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.

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.

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     11    0.031    0.003    0.031    0.003 {built-in method builtins.input}
   27/7    0.001    0.000    0.002    0.000 sre_parse.py:493(_parse)
   58/7    0.001    0.000    0.001    0.000 sre_compile.py:71(_compile)
    495    0.000    0.000    0.000    0.000 sre_parse.py:233(__next)
      3    0.000    0.000    0.000    0.000 {built-in method marshal.loads}
     23    0.000    0.000    0.000    0.000 sre_compile.py:276(_optimize_charset)
    358    0.000    0.000    0.000    0.000 sre_parse.py:164(__getitem__)
    415    0.000    0.000    0.001    0.000 sre_parse.py:254(get)
  62/11    0.000    0.000    0.000    0.000 sre_parse.py:174(getwidth)
     12    0.000    0.000    0.003    0.000 functions.py:48(__find_all_expressions_power_d)
    553    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
      1    0.000    0.000    0.000    0.000 {built-in method _imp.create_dynamic}
   19/7    0.000    0.000    0.002    0.000 sre_parse.py:435(_parse_sub)
634/569    0.000    0.000    0.000    0.000 {built-in method builtins.len}
      9    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1536(find_spec)
     61    0.000    0.000    0.005    0.000 re.py:288(_compile)
    647    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
     24    0.000    0.000    0.000    0.000 {method 'search' of 're.Pattern' objects}
    111    0.000    0.000    0.000    0.000 {method 'join' of 'str' objects}
     18    0.000    0.000    0.000    0.000 {built-in method posix.stat}
    147    0.000    0.000    0.000    0.000 sre_parse.py:160(__len__)
    122    0.000    0.000    0.000    0.000 sre_parse.py:249(match)
      7    0.000    0.000    0.000    0.000 sre_compile.py:536(_compile_info)
      7    0.000    0.000    0.004    0.001 sre_compile.py:759(compile)
      4    0.000    0.000    0.000    0.000 {built-in method io.open_code}
```

Obrázok 1: Prvých 25 riadkov výstupu profileru pri spustení s 10 hodnotami

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1000002    7.644    0.000   24.758    0.000 functions.py:48(__find_all_expressions_power_d)
2000004    4.597    0.000    4.597    0.000 {method 'search' of 're.Pattern' objects}
5000061    3.531    0.000    3.531    0.000 {method 'join' of 'str' objects}
5000011    3.003    0.000    4.304    0.000 re.py:288(_compile)
1000001    2.751    0.000    2.757    0.000 {built-in method builtins.input}
1000001    2.052    0.000   28.746    0.000 calc.py:25(power)
2000004    1.505    0.000    4.616    0.000 re.py:232(findall)
2000004    1.502    0.000    7.804    0.000 re.py:197(search)
2000004    1.389    0.000    1.389    0.000 {method 'findall' of 're.Pattern' objects}
5000503    1.296    0.000    1.296    0.000 {built-in method builtins.isinstance}
      1    1.259    1.259    4.197    4.197 profiling.py:19(get_input)
1000002    0.937    0.000    0.937    0.000 {method 'split' of 're.Pattern' objects}
1000002    0.757    0.000    2.570    0.000 re.py:222(split)
      1    0.480    0.480   29.236   29.236 profiling.py:48(inside_function)
1000002    0.336    0.000    0.336    0.000 {method 'replace' of 'str' objects}
1000002    0.194    0.000    0.194    0.000 {method 'group' of 're.Match' objects}
1000000    0.181    0.000    0.181    0.000 {method 'split' of 'str' objects}
      1    0.009    0.009    0.009    0.009 {built-in method builtins.sum}
    965    0.003    0.000    0.006    0.000 codecs.py:319(decode)
    965    0.003    0.000    0.003    0.000 {built-in method _codecs.utf_8_decode}
   27/7    0.001    0.000    0.003    0.000 sre_parse.py:493(_parse)
   58/7    0.001    0.000    0.001    0.000 sre_compile.py:71(_compile)
    495    0.001    0.000    0.001    0.000 sre_parse.py:233(__next)
      3    0.000    0.000    0.000    0.000 {built-in method marshal.loads}
    415    0.000    0.000    0.001    0.000 sre_parse.py:254(get)
```

Obrázok 2: Prvých 25 riadkov výstupu profileru pri spustení s 1 000 000 hodnotami

Zvyšok výstupu profile nájdete v súbore vystup.out.