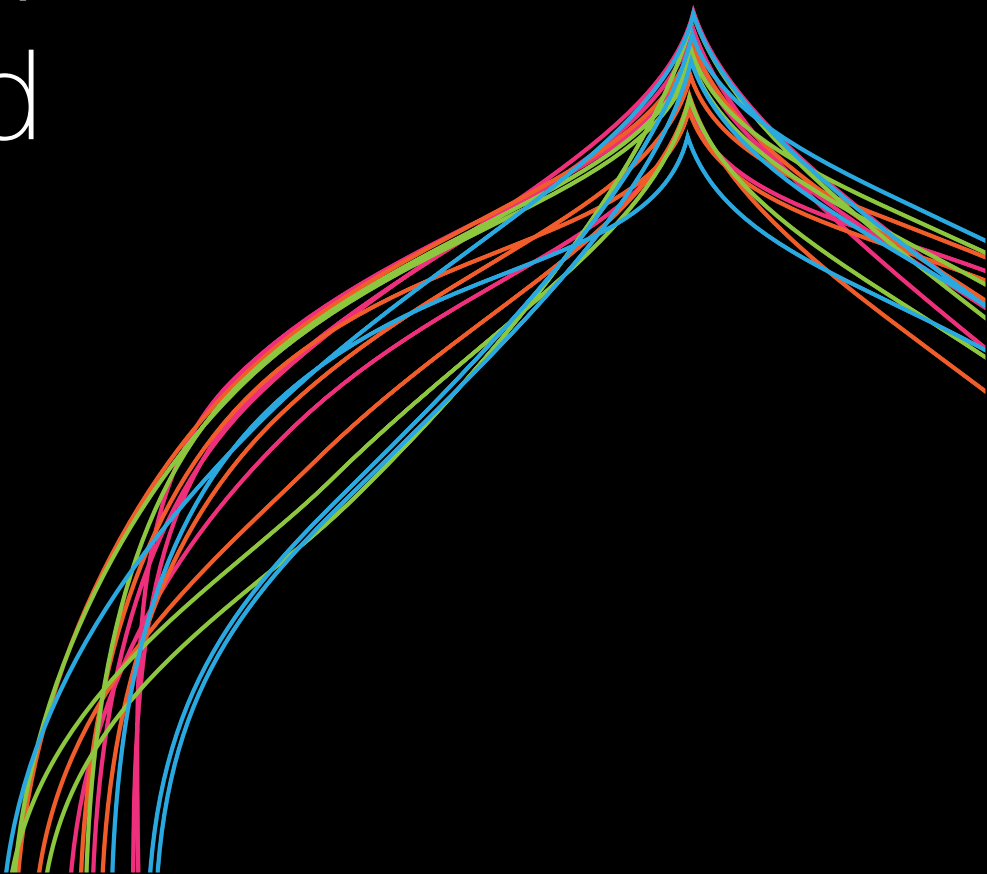
Acquia® THINK AHEAD.

Simple Tips to Improve Drupal Performance: No Coding Required



By Erik Webb, Senior Technical Consultant, Acquia

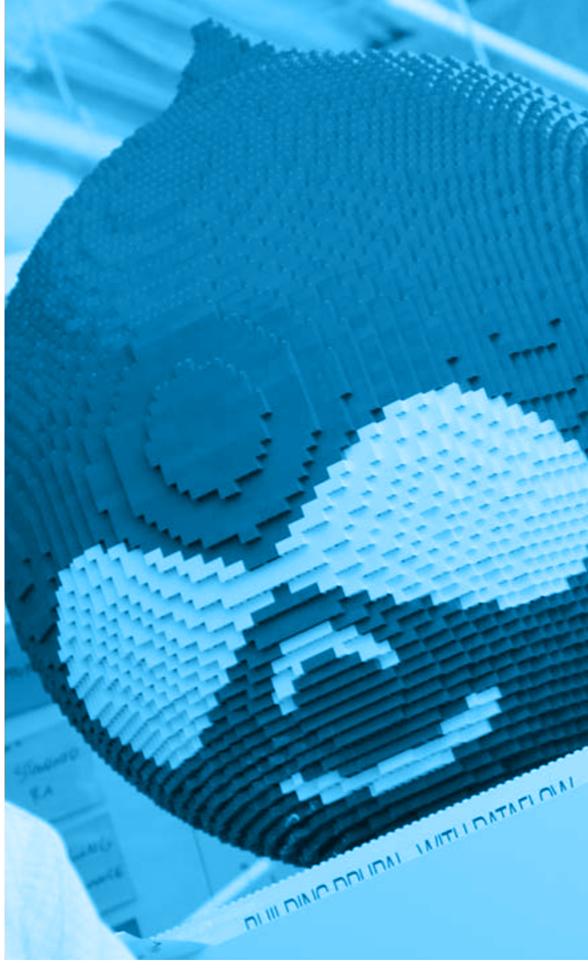
Table of Contents

Introduction 3
Types of Drupal Caches 5
Performance Workflow 6
Questions to Ask about New Modules 7
Build for Performance—Performance Page 9
Build for Performance—Other Improvements 11
Troubleshooting—Where Problems Occur 12
Troubleshooting—Process 13
Performance-related Modules 14
Third-party Tools 16
Conclusion 17

Introduction

Drupal is an extremely flexible framework, with an abundance of configuration options. With any large framework, flexibility and performance must be balanced. With correct configuration, Drupal can be lightning fast.

In this eBook, we will show you how to optimize performance of your Drupal site without any knowledge of coding, server configuration, or the command line.



Focus of This eBook

We will focus on those issues commonly found in Drupal, which any Drupal site builder can easily configure.

[Note: Drupal runs on a stack of software, and each of the layers can be a source of performance issues. Tuning your server software (usually Apache), database (usually MySQL), or programming language (PHP) are beyond the scope of this eBook].

Drupal 101 Page Building

To optimize Drupal performance, it is helpful to compare how a webpage is built from a static HTML file versus how Drupal builds one. The code and content of a static HTML page are in a file that is loaded when the file's URL is called from a browser. The file might call on other files for CSS or JavaScript, but the load is quick.

Drupal, on the other hand, must execute a number of complex operations. Initializing Drupal is called *bootstrapping,* and is done on virtually every page load. Bootstrapping involves initializing core Drupal libraries, initializing the database, then stepping through a number of additional operations. These operations include checking for modules that must be initialized, checking user access permissions, and retrieving data from cache. A single page may also call several CSS and other support files to build the page.

All Drupal content is saved in a database. The operations that consume the most time in bootstrapping are numerous trips to the database, for content, settings, and so on. The more database calls to create the page, the longer it takes to build. Minimizing the number of operations needed to build a page is key to improving Drupal performance.

Cache for Speed

Through the use of caching, elements of the page are saved in their rendered form so that a trip to the database is not necessary. When the cached page or section is called, HTML is returned, and the speed of the static HTML page is regained. Caching is the primary approach to improving Drupal performance.

Drupal 101 Definitions

Throughout this eBook, we will refer to specific aspects of Drupal, as follows:

- **Anonymous user:** Visitor to the site who is not logged in.
- **Authenticated user:** Visitor to the site who has a username and password and is logged in.
- **Node:** Main unit of content on Drupal sites.
- **Block:** Bits of content that can be placed around the main content in regions defined by the theme.
- **Theme:** Collection of files (HTML, CSS, JavaScript, and so on.) used to determine look and feel of Drupal sites.
- **Views:** Drupal module that creates SQL queries, determined by settings in Views UI, and displays the results.
- **Panels:** Drupal module that provides a drag and drop content manager for creating customized layouts.

...Drupal caches data at the application, component, and page level. Knowing basics about these can help you understand why performance bottlenecks exist in certain places and not in others.

Types of Drupal Caches

Drupal has three types of caches. Knowing some basics about these can help you understand why performance bottlenecks exist in certain places and not in others.

Application-level Caching

Drupal has many caches built into core. These application-level caches can't be turned on or off, or be optimized. When instructions for various Drupal operations suggest "clearing the cache" to see changes, it's these caches they refer to.

Component-level Caching

For the purposes of this eBook, components are the visual elements that make up the sites, including everything on the sidebars, the headers, and the main content areas. Views, Panels, and Blocks come under this heading. Caching components can greatly improve performance. If you have the same block displayed on every page of your site, for example, caching that one block will improve performance on your entire site.

Page-level Caching

This is your most efficient caching. If the entire page is cached, Drupal bootstrapping is not necessary. This type of caching only works for anonymous users, so its usefulness depends on your site traffic.

To build a performant site and identify problems when they occur, establish a performance goal, measure performance with hard numbers, and document continually.

Performance Workflow

The following suggestions will help you build a performant site and identify problems when they occur. Ideally, these processes should be part of your initial workflow, but you can benefit from adding them at any point.

Establish a Performance Goal

How fast does your site need to be? For an important customer-facing site, every millisecond counts. For an intranet site, performance may not be the top priority. If the aesthetics of the site are important, with a rich theme or interactive JavaScript, longer load times may be acceptable.

Measure Performance

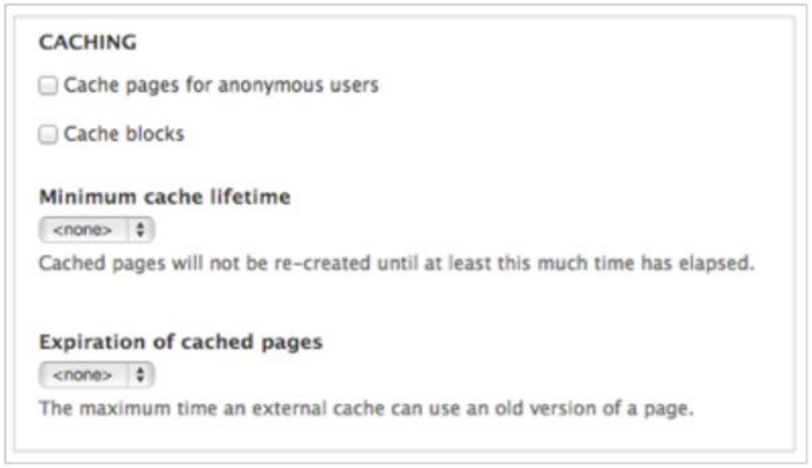
Use tools to measure performance. Which parts of a site are fast or slow? Set up sources of good analytical data, using some of the tools we'll suggest later in the eBook.

Document

The most important thing you can do to manage performance is document the project, keep track of changes, and correlate that with performance. Consistently identify when things are getting worse or better and exactly how they're changing. Any time you add a module to your site, record a baseline of your site's performance before you install it, after you install it, and after you fully configure it. Do the same for any change you implement on your site.

Knowing the modules on your site, how they work, and where they could cause a problem can save headaches later on.

Questions to Ask about New Modules


CACHING

☐ Cache pages for anonymous users

☐ Cache blocks

Minimum cache lifetime

<none>

Cached pages will not be re-created until at least this much time has elapsed.

Expiration of cached pages

<none>

The maximum time an external cache can use an old version of a page.

Knowing the modules on your site, how they work, and where they could cause a problem can save headaches later on. Any time you consider adding a module to your site, ask the following questions:

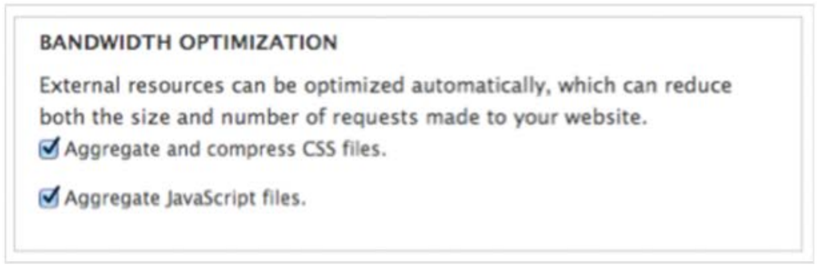
- What does this module do?
- Does the site really need it?
- When does this module run?

– What other functions does this module affect?

As a general rule, the more modules you add to your site, the more performance demands you create. No specific cap on the number of modules exists to predict slow performance.

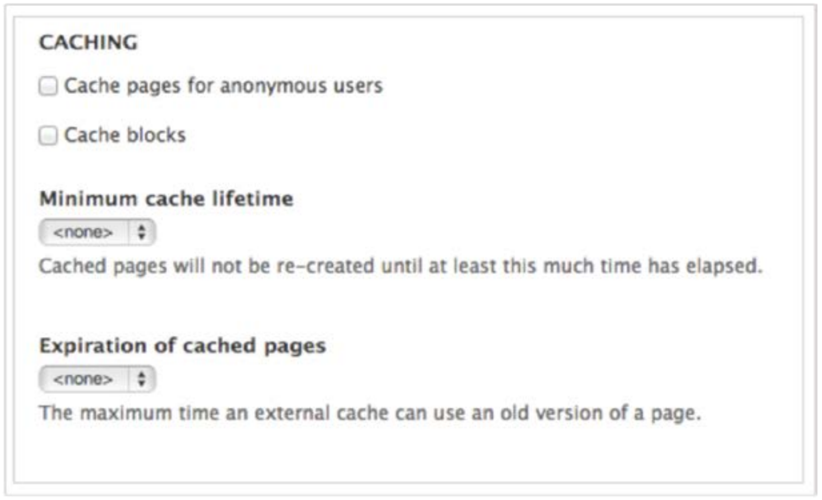
Check the module's issue queue using the tag "Performance," rather than "Slow" or something similar. The "Performance" tag is more likely to give you relevant information.

Read and understand the module's description on the project page and look through any available documentation on drupal.org. Load the module on a test or development site, and try it out. Read the readme.txt file. Does the module function when a user logs in or when content is being changed, or does it act in the background through cron or some other periodic system?


BANDWIDTH OPTIMIZATION

External resources can be optimized automatically, which can reduce both the size and number of requests made to your website.

- [✓] Aggregate and compress CSS files.
- [✓] Aggregate JavaScript files.


CACHING

- [ ] Cache pages for anonymous users
- [ ] Cache blocks

Minimum cache lifetime

<none>

Cached pages will not be re-created until at least this much time has elapsed.

Expiration of cached pages

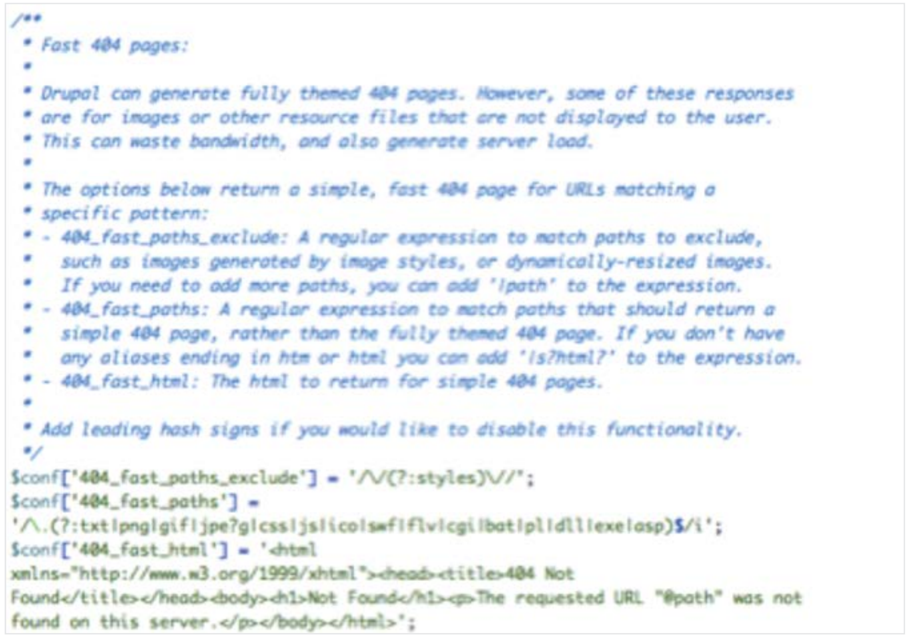
<none>

The maximum time an external cache can use an old version of a page.

Build for Performance
— Performance Page

In the next sections, we'll suggest specific settings to improve the performance of your Drupal site. All examples are for Drupal 7, but for most examples, similar setting exist in Drupal 6 as well.

For the settings in this section, visit the performance page on your site. (example.com/admin/config/development/performance.)



```
/**
 * Fast 404 pages:
 *
 * Drupal can generate fully themed 404 pages. However, some of these responses
 * are for images or other resource files that are not displayed to the user.
 * This can waste bandwidth, and also generate server load.
 *
 * The options below return a simple, fast 404 page for URLs matching a
 * specific pattern:
 * - 404_fast_paths_exclude: A regular expression to match paths to exclude,
 *   such as images generated by image styles, or dynamically-resized images.
 *   If you need to add more paths, you can add '|path' to the expression.
 * - 404_fast_paths: A regular expression to match paths that should return a
 *   simple 404 page, rather than the fully themed 404 page. If you don't have
 *   any aliases ending in htm or html you can add '|s?html?' to the expression.
 * - 404_fast_html: The html to return for simple 404 pages.
 *
 * Add leading hash signs if you would like to disable this functionality.
 */
$conf['404_fast_paths_exclude'] = '/\/(?:styles)\//';
$conf['404_fast_paths'] =
'/\.(?:txt|png|gif|jpe?g|css|js|ico|swf|flv|cgi|bat|pl|dll|exe|asp)$/i';
$conf['404_fast_html'] = '<html
xmlns="http://www.w3.org/1999/xhtml"><head><title>404 Not
Found</title></head><body><h1>Not Found</h1><p>The requested URL "@path" was not
found on this server.</p></body></html>';
```

ACQUIA® THINK AHEAD.

Block Caching

Blocks can be cached even if other parts of the page cannot, so even authenticated users can benefit. However, block caching doesn't work with content access modules. Content access is a Drupal system that determines permissions on the site and allows users to see defined sets of content. With Organic Groups or any module that implements node_grants, the option for block caching will be disabled on the performance page.

Minimum Cache Lifetime

Updating Drupal content clears all block and page caches. If you update content frequently, that cache will clear frequently and slow your site's performance. In this situation, setting "minimum cache lifetime" can help. No matter how often content is updated, the cache will not clear for at least the minimum amount of time you specify.

Expiration of Cached Pages

A reverse proxy, like Varnish, is a server that sits in front of your web servers and caches their content, including page HTML as well as CSS or JavaScript. If you're using a reverse proxy, the "Expiration of cached pages" setting will tell the external system how long to store a piece of content before requesting a new copy from the server.

Bandwidth Optimization

Aggregating and compressing CSS and JavaScript is a front-end performance issue. There are generally two to three dozen of these files on a fair sized Drupal site. Aggregating them compresses all those files to a handful and speeds up site performance on the browser side. On dev sites, this setting can complicate development.

Build for Performance
—Other Improvements

Fast 404

404 errors can be a big performance problem for Drupal. Without Fast 404, Drupal implements a full bootstrap before it determines that content does not exist. Fast 404 allows you to exclude certain paths and extensions and send an error message immediately rather than using unnecessary resources. Although Drupal 7 includes a rudimentary version of the functionality in settings.php, the Fast 404 module is more extensive [http://drupal.org/project/fast_404].

Individual Module Caching

Many modules, including Views and Panels, have settings for caching in the administrative UI.

UI Modules

Large modules like Views often separate the administrative interface from the functional parts. These UI modules are extra overhead that can be disabled on production sites.

Database Logging

Your database is the ultimate bottleneck. Once you've optimized everything else, the database is the hardest thing to scale. Drupal's default core Database logging module sends a message to the database for every event. On a high traffic site, you don't want all those messages to hit the one thing that you can't scale very well. Instead, use the Syslog module, which is also part of Drupal core. Syslog logs to the operating system logs instead of to the database logs and doesn't affect performance. Other options also exist to get the same information. Doing this one thing can take 5–10% of the load off the database.

The most likely sources of performance problems include page building modules, external or third-party services, overall complexity, and misconfigured components.

Troubleshooting —Where Problems Occur

So you've configured your site for performance, established a performance baseline, and documented your work. What do you do when a performance problems occurs?

First of all, be aware of the most likely sources of performance problems. These include:

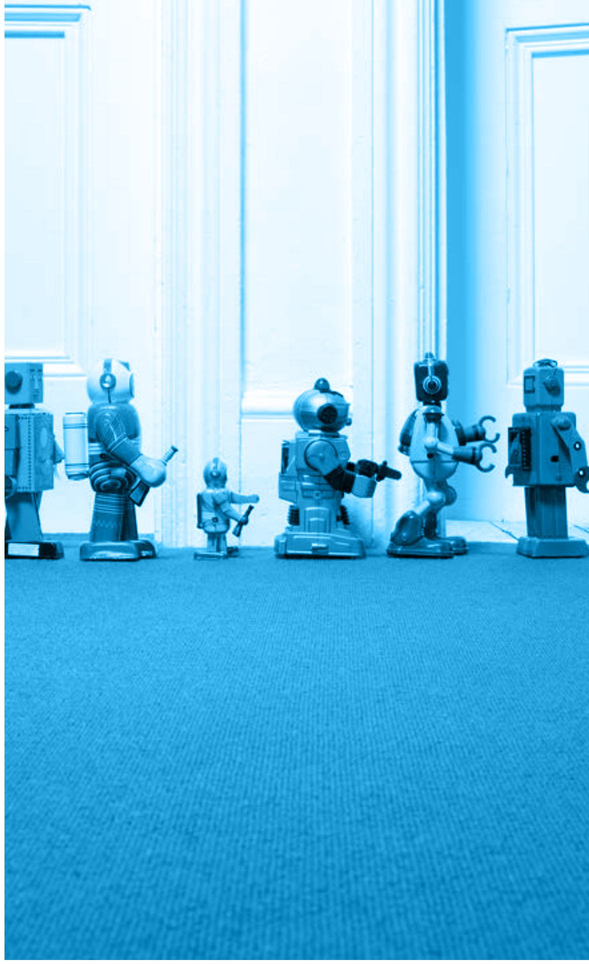
- Page building modules, like Views and Panels. These are the data-driven components of the site that manipulate a lot of content. These are the first place that a site tends to encounter a bottleneck when trying to scale. As mentioned earlier, these can benefit from caching.
- Any external or third-party service. Generally, you will not have control of how fast these perform or even whether they are running. When you experience performance problems, it's important to know how those external services affect the site.
- Overall complexity. Drupal's flexibility makes it easy to build complex layouts and big functional sites. At a certain point, that complexity can be slow from a standpoint of overall execution. An example would be a view inside a panel inside another panel, or a very large number of modules.
- Misconfigured components. Many Drupal configuration defaults are good for development sites but not for production. Almost nothing is cached by default in Drupal. Learn what needs caching on your site and which defaults are best changed. We covered some of these settings in the “Build for Performance—Other Improvements” section.

Track down the source of a problem by asking when it occurs, when did it start, and who was responsible.

Troubleshooting—Process

When problems do occur, analyze them to discover the source. If you have set up the recommended workflow, answering the questions should be easy and lead you to solutions.

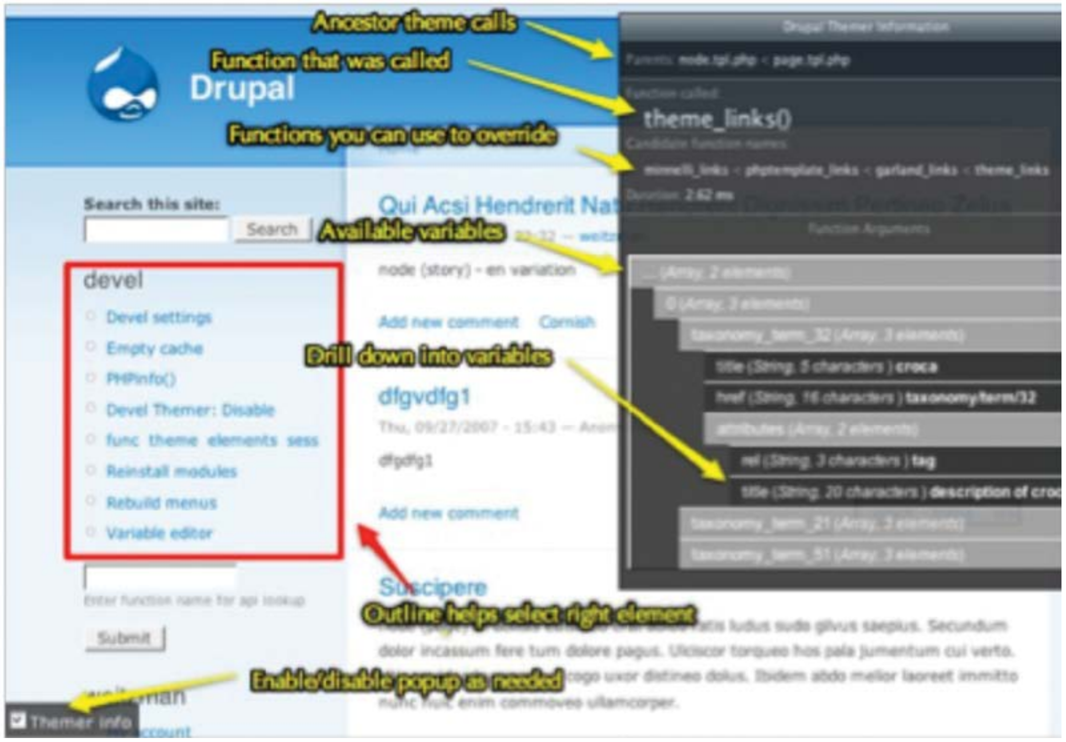
- When does it occur? Is this happening on all pages? For all users, or just for logged in users? Does this occur at the time a user logs in? If you're getting help requests and can reproduce the problem, this first question is easy to answer.
- Once you have this answer, go back to those initial notes you made about your modules and changes to determine which ones affect the process in question. When you know which modules or other changes could be the source of the performance issue, you can act on that information.
- When did it start happening? What changed on that site at that time? Use the documentation you have in place—your version control system if available—to correlate the changes in performance with the changes on the site. It's important that you keep track of the numbers for the overall performance of the site, on a daily or weekly basis. We'll talk about some monitoring tools later that will give you good hard numbers.
- If you are working with a team, who made the changes in question? If you have several developers, not everyone knows what everyone else is doing. It's important that when new modules or other changes are made to the site, documentation indicates who was responsible and the reasoning behind the decision.



Performance-related Modules

Devel

Whether you are a developer, coder, or site builder, the Devel module provides many benefits. It can help by showing you the number of queries run on a page, the execution time, how much memory was used, and other metrics. Even if you don't fully understand these numbers, you can use them for comparison. For example, note how numbers change with installation of a new module, disabling a module, logging in, and so on.



Boost

This module works well for shared hosting. It provides static page caching for anonymous users and bypasses Drupal bootstrap completely. (In a full-size infrastructure like Acquia hosting, more efficient caching methods, such as Varnish or Memcached, are available.)

Entity Cache

With this module enabled, entity objects will be cached. With simple content types or small amounts of content, the gain is not large. With complex content types or large amounts of content, it can be extremely helpful.

Block Cache Alter

If you don't want to cache all your blocks through the setting we mentioned in the Performance page, use Block Cache Alter. With this module, you can determine cache settings for each block individually.

Views Litepager

Views Litepager targets large sites with certain database configurations (InnoDB tables) and speeds up creation of pagers for Views. You lose some functionality, but the performance gain is very noticeable.

Views Content Cache More here

Cache Actions

Used in conjunction with the Rules module

These modules provide content-driven expiration. Most default caching in Drupal is based on timeframes, with the cache expiring according to the time setting. Views Content Cache and Cache Actions clear the cache only when content changes. With Views and other content that changes rarely, this provides a performance benefit. Note that Cache Actions provides actions that can be used with the Rules module.

Third-party Tools

Smush.it

A tool for optimizing images.

SpriteMe

Helps in creating sprites. Sprites aggregate images from a webpage into one file used in several places to reduce the number of server requests.

Web Page Testing Tools

Webpagetest.org

Gives metrics for how fast your site is. Simply throw in a web address and get numbers.

Google PageSpeed online

An online service providing the same PageSpeed tools in your browser. Being an online service, it can be run from other computers or automated.

Firebug More here

Web Inspector More here

These tools can analyze requests, providing information like the number of requests coming back. They look at headers to determine whether the page is cached and measure how long each item takes to return, among other things.

SaaS Products

New Relic

This tool analyzes where your application is slow: for example, slow database queries, external services, or types of pages. It analyzes at a low level so that you can identify how a module or a specific function performs.

Yottaa

Yottaa will tell you how fast your site loads in different locations around the world and what external visitors see rather than what you're experiencing locally. Yottaa gives you eyes out in the wild so you can see what an average person is actually experiencing with the site.

Conclusion

Improving the performance of a Drupal site doesn't require knowledge of servers, coding, or the command line, although all those things can be helpful. Keep the performance tips from this eBook in mind as you build your site. Incorporate your knowledge of optimal performance into your workflow, your project notes, and your module decisions.

These simple performance tips will help you keep your site running smoothly.