

Plone Performance And Caching

ZPUG/DC Meeting

Author: Joel Burton <joel@joelburton.com>
Copyright: Copyright 2006 Joel Burton
Covering: Plone 2.1 or newer, CacheFu 1.0 beta

Contents

1	Plone Performance and Caching	3
1.1	Introduction	4
1.1.1	Plone Does A Lot	4
1.1.2	Plone Does A Lot II	4
1.1.3	This Is Good	4
1.1.4	This Can Be Bad	5
1.1.5	Rephrasing	5
1.1.6	Benchmarking In 1 Minute	5
1.1.7	Using ab	5
1.1.8	ab Results	6
1.1.9	ab Shortcomings	6
1.1.10	Holistic Benchmarking	6
1.2	ZEO	7
1.2.1	ZEO Overview	7
1.2.2	ZEO Speed Improvements	7
1.2.3	Creating a ZEO Server	7
1.2.4	Using New ZEO Server	8
1.3	Caching	9
1.3.1	Where Things Get Cached	9
1.3.2	Where Things Get Cached	9
1.3.3	Where Things Get Cached	9
1.3.4	Where Things Get Cached	10
1.3.5	Disk Caching by HD	10
1.3.6	ZODB Caching	10
1.3.7	ZEO Caching	10
1.3.8	RAMCache Manager	11
1.3.9	RAMCache Unconditionally	11
1.3.10	RAMCache For Limited Time	11
1.3.11	RAMCache On User + Time	12
1.3.12	ZSQL Method Caching	12

1.3.13	Accel HTTP Cache	12
1.3.14	Strategies for Caching	13
1.3.15	Caching Risks	13
1.3.16	CacheFu	13
1.3.17	CacheFu Goal	13
1.3.18	Getting CacheFu	14
1.3.19	CacheFu Concepts	14
1.3.20	Header Policies	14
1.3.21	Rulesets	15
1.3.22	Default CacheFu Rulesets	15
1.3.23	The Sibling Case	15
1.3.24	Other Tricks CacheFu Does	16
1.3.25	Proxy Server	16
1.3.26	Apache	16
1.3.27	Squid	16
1.3.28	Apache -> Squid -> Zope	17
1.3.29	Squid Is New To Us	17
1.3.30	Proxy Server Setup	17
1.3.31	In-Memory Caching By CacheFu	17
1.3.32	Speed of Caches	18
1.3.33	Strategies	18
1.3.34	Strategies II	18
1.3.35	Strategies III	18
1.3.36	Strategies IV	19
1.3.37	Strategies V	19
1.3.38	Advanced Strategy	19
1.3.39	Getting More Information	19
1.3.40	How You Can Help	20

2 Footnotes**21**

1 Plone Performance and Caching

1.1 Introduction

1.1.1 Plone Does A Lot

- Checks for authentication
 - HTTP is stateless
 - Find the object in the DB
 - Check that user has rights to object
 - And to everything (skins, etc.)
-

1.1.2 Plone Does A Lot II

- “Skin” object
 - Perform any logic (portlets, etc.)
 - Add custom info about user
 - Decided which JS/CSS to show
 - Create dynamic navigation elements
-

1.1.3 This Is Good

- Everything is fresh
- Everything is security-aware
- High degree of personalization

1.1.4 This Can Be Bad

- High stress on server
-

1.1.5 Rephrasing

- It's not true that "Plone is slow"
 - It's quite efficient
 - Fair to say "It's doing more than I need"
 - We can trade off freshness for speed
-

1.1.6 Benchmarking In 1 Minute

- ab, Apache Benchmark
 - From Apache Foundation
 - Included in most Linux distributions
-

1.1.7 Using ab

```
ab -n 50 -c 2 http://yoursite/
```

- n: number of requests
- c: concurrency
- Don't forget the trailing slash!

1.1.8 ab Results

```
HTML transferred:      152650 bytes
Requests per second:   64.35 [#/sec] (mean)
Time per request:      15.540 [ms] (mean)
Transfer rate:         203.35 [Kbytes/sec] received
```

1.1.9 ab Shortcomings

- Doesn't act like a real user
 - Requesting the same page 50 times?!
 - Measures just HTTP page
 - * Not CSS/javascript
-

1.1.10 Holistic Benchmarking

- Specialized tools & web applications
- In a pinch, Selenium can work fine

1.2 ZEO

1.2.1 ZEO Overview

- ZEO
 - Object server (ORB)
 - Zope server is ZEO client
 - Allows multiple Zope clients
 - Use ZEO all the time
-

1.2.2 ZEO Speed Improvements

- Page production takes same time
 - But often starts sooner
 - Also provides failover capability
-

1.2.3 Creating a ZEO Server

- Script to create new ZEO server:

```
$SOFTWARE_HOME/bin/mkzeoinstance.py home [port]
```

 - *port* is port ZEO server runs on
- Move existing `Data.fs` from `var` to `zoo/var`

1.2.4 Using New ZEO Server

- Edit `$INSTANCE_HOME/etc/zope.conf`
 - Comment out first `zope_db` main; uncomment second
 - Port should be set to ZEO server

1.3 Caching

1.3.1 Where Things Get Cached

- Low-level than Zope
 - Completely transparent
 - * ie, no side effects, Zope is unaware of
-

1.3.2 Where Things Get Cached

- Inside Zope
 - Least dramatic benefits
 - Easy to recognize changes + update
-

1.3.3 Where Things Get Cached

- On a proxy server
 - Excellent benefits
 - Changes can be often be handled
 - Great for lots of different users

1.3.4 Where Things Get Cached

- In browser
 - Can be best benefits
 - Hard to notify changes
 - Tremendous benefit for same users coming back
-

1.3.5 Disk Caching by HD

- Completely transparent to Zope
-

1.3.6 ZODB Caching

- Retrieval of raw object is cached
 - Prior to skinning / work
 - Completely transparent to your app
 - Settings in `zope.conf`
 - May be worthwhile to change
-

1.3.7 ZEO Caching

- Retrieval of object over ZEO is cached
 - Prior to skinning / work
- Completely transparent to app
- Settings in `zeo.conf`

1.3.8 RAMCache Manager

- Generalized cache for result-of-rendering
 - Commonly PythonScripts, PageTemplates
-

1.3.9 RAMCache Unconditionally

```
Script (Python) "mult.py"  
## parameters = x, y  
  
return x * y
```

- Can cache forever on the parameters
 - Parameters always considered in cache
-

1.3.10 RAMCache For Limited Time

```
Script (Python) "getWeather.py"  
## parameters = zipcode  
  
return context.getWeatherUsingSomeWebService(zipcode)
```

- Results should be stale after some period

1.3.11 RAMCache On User + Time

```
Script (Python) "getArticles.py"
```

```
return context.portal_catalog(  
    Type='Article')
```

- Need to separate my results from yours
 - Cache using AUTHENTICATED_USER as key
-

1.3.12 ZSQL Method Caching

- For each ZSQL Method, can cache
 - Cached for search keys
 - DBs are often slow, can be helpful
 - Don't cache INSERT/UPDATE, etc.
 - Though this might be useful for logging apps
-

1.3.13 Accel HTTP Cache

- Caches entire HTTP request (eg, web page)
 - Actual cache in *browser* or *proxy*
 - Not each to invalidate
 - Includes personalization (eg, username)
- Useful for images, CSS
 - Some use for static-ish pages (home pages, etc)

1.3.14 Strategies for Caching

- Microcaching is very helpful
 - 10 sec - 10 mins not likely to be noticed
 - Can reduce work by 1000x
-

1.3.15 Caching Risks

- Staleness
 - Content has changed / been deleted
 - Inappropriate delivery
 - ie, showing something to user they shouldn't see
-

1.3.16 CacheFu

- Combination of useful caching strategies
 - Merged and enhanced by Geoff Davis
 - * Plone Foundation board member, TriZPUG member
 - * Math PhD with a penchant for physics
 - * Good man to get barbecue with
-

1.3.17 CacheFu Goal

- Maximum Caching with Minimum Side Effects
 - But lots of knobs for making more tradeoffs

1.3.18 Getting CacheFu

- Download from Plone Software Center at plone.org
 - Several products, install them all
 - Beta software
 - You’ve been warned
-

1.3.19 CacheFu Concepts

- Caches some things in Zope
 - When we need fast invalidation, user-specific cache
 - Caches some stuff in proxy server
 - When we need invalidation but not user-specific
 - Caches some stuff in browser
 - When it can live forever and a day
-

1.3.20 Header Policies

- Controls actual HTTP cache settings emitted
- Geek-land
 - Few moving parts for mortals
- Have names like “Cache in Squid for 24h”, etc.

1.3.21 Rulesets

- Chain of rules that are attempted
 - In order
 - Try to find right policy (in-Zope or output header)
-

1.3.22 Default CacheFu Rulesets

- Generally, conservative
 - Everything is invalidated on change
 - Auth users never use each others cached data
-

1.3.23 The Sibling Case

- Only affects anonymous users
- When editing something (newsitems/project-a)
 - Display of that is 100% up-to-date
 - However, navigation may show sibling items in folder
 - * These titles/URLs can be stale
- Times out after an hour

1.3.24 Other Tricks CacheFu Does

- Can compress (gzip) output
 - Can remove extraneous whitespace
 - Takes less time to travel over the wire
 - I turn these off
 - I worry about buggy browsers
-

1.3.25 Proxy Server

- Technically, this is “reverse proxy”
 - Squid, Apache, other caching apps
 - Can only do invalidation in Squid
-

1.3.26 Apache

- Excellent, world-class web server
 - Weak proxy server
 - Since we can’t invalidate, we don’t emit headers for things that might need it
-

1.3.27 Squid

- Excellent proxy server
- Not as featureful for generic web serving
- Can invalidate, so can cache and handle stale cases

1.3.28 Apache -> Squid -> Zope

- Apache handles PHP, other stuff
 - Squid caches Zope output
 - Everyone wins!
 - More complex to set up
-

1.3.29 Squid Is New To Us

- Very stable product, used in serious deployments
 - plone.org, lots of my deployments have used
 - Not widely used among Plone people
 - Sample squid configs and (new!) config wizard
 - Good (longish) book by O'Reilly
 - Only have to read the reverse-caching parts!
-

1.3.30 Proxy Server Setup

- Must tell CacheFu what proxy server, if any, is used
 - And tell it info about proxy server
-

1.3.31 In-Memory Caching By CacheFu

- Examines if content could be stale
 - Modification date, state of catalog, etc.
- Returns a not-changed header by Zope

1.3.32 Speed of Caches

- CacheFu In Memory: ~30-40x
 - Squid: ~5-10x CacheFu in Memory
 - Up to ~400x! plain Plone
-

1.3.33 Strategies

- Got servers?
 - Use ZEO
 - Handle requests sooner
-

1.3.34 Strategies II

- Got memory?
 - Increase ZODB and/or ZEO disk caches
-

1.3.35 Strategies III

- Got stable functions?
 - eg, tax calculator
 - Cache them forever with RAMCache

1.3.36 Strategies IV

- Got time-cachable functions?
 - eg, weather retrieval
 - Cache them for a while with RAMCache
-

1.3.37 Strategies V

- Use CacheFu to handle the rest
-

1.3.38 Advanced Strategy

- Use a “edit URL” different from the usual URL
 - Then for usual URL, treat members as CacheFu treats anonymous
 - For edit URL, treat as CacheFu treats members
 - May become a possible setting for CacheFu
-

1.3.39 Getting More Information

- CacheFu README.txt
- CacheFu “Audiences” documentation
 - Still in progress
- O’Reilly Squid book

1.3.40 How You Can Help

- Help Geoff support the product
 - He's getting drowned in Squid support
- Help with documentation
- Blog about its coolness
- Hire/Bribe Geoff: geoff@geoffdavis.net

2 Footnotes