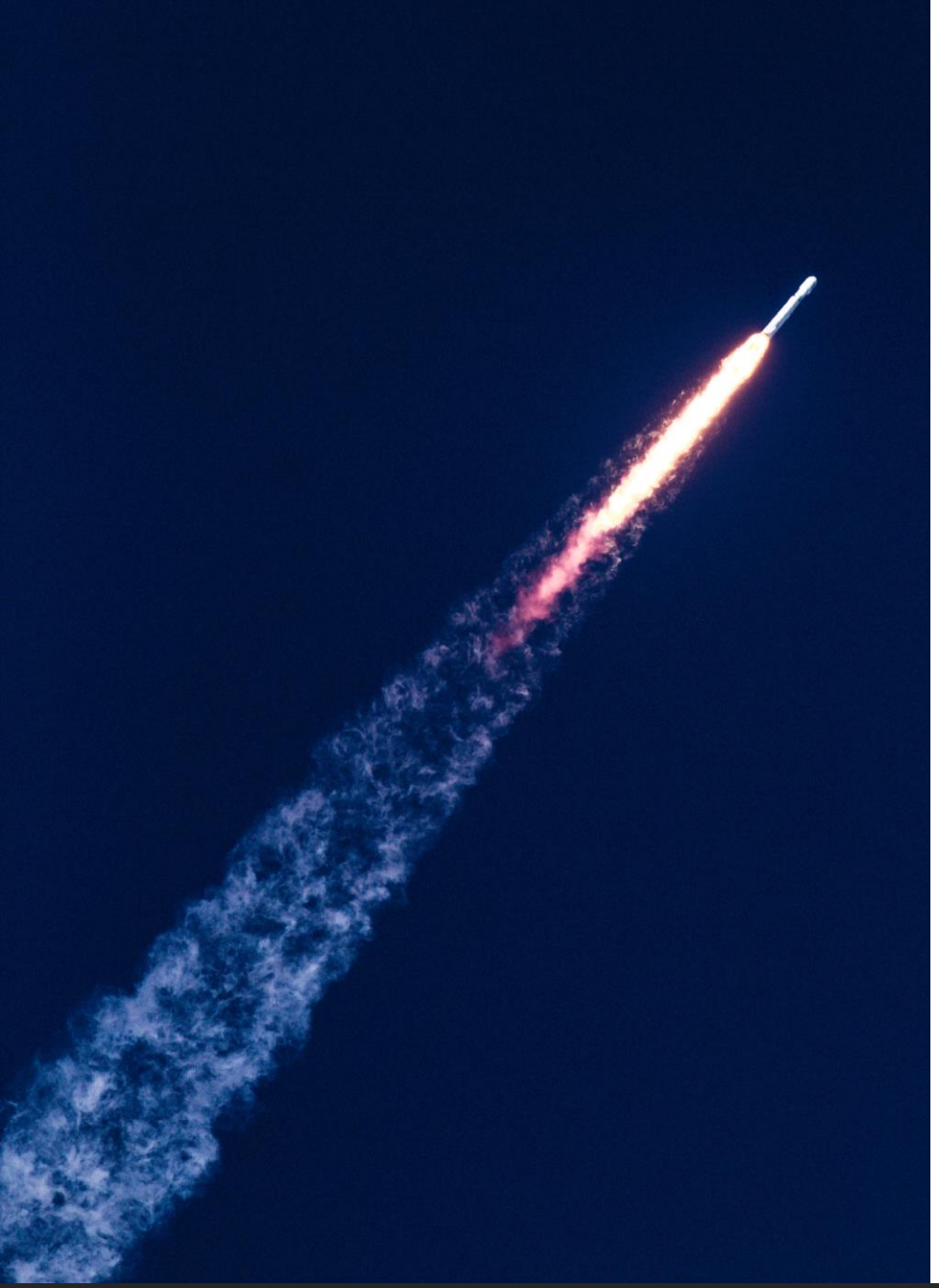


Profiler

MAKE YOUR SCRIPTS FASTER

@NOHWND

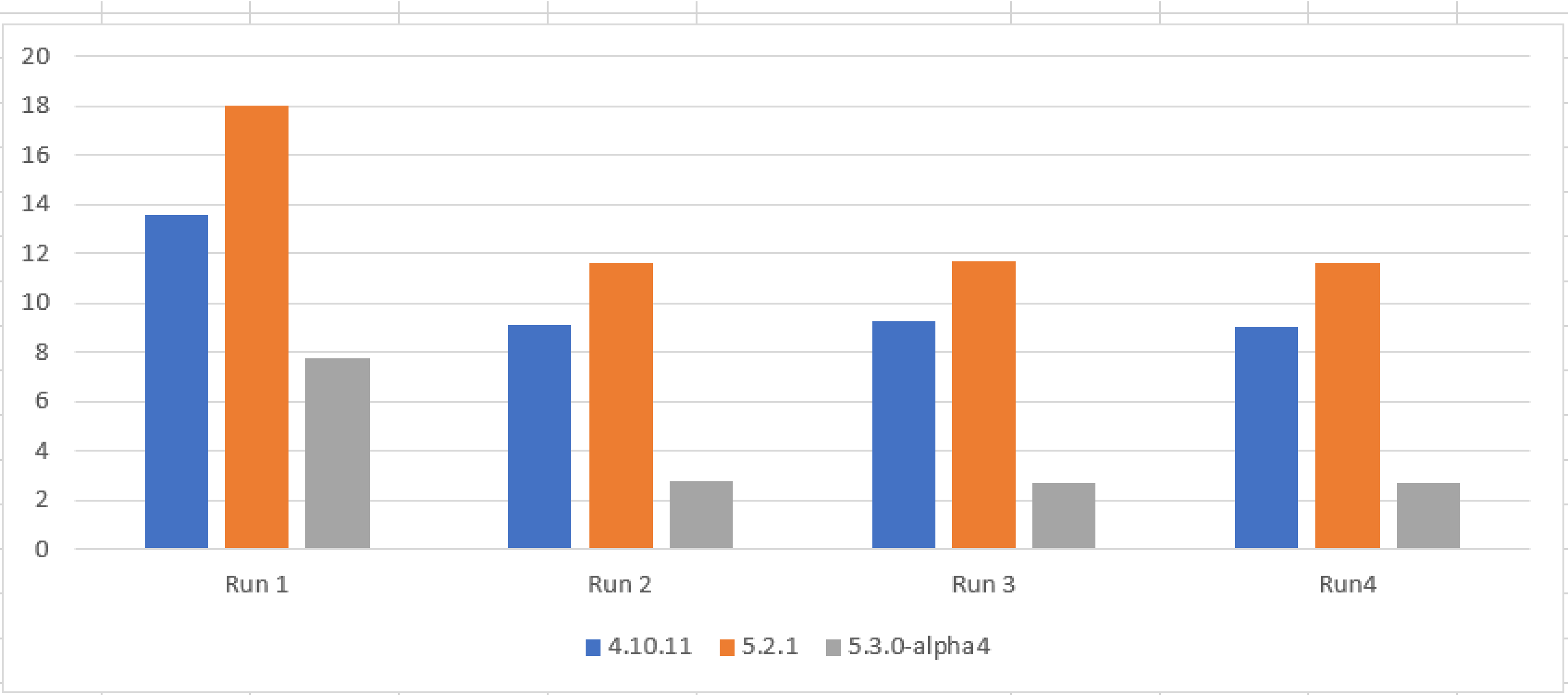
```
PS> $trace = Trace-Script -ScriptBlock { /p/Pester/test.ps1 -SkipPTests }
Running in PowerShell 7.1.3 64-bit.
Starting trace.
Stopwatch is high resolution, max resolution of timestamps is 100ns.

[...]

Tracing done. Got 2796300 trace events.
Processing 2796300 trace events. This might take a while...
Figuring out flow. (1.02s)
Sorting events into lines. (683ms)
Counting averages and percentages. (724ms)
Getting Top50 with the longest Duration. (172ms)
Getting Top50 with the longest average Duration. (171ms)
Getting Top50 with the longest SelfDuration. (174ms)
Getting Top50 with the longest average SelfDuration. (169ms)
Getting Top50 with the most hits. (172ms)
Duration: A: 00:01:08.8318156
Done. Try $trace.Top50Duration | Format-Table to get the report. There are also Top50A
lfAverage, Top50HitCount, AllLines and Events.
```

```
C:\p\pester [test-drive-perf ≡ +0 ~1 -0 !]> $trace.Top50SelfDuration | Format-Table Percent, HitCount, Duration, SelfDuration, Name, Line, Text

Percent HitCount Duration        SelfDuration     Name                      Line Text
------- -------- --------        ------------     ----                      ---- ----
 84.363     1297 00:00:59.0260538 00:00:04.9330607 Pester.Runtime.ps1       1459 . $_____parameters.ScriptBlock @_____innerSplat
  6.363      322 00:00:04.4516583 00:00:04.3534614 TestDrive.ps1            146 & $SafeCommands['Get-ChildItem'] -Recurse -Path $Path |
  6.139      322 00:00:04.2950883 00:00:04.2950883 TestDrive.ps1            192 & $SafeCommands['Get-ChildItem'] -Recurse -Path $Path
  5.944      942 00:00:04.1587479 00:00:04.1329128 Pester.Runtime.ps1       643 $____Pester.CurrentTest.ExpandedName = & ([ScriptBlock]::Create(('"' + ($____Pe
  9.127    52477 00:00:06.3858909 00:00:03.1988854 Pester.Runtime.ps1      1540 Write-PesterDebugMessage -Scope "RuntimeCore" $Message -ErrorRecord $ErrorRecor
  4.245      684 00:00:02.9702896 00:00:02.9702896 TestRegistry.ps1         60 "Microsoft.PowerShell.Core\Registry::" + (& $SafeCommands['Get-PSDrive'] -Name
  5.103      310 00:00:03.5702206 00:00:02.2142056 Pester.Runtime.ps1       279 & $ScriptBlock
  3.067      315 00:00:02.1458092 00:00:02.1399780 Pester.Runtime.ps1       389 $____Pester.CurrentBlock.ExpandedName = & ([ScriptBlock]::Create(('"' + ($____P
  2.259      322 00:00:01.5808275 00:00:01.5808275 TestDrive.ps1            190 $Path = (& $SafeCommands['Get-PSDrive'] -Name TestDrive).Root
   1.83      322 00:00:01.2801392 00:00:01.2801392 TestDrive.ps1            131 $drive = & $SafeCommands['Get-PSDrive'] -Name TestDrive -ErrorAction Ignore
  1.815      322 00:00:01.2697495 00:00:01.2697495 TestDrive.ps1            140 $Path = (& $SafeCommands['Get-PSDrive'] -Name TestDrive).Root
  1.725      322 00:00:01.2072556 00:00:01.2072556 TestDrive.ps1            141 if (& $SafeCommands['Test-Path'] -Path $Path ) {
   1.69      362 00:00:01.1821128 00:00:01.1821128 TestDrive.ps1             22 if (-not (& $script:SafeCommands['Test-Path'] TestDrive:\)) {
  1.638      322 00:00:01.1458985 00:00:01.1458985 TestDrive.ps1            191 if (& $SafeCommands['Test-Path'] -Path $Path ) {
  8.193      511 00:00:05.7326914 00:00:01.1030207 Pester.Runtime.ps1      1410 . $_____current @_____outerSplat
  1.441   257244 00:00:01.0082278 00:00:01.0082278 Pester.Utility.ps1       284 foreach ($p in $messagePreference) {
  1.402   257244 00:00:00.9808710 00:00:00.9808710 Pester.Utility.ps1       280 foreach ($s in $Scope) {
  3.431      178 00:00:02.4004916 00:00:00.8737101 Verify-AssertionFailed.ps1  11 $null = & $ScriptBlock
  0.958       40 00:00:00.6701974 00:00:00.6701974 TestDrive.ps1            244 & $SafeCommands['Remove-Item'] -Path $Path -Force -Recurse
  0.913     8358 00:00:00.6385139 00:00:00.6385139 HaveParameter.ps1         67 $token | & $SafeCommands['Add-Member'] Depth -MemberType NoteProperty -Value $j
  6.954       44 00:00:04.8658362 00:00:00.5961468 Pester.Runtime.ps1      2446 . $private:p @d
  0.844      199 00:00:00.5901775 00:00:00.5645992 Mock.ps1               1014 & ${Script Block} @___BoundParameters___ @___ArgumentList___
  0.803    85748 00:00:00.5616258 00:00:00.5616258 Pester.Utility.ps1       278 $messagePreference = $PesterPreference.Debug.WriteDebugMessagesFrom.Value
  0.777    85748 00:00:00.5439312 00:00:00.5439312 Pester.Utility.ps1       332 }
   0.74    85748 00:00:00.5180970 00:00:00.5180970 Pester.Utility.ps1       274 if (-not $PesterPreference.Debug.WriteDebugMessages.Value) {
  1.563       80 00:00:01.0934515 00:00:00.4537347 PesterThrow.ps1          51 $null = & $ActualValue
  0.637    85748 00:00:00.4457720 00:00:00.4457720 Pester.Utility.ps1       285 if ($s -like $p) {
  0.482    85748 00:00:00.3374487 00:00:00.3374487 Pester.Utility.ps1       279 $any = $false
   0.46      205 00:00:00.3217697 00:00:00.3120794 Mock.ps1               1133 & $private:_____mock_parameters.ScriptBlock @_____arguments
  2.294       37 00:00:01.6052573 00:00:00.3091407 test.ps1                 117 . $module $ScriptBlock
  0.429        1 00:00:00.2999982 00:00:00.2999982 Mock.Tests.ps1          1952 Add-Type -TypeDefinition '
```


20
18
16
14
12
10
8
6
4
2
0
Run 1
Run 2
Run 3
Run4
4.10.11
5.2.1
5.3.0-alpha4

Guido Oliveira @_Guido_Oliveira · May 2

PSProfiler has helped me identify "heat spots" on a script I found at a client, reducing the runtime from 11 minutes to about 1 minute :) thank you for working hard on this project

1 1 11

Install it:

```
Install-Module Profiler
```

github.com/nohwnd/profiler

In this talk:

- Performance considerations
- Profiling your code
  - Trace-Script
  - Flags
  - Invoke-Script
- Patterns
- New Pester Code Coverage

Context

My naïve approach to PowerShell performance

- Stare at code
- Guess how many times it is executed
- Hope changes will make it faster

EXAMPLE 1
SLOW CODE

The three problems

- Which code is slow?
- Which code is worth optimizing?
- How do I optimize it?

Which code is slow?

TRACE-SCRIPT

Profile a script, even with parameters

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

Profile a module, including module load

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

Profile your $profile script

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

Profile any scriptblock, as in my demos

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

EXAMPLE 2
TRACE-SCRIPT

EXAMPLE 3
DURATION
SELFDURATION

Which code is slow?

- Look at .Top50SelfDuration to see which code is slowest
- Look at .Top50Duration to see which parts of your script take the most time, and decide if that is appropriate

Which code is worth optimizing?

```
C:\p\pester [test-drive-perf ≡ +0 ~1 -0 !]> $trace.Top50SelfDuration | Format-Table Percent, HitCount, Duration, SelfDuration, Name, Line, Text

Percent HitCount Duration        SelfDuration     Name                      Line Text
------- -------- --------        ------------     ----                      ---- ----
 84.363     1297 00:00:59.0260538 00:00:04.9330607 Pester.Runtime.ps1       1459 . $_____parameters.ScriptBlock @_____innerSplat
  6.363      322 00:00:04.4516583 00:00:04.3534614 TestDrive.ps1            146 & $SafeCommands['Get-ChildItem'] -Recurse -Path $Path |
  6.139      322 00:00:04.2950883 00:00:04.2950883 TestDrive.ps1            192 & $SafeCommands['Get-ChildItem'] -Recurse -Path $Path
  5.944      942 00:00:04.1587479 00:00:04.1329128 Pester.Runtime.ps1       643 $____Pester.CurrentTest.ExpandedName = & ([ScriptBlock]::Create(('"' + ($____Pe
  9.127    52477 00:00:06.3858909 00:00:03.1988854 Pester.Runtime.ps1      1540 Write-PesterDebugMessage -Scope "RuntimeCore" $Message -ErrorRecord $ErrorRecor
  4.245      684 00:00:02.9702896 00:00:02.9702896 TestRegistry.ps1         60 "Microsoft.PowerShell.Core\Registry::" + (& $SafeCommands['Get-PSDrive'] -Name
  5.103      310 00:00:03.5702206 00:00:02.2142056 Pester.Runtime.ps1       279 & $ScriptBlock
  3.067      315 00:00:02.1458092 00:00:02.1399780 Pester.Runtime.ps1       389 $____Pester.CurrentBlock.ExpandedName = & ([ScriptBlock]::Create(('"' + ($____P
  2.259      322 00:00:01.5808275 00:00:01.5808275 TestDrive.ps1            190 $Path = (& $SafeCommands['Get-PSDrive'] -Name TestDrive).Root
   1.83      322 00:00:01.2801392 00:00:01.2801392 TestDrive.ps1            131 $drive = & $SafeCommands['Get-PSDrive'] -Name TestDrive -ErrorAction Ignore
  1.815      322 00:00:01.2697495 00:00:01.2697495 TestDrive.ps1            140 $Path = (& $SafeCommands['Get-PSDrive'] -Name TestDrive).Root
  1.725      322 00:00:01.2072556 00:00:01.2072556 TestDrive.ps1            141 if (& $SafeCommands['Test-Path'] -Path $Path ) {
   1.69      362 00:00:01.1821128 00:00:01.1821128 TestDrive.ps1             22 if (-not (& $script:SafeCommands['Test-Path'] TestDrive:\)) {
  1.638      322 00:00:01.1458985 00:00:01.1458985 TestDrive.ps1            191 if (& $SafeCommands['Test-Path'] -Path $Path ) {
  8.193      511 00:00:05.7326914 00:00:01.1030207 Pester.Runtime.ps1      1410 . $_____current @_____outerSplat
  1.441   257244 00:00:01.0082278 00:00:01.0082278 Pester.Utility.ps1       284 foreach ($p in $messagePreference) {
  1.402   257244 00:00:00.9808710 00:00:00.9808710 Pester.Utility.ps1       280 foreach ($s in $Scope) {
  3.431      178 00:00:02.4004916 00:00:00.8737101 Verify-AssertionFailed.ps1  11 $null = & $ScriptBlock
  0.958       40 00:00:00.6701974 00:00:00.6701974 TestDrive.ps1            244 & $SafeCommands['Remove-Item'] -Path $Path -Force -Recurse
  0.913     8358 00:00:00.6385139 00:00:00.6385139 HaveParameter.ps1         67 $token | & $SafeCommands['Add-Member'] Depth -MemberType NoteProperty -Value $j
  6.954       44 00:00:04.8658362 00:00:00.5961468 Pester.Runtime.ps1      2446 . $private:p @d
  0.844      199 00:00:00.5901775 00:00:00.5645992 Mock.ps1               1014 & ${Script Block} @___BoundParameters___ @___ArgumentList___
  0.803    85748 00:00:00.5616258 00:00:00.5616258 Pester.Utility.ps1       278 $messagePreference = $PesterPreference.Debug.WriteDebugMessagesFrom.Value
  0.777    85748 00:00:00.5439312 00:00:00.5439312 Pester.Utility.ps1       332 }
   0.74    85748 00:00:00.5180970 00:00:00.5180970 Pester.Utility.ps1       274 if (-not $PesterPreference.Debug.WriteDebugMessages.Value) {
  1.563       80 00:00:01.0934515 00:00:00.4537347 PesterThrow.ps1          51 $null = & $ActualValue
  0.637    85748 00:00:00.4457720 00:00:00.4457720 Pester.Utility.ps1       285 if ($s -like $p) {
  0.482    85748 00:00:00.3374487 00:00:00.3374487 Pester.Utility.ps1       279 $any = $false
   0.46      205 00:00:00.3217697 00:00:00.3120794 Mock.ps1               1133 & $private:_____mock_parameters.ScriptBlock @_____arguments
  2.294       37 00:00:01.6052573 00:00:00.3091407 test.ps1                 117 . $module $ScriptBlock
  0.429        1 00:00:00.2999982 00:00:00.2999982 Mock.Tests.ps1          1952 Add-Type -TypeDefinition '
```

Which code is worth optimizing?

- Look at Percent, and HitCount
- Look for code that runs many times, or for every item you have

How do I optimize it?

EXAMPLE 4
OPTIMIZE

How do I know I did a good job optimizing it?

FLAGS

EXAMPLE 5
FLAGS
INVOKE-SCRIPT

Do try this at home!

Try this:

```
# profile a script, even if it takes parameters
$trace = Trace-Script { .\myScript.ps1 -Name 'Jakub' -Age 32 }


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

How Profiler works?

IN 3 MINUTES

EXAMPLE 6
SET-PSDEBUG
EVENTS

Pester Code Coverage

Profiling x Code Coverage

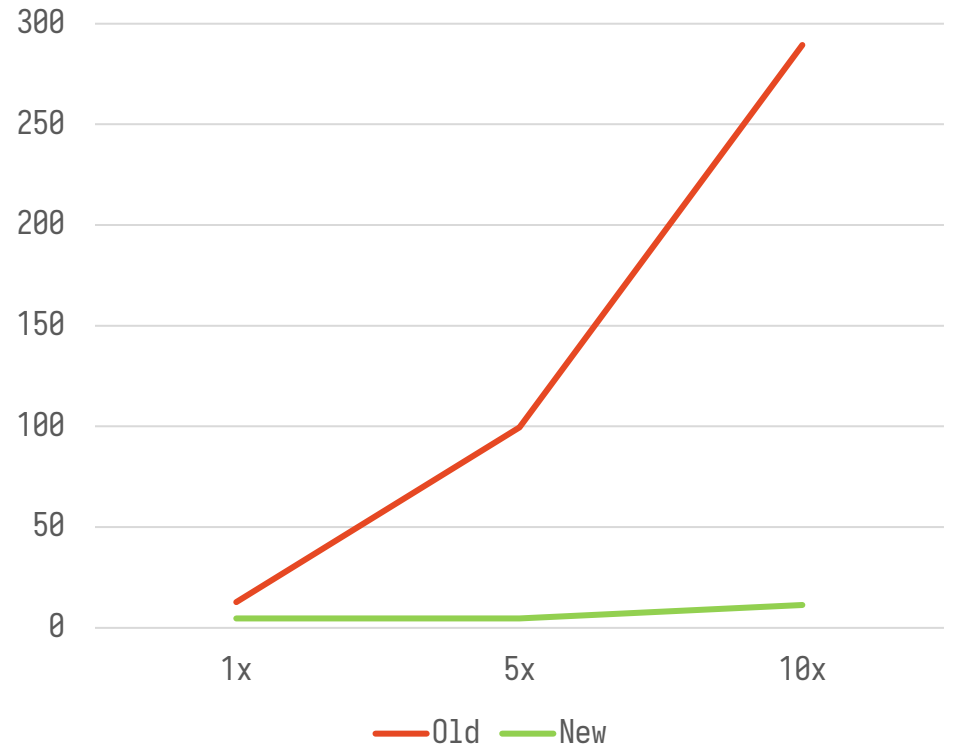
- All code that runs
- Which code run?
- ~~Which code did not run?~~
- How many times?
- How fast?

- ~~All~~ Selected code that runs
- Which code run?
- Which code did not run?
- At least once?
- ~~How fast?~~

Old vs New

	Breakpoints	New - Profiler
1x	13s	4.5s
5x	99s	4.6s
10x	289s	12s

(And for 10x, 289s vs 5s on second run)



```
Tests completed in 12.7s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 93.07% / 75%. 779 analyzed Commands in 29 Files.
```

```
Tests completed in 4.6s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 93.07% / 75%. 779 analyzed Commands in 29 Files.
```

5x more code:

```
Tests completed in 99.57s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 18.61% / 75%. 3,895 analyzed Commands in 145 Files.
```

```
Tests completed in 4.46s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 18.61% / 75%. 3,895 analyzed Commands in 145 Files.
```

10x more code:

```
Tests completed in 289.37s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 9.31% / 75%. 7,790 analyzed Commands in 290 Files.
```

```
Tests completed in 11.37s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 9.31% / 75%. 7,790 analyzed Commands in 290 Files.
```

10x more code, second run:

```
Tests completed in 291.97s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 9.31% / 75%. 7,790 analyzed Commands in 290 Files.
```

```
Tests completed in 5.02s
Tests Passed: 633, Failed: 0, Skipped: 0 NotRun: 0
Processing code coverage result.
Covered 9.31% / 75%. 7,790 analyzed Commands in 290 Files.
```

New coverage

Install-Module Pester **5.3.0-alpha4** -AllowPrerelease

$Configuration.CodeCoverage.UseBreakpoints = $false

Details, setup, comparing old CC and new CC:

Release 5.3.0-alpha2 · pester/Pester (github.com)

Install it:
Install-Module Profiler

Share feedback:
github.com/nohwnd/profiler

Follow me on twitter:
@nohwnd