1b4fe        7320                jae      SHORT G_M58758_IG08
; ............................... 32B boundary ...............................
 00007ffd`10d1b500        4D63D1              movsxd   r10, r9d
 00007ffd`10d1b503        4203449110          add      eax, dword ptr [rcx+4*r10+16]
```

Loop selection

- Align only non-nested loops
- Expensive to align every loop
- Developer controls alignment
  - LLVM: “-align-all-*”
  - GCC: “-falign-loops”

```
void Method(int N, int M, bool some_condition) {
    for (int i = 0; i < N; i++) {

        // alignment candidate
        for (int j = 0; j < M; j++) {
            // body
        }
    }

    if (some_condition) {
        return;
    }

    // alignment candidacy depends on frequency
    // we come here
    for (int i = 0; i < M + N; i++) {
        // body
    }
}
```

COMPlus_JitAlignLoopMinBlockWeight

No alignment – Loops with calls

- Alignment reduces code fetches
- Method call swaps caller code with callee code
- Alignment cannot benefit such loops
- For inlined calls, continue loop alignment

```
void Method(int N, bool some_condition) {

    // no alignment because of method call
    for (int i = 0; i < N; i++) {

        if (some_condition) {
            MethodCall();
        }
    }
}
```

No alignment – Cloned loops

```
public static void M(int[] c, int N) {
    for (int i = 0; i < N; i++) {
        c[i] += 1;
    }
}
```

```
public static void M(int[] c, int N) {
    if (N == 0) {
        return;
    }

    int i = 0;

    if ((c != null) && (c.Length > N)) {
        // fast loop - alignment candidate
        for (; i < N; i++) {
            c[i] += 1;
        }
    }
    else {
        // slow loop - no alignment
        for (; i < N; i++) {
            if (c.Length <= i) {
                throw new IndexOutOfRangeException("");
            }

            c[i] += 1;
        }
    }
}
```

No alignment – Unrolled loops

```java
public static int M(int sum) {
    for (int i = 0; i < 5; i++) {
        sum += i;
    }

    return sum;
}
```

```java
public static int M(int sum) {
    sum += 1;
    sum += 2;
    sum += 3;
    sum += 4;

    return sum;
}
```

Loop size matters!

- Small loops shows most benefits
- Large loops needs several code fetches anyway
  - Alignment won’t help prevent or reduce the fetches
- Align loops only if they fit in 3 chunks of 32B i.e. 96 bytes long

COMPlus_JitAlignLoopMaxCodeSize=96

Alignment boundary choices

- Alignment boundary choices = 16B, 32B or 64B
- Recommended boundary by Intel/AMD/Arm = 32B
- Default to 32B alignment boundary*

* Varies for adaptive vs. non-adaptive loop alignment

COMPlus_JitAlignLoopBoundary=32

32B alignment

```
public static void M(int[] c, int N) {

    for (int i = 0; i < N; i++) {
        // some code

        // loop alignment candidate
        for (int j = 0; j < c.Length; j++) {
            // some code
        }

    }

    // some code
}
```

```
G_M58758_IG03:
 00007ffd`10d0b4cb        movsxd   rax, r10d
 00007ffd`10d0b4ce        add      r8d, dword ptr [rcx+4*rax+16]
 00007ffd`10d0b4d3        add      r9d, r10d
; ............................... 32B boundary ...................
 00007ffd`10d0b4d6        sub      r8d, r9d
 00007ffd`10d0b4d9        add      r8d, edx
 00007ffd`10d0b4dc        xor      esi, esi
 00007ffd`10d0b4de        test     r11d, r11d
 00007ffd`10d0b4e1        jle      SHORT G_M58758_IG05
 00007ffd`10d0b4e3        align    [29 bytes]
; ............................... 32B boundary ...................

G_M58758_IG04:
 00007ffd`10d0b500        lea      rdi, bword ptr [rcx+4*rax+16]
 00007ffd`10d0b505        add      dword ptr [rdi], r10d
 00007ffd`10d0b508        add      r8d, r10d
 00007ffd`10d0b50b        add      r9d, r10d
 00007ffd`10d0b50e        inc      esi
 00007ffd`10d0b510        cmp      r11d, esi
 00007ffd`10d0b513        jg       SHORT G_M58758_IG04

G_M58758_IG05:
 00007ffd`10d0b515        inc      r10d
 00007ffd`10d0b518        cmp      r10d, edx
 00007ffd`10d0b51b        jl       SHORT G_M58758_IG03
```

Drawbacks

- To align code to N-byte boundary, need at most N-1 bytes padding
- Sometimes, more padding added than the loop size
- More code memory is consumed
- Penalty of fetching and decoding NOPs if padding is for deeply nested loop
- aka Non-adaptive loop alignment

Adaptive loop alignment

- Padding amount threshold depends on the loop size
  - More padding for small loops
  - Less padding for larger loops

- Adjust alignment boundary from 32B -> 16B
  - For 32B, if padding needed is large, try to align to 16B
  - Instead of bailing out, give one more try to align loop

COMPlus_JitAlignLoopAdaptive=1

Adaptive loop alignment cont.

- 32B boundary

Max Pad (bytes)	Minimum 32B blocks needed to fit the loop
15	1 (32 bytes)
7	2 (64 bytes)
3	3 (96 bytes)
1	4 (128 bytes)

- 16B boundary

Max Pad (bytes)	Minimum 32B blocks needed to fit the loop
7	1 (32 bytes)
3	2 (64 bytes)
1	3 (96 bytes)

COMPlus_JitAlignLoopAdaptive=1

32B adaptive alignment

```
public static void M(int[] c, int N) {

    for (int i = 0; i < N; i++) {
        // some code

        // loop alignment candidate
        for (int j = 0; j < c.Length; j++) {
            // some code
        }

    }

    // some code
}
```

```
G_M58758_IG03:
 00007ffd`10d3b496        movsxd   rax, r10d
 00007ffd`10d3b499        add      r8d, dword ptr [rcx+4*rax+16]
 00007ffd`10d3b49e        add      r9d, r10d
; ............................... 32B boundary .......................
 00007ffd`10d3b4a1        sub      r8d, r9d
 00007ffd`10d3b4a4        add      r8d, edx
 00007ffd`10d3b4a7        xor      esi, esi
 00007ffd`10d3b4a9        test     r11d, r11d
 00007ffd`10d3b4ac        jle      SHORT G_M58758_IG05
 00007ffd`10d3b4ae        align    [2 bytes]

G_M58758_IG04:
 00007ffd`10d3b4b0        lea      rdi, bword ptr [rcx+4*rax+16]
 00007ffd`10d3b4b5        add      dword ptr [rdi], r10d
 00007ffd`10d3b4b8        add      r8d, r10d
 00007ffd`10d3b4bb        add      r9d, r10d
 00007ffd`10d3b4be        inc      esi
; ............................... 32B boundary .......................
 00007ffd`10d3b4c0        cmp      r11d, esi
 00007ffd`10d3b4c3        jg       SHORT G_M58758_IG04

G_M58758_IG05:
 00007ffd`10d3b4c5        inc      r10d
 00007ffd`10d3b4c8        cmp      r10d, edx
 00007ffd`10d3b4cb        jl       SHORT G_M58758_IG03
```

Skip alignment for pre-aligned loops

- Loops that start at 32B boundary

```
00007ff9a91f597a        cmp      dword ptr [rbp+8], r8d
00007ff9a91f597e        jl       SHORT G_M24050_IG12
; ............................... 32B boundary ....................
00007ff9a91f5980        align    [0 bytes]

G_M24050_IG10:
00007ff9a91f5980        movsxd   rdx, ecx
00007ff9a91f5983        mov      r9, qword ptr [rbp+8*rdx+16]
00007ff9a91f5988        mov      qword ptr [rsi+8*rdx+16], r9
00007ff9a91f598d        inc      ecx
00007ff9a91f598f        cmp      r8d, ecx
00007ff9a91f5992        jg       SHORT G_M24050_IG10
```

Skip alignment for pre-aligned loops

- Loops that fit in a single 32B block

```
; ............................... 32B boundary ..................
00007ff9a921a903        call     CORINFO_HELP_NEWARR_1_VC
00007ff9a921a908        xor      ecx, ecx
00007ff9a921a90a        mov      edx, dword ptr [rax+8]
00007ff9a921a90d        test     edx, edx
00007ff9a921a90f        jle      SHORT G_M24257_IG05
00007ff9a921a911        align    [0 bytes]

G_M24257_IG04:
00007ff9a921a911        movsxd   r8, ecx
00007ff9a921a914        mov      qword ptr [rax+8*r8+16], rsi
00007ff9a921a919        inc      ecx
00007ff9a921a91b        cmp      edx, ecx
00007ff9a921a91d        jg       SHORT G_M24257_IG04

G_M24257_IG05:
00007ff9a921a91f        add      rsp, 40
; ............................... 32B boundary ..................
```

Skip alignment for pre-aligned loops

- Loop already present in minimum blocks needed

```
; ............................... 32B boundary .......................
00007ff9a921c662        mov      r12d, dword ptr [r14+8]
00007ff9a921c666        test     r12d, r12d
00007ff9a921c669        jle      SHORT G_M11250_IG07
00007ff9a921c66b        align    [0 bytes]

G_M11250_IG04:          ; 35-bytes loop
00007ff9a921c66b        cmp      r15d, ebx
00007ff9a921c66e        jae      G_M11250_IG19
00007ff9a921c674        movsxd   rax, r15d
00007ff9a921c677        shl      rax, 5
00007ff9a921c67b        vmovupd  ymm0, ymmword ptr[rsi+rax+16]
; ............................... 32B boundary .......................
00007ff9a921c681        vmovupd  ymmword ptr[r14+rax+16], ymm0
00007ff9a921c688        inc      r15d
00007ff9a921c68b        cmp      r12d, r15d
00007ff9a921c68e        jg       SHORT G_M11250_IG04

G_M11250_IG05:
00007ff9a921c690        jmp      SHORT G_M11250_IG07
; ............................... 32B boundary .......................
```

Padding placement

- Currently, placed before the loop's first instruction
- Can be in a blind spot behind an unconditional jump
- Can be in a cold block

```
00007ff9a59feb6b        jmp      SHORT G_M17025_IG30

G_M17025_IG29:
00007ff9a59feb6d        mov      rax, rcx

G_M17025_IG30:
00007ff9a59feb70        mov      ecx, eax
00007ff9a59feb72        shr      ecx, 3
00007ff9a59feb75        xor      r8d, r8d
00007ff9a59feb78        test     ecx, ecx
00007ff9a59feb7a        jbe      SHORT G_M17025_IG32
00007ff9a59feb7c        align    [4 bytes]
; ............................... 32B boundary ................
G_M17025_IG31:
00007ff9a59feb80        vmovupd  xmm0, xmmword ptr [rdi]
00007ff9a59feb84        vptest   xmm0, xmm6
00007ff9a59feb89        jne      SHORT G_M17025_IG33
00007ff9a59feb8b        vpackuswb xmm0, xmm0, xmm0
00007ff9a59feb8f        vmovq    xmmword ptr [rsi], xmm0
00007ff9a59feb93        add      rdi, 16
00007ff9a59feb97        add      rsi, 8
00007ff9a59feb9b        inc      r8d
00007ff9a59feb9e        cmp      r8d, ecx
; ............................... 32B boundary ................
00007ff9a59feba1        jb       SHORT G_M17025_IG31
```

Padding placement

- Padding can be converted to an unconditional jump
- Spread the padding across cold blocks

From:

```
    align [14 bytes]
IG_05:
    ...
    jne JG_05
```

To:

```
    jmp IG_05
    align [12 bytes]   ; assuming 3 bytes were consumed in jmp above
IG_05:
   ...
   jne IG_05
```

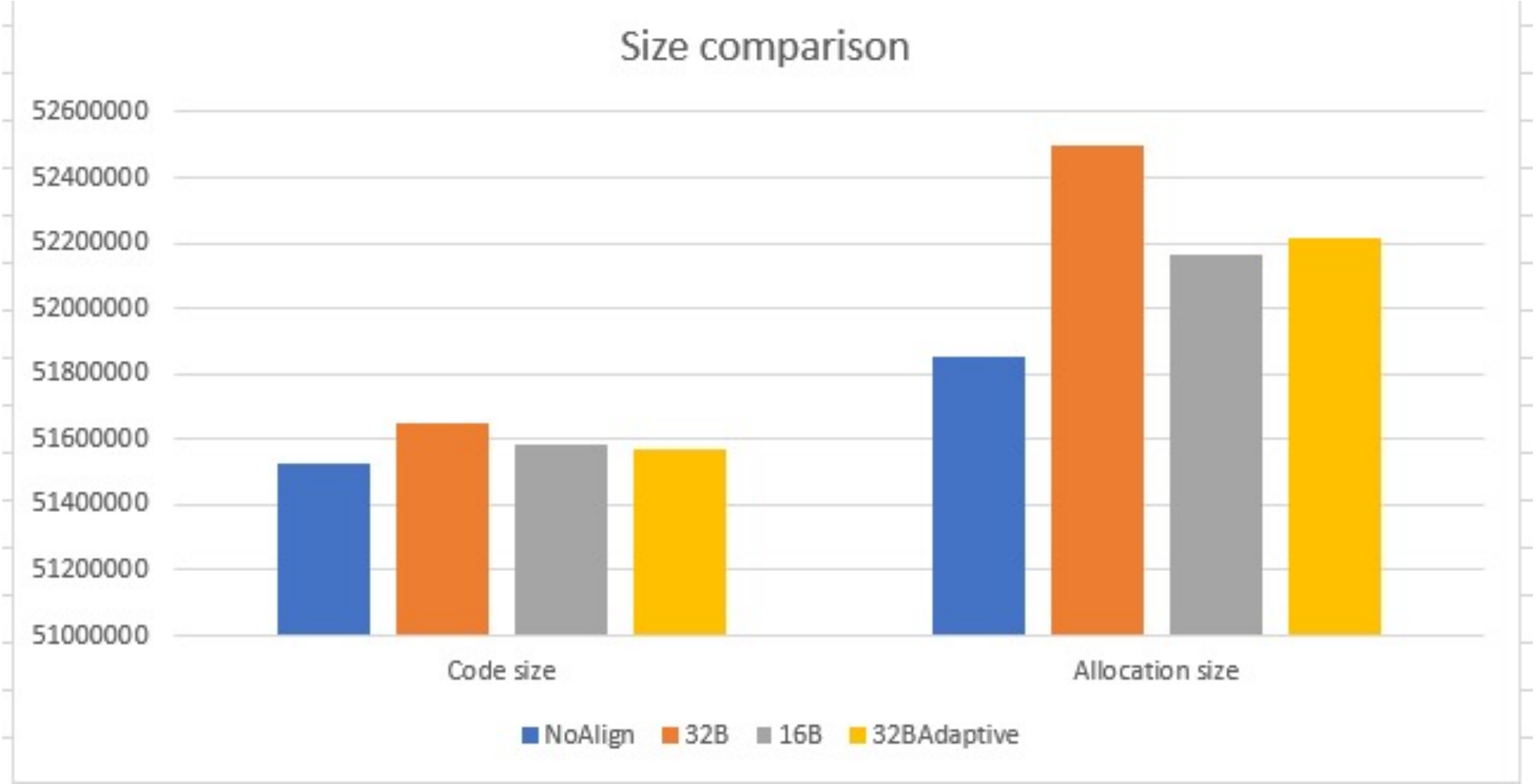
Recap

- Identify hot non-nested loop that needs alignment
  - Filtered loops with calls, cloned loops, etc.
- Determine if loop is small enough to benefit from padding
- Determine if the loop is not pre-aligned
- Determine the padding amount to be added
- Add the padding before the loop

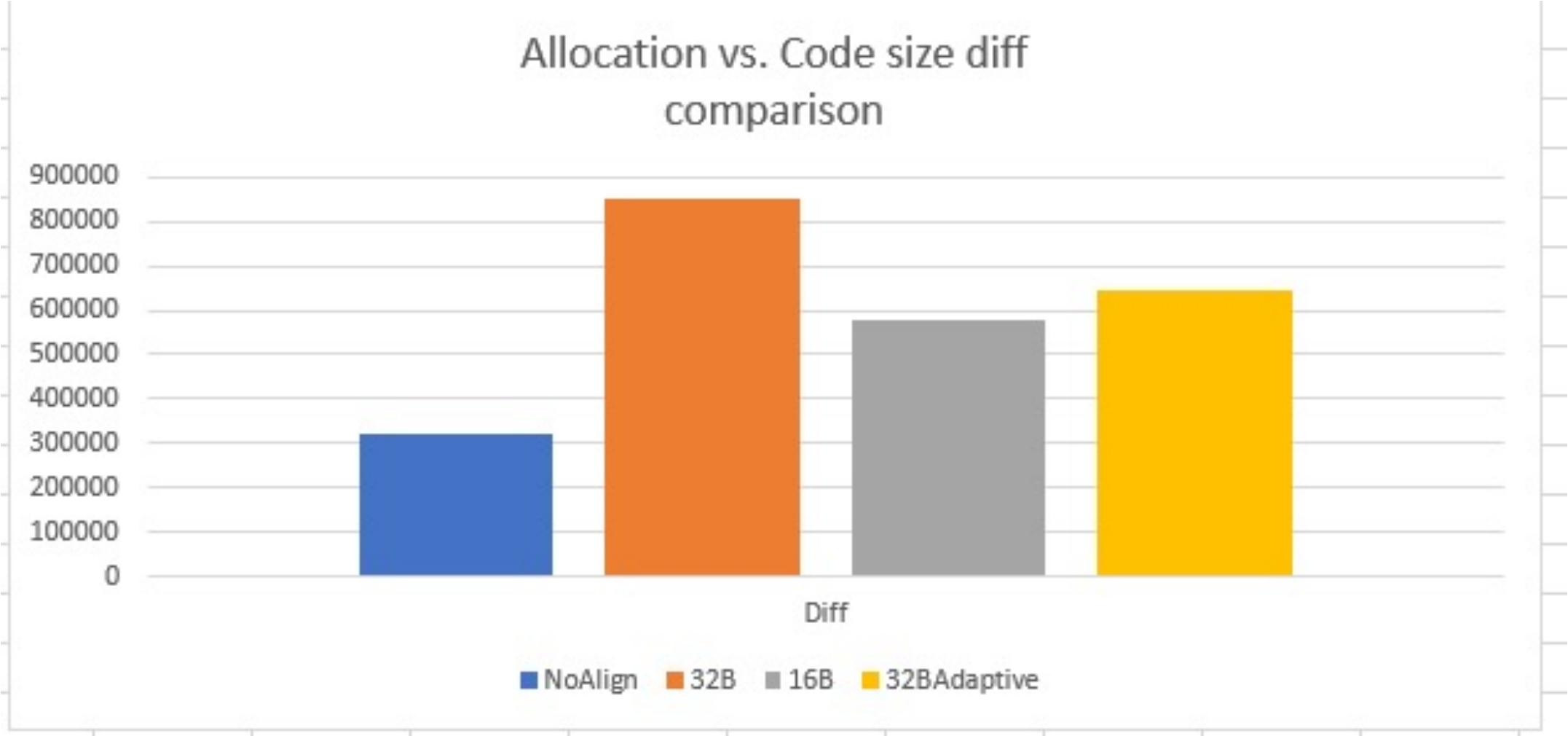
[Optional] Nasty details

- During codegen, walk the program tree and calculate instruction sizes to find out memory needed to store machine code
- Based on instruction size estimate, we predetermine how much padding to add
- Allocate memory from runtime
- During outputting instructions in final memory, we see some are over-estimated
- Need to add extra NOP to compensate over-estimation such that the padding amount is still valid

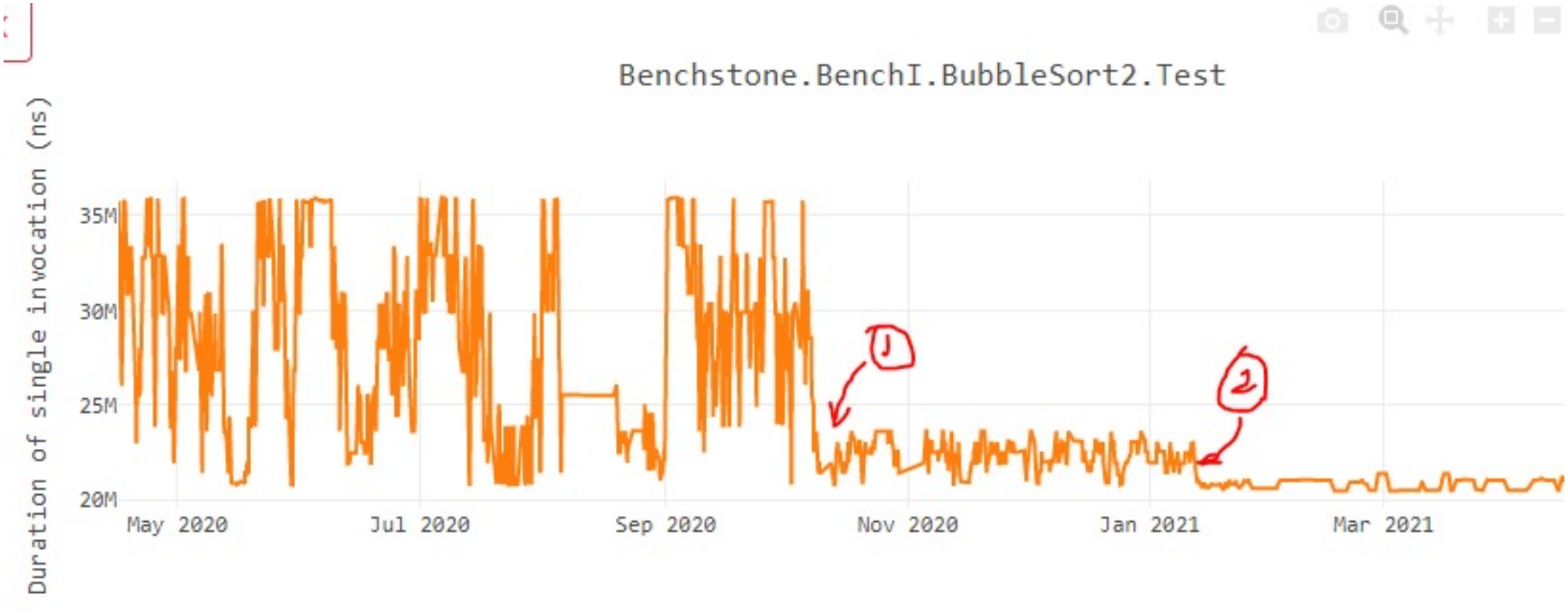
Impact on Memory cost



Impact on Memory cost

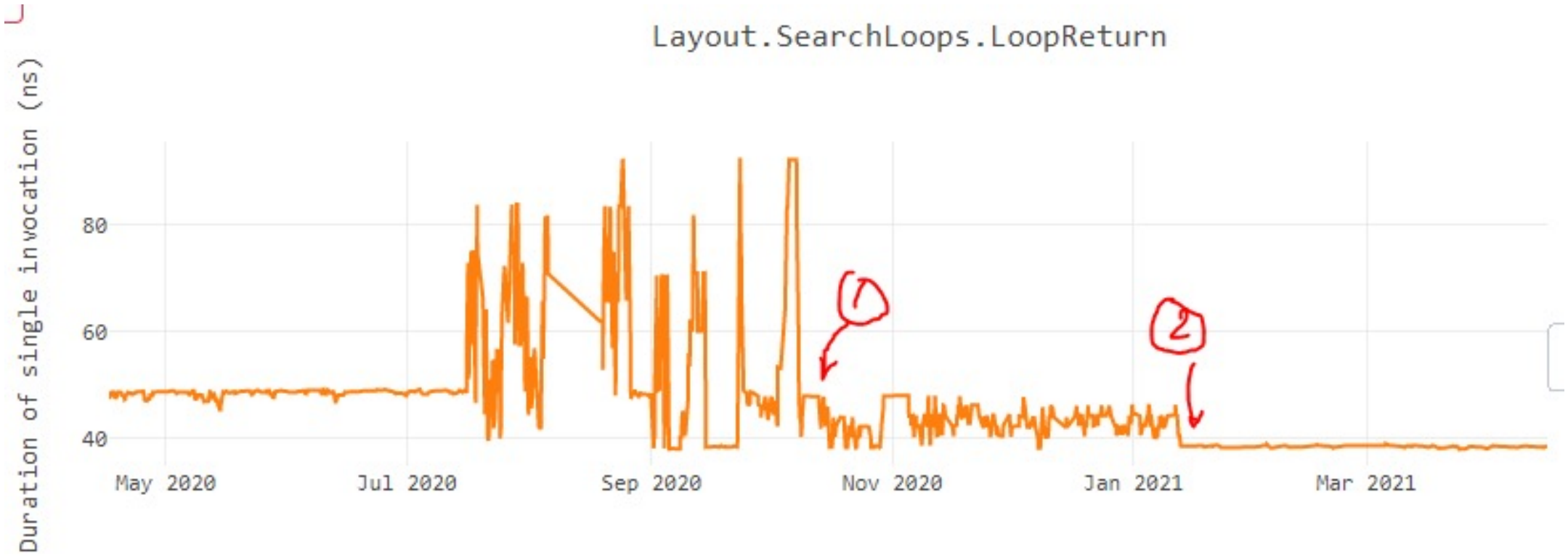


Impact on Performance/Stability



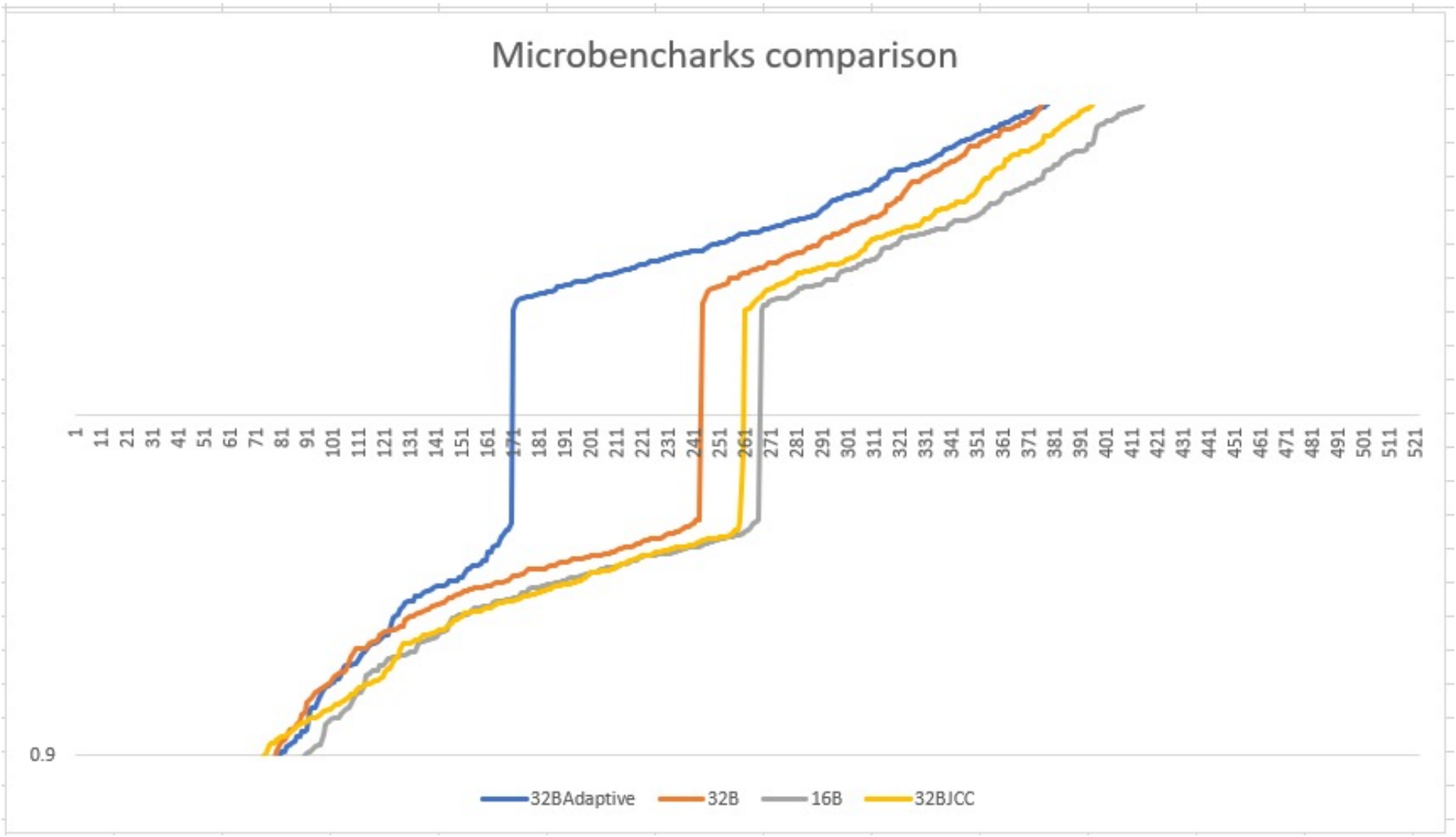
https://pvscmdupload.blob.core.windows.net/reports/allTestHistory%2frefs%2fheads%2fmain_x64_ubuntu%2018.04%2fBenchstone.BenchI.BubbleSort2.Test.html

Impact on Performance/Stability



https://pvscmdupload.blob.core.windows.net/reports/allTestHistory%2frefs%2fheads%2fmain_x64_ubuntu%2018.04%2fLayout.SearchLoops.LoopReturn.html

Overall comparison



TODOs for .NET 7

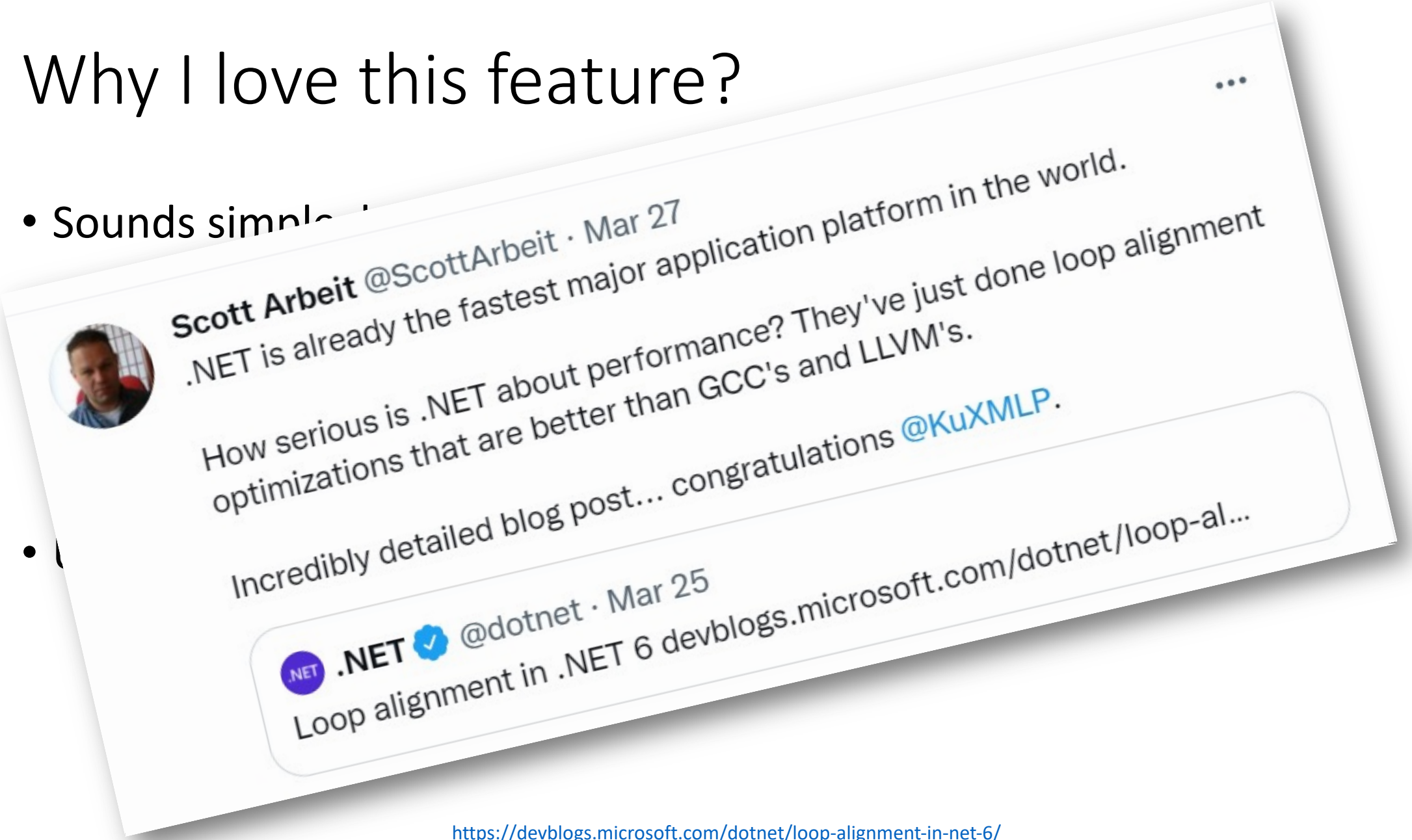
- Loop alignment for ReadyToRun code
- Loop alignment for Arm64
- Improve padding placement

https://github.com/dotnet/runtime/issues/43227

Why I love this feature?

- Sounds simple

-



https://devblogs.microsoft.com/dotnet/loop-alignment-in-net-6/

Data alignment

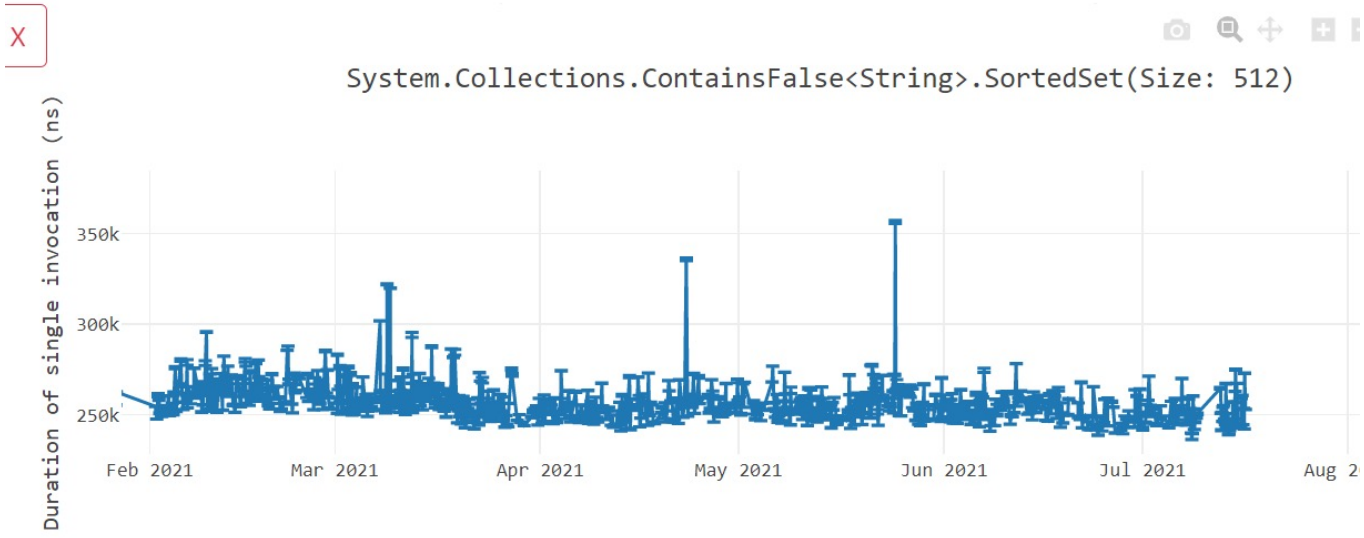
What is the issue?

- Performance *can* degrade due to unaligned memory access
- Degradation amplifies if access is inside a loop
- MicroBenchmarks allocates data once and access/update it inside benchmark code
- Source of performance instability if nothing else changed between two runs

Sample benchmark code

```
[GlobalSetup]
public void Setup() {
    var values = ValuesGenerator.ArrayOfUniqueValues<T>(Size * 2);
    _notFound = values.Take(Size).ToArray();
    _sortedSet = new SortedSet<T>();
}

[Benchmark]
public bool SortedSet() {
    bool result = default;
    SortedSet<T> collection = _sortedSet;
    T[] notFound = _notFound;
    for (int i = 0; i < notFound.Length; i++)
        result ^= collection.Contains(notFound[i]);
    return result;
}
```



https://github.com/dotnet/performance/blob/bb492c8fb9a27c087154cfec522d874bc1e96a1e/src/benchmarks/micro/libraries/System.Collections/Contains/ContainsFalse.cs#L152-L160

Memory randomization

- Allocate random memory in GlobalSetup
- Invoke GlobalSetup after every iteration of benchmark execution
- Verify the histogram of measurement distribution

https://github.com/dotnet/BenchmarkDotNet/pull/1587
https://github.com/dotnet/performance/pull/1587

Sample usage

```
public class IntroMemoryRandomization {
    [Params(512 * 4)]
    public int Size;

    private int[] _array;
    private int[] _destination;

    [GlobalSetup]
    public void Setup() {
        _array = new int[Size];
        _destination = new int[Size];
    }

    [Benchmark]
    public void Array() => System.Array.Copy(_array, _destination, Size);
}
```

https://github.com/dotnet/BenchmarkDotNet/pull/1587

- Default

```
-------------------- Histogram --------------------
[502.859 ns ; 508.045 ns) | @@@@@@@@@@@@@@
---------------------------------------------------
```

- Memory randomization with no outlier removed

```
dotnet run -c Release --memoryRandomization true --outliers DontRemove --maxIterationCount 50
```

```
-------------------- Histogram --------------------
[108.803 ns ; 213.537 ns) | @@@@@@@@@@@@@@
[213.537 ns ; 315.458 ns) |
[315.458 ns ; 446.853 ns) | @@@@@@@@@@@@@@@@@@@
[446.853 ns ; 559.259 ns) | @@@@@@@@@@@@@@
---------------------------------------------------
```

TODOs for .NET 7

- Turn ON by default
- Investigate the impact of measurement history, if made ON
- Annotate 1000+ benchmarks if they are impacted by memory randomization?

Tooling improvements

Superpmi collections

- Collection of cached method contexts

Collection type	Number of methods
Libraries pmi	233,282
Benchmarks run	26,777
Coreclr tests	254,004
Libraries tests	344,291
ASP.NET benchmarks run	43,533

- Metrics: Code size, PerfScore, Allocation Size, Instruction count

https://github.com/dotnet/runtime/blob/main/src/coreclr/scripts/superpmi.md

Superpmi diffs

```
Summary of Perf Score diffs:
(Lower is better)

Total PerfScoreUnits of base: 767855166.8899993
Total PerfScoreUnits of diff: 766740526.5299989
Total PerfScoreUnits of delta: -1114640.36 (-0.15% of base)

Total relative delta: -15.42
    diff is an improvement.
    relative diff is an improvement.

1114 total files with Perf Score differences (941 improved, 173 regressed), 721 unchanged.

Top method regressions (PerfScoreUnits):
    14260.78 ( 2.99% of base) : 12425.dasm - DynamicClass:Regex1_Go(System.Text.RegularExpressions.RegexRunner)
      528.00 ( 8.70% of base) : 25211.dasm - System.Tests.Perf_Array:ArrayRetrieve3D():int:this
      368.40 ( 2.34% of base) : 9909.dasm - System.Text.StringBuilder:AppendFormatHelper(System.IFormatProvider,System.Stri
      366.70 ( 2.34% of base) : 365.dasm - System.Text.ValueStringBuilder:AppendFormatHelper(System.IFormatProvider,System.

Top method improvements (PerfScoreUnits):
    -1048874.40 (-0.14% of base) : 6239.dasm - Utf8Json.Resolvers.Internal.DynamicObjectTypeBuilder:BuildSerialize(System.T
    -36872.00 (-30.41% of base) : 19476.dasm - Microsoft.CodeAnalysis.CSharp.Symbols.Metadata.PE.PENamedTypeSymbol:MakeDecl
    -21876.30 (-8.64% of base) : 19674.dasm - Microsoft.CodeAnalysis.CSharp.Symbols.SourceMemberContainerTypeSymbol:Compute
    -3120.85 (-5.99% of base) : 19000.dasm - Microsoft.CodeAnalysis.CSharp.CSharpCompilation:GetDiagnostics(int,bool,Micros
```

https://gist.github.com/kunalspathak/9dd9c76a740f3399dd19a3ed0edabddf

Thank you