IVS - Profiling

Tým: Segmentation Fault

David Chocholatý (xchoch09@stud.fit.vutbr.cz)
Štěpán Bakaj (xbakaj00@stud.fit.vutbr.cz)
Adam Kaňkovský (xkanko00@stud.fit.vutbr.cz)
Martin Baláž (xbalaz15@stud.fit.vutbr.cz)

22. dubna 2021

1 Testovací prostředí

1.1 Hardware

CPU: Intel Core i7-3520M
RAM: 8GiB (2x4 GiB SODIMM DDR3 1600MHz)

1.2 Software

Operační systém: Ubuntu 20.04.1 LTS
Python: Python 3.8.5

2 Vstupní data

Soubor ./src/profiling_10.txt - testovací data obsahující 10 čísel
Soubor ./src/profiling_100.txt - testovací data obsahující 100 čísel
Soubor ./src/profiling_1000.txt - testovací data obsahující 1000 čísel
Soubor ./src/profiling_100000.txt - testovací data obsahující 100000 čísel

Pozn. : Testovací data obsahují celočíselné hodnoty v rozmezí intervalu od 1 do 1000 včetně. Jako relevantní vstupní data byl použit soubor profiling_100000.txt. Vstupní soubory s menším objemem dat při profilingu neposkytovaly dostatečné informace pro určení výsledků a závěru testování.

3 Spuštění

Pro profiling byl použit Python modul cProfile. Tento modul poskytuje deterministické profilování programů Pythonu. Poskytuje sadu statistik, která popisuje, jak často a jak dlouho se různé části programu provádějí.

Spuštění profilingu:
Pomocí Makefile: ***make profile***
Celý příkaz: ***python3 -m cProfile profiling.py < profiling_10.txt > ../profiling/vystup.out***

Pozn. : Při příkladu celého příkazu je použit vstupní soubor profiling_10.txt.

4 Výstup měření

Výstup měření pro vstupní soubor profiling_100000.txt:

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
 200000    0.015    0.000    0.015    0.000 calc.py:15(add)
      2    0.000    0.000    0.000    0.000 calc.py:23(subtract)
      1    0.000    0.000    0.000    0.000 calc.py:31(multiply)
      2    0.000    0.000    0.000    0.000 calc.py:40(divide)
      1    0.000    0.000    0.000    0.000 calc.py:68(square_root)
 100001    0.014    0.000    0.014    0.000 calc.py:98(power)
```

5 Výsledky

Na získání dostatečně podrobných detailů o tom, kde program tráví nejvíce času má největší vliv velikost vstupních dat. Výstupní data pro jednotlivé vstupní soubory jsou obsaženy v souboru vystup.out. Z výsledných dat lze odvodit, že nejvíce stráveného času v matematické knihovně je ve funkci *add* (součet dvou reálných čísel) a ve funkci *power* (druhá mocnina).

6 Závěr

Pro optimalizaci řešení funkcí *add* a *power* by bylo možné použití knihovny či knihoven, které mají optimálnější výpočet právě součtu a druhé mocniny oproti vestavěným operacím v jazyku Python.