ТЕХНОАТОМ

Углубленный Python

Лекция 6: профилирование, тестирование, логирование

Кандауров Геннадий

Не забудьте отметиться на занятии!

Лекция 6: повестка дня

ТЕХНОАТОМ

1. Мониторинг потребления ресурсов
2. Профилирование
3. Тестирование
4. Логирование

Мониторинг потребления ресурсов

top - консольная команда, которая выводит список работающих в системе процессов и информацию о них.

PID - идентификатор процесса
USER - пользователь, под которым запущен процесс
VIRT - объем виртуальной памяти, занимаемой процессом
RES - текущее использование RAM
%CPU - процент доступного времени процессора

atop - продвинутый интерактивный полноэкранный монитор производительности, написанный для Linux.

Чтение из файла:

```
atop -r /var/log/atop/atop_<date> [-b hh:mm]
```

Для перехода к следующему фрейму **t**.

iotop - утилита, выводящая данные по использованию жесткого диска.

Посмотреть активные процессы:

```
iotop -o
```

Собрать статистику за определенное время:

```
iotop -o -a
```

ТЕХНОАТОМ

iostat - утилита, предназначенная для мониторинга использования дисковых разделов.

```
iostat -d -t -p sda -x
```

`-c` вывести отчет по CPU
`-d` вывести отчет по использованию диска
`-t` интервал, за который усредняются значения
`-x` вывести расширенную статистику

Профилирование

сбор характеристик работы программы, таких как время выполнения отдельных фрагментов (обычно подпрограмм), число верно предсказанных условных переходов, число кэш-промахов и т. д.

Цель:

- найти узкие места в коде

Основные способы:

- CPU
- Память
- Частота/продолжительность вызовов функций

Методы:

- Статистический (сэмплирование)
- Детерминированный (инструментирование)

- ***cProfile*** - написанная на C, быстрая реализация профилировщика
- ***profile*** - нативная реализация профилировщика на чистом python, значительно медленнее

```
python -m cProfile -o output.txt ptest.py
```

```
import pstats
p = pstats.Stats('output.txt')
p.strip_dirs().sort_stats(-1).print_stats()
```

```
import cProfile, pstats, io

pr = cProfile.Profile()
pr.enable()
# ... do something ...
pr.disable()

s = io.StringIO()
sortby = 'cumulative'
ps = pstats.Stats(pr, stream=s).sort_stats(sortby)
ps.print_stats()
print(s.getvalue())
```

Лекция 6: cProfile

ТЕХНОАТОМ

```
         1567629 function calls (1166637 primitive calls) in 809.730 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.164    0.164  809.738  809.738 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/ioloop.py:568(start)
     4961  806.444    0.163  806.444    0.163 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/platform/kqueue.py:66(poll)
9982/8005    0.086    0.000    3.095    0.000 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/stack_context.py:269(wrapped)
     5657    0.011    0.000    2.767    0.000 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/ioloop.py:471(_run_callback)
6766/2479    0.083    0.000    1.869    0.001 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/gen.py:507(run)
     2445    0.009    0.000    1.775    0.001 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/gen.py:567(inner)
     2445    0.005    0.000    1.764    0.001 /Users/project/src/.env3/lib/python3.7/site-packages/tornado/gen.py:497(set_result)
      430    0.008    0.000    0.902    0.002 /Users/project/src/gekko/net/resolver.py:414(resolve)
       75    0.000    0.000    0.669    0.009 /Users/project/src/gekko/handlers2/executor.py:93(callback)
       75    0.000    0.000    0.669    0.009 /Users/project/src/gekko/handlers2/executor.py:72(_handler_callback)
       48    0.000    0.000    0.669    0.014 /Users/project/src/gekko/handlers2/executor.py:114(_done)
       72    0.000    0.000    0.612    0.009 /Users/project/src/gekko/location2.py:266(_call_location_method)
       60    0.000    0.000    0.610    0.010 /Users/project/src/gekko/location2.py:91(create_gen_tasks)
       63    0.000    0.000    0.609    0.010 /Users/project/src/gekkoapps/gosearch/locations/ajax_web.py:27(get)
        9    0.000    0.000    0.576    0.064 /Users/project/src/gekkoapps/common/locations/base.py:104(create_response)
        9    0.001    0.000    0.572    0.064 /Users/project/src/gekkoapps/common/locations/base.py:97(render_view)
        9    0.000    0.000    0.242    0.027 /Users/project/src/gekkoapps/common/locations/base.py:173(get_data_from_view)
        9    0.000    0.000    0.242    0.027 /Users/project/src/gekkoapps/common/views/base.py:136(get_data)
        9    0.000    0.000    0.239    0.027 /Users/project/src/gekkoapps/gosearch/v1/web/view/compat.py:14(create_location_data)
        9    0.000    0.000    0.238    0.026 /Users/project/src/gekkoapps/gosearch/v1/web/view/produce.py:518(get_data)
        9    0.000    0.000    0.220    0.024 /Users/project/src/gekkoapps/common/locations/base.py:183(render_json)
        9    0.000    0.000    0.220    0.024 /Users/project/src/gekko/template/helpers.py:148(do_json)
        9    0.013    0.001    0.220    0.024 /Users/project/src/.env3/lib/python3.7/site-packages/simplejson/encoder.py:371(encode)
     3626    0.030    0.000    0.214    0.000 /Users/project/src/gekko/net/resolver.py:185(resolve)
       27    0.000    0.000    0.209    0.008 /Users/project/src/gekkoapps/common/views/serp/v1/creator.py:23(create)
```

```
pip install memory_profiler

# run.py
from memory_profiler import profile

@profile
def some_func():
    lst1 = []
    lst2 = "1" * 100000

python -m memory_profiler run.py
```

```
python3 -m pdb script.py
```

или

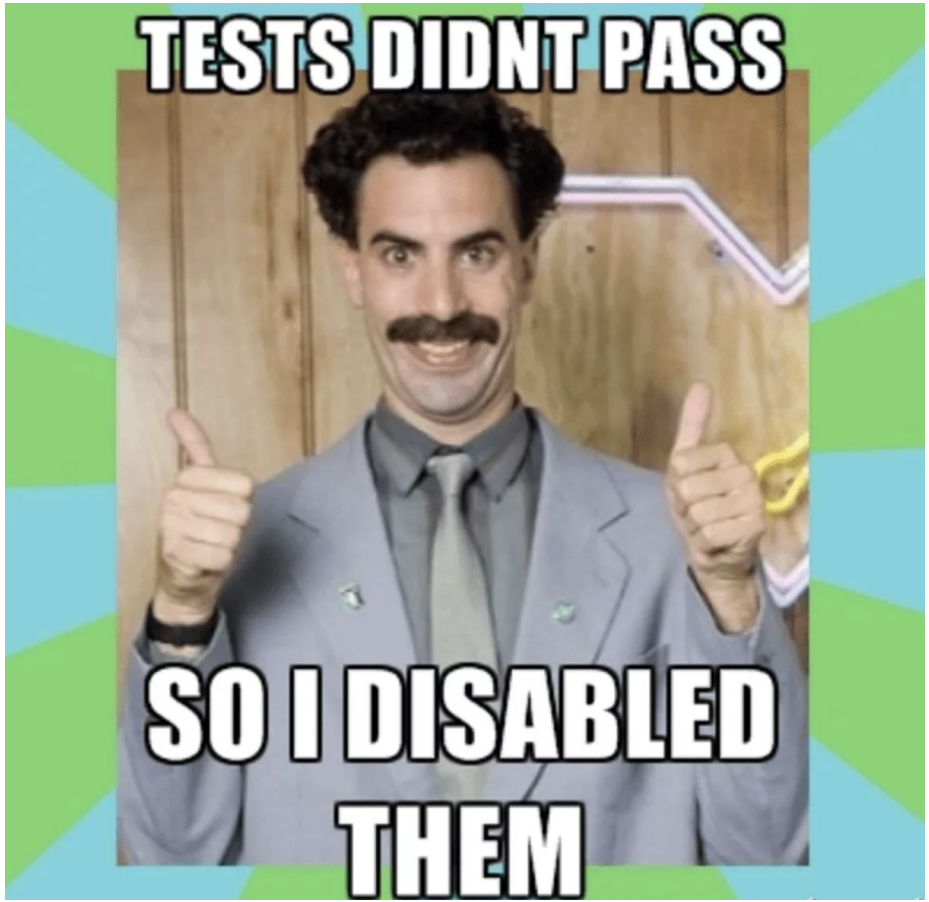
```
# script.py
def some_func():
    lst1 = []
    import pdb; pdb.set_trace()
    lst2 = "1" * 100000

python script.py
```

Тестирование

“Something that is untested is broken”

- unit (модульное)
- функциональное
- интеграционное/системное
- нагрузочное

- проверка правильности реализации
- проверка обработки граничных условий
- проверка обработки внештатных ситуаций
- минимизация последствий


TESTS DIDNT PASS
SO I DISABLED THEM

- doctest
- unittest
- pytest
- hypothesis

```
def do_action(action):
    """
    >>> do_action("move")
    1
    >>> do_action("sleep")
    2
    """
    return 1 if action == "move" else 2


if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

```
import unittest
class TestDoAction(unittest.TestCase):
    def test_valid(self):
        self.assertEqual(1, do_action('move'))
        self.assertEqual(2, do_action('sleep'))

    def test_unexpected(self):
        self.assertEqual(2, None)
        self.assertEqual(2, 999)

python -m unittest project
```

```
assertEqual(a, b)
assertNotEqual(a, b)
assertTrue(x)
assertFalse(x)
assertIs(a, b)
assertIsNot(a, b)
assertIsNone(x)
assertIn(a, b)
assertIsInstance(a, b)
assertLessEqual(a, b)
assertListEqual(a, b)
assertDictEqual(a, b)
assertRaises(exc, fun, *args, **kwargs)
```

mock подменяет объекты (функции, классы, атрибуты) на так называемые mock-объекты, заглушки.

```
class TestSome(unittest.TestCase):
    def test_some(self):
        with mock.patch('path.to.object') as m_obj:
            m_obj.side_effect = lambda x: x + 20
            do_some()
```

```python
# test_sample.py
def inc(x):
    return x + 1

def test_answer():
    assert inc(3) == 5
```

```
$ pytest
test_sample.py:6: AssertionError
FAILED test_sample.py::test_answer - assert 4 == 5
```

Логирование

- логгеры
- хендлеры
- фильтры
- форматтеры

```
import logging
logging.basicConfig(filename='example.log',
                    level=logging.DEBUG)
logging.debug('This message should go to the log file')
logging.info('So should this')
logging.warning('And this, too')
```

```
$ cat example.log
DEBUG:root:This message should go to the log file
INFO:root:So should this
WARNING:root:And this, too
```

Лекция 6: домашнее задание

ТЕХНОАТОМ

1. Написать функцию, принимающую список чисел и возвращающую список произведений всех чисел, кроме текущего:

   func([1, 2, 3, 4]) -> [24, 12, 8, 6];
2. Выполнить профилирование через cProfile;
3. Добавить тесты на функцию (обязательно unittest и моки, другие фреймворки по желанию);
4. Логировать ход выполнения функции и её частей.

ТЕХНОАТОМ

Спасибо за внимание!

Кандауров Геннадий

g.kandaurov@corp.mail.ru