Performance Tunning de Clusters Plone

Performance Tunning de Clusters Plone

Fabiano Weimar dos Santos [Xiru]

xiru@xiru.org

***II PyCon Brasil** - 2006*

Interlegis - Brasília - DF

O que iremos ver?

- O Problema
  - Uma breve explanação sobre os problemas relacionados a otimização de performance de aplicações web dinâmicas
- Medindo Performance
  - Considerações sobre métricas e as armadilhas encodidas por nas ferramentas de benchmark
- Benchmarks
  - Zope, Plone, Plone + CacheFu + Squid e Apache
- Dicas de Otimização de Performance

O que não iremos ver?

- Como configurar o Plone em cluster
- Detalhes sobre as otimizações do CacheFu

O Problema

- Todo site dinâmico é lento, principalmente com grandes quantidades de acessos.
- Geralmente não é possível prever a quantidade de acessos de um site.
- Toda aplicação está sujeita a picos de acessos em situações imprevisíveis, mas a infra- estrutura de um site deve estar preparada para catástrofes!

Algumas situações imprevisíveis...

O Congresso da Vergonha Nacional

Mensalão, sanguessugas, dinheiro na cueca, dança da pizza... O parlamento vive a mais grave crise moral desde a redemocratização do país. É a instituição em que o brasileiro menos confia. O descrédito impera até entre parlamentares. "É o pior Congresso da história", desabada o presidente do Conselho de Ética da Câmara, Ricardo Izar.

(capa do Correio Braziliense, 21 de maio)

Crise em São Paulo aumenta audiência de portais da internet em 30%, na média

São Paulo - Mesmo com queda de usuários, sites como iG, UOL e Globo.com experimentam picos de tráfego nesta segunda-feira graças aos ataques do PCC.

(IDGNow, 16 de maio de 2006)

Medindo Performance

- **Tarefa não trivial.**
- Ferramentas dificilmente simulam o comportamento real das aplicações.
- **Apache Benchmark** - ab – costuma ser a forma mais simples de medir a performance de um site.
  - Não testa o carregamento de css, javascript, imagens e não leva em consideração o cache que um browser faria, por exemplo.
- Há muitas outras ferramentas de Benchmark
  - http://www.opensourcetesting.org/performance.php

Zope 2.8.6 Padrão

```
$ ab -n 10 http://localhost:9999/
Concurrency Level:      1
Time taken for tests:   0.309 seconds
Complete requests:      10
Failed requests:        0
Broken pipe errors:     0
Total transferred:      32560 bytes
HTML transferred:       30530 bytes
Requests per second:    32.36 [#/sec] (mean)
Time per request:       30.90 [ms] (mean)
Time per request:       30.90 [ms] (mean, across all concurrent
requests)
Transfer rate:          105.37 [Kbytes/sec] received
```

Plone 2.1 Padrão

```
$ ab -n 10 http://localhost:9999/pyconbrasil/
Concurrency Level:      1
Time taken for tests:   5.354 seconds
Complete requests:      10
Failed requests:        0
Broken pipe errors:     0
Total transferred:      242910 bytes
HTML transferred:       240120 bytes
Requests per second:    1.87 [#/sec] (mean)
Time per request:       535.40 [ms] (mean)
Time per request:       535.40 [ms] (mean, across all concurrent
requests)
Transfer rate:          45.37 [Kbytes/sec] received
```

Plone 2.1
(sem PlacelessTranslationService)

```
$ ab -n 10 http://localhost:9999/pyconbrasil/
Concurrency Level:      1
Time taken for tests:   3.729 seconds
Complete requests:      10
Failed requests:        0
Broken pipe errors:     0
Total transferred:      242980 bytes
HTML transferred:       240190 bytes
Requests per second:    2.68 [#/sec] (mean)
Time per request:       372.90 [ms] (mean)
Time per request:       372.90 [ms] (mean, across all concurrent
requests)
Transfer rate:          65.16 [Kbytes/sec] received
```

Plone 2.1 (com CacheFu)

```
$ ab -n 10 http://localhost:9999/pyconbrasil/
Concurrency Level:      1
Time taken for tests:   0.385 seconds
Complete requests:      10
Failed requests:        0
Broken pipe errors:     0
Total transferred:      245320 bytes
HTML transferred:       240280 bytes
Requests per second:    25.97 [#/sec] (mean)
Time per request:       38.50 [ms] (mean)
Time per request:       38.50 [ms] (mean, across all concurrent
requests)
Transfer rate:          637.19 [Kbytes/sec] received
```

Está começando a melhorar...
mas o que aconteceria quando tivessemos alguns usuários concorrentes?

Zope 2.8.6 Padrão (mais requests!)

```
$ ab -n 1000 -c 10 http://localhost:9999/
Concurrency Level:      10
Time taken for tests:   15.728 seconds
Complete requests:      1000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      3259256 bytes
HTML transferred:       3056053 bytes
Requests per second:    63.58 [#/sec] (mean)
Time per request:       157.28 [ms] (mean)
Time per request:       15.73 [ms] (mean, across all concurrent
requests)
Transfer rate:          207.23 [Kbytes/sec] received
```

Zope 2.8.6 Padrão (RAM Cache Manager)

```
$ ab -n 1000 -c 10 http://localhost:9999/
Concurrency Level:      10
Time taken for tests:   9.132 seconds
Complete requests:      1000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      3256000 bytes
HTML transferred:       3053000 bytes
Requests per second:    109.51 [#/sec] (mean)
Time per request:       91.32 [ms] (mean)
Time per request:       9.13 [ms] (mean, across all concurrent
requests)
Transfer rate:          356.55 [Kbytes/sec] received
```

Plone 2.1
(com CacheFu e mais requests!)

```
$ ab -n 1000 -c 10 http://localhost:9999/pyconbrasil/
Concurrency Level:      10
Time taken for tests:   28.717 seconds
Complete requests:      1000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      24532000 bytes
HTML transferred:       24028000 bytes
Requests per second:    34.82 [#/sec] (mean)
Time per request:       287.17 [ms] (mean)
Time per request:       28.72 [ms] (mean, across all concurrent
requests)
Transfer rate:          854.27 [Kbytes/sec] received
```

Conclusões iniciais

- Processando ***34,8 requests/s***, uma intranet corporativa processaria, durante as 8 horas de expediente da empresa, aproximadamente ***1 milhão de hits.***
- Apesar de parecer que já obtemos uma performance satisfatória, na prática isso provavelmente não seria o suficiente.
- Vejamos então o que conseguimos usando um proxy/ cache Squid...

CacheFu/squid/squid.cfg

```
[python]
binary: /usr/bin/python
[squid]
binary: /opt/squid/sbin/squid
user: www
config_dir: /opt/squid/etc
log_dir: /opt/squid/var/logs
cache_dir: /opt/squid/var/cache
cache_size_mb: 1000
direct: True
port: 3128
admin_email: xiru@xiru.org
[supported-protocols]
http: 80
[accelerated-hosts]
pyconbrasil.xiru.intranet: 127.0.0.1:9999/pyconbrasil
```

Gerando configurações do Squid

```
$ export PYTHONPATH=/opt/zope/zope-
2.8.6/lib/python
$ python makeconfig
Configuration file [squid.cfg]:
Template directory [templates]:
Output directory [output]:
Generating files for standalone squid
Generating output/deploy
Generating output/iRedirector.py
Generating output/purge_squid
Generating output/squid.conf
Generating output/squidAcl.py
Generating output/squidRewriteRules.py
Generating output/timing.py
```

Instalando configurações do Squid

```
$ cd output
$ sudo ./deploy
Copying config / helper files to /opt/squid/etc
Giving www ownership of its config and helper
files
Giving www read and execute access to its config
/ helper files
Giving www write access to /opt/squid/var/logs
and /opt/squid/var/cache
Remember to initialize the squid cache (squid -z)
```

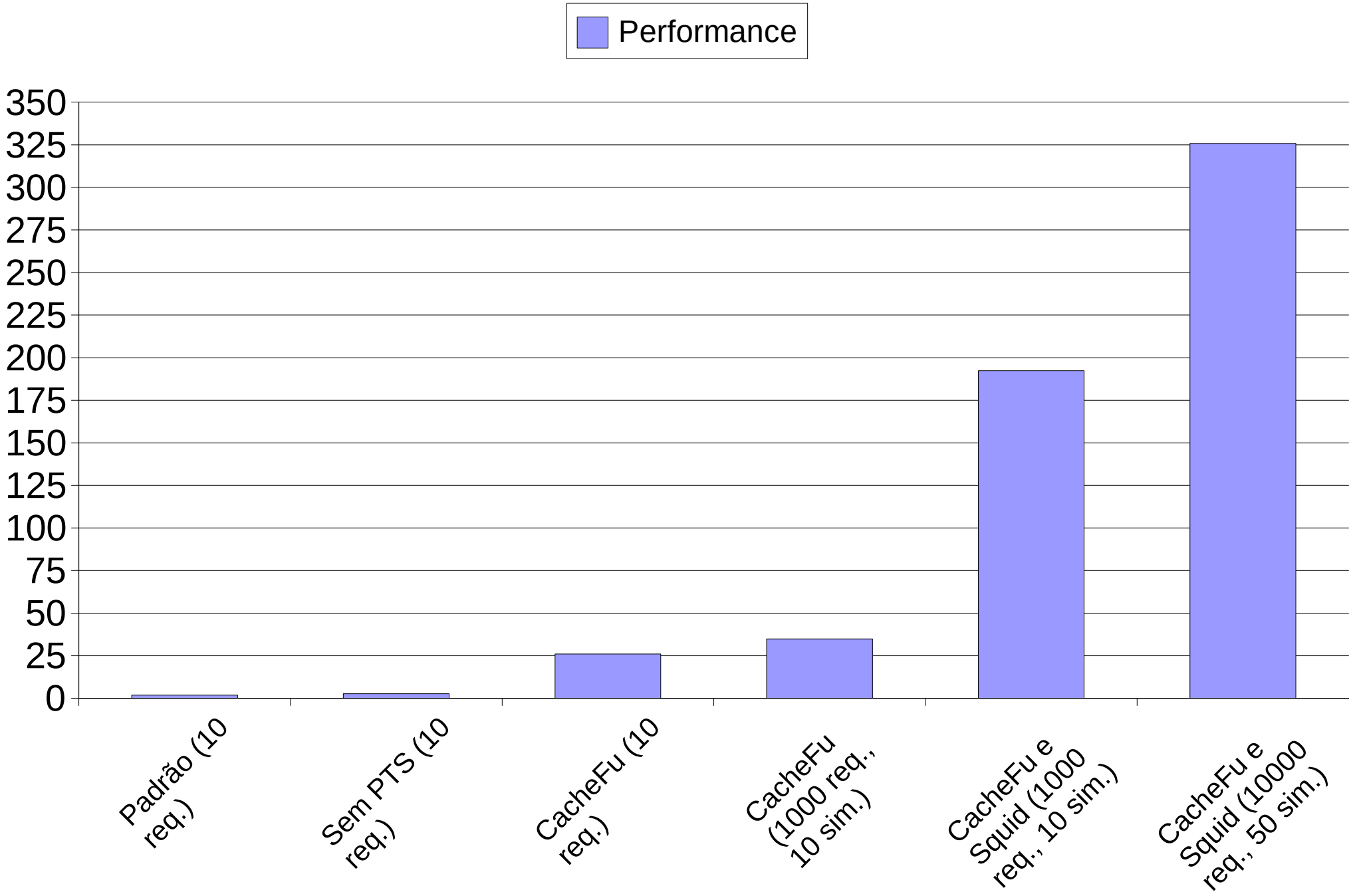
Plone 2.1
(com CacheFu e Squid)

```
$ ab -n 1000 -c 10 http://pyconbrasil.xiru.intranet/
Concurrency Level:      10
Time taken for tests:   5.198368 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      24615782 bytes
HTML transferred:       23985524 bytes
Requests per second:    192.37 [#/sec] (mean)
Time per request:       51.984 [ms] (mean)
Time per request:       5.198 [ms] (mean, across all concurrent
requests)
Transfer rate:          4624.14 [Kbytes/sec] received
```

Plone 2.1 (com CacheFu e Squid e mais requests ainda!)

```
$ ab -n 10000 -c 50 http://pyconbrasil.xiru.intranet/
Concurrency Level:      50
Time taken for tests:   30.693625 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      246174925 bytes
HTML transferred:       239867942 bytes
Requests per second:    325.80 [#/sec] (mean)
Time per request:       153.468 [ms] (mean)
Time per request:       3.069 [ms] (mean, across all concurrent
requests)
Transfer rate:          7832.41 [Kbytes/sec] received
```

Benchmark CacheFu



Mais conclusões

- Processando ***325,8 requests/s***, a mesma intranet do exemplo anterior processaria aproximadamente ***9,3 milhões de hits.***
- Servidores de médio porte conseguem processar (com folga) cerca de ***800 requests/s***. Servidores mais rápidos e bem configurados podem chegar a processar mais de ***3000 requests/s***.
- Squid é rápido! Mas o que acontece ao compararmos essa arquitetura com um servidor Apache, servindo conteúdo estático?

Apache Padrão

```
$ ab -n 10000 -c 50 http://localhost/
Concurrency Level:      50
Time taken for tests:   78.525 seconds
Complete requests:      10000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      18752360 bytes
HTML transferred:       14600768 bytes
Requests per second:    127.35 [#/sec] (mean)
Time per request:       392.62 [ms] (mean)
Time per request:       7.85 [ms] (mean, across all concurrent
requests)
Transfer rate:          238.81 [Kbytes/sec] received
```

O Apache padrão está aproximadamente 2,5 vezes mais lento que o Plone!?

Tem algo errado aqui...

Apache Padrão ("contornando" o MultiViews)

```
$ ab -n 10000 -c 50 http://localhost/index.html.pt-br
Concurrency Level:      50
Time taken for tests:   12.694 seconds
Complete requests:      10000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      23177742 bytes
HTML transferred:       20427330 bytes
Requests per second:    787.77 [#/sec] (mean)
Time per request:       63.47 [ms] (mean)
Time per request:       1.27 [ms] (mean, across all concurrent
requests)
Transfer rate:          1825.88 [Kbytes/sec] received
```

Agora o Apache está aproximadamente 2,5 vezes mais rápido que o Plone

(rodando com Squid e CacheFu).

Vejamos então alguns ajustes nas configurações no Apache para implementar uma política de cache similar ao CacheFu...

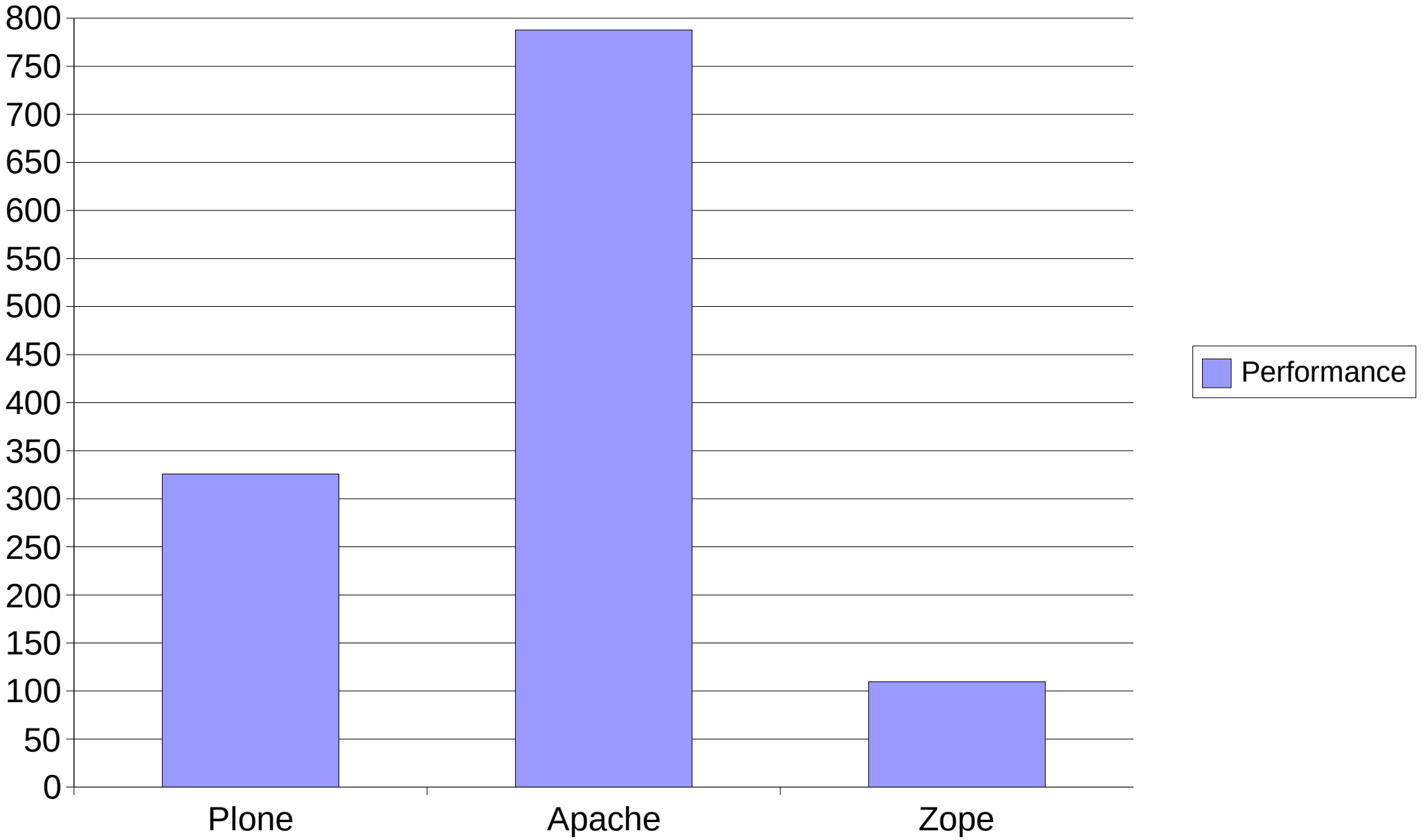
mod_expires

```
ExpiresActive On
ExpiresDefault "access plus 5 minutes"
ExpiresByType image/gif "access plus 1 day"
ExpiresByType image/jpeg "access plus 1 day"
ExpiresByType image/png "access plus 1 day"
ExpiresByType text/css "access plus 1 day"
ExpiresByType application/x-javascript "access plus 1 day"
```

mod_proxy

```
ProxyRequests On
ProxyVia On
<Directory proxy:*>
 Order deny,allow
 Deny from all
 Allow from localhost
</Directory>
CacheRoot "/private/var/run/proxy"
CacheSize 5
CacheGcInterval 4
CacheMaxExpire 24
CacheLastModifiedFactor 0.1
CacheDefaultExpire 1
```

Melhores Performances



Mais conclusões

- Apesar do Apache parecer ser o mais rápido, convém lembrar que estamos comparando coisas muito diferentes.
- É importante perceber que o “Plone + CacheFu + Squid” não perdem muito para o Apache (bem... perdem, mas não tanto quanto o Plone padrão :-)

Mais conclusões

- Com um Performance Tunning adequado, um cluster Plone pode rodar tão rápido quanto um servidor Apache.
- O servidor HTTP do Zope é lento, mesmo configurado para fazer cache em memória RAM.

Dicas

- Cuidado para não “acordar” muitos objetos do ZODB.
  - Evite usar o método getObject do brain quando fizer uma busca no portal_catalog.
  - Não use o método objectValues, objectIds. Use o ExtendedPathIndex ou o NavtreeIndexNG.
  - Evite usar o método listFolderContents (Plone 2.0).
- Ajuste do tamanho do cache do ZODB.
- Evite usar aquisição.
- Evite usar redirects.

Dicas

- Não tente advinhar porque seu site está lento: use um profiler:

  ZopeProfiler

  - http://www.dieter.handshake.de/pyprojects/zope

  CallProfiler

  - http://zope.org/Members/richard/CallProfiler

  PTProfiler

  - http://zope.org/Members/guido_w/PTProfiler

Dicas

- Combine múltiplos arquivos javascript e css em um único arquivo (ResourceRegistry).
- Entenda como os cabeçalhos HTTP funcionam.
- Depure os cabeçalhos HTTP de sua aplicação usando a extensão Live HTTP Headers do Firefox.
  - http://livehttpheaders.mozdev.org
- Faça cache no browser de javascript, css e imagens.
- Use NTP para sincronizar os relógios dos servidores.

Links

- Make Plone Go Fast
  - http://tinyurl.com/hcyl3
- CacheFu
  - http://plone.org/products/cachefu
- Cacheability Engine Query
  - http://www.web-caching.com/cacheability.html
- Caching Tutorial for Web Authors and Webmasters
  - http://www.web-caching.com/mnot_tutorial/

Links

- The Definitive Guide to Plone – Chapter 14 - Administering and Scaling Plone
  - http://docs.neuroinf.de/PloneBook/ch14.rst