Яндекс

Яндекс Облако

Py-spy и с чем его едят

Анкудинов Александр, разработчик Яндекс.Облака

Что такое профилирование?

Профилирование — сбор метрик про то, как выполняется наша программа

```
def update_metalist(self, metalist):
    # ...
    old_metalist = []                                       # 0.1 %
    for m in metalist:                                      # 0.2 %
        meta_name = m.name                                  # 0.1 %
        if meta_name in self_metas:                         # 0.1 %
            old_metalist.append(self_metas[meta_name])      # 0.2 %
            self_metas[meta_name][id2] = m                  # 0.2 %
        else:
            self_metas[meta_name] = {id2 : m}               # 0.2 %

    old_request = \
      self._build_request(self_imid, id2, old_metalist) # 98.1 %

    if request != old_request:                              # 0.5 %
        requests.append(request)                            # 0.3 %
```

Инструментирование

```
def func_a():
    start = time.perf_counter()
    # полезные действия
    end = time.perf_counter()
    stats['func_a'].add(end - start)

# а ещё есть sys.setprofile()
```

Сэмплирование

```
class SamplingProfiler(threading.Thread):
    def run(self):
        self.result = Counter()
        main_thread_ident = threading.main_thread().ident

        for i in range(200):
            main_frame = sys._current_frames()[main_thread_ident]

            filename = main_frame.f_code.co_filename
            lineno = main_frame.f_lineno

            self.result[(filename, lineno)] += 1
            time.sleep(0.005)


profile = SamplingProfiler()
profile.start()
# code to profile
profile.join()
print(profile.result.most_common())
```

Что лучше?

Инструментирование:

- Полное покрытие
- Огромный оверхед

Сэмплирование:

- Маленький, контролируемый оверхед
- Контролируемая точность
- Недетерминированность
- Маленькое покрытие

Py-spy

Ставится меньше чем за минуту:

```
pip install py-spy
cargo install py-spy
```

- Сэмплирующий
- Запускается в отдельном процессе
- Написан на Rust
- Почти без оверхеда
- Можно использовать на проде

Простое демо: конкатенация строк

```
import io
def func1_simple(n):
    s = ''
    for i in range(n):
        s += str(i)
    return s

def func2_stringio(n):
    with io.StringIO() as s:
        for i in range(n):
            s.write(str(i))
        return s.getvalue()

def func3_join(n):
    return ''.join(str(i) for i in range(n))
```

Тёмное прошлое vs светлое настоящее

Pyflame

```
root@xelez-xubuntu:~# pyflame --pid 31589
demo1.py:<module>:28;demo1.py:run_all:26;demo1.py:func3_join:19
;demo1.py:<genexpr>:19 23
demo1.py:<module>:28;demo1.py:run_all:25;demo1.py:func2_stringi
o:16 1
demo1.py:<module>:28;demo1.py:run_all:25;demo1.py:func2_stringi
o:15 26
demo1.py:<module>:28;demo1.py:run_all:26;demo1.py:func3_join:19
 6
demo1.py:<module>:28;demo1.py:run_all:24;demo1.py:func1_simple:
9 37
demo1.py:<module>:28;demo1.py:run_all:24;demo1.py:func1_simple:
8 6
root@xelez-xubuntu:~#
```

Py-spy

```
Collecting samples from 'python demo1.py' (python v3.6.8)
Total Samples 700
GIL: 100.00%, Active: 100.00%, Threads: 1

  %Own   %Total  OwnTime  TotalTime  Function (filename:line)
 38.00%  38.00%   2.04s     2.04s   func1_simple (demo1.py:9)
 31.00%  31.00%   2.29s     2.29s   func2_stringio (demo1.py:1
 21.00%  21.00%   1.91s     1.91s   <genexpr> (demo1.py:19)
  6.00%   6.00%   0.270s    0.270s  func1_simple (demo1.py:8)
  3.00%  24.00%   0.310s     2.22s   func3_join (demo1.py:19)
  1.00%   1.00%   0.170s    0.170s  func2_stringio (demo1.py:1
  0.00%  32.00%   0.000s     2.47s   run_all (demo1.py:25)
  0.00% 100.00%   0.000s     7.00s   <module> (demo1.py:28)
  0.00%   0.00%   0.010s    0.010s  func2_stringio (demo1.py:1
  0.00%  44.00%   0.000s     2.31s   run_all (demo1.py:24)
  0.00%  24.00%   0.000s     2.22s   run_all (demo1.py:26)

Press Control-C to quit, or ? for help.
```

(Не)сложная реальность

Вводные:

- contrail-api
- Python 2.7
- REST-API на gevent
- Нагрузочное тестирование
- cpu 100%

vnc_cassandra.py до

```
class ObjectCacheManager(object):
    #...

    def read(self, obj_uuids, req_fields, include_backrefs_children):
        # find which keys are a hit, find which hit keys are not stale
        # return hit entries and miss+stale uuids.
        cached_uuid_set = set(self._cache.keys())   # <------ 1669 -------
        request_uuid_set = set(obj_uuids)
        hit_uuid_set = set(obj_uuids) & cached_uuid_set
        miss_uuid_set = set(obj_uuids) - cached_uuid_set
        #...
```

vnc_cassandra.py после

```
class ObjectCacheManager(object):
    #...

    def read(self, obj_uuids, req_fields, include_backrefs_children):
        # find which keys are a hit, find which hit keys are not stale
        # return hit entries and miss+stale uuids.

        #cached_uuid_set = set(self._cache.keys())   # <------ 1669 -------
        #request_uuid_set = set(obj_uuids)
        #hit_uuid_set = set(obj_uuids) & cached_uuid_set
        #miss_uuid_set = set(obj_uuids) - cached_uuid_set

        hit_uuids = []
        miss_uuids = []
        for obj_uuid in obj_uuids:
            if obj_uuid in self._cache:
                hit_uuids.append(obj_uuid)
            else:
                miss_uuids.append(obj_uuid)
        #...
```

Результат

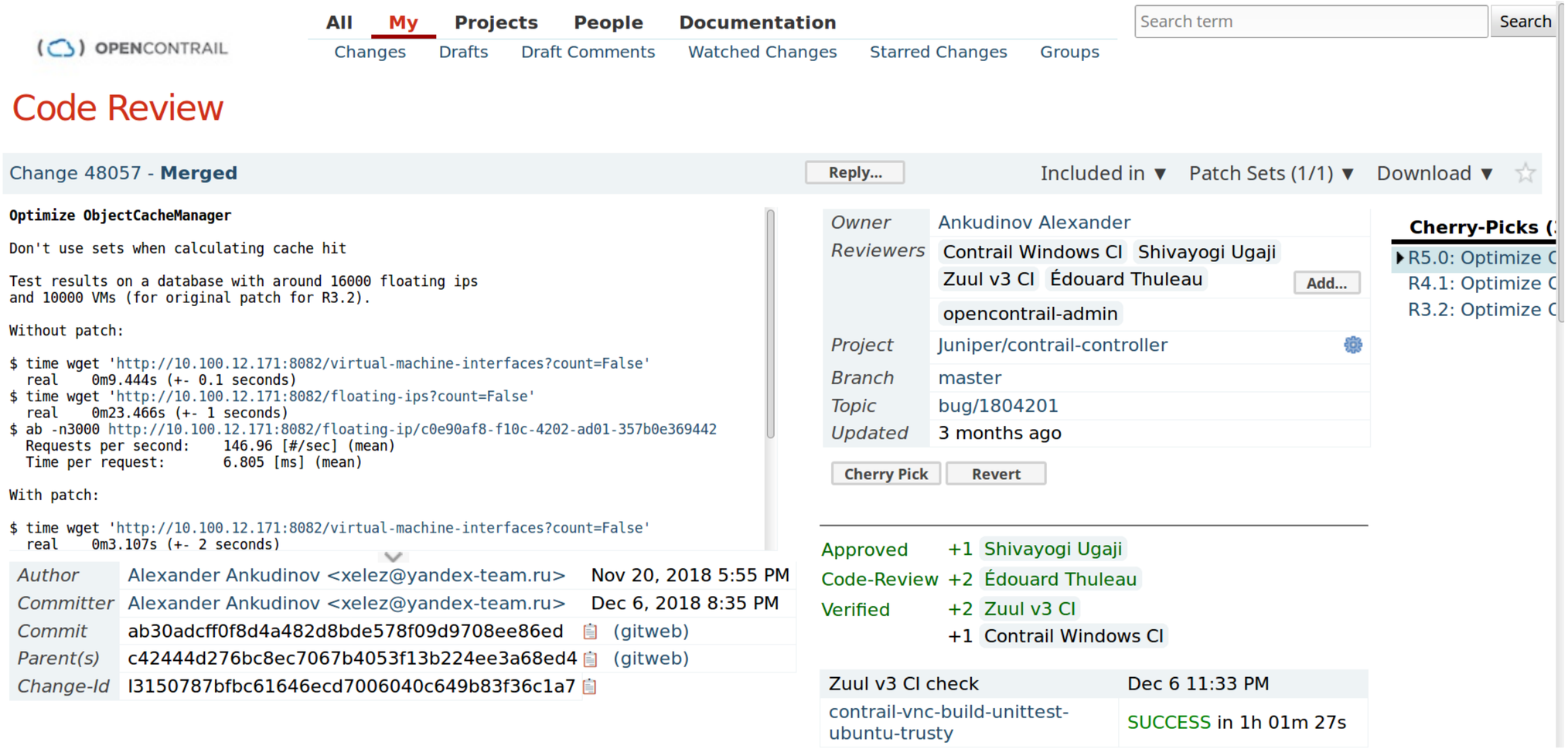
Ускорили в 2 раза!

До:
$ ab -n3000 http://10.100.12.171:8082/floating-ip/c0e90af8-f10c-4202-ad01-357b0e369442
  Requests per second: **146.96** [#/sec] (mean)
  Time per request: **6.805** [ms] (mean)

После:
$ ab -n3000 http://10.100.12.171:8082/floating-ip/c0e90af8-f10c-4202-ad01-357b0e369442
  Requests per second: **271.36** [#/sec] (mean)
  Time per request: **3.685** [ms] (mean)

Upstream!

https://review.opencontrail.org/#/c/47922/



Заключение

- Профилируйте больше
- Оптимизируйте код
- Коммитьте в open-source
- Делайте мир лучше

Яндекс Облако

Спасибо

Анкудинов Александр,

Разработчик Яндекс.Облака

xelez@yandex-team.ru

@xelez