Make Plone Fast!

Strategies and Tools for Faster Sites

Geoff Davis
geoff@geoffdavis.net

Plone Symposium, 2005

Overview

- High level talk
- Goals today:
  - Sketch strategies for speeding up your sites
  - Point to useful tools
- Will leave details to other references

How fast is your site?

- Simplest measurement: Apache benchmark (ab)
  - comes with Apache 2.0 distribution
  - simulates lots of users hitting a single page sequentially and / or simultaneously
  - measures pages served / second
- Limitations of ab
  - doesn't load associated images, css, js (matters a lot!)
  - doesn't know about browser caching, etc
- Better benchmarks feasible with Selenium??

Targets for Speedups

- 3 main areas (in order of decreasing importance):

    1) page rendering time in Zope
    2) Zope authentication and traversal
    3) network latency

General Strategies

- Cache static content in browsers using HTTP headers
  - helps: page rendering time, traversal time, latency
- Use a fast proxy cache to serve static content
  - helps: page rendering time, traversal time
- Load balancing
  - helps: page rendering time, traversal time (under load)
- Optimize your code
  - helps: page rendering time
- Cache intermediate code results
  - helps: page rendering time

New Ideas

- Smarter browser caching with ETag validation
  - helps: page rendering time, traversal time, latency
  - more widely applicable than other kinds of browser caching

Speed Strategy 1: Cache static content on browsers

- When users visit a site, content gets stored in their browser caches
- HTTP headers tell browsers how long to cache content
- On subsequent page visits, users see locally cached versions of content rather than hitting your site again
- Most useful for *static content* that is *viewed frequently* (images, css, js, etc)

HTTP headers

- Understand HTTP headers to do caching right
- Good tutorial at http://www.web-caching.com/mnot_tutorial/

HTTP header basics

- HTTP 1.0
  - Expires, Last-Modified headers:
    - browser will cache your content if expiration date is in future; may cache for some data types (images) if Last-Modified date is in the past
- HTTP 1.1
  - Cache-Control headers:
    - max-age=N: browser will cache your content for N seconds
      - preferable to Expires because doesn't require user's clock to be right
    - no-cache, must-revalidate: don't include these!
- Use both HTTP 1.0 and 1.1 headers

Setting HTTP headers

- AcceleratedCacheManager (ships with CMF) can set cache headers for skin elements
- CMF Caching Policy Manager (ships with CMF) also useful – more flexible than ACM
- See Definitive Guide to Plone, Chapter 14
  - http://docs.neuroinf.de/PloneBook/ch14.rst
  - Plone 2.1 takes care of caching for images, js, css
    - See HTTPCache in ZMI
    - Quick win: configure HTTPCache to increase time images/js/css are cached from 1 hour to, say, 1 week

HTTP Headers

- Plone explicitly tells browsers NOT to cache most content
- Anything using main_template has headers set in global_cache_headers.pt (see portal_skins/plone_templates)
- You may wish to override default headers
  - customize global_cache_headers (affects *all* templates)
  - call request.RESPONSE.setHeader in body of template (overrides previous header, affects only template in question)

Limitations of browser caching

- Effective only if content is accessed multiple times
  - Great for images, css, js that appears on every page
  - Less helpful for content
- Users may see stale content
  - No way to tell users that their content is out of date
  - With more work can get around this – will discuss how later

Brief Aside: Resource Registries

- Very useful new feature in Plone 2.1
  - In ZMI, register your javascript and css files with portal_javascripts and portal_css, respectively
  - Be sure to click Save button when you are done
  - No longer need to include js, css separately in your files

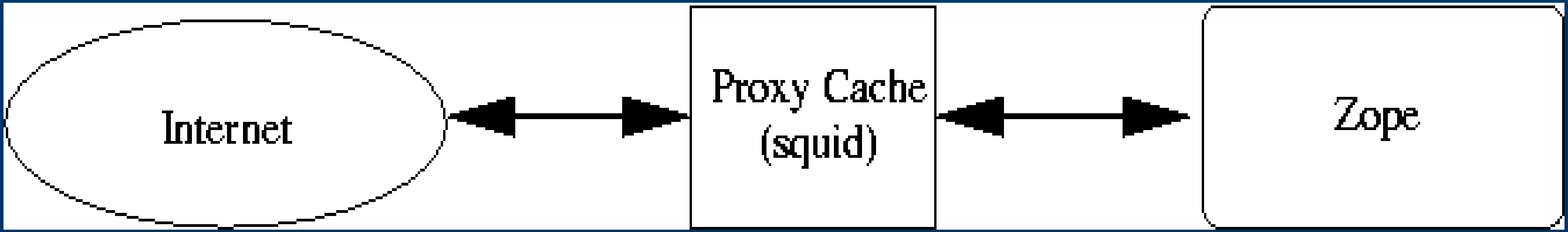
Aside: Resource Registries

- Why is this useful?

- All js (or css) files get collapsed into a single file
  - Reduces number of connections browser must make, reduces network overhead
- File is renamed every time you press Save
  - Lets you set very long cache times without worrying about stale content on client side

Speed Strategy 2: Proxy Caching

- Idea: put a fast but dumb proxy cache in front of Zope
- Proxy cache serves up static content, keeps load off Zope
- Zope can tell proxy cache when content expires so you don't serve up stale content



Proxy Caches

- Squid
  - free, open source; works both on Linux and Windows (via cygwin)
  - http://www.squid-cache.org
  - Super fast (~1000 requests/second on mid-range box)

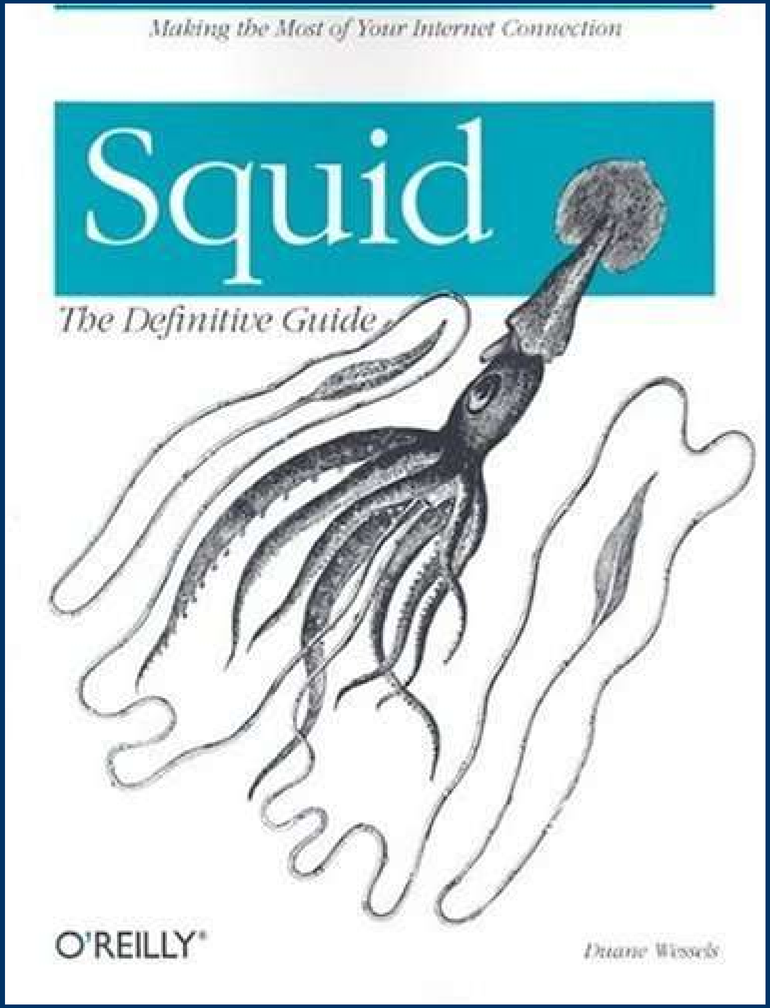
- Microsoft IIS + Enfold Enterprise Proxy
  - http://www.enfoldsystems.com/Products/EEP

Proxy Caches

- Apache + mod_proxy / mod_cache
  - Lots of documentation about using Apache for caching
  - ***Not recommended!***
- mod_cache is buggy:
  - intermittently serves up incomplete content
    - http://issues.apache.org/bugzilla/show.bug.cgi?id=32950
    - http://issues.apache.org/bugzilla/show.bug.cgi?id=33512
  - bad interaction with Plone's http compression
    - Compression enabled by default in 2.0.5; disabled in 2.1
    - Set in skins/plone_scripts/enableHTTPCompression.py

Using Squid

- Excellent documentation available
- (Only need to read a few chapters, not whole book)



Using Squid

- Easy to set up on Linux
  - pre-installed on Fedora Core
  - Only a handful of changes needed to default squid.conf
  - Good references:
    - http://www.zope.org/Members/JCLawrence/LocalhostSquidHOWTO/
    - http://www.zope.org/Members/htrd/howto/squid

Squid Benefits

- Even without any special setup, squid gives a sizable performance boost
- Squid caches your images, css, and js, and anything else that has HTTP headers that enable browser caching
- Squid serves up images, css, js instead of Zope
  - Squid is much faster than Zope
  - Lets Zope work on other things

Squid Strategy

- Have 2 URLs for site
  - one for users (cached by squid)
  - one for administrators (not cached)
- For example, plone.org and members.plone.org
- Reason: we don't want squid to serve authenticated user the anonymous version of a page and vice versa

Squid Strategy

- Use CMFSquidTool to keep cached content fresh
  - Hooks Zope's object cataloging.
  - When an object is recataloged, it is purged from squid.
  - Also works with IIS with Enfold Enterprise Proxy
  - Available from Enfold Systems
    - http://www.enfoldsystems.com/Products/Open/CMFSquidTool

Squid Strategy 2

- Alternative, if using cookie-based authentication (default for Plone)
- Much simpler, only need one URL
- Not tested! (But I am confident it will work)

Squid Strategy 2

- Idea from Wikipedia admins
- Set Vary: Cookies HTTP header
  - Tells squid to serve different pages depending on value of client's cookies
  - Result is that squid should distinguish pages from authenticated vs. non-authenticated users
  - Voila, no need for second URL
- Second benefit: can set different Cache-Control headers for anonymous vs. authenticated users

Speed Strategy 3: Load Balancing

- Zope Enterprise Objects let you do load balancing
  - ZEO server = essentially an object database
  - ZEO client executes your python scripts, serves up your content, etc
  - ZEO comes with Zope
- Set up multiple ZEO clients on multiple machines or multiple processors (single instance of Zope won't take much advantage of multiple processors)

Setting up ZEO

- You can transform a Zope site into a ZEO site using the mkzeoinstance.py script in ~Zope/bin
- Change a few lines in ~instance/etc/zope.conf and ~instance/etc/zeo.conf and you are good to go
- See Definitive Guide to Plone, Chapter 14
  - http://docs.neuroinf.de/PloneBook/ch14.rst

Squid + ZEO

- Main idea: give your proxy cache lots of places from which to get content it can't serve
- Squid can take care of load balancing
- Details:
  - http://www.infrae.com/products/silva/auxiliary_docs/archive/squid_notes
  - http://www.zope.org/Members/htrd/howto/squid
  - http://www.zope.org/Members/htrd/icp/intro

Speed Strategy 4: Optimize Your Code

- Don't guess about what to optimize – use a profiler
- Several available
  - Zope Profiler:
    - http://www.dieter.handshake.de/pyprojects/zope/
  - Call Profiler:
    - http://zope.org/Members/richard/CallProfiler
  - Page Template Profiler:
    - http://zope.org/Members/guido_w/PTProfiler
- Identify and focus on slowest macros / calls

SpeedPack

- Simplest speedup: install SpeedPack and psyco
  - Boosts page rendering speed by 10%-40%
  - Biggest wins on Windows
  - Works well with Plone 2.0.x and Zope 2.7.x
  - (Untested with Plone 2.1.x or Zope 2.8.x – there may be some issues – will be fixed eventually)
  - get SpeedPack from Plone SVN collective
    - http://svn.plone.org/svn/collective/SpeedPack/trunk/
  - get psyco from http://psyco.sourceforge.net/
  - Be sure to read the SpeedPack README.txt!!!

More Caching

- Suppose you find that a portlet is your bottleneck
  - Calendar portlet, for example, is pretty expensive
- How to fix?
- Idea: don't update calendar portlet every hit
  - Update, say, every hour
  - Cache the result in memory
  - Serve up the cached result
- Similar idea applies to other possible bottlenecks

RAMCacheManager

- RAMCacheManager is a standard Zope product
- Caches results of associated templates / scripts in memory
- Caveats:
  - Can't cache objects – only text, ints, floats, etc
  - Can't cache macros, only output of macros (portlet is a macro)
- How can we cache the calendar?

Trick: Caching Macro Output

- Idea:
  - create a template that renders the macro
  - output of template is snippet of HTML, i.e. a string
  - cache output of the template

Caching the Calendar

- Step 1: Create a template called cache_calendar.pt: <metal:macro use-macro=”here/portlet_calendar/macros/portlet” />
- Step 2: In the ZMI, add a RAMCacheManager to your site root
- Step 3: in the RAMCacheManager, set the REQUEST variables to AUTHENTICATED_USER, leave the others as defaults (this caches one calendar per user)

Caching the Calendar

- Step 4: Associate cache_calendar.pt with your new RAMCacheManager. Output of cache_calendar.pt will now be cached for 1 hour.
- Step 5: In your site's properties tab, replace here/portlet_calendar/macros/portlet with here/cache_calendar
- Voila!

- Use RAMCacheManager to cache output of slow scripts, etc.

New Idea: Smarter Browser Caching with Validation and ETags

- All the browser caching we have discussed so far has been time-based with no validation
- Browser checks age of cached page and returns cached page or hits server accordingly
- As a result, efficacy of this kind of caching is limited

- Browsers are smarter than this – we can do more

Validation and ETags

- With HTTP 1.1 we can force browsers to validate their cached content (must-revalidate directive)
- Browser checks with the server before serving up cached content - “Is what I have in my cache valid?”
- ETags are the key to smart validation
- Server sends out an ETag with a page
- To check freshness, a browser sends the Etag of the cached page and asks if it's current

ETags

- If page is stale, server sends back Status 200 plus the new page
- If page is still good, server sends back Status 304 and an *empty* page
- Validation is cheap and fast. No need for server to
  - render the full page
  - send the page over the network

ETags

- Main idea for implementation:
  - Have your pages supply an ETag header
  - ETag is an arbitrary string. Make sure it contains enough info to tell if a page is fresh, e.g.
    - a time stamp, the authenticated user, etc
  - Before rendering a page, check for the request header If-None-Match -- this is a browser sending an ETag for your inspection
  - If the ETag in the If-None-Match header matches the current ETag, send a 304 status header and stop.

ETagCacheManager

- Proof of concept of ETag validation idea
- Associate it with a page template (e.g. document_view)
- Takes care of ETag generation and checking
- As a bonus, it includes a fallback RAMCache

- It's in the collective – try it out! Note: alpha code
  - https://svn.plone.org/svn/collective/ETagCacheManager/