DRUPAL CACHING AND OPTIMIZATION STRATEGIES

Melbourne Drupal Mini-Conference
February 3rd, 2007
Who am I?

✶ Simon Roberts
✶ IT Consultant, Taniwha Solutions
✶ Using PHP since ~1996
✶ Using Drupal since 2005
   http://drupal.org/user/33667
✶ Co-maintainer of memcache module
Contents

⋆ Why optimize?
  ⋆ better user experience from faster pages
  ⋆ support more concurrent users
  ⋆ require less hardware
  ⋆ (maybe) offload network traffic
⋆ Performance != Scalability (though they’re related)
(Almost) Not Covered!

✱ There’s so many things that contribute to system performance:

✱ physical hardware: connectivity, network hardware, CPU(s), memory, IO, etc

✱ operating-system parameters

✱ database configuration

✱ web-server configuration

✱ PHP configuration
Briefly... Unix Tools

- **top**: shows current memory+CPU for current processes
- **vmstat**: shows realtime memory/swap/io/CPU statistics
- **iostat**: shows which devices are getting IO
- **mpstat**: slightly more info about CPU than vmstat
- **http://2bits.com/articles/tools-for-performance-tuning-and-optimization.html**
Briefly... Firebug Plugin

* Awesome tool for viewing end-user performance, CSS, Javascript, HTML etc

http://www.getfirebug.com/
Briefly… yslow

Plugin for firebug. Makes performance recommendations (with explanations)


<table>
<thead>
<tr>
<th>Performance Grade: D (67)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D 1. Make fewer HTTP requests</td>
</tr>
<tr>
<td>F 2. Use a CDN</td>
</tr>
<tr>
<td>A 3. Add an Expires header</td>
</tr>
<tr>
<td>F 4. Gzip components</td>
</tr>
<tr>
<td>A 5. Put CSS at the top</td>
</tr>
<tr>
<td>B 6. Put JS at the bottom</td>
</tr>
<tr>
<td>A 7. Avoid CSS expressions</td>
</tr>
<tr>
<td>n/a 8. Make JS and CSS external</td>
</tr>
<tr>
<td>A 9. Reduce DNS lookups</td>
</tr>
<tr>
<td>A 10. Minify JS</td>
</tr>
<tr>
<td>A 11. Avoid redirects</td>
</tr>
<tr>
<td>A 12. Remove duplicate scripts</td>
</tr>
<tr>
<td>F 13. Configure ETags</td>
</tr>
</tbody>
</table>

**Inspect**  **Performance**  **State**  **Components**  **Tools**  **Help**

<table>
<thead>
<tr>
<th>Empty Cache</th>
<th>Primed Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>146.4K 1 HTML/text</td>
<td>146.4K 1 HTML/text</td>
</tr>
<tr>
<td>25.3K 3 JavaScript Files</td>
<td>25.3K 3 JavaScript Files</td>
</tr>
<tr>
<td>28.5K 9 Stylesheet Files</td>
<td>28.5K 9 Stylesheet Files</td>
</tr>
<tr>
<td>8.5K 8 CSS Images</td>
<td>8.5K 8 CSS Images</td>
</tr>
<tr>
<td>6.1K 2 Images</td>
<td>6.1K 2 Images</td>
</tr>
<tr>
<td>215.1K Total size</td>
<td>215.1K Total size</td>
</tr>
<tr>
<td>23 HTTP requests</td>
<td>23 HTTP requests</td>
</tr>
</tbody>
</table>

- **Type**: doc, js, js, js, css, css
- **URL**: http://localhost/performance/
  - js http://localhost/performance/misc/jquery.js
  - js http://localhost/performance/misc/drupal.js
  - js http://localhost/performance/sites/all/modules/devel/devel.js
  - css http://localhost/performance/sites/all/modules/devel/devel.css
- **Gzip RespTime**: 74, 60, 8, 11, 32, 33
- **Size (Ungzip)**: 146.4K, 19.3K '28f22c-4b8', 5.6K '28f221-162', 0.3K '28f343-134', 0.06K '28f340-41', 0.5K '28f257-240'
- **ET**: 
Briefly... Apache Bench


* Can test performance + scalability

* HowTo: Benchmark Drupal Code  
  [http://drupal.org/node/79237](http://drupal.org/node/79237)
Web Server using excessive CPU?

- Remove unneeded Apache modules & PHP extensions

- PHP opcode cache (APC, eAccelerator)

- Use a lighter webserver for static content

- Cache within Drupal (see later)

- More webservers, load balancing (sorry)
Briefly...

* Web Server running out of memory?
  * Remove unneeded Apache modules & PHP extensions (no, really)
  * Reduce MaxClients, Reduce KeepAlive
  * Consider PHP memory_limit?

* See http://httpd.apache.org/docs/2.2/misc/perf-tuning.html

* Use devel module to check on Drupal memory usage

* Swap = Death of Performance; Add RAM!
Briefly...

- Database running out of CPU?
  - Query caching (easy win)
  - Optimize expensive queries
  - Consider increasing DB memory, to allow more in-memory caching
  - Cache on the DB client (ie: PHP/Drupal - see later)
Briefly...

- Excessive disk IO? (see vmstat)
- Web server logging? disable or log to separate server
- Slow-query log? disable
- DB activity, temporary tables? more memory
- Disable per-directory web server settings
Briefly...

∙ Too much network bandwidth used?
  ∙ mod_gzip or mod_deflate (trade-off CPU vs bandwidth)
  ∙ check HTTP content expiry (mod_expires) - Drupal5 already has this in .htaccess
  ∙ optimize HTML/CSS/JS and graphics for size
  ∙ disable DNS resolution on webserver
  ∙ consider CDN for large content
Simple Fix #1

- Administer > Site Configuration > Performance : Aggregate and Compress CSS files

- Reduce number of CSS files by creating a file containing (most) CSS files “compressed” and concatenated together

- Drupal 6 also does this for javascript files! Cool.
And now, what I was supposed to talk about :)

- Drupal-specific techniques:
  - Standard Drupal Options
  - Application Caching
  - Modules
  - Code
  - Memcache
Live Tuning

* In this presentation we’re going to actually perform each of our strategies on a live apache server, and see how it works!!

* For the sake of simplicity, we will only be using anonymous users

* Logged in users don’t benefit from Drupal page-caching, so need special attention!

* Of course, your environment will be different, and perform differently
Let’s establish a baseline per 
http://buytaert.net/drupal-webserver-configurations-compared

* install, login, change password
* enable forum, blog, book (more content types)
* enable path, pathauto, token module (to generate paths)
* enable Recent Comments, Who’s New, Who’s Online blocks on left
* enable clean urls
Live Tuning - Setup

* enable devel & generator modules

* 2000 users, 250 terms, 15 vocabularies, 5000 nodes, 10000 comments, 5000 path aliases

* enable devel block

* (as admin) enable query log, display query log, page timer, memory usage : discuss
Live Tuning - Setup

* Now open http://localhost/performance/

* Look down the bottom of the page for the devel module output:
Live Tuning - Analysis

✶ You can see that (for **logged-in** admin user), 241 SQL queries were required to generate this page.

✶ These queries took 131ms to execute, out of the 312ms that the whole page took (42%).

✶ This 312ms doesn’t necessarily represent what firebug shows:

  ✶ about 230ms for the page (logged in)
  ✶ about 190ms for the page (anonymous)
Live Tuning - Analysis

* Lets see how apache bench does (just the front page for now)

* `ab -c1 -n250 http://localhost/performance/`

<table>
<thead>
<tr>
<th>Percentage of requests served within a certain time (ms)</th>
<th>50%</th>
<th>336</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>66%</td>
<td>341</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>343</td>
</tr>
<tr>
<td></td>
<td>80%</td>
<td>346</td>
</tr>
<tr>
<td></td>
<td>90%</td>
<td>356</td>
</tr>
<tr>
<td></td>
<td>95%</td>
<td>378</td>
</tr>
<tr>
<td></td>
<td>98%</td>
<td>524</td>
</tr>
<tr>
<td></td>
<td>99%</td>
<td>672</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>1167 (longest request)</td>
</tr>
</tbody>
</table>

The average page-time is about 343ms (including network stuff)

Note: this does NOT include JS/CSS etc!
Okay, so we have a baseline performance for a Drupal site

Your site will certainly be different! Do your testing with representative hardware and configuration

Some modules are known to be slow, others just need some care for best performance...

Before we get into how to improve these numbers, a bit of background...
How Caching Works

- A cache is just a quick lookup, from “key” to a “value”. Values expire after some time, or when explicitly cleared.

- Whenever a complex calculation is performed, the results may be stored in the cache, so it’s quick to look up the same result next time (using the key)
How **Drupal** Caching Works

* The default implementation of Drupal caching uses a database table for each different cache

* See [http://localhost/phpMyAdmin/](http://localhost/phpMyAdmin/)

* This works well, but there are other alternatives:

  1. files in the filesystem (fscache, boost)
  2. in memory in a separate process (memcache)
  3. in memory in the webserver process (APC, xcode, etc)

* These implementations covered later, but first...
How Drupal Caching Works

Drupal already makes use of the cache system:

- menus
- filters
- locale
- variables
- pages - when enabled ➞ PTO
- contributed modules
Simple Fix #2

† Administer > Site Configuration > Performance : Page Cache

† **normal**: probably what you want. Improved apachebench from 340ms to 42ms!!

† **aggressive**: enable skipping of module invocation during cache hit. Improved apachebench to 11ms!!

† can set Minimum Cache Life if required

† cache is stored using Drupal cache system (ie: DB by default)

† doesn’t help logged-in users at all
Block Cache Module (#3)

- [ ] http://drupal.org/project/blockcache

- If a block is “complex” to generate, it may be worthwhile caching the HTML output

- Enable “blockcache” module, then replace blocks with cached blocks. Set cache time explicitly if required. Caching can by block (default), or per-page or per-user (explain)

- Complex DB queries, or blocks that access external systems, may be good candidates for block-cache.

- Default blocks + DB caching - probably not worthwhile?

- Don’t bother caching “simple” blocks using standard Drupal cache - since that’s stored in the DB anyway
fastpath_fscache Module

* [http://drupal.org/project/fastpath_fscache](http://drupal.org/project/fastpath_fscache)

* Instead of using the database for the caches, this module uses the filesystem. Under some circumstances, this is faster.

* Plus a hook for the pre-database *page caching* - fastpath

* Install and enable module, paste settings into settings.php. With concurrency of 5: <=====

  * With “normal” page cache and default caching, the average was 103ms

  * With “normal” page cache and fastpath_fscache, the average was 18ms
The boost module writes static HTML (for anonymous users) to the filesystem, then uses Apache mod_rewrite rules to fetch them, without ever invoking any PHP!

Boost increased anonymous page loads from 103ms (normal cache) to 1-2ms. Wow!

sometimes a bit tricky to get the mod_rewrite rules correct (especially with non / installs) - check INSTALL.txt

http://bendiken.net/2006/05/28/static-page-caching-for-drupal
Instead of caching in the DB, use a “high-performance, distributed memory object caching system” (memcache)
http://www.danga.com/memcached/

+ PECL Memcache library for PHP
http://pecl.php.net/package/memcache

+ Drupal module for using PECL Library
http://drupal.org/project/memcache
Memcache - Advantages

- memcached can be put on any server with spare memory - webserver, DB, load balancer, etc

- Multiple memcached instances possible, each of which serving one or more “cache buckets”

- Separate process to Apache, so multiple web servers can share multiple memcache instances (better cache reuse)

- Since it’s just a quicker implementation of Drupal caching, this works with everything, including logged-in users!
Memcache Deployment

* memcache-5.x-1.7 was released this week

* Installation involves a small patch to core (which currently serializes everything, even if we don’t need it to)

* Extra patches for Views and CCK

* Read the README.txt for troubleshooting
Memcache - Testing

* Demo: enable module, apply patch, start memcached -vv, clean cache tables in DB
* Open / - view memcached console
* Restart memcached without -vv
* Run benchmark

* I got 57ms average request with five concurrent threads, vs 320ms with standard caching (“normal” caching mode) - 82% saving!
Memcache - References

* Memcached - Lightning Fast Drupal Sites

* How to install memcache on Debian Etch
  http://www.lullabot.com/articles/how_install_memcache_debian_etch

* Install the Memcached service on Mac OSX
APC (xcode, etc)

- APC is both a bytecode cache, and a general object cache
  [http://drupal.org/project/apc](http://drupal.org/project/apc)
- APC module for Drupal - somewhat similar to memcache, but stores data in the APC memory
- Potentially a little more efficient, since objects don’t have to be serialized or transmitted over the network
- Not suitable for multi-webserver environments due to cache invalidation issues
Advanced Caching Module

- http://drupal.org/project/advcache

- From the module page:
  "The advanced caching module is mostly a set of patches and a supporting module to bring caching to Drupal core in places where it is needed yet currently unavailable. These include caching nodes, comments, taxonomy (terms, trees, vocabularies and terms-per-node), path aliases, and search results.

- Because it uses Drupal’s caching system, Advanced Caching Module is compatible with memcache, APC cache, etc.

- Advanced Caching + Memcache brought the 57ms in the previous test down to 41ms (-30%). And these are application level caches, so will work with logged in users!"
Like we said before, whenever an “expensive” calculation/operation is performed, it may be worthwhile caching the result and using that.

The value of “worthwhile” may depend on your caching system (eg: memcache vs DB).

Don’t get carried away, probably don’t cache simple selects.

http://www.lullabot.com/articles/a_beginners_guide_to_caching_data
Using Caching in YOUR code!

Simple Example: put the following code in a custom block (PHP format)

```php
// Pretend to get a piece of really important content from a slow server.
sleep(1);
$content = time();

// Emit block output
print t('This is a slow block: !content', array('!content'=>$content));
```

- Of course, now it benchmarks at pretty close to 1.3 seconds per request (including the 300ms it really takes with no cache)

- Note: disable the page-cache to allow testing this block as anonymous - but the idea still works for logged-in users!
Using Caching in YOUR code!

* Obviously, it would be quicker if we didn’t have to do this “slow web-service” every page request, so let’s cache it:

```php
$cach = cache_get('blocktest_content');
if ($cache && !empty($cache->data)) {
    // cache hit
    $content = unserialize($cache->data);
} else {
    // cache miss: pretend to get a piece of really important content from a slow server.
    $content = time();
    cache_set('blocktest_content', 'cache', serialize($content), time() + 60);
}
return t('This is a cached slow block: !content', array('!content'=>$content));
```

* Yes, this could be more-or-less accomplished using block-cache (discuss output caching vs data caching)

* If the function you’re adding caching to could be called more than once per request, it may be worth caching it in a static variable too (even faster than drupal caching :)

Demo
Using Caching in YOUR code!

```php
cache_set('blocktest_content', 'cache', serialize($content), time() + 60);
```

Notes about the code:

- 1st parameter is the “cache key” and forms the unique identifier for the piece of content
- 2nd parameter is which cache to use (in this case, the default cache)
- 3rd parameter is the data to be cached
- 4th parameter is the expiry time for this item (+60s)

See [http://api.drupal.org/api/function/cache_get/5](http://api.drupal.org/api/function/cache_get/5) and [http://api.drupal.org/api/function/cache_set/5](http://api.drupal.org/api/function/cache_set/5)
You may also need to invalidate entries from your cache. You do this by calling

```
cache_clear_all($cid = NULL, $table = NULL, $wildcard = FALSE)
```

http://api.drupal.org/api/function/cache_clear_all/

For example:

- `cache_clear_all('blocktest_content', 'cache')`
  clears the entry put in previously

- `cache_clear_all('blocktest', 'cache', 'blocktest')`
  clears all entries in “cache” starting with “blocktest”
Using Caching in YOUR code!

★ This kind of caching is compatible with any of the Drupal-cache replacements (eg: memcache, APC cache, fscache, etc)

★ You can cache (for example):

★ HTML output (fragments or your whole output)
★ Results from complex queries
★ Results from external systems
★ Anything that is slow!

★ Remember: you must be able to recreate the result, caches are not permanent stores
That’s it!
Some of the material for this talk comes from the Performance and Scalability Seminar at the OSCMS Summit
http://www.lullabot.com/articles/performance_and_scalability_seminar_slides

More useful articles at

Server Tuning Considerations
http://drupal.org/node/2601

Interesting series at
http://www.johnandcailin.com/blog/john/scaling-drupal-open-source-infrastructure-high-traffic-drupal-sites

This presentation available on conference website