PowerShell Conference Europe

10 things you should know about Profiler

Jakub Jareš

@nohwnd

Antwerp24

PSCONF.EU

Antwerp24

Many thanks to our sponsors:

Chocolatey

PATCH MY PC

semperis

ScriptRunner®
Your #1 for PowerShell Automation

@nohwnd

Jakub Jareš

Profiler and Pester owner and maintainer.

Senior software engineer, developing VSTest, Testing Platform and MSTest at Microsoft. All opinions are mine.

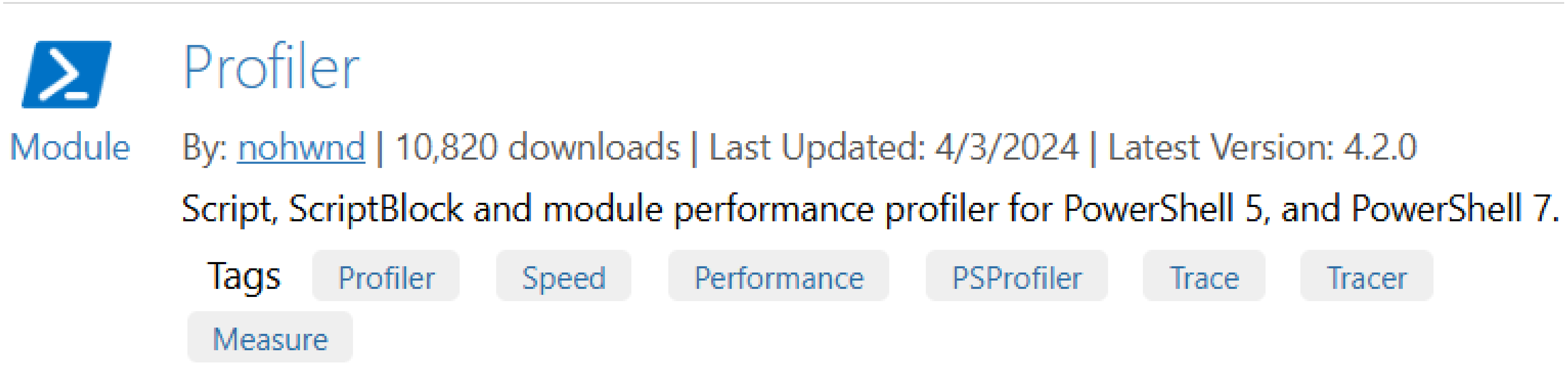
Please consider sponsoring my open-source development:

[Sponsor @nohwnd on GitHub Sponsors]

@nohwnd
@pspester
me@jakubjares.com

X @nohwnd

1. Profiler is free



Module

Profiler

By: nohwnd | 10,820 downloads | Last Updated: 4/3/2024 | Latest Version: 4.2.0

Script, ScriptBlock and module performance profiler for PowerShell 5, and PowerShell 7.

Tags Profiler Speed Performance PSProfiler Trace Tracer Measure

```
Install-Module Profiler
```

@nohwnd

2. Trace-Script

```
$trace = Trace-Script {
    Write-Host 👋 PSConfEU
    Start-Sleep -Milliseconds 2024
}
```

PSCONF.EU
Antwerp24

DEMO ITEM 02

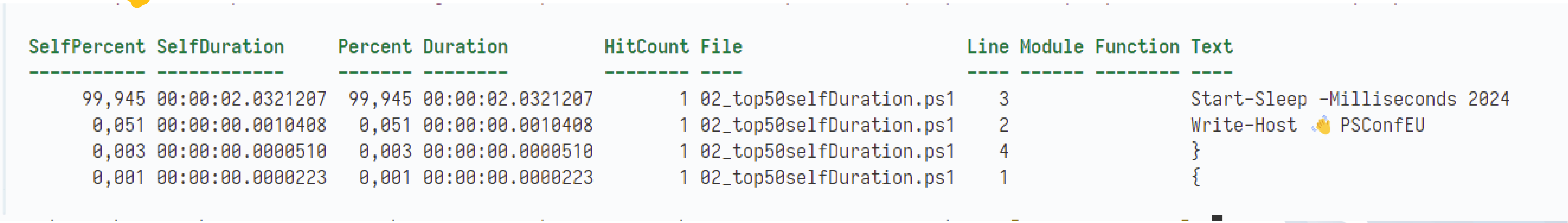
@nohwnd

3. $trace.Top50SelfDuration

Time spent
on this line
of code

Time spent
on this line
of code, and all code
it called

The slow code and
where it came from



```
SelfPercent SelfDuration     Percent Duration         HitCount File                    Line Module Function Text
----------- ------------     ------- --------         -------- ----                    ---- ------ -------- ----
     99,945 00:00:02.0321207  99,945 00:00:02.0321207        1 02_top50selfDuration.ps1   3                 Start-Sleep -Milliseconds 2024
      0,051 00:00:00.0010408   0,051 00:00:00.0010408        1 02_top50selfDuration.ps1   2                 Write-Host 👋 PSConfEU
      0,003 00:00:00.0000510   0,003 00:00:00.0000510        1 02_top50selfDuration.ps1   4                 }
      0,001 00:00:00.0000223   0,001 00:00:00.0000223        1 02_top50selfDuration.ps1   1                 {
```

PSCONF.EU
Antwerp24

DEMO ITEM 03

@nohwnd

4. $trace.Top50SelfMemory

Memory allocated by this line

Memory allocated by this line and all code it called

The slow code and where it came from

```
SelfMemoryPercent SelfMemory SelfGc MemoryPercent     Memory  Gc Duration         HitCount File                   Line Module Function Text
----------------- ---------- ------ -------------     ------  -- --------         -------- ----                   ---- ------ -------- ----
         99,34300 1554,70824    220      99,34300 1554,70824 220 00:00:02.1864900    10000 04_top50SelfMemory.ps1    6                 $newNumbers += $number
          0,63700    9,97462      0       0,63700    9,97462   0 00:00:00.0425057    10002 04_top50SelfMemory.ps1    4                 foreach ($number in $numbers) {
          0,02000    0,31092      0       0,02000    0,31092   0 00:00:00.0001179        1 04_top50SelfMemory.ps1    2                 $numbers = 1..10000
                0    0,00000      0             0    0,00000   0 00:00:00.0000279        1 04_top50SelfMemory.ps1    1                 {
                0    0,00000      0             0    0,00000   0 00:00:00.0000102        1 04_top50SelfMemory.ps1    3                 $newNumbers = @()
                0    0,00000      0             0    0,00000   0 00:00:00.0000335        1 04_top50SelfMemory.ps1    8                 }
```

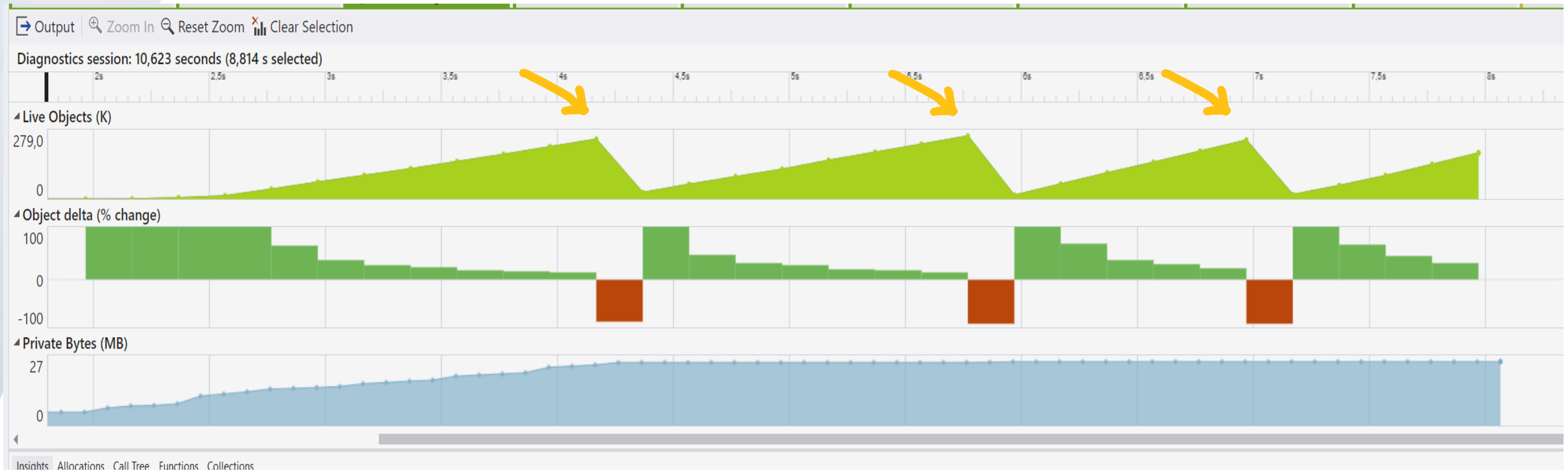
@nohwnd

PSCONF.EU
Antwerp24

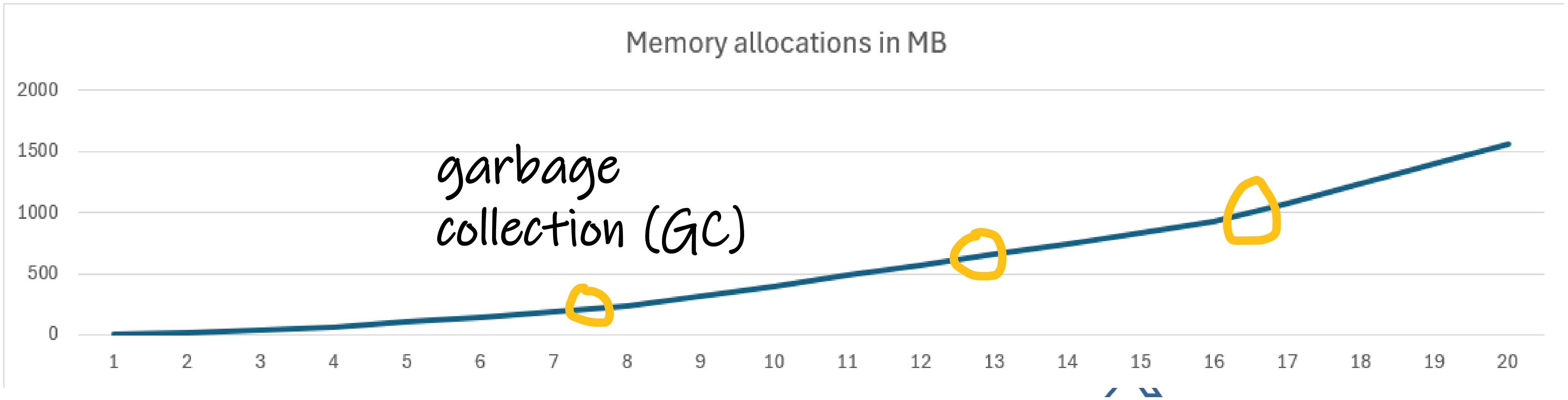
DEMO ITEM 04

@nohwnd

How we measure memory

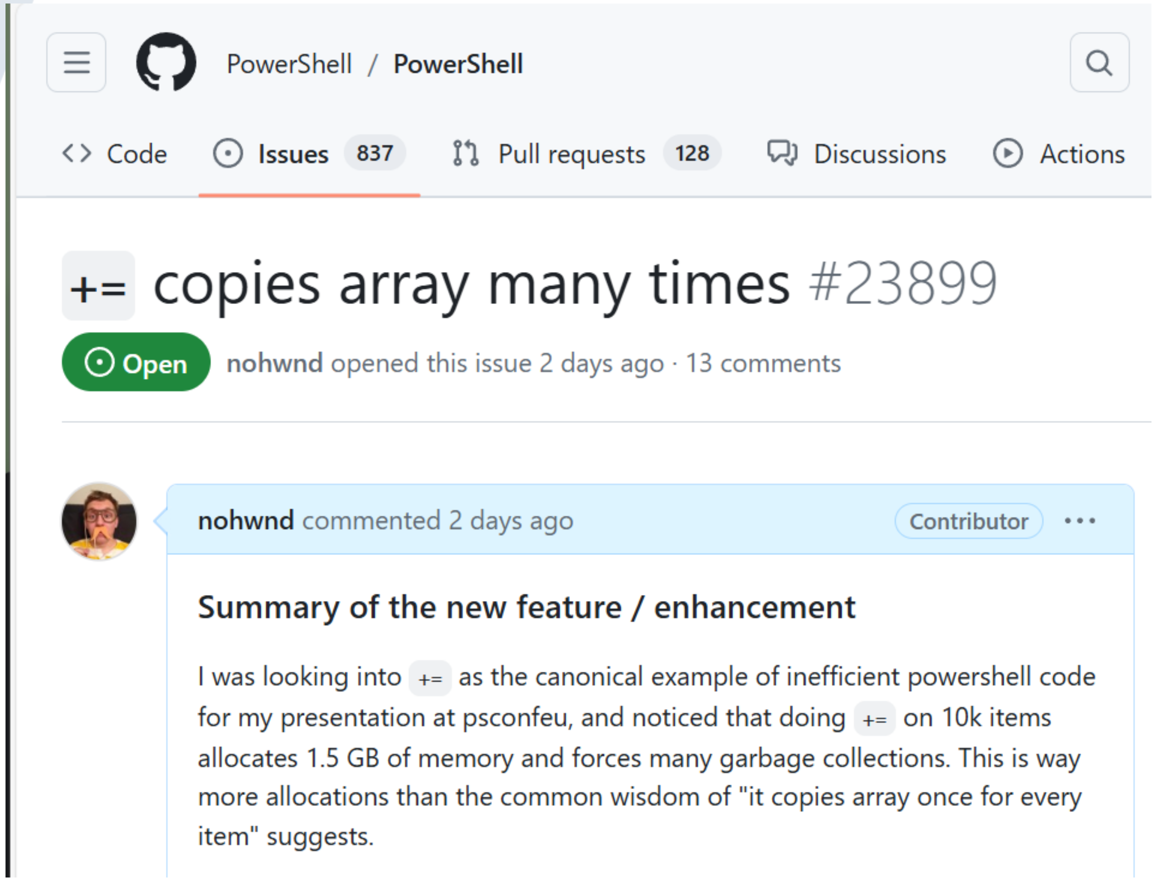


How .NET thinks about memory



How Profiler thinks about memory

5. Profiler can help you make PowerShell better for everyone



+= copies array many times #23899

Open nohwnd opened this issue 2 days ago · 13 comments

nohwnd commented 2 days ago Contributor

Summary of the new feature / enhancement

I was looking into `+=` as the canonical example of inefficient powershell code for my presentation at psconfeu, and noticed that doing `+=` on 10k items allocates 1.5 GB of memory and forces many garbage collections. This is way more allocations than the common wisdom of "it copies array once for every item" suggests.

Thanks @BoreanJordan
and @SeeminglyScienc
for helping with the
investigation
and looking
into fixing this!

@nohwnd

PSCONF.EU Antwerp24

Why is += so bad?

```
$numbers = 1..10000
$newNumbers = @()
foreach ($number in $numbers) {
    $newNumbers += $number
}
```

it copies the array
for every number

Why is += so bad?

for every item 😳

```
var result = new List<object>();

foreach (var num in 1..9999)
    result.Add(Current(num));

result.Add(10000);

return result.ToArray();
```

has capacity of 4 items

grow and copy 8, 16 ... 8192, 16384, 12 times

copy one last time for good measure

PSCONF.EU Antwerp24

@nohwnd

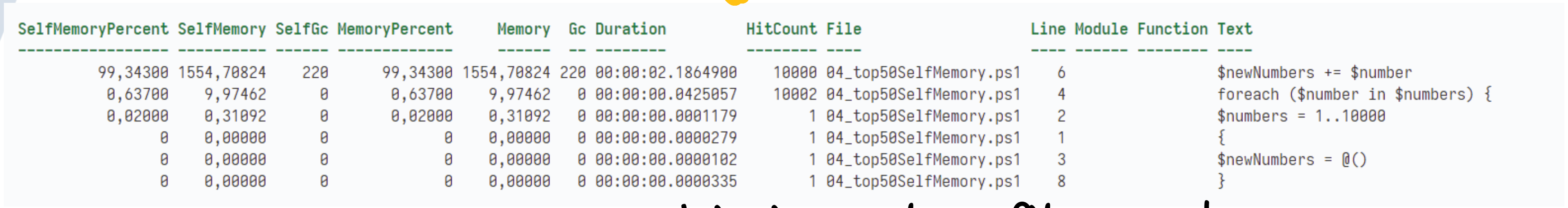
💡 Optimization

Assign results of foreach directly to a variable.

```
$numbers = 1..10000
$newNumbers = foreach ($number in $numbers) {
        $number
}
```

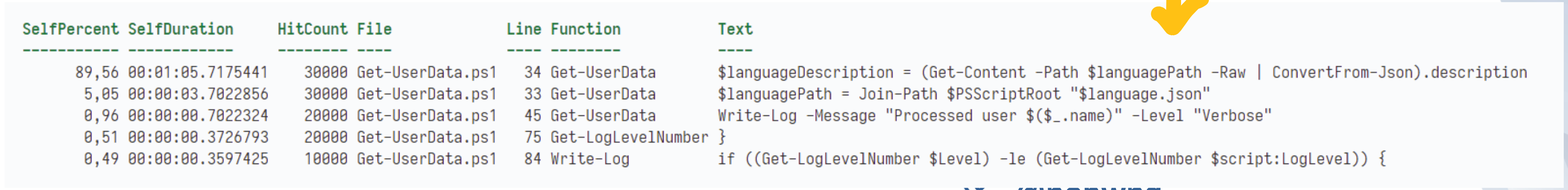
6. HitCount column

we process 10 000 items,
we want to see 10 000 or less here



```
SelfMemoryPercent SelfMemory SelfGc MemoryPercent    Memory Gc Duration         HitCount File                Line Module Function Text
----------------- ---------- ------ -------------    ------ -- --------         -------- ----                ---- ------ -------- ----
         99,34300 1554,70824    220      99,34300 1554,70824 220 00:00:02.1864900    10000 04_top50SelfMemory.ps1   6                 $newNumbers += $number
          0,63700    9,97462      0       0,63700    9,97462   0 00:00:00.0425057    10002 04_top50SelfMemory.ps1   4                 foreach ($number in $numbers) {
          0,02000    0,31092      0       0,02000    0,31092   0 00:00:00.0001179        1 04_top50SelfMemory.ps1   2                 $numbers = 1..10000
                0    0,00000      0             0    0,00000   0 00:00:00.0000279        1 04_top50SelfMemory.ps1   1                 {
                0    0,00000      0             0    0,00000   0 00:00:00.0000102        1 04_top50SelfMemory.ps1   3                 $newNumbers = @()
                0    0,00000      0             0    0,00000   0 00:00:00.0000335        1 04_top50SelfMemory.ps1   8                 }
```

this is a slow file read,
look at hit count



```
SelfPercent SelfDuration     HitCount File             Line Function          Text
----------- ------------     -------- ----             ---- --------          ----
      89,56 00:01:05.7175441    30000 Get-UserData.ps1   34 Get-UserData      $languageDescription = (Get-Content -Path $languagePath -Raw | ConvertFrom-Json).description
       5,05 00:00:03.7022856    30000 Get-UserData.ps1   33 Get-UserData      $languagePath = Join-Path $PSScriptRoot "$language.json"
       0,96 00:00:00.7022324    20000 Get-UserData.ps1   45 Get-UserData      Write-Log -Message "Processed user $($_.name)" -Level "Verbose"
       0,51 00:00:00.3726793    20000 Get-UserData.ps1   75 Get-LogLevelNumber }
       0,49 00:00:00.3597425    10000 Get-UserData.ps1   84 Write-Log         if ((Get-LogLevelNumber $Level) -le (Get-LogLevelNumber $script:LogLevel)) {
```

💡 Optimization

- Move file reads out of loops.
- Get-Content **-Raw**
- Cache data

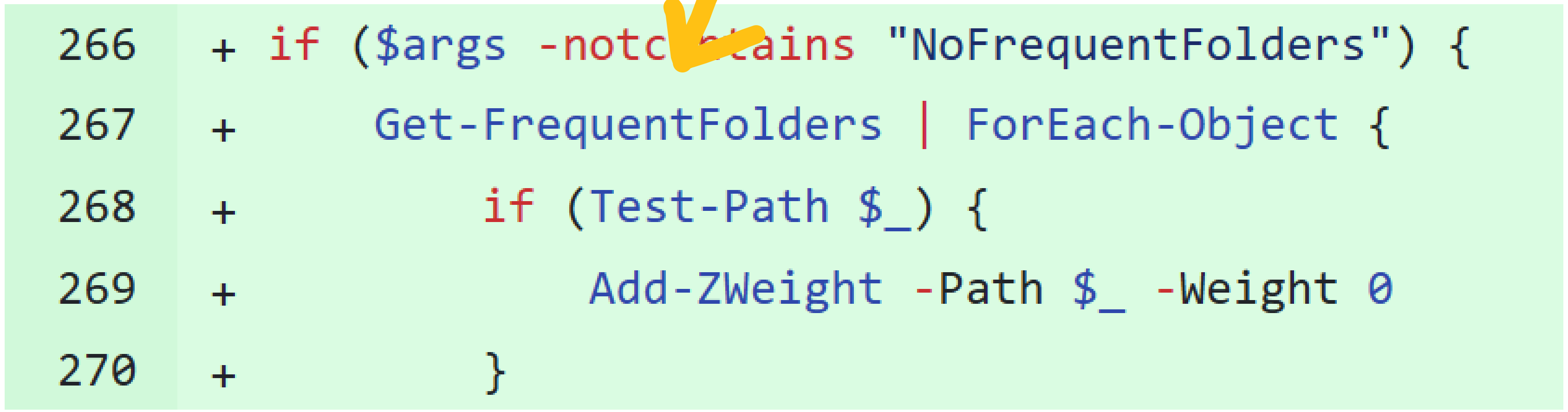
My PSConfEU 2022 talk has good examples:
[Make your scripts faster with Profiler - Jareš Jakub - PSConfEU 2022 - YouTube]

Getting data to analyze

7. Profile your $profile

```
pwsh -NoProfile -NoExit {
  $trace = Trace-Script { . $profile }
}
```

500ms 😰



```
+ if ($args -notcontains "NoFrequentFolders") {
+     Get-FrequentFolders | ForEach-Object {
+         if (Test-Path $_) {
+             Add-ZWeight -Path $_ -Weight 0
+         }
```

PSCONF.EU
Antwerp24

DEMO ITEM 07

@nohwnd

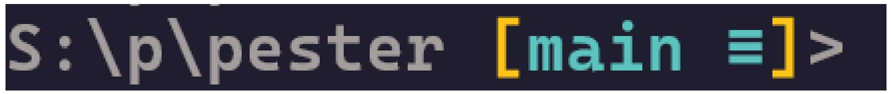
💡 Optimization

Skip the code you don't need.

8. Profile your prompt

```
$trace = Trace-Script { prompt }
```

120ms 👍



```
S:\p\pester [main ≡]>
```

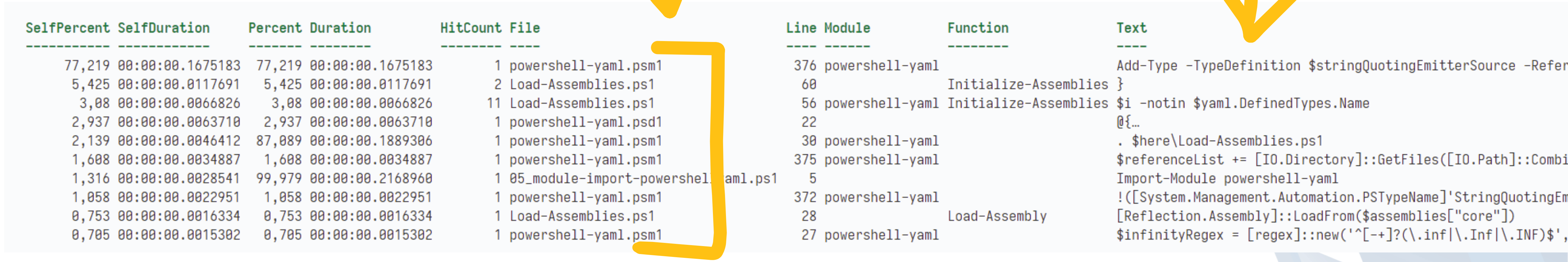
@nohwnd

9. Profile your module import

powershell-yaml: 0.4.7, 40 million downloads

pre-compile this

merge into 1 file



```
SelfPercent SelfDuration     Percent Duration         HitCount File                                   Line Module          Function             Text
----------- ------------     ------- --------         -------- ----                                   ---- ------          --------             ----
     77,219 00:00:00.1675183  77,219 00:00:00.1675183        1 powershell-yaml.psm1                    376 powershell-yaml                      Add-Type -TypeDefinition $stringQuotingEmitterSource -Refer
      5,425 00:00:00.0117691   5,425 00:00:00.0117691        2 Load-Assemblies.ps1                      60                 Initialize-Assemblies }
       3,08 00:00:00.0066826    3,08 00:00:00.0066826       11 Load-Assemblies.ps1                      56 powershell-yaml Initialize-Assemblies $i -notin $yaml.DefinedTypes.Name
      2,937 00:00:00.0063710   2,937 00:00:00.0063710        1 powershell-yaml.psd1                     22                                      @{…
      2,139 00:00:00.0046412  87,089 00:00:00.1889306        1 powershell-yaml.psm1                     30 powershell-yaml                      . $here\Load-Assemblies.ps1
      1,608 00:00:00.0034887   1,608 00:00:00.0034887        1 powershell-yaml.psm1                    375 powershell-yaml                      $referenceList += [IO.Directory]::GetFiles([IO.Path]::Combi
      1,316 00:00:00.0028541  99,979 00:00:00.2168960        1 05_module-import-powershel[illegible]aml.ps1   5                                      Import-Module powershell-yaml
      1,058 00:00:00.0022951   1,058 00:00:00.0022951        1 powershell-yaml.psm1                    372 powershell-yaml                      !([System.Management.Automation.PSTypeName]'StringQuotingEm
      0,753 00:00:00.0016334   0,753 00:00:00.0016334        1 Load-Assemblies.ps1                      28                 Load-Assembly        [Reflection.Assembly]::LoadFrom($assemblies["core"])
      0,705 00:00:00.0015302   0,705 00:00:00.0015302        1 powershell-yaml.psm1                     27 powershell-yaml                      $infinityRegex = [regex]::new('^[-+]?(\.inf|\.Inf|\.INF)$',
```

@nohwnd

PSCONF.EU
Antwerp24

DEMO ITEM 09

@nohwnd

💡 Optimization

Pre-compile your code.

Merge your module files to 1.

10. Profile anything else

- Your Pester tests.
- Your most used function from your favorite module.
- Any code that imports or exports data that you are using at work.
- Any code you can think of.

Summary

- Use profiler to find slow code, become the goto perf person in your team.
- Use the columns and views that are not SelfDuration to hint at the problem.
- Report the issues you find.

Go see PowerShell Performance - YouTube for other talks on this topic.

@nohwnd

Q&A

15 minutes

@nohwnd