PowerShell Conference Europe

Going past Profiler, profiling PowerShell code using PerfView.

Jakub Jareš

PRAGUE23

Many thanks to our sponsors:

Centino systems

PATCH MY PC

ScriptRunner®
The #1 for PowerShell Management

@nohwnd

Jakub Jareš

PSCONF.EU
PRAGUE23

- Pester owner and maintainer. Author of Assert module, and Profiler module. Senior software engineer at Microsoft, developing VSTest and MSTest. I don't represent Microsoft here.

@nohwnd

Need info about Prague, ask me.

Prizes!!!

https://rb.gy/kk77g

PSCONF.EU

PRAGUE23

My Math module is broken 😢

@nohwnd

Profiler

```
S:\p\profiler [tracesource +2 ~2 -0 !]> $r.Top50SelfDuration | select SelfPercent, SelfDuration, HitCount, File, Module, Function, Text | ft

SelfPercent SelfDuration     HitCount File          Module  Function          Text
----------- ------------     -------- ----          ------  --------          ----
      98.29 00:00:00.6597939        4 Avocado.psm1  Avocado Get-EmojiInternal $r = Invoke-WebRequest -Method GET -Uri "https://emojipedia.org/$(
       1.44 00:00:00.0097032        2 Avocado.psm1  Avocado Get-EmojiInternal $match = $r.Content.ToString() -split "`n" | Select-String '<h1><s
       0.22 00:00:00.0014554        2 demo.ps1                                & "$PSScriptRoot/demo-scripts/Get-Icons.ps1" }
       0.01 00:00:00.0001037        1 Get-Icons.ps1                           Import-Module $PSScriptRoot\Avocado.psm1
       0.01 00:00:00.0000425        2 Avocado.psm1  Avocado Get-EmojiInternal $match.Matches.Groups[-1].Value
       0.01 00:00:00.0000350        1 Get-Icons.ps1         Get-Icons         Get-Unicorn
       0.00 00:00:00.0000216        1 Avocado.psm1  Avocado Get-Unicorn       Get-EmojiInternal -Emoji unicorn
       0.00 00:00:00.0000209        2 Avocado.psm1  Avocado Get-EmojiInternal if (200 -ne $r.StatusCode) {
       0.00 00:00:00.0000186        1 Get-Icons.ps1                           Get-Icons
       0.00 00:00:00.0000177        1 Avocado.psm1  Avocado Get-Avocado       Get-EmojiInternal -Emoji avocado
       0.00 00:00:00.0000167        2 Avocado.psm1  Avocado Get-EmojiInternal }
       0.00 00:00:00.0000136        1 Get-Icons.ps1         Get-Icons         Get-Avocado
       0.00 00:00:00.0000086        1 Get-Icons.ps1         Get-Icons         }
       0.00 00:00:00.0000062        2 Avocado.psm1  Avocado Get-EmojiInternal $ProgressPreference = 'SilentlyContinue'
       0.00 00:00:00.0000050        1 Avocado.psm1  Avocado Get-Avocado       }
       0.00 00:00:00.0000046        1 Avocado.psm1  Avocado Get-Unicorn       }
       0.00 00:00:00.0000045        2 Avocado.psm1  Avocado Get-EmojiInternal function Get-EmojiInternal ($Emoji) {
       0.00 00:00:00.0000029        1 Get-Icons.ps1         Get-Icons         function Get-Icons () {
       0.00 00:00:00.0000020        1 Avocado.psm1  Avocado Get-Unicorn       function Get-Unicorn {
       0.00 00:00:00.0000020        1 demo.ps1                                {
       0.00 00:00:00.0000018        1 Avocado.psm1  Avocado Get-Avocado       function Get-Avocado {
```

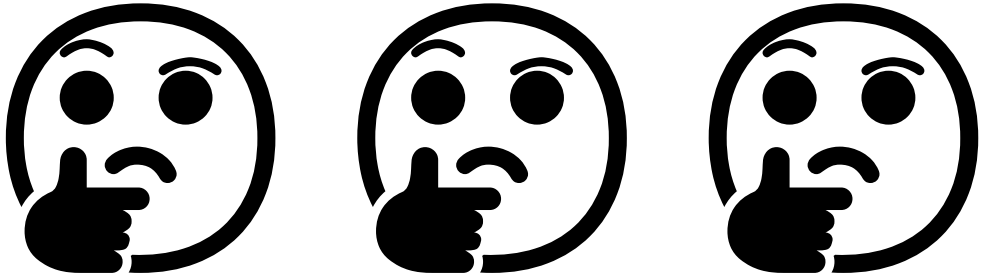
[nohwnd/Profiler: Script, ScriptBlock and module performance profiler for PowerShell 5, and PowerShell 7. (github.com)]

@nohwnd

Profiler limitations

- Can only see PowerShell code.
- Can see only entry point of C# code (e.g. binary cmdlet, [IO.File]::ReadAllText()) but not the internals.
- Can't see PowerShell internals.

Often this is blessing, but sometimes it is a curse.



We need to go deeper.

PSCONF.EU

PRAGUE23

@nohwnd

PSCONF.EU

PRAGUE23

PerfView

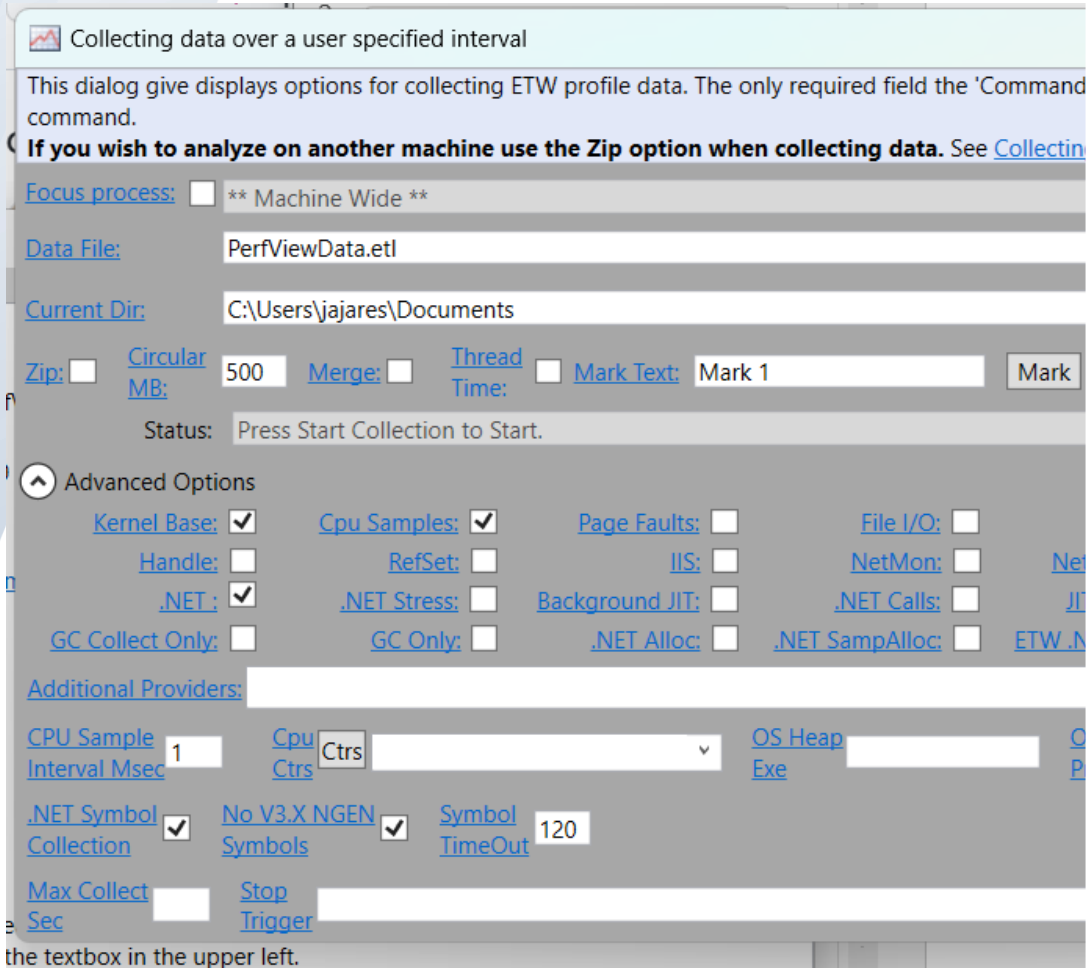
@nohwnd

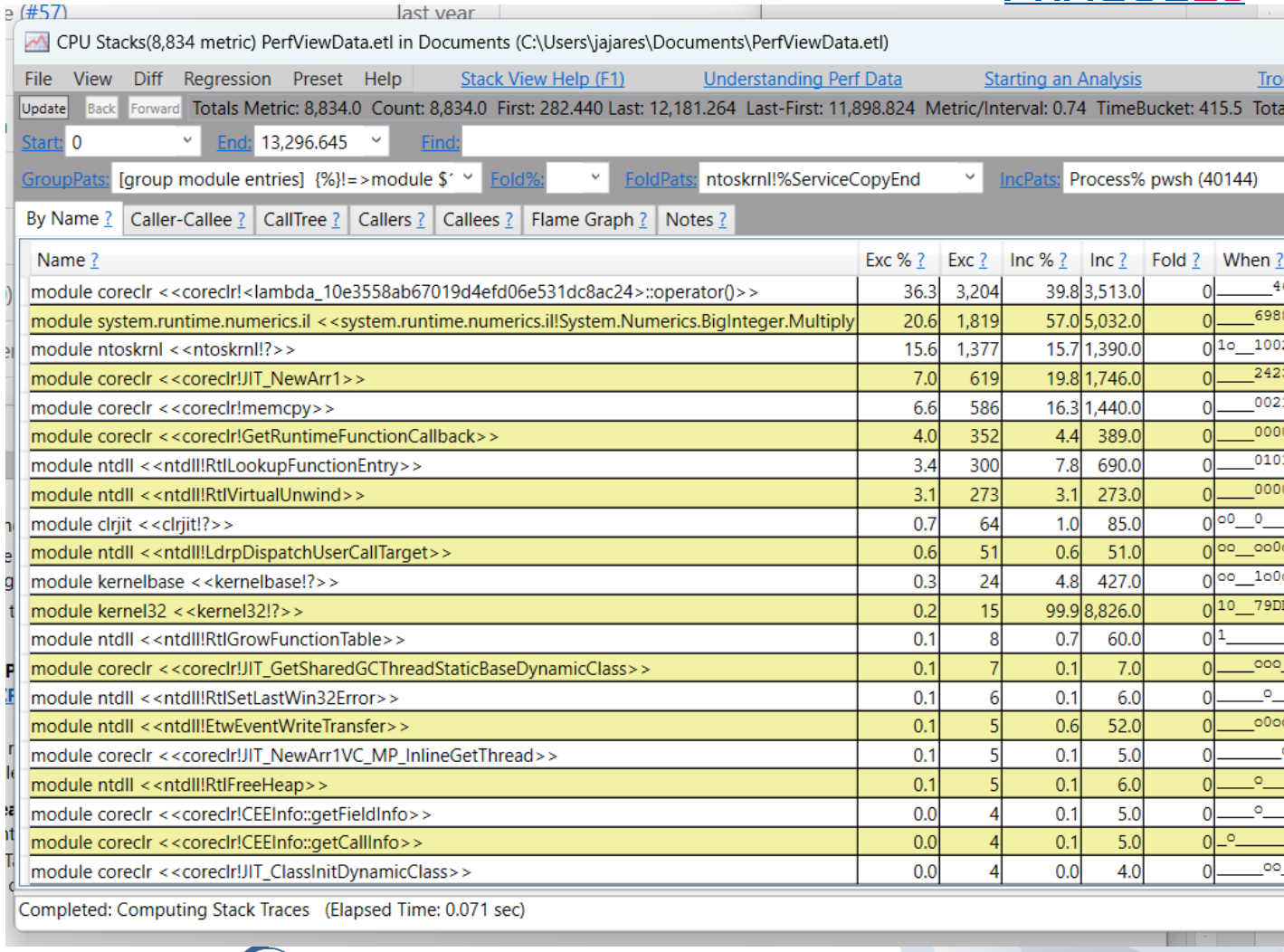
Installing PerfView

choco install PerfView –y

[microsoft/perfview: PerfView is a CPU and memory performance-analysis tool (github.com)]

PerfView





Name ?	Exc % ?	Exc ?	Inc % ?	Inc ?	Fold ?	When ?
module coreclr <<coreclr!<lambda_10e3558ab67019d4efd06e531dc8ac24>::operator()>>	36.3	3,204	39.8	3,513.0	0	
module system.runtime.numerics.il <<system.runtime.numerics.il!System.Numerics.BigInteger.Multiply	20.6	1,819	57.0	5,032.0	0	
module ntoskrnl <<ntoskrnl!?>>	15.6	1,377	15.7	1,390.0	0	
module coreclr <<coreclr!JIT_NewArr1>>	7.0	619	19.8	1,746.0	0	
module coreclr <<coreclr!memcpy>>	6.6	586	16.3	1,440.0	0	
module coreclr <<coreclr!GetRuntimeFunctionCallback>>	4.0	352	4.4	389.0	0	
module ntdll <<ntdll!RtlLookupFunctionEntry>>	3.4	300	7.8	690.0	0	
module ntdll <<ntdll!RtlVirtualUnwind>>	3.1	273	3.1	273.0	0	
module clrjit <<clrjit!?>>	0.7	64	1.0	85.0	0	
module ntdll <<ntdll!LdrpDispatchUserCallTarget>>	0.6	51	0.6	51.0	0	
module kernelbase <<kernelbase!?>>	0.3	24	4.8	427.0	0	
module kernel32 <<kernel32!?>>	0.2	15	99.9	8,826.0	0	
module ntdll <<ntdll!RtlGrowFunctionTable>>	0.1	8	0.7	60.0	0	
module coreclr <<coreclr!JIT_GetSharedGCThreadStaticBaseDynamicClass>>	0.1	7	0.1	7.0	0	
module ntdll <<ntdll!RtlSetLastWin32Error>>	0.1	6	0.1	6.0	0	
module ntdll <<ntdll!EtwEventWriteTransfer>>	0.1	5	0.6	52.0	0	
module coreclr <<coreclr!JIT_NewArr1VC_MP_InlineGetThread>>	0.1	5	0.1	5.0	0	
module ntdll <<ntdll!RtlFreeHeap>>	0.1	5	0.1	6.0	0	
module coreclr <<coreclr!CEEInfo::getFieldInfo>>	0.0	4	0.1	5.0	0	
module coreclr <<coreclr!CEEInfo::getCallInfo>>	0.0	4	0.1	5.0	0	
module coreclr <<coreclr!JIT_ClassInitDynamicClass>>	0.0	4	0.0	4.0	0	

PerfView

- is free
- is non-invasive
- is very portable

- Only works on windows
- Is little hard to use

Profiling basics

- Samples and events.
- Kernel CPU sampler provides samples every 1ms. This sample have callstack of every CPU.
- Profiler (and Pwsh) emit events when interesting stuff happens, via additional Providers.

- PerfView observes samples and events via ETW and writes them into etl file.

Capture trace

- Capture trace of Import-Module that is running under Trace-Script from latest Profiler.

- Including provider *Profiler.

  Look at events, from Profiler.

Focus the correct time

- Find Index and Return index of the event we are interested in.
- Find the text we are interested in.
- Limit PerfView view to just that time.

How can PerfView see the internals of my module?

- Thanks to PDB / symbols.
- A map from compiled code in a dll, to source files from which it was produced.

- Enables debugging, and stopping at breakpoints.

- Embedding pdbs and source link · PowerShell · Discussion #19774 (github.com)

Building PowerShell locally

```
$version = $PSVersionTable.GitCommitId
git checkout “v$version”
Import-Module build.psm1
Start-PSBootstrap
Start-PSBuild

Function Start-DevPwsh {
        C:\p\powershell\src\powershell-win-
        core\bin\Debug\net7.0\win7-x64\publish\pwsh.exe
}
```

PSCONF.EU
PRAGUE23

🤔 The curious case of slow Pester import

Raffle reminder!

https://rb.gy/kk77g

Import-Module Pester 5.4.1

- Import takes ~4 seconds.
- Reported by dbachecks / Rob.
- Profiler tells us Get-Help is causing this.

- Later Frode notices that only the shipped version has this problem.

Profiling Pester with Profiler

PSCONF.EU
PRAGUE23

- import-module /p/profiler/profiler/profiler.psd1
- $trace = trace-script { import-module pester -force }

- $trace.Top50SelfDuration | select -First 10 SelfPercent, SelfDuration, HitCount, Module, Function, Text | Format-Table

Getting help, in and out


```powershell
# .SYNOPSIS
# Gets an AD user.
function Get-User {

    param (
        # The name of the user to find.
        $Name
    )
}
```

WPA

Using WPA

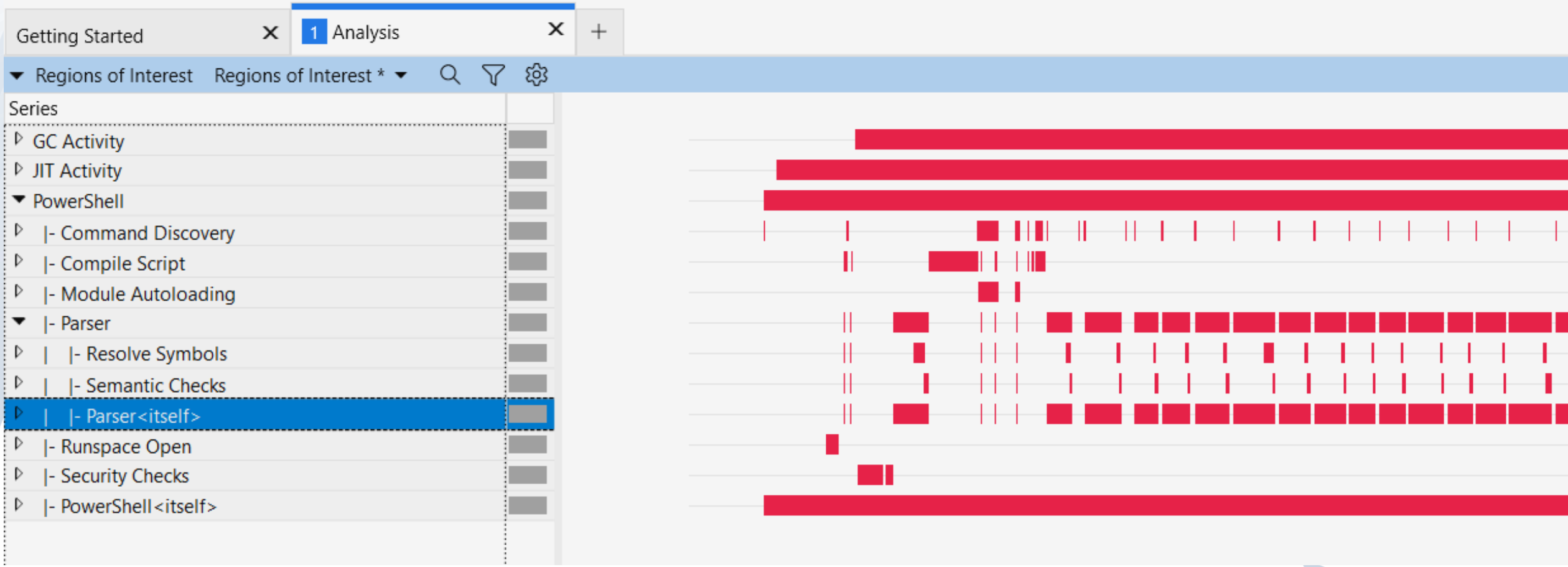
- PowerShell itself has this nice guide on performance
- PowerShell/tools/performance at master · PowerShell/PowerShell · GitHub

which is great at giving you visual view of what is happening, when your problem is actually in powershell "engine".

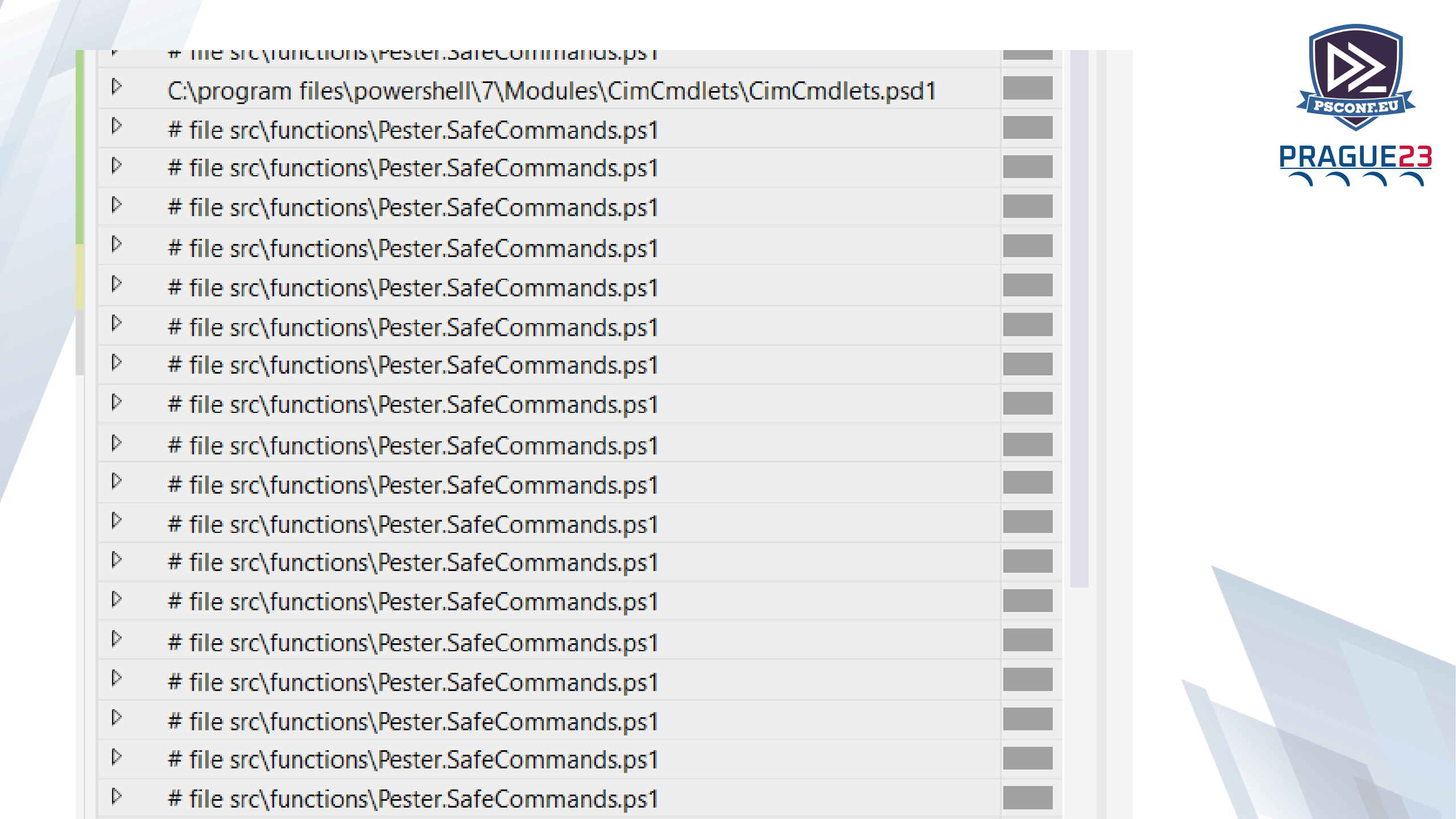
WPA

- windows performance analyzer. Nice tool, super powerful. You need a PHd to use it.

  Makes it easy to see that parser is busy.


Getting Started
1 Analysis
Regions of Interest
Regions of Interest *
Series
GC Activity
JIT Activity
PowerShell
|- Command Discovery
|- Compile Script
|- Module Autoloading
|- Parser
| |- Resolve Symbols
| |- Semantic Checks
| |- Parser<itself>
|- Runspace Open
|- Security Checks
|- PowerShell<itself>

@nohwnd


C:\program files\powershell\7\Modules\CimCmdlets\CimCmdlets.psd1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1
file src\functions\Pester.SafeCommands.ps1

WPA

- WPA is super useful for a developer of PowerShell, the setup they have is really nice.

  Our problem is a great fit for what is being observed so it is easy to see the problem.

  What I can't see easily though is why Get-Help is marked as taking so much time.

Summary

Summary

- PerfView is a free powerful tool.
- Try it, it is not scary.

Q&A

15 minutes

PSCONF.EU
PRAGUE23

@nohwnd