Scaling python webapps from 0 to 50 million users - A top-down approach

Jinal Jhaveri

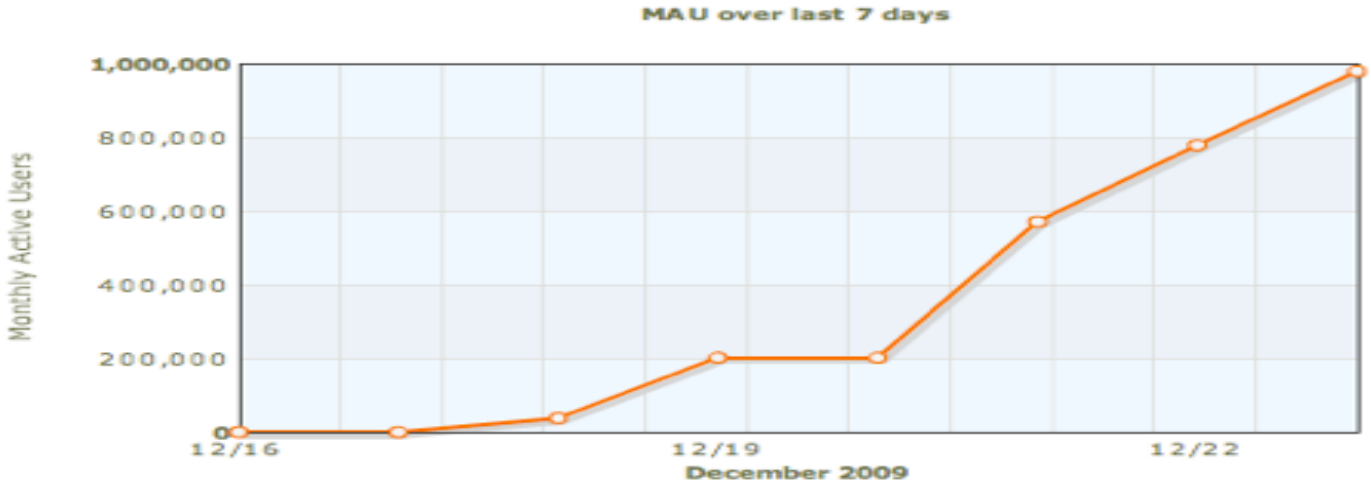
jinal@lolapps.com

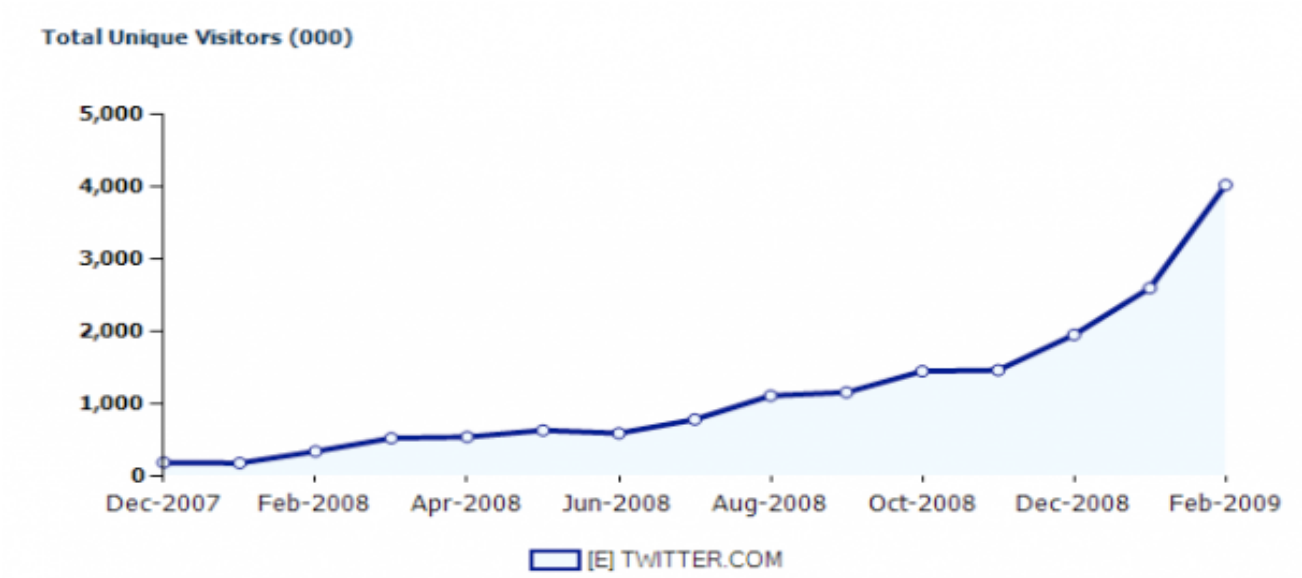
Social Games!!

Agenda

- Why is performance a big issue for social games?
- Architecture
- Bottlenecks and solutions
- Performance strategy
- Questions

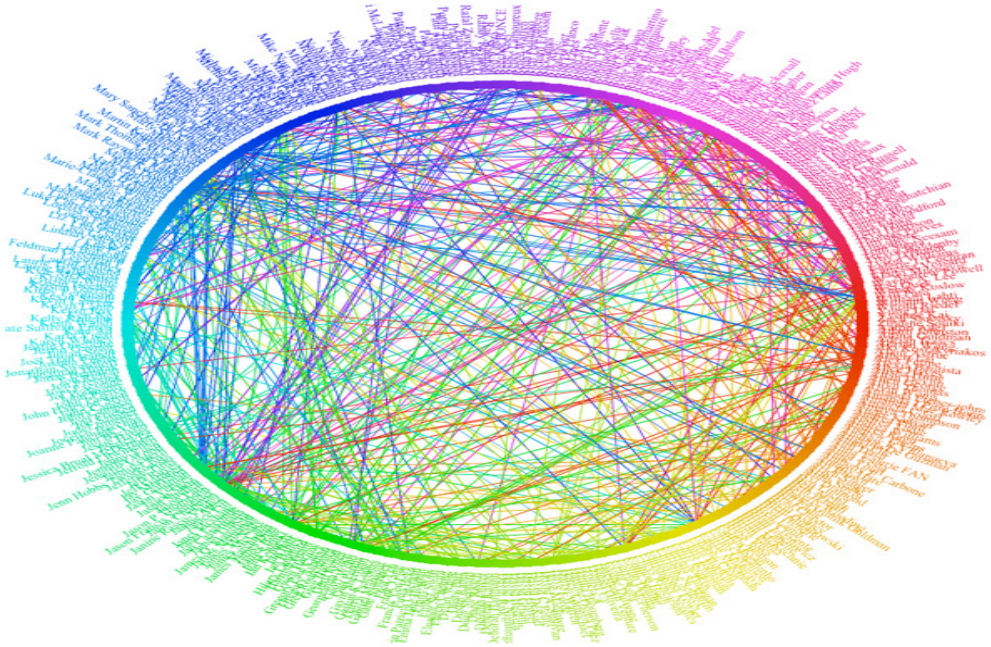
Why is performance a big issue for social games?





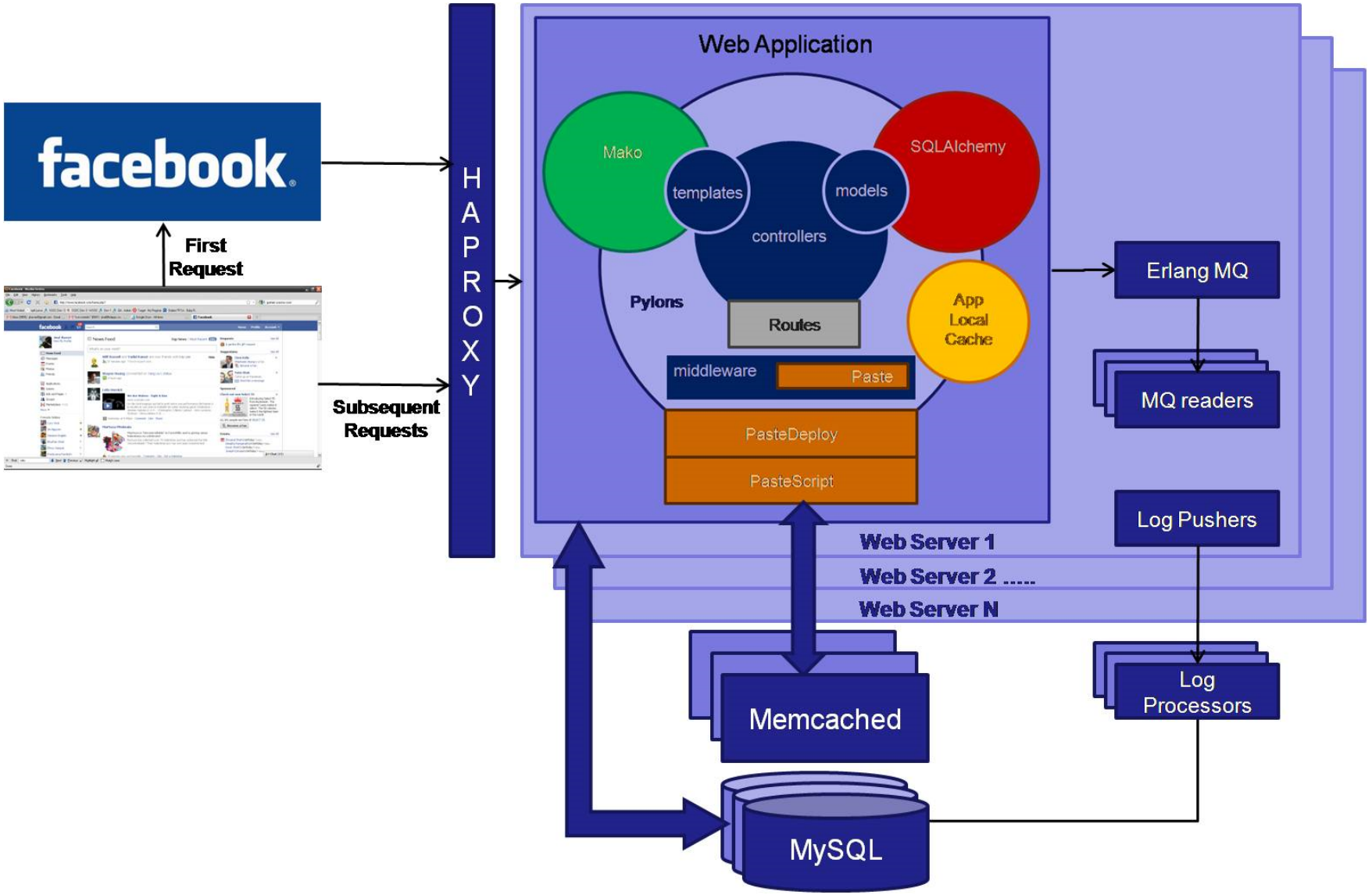
Why is performance a big issue for social games?

- Extremely high virality
  installs, notifications, emails, feeds, events
- Amount of time spent is high



http://www.bestfacebookapplications.com

Game Architecture



Bottlenecks and Solutions

- **Load Balancer**
- Web Server
- Web Application
- Browser

Load Balancer

- HAProxy
- Roundrobin
- No gzip / no file serving
- Supports ipbased / regex based load balancing

Bottlenecks and Solutions

- Load Balancer
- **Web Server**
- Web Application
- Browser

Webserver

- Paster
- n instances (n = no.of cpu) (10 threads each)
- Timeout (10 seconds)
- Disable Nagles optimization

Bottlenecks and Solutions

- Load Balancer
- Web Server
- **Web Application**
- Browser

Web Application

- Memcached to avoid DB trips
  - ORM integration
  - Compression
  - Caching non-existence
  - Lists cache

Memcache / ORM integration

```
def get(self, query, ident, *args, **kwargs):
 key = query.mapper.identity_key_from_primary_key(ident)
 obj = query.session.identity_map.get(key)
 if obj:
      return obj
 mkey = gen_cache_key(key[0].__name__, key[1], self.version_string)
 obj = self.mclient.get(mkey)
 if obj is None:
      obj = query._get(key, ident, **kwargs)
      if obj is not None:
          query.session.expunge(obj)
          self.mclient.set(mkey, obj)
 if obj:
      return query.session.merge(obj, dont_load=True)
 else:
      return None
```

Mike Nelson

- Memcached to avoid DB trips
  - ORM integration
  - Compression
  - Caching non-existence
  - Lists cache
  - Best Effort caching

Web Application

- Local cache
- PythonSpeed/PerformanceTips (wiki.python.org)
- Asynchronous
  - Facebook api calls
  - Log processing
  - Event tracking
- Partial rendering / json / ajax

Bottlenecks and Solutions

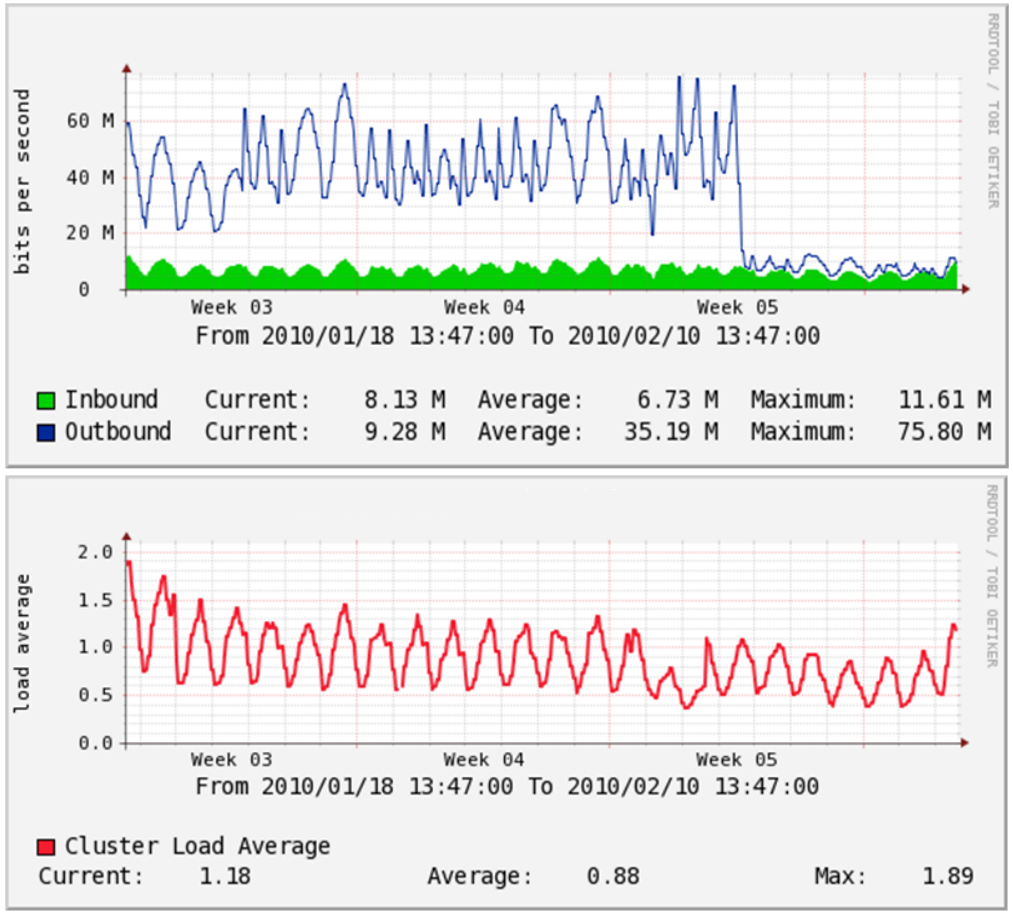
- Load Balancer
- Web Server
- Web Application
- **Browser**

Browser – Best practices

- Gzip
- CDN
- Loading images in parallel
- Ajaxification
- Client side caching

Gzip

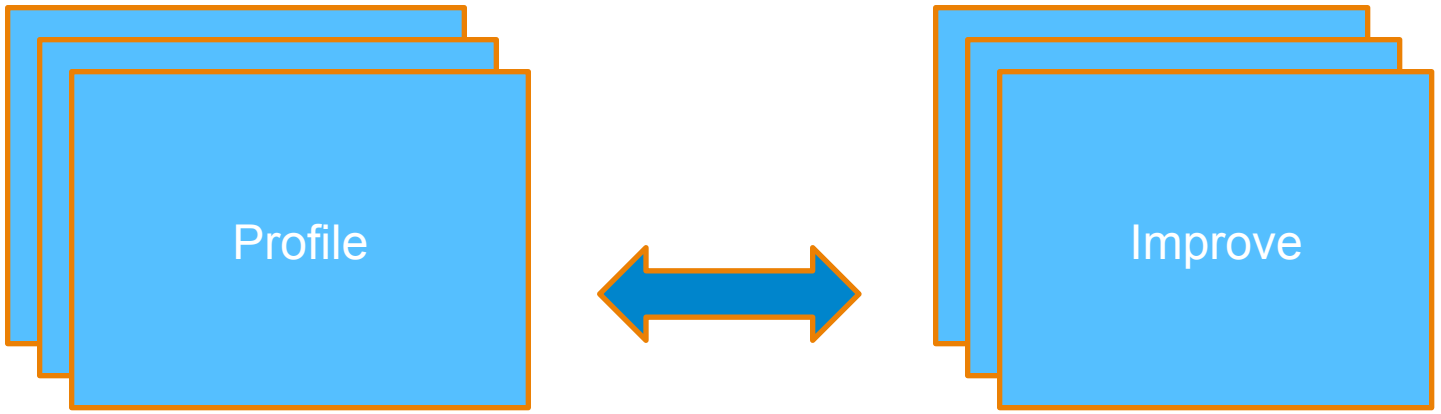
```
from paste.gzipper import make_gzip_middleware
app = make_gzip_middleware(app, global_conf, compress_level=1)
```



Browser – Best practices

- Gzip
- CDN
- Loading images in parallel
- Ajaxification
- Client side caching

Performance strategy



- Measure, Measure, Measure
  - load balancer request time
  - Web server request time
  - Controller request time
  - Rendering time

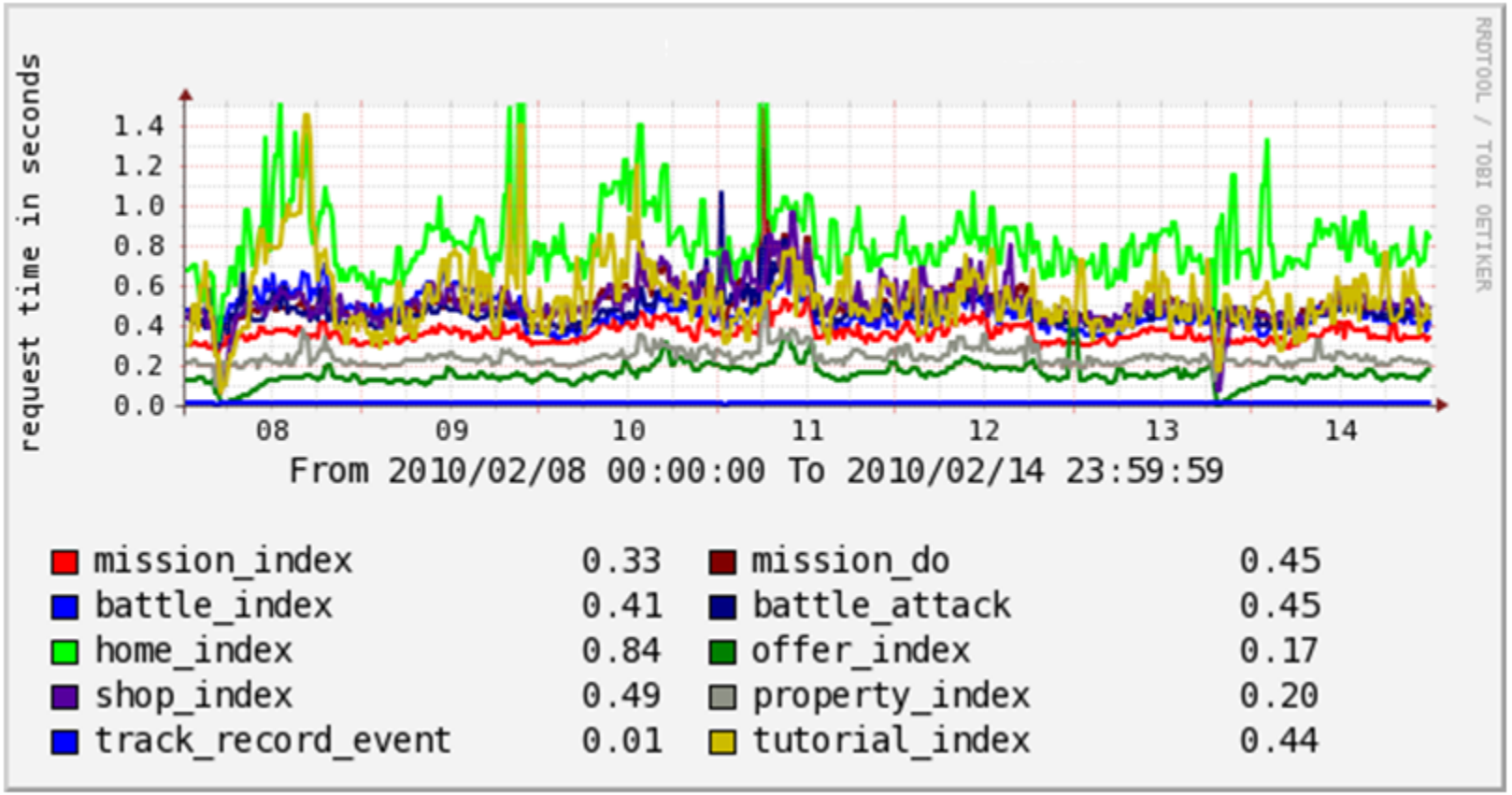
Profiling - Middleware

```
class TimerMiddleware(object):
    """
    Simple middleware that logs the time of each request to the provided logger.
    @author Brian Rue
    """
    def __init__(self, app, log, name='outer'):
        self.app = app
        self.log = log
        self.name = name

    def __call__(self, environ, start_response):
        start_time = time.time()
        try:
            return self.app(environ, start_response)
        finally:
            end_time = time.time()
            url = environ.get('PATH_INFO', '')
            if environ.get('QUERY_STRING'):
                url += '?' + environ['QUERY_STRING']
            self.log.debug("%f %s-%s" % ((end_time - start_time), self.name, url))
```

Profiling

```
2010-02-07 13:27:07 0.182282 mission/index /app/20/mission
user: 10000270498442

2010-02-07 13:27:07 0.105489 outer-/app/20/mission?
dummyid=1

2010-02-07 13:27:07 0.287437 battle/attack /app/19/battle/
attack user: 501879126

2010-02-07 13:27:07 0.006339 track/record_event /app/21/
track/record_event user: 1163511266

2010-02-07 13:27:07 0.032981 outer-/app/21/track/
record_event?file_name=base.js&cache_key=228500126472317 5

2010-02-07 13:27:07 0.006186 track/record_event /app/19/
track/record_event user: 1039662536

2010-02-07 13:27:07 0.072400 outer-/app/19/track/
record_event?file_name=base.js&cache_key=228500126542525 8
```

Profiling


request time in seconds
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
08
09
10
11
12
13
14
From 2010/02/08 00:00:00 To 2010/02/14 23:59:59
RRDTOOL / TOBI OETIKER
mission_index 0.33
battle_index 0.41
home_index 0.84
shop_index 0.49
track_record_event 0.01
mission_do 0.45
battle_attack 0.45
offer_index 0.17
property_index 0.20
tutorial_index 0.44

Profiling - Repoze

```
# establish the Registry for this application
app = registry.RegistryManager(app)
from repoze.profile.profiler import AccumulatingProfileMiddleware

app = AccumulatingProfileMiddleware(
       app,

       log_filename='/tmp/gameprofile.log',

       cachegrind_filename='/tmp/cachegrind.out.bar',

       discard_first_request=True,

       flush_at_shutdown=True,

       path='/__profile__')
```

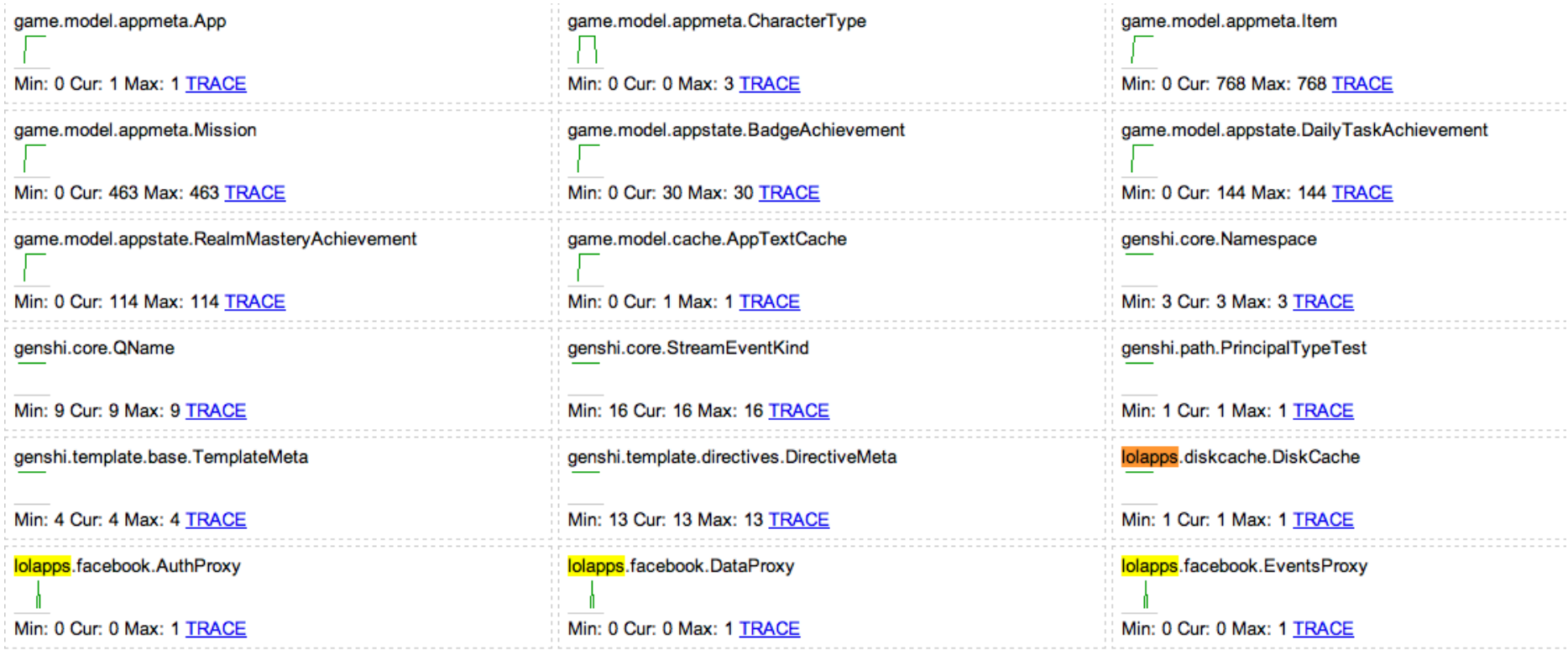
Profiling - Repoze

```
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   168734  126.897    0.001  126.897    0.001 {method 'recv' of '_socket.socket' objects}
    48690   32.525    0.001   47.363    0.001 {built-in method load}
     6483   12.685    0.002   12.685    0.002 {method 'query' of '_mysql.connection' objects}
   326993   11.434    0.000   11.437    0.000 /usr/lib/python2.5/site-packages/SQLAlchemy-0.4.4-py2.5.egg/sql
5614413/4958145    7.887    0.000   68.148    0.000 /usr/lib/python2.5/site-packages/SQLAlchemy-0.4.4-py2.5.eg
5581633/4925365    7.215    0.000   64.252    0.000 /usr/lib/python2.5/site-packages/SQLAlchemy-0.4.4-py2.5.eg
7460832/7374232    6.613    0.000    9.811    0.000 {getattr}
    30605    5.994    0.000   12.266    0.000 /usr/lib/python2.5/site-packages/SQLAlchemy-0.4.4-py2.5.egg/sql
  6670080    5.264    0.000    8.252    0.000 /usr/lib64/python2.5/hmac.py:9(<lambda>)
  1250613    4.565    0.000   14.245    0.000 /usr/lib/python2.5/site-packages/SQLAlchemy-0.4.4-py2.5.egg/sql
315570/312667    4.531    0.000  134.056    0.000 /usr/lib/python2.5/site-packages/SQLAlchemy-0.4.4-py2.5.egg/
465637/465486    3.763    0.000    4.816    0.000 {built-in method sub}
```

- **ncalls**: number of calls
- **tottime**: time spent in given function and excluding the time spent in sub-functions
- **percall**: tottime / ncalls
- **cumtime**: total time spent in this and all sub-functions.
- **percall**: cumtime / ncalls
- **filename:lineno**(function): function info.

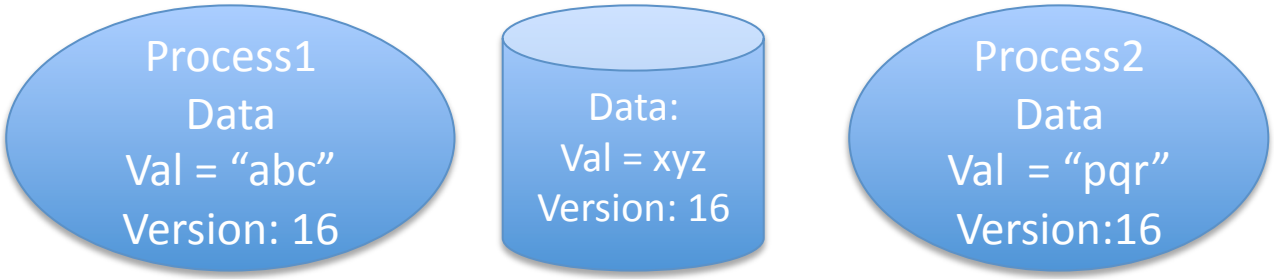
Profiling - Dozer

```
from dozer import Dozer, Logview
app = Logview(app, config)
app = Dozer(app)
```



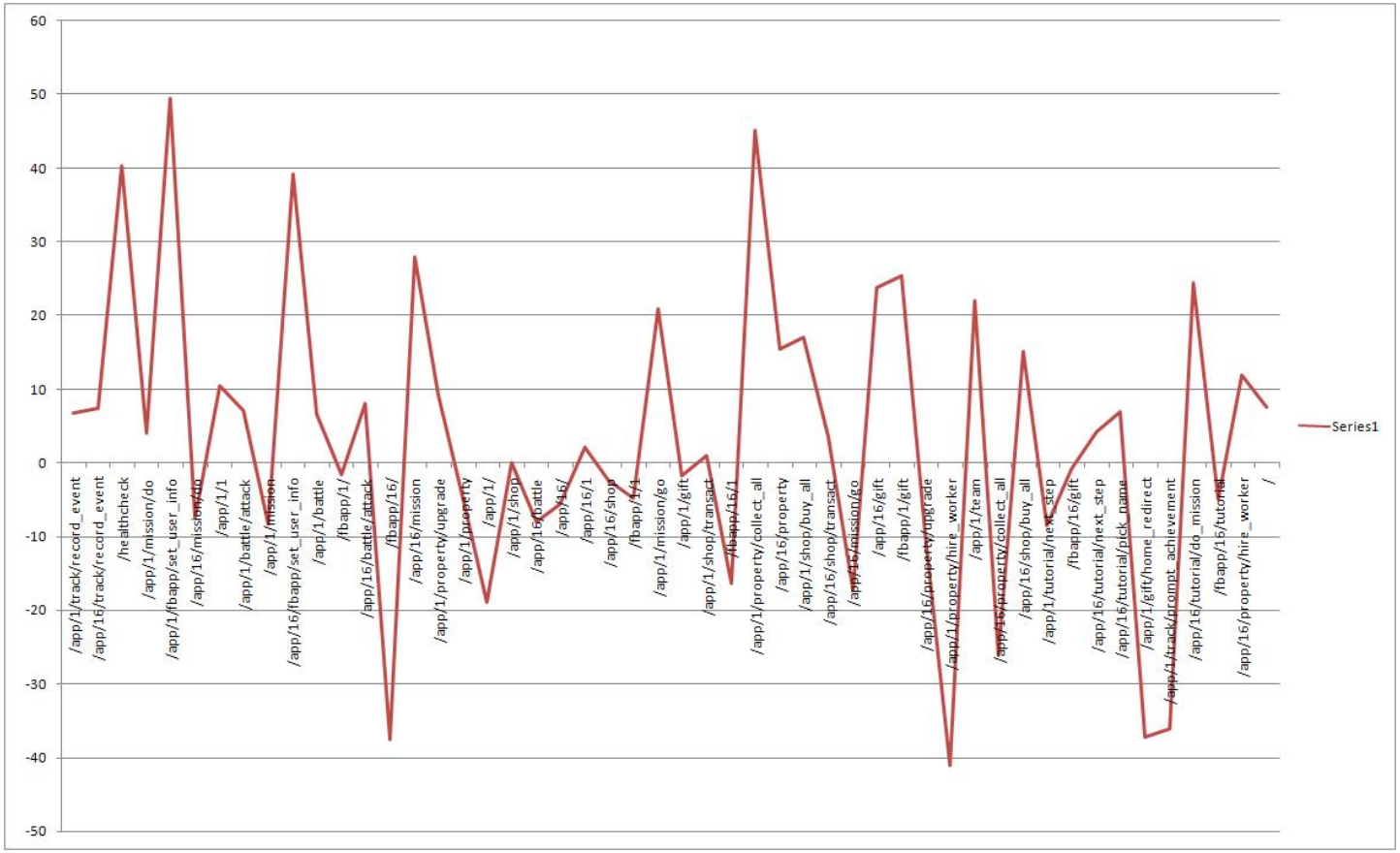
Database

- Optimistic vs. Pessimistic locking
  - version_id



- *Update table set data = xyz where version = 16.*
- SQLAlchemy (echo, echo_pool and logger)
- Remove/rollback

Paster vs. Tornado



Tornado

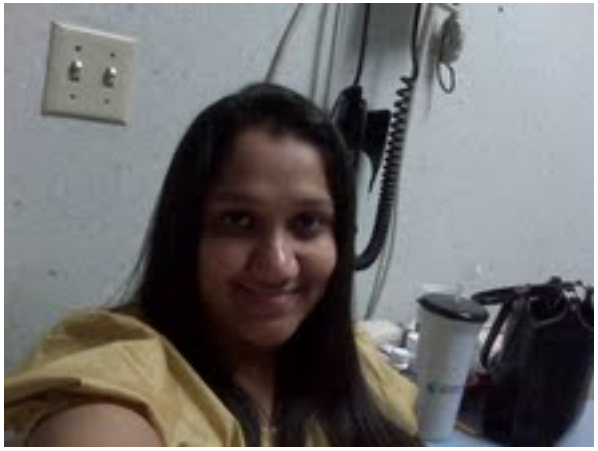
- Used over WSGI
- CPU and Memory usage down
- Didn’t do well for high response size
- Appropriate for asynchronous / realtime

Acknowledgements

- Lolapps team

  Brian Rue, AJ Cantu, Fred Blau, Cory Virok, Justin Rosenthal, Joseph Estrada, Allen Cheung, Vivek Tatineni, Jason Kim, Vikram Adukia

- Family



Questions