

IVS - profiling

type_name_here (xnovot2j, xdrdla04, xhemza05, xsimap00)

19. dubna 2022

K profilingu byl využit nástroj *cProfile*. Profilovali jsme postupně s 10, 100 a 1000 náhodně vygenerovanými čísly typu *float* z intervalu $[-10\,000, 10\,000]$ a maximálně třemi desetinnými místy.

Na následujících snímcích jsou vidět výstupy profileru. Z důvodu přehlednosti je uváděno pouze 5 funkcí, ve kterých program strávil nejvíce času (viz sloupec *cumtime*). V ostatních funkcích bylo stráveno minimum času (< 0.000 s), proto je jejich uvedení zde irelevantní.

```
[libor@fedora src]$ python3 stddev.py < data.txt
136130760 function calls in 32.722 seconds

Ordered by: cumulative time
List reduced from 13 to 5 due to restriction <5>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1      0.000    0.000   32.722   32.722 stddev.py:24(standard_deviation)
3     22.652    7.551   32.722   10.907 mathlib.py:76(div)
136130723 10.069    0.000   10.069    0.000 mathlib.py:49(abs_val)
1      0.000    0.000    0.000    0.000 mathlib.py:166(rad)
1      0.000    0.000    0.000    0.000 stddev.py:28(<listcomp>)
```

Obr. 1 — Výstup profileru se vstupem o 10 číselných hodnotách

```
[libor@fedora src]$ python3 stddev.py < data.txt
77469492 function calls in 19.675 seconds

Ordered by: cumulative time
List reduced from 13 to 5 due to restriction <5>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1      0.000    0.000   19.675   19.675 stddev.py:24(standard_deviation)
3     13.845    4.615   19.674    6.558 mathlib.py:76(div)
77469275  5.829    0.000    5.829    0.000 mathlib.py:49(abs_val)
1      0.000    0.000    0.000    0.000 stddev.py:28(<listcomp>)
1      0.000    0.000    0.000    0.000 mathlib.py:166(rad)
```

Obr. 2 — Výstup profileru se vstupem o 100 číselných hodnotách

```

[libor@fedora src]$ python3 stddev.py < data.txt
90469044 function calls in 23.110 seconds

Ordered by: cumulative time
List reduced from 13 to 5 due to restriction <5>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1   0.000    0.000   23.110   23.110 stddev.py:24(standard_deviation)
      3  16.144    5.381   23.110    7.703 mathlib.py:76(div)
90467027  6.966    0.000    6.966    0.000 mathlib.py:49(abs_val)
      1   0.000    0.000    0.000    0.000 stddev.py:28(<listcomp>)
     1001  0.000    0.000    0.000    0.000 mathlib.py:140(exp)

```

Obr. 3 — Výstup profileru se vstupem o 1000 číselných hodnotách

Jak je možné vidět z výstupů, nejvíce času program stráví ve funkci `div()`, při optimalizaci tak bude vhodné se zaměřit právě na tuto funkci. Nejrychlejším řešením by bylo využití integrovaného operátoru `"\"` místo vlastního řešení algoritmu dělení.

Nejčastěji volána pak byla funkce `abs_val()`, vzhledem k její jednoduché implementaci zde však již není moc prostoru k optimalizaci. Na druhou stranu by bylo vhodné se podívat, z jakých dalších funkcí je tak často volána a na jednotlivých místech zvážit, jestli je její volání opravdu potřeba.