

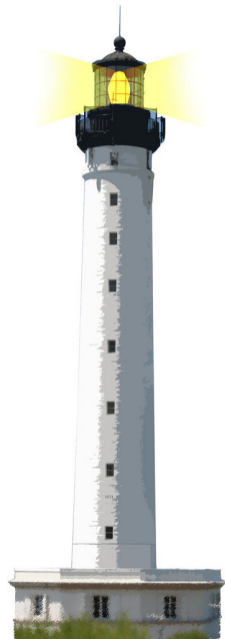
Benchmarking in Pharo

Damien Cassou, Stéphane Ducasse and Luc Fabresse

W5S09



<http://www.pharo.org>



Common Wisdom

If you did not profile your code you may have 40-50% speed up waiting for you.



Measuring Execution Speed

We create an expression and use [expression] timeToRun

```
[ 1000 factorial ] timeToRun
```

Comparing Two Executions

Is `select:, then collect:` **slower than** `select:thenCollect:`?



timeToRun example

```
| coll |  
coll := #(1 2 3 4 5 6 7 8 9 10) asOrderedCollection.  
[ 1000000 timesRepeat: [  
  (coll select: [:each | each > 5]) collect: [:i | i * i ] ]  
] timeToRun  
> "0:00:00:00.517"
```

```
| coll |  
coll := #(1 2 3 4 5 6 7 8 9 10) asOrderedCollection.  
[ 1000000 timesRepeat: [  
  coll  
  select: [:each | each > 5]  
  thenCollect: [:i | i * i ] ]  
] timeToRun  
> "0:00:00:00.362"
```



bench

- Returns how many times the code can get executed in 5 seconds
- Answer a string with meaningful description

```
[ 1000 factorial ] bench  
> '610.234 per second'
```

The higher the better!

```
[ 1234 factorial ] benchFor: 2 seconds
```

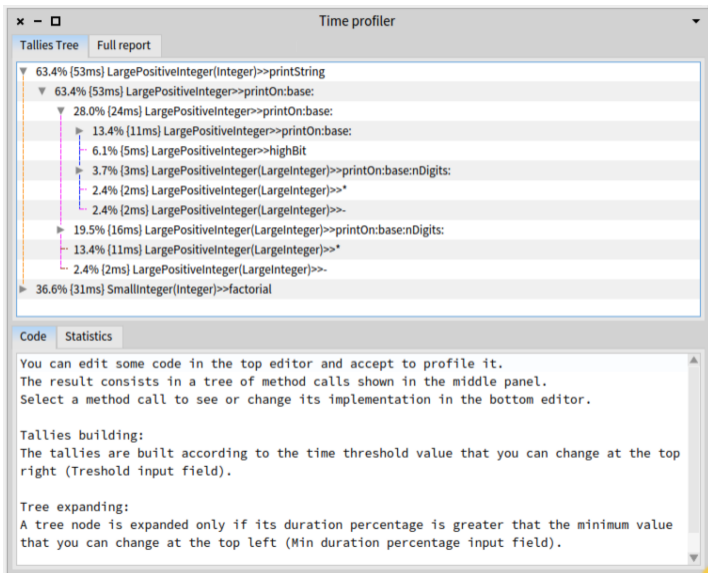
Time Profiler

- TimeProfiler: a sampling-based code profiler
- At regular interval, take information from the execution stack

TimeProfiler

```
spyOn: [ 20 timesRepeat: [  
    Transcript show: 1000 factorial printString ] ]
```

Time Profiler



The screenshot shows the 'Time profiler' window with two tabs: 'Tallies Tree' and 'Full report'. The 'Tallies Tree' tab is active, displaying a hierarchical tree of method calls. The root node is '63.4% [53ms] LargePositiveInteger(Integer)>>printString', which is expanded to show '63.4% [53ms] LargePositiveInteger>>printOn:base:'. This node is further expanded to show several sub-nodes, including '28.0% [24ms] LargePositiveInteger>>printOn:base:', '13.4% [11ms] LargePositiveInteger>>printOn:base:', '6.1% [5ms] LargePositiveInteger>>highBit', '3.7% [3ms] LargePositiveInteger(LargeInteger)>>printOn:base:nDigits:', '2.4% [2ms] LargePositiveInteger(LargeInteger)>>*', '2.4% [2ms] LargePositiveInteger(LargeInteger)>>-', '19.5% [16ms] LargePositiveInteger(LargeInteger)>>printOn:base:nDigits:', '13.4% [11ms] LargePositiveInteger(LargeInteger)>>*', and '2.4% [2ms] LargePositiveInteger(LargeInteger)>>-'. The bottom-most node is '36.6% [31ms] SmallInteger(Integer)>>factorial'. The 'Code' tab is also visible, containing instructions on how to use the profiler.

Tallies building:
The tallies are built according to the time threshold value that you can change at the top right (Treshold input field).

Tree expanding:
A tree node is expanded only if its duration percentage is greater that the minimum value that you can change at the top left (Min duration percentage input field).

Summary

- [anExpression] timeToRun
- [anExpression] bench
- TimeProfiler spyOn: [anExpression]
- Check Profiling Applications Chapter in **Deep into Pharo**
(at <http://books.pharo.org>)



A course by



and



in collaboration with



Inria 2020

Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France

<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>