

Profajleri i njihova vizualizacija za jezik Python

Anđelka Milovanović, David Popov
Jelisaveta Smiljanić, Petar Zečević

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

April 22, 2020

Sadržaj

- 1 Profajliranje
- 2 Profajliranje u Pythonu
 - Moduli cProfile i pstats
- 3 Alati za vizualizaciju profajliranja
- 4 Zaključak
- 5 Literatura

Profajliranje

- Šta je profajliranje?
- Podela na determinističko i statističko
 - determinističko - profajler beleži svako pokretanje i svako završavanje funkcija
 - statističko - uzorkovanje vrednosti u IP registru tokom rada programa
- Profajler može da meri vremensko i prostorno opterećenje

Profajliranje u Pythonu

- Interpreterski jezik
- Vremensko profajliranje
 - modul time
 - timeit
 - UNIX komanda **time** sa 3 vrste vremena
- Lociranje delova koda koji traju kratko/dugo?

Moduli cProfile i pstats

```
→ msnr python -m cProfile test_kod.py
Input number is: 20
    21933004 function calls (4004 primitive calls) in 5.100 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1   0.001   0.001   5.100    5.100 test_kod.py:1(<module>)
    1000   0.002   0.000   0.002   0.000 test_kod.py:1(for_fib)
21891000/1000   5.085   0.000   5.085   0.005 test_kod.py:10(recur_fib)
20000/1000   0.006   0.000   0.006   0.000 test_kod.py:16(tail_recur_fib)
21000/1000   0.006   0.000   0.006   0.000 test_kod.py:23(<lambda>)
      1   0.000   0.000   5.100   5.100 {built-in method builtins.exec}
      1   0.000   0.000   0.000   0.000 {built-in method builtins.print}
      1   0.000   0.000   0.000   0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Slika 1: cProfile izveštaj

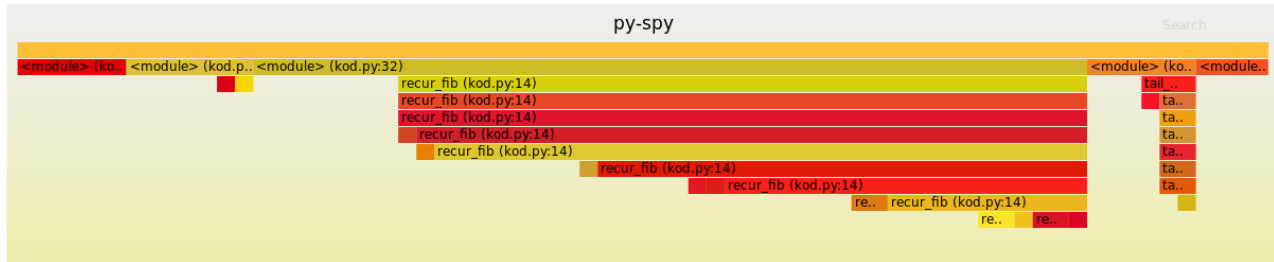
primer pStats

```
sortby = 'tottime'
ps = pstats.Stats(prof, stream=s).sort_stats(sortby)
```

Alati za vizualizaciju profajliranja

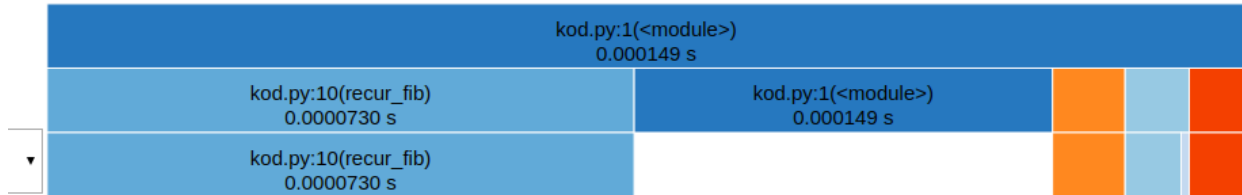
- Statistika koja se dobije profajliranjem je nečitljivija što je program kompleksniji
- U cilju preglednosti, razvijeni su alati za vizualizaciju
- Alati za vizualizaciju u Pythonu:
 - Py-Spy
 - SnakeViz
 - Pycallgraph
 - gprof2dot
 - vprof

Py-Spy

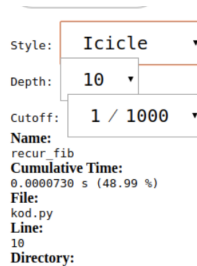


Slika 2: Py-Spy vizualizacija

SnakeViz

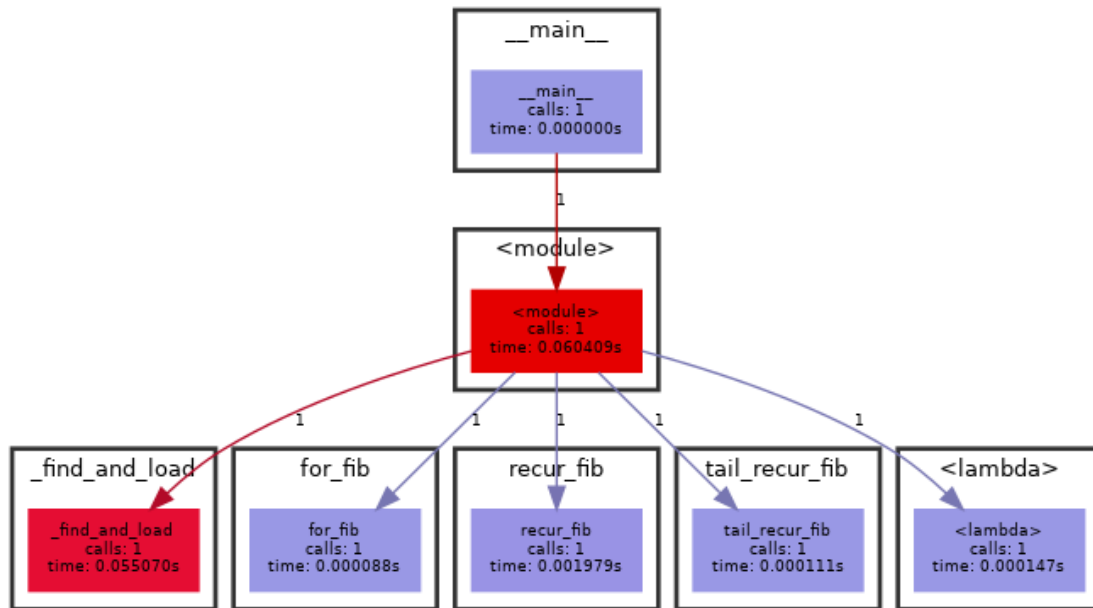


Slika 3: SnakeViz vizualizacija



Slika 4: SnakeViz statistike

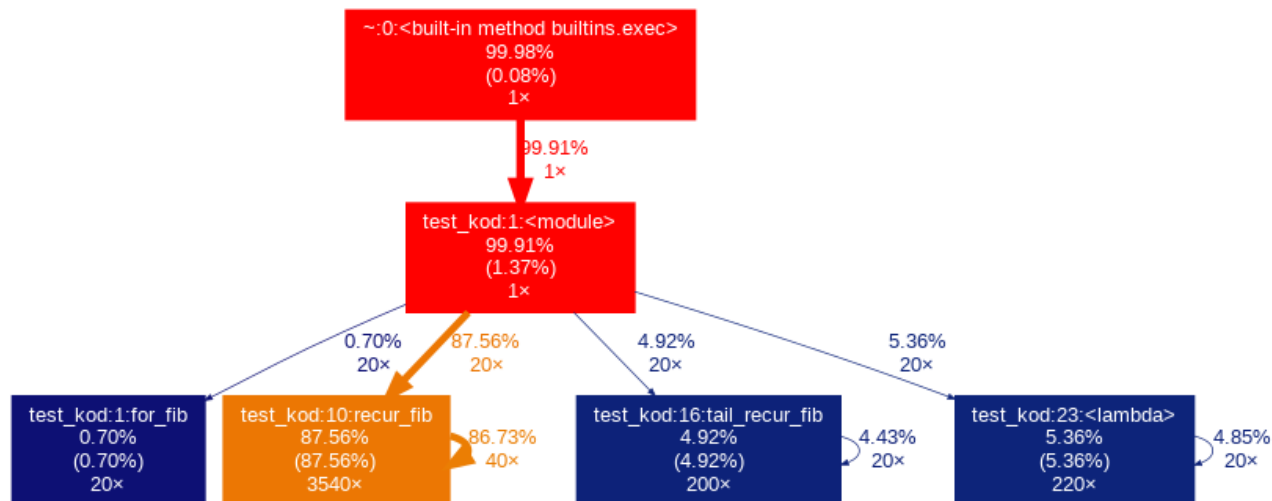
Pycallgraph



Generated by Python Call Graph v1.0.1
<http://pycallgraph.slowchop.com>

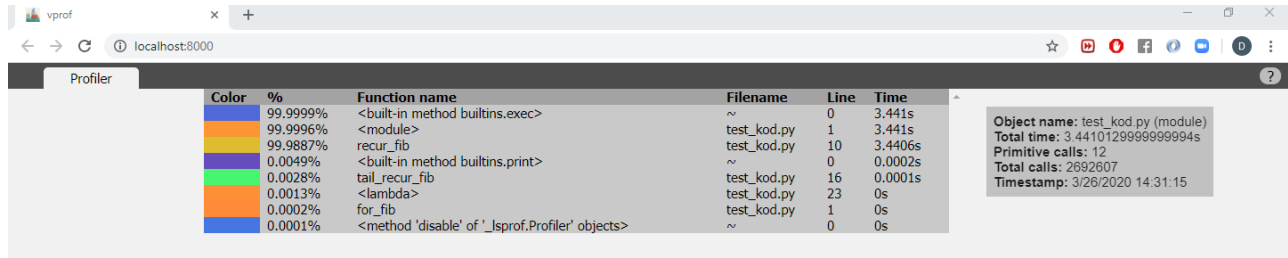
Slika 5: Pycallgraph vizualizacija za graf dubine 2

Gprof2dot



Slika 6: Gprof2dot vizualizacija

Vprof



The screenshot shows the Vprof web interface in a browser window. The address bar shows 'localhost:8000'. The main content area has a 'Profiler' tab selected. It displays a table with columns: Color, %, Function name, Filename, Line, and Time. The table lists various functions and their execution times. To the right of the table, a summary box provides additional statistics.

| Color | % | Function name | Filename | Line | Time |
|--------------|----------|--|-------------|------|---------|
| Blue | 99.9999% | <built-in method builtins.exec> | ~ | 0 | 3.441s |
| Orange | 99.9996% | <module> | test_kod.py | 1 | 3.441s |
| Yellow | 99.9887% | recur_fib | test_kod.py | 10 | 3.4406s |
| Purple | 0.0049% | <built-in method builtins.print> | ~ | 0 | 0.0002s |
| Green | 0.0028% | tail_recur_fib | test_kod.py | 16 | 0.0001s |
| Light Green | 0.0013% | <lambda> | test_kod.py | 23 | 0s |
| Light Orange | 0.0002% | for_fib | test_kod.py | 1 | 0s |
| Light Blue | 0.0001% | <method 'disable' of '_Isprof.Profiler' objects> | ~ | 0 | 0s |

Object name: test_kod.py (module)
Total time: 3.4410129999999994s
Primitive calls: 12
Total calls: 2692607
Timestamp: 3/26/2020 14:31:15

Slika 7: Vprof vizualizacija

Zaključak

- Osnovne ideje vremenskog profajliranja
- Dobijanje jasnije slike o životu funkcija i izvršavanju celina koda
- Ubrzavanje procesa detekcije kritičnih delova koda

Literatura



Jon Louis Bentley.

Writing Efficient Programs (Prentice-Hall Software Series).
Prentice Hall Ptr, 1982.



Python Software Foundation.

The Python Profilers, 1990-2020.

on-line at: <https://docs.python.org/2/library/profile.html>.



Gabriele Lanaro.

Python high performance programming.

Packt Publishing Ltd, 2013.

pages: 7-30.



Steve Scargall.

Profiling and Performance, pages 295–312.

01 2020.