Fakulta informačních technologií
Vysoké učení technické v Brně

IVS - Praktické aspekty vývoje software

Profilování

Machač Michal (xmacha72)
Viliam Podhajecký (xpodha01)
Tomáš Kiss (xkisst00)

25. dubna 2020

1 Úvod

Podle požadavků v zadání byl vytvořený program pro výpočet výběrové směrodatné odchylky (anglicky *Standard deviation*). Výsledný program se skládá z jednoho spustitelného souboru vytvořeného v jazyce *Python*. Svůj vstup načítá ze standardního vstupu po řádcích. Řádky jsou dále členěny dle bílých znaků na další části. Při implementaci samotného výpočtu výběrové směrodatné odchylky byly použité funkce z vlastní matematické knihovny *matlib.py*.

2 Použité nástroje

Profilování skriptů v jazyce Python je možné dělat pomocí modulu *cProfile*[1]. Tento modul poskytuje statistické informace o běhu programu, např. čas potřebný pro běh jednotlivé části programu, počet volání funkcí či celkový čas běhu programu. Tyto informace je možné převést do přehledných zpráv.

3 Výsledky profilování

Program byl profilovaný s třemi různými počty vstupních dat. Výsledky běhů programu je možné vidět na obrázcích: pro deset 1, sto 2 a tisíc 3 vstupů.

```
         94 function calls in 0.000 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        3    0.000    0.000    0.000    0.000 codecs.py:318(decode)
       20    0.000    0.000    0.000    0.000 matlib.py:16(add)
        2    0.000    0.000    0.000    0.000 matlib.py:24(sub)
        1    0.000    0.000    0.000    0.000 matlib.py:32(mul)
        2    0.000    0.000    0.000    0.000 matlib.py:40(div)
       11    0.000    0.000    0.000    0.000 matlib.py:64(pow)
        1    0.000    0.000    0.000    0.000 matlib.py:75(sqrt)
        1    0.000    0.000    0.000    0.000 standard_deviation.py:46(arithmetic_mean)
        1    0.000    0.000    0.000    0.000 standard_deviation.py:59(solve_deviation)
        1    0.000    0.000    0.000    0.000 standard_deviation.py:81(<listcomp>)
        3    0.000    0.000    0.000    0.000 {built-in method _codecs.utf_8_decode}
       11    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.round}
       10    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 {method 'readlines' of '_io._IOBase' objects}
       10    0.000    0.000    0.000    0.000 {method 'rstrip' of 'str' objects}
       10    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
```

Obrázek 1: Statistické výsledky pro 10 vstupů

[1] https://docs.python.org/3.6/library/profile.html#module-cProfile

```
      724 function calls in 0.000 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     3    0.000    0.000    0.000    0.000 codecs.py:318(decode)
   200    0.000    0.000    0.000    0.000 matlib.py:16(add)
     2    0.000    0.000    0.000    0.000 matlib.py:24(sub)
     1    0.000    0.000    0.000    0.000 matlib.py:32(mul)
     2    0.000    0.000    0.000    0.000 matlib.py:40(div)
   101    0.000    0.000    0.000    0.000 matlib.py:64(pow)
     1    0.000    0.000    0.000    0.000 matlib.py:75(sqrt)
     1    0.000    0.000    0.000    0.000 standard_deviation.py:46(arithmetic_mean)
     1    0.000    0.000    0.000    0.000 standard_deviation.py:59(solve_deviation)
     1    0.000    0.000    0.000    0.000 standard_deviation.py:81(<listcomp>)
     3    0.000    0.000    0.000    0.000 {built-in method _codecs.utf_8_decode}
   101    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
     3    0.000    0.000    0.000    0.000 {built-in method builtins.len}
     1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
     1    0.000    0.000    0.000    0.000 {built-in method builtins.round}
   100    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
     1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
     1    0.000    0.000    0.000    0.000 {method 'readlines' of '_io._IOBase' objects}
   100    0.000    0.000    0.000    0.000 {method 'rstrip' of 'str' objects}
   100    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
```

Obrázek 2: Statistické výsledky pro 100 vstupů

```
      7028 function calls in 0.002 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     5    0.000    0.000    0.000    0.000 codecs.py:318(decode)
  2000    0.000    0.000    0.000    0.000 matlib.py:16(add)
     2    0.000    0.000    0.000    0.000 matlib.py:24(sub)
     1    0.000    0.000    0.000    0.000 matlib.py:32(mul)
     2    0.000    0.000    0.000    0.000 matlib.py:40(div)
  1001    0.000    0.000    0.000    0.000 matlib.py:64(pow)
     1    0.000    0.000    0.000    0.000 matlib.py:75(sqrt)
     1    0.000    0.000    0.000    0.000 standard_deviation.py:46(arithmetic_mean)
     1    0.001    0.001    0.002    0.002 standard_deviation.py:59(solve_deviation)
     1    0.000    0.000    0.000    0.000 standard_deviation.py:81(<listcomp>)
     5    0.000    0.000    0.000    0.000 {built-in method _codecs.utf_8_decode}
  1001    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
     3    0.000    0.000    0.000    0.000 {built-in method builtins.len}
     1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
     1    0.000    0.000    0.000    0.000 {built-in method builtins.round}
  1000    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
     1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
     1    0.000    0.000    0.000    0.000 {method 'readlines' of '_io._IOBase' objects}
  1000    0.000    0.000    0.000    0.000 {method 'rstrip' of 'str' objects}
  1000    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
```

Obrázek 3: Statistické výsledky pro 1000 vstupů

Z výsledkov jednotlivých behov programu je dobre vidieť že najčastejšie sú volané funkcie *add()* a *pow()*. Veľké množstvo volania funkcie add(), presnejšie $2 * N$ kde N je počet všetkých vstupov, je spôsobené s tým že v rámci vzorca je potrebné dvakrát vypočítať súčet všetkých čísel. Funkcia pre mocninu sa volá vždy $N + 1$ krát. Raz pri umocňovaní aritmetického priemeru a Nkrát pri výpočte druhých mocnín všetkých vstupov.

Za povšimnutie stojí aj počet volaní funkcií pri načítavaní a následnej prekonvertovaní vstupov na *float*.

Optimalizácia by mohla byť zameraná na zníženie počtu volaní funkcií používaných pri výpočtu

částí vzorca. Možnosťou by bolo pralelizovať výpočet napríklad súčtu všetkých vstupných čísel.

Z výsledků jednotlivých běhů programu je dobře vidět, že nejčastěji volané funkce jsou add() a *pow()*. Velké množství volání funkcí add(), přesněji $2 * N$, kde N je počet všech vstupů, je způsobené tím, že v rámci vzorce je potřeba dvakrát vypočítat součet všech čísel. Funkce pro mocninu se volá vždy $N + 1$ krát. Jednou při umocňování aritmetického průměru, a Nkrát při výpočtu druhých mocnin všech vstupů.

Za povšimnutí stojí i počet volání funkcí při načítání, a následném překonvertování vstupů na *float*.

Optimalizace by mohla být zaměřená i na snížení počtu volání funkcí používaných při výpočtu částí vzorce. Možností by bylo paralelizovat výpočet např. součtu všech vstupních čísel.