Profiling a návrh na optimalizácie

Tím G.I.I.T.

27. apríla 2022

Výpočet smerodajnej odchýlky sme s použtím našej matematickej knižnice profilovali so vstupmi o veľkosti: 10, 100, 1000 a 1 000 000 hodnôt. Použili sme profiler pre python cProfile, ktorého výstup je priamo v súbore "vystup.out".

Pri prechádzaní výsledkov profilingu sme zistili, že najviac času pri behu nášho programu zaberá volanie a funkcie modulu re. Tento výsledok je spôsobený tým, že niektoré funkcie našej matematickej knižnice prevádzajú textový vstup na matematické operácie s využitím zložitejších regulárnych výrazov. Konkrétne, zlepšenie by vyžadovala napríklad funkcia "__find_all_expressions_power_d", ktorá vo veľkej miere využíva regulárne výrazy.

Ďalej zaznamenali veľkú mieru času sráveného vo funkcii, napríklad funkcia root alebo power. Hlavne power, keďže aj samotná funkcia root využíva funkciu power (napr. sqrt(x)= x^(1/2)). S optimalizáciou tejto funckie by tiež pomohlo nájsť rýchlejší spôsob ako nadmerné využívanie regulárnych výrazov.

```
ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    11    0.020    0.002    0.020    0.002 {built-in method builtins.input}
 45/10    0.002    0.000    0.004    0.000 sre_parse.py:493(_parse)
     5    0.001    0.000    0.001    0.000 {built-in method marshal.loads}
106/10    0.001    0.000    0.003    0.000 sre_compile.py:71(_compile)
    39    0.001    0.000    0.001    0.000 sre_compile.py:276(_optimize_charset)
    35    0.001    0.000    0.005    0.000 {built-in method builtins.__build_class__}
   633    0.001    0.000    0.001    0.000 sre_parse.py:233(__next)
   650    0.000    0.000    0.001    0.000 sre_parse.py:164(__getitem__)
119/23    0.000    0.000    0.001    0.000 sre_parse.py:174(getwidth)
     2    0.000    0.000    0.000    0.000 {built-in method _imp.create_dynamic}
   530    0.000    0.000    0.001    0.000 sre_parse.py:254(get)
    21    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap_external>:1536(find_spec)
 31/10    0.000    0.000    0.004    0.000 sre_parse.py:435(_parse_sub)
94/979    0.000    0.000    0.000    0.000 {built-in method builtins.len}
   850    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
    39    0.000    0.000    0.000    0.000 sre_parse.py:355(_escape)
  1157    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
    38    0.000    0.000    0.000    0.000 {built-in method posix.stat}
    10    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:132(_path_split)
     9    0.000    0.000    0.000    0.000 {method 'read' of '_io.BufferedReader' objects}
     4    0.000    0.000    0.000    0.000 {built-in method _imp.source_hash}
    12    0.000    0.000    0.003    0.000 functions.py:20(__find_all_expressions_power_d)
    53    0.000    0.000    0.008    0.000 re.py:288(_compile)
```

Obrázok 2: Prvých 25 riadkov výstupu profileru pri spustení s 10 hodnotami

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1000002    7.374    0.000   23.784    0.000 functions.py:20(__find_all_expressions_power_d)
2000005    4.520    0.000    4.520    0.000 {method 'search' of 're.Pattern' objects}
5000124    3.349    0.000    3.349    0.000 {method 'join' of 'str' objects}
1000001    2.667    0.000    2.670    0.000 {built-in method builtins.input}
4000013    2.241    0.000    3.217    0.000 re.py:288(_compile)
2000004    1.491    0.000    7.624    0.000 re.py:197(search)
1000001    1.334    0.000   25.118    0.000 functions.py:268(power)
      1    1.114    1.114    3.943    3.943 profiling.py:14(get_input)
1000003    0.979    0.000    0.979    0.000 {method 'findall' of 're.Pattern' objects}
4000810    0.968    0.000    0.968    0.000 {built-in method builtins.isinstance}
1000002    0.835    0.000    0.835    0.000 {method 'split' of 're.Pattern' objects}
1000003    0.772    0.000    2.568    0.000 re.py:232(findall)
1000002    0.740    0.000    2.354    0.000 re.py:222(split)
      1    0.437    0.437   25.563   25.563 profiling.py:30(inside_function)
1000002    0.332    0.000    0.332    0.000 {method 'replace' of 'str' objects}
1000002    0.183    0.000    0.183    0.000 {method 'group' of 're.Match' objects}
1000000    0.159    0.000    0.159    0.000 {method 'split' of 'str' objects}
      1    0.009    0.009    0.009    0.009 {built-in method builtins.sum}
    965    0.002    0.000    0.002    0.000 {built-in method _codecs.utf_8_decode}
  45/10    0.002    0.000    0.004    0.000 sre_parse.py:493(_parse)
      5    0.001    0.000    0.001    0.000 {built-in method marshal.loads}
```

Obrázok 1: Prvých 25 riadkov výstupu profileru pri spustení s 1 000 000 hodnotami

Pre ukážku sme sa rozhodli priamo v texte zahrnúť dva screenshoty výstupu profileru, kde ukazujeme v oboch prípadoch prvých dvadsať päť riadkov. Vidíme, že pri krátkom vstupe sa volanie našich funkcií takmer vôbec neprejavilo, no pri dlhom vstupe, bola naša už spomínaná funkcia časovo najnáročnejšia.

Zvyšok výstupu profile nájdete v súbore vystup.out.