Praktické aspekty vývoje software

2023/2024

Project 2 – Profiling report

Team ”Kuře před E112”

Jakub Havlík (xhavlij00)
Martin Vaculík (xvaculm00)
David Bujzaš (xbujzad00)
Erik Polák (xpolake00))

Brno, April 17, 2024

1 Assignment

Using functions from your mathematical library, create program (separate executable file) that calculates the standard deviation from a sequence of numbers, which are read from the standard input (e.g. in C using scanf) to the end of the file and it has to load min. 1000 numbers. On stdin there will be only numbers separated using whitespace characters (space, end of a line, tab) and their number is not defined in advance. Formula for the standard deviation, which is used:

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Figure 1: Formula for the standard deviation from assignment

2 Profiling implementation

For implementation of profiling program our team chose profiler for Python - cProfile. It's output is typically a flat graph, from which we can analyze what we have to pay attention to and what to optimalize. The output is sorted by "tottime", which means the total time spent in each function. Apart from that we also analyze the number of times each function has been called.

According to the documentation, cProfile function is based on "deterministic profiling", which means that the code doesn't have to be instrumented, because Python is a scripting language and the code is already interpreted.

3 Measuring

The program was run with pseudo-randomlz generated inputs, values were typically in the range of thousands. They were used to calculate the standard deviation. For this measuring we considered 10^1,10^3,10^6 inputs[1], additionally even 50, 100 millions and 1 billion. For individual outputs, see section 5

Program was executed using the command python3 profiling.py ¡ input_numofinputs.txt

The measuring was done on a system with Python version 3.10.12, Windows 10 22H2 instaled. Device specifications: Intel Core i5-12400F, 16GB RAM DDR4 2667 MHz, RTX 3060Ti, MSI PRO B660-A, Samsung M.2 NVMe 980.

4 Analysis

Based on the measured values, it's clear that using a huge set and number of inputs, the mathematical library calculates slowly. One of the reasons can be the speed of Python itself, but a huge problem was function pow implemented by use, which calculates the power of a number. In this function program spent too much time and thus it needs to be optimalized. Other than that big burden was function contains_decimal, which validates if a number is decimal or not to correctly evaluate the input numbers. It would be ideal to implement this function differently or try to reduce the number of times the function has been called.

[1] For huge amount of inputs the program was really slow, but we could measure unlimited number of inputs

5 Outputs

Measuring with nearly each set of inputs was done repeatedly and the time varied only by hundredths of a second.

```
2910.1362758696464
         62 function calls in 0.000 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 profiling.py:30(profiler_function)
        2    0.000    0.000    0.000    0.000 mathlib.py:50(sub)
        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.exec}
       11    0.000    0.000    0.000    0.000 mathlib.py:108(pow)
        2    0.000    0.000    0.000    0.000 codecs.py:319(decode)
       20    0.000    0.000    0.000    0.000 mathlib.py:40(add)
        5    0.000    0.000    0.000    0.000 mathlib.py:23(contains_decimals)
       10    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        2    0.000    0.000    0.000    0.000 {built-in method _codecs.utf_8_decode}
        2    0.000    0.000    0.000    0.000 mathlib.py:73(mul)
        1    0.000    0.000    0.000    0.000 mathlib.py:83(root)
        1    0.000    0.000    0.000    0.000 <string>:1(<module>)
        2    0.000    0.000    0.000    0.000 mathlib.py:60(div)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figure 2: 10 inputs

```
2906.8049270413208
         4519 function calls in 0.001 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.001    0.001    0.001    0.001 profiling.py:30(profiler_function)
     1001    0.000    0.000    0.000    0.000 mathlib.py:108(pow)
     2000    0.000    0.000    0.000    0.000 mathlib.py:40(add)
      500    0.000    0.000    0.000    0.000 mathlib.py:23(contains_decimals)
     1000    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        1    0.000    0.000    0.001    0.001 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        3    0.000    0.000    0.000    0.000 codecs.py:319(decode)
        3    0.000    0.000    0.000    0.000 {built-in method _codecs.utf_8_decode}
        1    0.000    0.000    0.001    0.001 <string>:1(<module>)
        2    0.000    0.000    0.000    0.000 mathlib.py:60(div)
        2    0.000    0.000    0.000    0.000 mathlib.py:73(mul)
        1    0.000    0.000    0.000    0.000 mathlib.py:83(root)
        2    0.000    0.000    0.000    0.000 mathlib.py:50(sub)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figure 3: 1000 inputs

```
2884.83040228174
         4501610 function calls in 1.142 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.690    0.690    1.142    1.142 profiling.py:30(profiler_function)
  1000001    0.182    0.000    0.265    0.000 mathlib.py:108(pow)
  2000000    0.115    0.000    0.115    0.000 mathlib.py:40(add)
   499547    0.083    0.000    0.083    0.000 mathlib.py:23(contains_decimals)
  1000000    0.070    0.000    0.070    0.000 {method 'split' of 'str' objects}
     1025    0.001    0.000    0.002    0.000 codecs.py:319(decode)
     1025    0.001    0.000    0.001    0.000 {built-in method _codecs.utf_8_decode}
        1    0.000    0.000    1.142    1.142 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        2    0.000    0.000    0.000    0.000 mathlib.py:60(div)
        2    0.000    0.000    0.000    0.000 mathlib.py:73(mul)
        1    0.000    0.000    1.142    1.142 <string>:1(<module>)
        2    0.000    0.000    0.000    0.000 mathlib.py:50(sub)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 mathlib.py:83(root)
```

Figure 4: million inputs

```
2886.6321205876125
         225100071 function calls in 58.683 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1   35.666   35.666   58.683   58.683 profiling.py:30(profiler_function)
 50000001    9.195    0.000   13.455    0.000 mathlib.py:108(pow)
100000000    5.849    0.000    5.849    0.000 mathlib.py:40(add)
 24997738    4.260    0.000    4.260    0.000 mathlib.py:23(contains_decimals)
 50000000    3.622    0.000    3.622    0.000 {method 'split' of 'str' objects}
    51160    0.046    0.000    0.046    0.000 {built-in method _codecs.utf_8_decode}
    51160    0.045    0.000    0.091    0.000 codecs.py:319(decode)
        1    0.000    0.000   58.683   58.683 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000   58.683   58.683 <string>:1(<module>)
        1    0.000    0.000    0.000    0.000 mathlib.py:83(root)
        2    0.000    0.000    0.000    0.000 mathlib.py:60(div)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        2    0.000    0.000    0.000    0.000 mathlib.py:73(mul)
        2    0.000    0.000    0.000    0.000 mathlib.py:50(sub)
```

Figure 5: 50 million inputs

```
2886.6528452207226
         4502054724 function calls in 1183.268 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1  721.920  721.920 1183.268 1183.268 profiling.py:30(profiler_function)
1000000001  186.467    0.000  270.734    0.000 mathlib.py:108(pow)
2000000000  117.080    0.000  117.080    0.000 mathlib.py:40(add)
500008357   84.267    0.000   84.267    0.000 mathlib.py:23(contains_decimals)
1000000000   71.724    0.000   71.724    0.000 {method 'split' of 'str' objects}
  1023177    0.907    0.000    0.907    0.000 {built-in method _codecs.utf_8_decode}
  1023177    0.902    0.000    1.809    0.000 codecs.py:319(decode)
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000 1183.268 1183.268 {built-in method builtins.exec}
        2    0.000    0.000    0.000    0.000 mathlib.py:60(div)
        1    0.000    0.000 1183.268 1183.268 <string>:1(<module>)
        2    0.000    0.000    0.000    0.000 mathlib.py:50(sub)
        1    0.000    0.000    0.000    0.000 mathlib.py:83(root)
        2    0.000    0.000    0.000    0.000 mathlib.py:73(mul)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figure 6: 1 billion inputs