(An) Optimal Drupal 7 Module Configuration for Site Performance

JOE PRICE

Intro

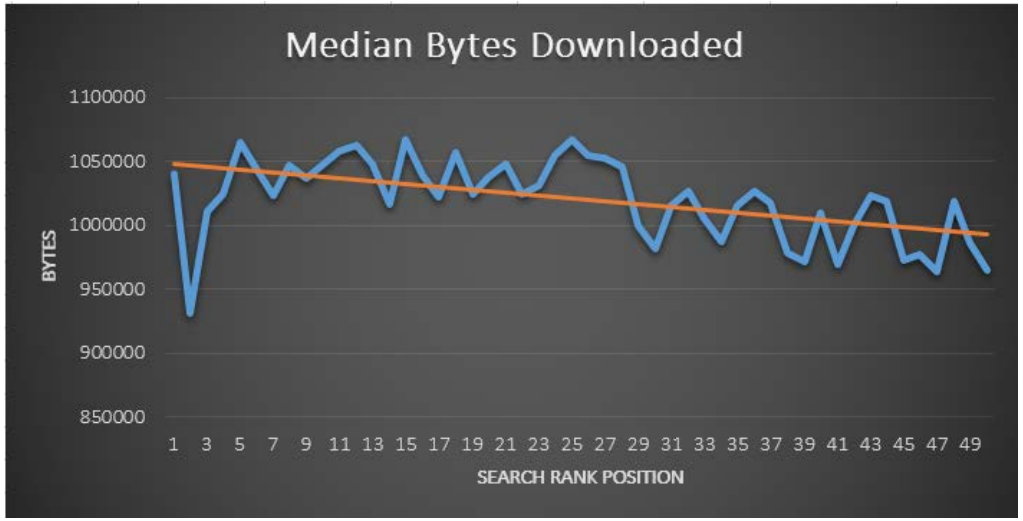
- I’m a performance junkie.
- My top three non-Drupal performance tools are Apache Bench, Google PageSpeed Insights, and NewRelic.
- I currently manage Broadstreet Consulting’s servers/site performance.

- 80% of today’s talk will cover **performance improving modules**
- The rest will cover **performance degrading modules**

Gist reference https://gist.github.com/pricejn2/10943822

Why We Care

- Performance (TTFB) impacts search ranking
- Users expect a fast site (that includes me)
- Delays drive users away
  - 47% of consumers expect a web page to load in 2 seconds or less.
  - 40% of people abandon a website that takes more than 3 seconds to load.
  - A 1 second delay in page response can result in a 7% reduction in conversions.
  - If an e-commerce site is making $100,000 per day, a 1 second page delay could potentially cost you $2.5 million in lost sales every year.



Types of Performance Improving Modules

Five categories for the purposes of our discussion today:

- Caching
- Compressing
- Concurrency
- Callbacks
- Queries

Caching

- This section **could be a whole discussion series** in and of itself
- This **isn't about server set up**, but its hard to completely avoid the topic of at least dependencies
  - DISCLOSURE: I have my preferred server configuration for non-enterprise scale sites, and server-specific modules will inevitably reflect that bias
- The goal here is to highlight the **most useful caching modules**, regardless of current popularity
- Drupal **modules aren't the magic bullet** to fix all your performance problems
- With that…

Caching: Novice

These provide more granularity for different Drupal parts and are all drop-in with frontend configuration required per part in most cases:

- Block Cache Alter (blockcache_alter)
- Display Cache (display_cache)
- Panels Content Cache (panels_content_cache)
- Views Content Cache (views_content_cache)

Enforces Views time-based caching for all uncached views:

- Views Cache Bully (views_cache_bully)

Caching: Intermediate

- Boost (boost)
  - Static page caching (ie, serve *.html instead of php) -- best for primarily anonymous traffic.
  - This is a must if you're stuck using a shared host.
  - Use with Cache Expiration (expire) for more intelligent handling of stale content on larger/busier sites with regular updates.
  - mikeytown2
- CDN (cdn)

Caching: Advanced

- Entity cache (entitycache)
  - Use with Redis (redis) for maximum benefit
- ESI - Edge Side Includes (esi)
  - ESI/SSI support; server-side authenticated user cache
- Redis (redis)
  - Alternative cache backend integration for Drupal

Avoiding stale cache:

- Expire (expire)
  - Ensures cached items are timely
- Purge (purge)
  - Clears URLs from reverse proxy caches like Varnish, Squid or Nginx

Compressing

- Advanced CSS/JS Aggregation (advagg)
  - mikeytown2
  - Core CSS & JS aggregation, compression, and optimization enhancement.
  - Build aggregates in background using httprl support
  - Drop-in ready
- Speedy (speedy)
  - Provides minified versions of core JavaScript files
  - Reduced benefits if used with advagg_js_compress
  - Drop-in ready

Concurrency

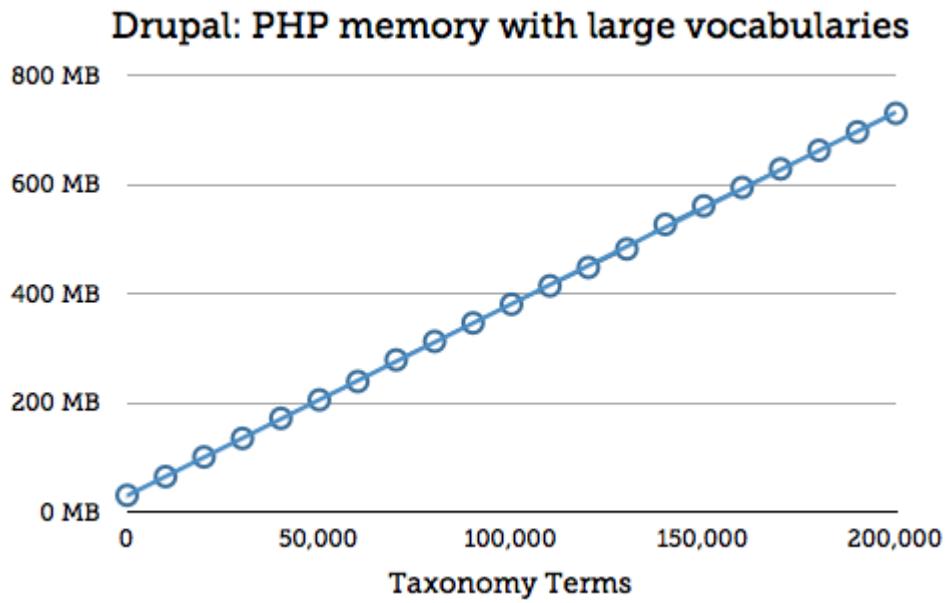
- CSS Embedded Images (css_emimage)
  - Really more about reducing total HTTP connections then concurrency per se
  - Integration with advagg
  - Drop-in ready
- HTTP Parallel Request Library (httprl)
  - Multiple requests using httprl can be over 5x faster then core's drupal_http_request() because multiple connections are open and the streams are downloaded in parallel.
  - Another mikeytown2 contribution. Works fantastically in conjunction with his advagg module.
  - Drop-in – integrates with advagg, linkchecker, purge, and others

Callbacks

- JavaScript/AJAX callback handler (js)
  - Simplified bootstrap -- include only the necessary files needed to serve the request.
  - API to other developers who want to improve their project's AJAX (, SOAP, AHAH, JSON, etc.) performance
  - Drop-in, but with limited benefits to most existing sites. One common exception being admin_menu users.
  - More reading for the interested -- http://www.pixelite.co.nz/article/high-performance-ajax-callbacks-drupal-7-and-js-module

Queries

- Taxonomy Edge (taxonomy_edge)
  - Specifically for optimizing Drupal's hierarchical taxonomy functions
  - Overrides expensive taxonomy_get_tree() and taxonomy_select_nodes()
  - Provides a transitive closure table data model
  - Important when you have > 20,000 terms



Performance Degrading Modules

- Database logging (dblog)
  - For a production site, alleviate the amount of writes on the database
  - Enable to debug else disable
- Update manager (update)
  - Reduce cron jobs
- Coder (coder)
- Devel (devel)
- Localization update (l10n_update)
- Any unused module

Every Production Site Should...

1. At a bare minimum, **core + advagg + httprl** with proper configuration and disable performance degrading modules (see gist)
2. Consider enabling **css_emimage + speedy** for most circumstances.
3. Drupal caching is not a one-size-fits all solution, but for any non-Commerce site using Views, **views_cache_bully** is a highly recommended easy win.
4. Utilize **entitycache**.
5. If you have more than 20k taxonomy terms, start leveraging **taxonomy_edge**.

A lot more to think about beyond Drupal modules

- CDN
- Server configuration
- Optimized images
- Render-blocking external resources

Thanks!

QUESTIONS?