.NET PERFORMANCE INVESTIGATION

Adam Sitnik

About Myself

- BenchmarkDotNet maintainer
- Bibliotecario #dotnet #Microsoft
  - Building Performance Culture
  - Preventing|Detecting|Solving regressions in .NET Core
  - Making .NET Core even faster
  - Making various .NET libraries faster: ML.NET, .NET for Apache Spark

Measure, measure, measure.

Without data you're just another person with an opinion

— W. Edwards Deming, a data scientist

Benchmark? Profiler?

„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”

Wikipedia

„In software engineering, profiling ("program profiling", "software profiling") is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or the frequency and duration of function calls. Most commonly, profiling information serves to aid program optimization.”

Wikipedia

Recommended Settings

- Release (not Debug)
- Symbols:

  <DebugType>pdbonly</DebugType>

  <DebugSymbols>true</DebugSymbols>
- Disable Tiered JIT (or warmup the code)

  <TieredCompilation>false</TieredCompilation>

Small Repro: ML.NET regression

	Before	**After**
.NET Core	2.2	3.0
Tiered JIT	Disabled by default	Enabled by default
Vectorized Math	Native library	Managed library

- Narrow down:
  - Disable Tiered JIT and run the .NET Core 3.0 benchmarks
  - Run version with native dependency as .NET Core 3.0

Problem:
  - Vectorized Math library!

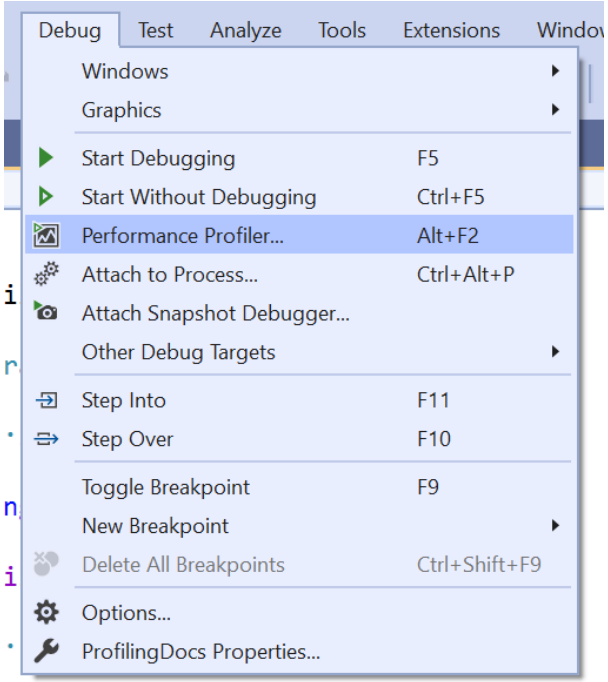
Choose the right Profiler

- Windows
  - VS Profiler – if you can reproduce the problem on your Dev machine
  - PerfView – very powerful, but has a high entry cost
- Cross platform
  - dotnet trace
    - Works always and everywhere with .NET Core 3.0+!
    - if you don’t need native call stacks
    - if you can’t run as Admin/sudo
- Linux
  - PerfCollect – if you need native call stacks and can run as sudo

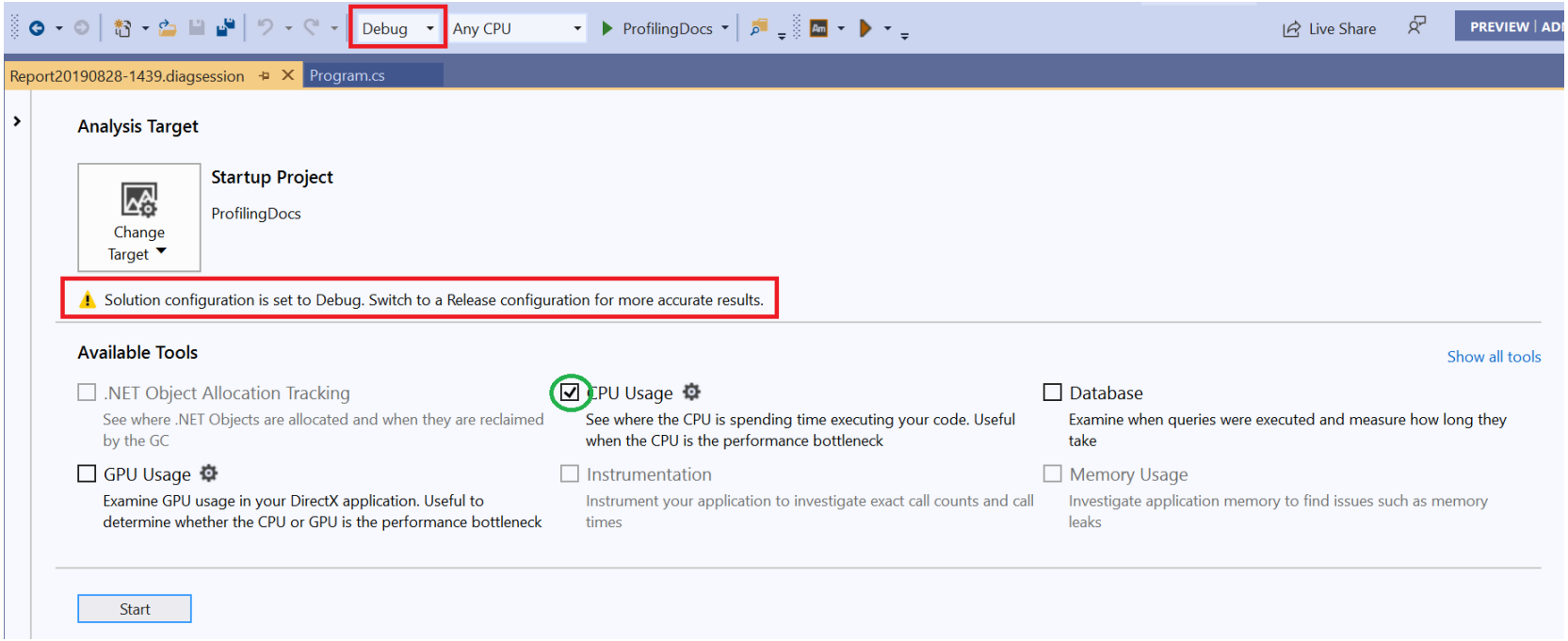
Visual Studio Profiler

- CPU, Memory, GPU
- **CPU time per code line!**
- .NET Core support

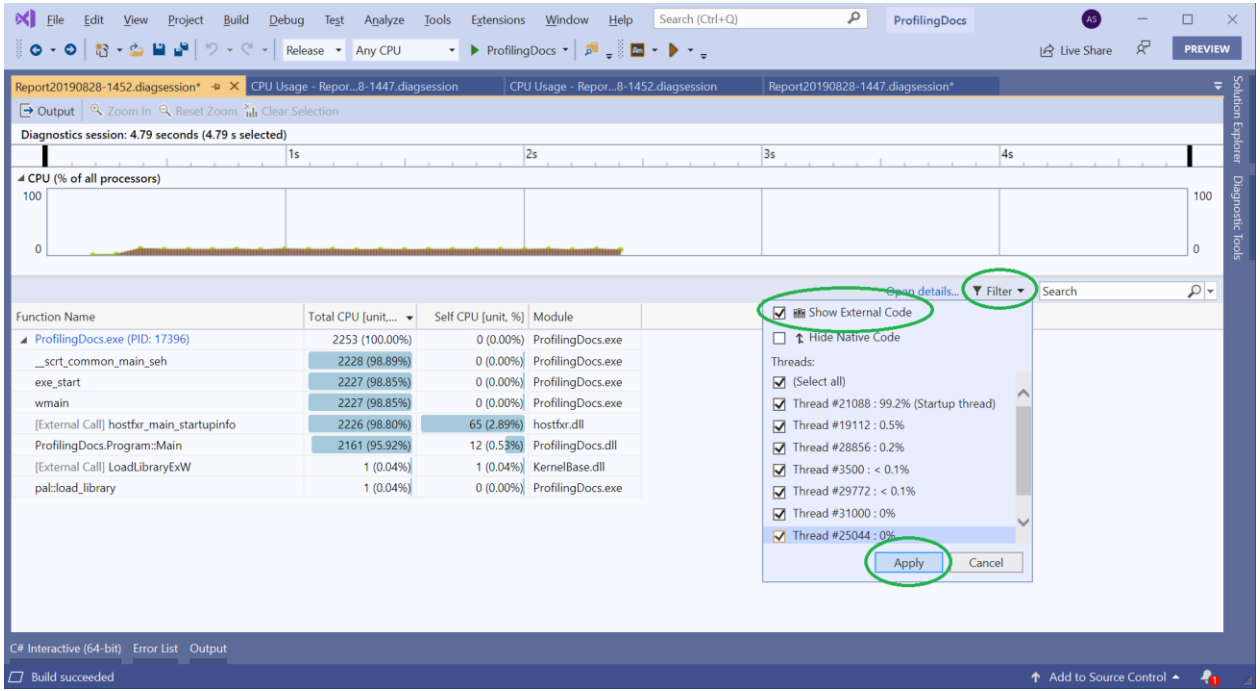
VS -> Alt+F2 or Debug Menu



Available Tools



Filter


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
ProfilingDocs
Release
Any CPU
ProfilingDocs
Live Share
PREVIEW
Report20190828-1452.diagsession*
CPU Usage - Repor...8-1447.diagsession
CPU Usage - Repor...8-1452.diagsession
Report20190828-1447.diagsession*
Output
Zoom In
Reset Zoom
Clear Selection
Diagnostics session: 4.79 seconds (4.79 s selected)
1s
2s
3s
4s
CPU (% of all processors)
100
0
Open details...
Filter
Search
Function Name
Total CPU [unit,...
Self CPU [unit, %]
Module
ProfilingDocs.exe (PID: 17396) 2253 (100.00%) 0 (0.00%) ProfilingDocs.exe
__scrt_common_main_seh 2228 (98.89%) 0 (0.00%) ProfilingDocs.exe
exe_start 2227 (98.85%) 0 (0.00%) ProfilingDocs.exe
wmain 2227 (98.85%) 0 (0.00%) ProfilingDocs.exe
[External Call] hostfxr_main_startupinfo 2226 (98.80%) 65 (2.89%) hostfxr.dll
ProfilingDocs.Program::Main 2161 (95.92%) 12 (0.53%) ProfilingDocs.dll
[External Call] LoadLibraryExW 1 (0.04%) 1 (0.04%) KernelBase.dll
pal::load_library 1 (0.04%) 0 (0.00%) ProfilingDocs.exe
Show External Code
Hide Native Code
Threads:
(Select all)
Thread #21088 : 99.2% (Startup thread)
Thread #19112 : 0.5%
Thread #28856 : 0.2%
Thread #3500 : < 0.1%
Thread #29772 : < 0.1%
Thread #31000 : 0%
Thread #25044 : 0%
Apply
Cancel
C# Interactive (64-bit)
Error List
Output
Build succeeded
Add to Source Control
Solution Explorer
Diagnostic Tools

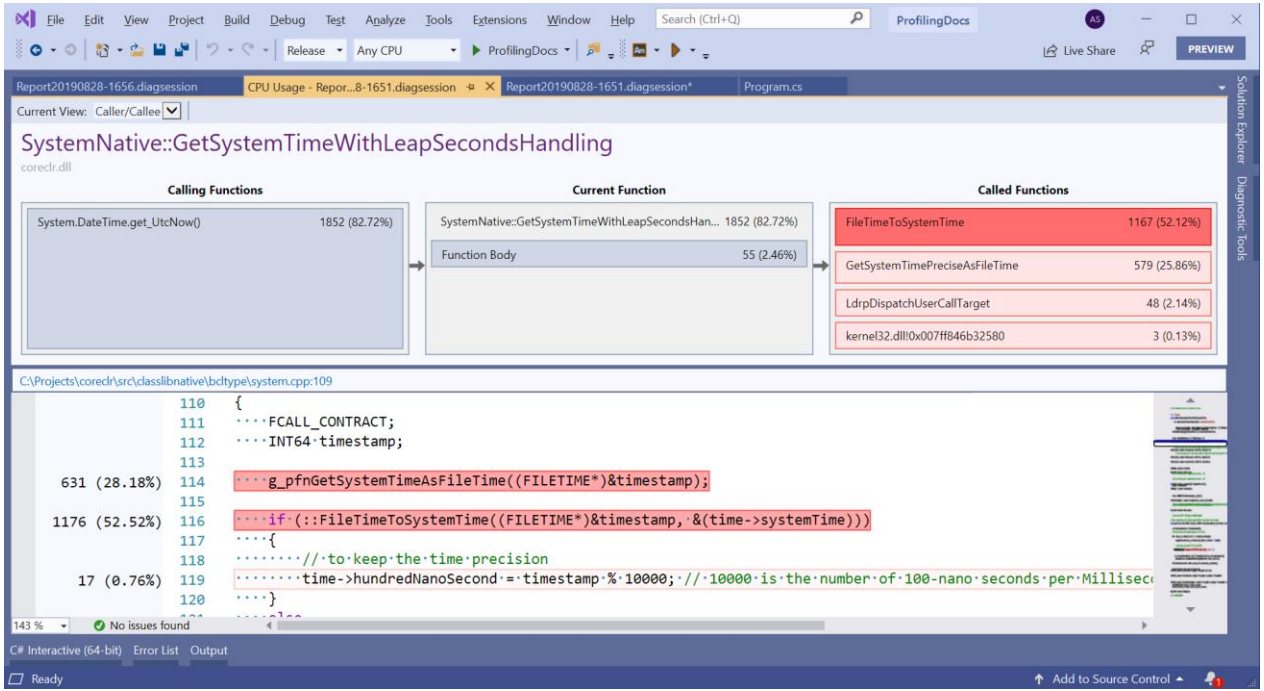
Analysis

coreclr.dll!0x007fffce570f01	2167 (96.18%)	0 (0.00%)	coreclr.dll
ProfilingDocs.Program::Main	2161 (95.92%)	12 (0.53%)	ProfilingDocs.dll
coreclr.dll!0x007fffce4c967a	2161 (95.92%)	0 (0.00%)	coreclr.dll
coreclr.dll!0x007fffce571163		0 (0.00%)	coreclr.dll

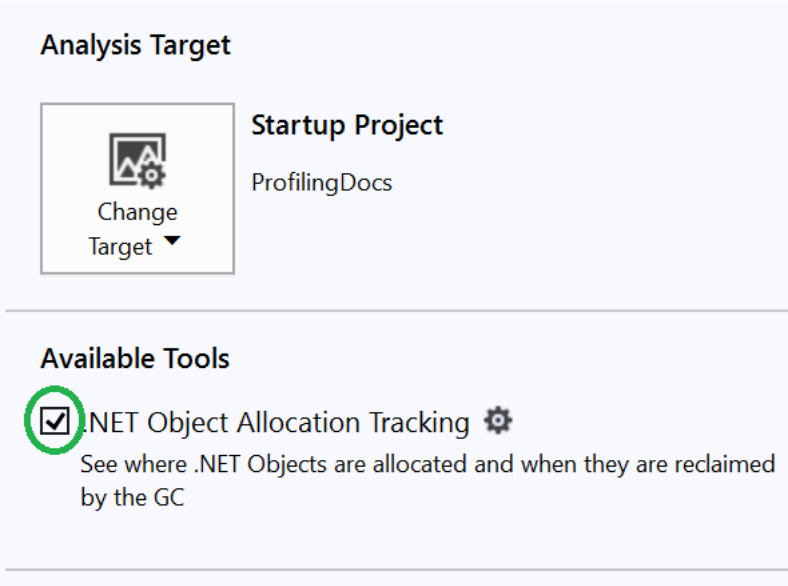
ProfilingDocs.Program::Main

Function Name	Total CPU [unit, %]	Self CPU [unit, %]	Module
ProfilingDocs.exe (PID: 17396)	2253 (100.00%)	0 (0.00%)	ProfilingDocs.exe
RtlpTimeToTimeFields	856 (37.99%)	856 (37.99%)	ntdll.dll
RtlQueryPerformanceCounter	351 (15.58%)	351 (15.58%)	ntdll.dll
FileTimeToSystemTime	1152 (51.13%)	282 (12.52%)	KernelBase.dll
RtlGetSystemTimePrecise	496 (22.02%)	145 (6.44%)	ntdll.dll
LdrpDispatchUserCallTarget	37 (1.64%)	37 (1.64%)	ntdll.dll
GetSystemTimePreciseAsFileTime	525 (23.30%)	29 (1.29%)	KernelBase.dll
coreclr.dll!0x007fffce7e51d0	15 (0.67%)	15 (0.67%)	coreclr.dll
__security_check_cookie	14 (0.62%)	14 (0.62%)	KernelBase.dll
System.Private.CoreLib.dll!0x007fffcdf18120	14 (0.62%)	14 (0.62%)	System.Private.Co...
ProfilingDocs.Program::Main	2161 (95.92%)	12 (0.53%)	ProfilingDocs.dll
System.Private.CoreLib.dll!0x007fffcdee5ee0	12 (0.53%)	12 (0.53%)	System.Private.Co...
System.Private.CoreLib.dll!0x007fffcdf18260	11 (0.49%)	11 (0.49%)	System.Private.Co...
System.Private.CoreLib.dll!0x007fffcdee5f3b	210 (9.32%)	10 (0.44%)	System.Private.Co...

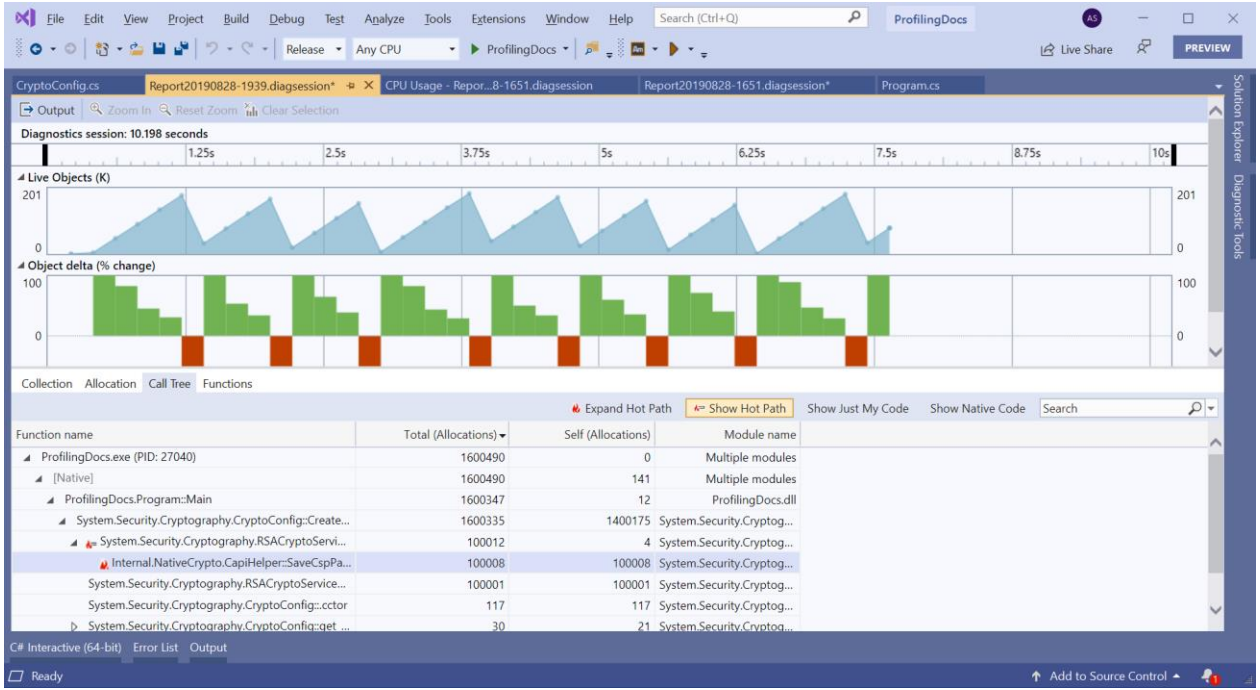
CPU time per code line!



.NET Object Allocation Tracking



Allocations!



Go to Source File

Collection Allocation Call Tree Functions

Function name	Total (Allocations)
ProfilingDocs.exe (PID: 27040)	1600490
[Native]	1600490
ProfilingDocs.Program::Main	1600347
System.Security.Cryptography.CryptoConfig::Create...	1600335
System.Security.Cryptography.RSACryptoServi...	100012
Internal.NativeCrypto.CapiHelper::SaveCspPa	100008
System.Security.Cryptography.RSACrypto	1
System.Security.Cryptography.CryptoCon	7
System.Security.Cryptography.CryptoCon	0

View in Functions
Go to Source File
Expand Hot Path
Copy

C# Interactive (64-bit) Error List Output

Ready

CapiHelper.Windows.cs CryptoConfig.cs Report20190828-1939.diagsession* CPU Usage - Repor...8-1651.diagsession

Miscellane

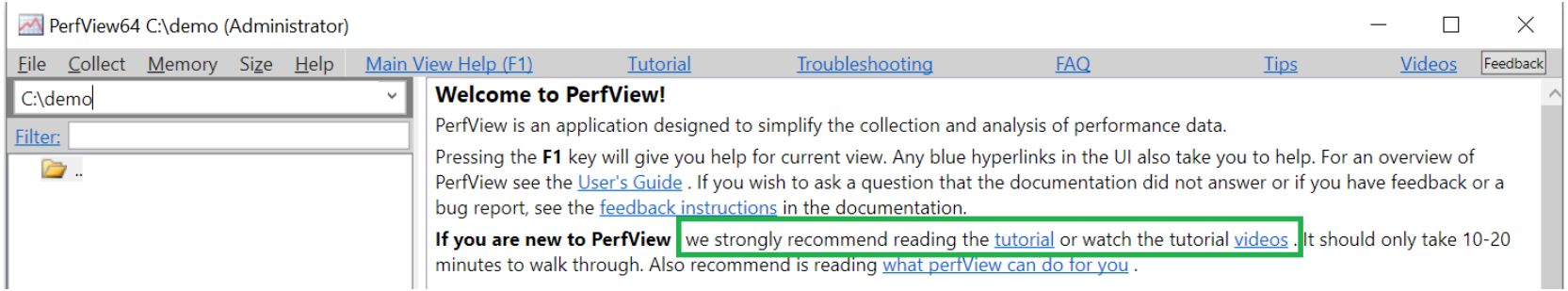
C:\Projects\corefx\src\System.Security.Cryptography.Csp\src\System\Security\Cryptography\CapiHelper.Windows.cs

654

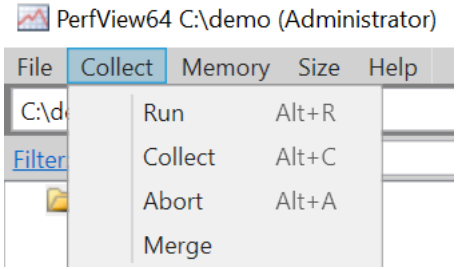
PerfView

- The most powerful .NET Profiler
- Actively Developed and Maintained
- Always up to date with latest .NET Runtime features
- Open Source https://github.com/Microsoft/perfview
- Small Overhead (up to 3-5%)
- Digitally signed by Microsoft
- Production Ready

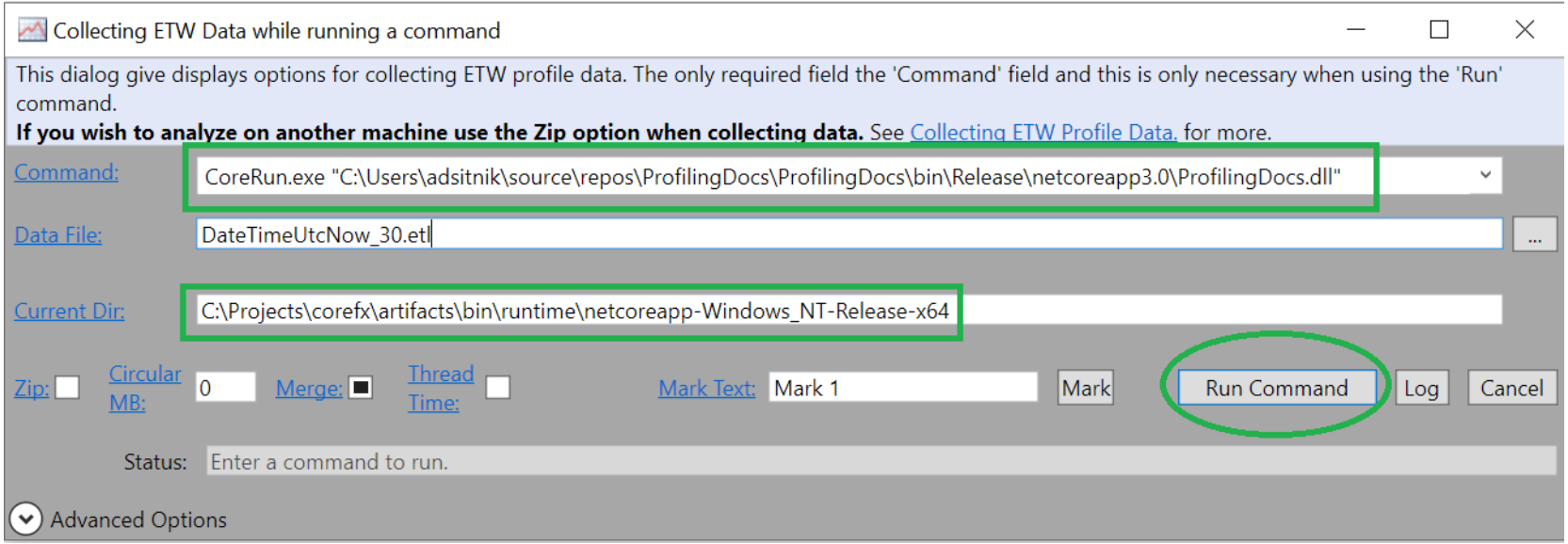
Free tutorial & videos!



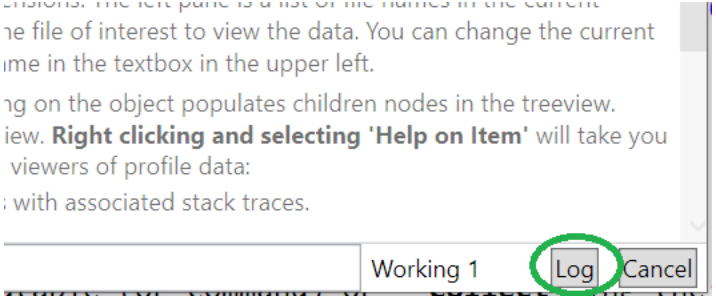
Run a Command or Collect w\ Start&Stop



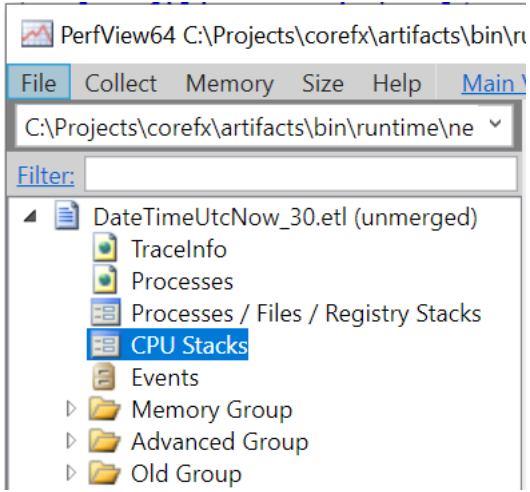
Small Repro App: Run an Executable



Experiencing a silent error? Log!



Trace Explorer



Select Process

Select Process Window

The data that was collect was machine wide, but typically you are interested in only one process. This dialog box allows you to choose a particular process to focus on by either double clicking on a process or by selecting a process and hitting OK. The 'All Procs' button can be used if you wish to look at all data. See Selecting A Process Help for more.

Name	ID	Parent	CPU MSec	Start	Duration	CommandLine
CoreRun	17900	31056	2,116	19-08-29 09:19:35	2.16 sec	"CoreRun.exe" "C:\Users\adsitnik\source\repos\Prc
firefox	6932	1420	1,073	19-08-29 09:19:35	5.26 sec	"C:\Program Files\Mozilla Firefox\firefox.exe" -cont
PerfView64	31056	8296	590	19-08-29 09:19:35	5.26 sec	"C:\Projects\perfview\src\PerfView\bin\Release\net
devenv	13320	8296	334	19-08-29 09:19:35	5.26 sec	"C:\Program Files (x86)\Microsoft Visual Studio\20
MsMpEng	4712	936	252	19-08-29 09:19:35	5.26 sec	"C:\ProgramData\Microsoft\Windows Defender\pl
System	4	-1	221	19-08-29 09:19:35	5.26 sec	
VBCSCompiler	15952	22336	216	19-08-29 09:19:35	5.26 sec	"C:\Program Files (x86)\Microsoft Visual Studio\20
firefox	12752	1420	174	19-08-29 09:19:35	5.26 sec	"C:\Program Files\Mozilla Firefox\firefox.exe" -cont
dwm	1432	1172	166	19-08-29 09:19:35	5.26 sec	"dwm.exe"
explorer	8296	8252	109	19-08-29 09:19:35	5.26 sec	C:\WINDOWS\Explorer.EXE
MsSense	4756	936	88	19-08-29 09:19:35	5.26 sec	"C:\Program Files\Windows Defender Advanced Th
firefox	1420	11532	76	19-08-29 09:19:35	5.26 sec	"C:\Program Files\Mozilla Firefox\firefox.exe" -osin

OK | All Procs | Cancel

Default Filters

CPU Stacks(2,117 metric) DateTimeUtcNow_30.etl in netcoreapp-Windows_NT-Release-x64 (C:\Projects\corefx\artifacts\bin\runtime\netcoreapp-Windows_NT-Release-x64\DateTimeUtcNow_30.etl)

File View Diff Regression Preset Help Stack View Help (F1) Understanding Perf Data Starting an Analysis Troubleshooting Tips

Update Back Forward Totals Metric: 2,117.0 Count: 2,117.0 First: 442.206 Last: 2,593.042 Last-First: 2,150.836 Metric/Interval: 0.98 TimeBucket: 100.6 TotalProcs 12

Start: 0 End: 3,219.360 Find:

GroupPats: [Just My App] \3.0.0\%!->;!=>OTHER Fold%: 1 FoldPats: ntoskrnl!%ServiceCopyEnd IncPats: Process% CoreRun (17900) ExcPats:

By Name ? Caller-Callee ? CallTree ? Callers ? Callees ? Flame Graph ? Notes ?

Name ?	Exc % ?	Exc ?	Inc % ?	Inc ?	Fold ?	When ?	First ?	Last ?
OTHER <<coreclr!CorHost2::ExecuteAssembly>>	92.6	1,961	92.6	1,961.0	0	99999AAA9A999999A996	607.483	2,578.764
OTHER <<ucrtbase!?>>	3.4	73	3.4	73.0	0	06 o	494.057	2,591.022
corerun!SString::Find	1.4	29	4.8	101.0	0	09	494.057	597.530
Process64 CoreRun (17900) Args: "C:\Users\adsitnik\source\repos\ProfilingDocs\ProfilingDocs\bin\Release\netcoreapp3.0\ProfilingDocs.d	0.8	17	100.0	2,117.0	17	4A99A99AAA9A999999A996	442.206	2,593.042
corerun!TryRun	0.8	17	98.4	2,084.0	17	2999999AAA9A999999A996	481.270	2,578.764
Thread (26688) CPU=2099ms (Startup Thread)	0.6	12	99.2	2,100.0	12	3999999AAA9A999999A996	442.206	2,593.042
corerun!HostEnvironment::AddFilesFromDirectoryToTPAList	0.2	5	5.0	106.0	5	19	492.057	598.531
BROKEN	0.1	2	0.1	2.0	2	0	2,591.036	2,593.042
OTHER <<ntdll!RtlUserThreadStart>>	0.0	1	98.5	2,086.0	0	2999999AAA9A999999A996	442.206	2,591.022
ROOT	0.0	0	100.0	2,117.0	0	4A99A99AAA9A999999A996	442.206	2,593.042
corerun!__scrt_common_main_seh	0.0	0	98.5	2,085.0	0	2999999AAA9A999999A996	481.270	2,591.022
corerun!wmain	0.0	0	98.4	2,084.0	0	2999999AAA9A999999A996	481.270	2,578.764
corerun!HostEnvironment::GetTpaList	0.0	0	5.0	106.0	0	19	492.057	598.531

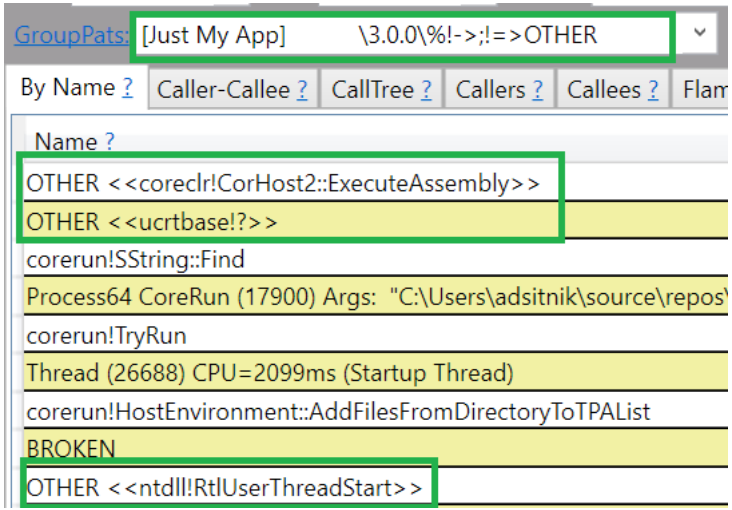
Is it CPU bound?

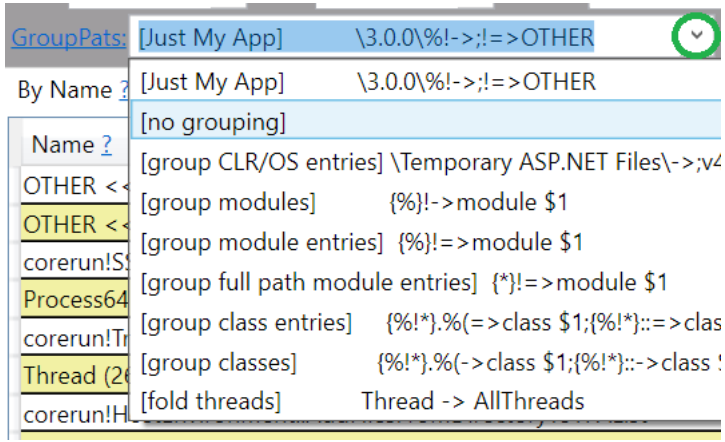
7 metric) DateTimeUtcNow_30.etl in netcoreapp-Windows_NT-Release-x64 (C:\Projects\corefx\artifacts\bin\runtime\netcoreapp-Windows_NT-Relea

Regression Preset Help Stack View Help (F1) Understanding Perf Data Starting an Analysis

Totals Metric: 2,117.0 Count: 2,117.0 First: 442.206 Last: 2,593.042 Last-First: 2,150.836 Metric/Interval: 0.98 TimeBucket: 100.6 TotalProcs 12

Grouping


GroupPats: [Just My App] \3.0.0\%!->;!=>OTHER
By Name ?
Caller-Callee ?
CallTree ?
Callers ?
Callees ?
Name ?
OTHER <<coreclr!CorHost2::ExecuteAssembly>>
OTHER <<ucrtbase!?>>
corerun!SString::Find
Process64 CoreRun (17900) Args: "C:\Users\adsitnik\source\repos
corerun!TryRun
Thread (26688) CPU=2099ms (Startup Thread)
corerun!HostEnvironment::AddFilesFromDirectoryToTPAList
BROKEN
OTHER <<ntdll!RtlUserThreadStart>>


GroupPats: [Just My App] \3.0.0\%!->;!=>OTHER
By Name ?
Name ?
[Just My App] \3.0.0\%!->;!=>OTHER
[no grouping]
[group CLR/OS entries] \Temporary ASP.NET Files\->;v4
[group modules] {%}!->module $1
[group module entries] {%}!=>module $1
[group full path module entries] {*}!=>module $1
[group class entries] {%!*}.%(=>class $1;{%!*}::=>clas
[group classes] {%!*}.%(->class $1;{%!*}::->class
[fold threads] Thread -> AllThreads

No grouping

GroupPats: [no grouping] Fold%: 1 Fold

By Name ? | Caller-Callee ? | CallTree ? | Callers ? | Callees ? | Flame Graph ? | Notes ?

Name ?
ntdll!RtlpTimeToTimeFields
ntdll!RtlQueryPerformanceCounter
kernelbase!FileTimeToSystemTime
system.private.corelib.il!System.DateTime.get_UtcNow()
ntdll!RtlGetSystemTimePrecise
coreclr!SystemNative::GetSystemTimeWithLeapSecondsHandling
ucrtbase!?
kernelbase!GetSystemTimePreciseAsFileTime
ntdll!LdrpDispatchUserCallTarget
corerun!SString::Find
system.private.corelib!System.DateTime.get_UtcNow()
profilingdocs!ProfilingDocs.Program.Main()
corerun!TryRun
Process64 CoreRun (17900) Args: "C:\Users\adsitnik\source\repos\ProfilingDocs\Profilin
?!?
Thread (26688) CPU=2099ms (Startup Thread)
coreclr!CallDescrWorkerInternal
corerun!HostEnvironment::AddFilesFromDirectoryToTPAList
ntoskrnl!?

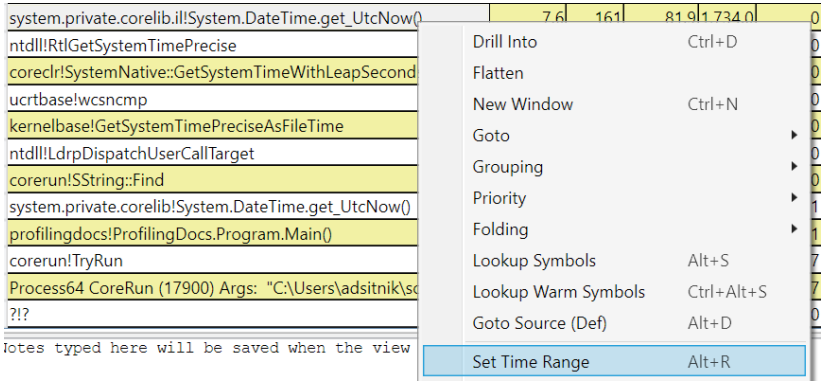
Symbols

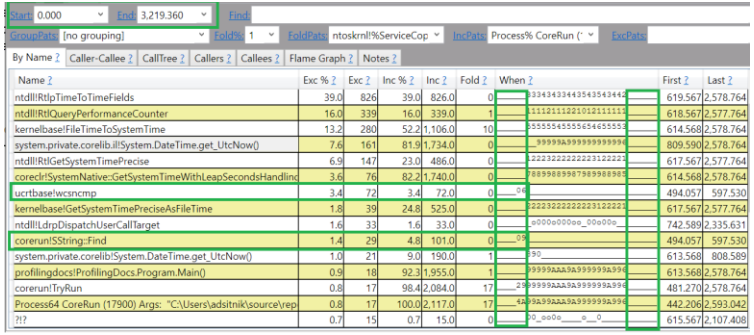
ucrtbase!?
kernelbase!GetSystemTim
ntdll!LdrpDispatchUserCa
corerun!SString::Find
system.private.corelib!Sys
profilingdocs!ProfilingDoc
corerun!TryRun
Process64 CoreRun (1790
?!?
Thread (26688) CPU=2099
coreclr!CallDescrWorkerIn
corerun!HostEnvironment
ntoskrnl!?

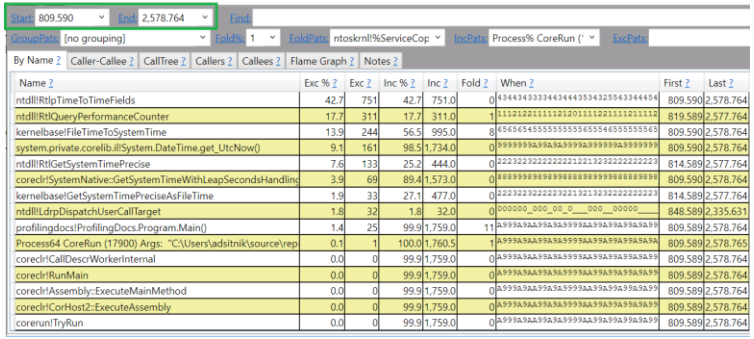
Name ?	Exc % ?	Exc ?	Inc %
ntdll!RtlpTimeToTimeFields	39.0	826	3
ntdll!RtlQueryPerformanceCounter	16.0	338	1
kernelbase!FileTimeToSystemTime	13.2	279	5
system.private.corelib.il!System.DateTime.get_UtcNow()	7.6	161	8
ntdll!RtlGetSystemTimePrecise	6.9	147	2
coreclr!SystemNative::GetSystem			8
ucrtbase!?			
kernelbase!GetSystemTimePrecis			2
ntdll!LdrpDispatchUserCallTarget			
corerun!SString::Find			
system.private.corelib!System.Da			
profilingdocs!ProfilingDocs.Progr			9
corerun!TryRun			9
Process64 CoreRun (17900) Args:			10

Drill Into	Ctrl+D
Flatten	
New Window	Ctrl+N
Goto	▸
Grouping	▸
Priority	▸
Folding	▸
Lookup Symbols	Alt+S

Set Time Range







Filtering by Name

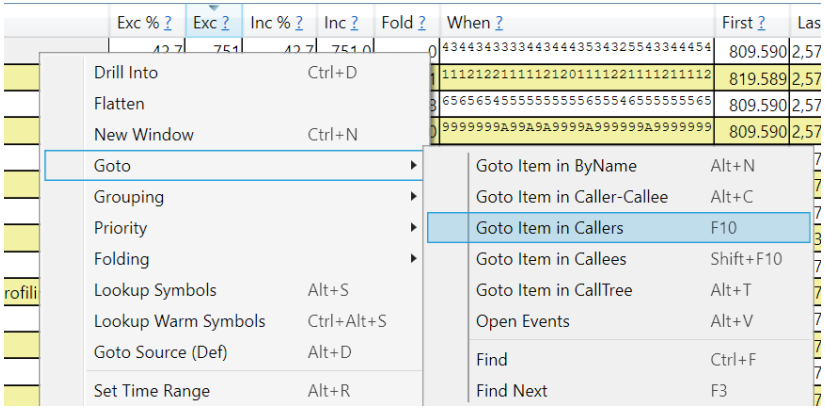
```
void Main() => WhatYouCareAbout(Setup());

[MethodImpl(MethodImplOptions.NoInlining)]
void $SomeType Setup() …

[MethodImpl(MethodImplOptions.NoInlining)]
void WhatYouCareAbout($SomeType initialized) …
```

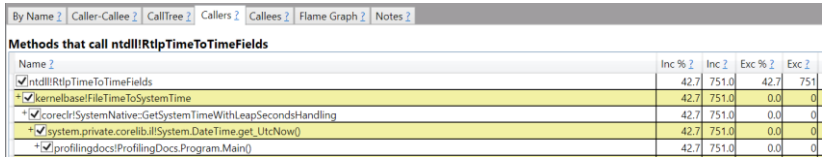
By Name tab: Bottom-Up analysis

By Name ? | Caller-Callee ? | CallTree ? | Callers ? | Callees ? | Flame Graph ? | Notes ?

Name ?	Exc % ?	Exc ?	Inc % ?	Inc ?
ntdll!RtlpTimeToTimeFields	42.7	751	42.7	751.0
ntdll!RtlQueryPerformanceCounter	17.7	311	17.7	311.0
kernelbase!FileTimeToSystemTime	13.9	244	56.5	995.0
system.private.corelib.il!System.DateTime.get_UtcNow()	9.1	161	98.5	1,734.0
ntdll!RtlGetSystemTimePrecise	7.6	133	25.2	444.0
coreclr!SystemNative::GetSystemTimeWithLeapSecondsHandling	3.9	69	89.4	1,573.0
kernelbase!GetSystemTimePreciseAsFileTime	1.9	33	27.1	477.0
ntdll!LdrpDispatchUserCallTarget	1.8	32	1.8	32.0
profilingdocs!ProfilingDocs.Program.Main()	1.4	25	99.9	1,759.0
Process64 CoreRun (17900) Args: "C:\Users\adsitnik\source\rep	0.1	1	100.0	1,760.5

Callers tab


Exc % ?
Exc ?
Inc % ?
Inc ?
Fold ?
When ?
First ?
Las
809.590
819.589
809.590
809.590
Drill Into Ctrl+D
Flatten
New Window Ctrl+N
Goto
Grouping
Priority
Folding
Lookup Symbols Alt+S
Lookup Warm Symbols Ctrl+Alt+S
Goto Source (Def) Alt+D
Set Time Range Alt+R
Goto Item in ByName Alt+N
Goto Item in Caller-Callee Alt+C
Goto Item in Callers F10
Goto Item in Callees Shift+F10
Goto Item in CallTree Alt+T
Open Events Alt+V
Find Ctrl+F
Find Next F3

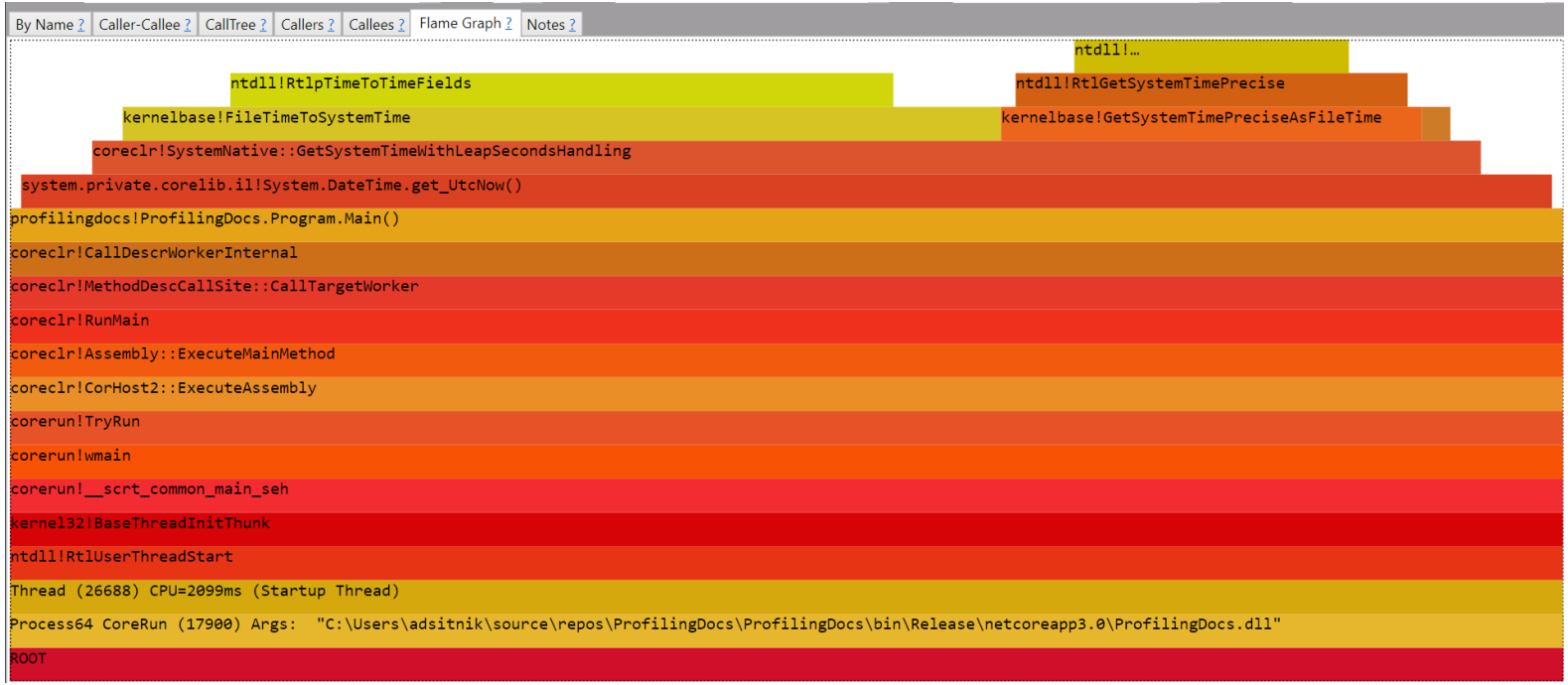

By Name ? Caller-Callee ? CallTree ? Callers ? Callees ? Flame Graph ? Notes ?
Methods that call ntdll!RtlpTimeToTimeFields
Name ? Inc % ? Inc ? Exc % ? Exc ?
ntdll!RtlpTimeToTimeFields 42.7 751.0 42.7 751
kernelbase!FileTimeToSystemTime 42.7 751.0 0.0 0
coreclr!SystemNative::GetSystemTimeWithLeapSecondsHandling 42.7 751.0 0.0 0
system.private.corelib.il!System.DateTime.get_UtcNow() 42.7 751.0 0.0 0
profilingdocs!ProfilingDocs.Program.Main() 42.7 751.0 0.0 0

CallTree

By Name ? | Caller-Callee ? | CallTree ? | Callers ? | Callees ? | Flame Graph ? | Notes ?

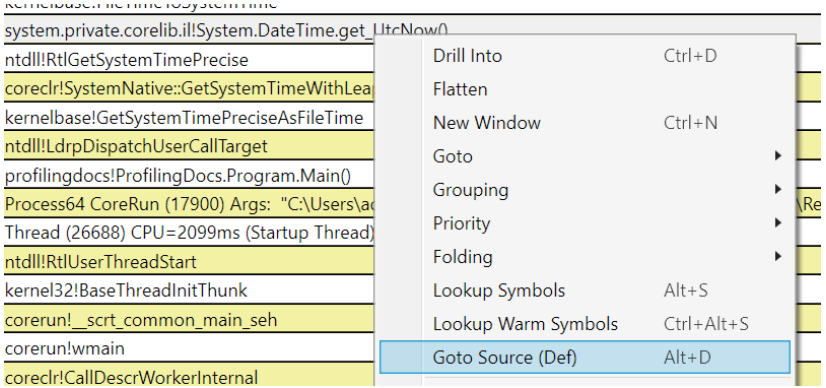
Name ?	Inc % ?	Inc ?	Exc % ?	Exc ?
ROOT	100.0	1,760.5	0.0	0
Process64 CoreRun (17900) Args: "C:\Users\adsitnik\source\repos\ProfilingDocs\ProfilingDocs\bin\Release\netcoreapp3.0\ProfilingDoc	100.0	1,760.5	0.1	1
Thread (26688) CPU=2099ms (Startup Thread)	99.9	1,759.0	0.0	0
ntdll!RtlUserThreadStart	99.9	1,759.0	0.0	0
kernel32!BaseThreadInitThunk	99.9	1,759.0	0.0	0
corerun!__scrt_common_main_seh	99.9	1,759.0	0.0	0
corerun!wmain	99.9	1,759.0	0.0	0
corerun!TryRun	99.9	1,759.0	0.0	0
coreclr!CorHost2::ExecuteAssembly	99.9	1,759.0	0.0	0
coreclr!Assembly::ExecuteMainMethod	99.9	1,759.0	0.0	0
coreclr!RunMain	99.9	1,759.0	0.0	0
coreclr!MethodDescCallSite::CallTargetWorker	99.9	1,759.0	0.0	0
coreclr!CallDescrWorkerInternal	99.9	1,759.0	0.0	0
profilingdocs!ProfilingDocs.Program.Main()	99.9	1,759.0	1.4	25
system.private.corelib.il!System.DateTime.get_UtcNow()	98.5	1,734.0	9.1	161
coreclr!SystemNative::GetSystemTimeWithLeapSecondsHandling	89.4	1,573.0	3.9	69
kernelbase!FileTimeToSystemTime	56.5	995.0	13.9	244
ntdll!RtlpTimeToTimeFields	42.7	751.0	42.7	751
kernelbase!GetSystemTimePreciseAsFileTime	27.1	477.0	1.9	33
ntdll!RtlGetSystemTimePrecise	25.2	444.0	7.6	133
ntdll!RtlQueryPerformanceCounter	17.7	311.0	17.7	311
ntdll!LdrpDispatchUserCallTarget	1.8	32.0	1.8	32

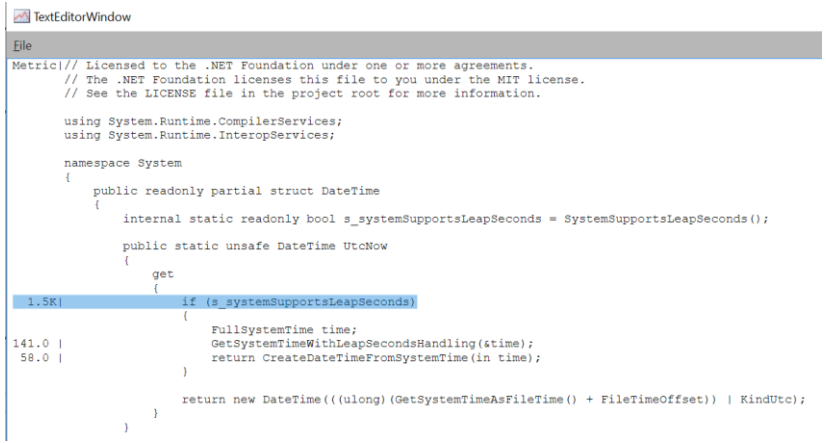
Flame Graph!


By Name
Caller-Callee
CallTree
Callers
Callees
Flame Graph
Notes
ntdll!…
ntdll!RtlpTimeToTimeFields
ntdll!RtlGetSystemTimePrecise
kernelbase!FileTimeToSystemTime
kernelbase!GetSystemTimePreciseAsFileTime
coreclr!SystemNative::GetSystemTimeWithLeapSecondsHandling
system.private.corelib.il!System.DateTime.get_UtcNow()
profilingdocs!ProfilingDocs.Program.Main()
coreclr!CallDescrWorkerInternal
coreclr!MethodDescCallSite::CallTargetWorker
coreclr!RunMain
coreclr!Assembly::ExecuteMainMethod
coreclr!CorHost2::ExecuteAssembly
corerun!TryRun
corerun!wmain
corerun!__scrt_common_main_seh
kernel32!BaseThreadInitThunk
ntdll!RtlUserThreadStart
Thread (26688) CPU=2099ms (Startup Thread)
Process64 CoreRun (17900) Args: "C:\Users\adsitnik\source\repos\ProfilingDocs\ProfilingDocs\bin\Release\netcoreapp3.0\ProfilingDocs.dll"
ROOT

Understanding Flame Graph

Goto Source

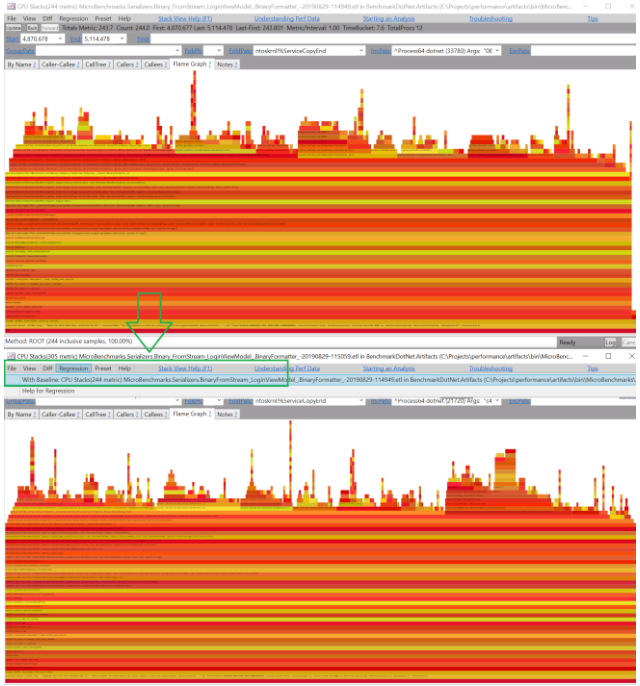

system.private.corelib.il!System.DateTime.get_UtcNow()
ntdll!RtlGetSystemTimePrecise
coreclr!SystemNative::GetSystemTimeWithLea
kernelbase!GetSystemTimePreciseAsFileTime
ntdll!LdrpDispatchUserCallTarget
profilingdocs!ProfilingDocs.Program.Main()
Process64 CoreRun (17900) Args: "C:\Users\a
Thread (26688) CPU=2099ms (Startup Thread)
ntdll!RtlUserThreadStart
kernel32!BaseThreadInitThunk
corerun!__scrt_common_main_seh
corerun!wmain
coreclr!CallDescrWorkerInternal
Drill Into Ctrl+D
Flatten
New Window Ctrl+N
Goto
Grouping
Priority
Folding
Lookup Symbols Alt+S
Lookup Warm Symbols Ctrl+Alt+S
Goto Source (Def) Alt+D


TextEditorWindow
File
Metric
// Licensed to the .NET Foundation under one or more agreements.
// The .NET Foundation licenses this file to you under the MIT license.
// See the LICENSE file in the project root for more information.
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
namespace System
{
public readonly partial struct DateTime
{
internal static readonly bool s_systemSupportsLeapSeconds = SystemSupportsLeapSeconds();
public static unsafe DateTime UtcNow
{
get
{
1.5K
if (s_systemSupportsLeapSeconds)
{
FullSystemTime time;
141.0
GetSystemTimeWithLeapSecondsHandling(&time);
58.0
return CreateDateTimeFromSystemTime(in time);
}
return new DateTime(((ulong)(GetSystemTimeAsFileTime() + FileTimeOffset)) | KindUtc);
}
}

Pro Tip: Start & Stop Events

```
[EventSource(Name = "DemoEventSource")]
sealed class DemoEventSource : EventSource
{
    public void DemoStart() => WriteEvent(1);
    public void DemoStop() => WriteEvent(2);

    public static DemoEventSource Log = new DemoEventSource();
}

if (DemoEventSource.Log.IsEnabled())
    DemoEventSource.Log.DemoStart();

// thing that you care about

if (DemoEventSource.Log.IsEnabled())
    DemoEventSource.Log.DemoStop();
```

Regressions



Regression Report between and

Back Forward Click in Window, Ctrl-F for find

Overweight report for symbols common between both files

Base (old) Time:	243.7
Test (new) Time:	304.8
Delta:	61.0
Delta %:	25.0

In this report, overweight is ratio of actual growth compared to 25.0%.
Interest level attempts to identify smaller methods which changed a lot. These are likely the most interesting frames to sart investigating
An overweight of greater than 100% indicates the symbol grew in cost more than average.
High overweights are a likely source of regressions and represent good places to investigate.
Only symbols that have at least 2% impact are shown.

Name	Base	Test	Delta	Responsibility %	Overweight %	Interest Level
system.runtime.serialization.formatters.il!System.Runtime.Serialization.Formatters.Binary.BinaryParser.ReadObjectWithMapTyped(value class System.Runtime.Serialization.Formatters.Binary.BinaryHeaderEnum)	109.0	40.0	-69.0	-113.08	-252.87	6
coreclr!Thread::StackWalkFrames	1.0	50.8	49.8	81.55	19877.52	5
coreclr!Thread::StackWalkFramesEx	1.0	45.8	44.8	73.36	17880.21	5
coreclr!WKS::GCHeap::Alloc	4.0	24.0	20.0	32.78	1997.31	5
system.runtime.serialization.formatters.il!System.Runtime.Serialization.Formatters.Binary.ObjectReader.Parse(class System.Runtime.Serialization.Formatters.Binary.ParseRecord)	56.0	1.0	-55.0	-90.14	-392.33	5
system.runtime.serialization.formatters.il!System.Runtime.Serialization.Formatters.Binary.ReadObjectInfo.GetObjectInfo(class System.Runtime.Serialization.Formatters.Binary.SerObjectInfoInit)	1.0	7.0	6.0	9.83	2396.77	4
coreclr!ReflectionSerialization::GetUninitializedObject	1.0	6.0	5.0	8.19	1997.31	4
coreclr!JIT_NewArr1	4.0	15.0	11.0	18.03	1098.52	4

dotnet trace

- Cross platform
- .NET Core 3.0+
- No need to run as Admin|sudo
- Lacks native call stacks

Simple commands

- dotnet tool install --global dotnet-trace
- dotnet trace list-processes
- dotnet trace collect -p $pid
- dotnet trace convert $inputFile --format speedscope
- dotnet trace collect -p $pid --format speedscope

PerfCollect

- Script, located at https://aka.ms/perfcollect
- Has an excellent docs
- **PerfCollect uses perf, which gives you native callstacks. dotnet-trace can only give you managed callstacks**.
- Knows how to install its dependencies
- Has a machine-wide scope
- PerfCollect can be started prior to the process start, whereas dotnet-trace can only be attached to a running process.
- Produces a zip file that can be opened with PerfView

Installation

```
curl -OL https://aka.ms/perfcollect
chmod +x perfcollect
sudo ./perfcollect install
```

Sample Usage

adsitnik@adsitnikUbuntu1804: ~/Projects/coreclr/bin/tests/Linux.x64.Release/Tests/Core_Root

```
adsitnik@adsitnikUbuntu1804:~/Projects/coreclr/bin/tests/Linux.x64.Release/Tests/Core_Root$ ./corerun ./ProfilingDocs.dll
```

adsitnik@adsitnikUbuntu1804: ~/Projects/tracing

```
adsitnik@adsitnikUbuntu1804:~/Projects/tracing$ sudo ./perfcollect collect slowStartsWith
```

When you find the bottleneck

- Try to get the big picture.
- Question the current design:
  - Why do we do that?
  - Can we use it less frequently?
  - Does faster alternative exist?
- Architecture changes require more effort but can boost the perf more than any micro-optimizations.

You have an idea.
What is the next step?

Correctness > Performance

- Make sure the code has good test coverage before you try to tune the perf.
- Ask for a detailed code review:
  - Explain your decisions, share the perf knowledge.
  - Make sure your changes don’t affect the results.
- Don’t push on the reviewers.

Example: string.StartsWith

dotnet / coreclr

Watch 1,012 | Unstar 12,107 | Fork 2,849

Code | Issues 2,007 | Pull requests 166 | Actions | Projects 9 | Wiki | Security | Insights

implement StartsWith and EndsWith as calls to CompareString for sliced string #26481

Edit

Closed adamsitnik wants to merge 1 commit into dotnet:master from adamsitnik:fasterStartsWith

Conversation 2 | Commits 1 | Checks 53 | Files changed 4 | +10 −338

Changes from all commits | File filter... | Jump to... | 0 / 4 files viewed | Review changes

16 ...stem.Private.CoreLib/shared/Interop/Unix/System.Globalization.Native/Interop.Collation.cs — Viewed

230 src/System.Private.CoreLib/shared/System/Globalization/CompareInfo.Unix.cs — Viewed

```
@@ -498,15 +498,9 @@ private bool StartsWith(string source, string prefix, CompareOptions options)
498 498             Debug.Assert(!string.IsNullOrEmpty(source));
499 499             Debug.Assert(!string.IsNullOrEmpty(prefix));
500 500             Debug.Assert((options & (CompareOptions.Ordinal | CompareOptions.OrdinalIgnoreCase)) == 0);
    501 +           Debug.Assert(source.Length >= prefix.Length);
501 502
502     -       #if CORECLR
503     -           if (_isAsciiEqualityOrdinal && CanUseAsciiOrdinalForOptions(options) && source.IsFastSort() && prefix.IsFastSort())
504     -           {
505     -               return IsPrefix(source, prefix, GetOrdinalCompareOptions(options));
506     -           }
507     -       #endif
508     -
509     -           return Interop.Globalization.StartsWith(_sortHandle, prefix, prefix.Length, source, source.Length, options);
    503 +           return CompareString(source.AsSpan(0, prefix.Length), prefix, options) == 0;
510 504         }
511 505
```

kevingosse on 2 Sep • edited Contributor

I don't think you can make this assumption in the non-ordinal case.

```
Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("fr-FR");

 string str1 = "œ";
 string str2 = "oe";

Console.WriteLine(str1.StartsWith(str2, StringComparison.CurrentCulture));
```

This should print true even though str2.Length > str1.Length

3

adamsitnik on 3 Sep Author Member

let's say that this was not my best idea ;p

string.StartsWith: Edge Cases

Culture	Source	Prefix	Windows	Unix	Comment
fr-FR	œ	oe	True	False	
hu-HU	dz	d	False	False	
pl-PL	cz	c	True	False	
	o\u0308	o	False	False	Combining character
	o\u0000\u0308	o	True	True	NULL (0) char
	\uD800\uDC00	\uD800	True	False	Surrogates
	b	new string('a', UInt16.MaxValue + 1)	False	True	18y old bug in ICU

You have an idea and good tests.
What is the next step?

Benchmarks!

- Write benchmarks to validate the gains.
- Keep them and measure the perf over time!

while (!goals.Achieved)

- Apply the optimizations
- Run the tests, verify the correctness
- Run the benchmarks, verify the gains
- Profile and analyze the data

Summary

- Have a small repro, try to narrow down the problem.
- Release + debug symbols
- Windows
  - Development - Visual Studio Profiler
  - Production - PerfView
- Linux
  - dotnet trace – default choice
  - PerfCollect – if you really need native call stacks or machine-wide
- Analyze
- Correctness over performance
- Use benchmarks to validate the gains

Questions?

Thank you!

@SitnikAdam

Adam.Sitnik@microsoft.com