IVS - profiling

Ačkostačí

Duben 2023

Profiling

Pomocí funkcí z Vaší matematické knihovny vytvořte program (jako samostatný spustitelný soubor) pro výpočet výběrové směrodatné odchylky z posloupnosti čísel, kterou program čte ze standardního vstupu (v C např. pomocí funkce *scanf*) až do konce souboru a musí být schopen načíst min. 1000 čísel (zkontrolujeme). Na vstupu boudou pouze čísla oddělená bílými znaky (mezera, konec řádku nebo tabulátor) a jejich počet není předem dán.

Program profilujte se vstupy o velikosti 10, 10^3 a 10^6 číselných hodnot. Odevzdejte protokol obsahující výstup profileru a stručné shrnutí – ve kterých místech program tráví nejvíce času a uveďte, na co se při optimalizaci kódu nejlépe zaměřit.

Výsledky a práce

K profilingu byl požit cProfile v Pythonu,níže jsou poskytnuty výsledky pro vstupy vstupy o velikosti 10, 10^3 a 10^6 číselných hodnot a analýza těchto výsledků.

1. 10

```
2723.809252580739
         60 function calls in 0.004 seconds

   Ordered by: call count

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
       25    0.000    0.000    0.000    0.000 calculator_logic.py:51(execute)
       11    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
       11    0.000    0.000    0.000    0.000 {built-in method builtins.pow}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        2    0.000    0.000    0.000    0.000 {built-in method builtins.sum}
        1    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.004    0.004    0.004    0.004 {method 'read' of
'_io.TextIOWrapper' objects}
        1    0.000    0.000    0.000    0.000 {built-in method
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
        1    0.000    0.000    0.000    0.000 codecs.py:319(decode)
        1    0.000    0.000    0.000    0.000 profiling.py:8(<listcomp>)
        1    0.000    0.000    0.000    0.000 profiling.py:5(derivation)
```

V prvním výstupu vidíme, že kód byl volán 60krát a celkově trvalo jeho spuštění 0,004 sekundy. Funkce execute v souboru calculator_logic.py byla volána 25krát, ale na její spuštění se nevyžadoval žádný čas. Většina ostatních funkcí volaných byly vestavěné metody Pythonu (např. isinstance, pow a len), které také nevyžadovaly čas na své spuštění. Jediný významný časový náklad pocházel z čtení vstupu z souboru (0,004 sekundy).

2. 10^3

```
2889.2154666256006
         4020 function calls in 0.005 seconds

   Ordered by: call count

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     2005    0.001    0.000    0.002    0.000 calculator_logic.py:51(execute)
     1001    0.000    0.000    0.000    0.000 {built-in method
builtins.isinstance}
     1001    0.000    0.000    0.000    0.000 {built-in method builtins.pow}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        2    0.000    0.000    0.000    0.000 {built-in method builtins.sum}
        1    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.003    0.003    0.003    0.003 {method 'read' of
'_io.TextIOWrapper' objects}
        1    0.000    0.000    0.000    0.000 {built-in method
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
        1    0.000    0.000    0.000    0.000 codecs.py:319(decode)
        1    0.001    0.001    0.002    0.002 profiling.py:8(<listcomp>)
        1    0.000    0.000    0.002    0.002 profiling.py:5(derivation)
```

V druhém výstupu byl kód volán 4020krát a celkově trvalo jeho spuštění 0,005 sekundy. Funkce execute byla volána 2005krát a celkově trvalo její spuštění 0,001 sekundy. Ostatní funkce byly podobné jako v prvním výstupu.

3. 10^6

```
288.8571956417237

         4000020 function calls in 2.468 seconds


   Ordered by: call count


   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
  2000005    1.224    0.000    1.653    0.000 calculator_logic.py:51(execute)
  1000001    0.165    0.000    0.165    0.000 {built-in method builtins.isinstance}
  1000001    0.264    0.000    0.264    0.000 {built-in method builtins.pow}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        2    0.022    0.011    0.022    0.011 {built-in method builtins.sum}
        1    0.092    0.092    0.092    0.092 {method 'split' of 'str' objects}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.021    0.021    0.028    0.028 {method 'read' of '_io.TextIOWrapper'
objects}
        1    0.008    0.008    0.008    0.008 {built-in method _codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
        1    0.000    0.000    0.008    0.008 codecs.py:319(decode)
        1    0.660    0.660    2.313    2.313 profiling.py:8(<listcomp>)
        1    0.013    0.013    2.347    2.347 profiling.py:5(derivation)
```

V třetím výstupu byl kód volán 4 000 020krát a celkově trvalo jeho spuštění 2,468 sekundy. Funkce execute byla volána 2 000 005krát a celkově trvalo její spuštění 1,224 sekundy. Ostatní funkce byly podobné jako v předchozích výstupech.