Python Profiling and Performance Tuning

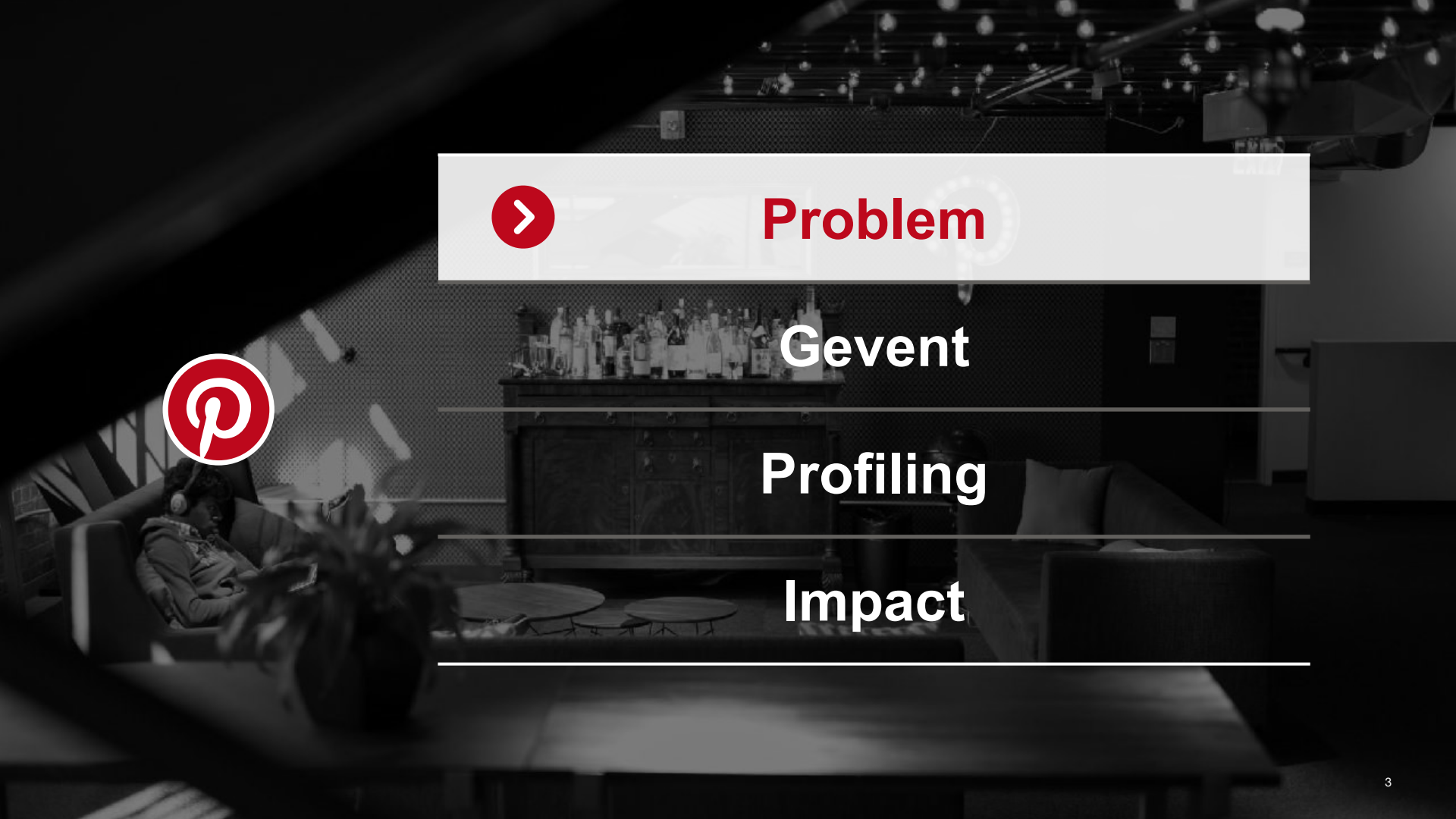
Python at Pinterest

Joe Gordon

August 14st, 2016

About Me

- **Site Reliability Engineer (SRE) @ Pinterest**
  - Availability
  - Performance
  - Efficiency
  - Developer Velocity
- **SRE embedded in Search Team**



Problem

Gevent

Profiling

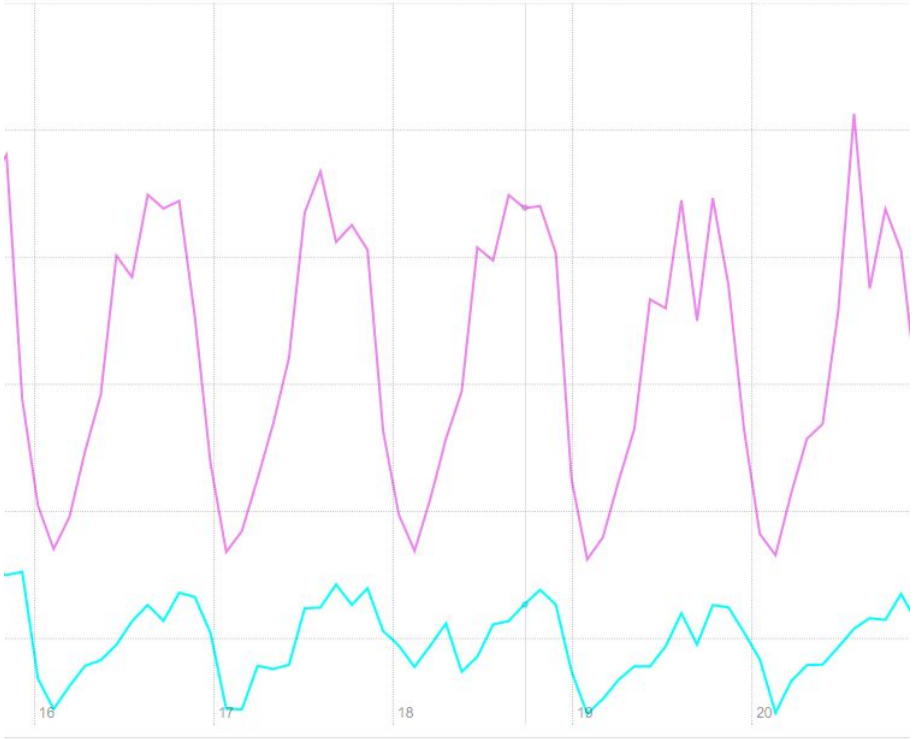
Impact

Pin Search P90 Latency

Q4 2015

- **Latency**
  - Varnish: 2200ms
  - Search system: 640ms

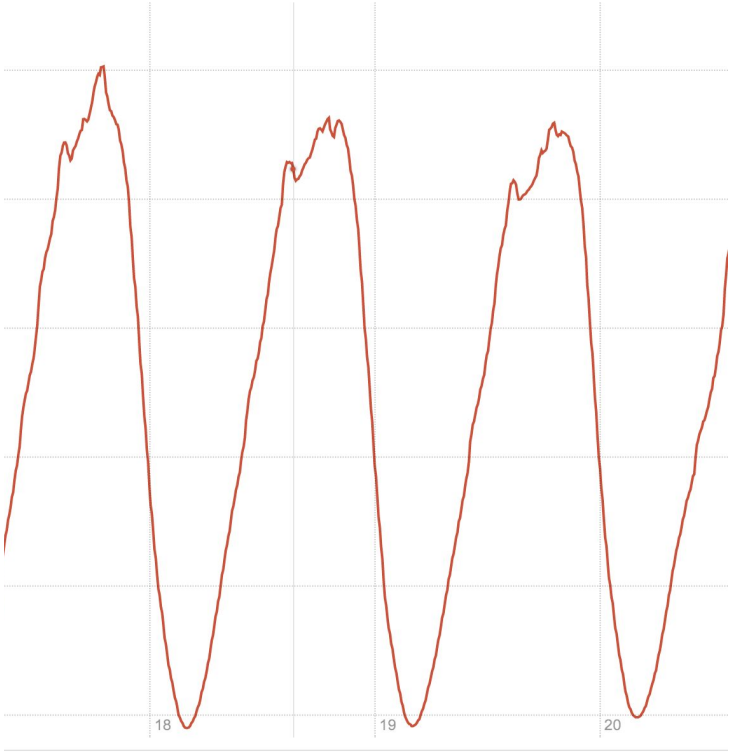
- **70% of latency outside of Search system**
- **Height of Varnish latency >> height of asterix latency**



GET API Latency

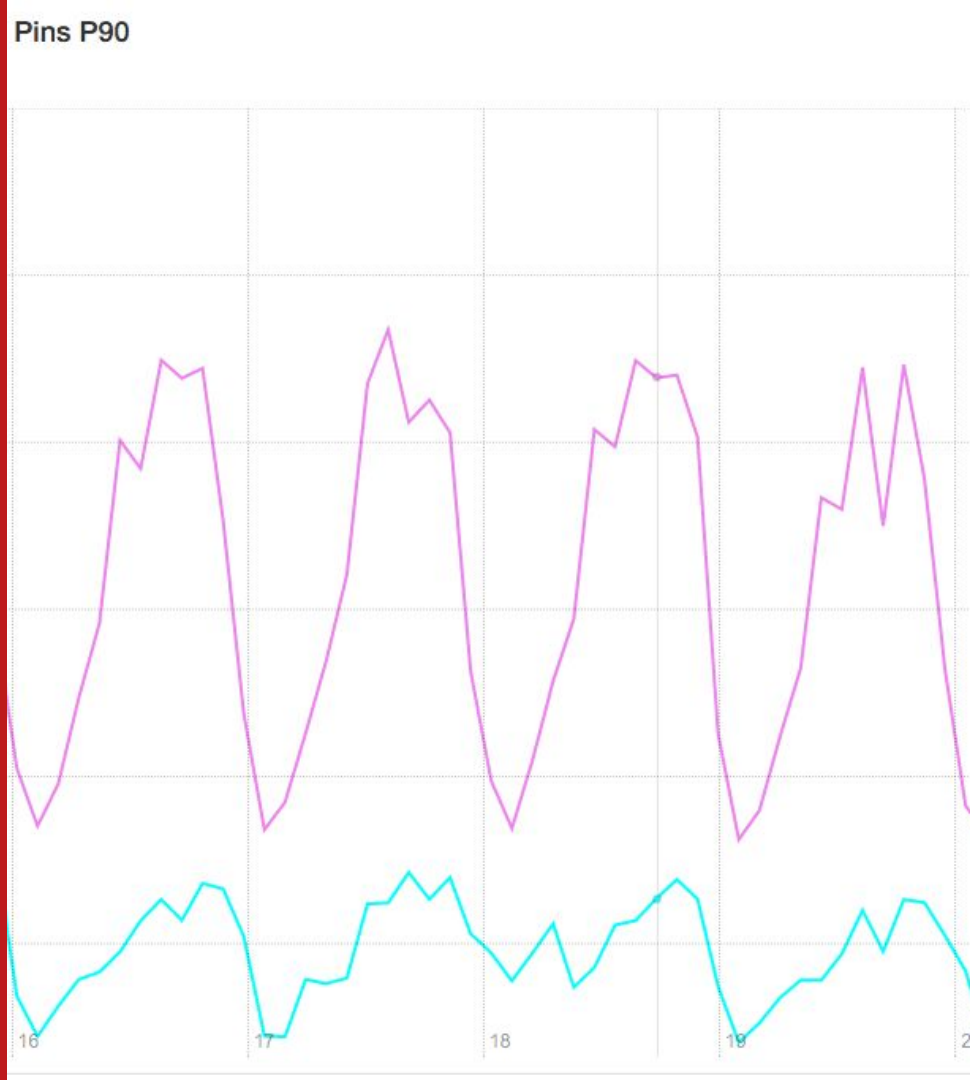
Q4 2015

- **Same pattern**
- **Height: 50%**



Why?

- *Latency from frontend*
- *Height of latency*



Problem

Gevent

Profiling

Impact

What is gevent?

gevent is a coroutine -based Python networking library that uses greenlet to provide a high-level synchronous API on top of the libev event loop.

Features include:

- **Fast event loop** based on libev (epoll on Linux, kqueue on FreeBSD).
- **Lightweight execution units** based on greenlet.

What is gevent?

gevent is a coroutine -based Python networking library that uses greenlet to provide a high-level synchronous API on top of the libev event loop.

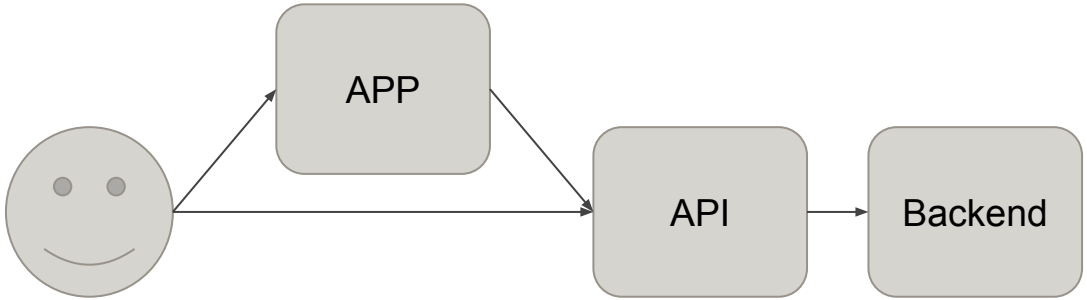
Example

The following example shows how to run tasks concurrently.

```
>>> import gevent
>>> from gevent import socket
>>> urls = ['www.google.com', 'www.example.com', 'www.python.org']
>>> jobs = [gevent.spawn(socket.gethostbyname, url) for url in urls]
>>> gevent.joinall(jobs, timeout=2)
>>> [job.value for job in jobs]
['74.125.79.106', '208.77.188.166', '82.94.164.162']
```

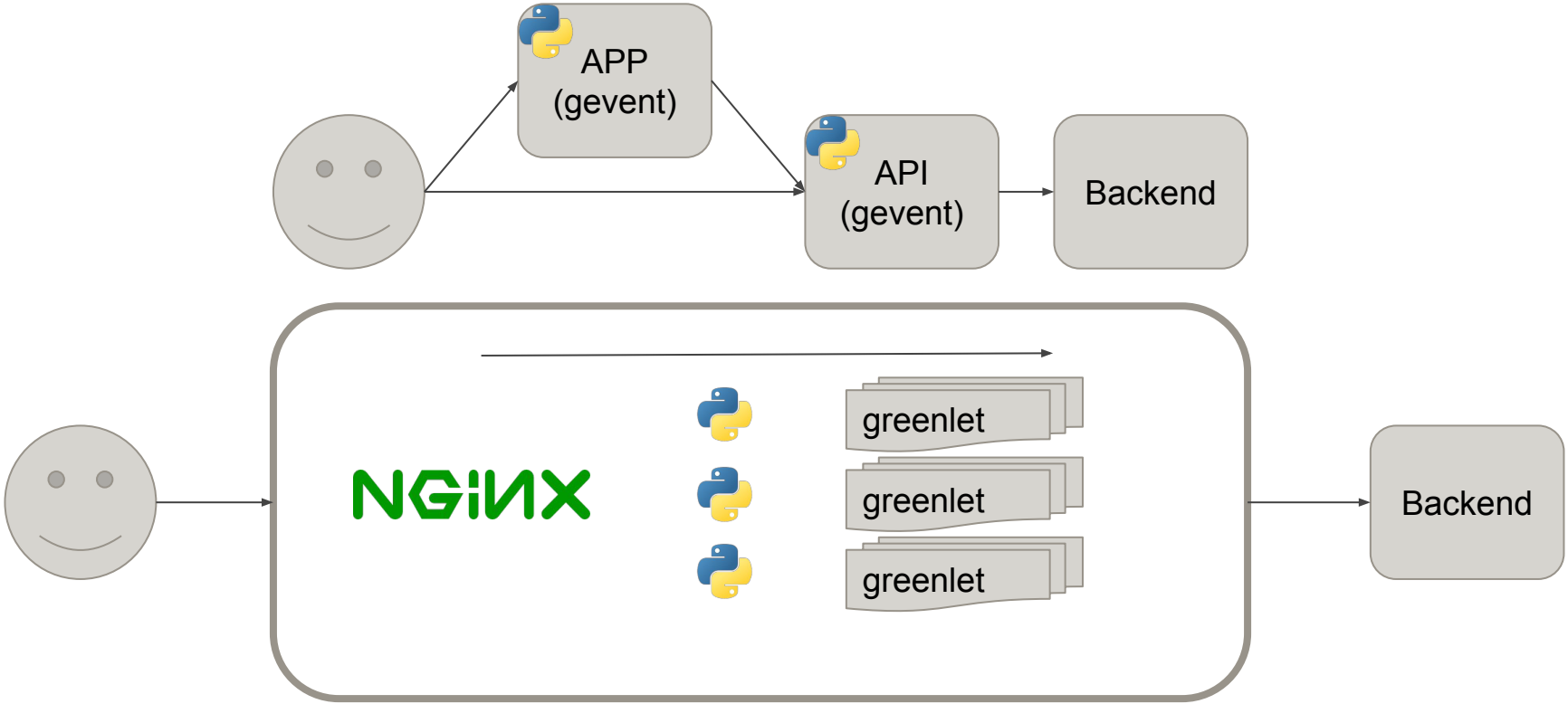
Software Stack

gevent



Software Stack

gevent



gevent

Cooperative multitasking

Cooperative multitasking

The greenlets all run in the same OS thread and are scheduled cooperatively. This means that until a particular greenlet gives up control, (by calling a blocking function that will switch to the Hub), other greenlets won't get a chance to run. This is typically not an issue for an I/O bound app, but one should be aware of this when doing something CPU intensive, or when calling blocking I/O functions that bypass the libev event loop.

gevent

Cooperative multitasking

Cooperative multitasking

The greenlets all run in the same OS thread and are scheduled cooperatively. This means that until a particular greenlet gives up control, (by calling a blocking function that will switch to the Hub), other greenlets won't get a chance to run. This is typically not an issue for an I/O bound app, but one should be aware of this when doing something CPU intensive, or when calling blocking I/O functions that bypass the libev event loop.

What happens if we mix CPU intensive and network bound workloads together?

Gevent Experiment

setup

Server:

```
62 def workload(request_id):
63     gevent_yield(request_id)
64     block(request_id)
65     network_call(request_id)
66     log.debug("%s: done" % request_id)
67
95 def main():
96     print('Serving on 8088...')
97     WSGIServer(('', 8088), application).serve_forever()
```

Concurrent Client:

```
 9 def main(concurrent, count):
10     session = grequests.Session()
11     urls = ['http://localhost:8088']*count
12     rs = (grequests.get(u, session=session) for u in urls)
13     responses = grequests.map(rs, size=concurrent)
14     for response in responses:
15         response.close()
16     print "%s calls, %s at a time" % (count, concurrent)
17     r = requests.get('http://localhost:8088/stats')
18     return r.json()
19
```

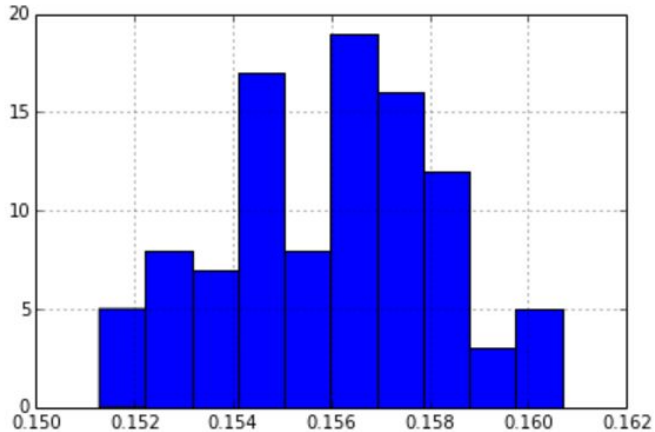
blocking time: 50ms

non blocking time: 100ms

Gevent Experiment

results

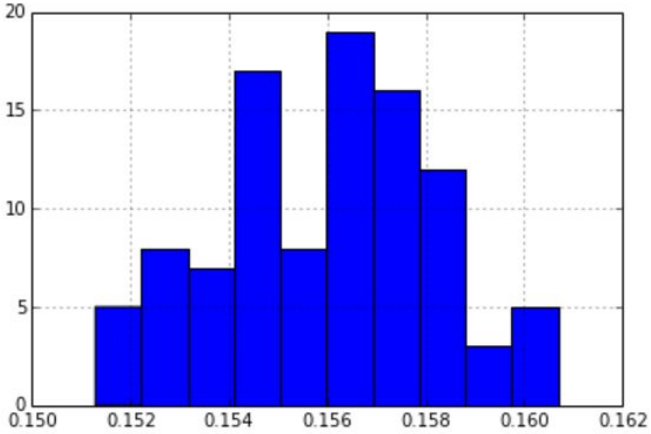
```
100 calls, 1 at a time
count    100.000000
mean       0.155956
std        0.002204
min        0.151283
25%        0.154440
50%        0.156152
75%        0.157455
max        0.160684
dtype: float64
```



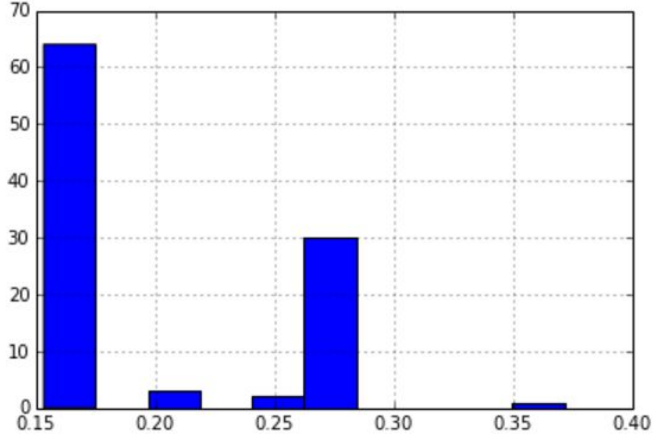
Gevent Experiment

results

```
100 calls, 1 at a time
count    100.000000
mean       0.155956
std        0.002204
min        0.151283
25%        0.154440
50%        0.156152
75%        0.157455
max        0.160684
dtype: float64
```



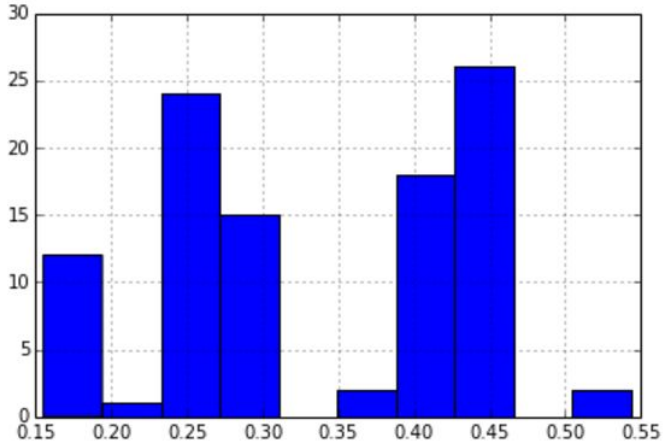

```
100 calls, 5 at a time
count    100.000000
mean       0.199159
std        0.052894
min        0.153300
25%        0.161063
50%        0.164095
75%        0.265418
max        0.371659
dtype: float64
```



Gevent Experiment

results

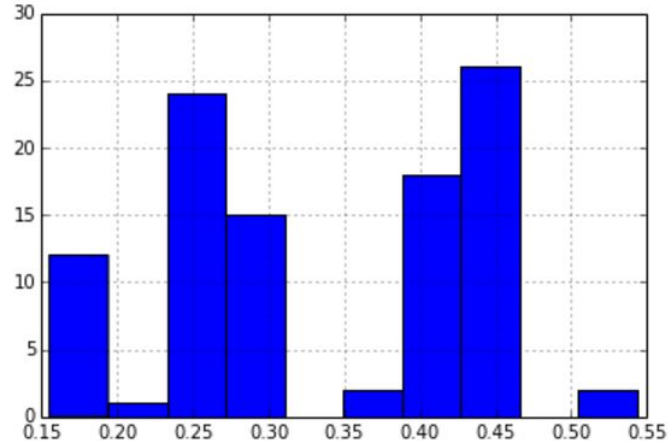
```
100 calls, 10 at a time
count    100.000000
mean       0.333240
std        0.102039
min        0.154991
25%        0.265487
50%        0.274264
75%        0.427456
max        0.543295
dtype: float64
```



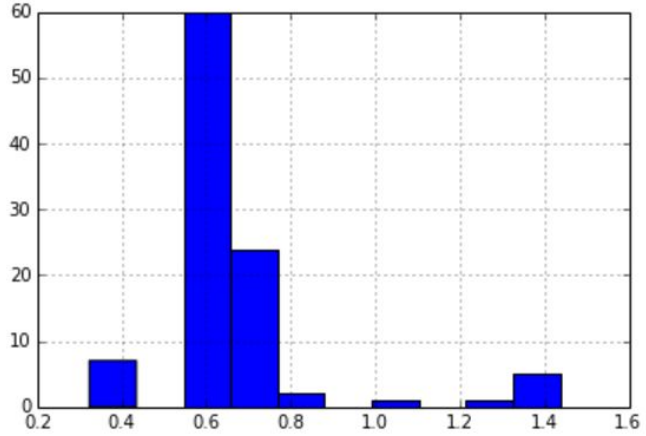
Gevent Experiment

results

```
100 calls, 10 at a time
count    100.000000
mean       0.333240
std        0.102039
min        0.154991
25%        0.265487
50%        0.274264
75%        0.427456
max        0.543295
dtype: float64
```




```
100 calls, 15 at a time
count    100.000000
mean       0.687463
std        0.194353
min        0.323455
25%        0.635788
50%        0.642643
75%        0.693844
max        1.437414
dtype: float64
```



Gevent Experiment: Server Log

5 concurrent requests

```
2016-02-19 15:50:40,334 - 92: enter gevent_yield
2016-02-19 15:50:40,334 - 91: exit gevent_yield
2016-02-19 15:50:40,334 - 91: enter block
2016-02-19 15:50:40,389 - 91: exit block
2016-02-19 15:50:40,389 - 91: enter network_call
2016-02-19 15:50:40,389 - 92: exit gevent_yield
2016-02-19 15:50:40,389 - 92: enter block
2016-02-19 15:50:40,443 - 92: exit block
2016-02-19 15:50:40,444 - 92: enter network_call
2016-02-19 15:50:40,444 - 89: exit network_call
2016-02-19 15:50:40,444 - 89: done
127.0.0.1 - - [2016-02-19 15:50:40] "GET / HTTP/1.1" 200
119 0.275503
2016-02-19 15:50:40,444 - 90: exit network_call
2016-02-19 15:50:40,444 - 90: done
127.0.0.1 - - [2016-02-19 15:50:40] "GET / HTTP/1.1" 200
119 0.165371
2016-02-19 15:50:40,445 - 93: enter gevent_yield
2016-02-19 15:50:40,445 - 93: exit gevent_yield
2016-02-19 15:50:40,445 - 93: enter block
2016-02-19 15:50:40,496 - 93: exit block
2016-02-19 15:50:40,496 - 93: enter network_call
2016-02-19 15:50:40,496 - 91: exit network_call
2016-02-19 15:50:40,496 - 91: done
127.0.0.1 - - [2016-02-19 15:50:40] "GET / HTTP/1.1" 200
119 0.163295
```

```
2016-02-19 15:50:40,497 - 94: enter gevent_yield
2016-02-19 15:50:40,497 - 95: enter gevent_yield
2016-02-19 15:50:40,497 - 94: exit gevent_yield
2016-02-19 15:50:40,497 - 94: enter block
2016-02-19 15:50:40,549 - 94: exit block
2016-02-19 15:50:40,549 - 94: enter network_call
2016-02-19 15:50:40,550 - 95: exit gevent_yield
2016-02-19 15:50:40,550 - 95: enter block
2016-02-19 15:50:40,603 - 95: exit block
2016-02-19 15:50:40,603 - 95: enter network_call
2016-02-19 15:50:40,603 - 92: exit network_call
2016-02-19 15:50:40,603 - 92: done
```

- Extra 50ms block in first `gevent_yield`
- `network_call` takes 50 extra ms (159 total)
- 100ms extra from blocks from other requests
- ~250ms (269ms) instead of expected 150ms

Gevent Experiment: Server Log

5 concurrent requests

```
2016-02-19 15:50:40,334 - 92: enter gevent_yield
2016-02-19 15:50:40,334 - 91: exit gevent_yield
2016-02-19 15:50:40,334 - 91: enter block
2016-02-19 15:50:40,389 - 91: exit block
2016-02-19 15:50:40,389 - 91: enter network_call
2016-02-19 15:50:40,389 - 92: exit gevent_yield
2016-02-19 15:50:40,389 - 92: enter block
2016-02-19 15:50:40,443 - 92: exit block
2016-02-19 15:50:40,444 - 92: enter network_call
2016-02-19 15:50:40,444 - 89: exit network_call
2016-02-19 15:50:40,444 - 89: done
127.0.0.1 - - [2016-02-19 15:50:40] "GET / HTTP/1.1" 200
119 0.275503
2016-02-19 15:50:40,444 - 90: exit network_call
2016-02-19 15:50:40,444 - 90: done
127.0.0.1 - - [2016-02-19 15:50:40] "GET / HTTP/1.1" 200
119 0.165371
2016-02-19 15:50:40,445 - 93: enter gevent_yield
2016-02-19 15:50:40,445 - 93: exit gevent_yield
2016-02-19 15:50:40,445 - 93: enter block
2016-02-19 15:50:40,496 - 93: exit block
2016-02-19 15:50:40,496 - 93: enter network_call
2016-02-19 15:50:40,496 - 91: exit network_call
2016-02-19 15:50:40,496 - 91: done
127.0.0.1 - - [2016-02-19 15:50:40] "GET / HTTP/1.1" 200
119 0.163295
```

```
2016-02-19 15:50:40,497 - 94: enter gevent_yield
2016-02-19 15:50:40,497 - 95: enter gevent_yield
2016-02-19 15:50:40,497 - 94: exit gevent_yield
2016-02-19 15:50:40,497 - 94: enter block
2016-02-19 15:50:40,549 - 94: exit block
2016-02-19 15:50:40,549 - 94: enter network_call
2016-02-19 15:50:40,550 - 95: exit gevent_yield
2016-02-19 15:50:40,550 - 95: enter block
2016-02-19 15:50:40,603 - 95: exit block
2016-02-19 15:50:40,603 - 95: enter network_call
2016-02-19 15:50:40,603 - 92: exit network_call
2016-02-19 15:50:40,603 - 92: done
```

- Extra 50ms block in first `gevent_yield`
- `network_call` takes 50 extra ms (159 total)
- 100ms extra from blocks from other requests
- ~250ms (269ms) instead of expected 150ms

Gevent

Impact

- **Performance degrades with increased**
  - CPU work
  - Requests per process
- **Autoscale frontend at 40% CPU usage**

- **Reduce latency by doing fewer requests per process**
  - config changes

Gevent

Goal

- **Frontend clusters are some of the most expensive**
- **High latency**

Increase throughput (RPS / host) and decrease latency by decreasing CPU usage

- **Goal: 33% efficiency gain**

Problem

Gevent

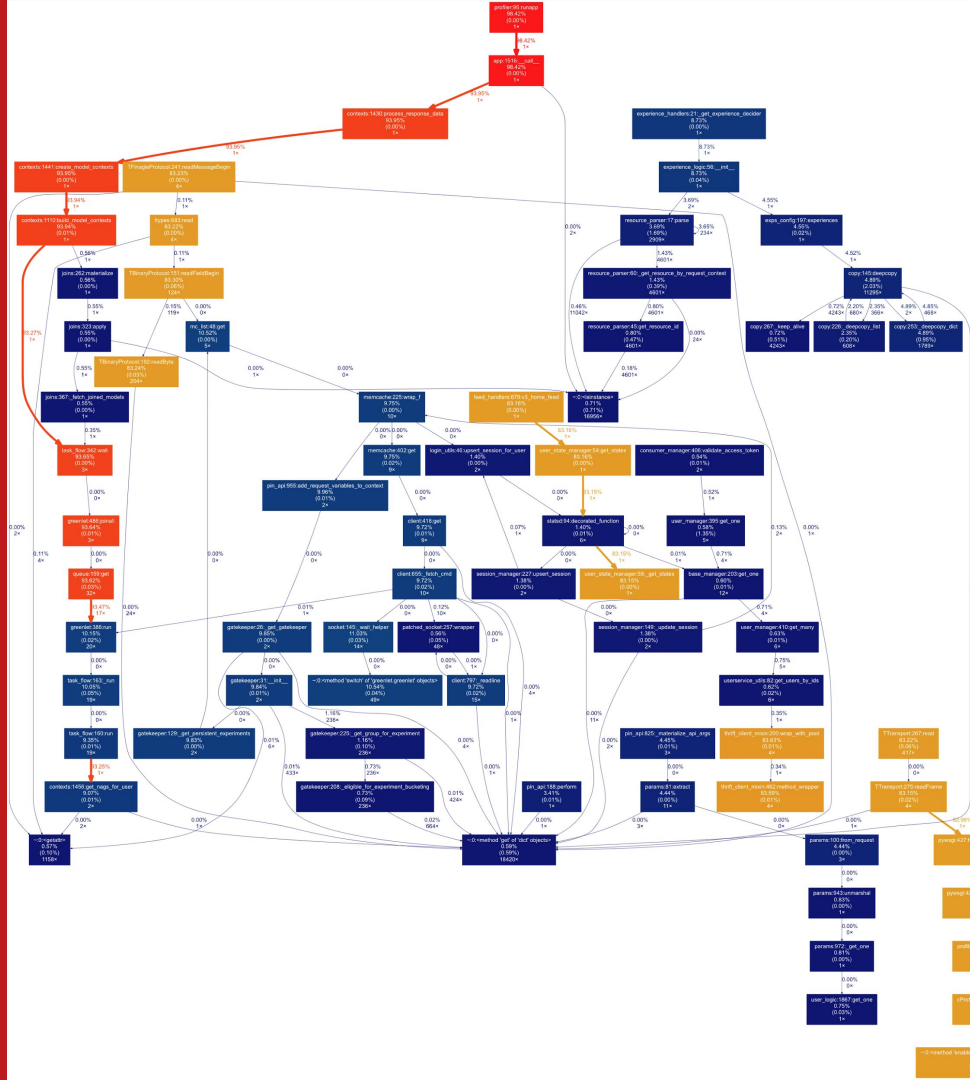
Profiling

Impact

Python Profilers

1. **API cProfiler middleware**
2. **Line profiler**
3. **Sampling profiler and flamegraphs**

API Profiler

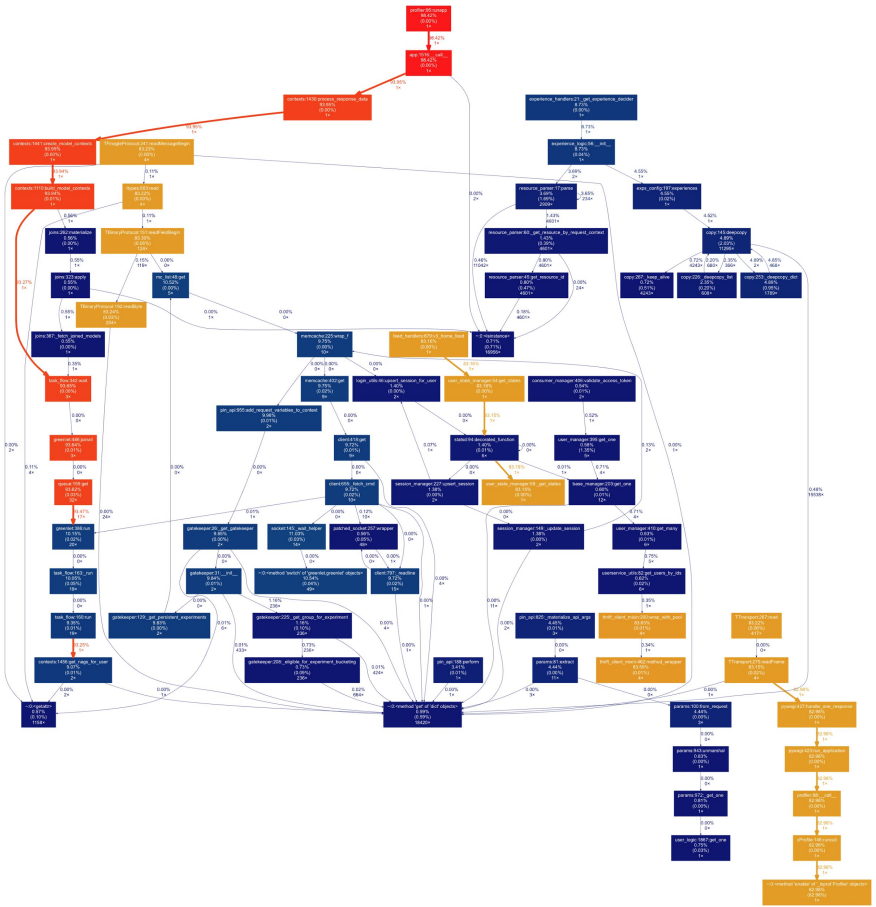


API Profiler

- **Profiles function calls**
- **Save each request to a separate file**
- **Large overhead**
  - can’t be used in production

```
diff --git a/api/start.py b/api/start.py
index e910aa9..b0f58f0 100755
--- a/api/start.py
+++ b/api/start.py
@@ -5,6 +5,9 @@ import sys
 sys.path.insert(0, '../')
 sys.path.insert(1, '../common')

+from werkzeug.contrib.profiler import ProfilerMiddleware
+
+
 import common.utils.patch_for_gevent
 common.utils.patch_for_gevent.patch()

@@ -45,7 +48,7 @@ def main(unused_argv):
     if settings.API_WSGI_SERVER_LOGS_ENABLED is not None:
         log = 'default' if settings.API_WSGI_SERVER_LOGS_ENABLED else None

-    pywsgi.WSGIServer(("0.0.0.0", port), app, log=log).serve_forever()
+    pywsgi.WSGIServer(("0.0.0.0", port), ProfilerMiddleware(app, profile_dir='/home/jogo/tmp'), log=log).serve_forever()
```

GET /v3/feeds/home



python -mpstats GET.v3.search.pins.000908ms.1452734245.prof

python `pstats` module

Module for profiler data analysis

pstats module

stats

```
GET.v3.search.pins.000908ms.1452734245.prof% stats 10
Wed Jan 13 17:26:41 2016    GET.v3.search.pins.000908ms.1452734245.prof

         34464 function calls (34150 primitive calls) in 0.956 seconds

   Ordered by: internal time
   List reduced from 1247 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.809    0.809    0.809    0.809 {method 'enable' of '_lsprof.Profiler' objects}
       12    0.056    0.005    0.057    0.005 /home/jogo/code/.venv/local/lib/python2.7/site-packages/gevent/socket.py:153(wait_read)
        3    0.048    0.016    0.860    0.287 /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/transport/TTransport.py:275(readFrame)
      189    0.001    0.000    0.001    0.000 ../core/gatekeeper/utils.py:6(pick_key_by_mod)
      872    0.001    0.000    0.001    0.000 {method 'search' of '_sre.SRE_Pattern' objects}
      414    0.001    0.000    0.012    0.000 ../core/gatekeeper/experiment.py:206(get_group)
      189    0.001    0.000    0.002    0.000 ../core/gatekeeper/experiment.py:291(get_mod_group)
      414    0.001    0.000    0.013    0.000 ../core/gatekeeper/user_experiments.py:80(_convert_experiment_object_to_dict)
      201    0.001    0.000    0.001    0.000 /usr/lib/python2.7/threading.py:120(acquire)
     2195    0.001    0.000    0.001    0.000 {method 'get' of 'dict' objects}


GET.v3.search.pins.000908ms.1452734245.prof% stats _convert_experiment_object_to_dict
Wed Jan 13 17:26:41 2016    GET.v3.search.pins.000908ms.1452734245.prof

         34464 function calls (34150 primitive calls) in 0.956 seconds

   Ordered by: internal time
   List reduced from 1247 to 1 due to restriction <'_convert_experiment_object_to_dict'>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      414    0.001    0.000    0.013    0.000 ../core/gatekeeper/user_experiments.py:80(_convert_experiment_object_to_dict)
```

`pstats` module

callees & callers

```
GET.v3.search.pins.000908ms.1452734245.prof% callees _convert_experiment_object_to_dict
   Random listing order was used
   List reduced from 1247 to 1 due to restriction <'_convert_experiment_object_to_dict'>

Function                                                                          called...
                                                                                      ncalls  tottime  cumtime
../core/gatekeeper/user_experiments.py:80(_convert_experiment_object_to_dict)  ->     414    0.001    0.012  ../core/gatekeeper/experiment.py:206(get_group)
                                                                                       414    0.000    0.000  {isinstance}


GET.v3.search.pins.000908ms.1452734245.prof% callers _convert_experiment_object_to_dict
   Random listing order was used
   List reduced from 1247 to 1 due to restriction <'_convert_experiment_object_to_dict'>

Function                                                                          was called by...
                                                                                      ncalls  tottime  cumtime
../core/gatekeeper/user_experiments.py:80(_convert_experiment_object_to_dict)  <-     414    0.001    0.013  ../core/gatekeeper/user_experiments.py:53(get_limited_user_experiments)
```

pstats module

Find CPU bound function calls

```
jogo@lappy:~/scratchpad/profiling/Jan-13-2015$ python walk.py GET.v3.search.pins.000908ms.1452734245.prof
cumtime 0.8088 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/gevent/pywsgi.py:427(handle_one_response)
cumtime 0.8088 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/gevent/pywsgi.py:423(run_application)
cumtime 0.8088 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/werkzeug/contrib/profiler.py:88(__call__)
cumtime 0.8088 - /usr/lib/python2.7/cProfile.py:146(runcall)
cumtime 0.8088 - ~:0(<method 'enable' of '_lsprof.Profiler' objects>)
cumtime 0.0040 - ../core/experiments/gatekeeper.py:225(_get_group_for_experiment)
cumtime 0.0038 - ../core/models/user_model.py:306(is_employee)
cumtime 0.0036 - ../common/utils/employee_utils.py:121(is_employee_user)
cumtime 0.0032 - ../common/utils/employee_utils.py:95(_is_employee_account)
cumtime 0.0026 - ../core/experiments/gatekeeper.py:208(_eligible_for_experiment_bucketing)
cumtime 0.0024 - ~:0(<getattr>)
cumtime 0.0023 - ../common/utils/decorators.py:438(__call__)
cumtime 0.0017 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/werkzeug/local.py:340(__getattr__)
cumtime 0.0016 - ../api/serializers/contexts.py:1341(from_request)
cumtime 0.0015 - ../common/utils/url_utils.py:434(__init__)
cumtime 0.0015 - ../core/gatekeeper/utils.py:6(pick_key_by_mod)
cumtime 0.0014 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/ua_parser/user_agent_parser.py:175(Parse)
cumtime 0.0013 - ../core/utils/ratelimit.py:103(keys_to_check)
cumtime 0.0012 - ../core/utils/ratelimit.py:94(generate_key)
cumtime 0.0011 - ../core/gatekeeper/filters/filters.py:549(test)
cumtime 0.0011 - ../config/experiments.py:2558(<lambda>)
cumtime 0.0010 - /usr/lib/python2.7/threading.py:120(acquire)
cumtime 0.0010 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/protocol/TBase.py:56(write)
cumtime 0.0010 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/protocol/TProtocol.py:373(writeStruct)
cumtime 0.0010 - ../common/utils/version.py:71(__init__)
cumtime 0.0010 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/protocol/TProtocol.py:393(writeFieldByTType)
```

```
"""
Idea: Automatically figure out which functions take a long time and don't use the network


Due to not collecting full stacktraces, we can only find out callees and go
that way, so would work if everything is only called once.
"""

import pstats
import sys

p = pstats.Stats(sys.argv[1])
p.calc_callees()


def get_children(func, visited=None):
    """return set of all functions called by func (recursive)"""
    if visited is None:
        visited = set()
    visited.add(func)
    callees = p.all_callees[func]
    for callee in callees:
        if callee not in visited:
            visited.update(get_children(callee, visited))
    return visited


def uses_network(children):
    """Checks a list if children to see if the network is called"""
    return "socket.py" in str(children)


def get_network_free_functions():
    functions = []
    for func in p.all_callees:
        children = get_children(func)
        if not uses_network(children):
            functions.append(func)

    # sort by cumtime
    functions = sorted(functions, key=lambda x: p.stats[x][3], reverse=True)
    return functions


def format_function_name(func):
    (path, line, name) = func
    return ("%s:%s(%s)" % (path, line, name))


# print top 50 slowest non network bound functions
for func in get_network_free_functions()[:50]:
    ccalls, ncalls, totttime, cumtime, callers = p.stats[func]
    print "cumtime %0.4f - %s" % (cumtime, format_function_name(func))
```

`pstats` module

Find CPU bound function calls

```
jogo@lappy:~/scratchpad/profiling/Jan-13-2015$ python walk.py GET.v3.search.pins.000908ms.1452734245.prof
cumtime 0.8088 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/gevent/pywsgi.py:427(handle_one_response)
cumtime 0.8088 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/gevent/pywsgi.py:423(run_application)
cumtime 0.8088 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/werkzeug/contrib/profiler.py:88(__call__)
cumtime 0.8088 - /usr/lib/python2.7/cProfile.py:146(runcall)
cumtime 0.8088 - ~:0(<method 'enable' of '_lsprof.Profiler' objects>)
cumtime 0.0040 - ../core/experiments/gatekeeper.py:225(_get_group_for_experiment)
cumtime 0.0038 - ../core/models/user_model.py:306(is_employee)
cumtime 0.0036 - ../common/utils/employee_utils.py:121(is_employee_user)
cumtime 0.0032 - ../common/utils/employee_utils.py:95(_is_employee_account)
cumtime 0.0026 - ../core/experiments/gatekeeper.py:208(_eligible_for_experiment_bucketing)
cumtime 0.0024 - ~:0(<getattr>)
cumtime 0.0023 - ../common/utils/decorators.py:438(__call__)
cumtime 0.0017 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/werkzeug/local.py:340(__getattr__)
cumtime 0.0016 - ../api/serializers/contexts.py:1341(from_request)
cumtime 0.0015 - ../common/utils/url_utils.py:434(__init__)
cumtime 0.0015 - ../core/gatekeeper/utils.py:6(pick_key_by_mod)
cumtime 0.0014 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/ua_parser/user_agent_parser.py:175(Parse)
cumtime 0.0013 - ../core/utils/ratelimit.py:103(keys_to_check)
cumtime 0.0012 - ../core/utils/ratelimit.py:94(generate_key)
cumtime 0.0011 - ../core/gatekeeper/filters/filters.py:549(test)
cumtime 0.0011 - ../config/experiments.py:2558(<lambda>)
cumtime 0.0010 - /usr/lib/python2.7/threading.py:120(acquire)
cumtime 0.0010 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/protocol/TBase.py:56(write)
cumtime 0.0010 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/protocol/TProtocol.py:373(writeStruct)
cumtime 0.0010 - ../common/utils/version.py:71(__init__)
cumtime 0.0010 - /home/jogo/code/.venv/local/lib/python2.7/site-packages/thrift/protocol/TProtocol.py:393(writeFieldByTType)
```

```
"""
Idea: Automatically figure out which functions take a long time and don't use the network


Due to not collecting full stacktraces, we can only find out callees and go
that way, so would work if everything is only called once.
"""

import pstats
import sys

p = pstats.Stats(sys.argv[1])
p.calc_callees()


def get_children(func, visited=None):
    """return set of all functions called by func (recursive)"""
    if visited is None:
        visited = set()
    visited.add(func)
    callees = p.all_callees[func]
    for callee in callees:
        if callee not in visited:
            visited.update(get_children(callee, visited))
    return visited


def uses_network(children):
    """Checks a list if children to see if the network is called"""
    return "socket.py" in str(children)


def get_network_free_functions():
    functions = []
    for func in p.all_callees:
        children = get_children(func)
        if not uses_network(children):
            functions.append(func)

    # sort by cumtime
    functions = sorted(functions, key=lambda x: p.stats[x][3], reverse=True)
    return functions


def format_function_name(func):
    (path, line, name) = func
    return ("%s:%s(%s)" % (path, line, name))


# print top 50 slowest non network bound functions
for func in get_network_free_functions()[:50]:
    ccalls, ncalls, totttime, cumtime, callers = p.stats[func]
    print "cumtime %0.4f - %s" % (cumtime, format_function_name(func))
```

Line Profiler

```
Function: get_url_structures at line 744

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   744                                           @profile
   745                                           def get_url_structures(url):
   746                                               """Parse and return domain/reverse-domain/host/l1 from a url.
   747                                               Example:
   748                                                   "http://image.pinterest.com/ben/?a=1":
   749                                                       {"domain": "pinterest.com",
   750                                                        "reverse-domain": "com.pinterest/",   # always have trailing
   751                                                        "host": "image.pinterst.com",
   752                                                        "reverse-host": "com.pinterest.image/",
   753                                                        "l1": "image.pinterst.com/ben/"  # always have trailing /
   754                                                        "reverse-l1": "com.pinterst.image/ben/"  # always have traili
   755                                                       }
   756                                               Returns:
   757                                                   A dict with fields above. It is not necessary to contain all field
   758                                                   For example, "http://pinterest.com" doesn't contain l1 path.
   759                                               """
   760     20001       17592      0.9      1.6      url_structure = {}
   761     20001       16756      0.8      1.5      if url:
   762     20001      140586      7.0     12.8          url_info = urlparse.urlparse(url)
   763     20001       26608      1.3      2.4          if not url_info.scheme:
   764                                                       url = "http://%s" % url
   765                                                       url_info = urlparse.urlparse(url)
   766     20001       73095      3.7      6.7          if url_info.hostname:
   767                                                       # host is case insensitive
   768                                                       # special treatment for "www."
   769     20001       73464      3.7      6.7              normalized_netloc = netloc = url_info.hostname.lower().rstrip(
   770     20001       23757      1.2      2.2              if netloc.startswith("www."):
   771     10000        9668      1.0      0.9                  netloc = netloc[4:]
   772     20001       18024      0.9      1.6              url_structure[URL_HOST_STR] = netloc
   773     20001       24235      1.2      2.2              host_segs = netloc.split('.')
   774     20001       18156      0.9      1.7              host_segs.reverse()
   775     20001       26056      1.3      2.4              url_structure[URL_REVERSE_HOST_STR] = '.'.join(host_segs) + "/
   776     20001       26807      1.3      2.4              path = url_info.path.lstrip("/")
   777     20001       16678      0.8      1.5              if path:
   778     20001       22443      1.1      2.0                  path_segs = path.split("/")
   779     20001       16154      0.8      1.5                  url_structure[URL_L1_STR] = "%s/%s/" % (
   780     20001       26940      1.3      2.5                      netloc, path_segs[0])
   781     20001       16161      0.8      1.5                  url_structure[URL_REVERSE_L1_STR] = "%s%s/" % (
   782     20001       23272      1.2      2.1                      url_structure[URL_REVERSE_HOST_STR], path_segs[0])
   783
   784     20001      337160     16.9     30.7              domain = psl.get_public_suffix(normalized_netloc.lower())
   785
   786     20001       19660      1.0      1.8              if domain:
   787     20001       18633      0.9      1.7                  url_structure[URL_DOMAIN_STR] = domain
   788     20001       25030      1.3      2.3                  domain_segs = domain.split('.')
   789     20001       18582      0.9      1.7                  domain_segs.reverse()
   790     20001       16944      0.8      1.5                  url_structure[URL_REVERSE_DOMAIN_STR] = '.'.join(
   791     20001       28092      1.4      2.6                      domain_segs) + "/"
   792     20001       16154      0.8      1.5      return url_structure
```

Line Profiler

https://github.com/rkern/line_profiler

```
Line Contents
==============
@profile
def _is_employee_test_user(email):
    """Return True if account has a pinterest email of a current employee

    This function does not check if the email is confirmed.
    """
    pattern = r"(^([a-zA-Z0-9_.-]+)([+][a-zA-Z0-9_.+-]*)?@([a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+)$)"
    if isinstance(email, str) or isinstance(email, unicode):
        match = re.match(pattern, email)
        if match:
            _, user, _, domain = match.groups()
            if domain == "pinterest.com":
                return LdapList().is_ldap(user)
    return False
```

Line Profiler

https://github.com/rkern/line_profiler

```
@profile
def slow_function(a, b, c):
    …
$ kernprof -l script_to_profile.py

$ python -m line_profiler script_to_profile.py.lprof
```

```
Function: _is_employee_test_user at line 80

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    80                                           @profile
    81                                           def _is_employee_test_user(email):
    82                                               """Return True if account has a pinterest email of a current employee
    83
    84                                               This function does not check if the email is confirmed.
    85                                               """
    86       200          128      0.6      1.5      pattern = r"(^([a-zA-Z0-9_.-]+)([+][a-zA-Z0-9_.+-]*)?@([a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+)$)"
    87       200          133      0.7      1.5      if isinstance(email, str) or isinstance(email, unicode):
    88       200         4808     24.0     54.7          match = re.match(pattern, email)
    89       200          123      0.6      1.4          if match:
    90       200          194      1.0      2.2              _, user, _, domain = match.groups()
    91       200          122      0.6      1.4              if domain == "pinterest.com":
    92       100         3220     32.2     36.6                  return LdapList().is_ldap(user)
    93       100           58      0.6      0.7      return False

Total time: 0.001775 s
File: common/utils/employee_utils.py
Function: _is_employee_account at line 95
```

Line Profiler

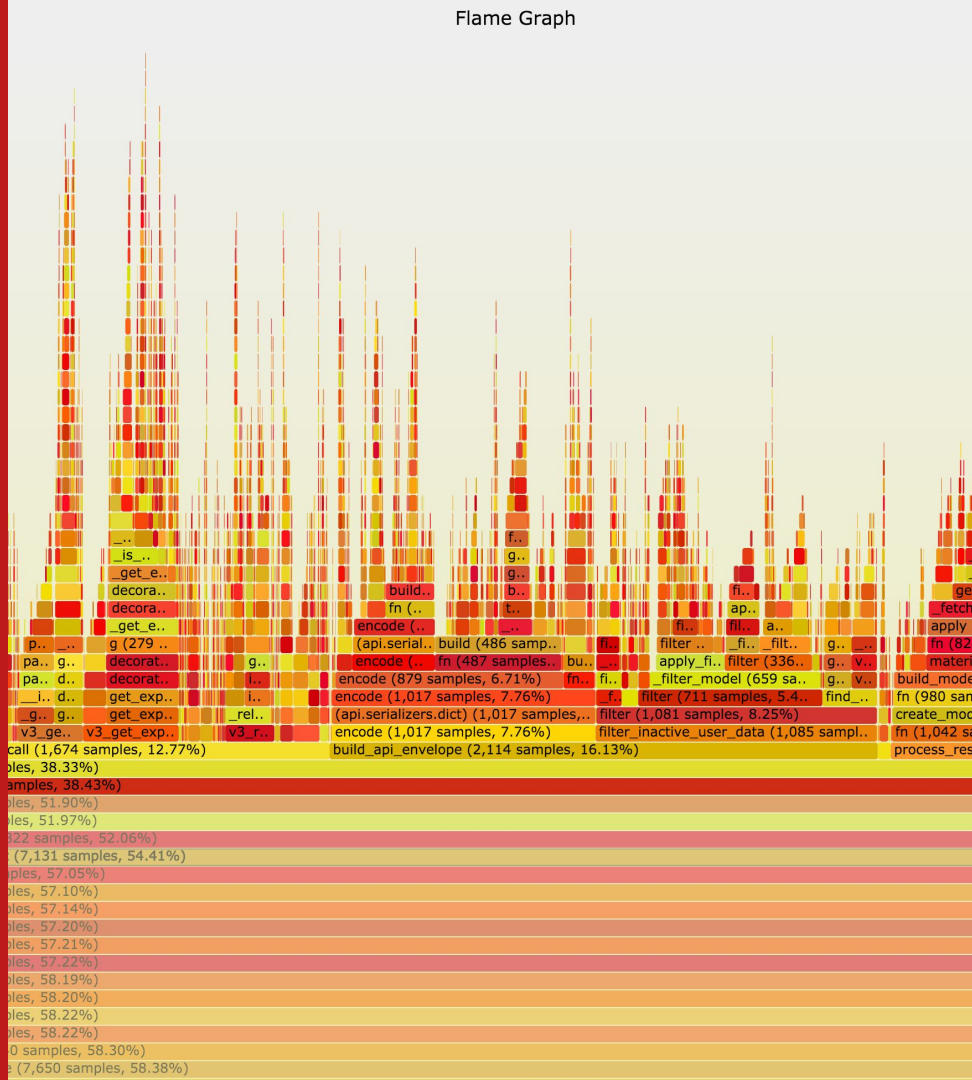
https://github.com/rkern/line_profiler

```
@profile
def slow_function(a, b, c):
    …
$ kernprof -l script_to_profile.py

$ python -m line_profiler script_to_profile.py.lprof
```

```
Function: _is_employee_test_user at line 80

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    80                                           @profile
    81                                           def _is_employee_test_user(email):
    82                                               """Return True if account has a pinterest email of a current employee
    83
    84                                               This function does not check if the email is confirmed.
    85                                               """
    86       200          128      0.6      1.5      pattern = r"(^([a-zA-Z0-9_.-]+)([+][a-zA-Z0-9_.+-]*)?@([a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+)$)"
    87       200          133      0.7      1.5      if isinstance(email, str) or isinstance(email, unicode):
    88       200         4808     24.0     54.7          match = re.match(pattern, email)
    89       200          123      0.6      1.4          if match:
    90       200          194      1.0      2.2              _, user, _, domain = match.groups()
    91       200          122      0.6      1.4              if domain == "pinterest.com":
    92       100         3220     32.2     36.6                  return LdapList().is_ldap(user)
    93       100           58      0.6      0.7      return False

Total time: 0.001775 s
File: common/utils/employee_utils.py
Function: _is_employee_account at line 95
```

Sampling Profiler

Flame Graphs

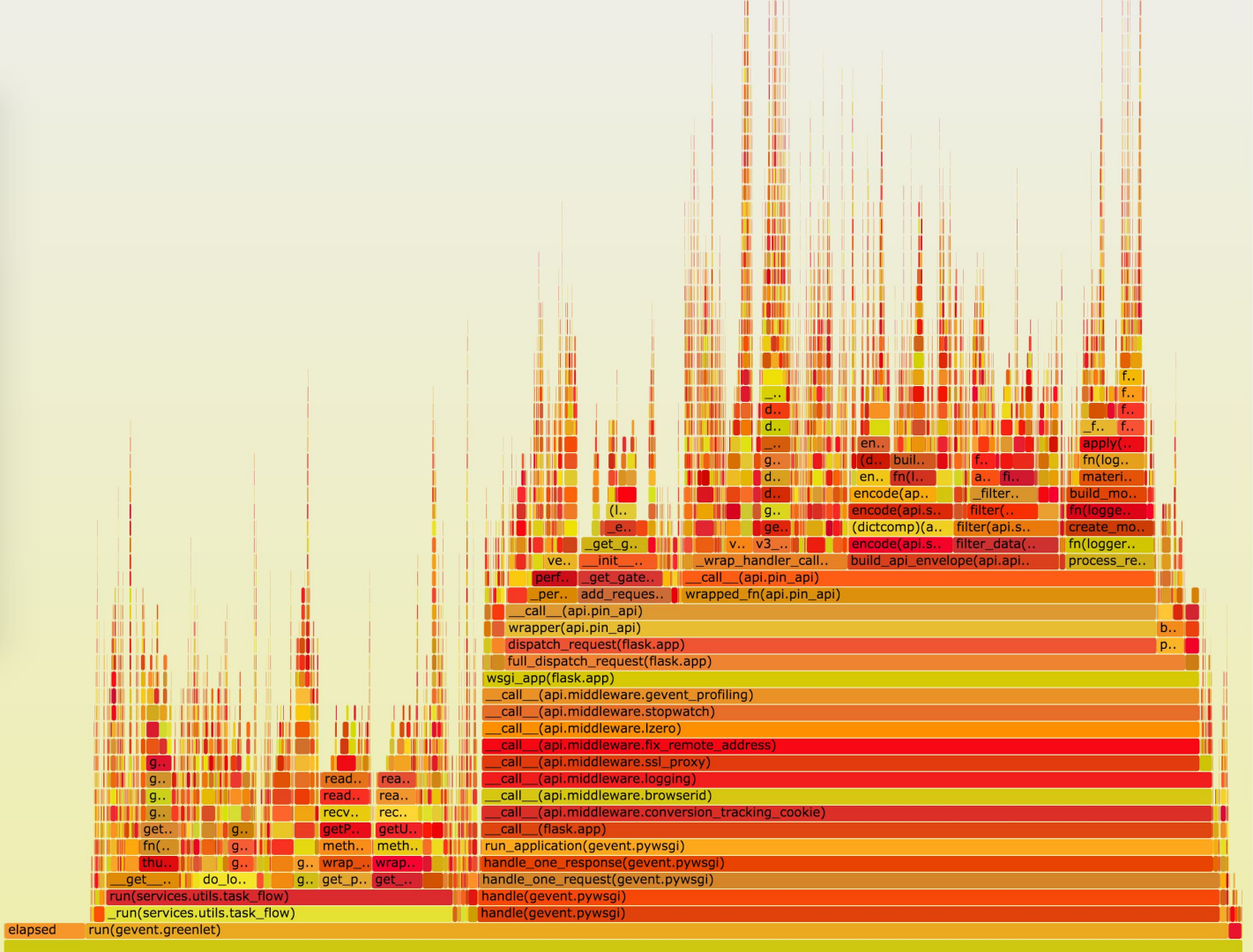


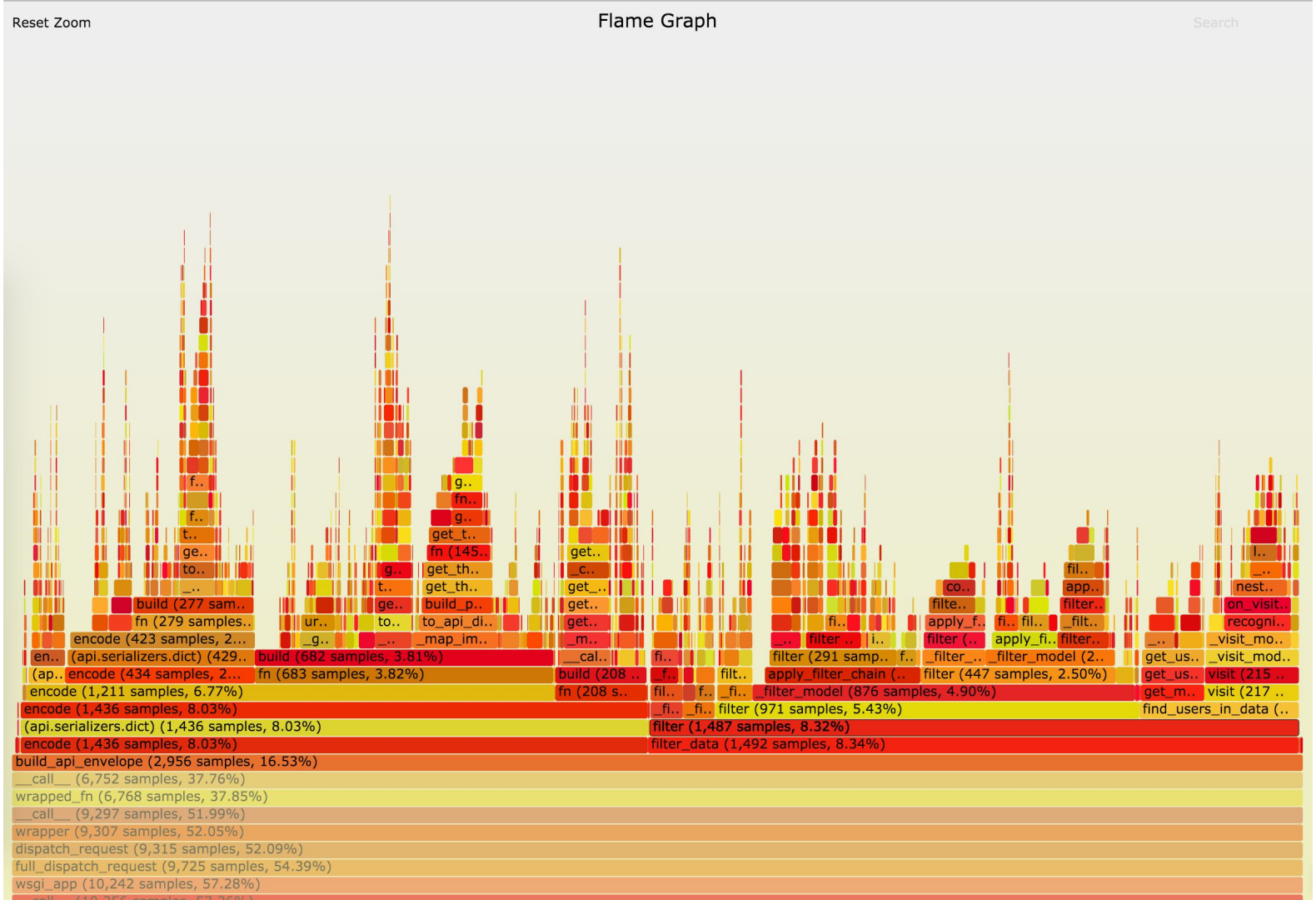
Sampling Profiler

https://github.com/nylas/nylas-perftools

- **Low overhead**
  - can be used in production
- **Records stack every 5ms**
- **Emits data over `http://localhost:16384`**
- **Format**
  - Stack: Count
- **Fuzzy**
- **Flame graphs**

```python
class Sampler(object):
    """
    A simple stack sampler for low-overhead CPU profiling: samples the call
    stack every `interval` seconds and keeps track of counts by frame. Because
    this uses signals, it only works on the main thread.
    """
    def __init__(self, interval=0.005):
        self.interval = interval
        self._started = None
        self._stack_counts = collections.defaultdict(int)

    def start(self):
        self._started = time.time()
        try:
            signal.signal(signal.SIGVTALRM, self._sample)
        except ValueError:
            raise ValueError('Can only sample on the main thread')

        signal.setitimer(signal.ITIMER_VIRTUAL, self.interval, 0)

    def _sample(self, signum, frame):
        stack = []
        while frame is not None:
            stack.append(self._format_frame(frame))
            frame = frame.f_back

        stack = ';'.join(reversed(stack))
        self._stack_counts[stack] += 1
        signal.setitimer(signal.ITIMER_VIRTUAL, self.interval, 0)
```


elapsed
run(gevent.greenlet)
_run(services.utils.task_flow)
run(services.utils.task_flow)
__get__..
do_lo..
thu..
fn(..
get..
g..
recv..
read..
meth..
wrap_..
get_p..
rec..
rea..
wrap..
get_..
getP..
getU..
handle(gevent.pywsgi)
handle_one_request(gevent.pywsgi)
handle_one_response(gevent.pywsgi)
run_application(gevent.pywsgi)
__call__(flask.app)
__call__(api.middleware.conversion_tracking_cookie)
__call__(api.middleware.logging)
__call__(api.middleware.ssl_proxy)
__call__(api.middleware.fix_remote_address)
__call__(api.middleware.lzero)
__call__(api.middleware.stopwatch)
__call__(api.middleware.gevent_profiling)
wsgi_app(flask.app)
full_dispatch_request(flask.app)
dispatch_request(flask.app)
wrapper(api.pin_api)
__call__(api.pin_api)
wrapped_fn(api.pin_api)
add_reques..
_get_gate..
__init__..
ve..
perf..
_per..
_get_g..
_wrap_handler_call..
__call__(api.pin_api)
build_api_envelope(api.api..
encode(api.s..
filter_data(..
v3_..
v..
ge..
(dictcomp)(a..
filter(api.s..
encode(ap..
filter(..
_filter..
en..
(d..
buil..
fn(l..
a..
fi..
f..
_f..
apply(..
fn(log..
materi..
build_mo..
fn(logge..
create_mo..
fn(logger..
process_re..
b..
p..

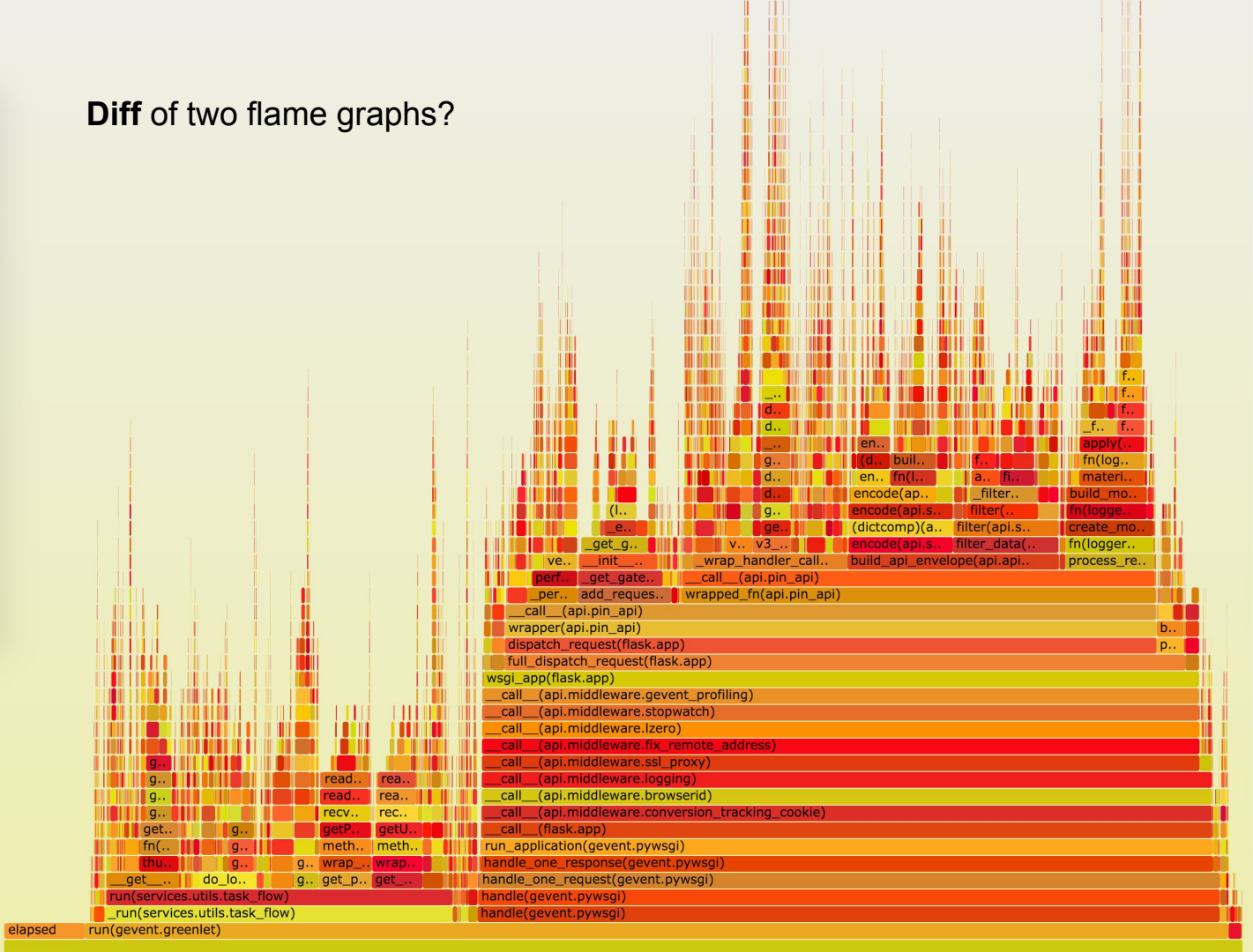

Reset Zoom
Flame Graph
Search
build_api_envelope (2,956 samples, 16.53%)
__call__ (6,752 samples, 37.76%)
wrapped_fn (6,768 samples, 37.85%)
__call__ (9,297 samples, 51.99%)
wrapper (9,307 samples, 52.05%)
dispatch_request (9,315 samples, 52.09%)
full_dispatch_request (9,725 samples, 54.39%)
wsgi_app (10,242 samples, 57.28%)
encode (1,436 samples, 8.03%)
(api.serializers.dict) (1,436 samples, 8.03%)
encode (1,436 samples, 8.03%)
filter (1,487 samples, 8.32%)
filter_data (1,492 samples, 8.34%)
encode (1,211 samples, 6.77%)
fn (208 s..
fil..
f..
_fi..
filter (971 samples, 5.43%)
find_users_in_data (..
_filter_model (876 samples, 4.90%)
get_m..
visit (217 ..
(ap..
encode (434 samples, 2...
fn (683 samples, 3.82%)
build (208 ..
_f..
filt..
apply_filter_chain (..
filter (447 samples, 2.50%)
get_us..
visit (215 ..
en..
(api.serializers.dict) (429..
build (682 samples, 3.81%)
__cal..
fi..
filter (291 samp..
f..
filter..
_filter_model (2..
get_us..
_visit_mod..
encode (423 samples, 2...
_g..
_..
_map_im..
_m..
_..
filter ..
i..
filter (..
apply_fi..
filter..
_..
_visit_mo..
build (277 sam..
ur..
to..
to_api_di..
get..
fi..
apply_f..
fi..
fil..
_filt..
recogni..
fn (279 samples..
t..
ge..
build_p..
get..
filte..
filter..
on_visit..
f..
g..
fn..
get_t..
fn (145..
get..
g..
get_th..
_c..
get_..
t..
get_th..
get_..
co..
app..
nest..
f..
t..
ge..
to..
_..
l..
_..
fil..

Sampling Profiler

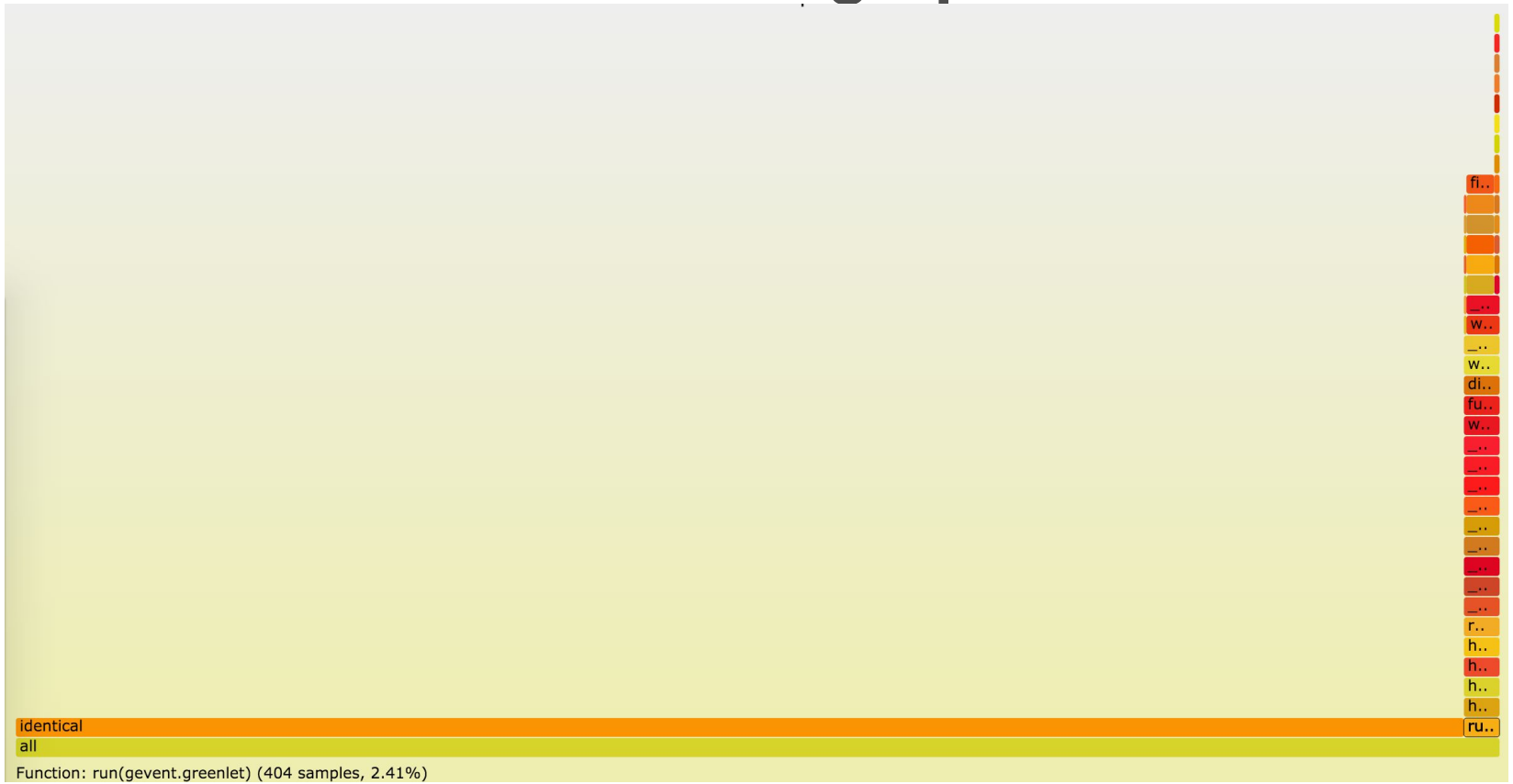
in production

- `$ sudo kill -SIGUSR2 <pid>`
- **wait a few minutes to collect enough data**
- `$ ssh host “curl localhost:16384” > profile.out`
- `$ sudo kill -SIGUSR2 <pid>`

Diff of two flame graphs?

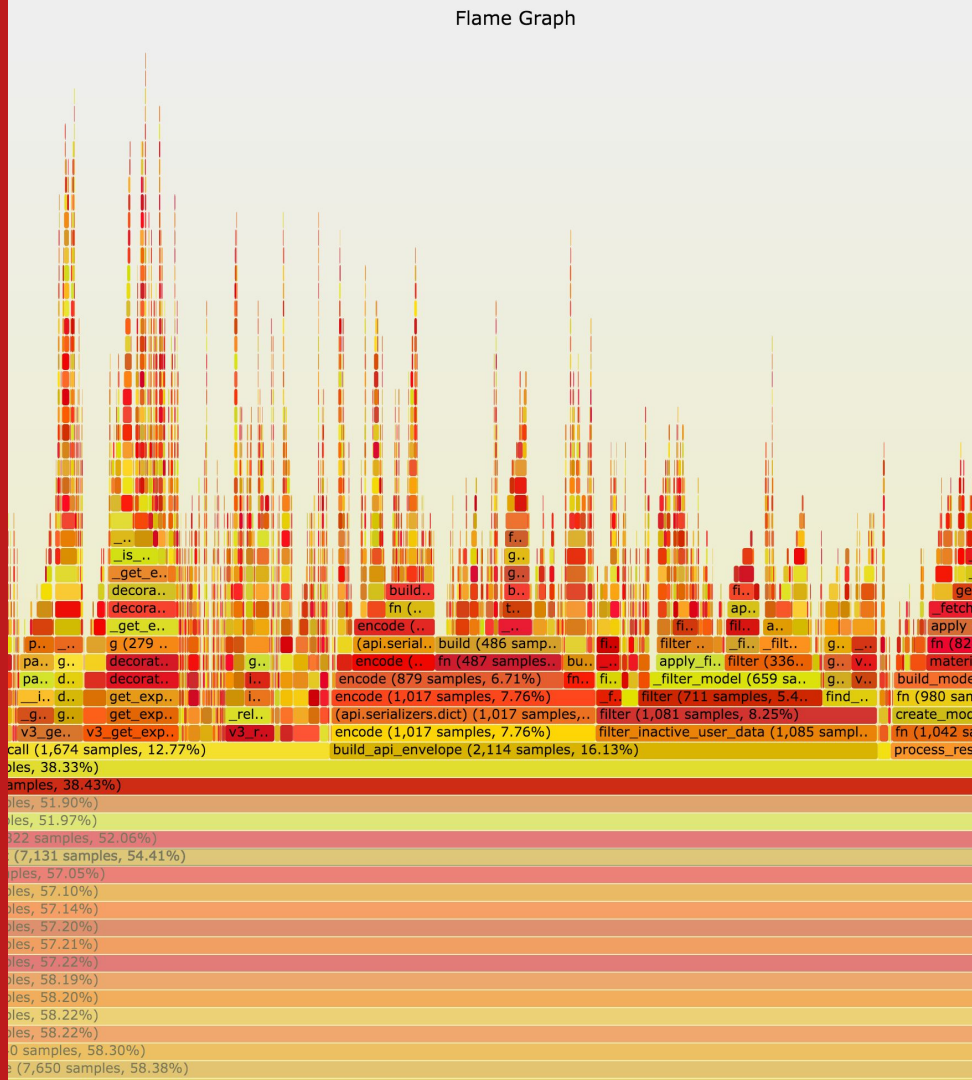


Diff Flamegraph



Diff Flamegraph

Examples


Flame Graph
encode (1,017 samples, 7.76%)
(api.serializers.dict) (1,017 samples,..
filter (1,081 samples, 8.25%)
encode (1,017 samples, 7.76%)
filter_inactive_user_data (1,085 sampl..
call (1,674 samples, 12.77%)
build_api_envelope (2,114 samples, 16.13%)
process_res
les, 38.33%)
amples, 38.43%)
les, 51.90%)
les, 51.97%)
22 samples, 52.06%)
(7,131 samples, 54.41%)
les, 57.05%)
les, 57.10%)
les, 57.14%)
les, 57.20%)
les, 57.21%)
les, 57.22%)
les, 58.19%)
les, 58.20%)
les, 58.22%)
les, 58.22%)
0 samples, 58.30%)
(7,650 samples, 58.38%)

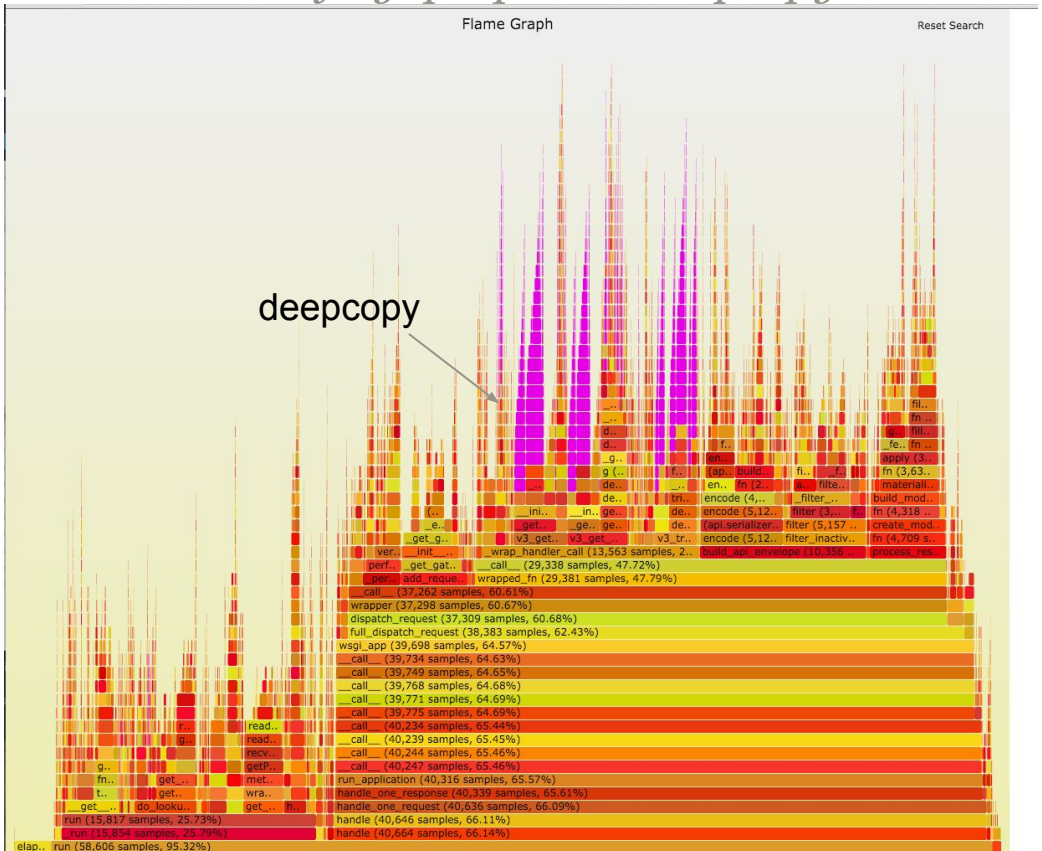
API Profiler

30ms of home feed spent in deepcopy

```
$ python walk.py GET.v3.feeds.home.000622ms.1452734743.prof
<snip>
cumtime 0.0553 - ../api/handlers/v3/experience_handlers.py:21(_get_experience_decider)
cumtime 0.0553 - ../core/logic/experiences/experience_logic.py:56(__init__)
cumtime 0.0309 - /usr/lib/python2.7/copy.py:145(deepcopy)
cumtime 0.0309 - /usr/lib/python2.7/copy.py:253(_deepcopy_dict)
cumtime 0.0288 - ../core/logic/experiences/exps_config/exps_config.py:197(experiences)
cumtime 0.0234 - ../core/logic/experiences/resource_parser.py:17(parse)
cumtime 0.0149 - /usr/lib/python2.7/copy.py:226(_deepcopy_list)
cumtime 0.0090 - ../core/logic/experiences/resource_parser.py:60(_get_resource_by_request_context)
```

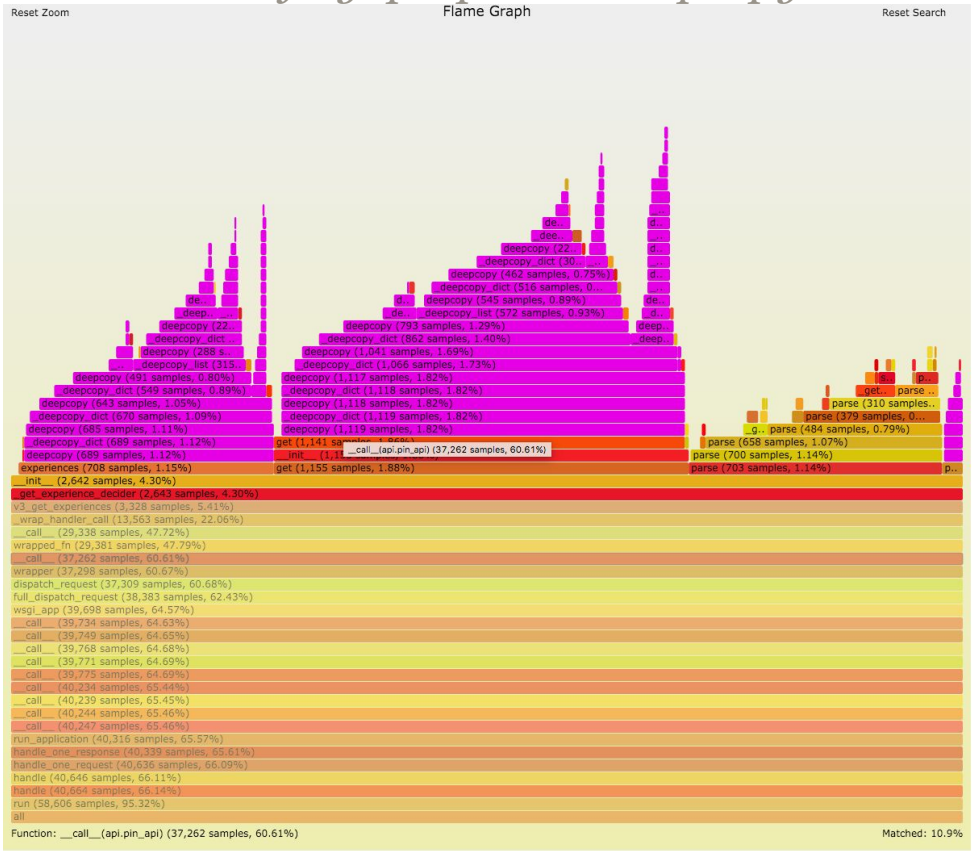
Sampling Profiler: Flame Graph

10% of ngapi spent in deepcopy



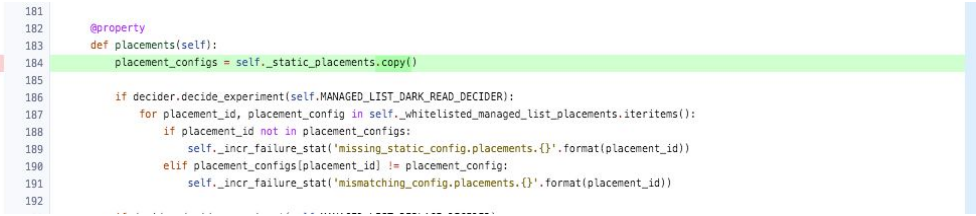
sampling profiler: flame graph

10% of ngapi spent in deepcopy



diff

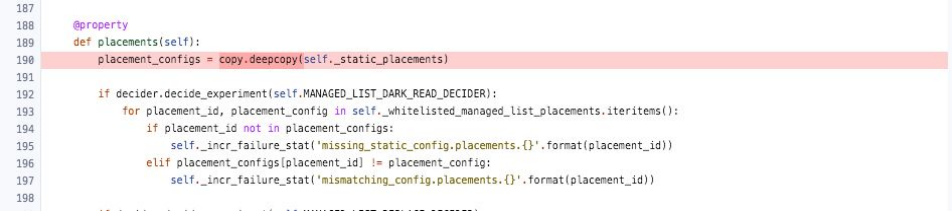
```
187
188     @property
189     def placements(self):
190         placement_configs = copy.deepcopy(self._static_placements)
191
192         if decider.decide_experiment(self.MANAGED_LIST_DARK_READ_DECIDER):
193             for placement_id, placement_config in self._whitelisted_managed_list_placements.iteritems():
194                 if placement_id not in placement_configs:
195                     self._incr_failure_stat('missing_static_config.placements.{}'.format(placement_id))
196                 elif placement_configs[placement_id] != placement_config:
197                     self._incr_failure_stat('mismatching_config.placements.{}'.format(placement_id))
198
```



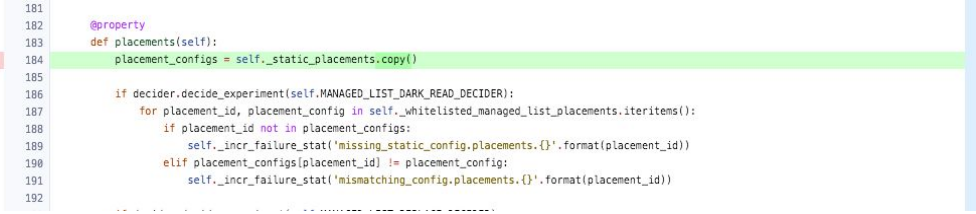
```
181
182     @property
183     def placements(self):
184         placement_configs = self._static_placements.copy()
185
186         if decider.decide_experiment(self.MANAGED_LIST_DARK_READ_DECIDER):
187             for placement_id, placement_config in self._whitelisted_managed_list_placements.iteritems():
188                 if placement_id not in placement_configs:
189                     self._incr_failure_stat('missing_static_config.placements.{}'.format(placement_id))
190                 elif placement_configs[placement_id] != placement_config:
191                     self._incr_failure_stat('mismatching_config.placements.{}'.format(placement_id))
192
```

```
115         self._config = data
116
117     def get(self):
118         # Deep copy to avoid passing data around by reference.
119         return copy.deepcopy(self._config)
120
121
122 managed_list_config = ManagedListConfig()
123 static_config = StaticConfig()
```

```
109         self._config = data
110
111     def get(self):
112         # Deep copy to avoid passing data around by reference.
113         return self._config
114
115
116 managed_list_config = ManagedListConfig()
117 static_config = StaticConfig()
```

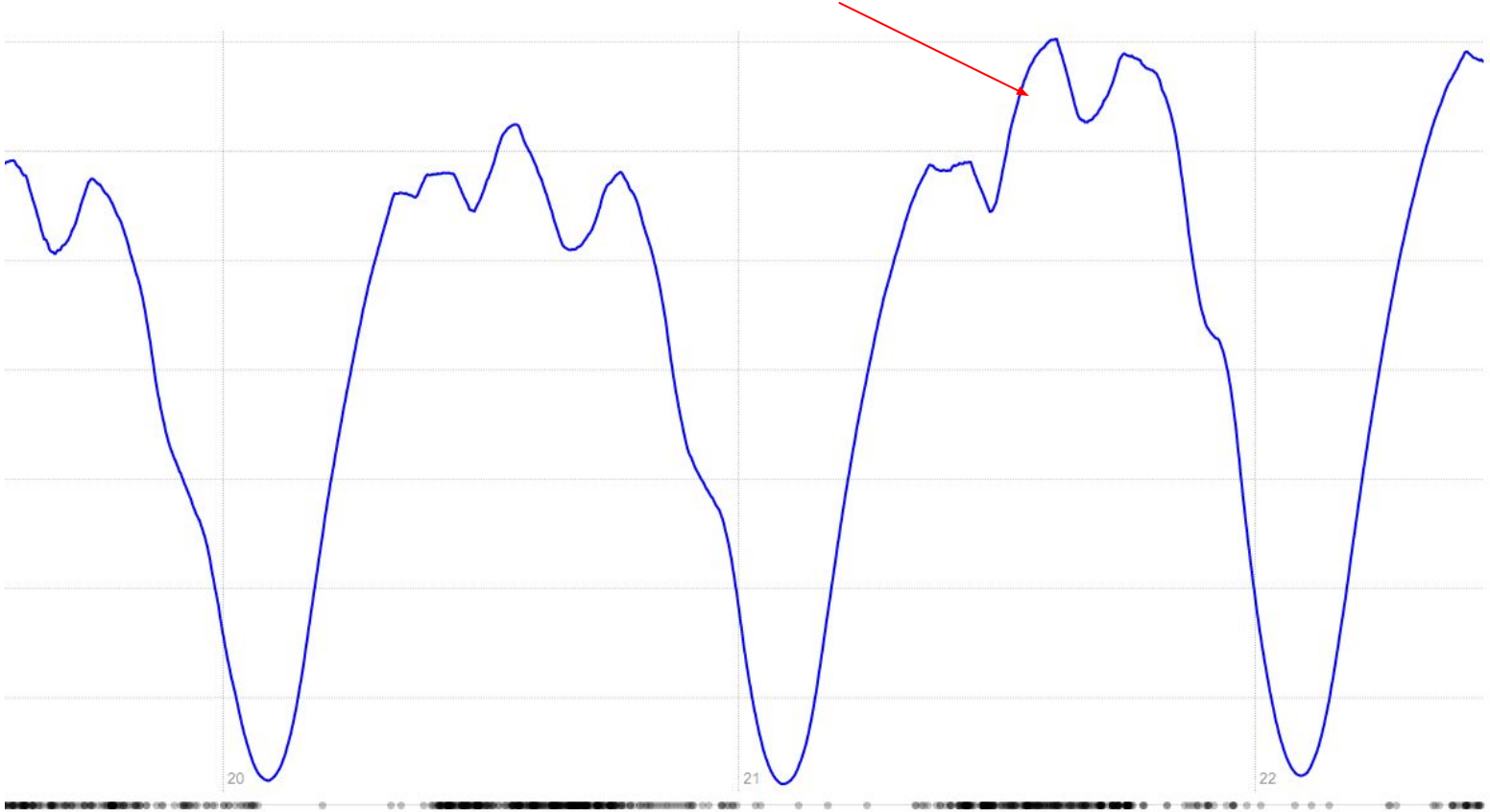


```
187
188     @property
189     def placements(self):
190         placement_configs = copy.deepcopy(self._static_placements)
191
192         if decider.decide_experiment(self.MANAGED_LIST_DARK_READ_DECIDER):
193             for placement_id, placement_config in self._whitelisted_managed_list_placements.iteritems():
194                 if placement_id not in placement_configs:
195                     self._incr_failure_stat('missing_static_config.placements.{}'.format(placement_id))
196                 elif placement_configs[placement_id] != placement_config:
197                     self._incr_failure_stat('mismatching_config.placements.{}'.format(placement_id))
198
```



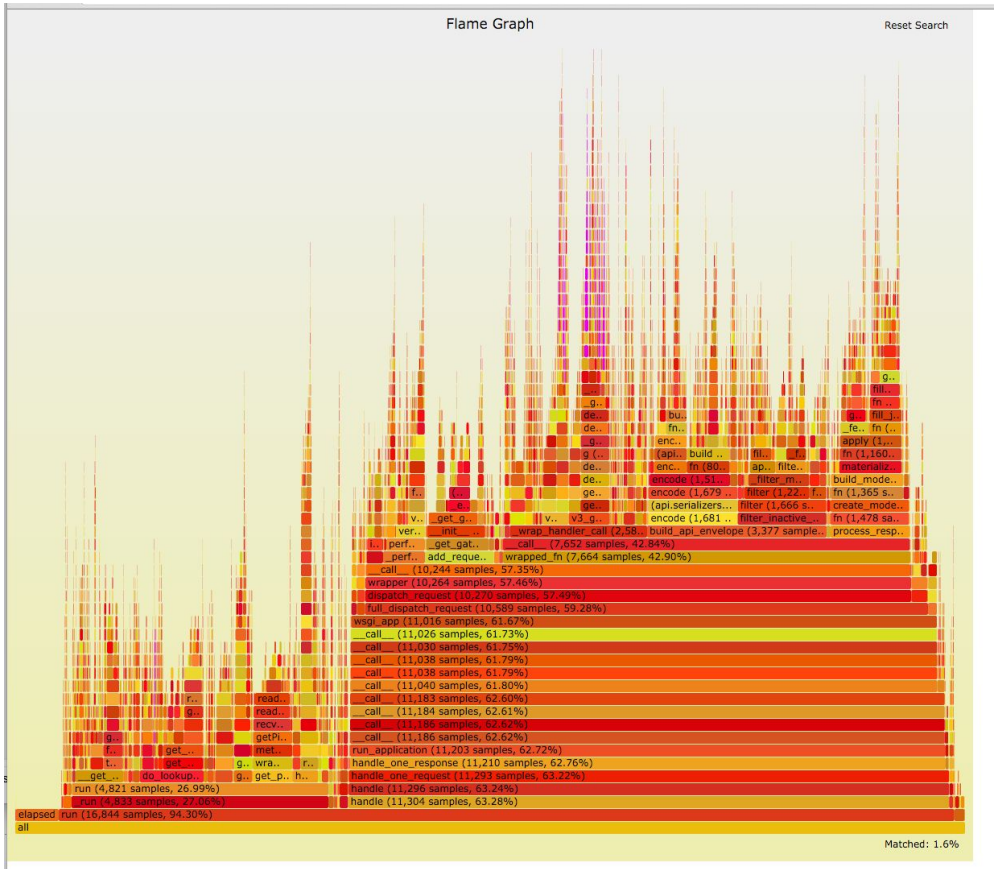
```
181
182     @property
183     def placements(self):
184         placement_configs = self._static_placements.copy()
185
186         if decider.decide_experiment(self.MANAGED_LIST_DARK_READ_DECIDER):
187             for placement_id, placement_config in self._whitelisted_managed_list_placements.iteritems():
188                 if placement_id not in placement_configs:
189                     self._incr_failure_stat('missing_static_config.placements.{}'.format(placement_id))
190                 elif placement_configs[placement_id] != placement_config:
191                     self._incr_failure_stat('mismatching_config.placements.{}'.format(placement_id))
192
```

RPS / host



sampling profiler: flame graph

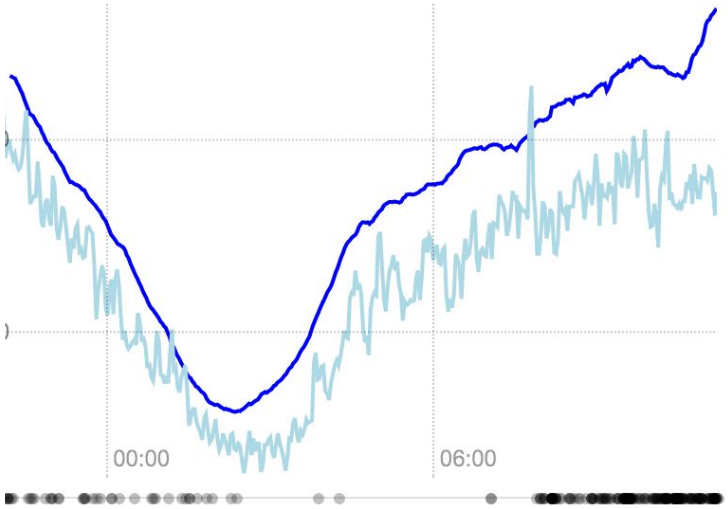
1.6% of ngapi spent in deepcopy



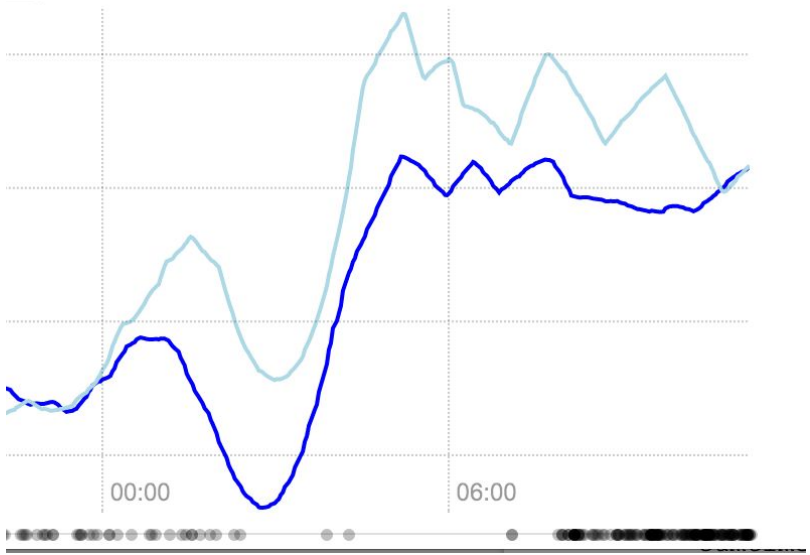
Example: Color Difference

API latency increase, RPS/host decrease

Latency

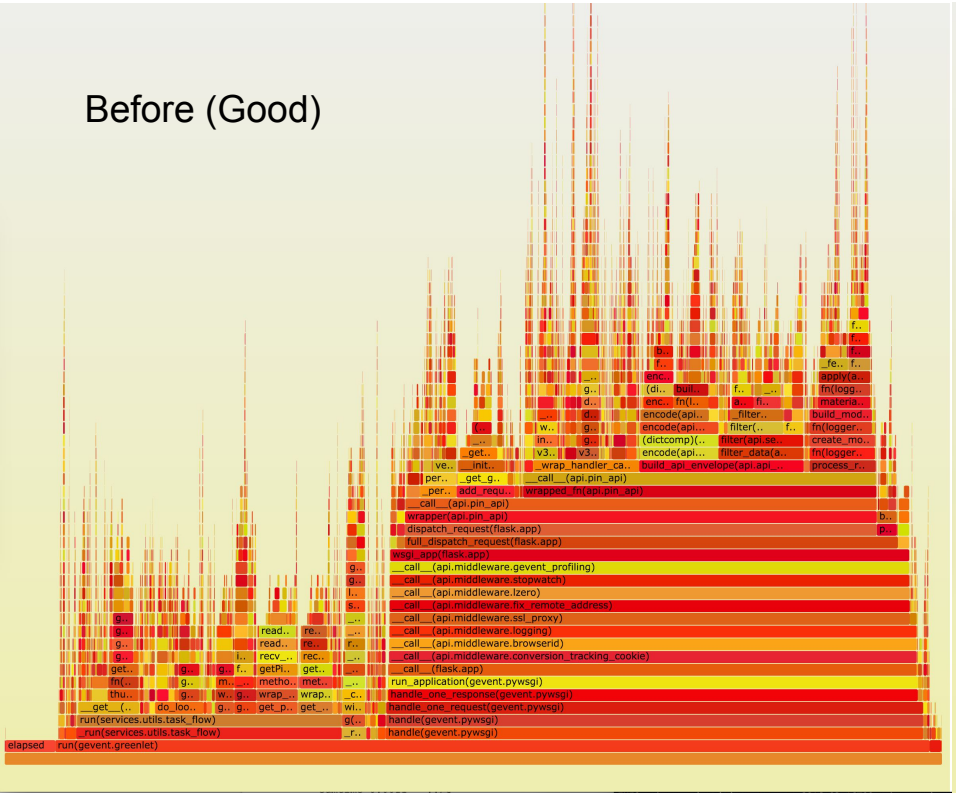


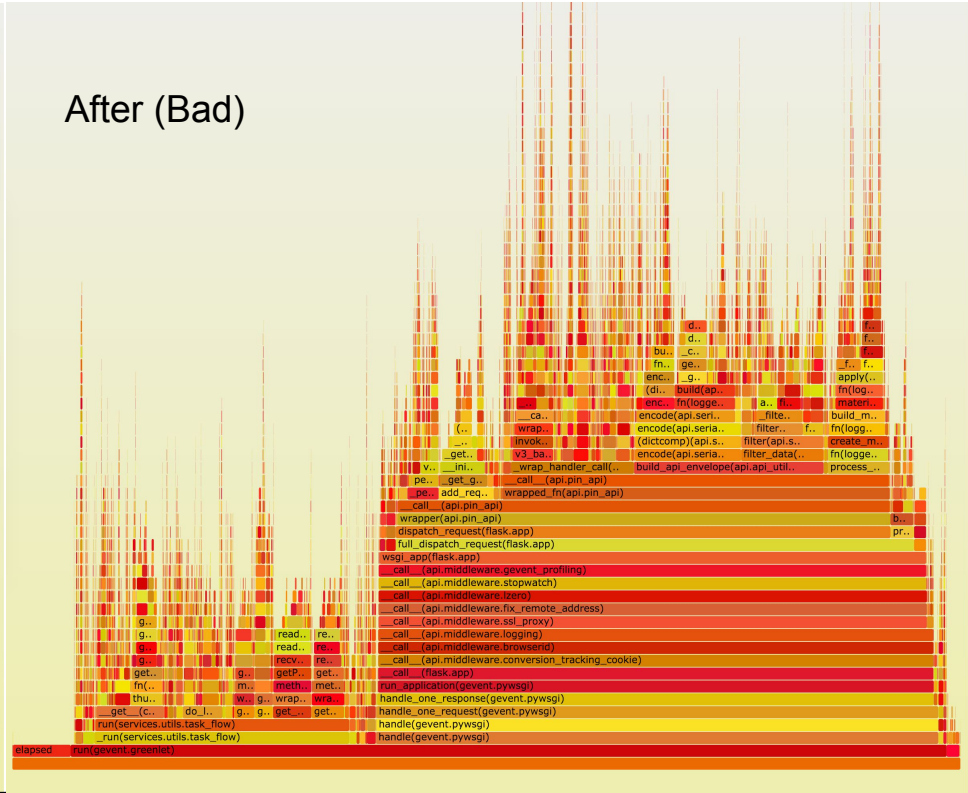
RPS/Host



Example: Color Difference

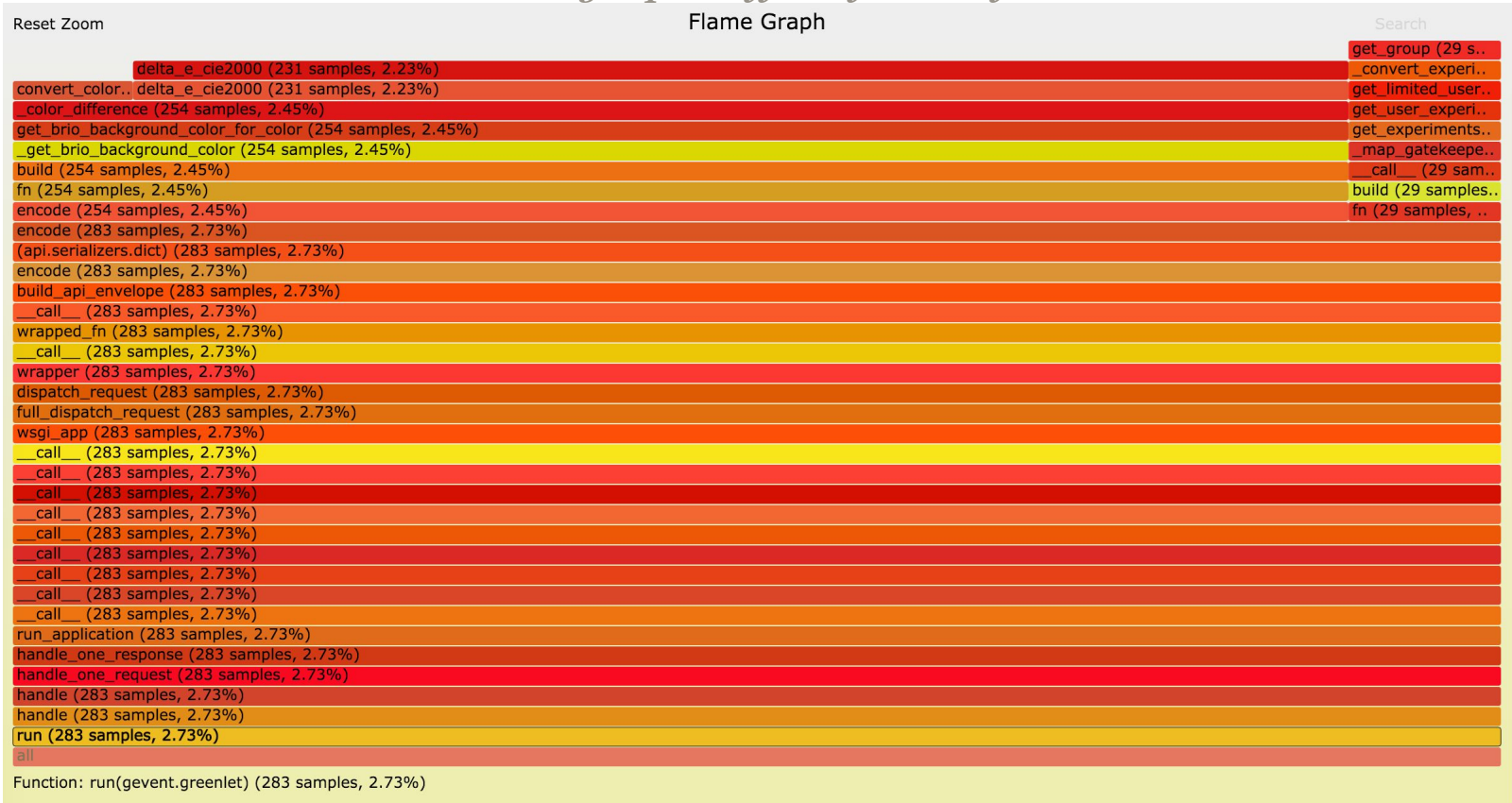
Flamegraphs





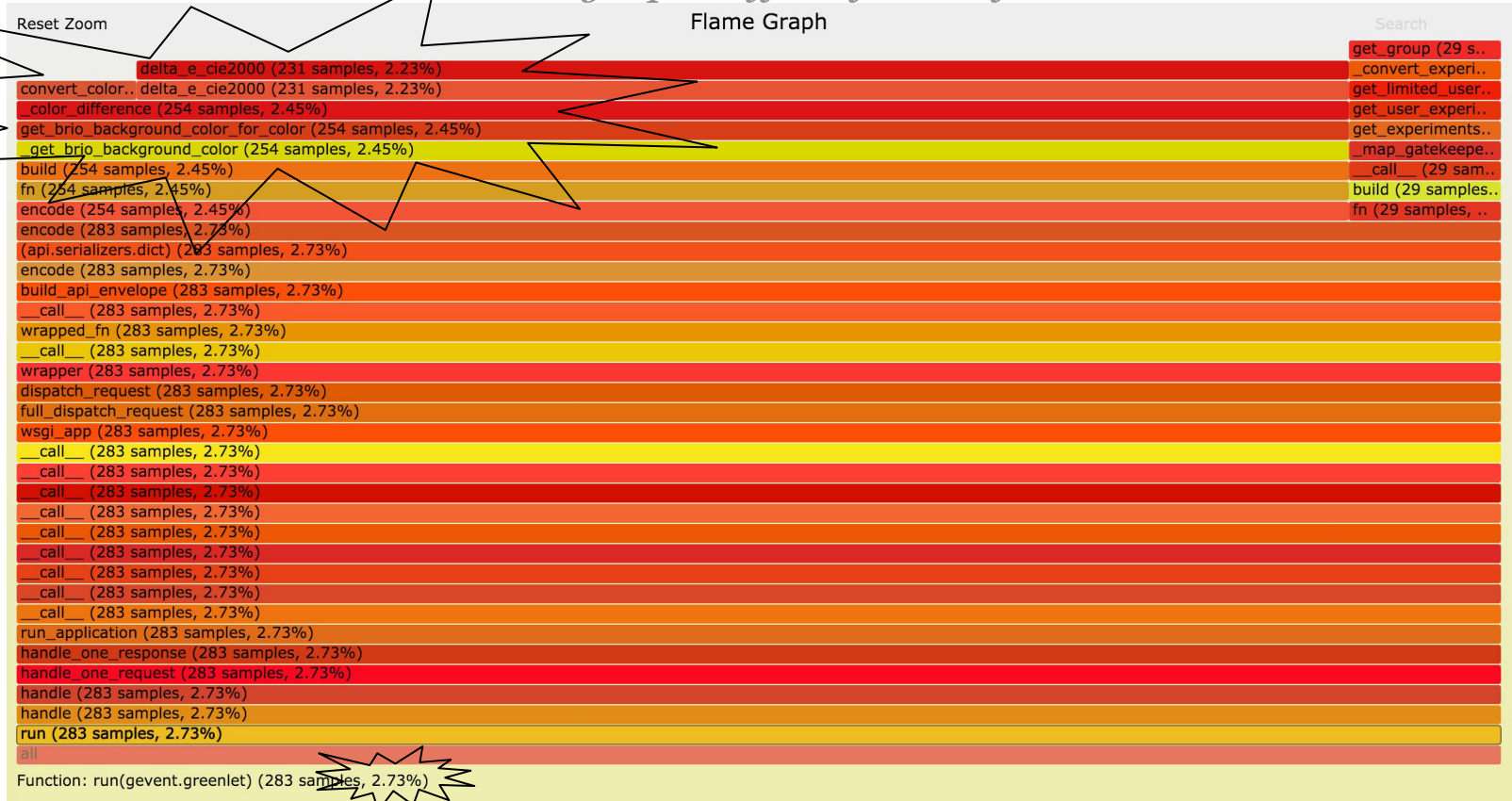
Example: Color Difference

Flamegraph Diff: Before - After



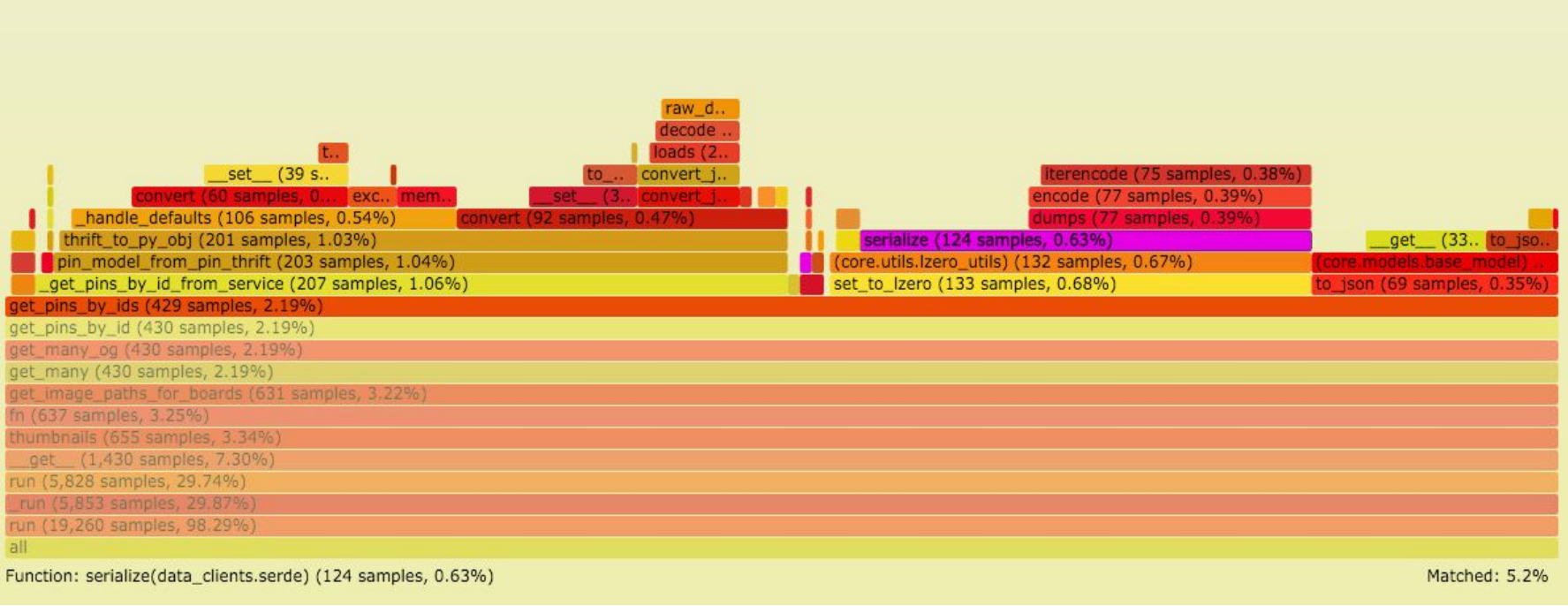
Example: Color Difference

Flamegraph Diff: Before - After



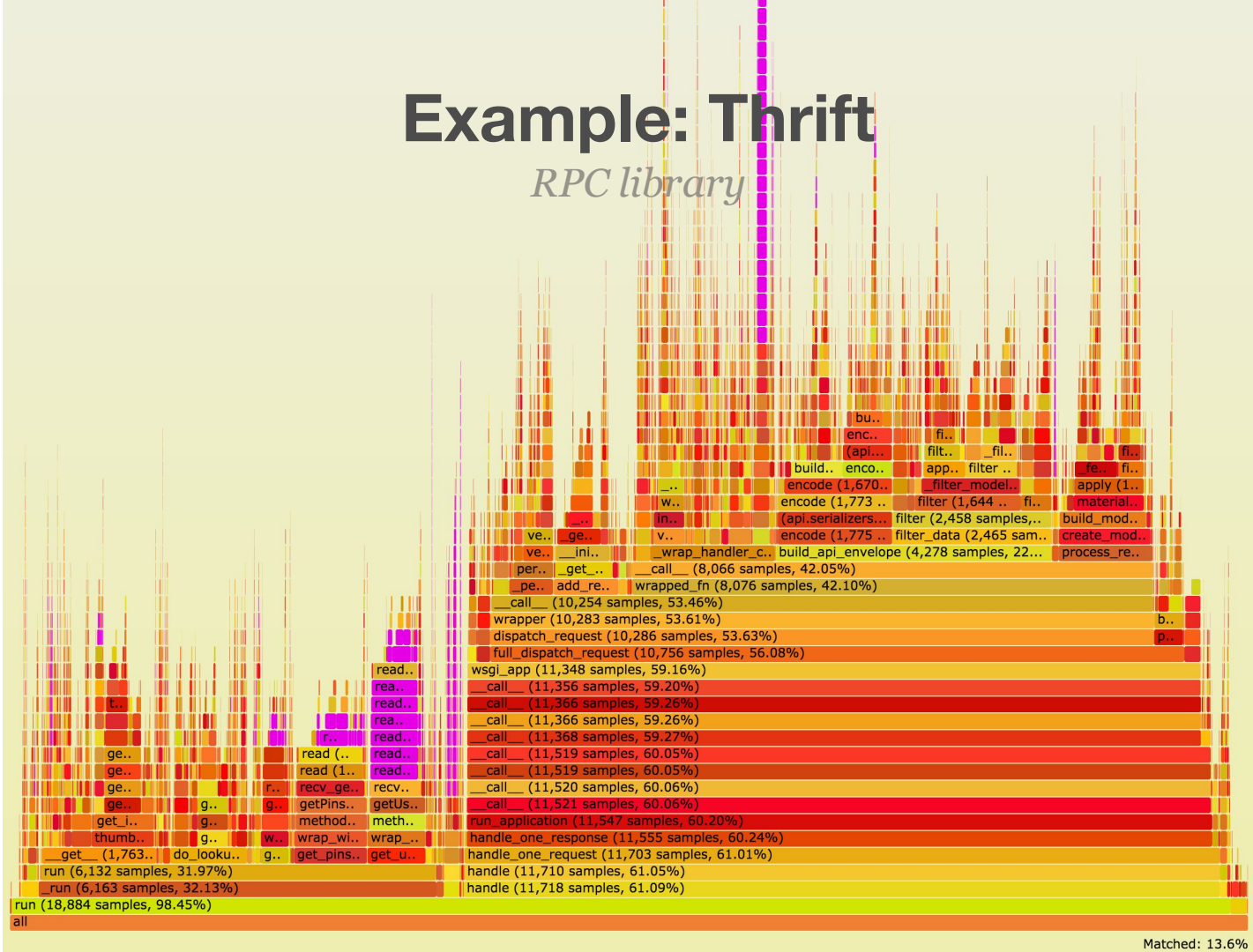
Example: Cache serialization

5%



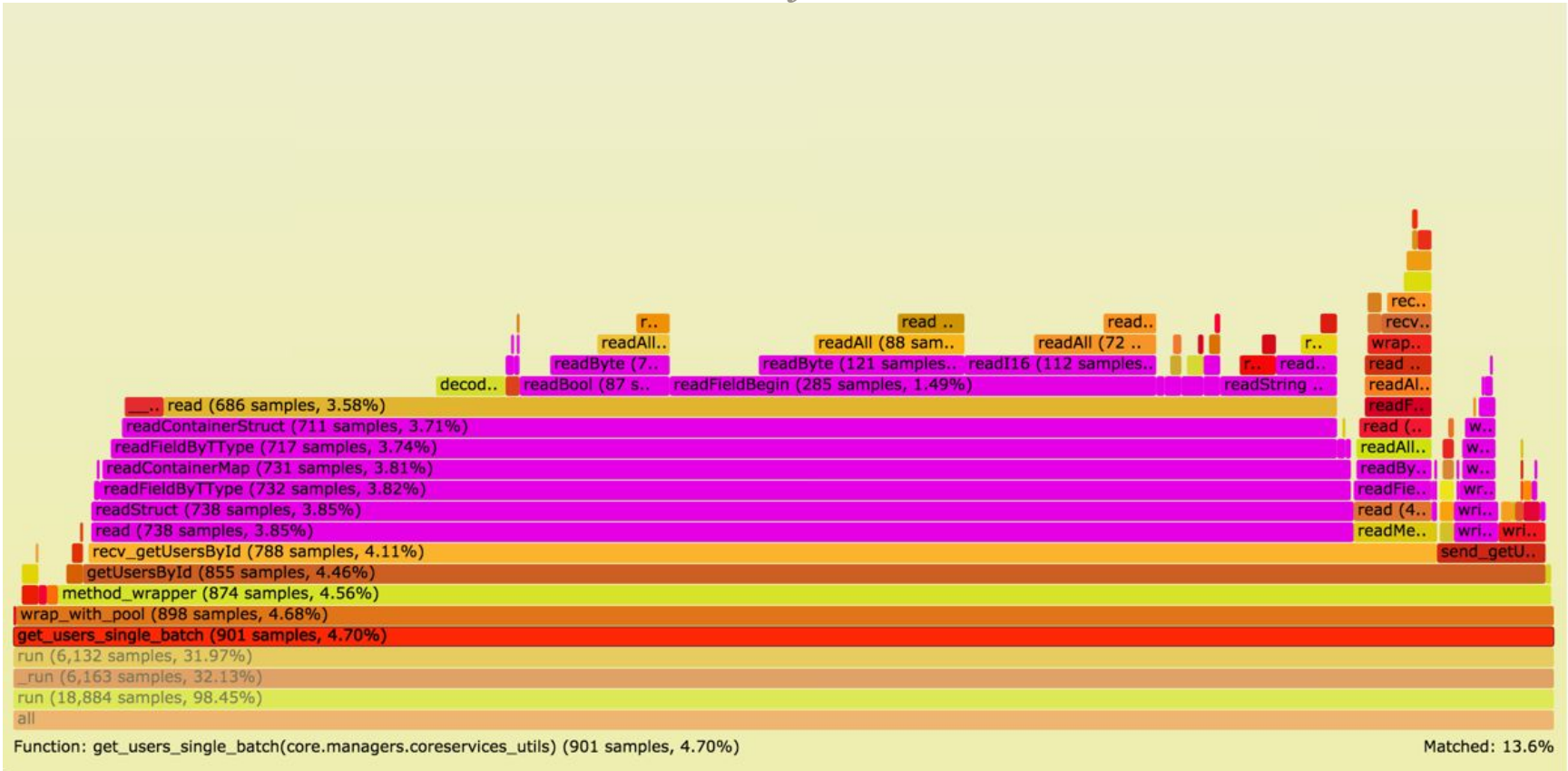
Example: Thrift

RPC library



Example: Thrift

Before



Example: Thrift

fastbinary

```
class TBinaryProtocolAccelerated(TBinaryProtocol):
    """C-Accelerated version of TBinaryProtocol.

    This class does not override any of TBinaryProtocol's methods,
    but the generated code recognizes it directly and will call into
    our C module to do the encoding, bypassing this object entirely.
    We inherit from TBinaryProtocol so that the normal TBinaryProtocol
    encoding can happen if the fastbinary module doesn't work for some
    reason.  (TODO(dreiss): Make this happen sanely in more cases.)
    To disable this behavior, pass fallback=False constructor argument.

    In order to take advantage of the C module, just use
    TBinaryProtocolAccelerated instead of TBinaryProtocol.
```

Example: Thrift

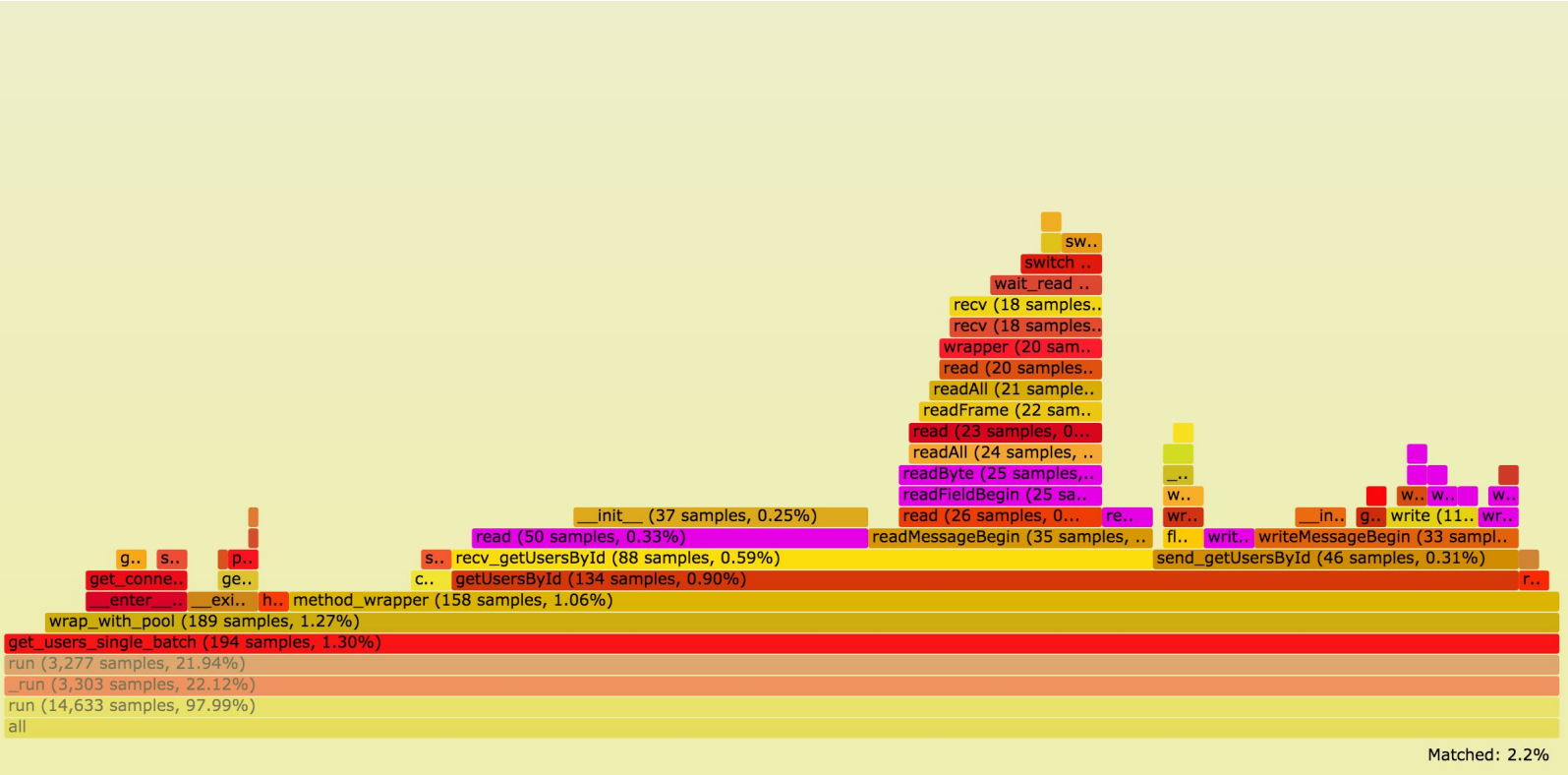
https://github.com/pinterest/thrift/commits/0.9.3-pinterest

- **Backport fixes for fastbinary from unreleased thrift 0.10.0**
- **Unicode support**
- **Support subclasses of `TBinaryProtocolFactory`**
- **Other bug fixes**

```
  indent_up();

-   indent(out) << "if iprot.__class__ == TBinaryProtocol.TBinaryProtocolAccelerated "
+   indent(out) << "if isinstance(iprot, TBinaryProtocol.TBinaryProtocolAccelerated) "
                   "and isinstance(iprot.trans, TTransport.CReadableTransport) "
                   "and self.thrift_spec is not None "
```

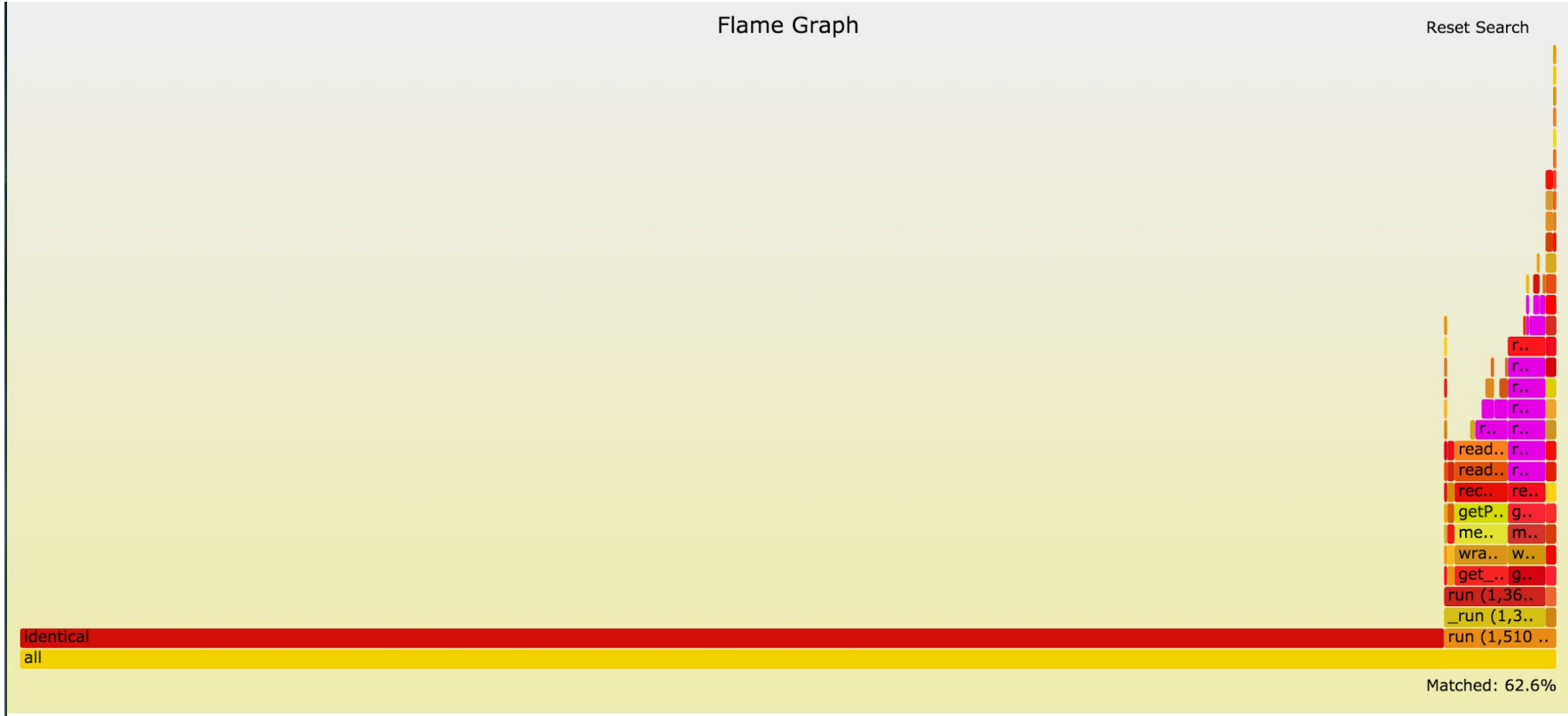
Example: Thrift

After



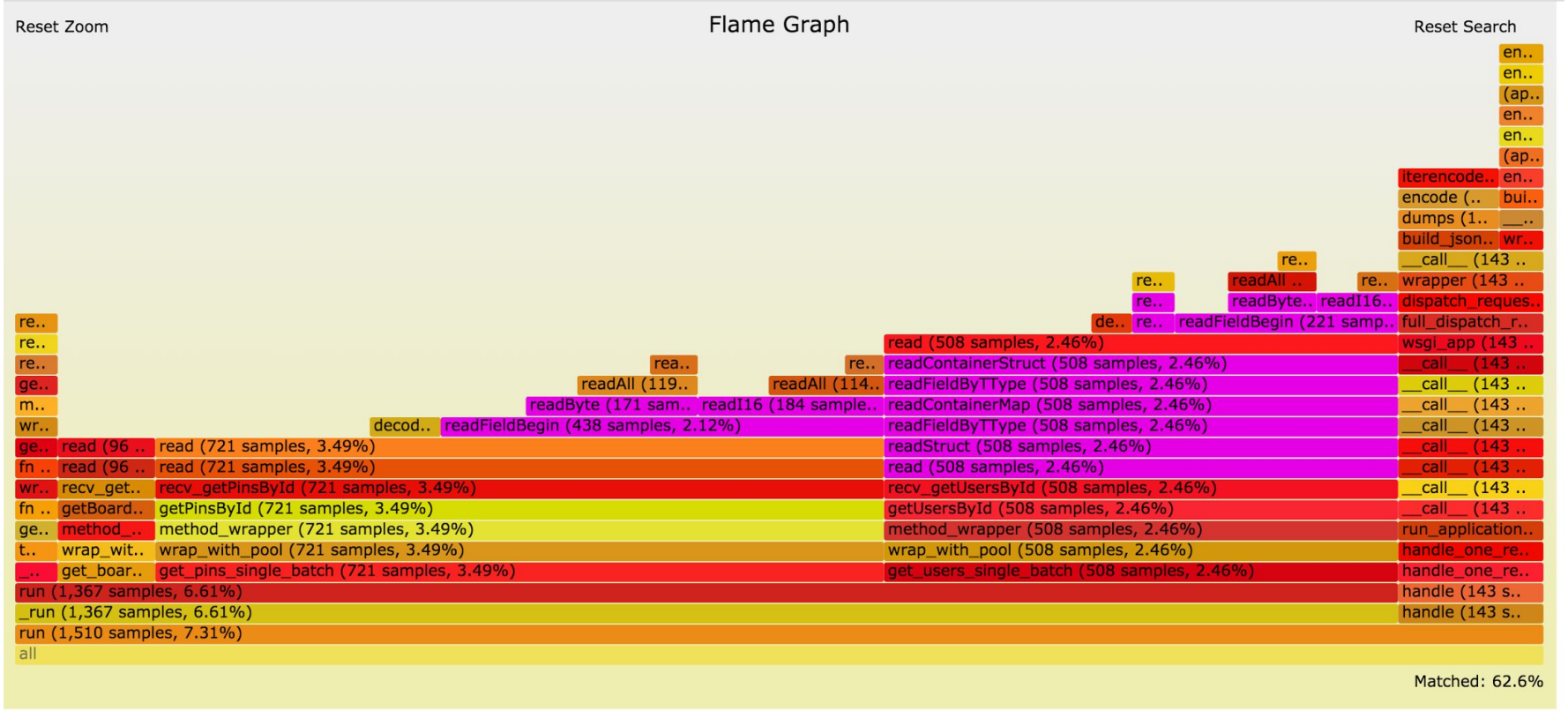
Example: Thrift

Difference: Before - After



Example: Thrift

Difference: Before - After



Problem

Gevent

Profiling

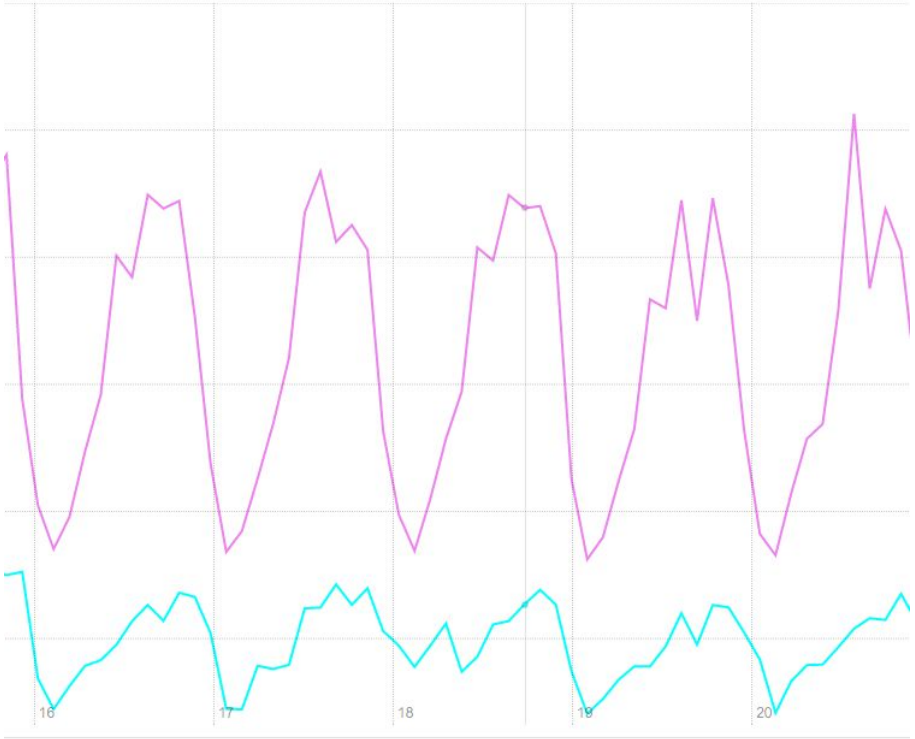
Impact

Pin Search P90 Latency

Q4 2015

- **Latency**
  - Varnish: 2200ms
  - Search system: 640ms

- **70% of latency outside of Search system**
- **Height of Varnish latency >> height of asterix latency**

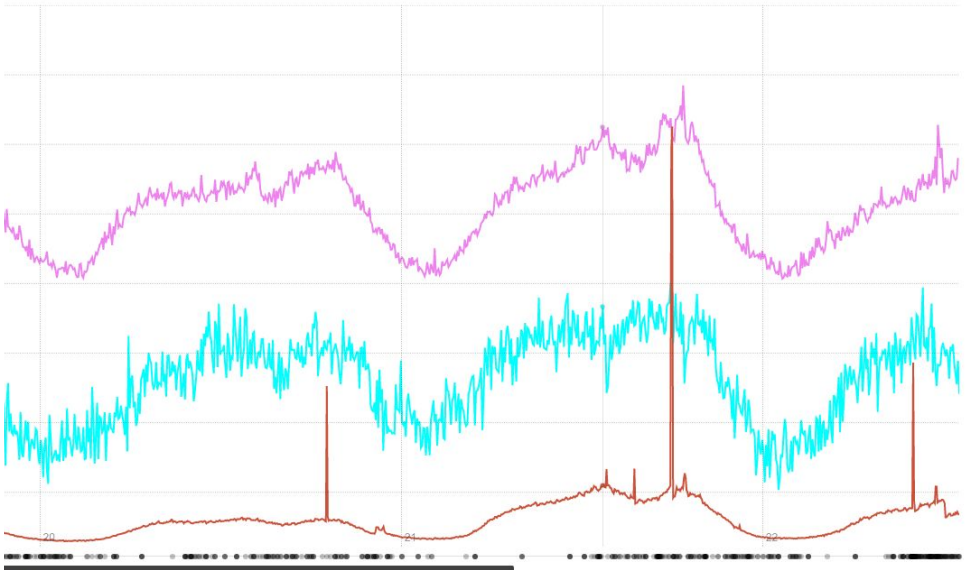


Pin Search P90 Latency

Today

- **Latency**
  - Varnish: 550ms
  - Search system: < 300ms

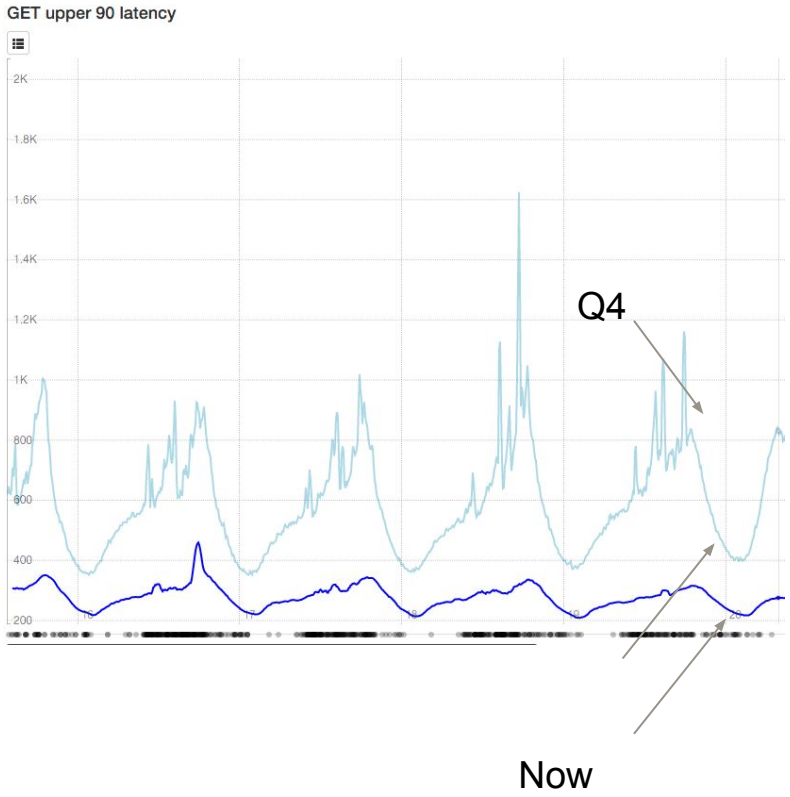
- **50% of search latency outside of Asterix**
- **Height of Varnish latency == height of asterix latency**



GET API Latency

Q4 vs today

- **Height (peak - valley)**
  - Q4: 50%
  - Now:22%



Conclusion

Conclusion

Lessions

- **Understand bottlenecks for latency and efficiency**
  - Network, CPU, Memory etc
  - Cooperative multitasking?
- **Profile in Production**
  - Synthetic benchmarks aren’t enough
- **Visibility**
  - Profile code
  - Define efficiency metric
  - There are flame graph tools for most languages
- **Finding what to fix is harder than fixing it**
- **Start with low hanging fruit**
- **Detect and fix performance regressions**

What is Pinterest?

Pinterest

Pinterest is the world's catalog of ideas. Find and save recipes, parenting hacks, style inspiration and other ideas to try.