FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Praktické aspekty vývoje software

Profiling

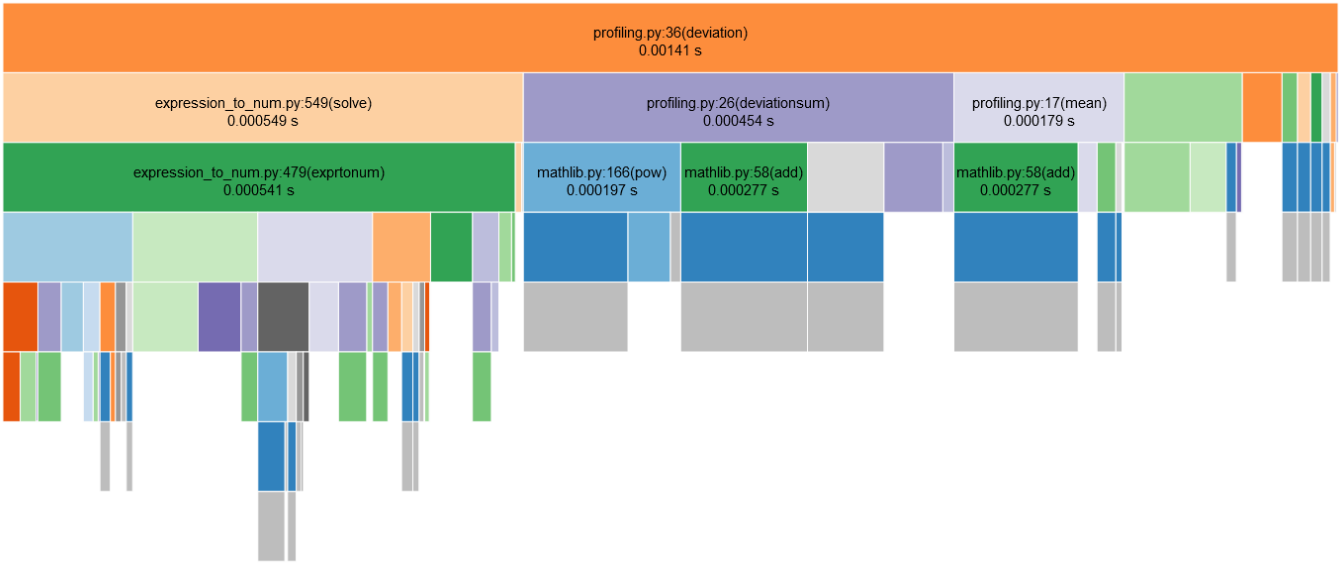
Martin Hlinský xhlins01
Petra Dudová xdudov02
Jan Kundrata xkundr07
Štěpán Nekula xnekul04

1 Úvod

V následující zprávě budu porovnávat vstupy a výstupy našeho profilovacího programu a stručně okomentuju, jaké pasáže bude potřeba optimalizovat. Některé funkce nebyly testovány, protože nebyly potřebné pro výpočet směrodatné odchylky, kterou profilovací program počítal.

2 Profiling (10 čísel)

Z následující vizualizace a zkráceného textového výstupu už je zde náznak toho, že nejvíce volání probíhá pro funkci `isnum()` a `isinstance()`. Čas tímhle u takhle nízkého počtu volání skoro není ovlivněn.



Obrázek 1: Profilování s 10 čísly na vstupu

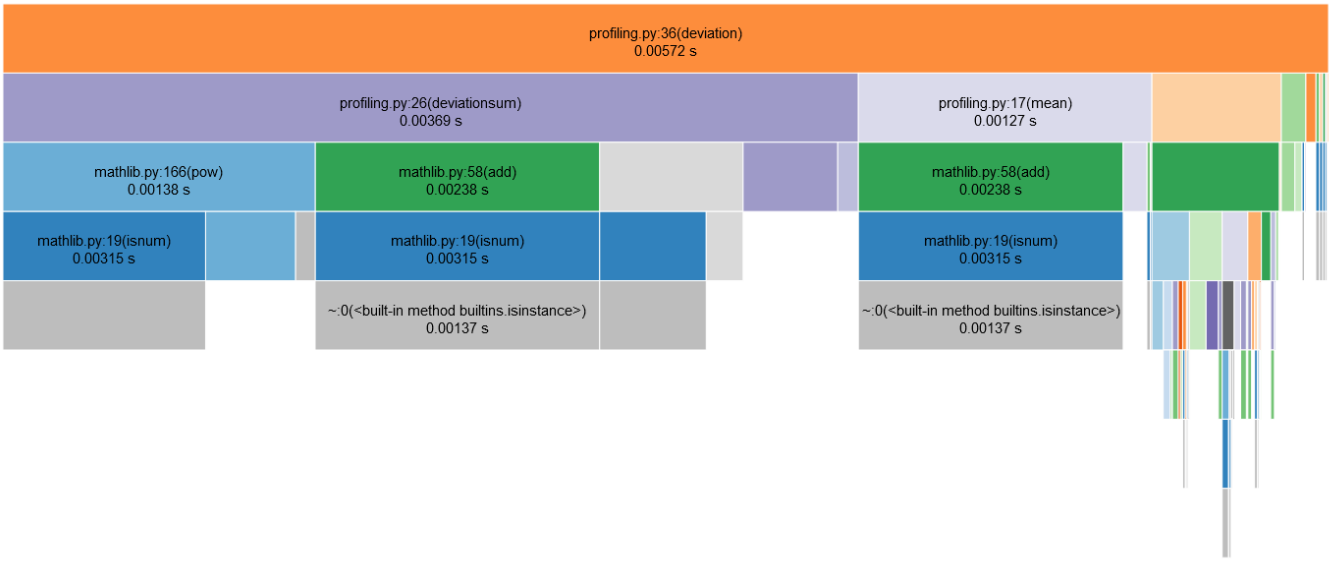
```
612 function calls in 0.001 seconds

Ordered by: internal time
List reduced from 34 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    92    0.000    0.000    0.000    0.000 mathlib.py:19(isnum)
   213    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
    21    0.000    0.000    0.000    0.000 mathlib.py:58(add)
     2    0.000    0.000    0.000    0.000 expression_to_num.py:170(rfunc)
    11    0.000    0.000    0.000    0.000 mathlib.py:166(pow)
    12    0.000    0.000    0.000    0.000 expression_to_num.py:156(rdupli)
     1    0.000    0.000    0.000    0.000 profiling.py:26(deviationsum)
    59    0.000    0.000    0.000    0.000 {built-in method builtins.len}
     1    0.000    0.000    0.000    0.000 mathlib.py:190(root)
    52    0.000    0.000    0.000    0.000 {method 'replace' of 'str' objects}
```

3 Profiling (100 čísel)

Zde vizualizace ukazuje, že nejvíce času program tráví ve funkci `deviationsum()`, která počítá sumu pro finální výpočet odchylky. Zde začíná být patrné, že na konci vždy zabere většinu času funkce, která zjišťuje, zda má na vstupu číslo (`isnum()`). Oproti minulému běhu s 10 čísly se zde zvedl počet volání funkce `isnum()` na 722 volání a `isinstance()` na 1 563 volání.



Obrázek 2: Profilování se 100 čísly na vstupu

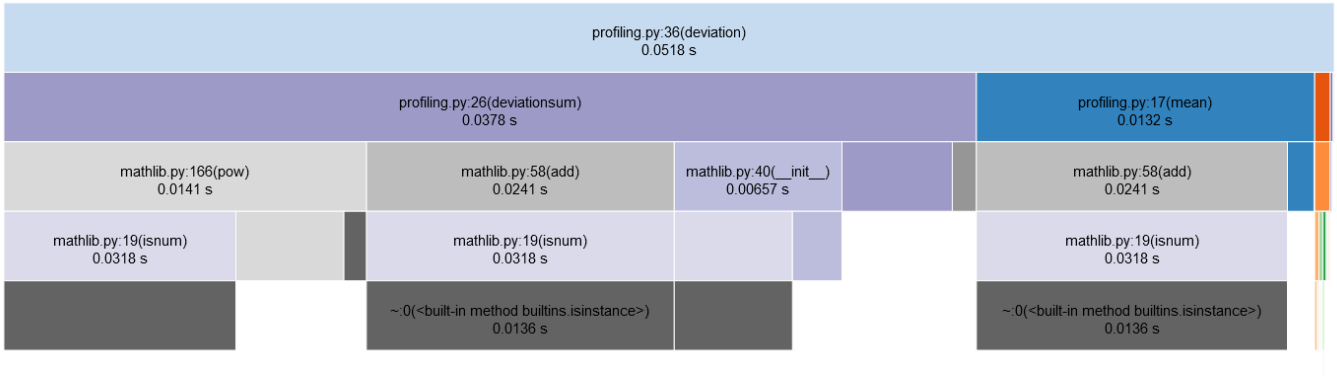
```
3049 function calls in 0.006 seconds

Ordered by: internal time
List reduced from 34 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   722    0.002    0.000    0.003    0.000 mathlib.py:19(isnum)
  1563    0.001    0.000    0.001    0.000 {built-in method builtins.isinstance}
   201    0.001    0.000    0.002    0.000 mathlib.py:58(add)
     1    0.000    0.000    0.004    0.004 profiling.py:26(deviationsum)
   101    0.000    0.000    0.001    0.000 mathlib.py:166(pow)
   106    0.000    0.000    0.001    0.000 mathlib.py:40(__init__)
     1    0.000    0.000    0.001    0.001 profiling.py:17(mean)
   103    0.000    0.000    0.000    0.000 mathlib.py:50(getvalue)
     2    0.000    0.000    0.000    0.000 expression_to_num.py:170(rfunc)
    71    0.000    0.000    0.000    0.000 {built-in method builtins.len}
```

4 Profiling (1 000 čísel)

Obrázek vypadá na první pohled skoro stejně, jako minulý, ale počet celkových volání všech funkcí se zvýšil na 27 374. Doba strávená ve funkcích `isnum()` a `isinstance()` nyní trvá déle, než půlku běhu celého programu. Počet volání těchto funkcí se zvýšil zhruba desetinásobně.



Obrázek 3: Profilování s 1 000 čísly na vstupu

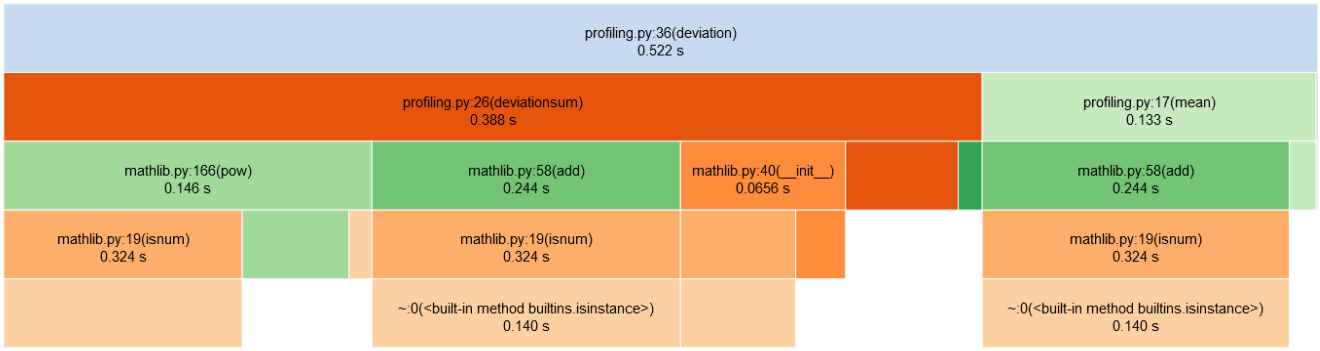
```
27374 function calls in 0.052 seconds

Ordered by: internal time
List reduced from 34 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
  7022    0.019    0.000    0.032    0.000 mathlib.py:19(isnum)
 15063    0.014    0.000    0.014    0.000 {built-in method builtins.isinstance}
  2001    0.006    0.000    0.024    0.000 mathlib.py:58(add)
     1    0.004    0.004    0.038    0.038 profiling.py:26(deviationsum)
  1001    0.004    0.000    0.014    0.000 mathlib.py:166(pow)
  1006    0.002    0.000    0.006    0.000 mathlib.py:40(__init__)
     1    0.001    0.001    0.013    0.013 profiling.py:17(mean)
  1003    0.001    0.000    0.001    0.000 mathlib.py:50(getvalue)
     2    0.000    0.000    0.000    0.000 expression_to_num.py:170(rfunc)
    83    0.000    0.000    0.000    0.000 {built-in method builtins.len}
```

5 Profiling (10 000 čísel)

Na závěr jsem zkusil profilování s 10 000 čísly, což je mírně nad rámec zadání, ale jde zde nejlépe vidět, že `isnum()` s `isinstance()` trvají nejdéle a z celkových 270 389 volání funkcí v celém programu se tyhle dvě funkce volají celkově 220 085x (což je něco málo přes 80 % všech volání).



Obrázek 4: Profilování s 10 000 čísly na vstupu

```
270389 function calls in 0.534 seconds

Ordered by: internal time
List reduced from 34 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
 70022    0.200    0.000    0.331    0.000 mathlib.py:19(isnum)
150063    0.140    0.000    0.140    0.000 {built-in method builtins.isinstance}
 20001    0.063    0.000    0.253    0.000 mathlib.py:58(add)
     1    0.046    0.046    0.394    0.394 profiling.py:26(deviationsum)
 10001    0.043    0.000    0.146    0.000 mathlib.py:166(pow)
 10006    0.020    0.000    0.067    0.000 mathlib.py:40(__init__)
     1    0.011    0.011    0.140    0.140 profiling.py:17(mean)
 10003    0.010    0.000    0.010    0.000 mathlib.py:50(getvalue)
     2    0.000    0.000    0.000    0.000 expression_to_num.py:170(rfunc)
    95    0.000    0.000    0.000    0.000 {built-in method builtins.len}
```

6 Závěr

Z profilování je patrné, že nejčastěji volané funkce jsou `isnum()` a `isinstance()`, kterou volá `isnum()`. Optimalizace by se tudíž měla provést na `isnum()` a to buď menší frekvencí volání, nebo úplně jinou implementací.

Dále je patrné, že se velmi často volá metoda `MathOperations.add()` kvůli sumaci čísel do `deviationsum()` a `mean()`. Velmi často je volaná metoda `__init__` kvůli dočasné proměnné v `deviationsum()`, která by se dala například nulovat po každém použití; zde by bylo ale nutno otestovat, zda by tahle implementace byla doopravdy optimalizací, nebo jaký vliv by to na běh programu mělo.