PSCONF.EU

PowerShell Conference Europe

Make your scripts faster with Profiler

Jakub Jareš
@nohwnd

VIENNA

20 - 23 June 2022

Many thanks to our sponsors:

SquaredUp

au2mator

PATCH MY PC

Chocolatey

System Frontier

ScriptRunner®

@nohwnd

Raffle

https://forms.office.com/r/Yyi2zfHZtA

Jakub Jareš

@nohwnd
@pspester
me@jakubjares.com

Pester owner and maintainer. Senior software engineer, developing VSTest and MSTest at Microsoft. All opinions are mine.

Please consider sponsoring my open-source development:

[Sponsor @nohwnd on GitHub Sponsors]()

@nohwnd

What is Profiler?

A useful tool to find slow PowerShell code,

no matter where it is.

- PowerShell module.
- Successor of PSProfiler.

https://github.com/nohwnd/Profiler

Profile a script file, with or without parameters

```
$trace = Trace-Script {
    & .\myScript.ps1 -Name 'Jakub' -Age 32
}
```

Profile a module function, including module load

```
$trace = Trace-Script {
  Import-Module .\Planets.psm1 -Force
  Get-Planet -Name 'M*'
}
```

Profile a module function, excluding module load

```
Import-Module .\Planets.psm1 -Force

$trace = Trace-Script {
  Get-Planet -Name 'M*'
}
```

Profile your $profile script

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

Profile any scriptblock, or any piece of code

```
$trace = Trace-Script {
  "hello"
}
```

Demo - 1

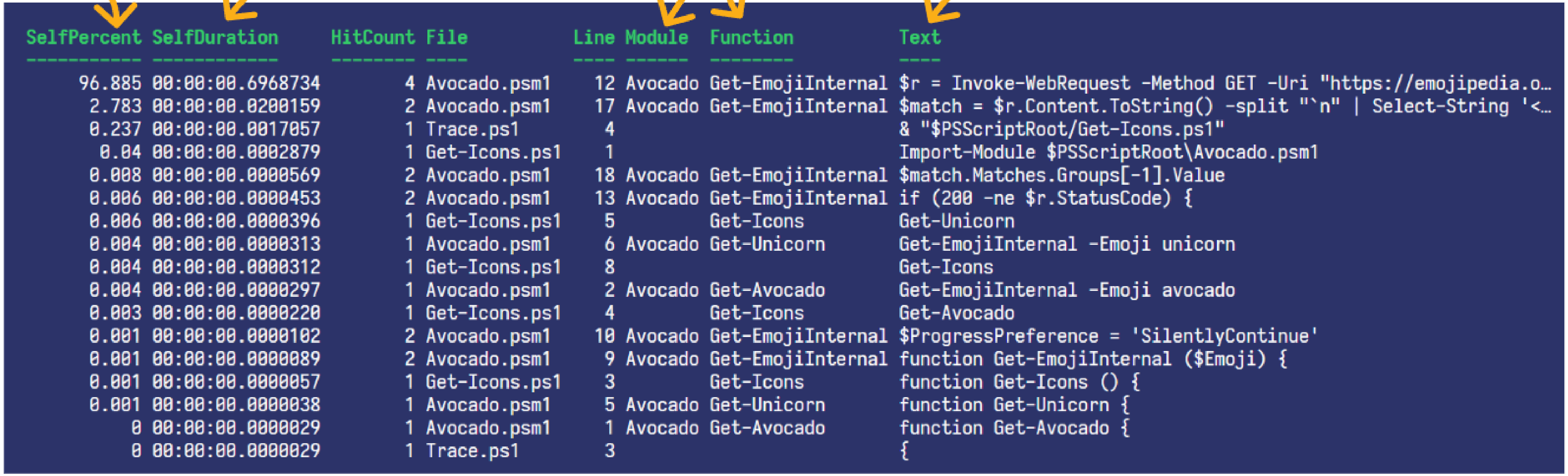
Why is it slow?

Understanding Profiler output

How much time it took NOT including any code it called

Where is the code coming from

The slow code

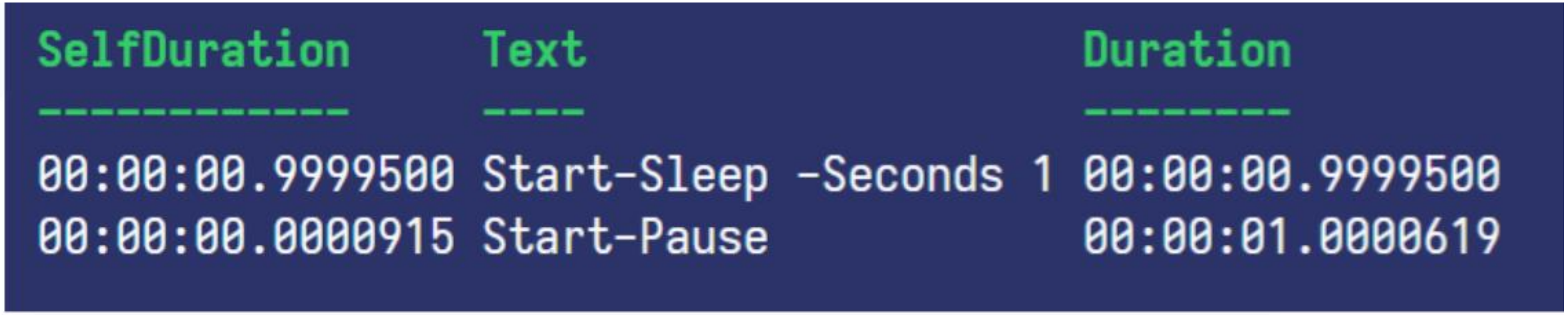


```
SelfPercent SelfDuration     HitCount File          Line Module  Function          Text
----------- ------------     -------- ----          ---- ------  --------          ----
     96.885 00:00:00.6968734        4 Avocado.psm1    12 Avocado Get-EmojiInternal $r = Invoke-WebRequest -Method GET -Uri "https://emojipedia.o…
      2.783 00:00:00.0200159        2 Avocado.psm1    17 Avocado Get-EmojiInternal $match = $r.Content.ToString() -split "`n" | Select-String '<…
      0.237 00:00:00.0017057        1 Trace.ps1        4                           & "$PSScriptRoot/Get-Icons.ps1"
       0.04 00:00:00.0002879        1 Get-Icons.ps1    1                           Import-Module $PSScriptRoot\Avocado.psm1
      0.008 00:00:00.0000569        2 Avocado.psm1    18 Avocado Get-EmojiInternal $match.Matches.Groups[-1].Value
      0.006 00:00:00.0000453        2 Avocado.psm1    13 Avocado Get-EmojiInternal if (200 -ne $r.StatusCode) {
      0.006 00:00:00.0000396        1 Get-Icons.ps1    5         Get-Icons         Get-Unicorn
      0.004 00:00:00.0000313        1 Avocado.psm1     6 Avocado Get-Unicorn       Get-EmojiInternal -Emoji unicorn
      0.004 00:00:00.0000312        1 Get-Icons.ps1    8                           Get-Icons
      0.004 00:00:00.0000297        1 Avocado.psm1     2 Avocado Get-Avocado       Get-EmojiInternal -Emoji avocado
      0.003 00:00:00.0000220        1 Get-Icons.ps1    4         Get-Icons         Get-Avocado
      0.001 00:00:00.0000102        2 Avocado.psm1    10 Avocado Get-EmojiInternal $ProgressPreference = 'SilentlyContinue'
      0.001 00:00:00.0000089        2 Avocado.psm1     9 Avocado Get-EmojiInternal function Get-EmojiInternal ($Emoji) {
      0.001 00:00:00.0000057        1 Get-Icons.ps1    3         Get-Icons         function Get-Icons () {
      0.001 00:00:00.0000038        1 Avocado.psm1     5 Avocado Get-Unicorn       function Get-Unicorn {
          0 00:00:00.0000029        1 Avocado.psm1     1 Avocado Get-Avocado       function Get-Avocado {
          0 00:00:00.0000029        1 Trace.ps1        3                           {
```

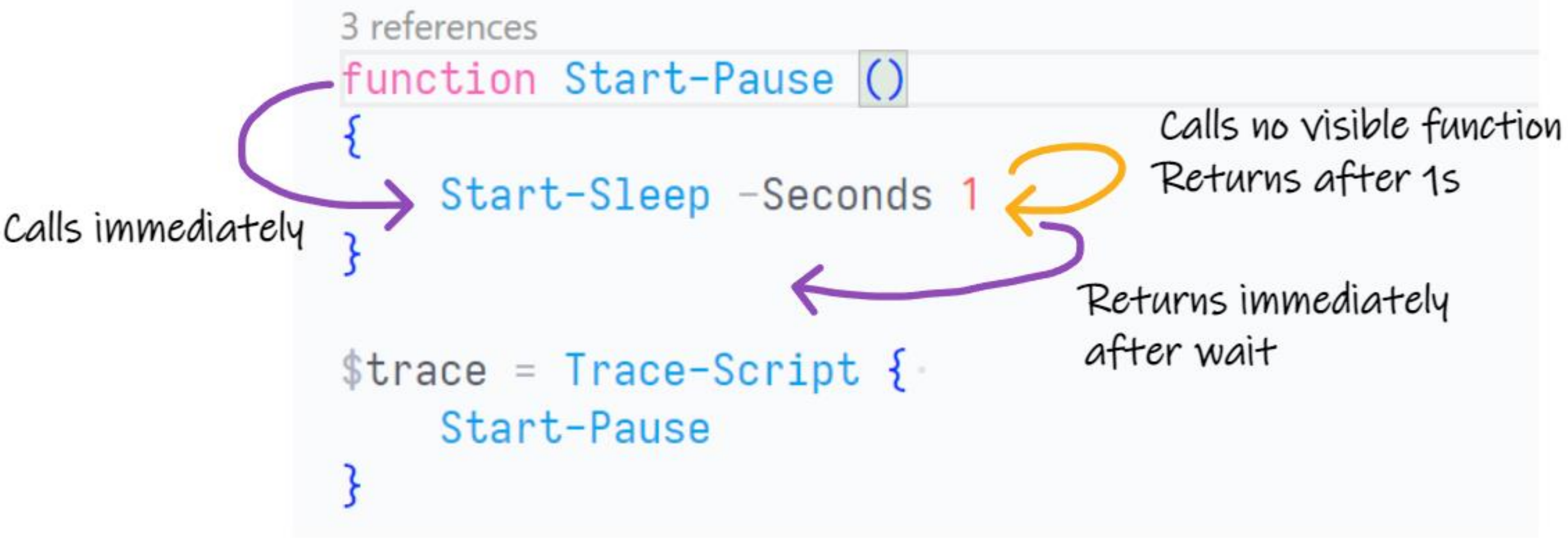
Columns

- SelfPercent, SelfDuration - How much time was spent in this code, excluding any called code.
- Percent, Duration - How much time was spent in this code and all the code it called.
- HitCount - How many times this command was called.

- File, Line, Module, Function, Text - Where the code is.



```
SelfDuration     Text                      Duration
------------     ----                      --------
00:00:00.9999500 Start-Sleep -Seconds 1 00:00:00.9999500
00:00:00.0000915 Start-Pause            00:00:01.0000619
```



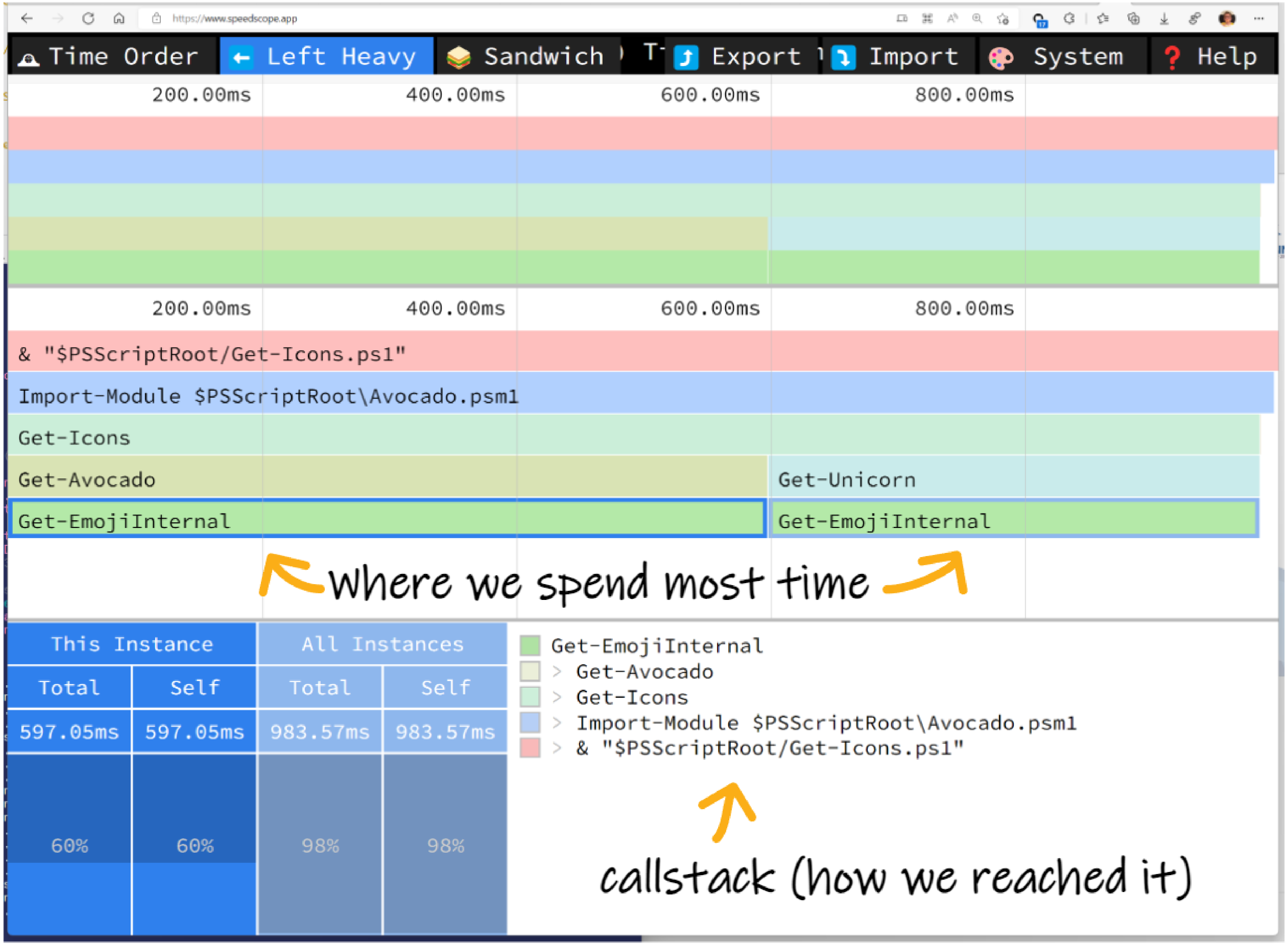
```
3 references
function Start-Pause ()
{
    Start-Sleep -Seconds 1
}

$trace = Trace-Script {
    Start-Pause
}
```

Calls immediately

Calls no visible function
Returns after 1s

Returns immediately
after wait



```
& "$PSScriptRoot/Get-Icons.ps1"
Import-Module $PSScriptRoot\Avocado.psm1
Get-Icons
Get-Avocado                                   Get-Unicorn
Get-EmojiInternal                             Get-EmojiInternal
```

Where we spend most time

This Instance		All Instances	
Total	Self	Total	Self
597.05ms	597.05ms	983.57ms	983.57ms
60%	60%	98%	98%

```
🟩 Get-EmojiInternal
   > Get-Avocado
   > Get-Icons
   > Import-Module $PSScriptRoot\Avocado.psm1
   > & "$PSScriptRoot/Get-Icons.ps1"
```

callstack (how we reached it)

PSCONF.EU
VIENNA
20 - 23 June 2022

Raffle

https://forms.office.com/r/Yyi2zfHZtA

Find slow code

Three steps to faster code

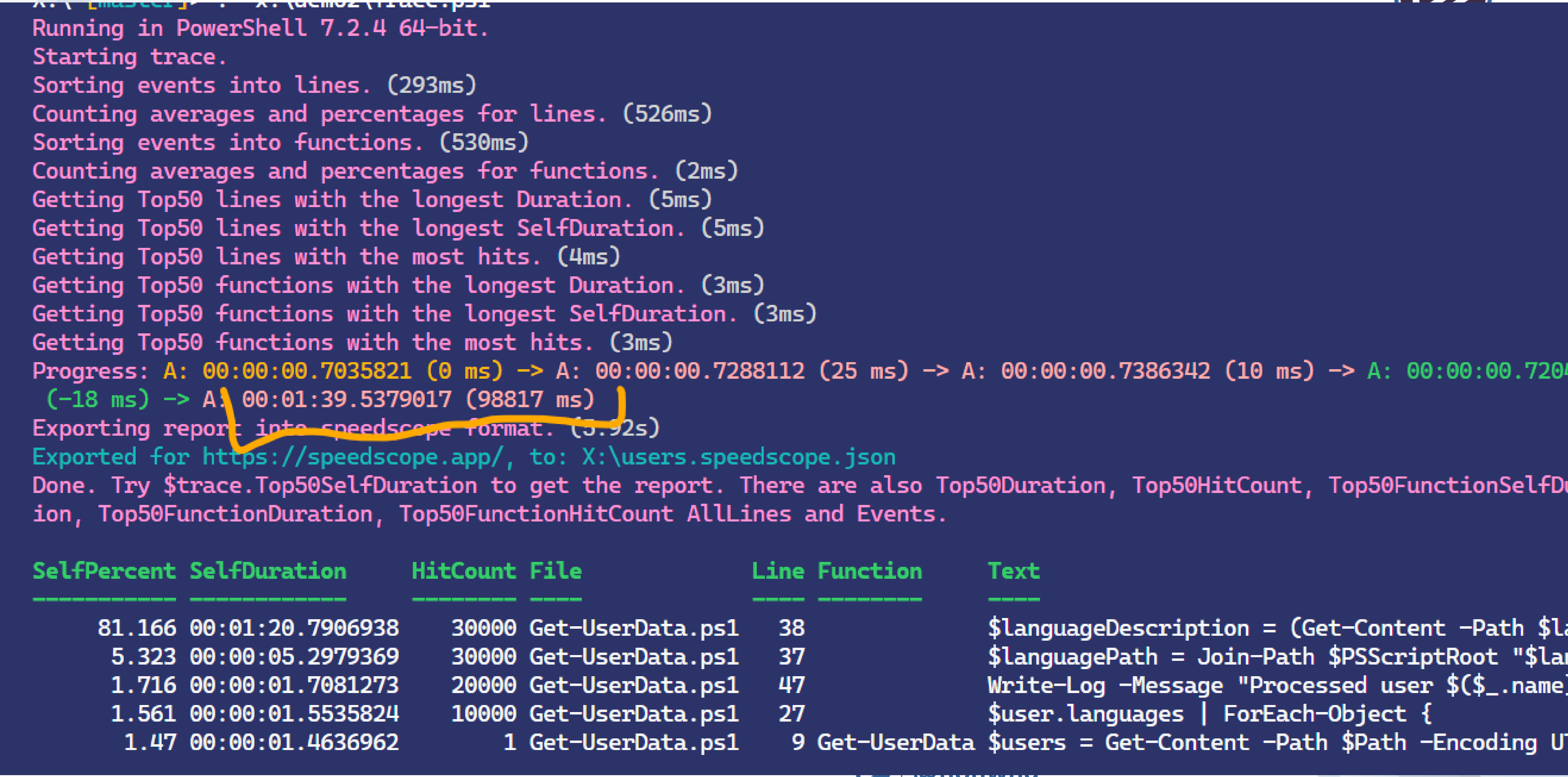
- Find slow code.
- Optimize it.
- Make sure it is faster.

See talks on youtube to learn more.

https://cutt.ly/hJn8u2o

Demo - 2

Optimizing code



```
X:\ [master]> . X:\demo2\trace.ps1
Running in PowerShell 7.2.4 64-bit.
Starting trace.
Sorting events into lines. (293ms)
Counting averages and percentages for lines. (526ms)
Sorting events into functions. (530ms)
Counting averages and percentages for functions. (2ms)
Getting Top50 lines with the longest Duration. (5ms)
Getting Top50 lines with the longest SelfDuration. (5ms)
Getting Top50 lines with the most hits. (4ms)
Getting Top50 functions with the longest Duration. (3ms)
Getting Top50 functions with the longest SelfDuration. (3ms)
Getting Top50 functions with the most hits. (3ms)
Progress: A: 00:00:00.7035821 (0 ms) -> A: 00:00:00.7288112 (25 ms) -> A: 00:00:00.7386342 (10 ms) -> A: 00:00:00.720
 (-18 ms) -> A: 00:01:39.5379017 (98817 ms)
Exporting report into speedscope format. (5.92s)
Exported for https://speedscope.app/, to: X:\users.speedscope.json
Done. Try $trace.Top50SelfDuration to get the report. There are also Top50Duration, Top50HitCount, Top50FunctionSelfD
ion, Top50FunctionDuration, Top50FunctionHitCount AllLines and Events.

SelfPercent SelfDuration     HitCount File             Line Function     Text
----------- ------------     -------- ----             ---- --------     ----
     81.166 00:01:20.7906938    30000 Get-UserData.ps1   38              $languageDescription = (Get-Content -Path $la
      5.323 00:00:05.2979369    30000 Get-UserData.ps1   37              $languagePath = Join-Path $PSScriptRoot "$lan
      1.716 00:00:01.7081273    20000 Get-UserData.ps1   47              Write-Log -Message "Processed user $($_.name
      1.561 00:00:01.5535824    10000 Get-UserData.ps1   27              $user.languages | ForEach-Object {
       1.47 00:00:01.4636962        1 Get-UserData.ps1    9 Get-UserData $users = Get-Content -Path $Path -Encoding U
```

Was this real life?

- Yeah! I did totally the same optimizations in AutoRest to cut runtime in half:
- Improve perf by nohwnd · Pull Request #8 · FriedrichWeinmann/AutoRest (github.com)

Flame graph - left heavy mode

- It is 100% on width
- What up direction means.
- What jaggedness means.

Summary

Summary

- Use Profiler to find slow code.
- Apply just the right amount of optimizations, where needed.
- You don't need to sacrifice your code style to performance.

Pick a winner!

Q&A

15 minutes

Ask me **here** about Profiler.

Ask me **outside** about:

Pester, testing, C#, .NET, dotnet, dotnet test, MSTest, Profiler, PerfView, coffee, beer etc.

@nohwnd