



# Drupal Performance Tuning

June, 2010

*Germán Villacreces*

# Performance Tuning Drupal

## Table of Contents

Introduction	3
Test environment	4
The Test Site	4
The Virtual Server	5
Testing tools	5
Single Page Load Testing Tools	5
Load Testing Tools	6
Evaluating the current performance	7
Single Page Load Results	7
Load Testing Results	12
Optimizing Drupal	12
Front-end Optimization	12
Use a CDN	13
Make Fewer Requests	13
Add Expires Headers	13
Configure Entity Tags (E-Tags)	14
Use Cookie-Free Domains	14
Back-end Optimization	15
Page & Block Caching	15
Boost	15
Optimizing your server	16
Tuning Apache	16
Tuning MySQL	17
Tuning PHP	19
APC	19
Memcache	21
Evaluating the performance after optimization	22
Single Page Load Results	22
Load Test Results	24
Conclusion	25
Drupal Success Story	26
Additional Drupal Resources from Oshyn	27
About German Villacreces	29
About Oshyn	29

## Introduction

This paper evaluates the performance of a small site built with Drupal on a standard virtual server running Ubuntu Server Edition 9.10. The evaluation will focus on the results of testing the response time and amount of requests per second (RPS) that the server can hold before the average load time is unacceptable. The first evaluation will be done with the default configuration of Drupal and the server. The second evaluation will take place after optimizing both. The results from the evaluations will be compared to get to a good idea of how fast can Drupal run on a virtual server with the minimum specs. Keep in mind that this publication is to evaluate the performance of Drupal for small sites with low traffic. The server will have the minimum specs. After reading this paper, you will know whether a simple virtual server is more than enough for a low traffic small website and you will learn how much the performance of Drupal can be improved by just tuning the settings and the server.

## Test environment

### The Test Site

The site used for this benchmark was built with the latest Drupal 6 release at the time of this writing (Drupal 6.16). The site makes use of the following modules in addition to the required core modules:

Administration menu	Token and token actions	ImageAPI with GD2
Views and ViewsUI	Database logging	ImageCache & ImageCacheUI
CCK and all Submodules	Help	CKEditor
FileField, ImageField	Locale	FlashNode
Panels (CTools, MiniPanels and PanelNodes)	Menu	Global redirect
Blog	Path	Pathauto
Color	PHP Filter	PDF version
Comment	Search	Printer-friendly pages
Contact	Taxonomy	Send by e-mail
Content Translation	Update status	SWFTools and SWFObject
	Upload	Google Analytics

The site was built using the features from all the above modules. The homepage will be the target for all the testing. This is what it looks like:



This page is built with the Panels module, 5 different views are used.

The page has numerous elements:

- 1 HTML page request (4.5 kb)
- 1 XML asynchronous page request (510 b)
- 16 stylesheet requests (10.9 kb)
- 5 JavaScript requests (29.6 kb)
- 9 image requests (102.5 kb)

1 initial Flash request (45 kb)

4 additional fFlash requests (712.6 kb)

Lazy loading is used for the Flash additional requests and the XML so the total initial page request size is **190.6 kb** and the additional request size is **713.1 kb**

The average total page size according to a [research](#) made is 130kb. So the homepage is a little over the above because of the main Flash element. We should keep that in mind.

## The Virtual Server

The virtual server used has the following specs:

Intel Core 2 Duo 2.0 Ghz 64-bit architecture

70GB disk space

2GB Ram

The virtual server runs under VMware ESX 3i, 3.5.0

The OS installed is Ubuntu Server Edition 9.10 64-bit.

The web server LAMP stack versions are:

Apache 2.2.12

PHP 5.2.10

MySQL 5.1.37 installed

Apache and PHP configurations are set with Drupal's requirements:

- Apache mod\_rewrite extension enabled
- PHP GD library extension enabled
- PHP Register Globals variable is set to off.
- Memory limit is set to 96Mb because this is the minimum recommended library when using the ImageCache module.

## Testing tools

When testing the performance of a site, we ask ourselves, how we should measure this? The response time when loading a page? The volume of users that can access the site at the same time without the server crashing? The answer is both.

Our approach will evaluate the performance of a Drupal site in a virtual server by measuring the response time of the homepage and the number of requests per second the server can take without crashing or causing request queues.

## Single Page Load Testing Tools

We need to measure how long it takes to load all the page elements that form the homepage. This includes the HTML, JavaScript files, CSS files,

images and Flash files. There are great measuring tools that let you know how fast or slow your page is loading.

The first tool we are going to use is very popular, it's called [YSlow](#). YSlow is a [Firebug](#) extension which is a [Firefox](#) plugin. YSlow grades your site's load time using several criteria. To improve your overall result you must get a better grade on each criteria.

The second measurement tool we are going to use is an online web page test tool. The site [webpagetest.org](#) conducts a test to the site you want from the different locations in the world and returns several results, including a "waterfall" view of the page load. The total page load time and a grade are the two main values we are interested in.

## Load Testing Tools

The load testing tools are needed to measure the amount of traffic that your server can take before user requests are queued or the server crashes. This task is not complicated; we can use [Apache's jMeter](#) to run the tests.

The load testing will be executed with different RPS values each time while monitoring the server's health. The RPS value will increase until the server's response time is not acceptable. Before beginning to test, we need to define the maximum waiting time we are to accept before considering that the server cannot take any more load. This [can be very complicated to answer](#) if we try to generalize a universal acceptance response time. This is because this value is different for people depending on the content they are waiting to see. To keep things simple for these tests, we are going to stick to the traditional 8-second rule. As soon as we have an average response time above 8 seconds, we will conclude that the server can't take any more requests.

It is important to point out that the tests are made from average 0.62 Mb/s connections. This is under the average ISP speed in the US but can be considered as an average worldwide according to this [global report](#) from most popular speed tests sites [speedtest.net](#).

## Evaluating the current performance

So far, we have not changed anything in the Drupal site to optimize it. We are starting with 1GB of RAM in the server and evaluating all the results.

### Single Page Load Results

The next figure displays the detailed grading provided by YSlow:

**The overall grade is D**, this is pretty bad but we can't really ask for more considering that none of the Drupal optimization features are enabled. Thanks to this tool we know what we can do to get a better grade and feel

confident that our site is optimized as much as possible in the front-end.

YSlow also displays that the page load time was:

onload: 2.88s

total load: 10.32s

This means that the page took 2.88 seconds to initially load but the total page response that includes the 4 additional Flash objects and the XML file that are loaded asynchronously took 10.32 seconds to load. This second number may seem high, but you need to understand that it's because of the Flash and XML files that sum up to a total load weight of 713.1 kb.

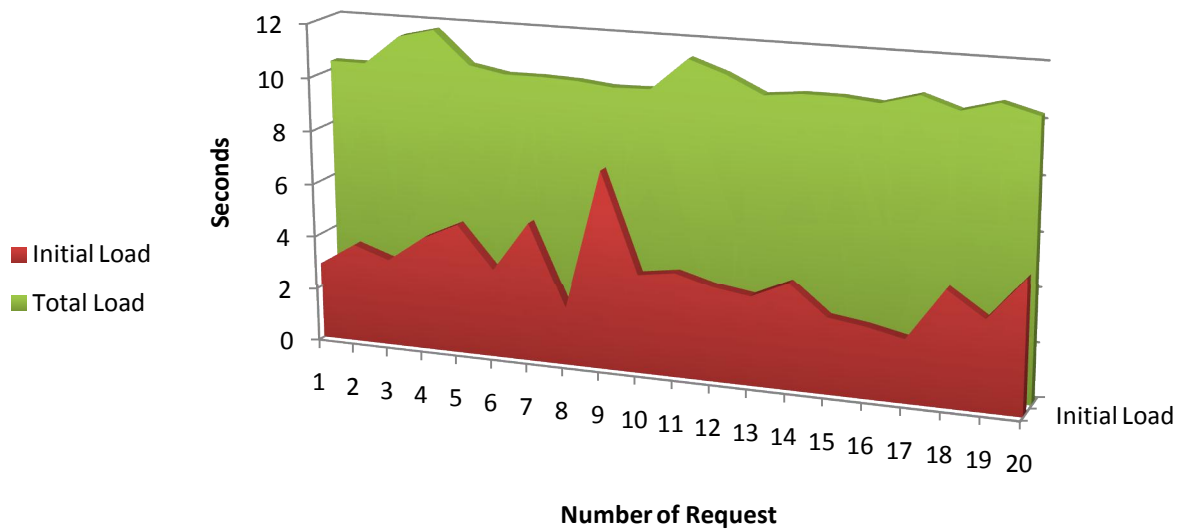
Now, we can't really judge by making a single page load, so we will try this 20 times to obtain average values.

<b>F</b>	<b>Make fewer HTTP requests</b>
<b>F</b>	<b>Use a Content Delivery Network (CDN)</b>
<b>F</b>	<b>Add Expires headers</b>
<b>A</b>	<b>Compress components with gzip</b>
<b>A</b>	<b>Put CSS at top</b>
<b>B</b>	<b>Put JavaScript at bottom</b>
<b>A</b>	<b>Avoid CSS expressions</b>
<b>n/a</b>	<b>Make JavaScript and CSS external</b>
<b>A</b>	<b>Reduce DNS lookups</b>
<b>B</b>	<b>Minify JavaScript and CSS</b>
<b>A</b>	<b>Avoid URL redirects</b>
<b>A</b>	<b>Remove duplicate JavaScript and CSS</b>
<b>F</b>	<b>Configure entity tags (ETags)</b>
<b>A</b>	<b>Make AJAX cacheable</b>
<b>A</b>	<b>Use GET for AJAX requests</b>
<b>A</b>	<b>Reduce the number of DOM elements</b>
<b>A</b>	<b>Avoid HTTP 404 (Not Found) error</b>
<b>A</b>	<b>Reduce cookie size</b>
<b>F</b>	<b>Use cookie-free domains</b>
<b>A</b>	<b>Avoid AlphaImageLoader filter</b>
<b>A</b>	<b>Do not scale images in HTML</b>
<b>A</b>	<b>Make favicon small and cacheable</b>

Here are the results; the graph for these results is displayed below:

Num	Initial Load	Total Load
1	2.88	10.32
2	3.72	10.32
3	3.27	11.42
4	4.24	11.75
5	4.88	10.54
6	3.307	10.3
7	5.195	10.31
8	2.141	10.26
9	7.387	10.12
10	3.67	10.17
11	3.87	11.36
12	3.49	10.9
13	3.29	10.26
14	3.913	10.41
15	2.818	10.41
16	2.627	10.29
17	2.333	10.68
18	4.261	10.24
19	3.347	10.62
20	4.953	10.26
<b>Average</b>	<b>3.7796</b>	<b>10.547</b>





The average initial load takes 3.78 seconds; the average total load takes 10.55 seconds. The initial load is within the acceptance limits, but we can improve this and we also want to get a good grade to leave the site as optimized as possible in the front-end.

The measurements displayed by YSlow are pretty good, but you are probably thinking that they can't really be considered as close to reality as possible because we are trying from a single location. This is why we are going to use the second tool we talked about before, [webpagetest.org](http://webpagetest.org). Using this tool will give us more accurate measurements because the tests are conducted from different locations around the world. We will again make an average of all the results, this time considering measurements from different locations in the world:

- Virginia, USA
- California, USA
- Gloucester, UK
- Jiangsu, China
- Wellington

Note: All these locations have a 1.5 Mbps ADSL connection.

Here is a screenshot of the results from the first location (East Coast USA)

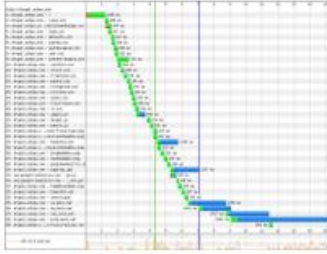
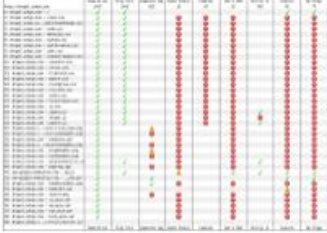

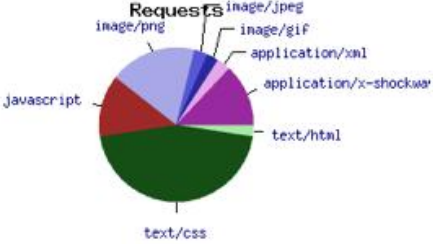
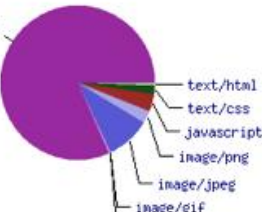
	Load Time	First Byte	Start Render	Document Complete				Fully Loaded		
				Time	Requests	Bytes In	Bandwidth	Time	Requests	Bytes In
First View	7.060s	1.194s	4.282s	7.060s	36	422 KB	1.22 Mbps	15.207s	38	873 KB

### Key Optimizations

Enable keep-alive	Compress Text	Compress Images	Cache static content	Combine js and css files	Use a CDN
A	A	D	F	F	F

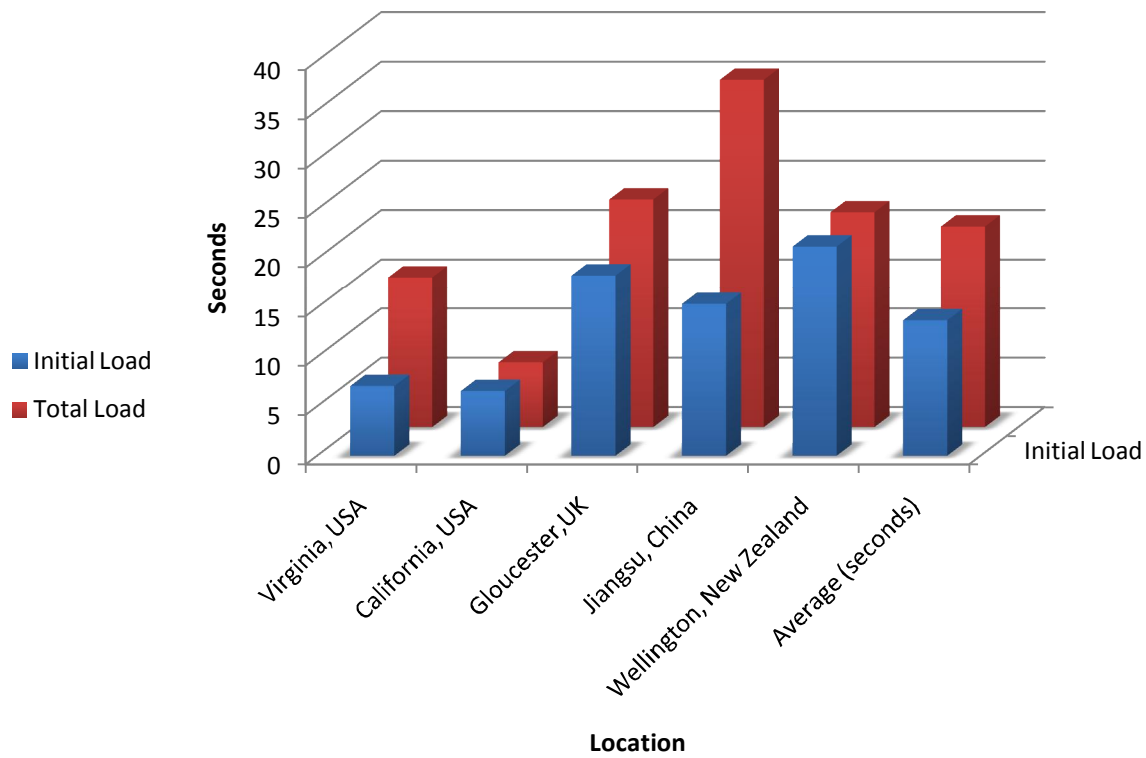
[click for optimization details](#)

### Test Results

	Waterfall	Optimization Checklist	Screen Shot
First View (7.060s)			
Content Breakdown	<div> <p><b>Requests</b></p>  </div> <div> <p><b>Bytes</b></p>  </div>		

As you can see, the average initial load of the page takes 7.06 seconds; the total load time is 15.207 seconds. This is a little higher than the results we made from

Here are the results for all the locations.



	Virginia, USA	California, USA	Gloucester, UK	Jiangsu, China	Wellington, New Zealand	Average (seconds)
Initial Load	7.06	6.554	18.308	15.382	21.249	<b>13.7106</b>
Total Load	15.207	6.554	23.109	35.194	21.807	<b>20.3742</b>

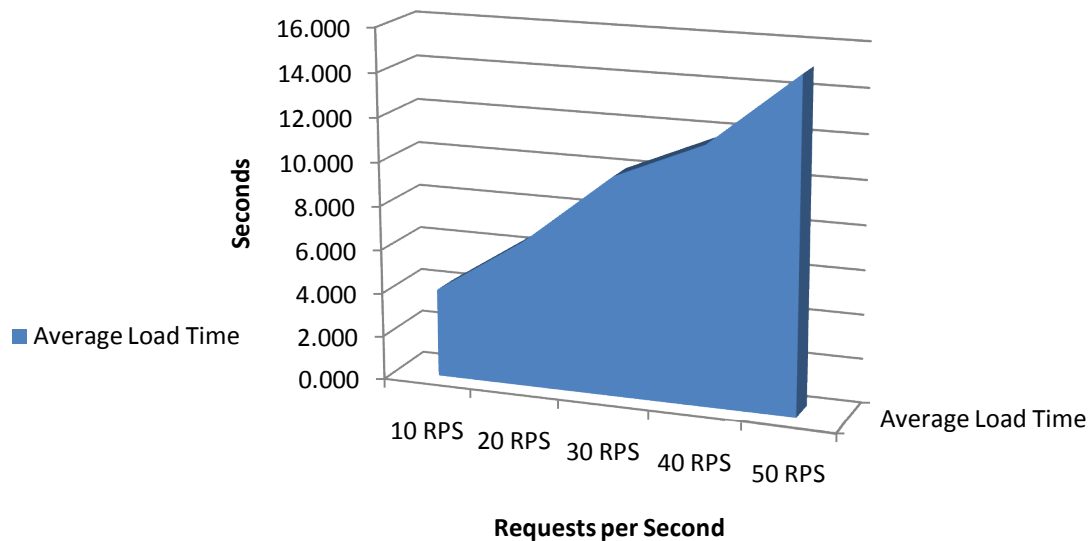
The results show that the average initial load value is **13.71 seconds**, and the average total load is **20.37 seconds**. This is above the acceptance limit (8s).

Evidently, we do need to make a few adjustments to the front-end optimization.

## Load Testing Results

To test the amount of load the site can handle, we mentioned we were going to use jMeter. This is a great tool and it's all we need for the purposes of this publication. The results after increasing the amount of RPS are the following:

### Average Load Time



	10 RPS	20 RPS	30 RPS	40 RPS	50 RPS
■ Average Load Time	4.027	6.654	9.943	11.685	15.031

The average load time when making 30 RPS is 9.943 seconds, this is above our acceptance value. We can safely say that the virtual machine, with the current configuration can only accept 20 requests per second. This value is low, but do keep in mind the server specifications and that no optimization changes have been done to the server or the site.

## Optimizing Drupal

### Front-end Optimization

Let's see how we can optimize the site to get a better load time. We are going to base our actions on the recommendations that YSlow gives us so we can get a better grade. Remember, our current overall grade is D, so let's apply all the recommendations possible and get A's on all possible criteria.

## Use a CDN

Integrating a CDN is not something we are going to do for now since it requires a considerable economic investment. It does help front-end performance enormously, but in this paper we are going to try to increase the performance of Drupal with what we have at hand.

## Make Fewer Requests

In this criterion, our grade is F because we have too many requests. Drupal can gather all the stylesheets and JavaScript files to two single requests. We do this by login in as the admin, going to **Site Configuration > Performance** and enabling the **"Optimize CSS files"** and **"Optimize JavaScript files"** in the Bandwidth Optimizations fieldset.

Bandwidth optimizations

Drupal can automatically optimize external resources like CSS and JavaScript, which can reduce both the size and number of requests made to your website. CSS files can be aggregated and compressed into a single file, while JavaScript files are aggregated (but not compressed). These optional optimizations may reduce server load, bandwidth requirements, and page loading times.

These options are disabled if you have not set up your files directory, or if your download method is set to private.

**Optimize CSS files:**

☐ Disabled  
☒ Enabled

This option can interfere with theme development and should only be enabled in a production environment.

**Optimize JavaScript files:**

☐ Disabled  
☒ Enabled

This option can interfere with module development and should only be enabled in a production environment.

Save the form and lets continue.

## Add Expires Headers

Using expires headers avoids unnecessary HTTP requests when the user visits the site more than once by making files cacheable. To add the expire headers to cacheable files, you need to first enable the mod\_expires module in Apache by uncommenting the line in the httpd.conf file that looks like this:

```
#LoadModule expires_module libexec/apache2/mod_expires.so
```

Keep in mind that the exact path of this line will vary on the different type of Linux distributions. In most cases, this module will be loaded by default, so you will not need to do anything.

In this Ubuntu server edition, module enabling is done a little differently; all we have to do is create a symbolic link of /etc/apache2/mods-available/expires.load in /etc/apache2/mods-enabled. We do this running the following command in the server:

```
sudo ln -s /etc/apache2/mods-available/expires.load  
/etc/apache2/mods-enabled/.
```

The next step is to set the expires value. If you are using a Linux distribution other than the one we are using here you might have to set this value by

adding the following line at the end of the httpd.conf file:

```
<IfModule expires_module>
    ExpiresActive On
    ExpiresDefault "access plus 6 months"
</IfModule>
```

In our distribution, instead of adding these lines to the httpd.conf file, we add those lines to an empty file we create called expires.conf. This file needs to be placed in the /etc/apache2/mods-available directory and we create a symbolic link to it on /etc/apache2/mods-enabled:

```
sudo ln -s /etc/apache2/mods-available/expires.conf
/etc/apache2/mods-enabled/.
```

Note: The directive value "access plus 6 months" means we set the expire to be added to the each file so that it caches the file for the next 6 months since the first time it was accessed.

You can restart Apache to test if this works. You should now get an A grade on the Expires Headers criteria.

## Configure Entity Tags (E-Tags)

E-Tags help the browser identify files in the cache. When a site runs on a server farm, e-tags can degrade performance because the tags can make two identical resources seem different and therefore skip caching.

Apache has e-tags enabled by default in most versions. We are going to disable them because they don't really help that much and this way we'll reduce overhead on the response. To disable the e-tags you will need to include two directives wherever your site folder <Directory> group is. This group can be in your httpd.conf file or inside your <VirtualHost> definition. These are the two lines you need to add to disable entity tags.

**Header unset ETag**

**FileETag None**

## Use Cookie-Free Domains

When the browser makes requests to the web server, it sends cookie data on the header. On static files like images, stylesheets, JavaScript files and swf files, the cookie data is not needed so these files should be placed on a separate cookie-free domain. This way we reduce the overhead on static files.

The **first step** is to create a domain that points to the same server. You can use a domain like www.static-yoursite.com. Once your domain is ready you can test it by opening it in the browser and you should be able to view your site.

The **second step** here is to create another virtual host on Apache. We are not going to get into any details of how to do this, but you can follow [this](#)

(<http://httpd.apache.org/docs/2.0/vhosts/examples.HTML>) or any tutorial online. Make sure you use the <Directory> group while creating your new virtual host. The server name and server alias should be the domain name you created in the first step and you should include in the virtual host <Directory> directive group the following lines that disabled the cookies:

```
RequestHeader unset Cookie
Header unset Set-Cookie
```

The **third and final step** is to make Drupal point all static files to your separate domain. We do this by installing the CDN Integration module (<http://drupal.com/project/cdn>) which is used to pull static content from another domain or subdomain. Install the module, go to the basic mode settings in Site Configuration > CDN Integration > Basic mode. In this screen input your static domain to the one you just created in this section's first step. In our case, the value we input is *http://oshyn-static.com*. Save the form, go to the tab called Settings and click on the Enable radio button under Status. Save the form and flush your site's cache.

We have made all the adjustments in the front-end, we will evaluate the results later on after completing all optimizations.

### Back-end Optimization

There are many ways we can improve the performance of Drupal without touching the server yet. The native optimization features help a great deal since they are caching systems within the core Drupal code.

### Page & Block Caching

By enabling page caching Drupal does not build a page each time an anonymous user visits, this will have a significant boost.

The same way Drupal can cache pages, it can cache blocks. This way all blocks won't have to be reconstructed on each page load.

Login to your admin account, go to Site Configuration > Performance. In the *Page Caching* fieldset, enable *Caching mode* to *Normal* and also enable *Page Compression*. Finally, enable *Block cache* too.

You might notice there is another property we can set, *Minimum Cache Lifetime*. This is best left alone in this case; it's useful when your site gets a big amount of traffic and content updates don't need to be "live" and can wait an hour, two hours, or even a day.

## Boost

This is a very helpful module for sites that have mostly anonymous traffic. It caches all static pages. This way when anonymous users open a content page, Drupal doesn't have to pull data from the database to render the page, it literally turns a Drupal page to a static HTML file. Download (<http://drupal.org/project/boost>) and install this module on your site, and we need to configure it.

To configure Boost properly and quickly, go to the admin/reports/status page

in your site and you will see all the things that need to be done to make this module work. First we need to prevent web crawlers from indexing the output of the module in certain files. So we add the following line at the end of the robots.txt file, located in the root folder of your site:

```
Disallow: /boost_stats.php
```

Next we need to create the following nested folders in your root folder:

*cache/perm/[yoursite.com]*

And give them writing access to Apache, run the "chmod" command from the root folder.

```
chmod -R 777 cache
```

There is an additional step we'll need to do but we can't yet. We'll do it when we tune Apache in the next section, so remember that. For now if you get a "file\_get\_contents.." warning in the admin section, then just go to Site Settings > Performance > Boost Settings and at the bottom of the form, find the *Ignore .htaccess warning* checkbox, selected and save the form. We don't need to change anything else here now. We can move on to our last step of Boost configuration, cron.

Setting up cron is really easy, we log in to the server using the admin account (not root). And we edit crontab with the following command:

```
sudo crontab -e
```

We then add the following new line:

```
0 * * * * wget -O - -q -t 1 http://yoursite.com/cron.php
```

What we are doing here is telling cron to execute a command every hour, that's all we need in our case. The command is to make an http request to the cron.php file in our site. You will need of course to replace the *yoursite.com* value for your real domain. For additional help on setting up cron, go to <http://drupal.org/cron>.

Remember, the last step to make Boost work is done in the Apache tuning section.

## Optimizing your server

So far we have done a lot of tuning, we made all possible optimizations in the front-end side, we have tuned Drupal enabling all its default caching options and we have installed the Boost module. All these changes have been done to Drupal, now let's optimize our server configuration.

### Tuning Apache



As you know, we have already done some tuning in Apache, we have setup the expire headers, disabled the entity tags and configured a cookie-free domain. The only optimization left to do is to move all the rewriting rules from your .htaccess file to your Apache configuration file. We do this so we can tell Apache not to look for .htaccess files on all folders and get a small performance improvement, everything counts right? To do this you can just paste the following code in your <Directory> directive within your host on top of the entity tag directives placed before.

```
AllowOverride None
Order allow,deny
allow from all
RewriteEngine on
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?q=$1 [L,QSA]
```

This code is the existing code in the .htaccess file. The directive *AllowOverride None* tells Apache not to look for the .htaccess files. Now, here is when we add the missing piece from the Boost module. Login as the admin in your site and go to Site configuration > Performance > Boost htaccess rule generation. Copy that code and paste it under the rewrite rules you just added to your <Directory> directive. Save the file, restart Apache and test your site so that friendly urls are still working. Also, check the cache/perm/[yoursite.com] folder you created for Boost before, if there are files there it means boost is working fine. If not, something is wrong, go back to make sure all steps were completed, use the documentation that Boost provides in their project page (<http://drupal.org/project/boost>).

This is all we need to do for Apache.

## Tuning MySQL

MySQL is probably one of the most complicated components to tune. The reason for this is because the optimal configuration is different depending on the site you have. In our case, our traffic is mostly anonymous, and users do more reads from the database than writes. To optimize our site we can do three things:

1. Hardware upgrade
2. Tune the mysql settings
3. Optimize your queries

In our case, **hardware upgrade** is not an option as we are trying to keep this performance with minimum expenses. There are a few **settings we can change to increase the performance of MySQL** for our site, since we have a lot of reads, its good to have a good amount of query cache size. Go to the MySQL settings file called my.cnf. It is usually located in /etc/my.cnf or /etc/mysql/my.cnf. You can type the following command anywhere to find the

location of yours:

```
locate my.cnf
```

Edit the file with sudo using any editor, find the *query\_cache\_size* and change it to 32M, it should look like this:

```
query_cache_size = 32M
```

This value should be changed depending on the traffic on your site, you can see if the value you input is the best by monitoring the query cache, you do this in the MySQL console. Login as the root user in the MySQL console and type:

```
mysql>SHOW STATUS LIKE 'qcache%';
```

You can monitor these values while testing your site, to see how MySQL responds.

The second change we need to make helps limit the server resources use in the MySQL process. We uncomment and change the following line in the my.cnf file:

```
max-connections=500
```

This sets the maximum number of allowed connections, its default value is 100, while testing your server, if you find that MySQL does not take a lot of your memory up and CPU, you can increase this value to allow more connections at the same time. Be careful though, if you allow too many connections MySQL may take all of your server resources and eventually crash. We set this value to 500 to see how it works.

You can see the maximum number of connections your database has handled using the following MySQL command:

```
mysql>SHOW STATUS LIKE 'max_used_connections';
```

The third and final change we are going to make in the MySQL settings is in the caches. Tables are stored as files in MySQL, this means that each table file needs to be opened for read. MySQL can leave these files open so that next time it reads them it won't need to reopen them. MySQL can leave a certain number of tables open, we can see if with this value MySQL has had the need of reopening tables. Go to the MySQL console and type:

```
mysql>SHOW STATUS LIKE 'open%tables';
```

If the "opened\_tables" value is 0, you don't need to change anything, if it's more than 0 then you should consider increasing the table\_cache value in my.cnf, that way MySQL doesn't spend much time opening table files.

There are many other changes you can make to optimize MySQL even more. In our case we won't make additional changes since our site won't benefit from all of them. We need to consider that with the caching settings we have set in Drupal and with the Boost module installed, our site won't need to make as many database reads as before. The only thing we need to make sure now is to **check if there are any slow queries** running on our site. To do this uncomment the following lines in your my.cnf file:

```
log_slow_queries          = /var/log/mysql/mysql-slow.log
long_query_time = 5
log-queries-not-using-indexes
```

Make sure you change the long\_query\_time to 5, the default is 2.

Save the file, and restart MySQL so all settings take effect:

```
sudo mysqld restart
```

While navigating through your site, you will need to monitor the mysql-slow.log file. To do this run:

```
tail -f /var/log/mysql/mysql-slow.log
```

You will then see what queries are running slow and you can optimize them.

Optimizing MySQL is probably the hardest since you have to use trial and error to get to the best settings for your site and because there are so many settings you can change. To change more advanced settings please refer to the following guide <http://www.ibm.com/developerworks/linux/library/l-tune-lamp-3.HTML>

## Tuning PHP

There are two ways to optimize PHP and both work great. One is to install APC (Alternative PHP Cache). This system caches the opcode generated when PHP is interpreted. The second way is to install Memcache, this caching system stores data objects in memory, usually results from the database. Reading from memory is much faster than reading from disk so it's a great way of optimizing the site.

## APC

The following instructions explain how to install APC on the Ubuntu server we have setup for our site. To get APC up and running, we need PECL so we can install APC from the latest repositories.

Install all required libraries from aptitude first:

```
sudo aptitude install php-pear
sudo aptitude install php5-dev
sudo aptitude install apache2-dev
```

Now we install APC using PECL:

```
sudo pecl install apc
```

Next you need to enable APC on Apache:

```
sudo echo "extension=apc.so" >
/etc/php5/apache2/conf.d/apc.ini
```

Then just restart Apache and that's it:

```
sudo /etc/init.d/apache2 restart
```

To make sure APC is working great we can use the apc.php script that comes in the package. Download and untar the latest built. At the moment of this publication APC's latest build is 3.1.3:

```
wget http://pecl.php.net/get/APC-3.1.3p1.tgz
tar xvzf APC-3.1.3p1.tgz
```

Now we will copy the apc.php file to our site folder:

```
cp APC-3.1.3p1/apc.php /var/www/.
```

After browsing through the site for the first time after APC is installed, the opcode is being generated and kept on cache. If you surf through the site again various times the opcode will be already created and the site will be faster. Here are the stats generated by the apc.php script:



The green bar displays the hits on cache.

## Memcache

The last optimization we are going to implement is memcache, it is a very simple caching system that uses memory to store objects. As you may already know, memory is faster to read than hard drive. A site will take a lot less time to read a database result cached in memory. To make the best use of memcache on Drupal, it's better to install the memcache module (<http://drupal.org/project/memcache>). First lets install the memcache binaries on Ubuntu. Doing this is as easy as executing the apt and pecl commands:

```
sudo apt-get install memcached  
sudo pecl install Memcache
```

Also, we need to enable the memcache extension on php:

```
sudo echo "extension=memcache.so" >  
/etc/php5/apache2/conf.d/memcache.ini
```

Now please follow the installation instructions in the memcache module page provided before to make memcache work with Drupal. You can start from step 3.

Once you are done we can now perform new tests to see how much the site's performance has improved.

## Evaluating the performance after optimization

We have successfully completed all the optimizations on our Drupal site. Starting with the front-end, then to the Drupal back-end and even server tuning. There are more ways to optimize even more a Drupal site. On our case, a small website with mostly static content, we have done a lot. Lets know see how helpful this was. First we are going to evaluate the single page load results and then the load testing results.

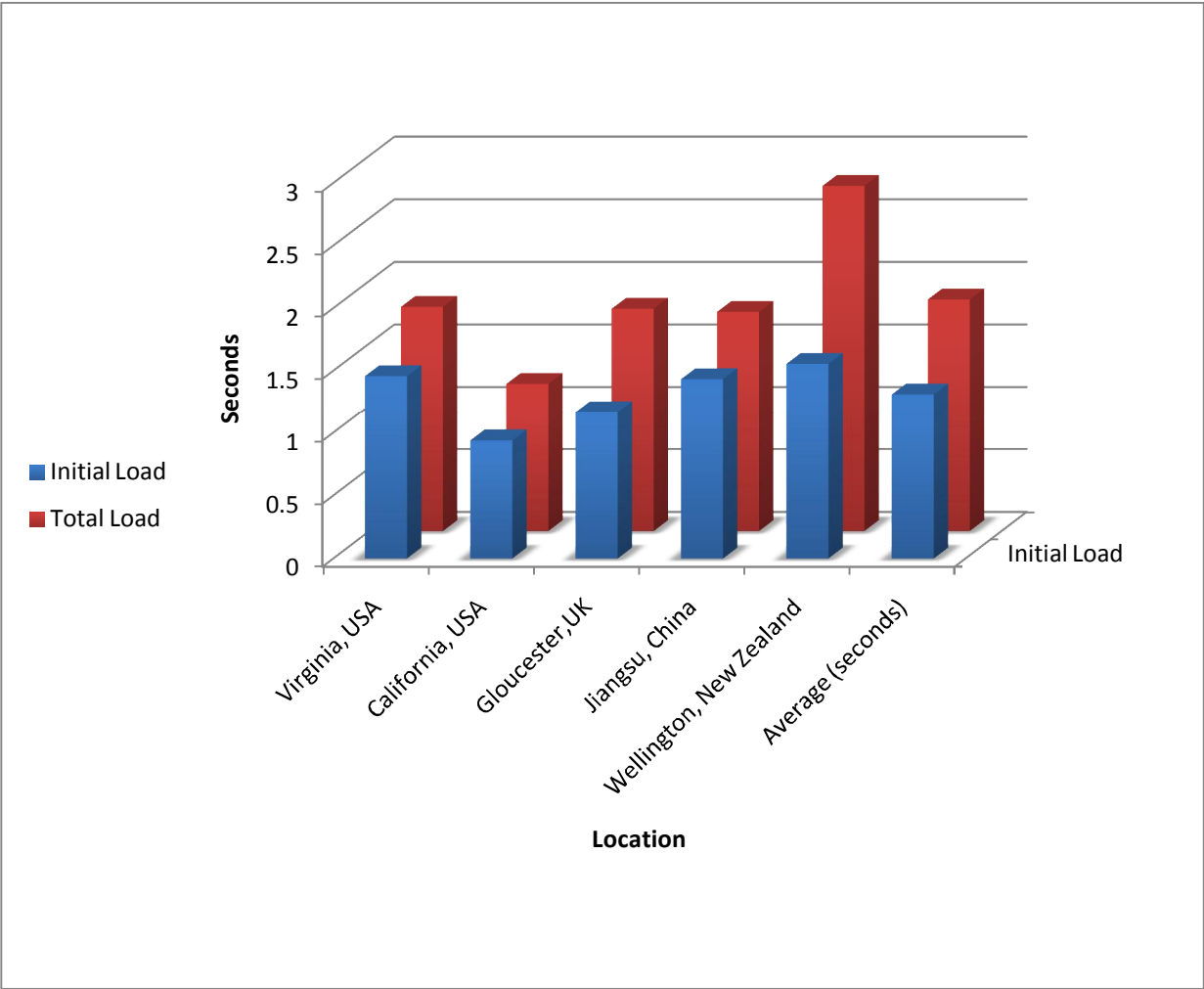
### Single Page Load Results

Going back to YSlow, you can see that our new grade is A! The total score we got was 92. We didn't get 100 because we are not using a content delivery network. Like we said before, using a CDN involves an investment, and we are trying to optimize our small site without spending.

The results are very good. Everything went well as planned. Next we are going to look into the webpagetest.org results. We make the exact same test from the different locations.

Grade <b>A</b> Overall performance score 92	
ALL (22) FILTER BY: CONTENT (6)   COOKIES (1)	
A	Make fewer HTTP requests
F	Use a Content Delivery Network (CDN)
A	Add Expires headers
A	Compress components with gzip
A	Put CSS at top
A	Put JavaScript at bottom
A	Avoid CSS expressions
n/a	Make JavaScript and CSS external
A	Reduce DNS lookups
A	Minify JavaScript and CSS
A	Avoid URL redirects
A	Remove duplicate JavaScript and CSS
A	Configure entity tags (ETags)
A	Make AJAX cacheable
A	Use GET for AJAX requests
A	Reduce the number of DOM elements
A	Avoid HTTP 404 (Not Found) error
A	Reduce cookie size
A	Use cookie-free domains
A	Avoid AlphaImageLoader filter
A	Do not scale images in HTML
A	Make favicon small and cacheable

Here are the results from the webpagetest.org site:



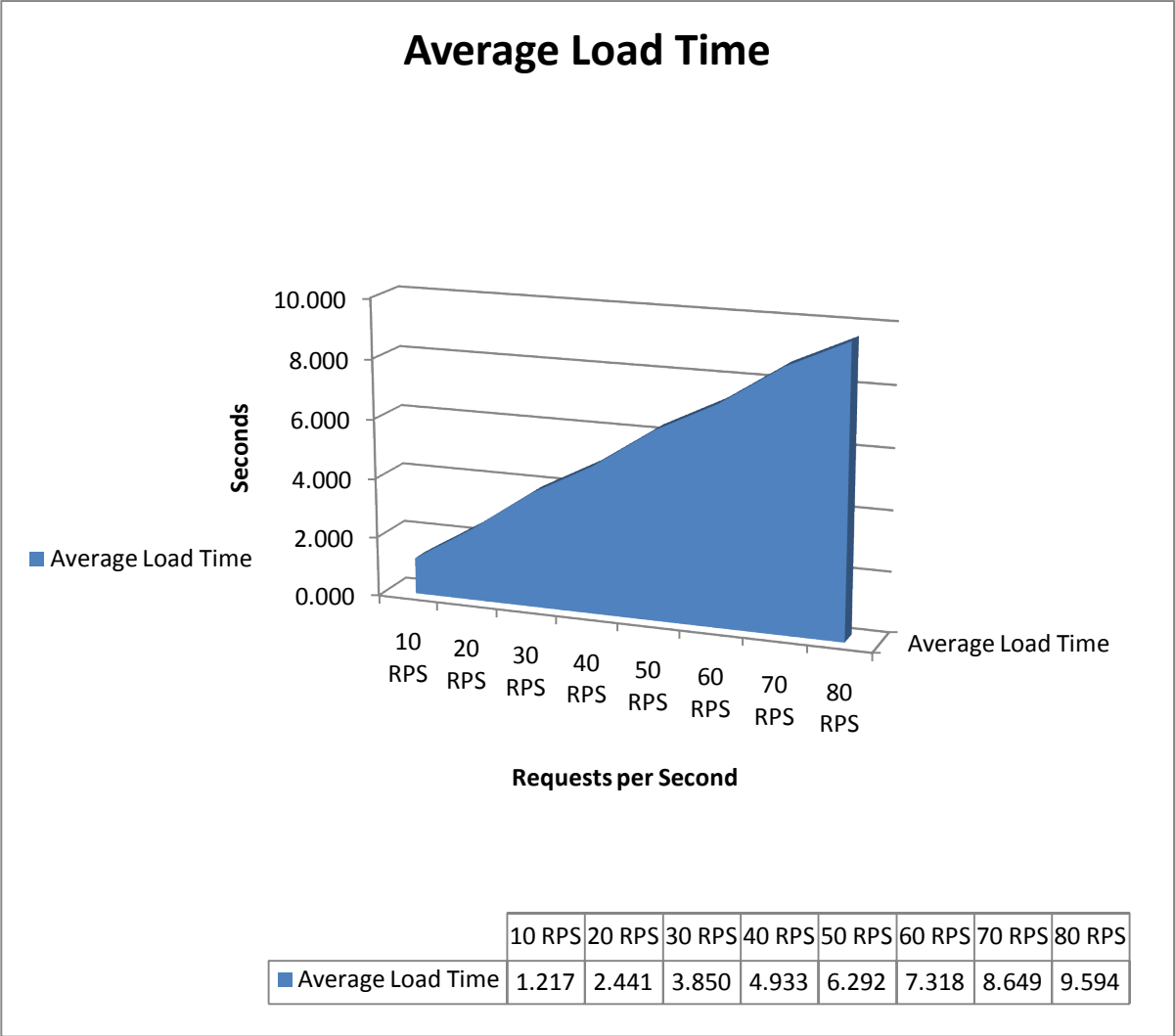
	Virginia, USA	California, USA	Gloucester, UK	Jiangsu, China	Wellington, New Zealand	Average (seconds)
Initial Load	1.457	0.938	1.168	1.433	1.555	1.3102
Total Load	1.793	1.177	1.778	1.753	2.759	1.852

Looking at these results, the single page load test value demonstrates how we have successfully decreased the average page load time globally. The

average initial load takes **1.31 seconds** and the total load time takes **1.85 seconds**.

Load Test Results

The load testing is done in the same way we did before, starting with 10 RPS until the average response time is not acceptable. Remember we are using the standard 8 seconds to determine when a page load time is not acceptable.



The total number of requests per second that our server can take after all the optimizations is 60 RPS. With 70 RPS the average load time is above 8 seconds.



## Conclusion

We have successfully optimized our Drupal site, but how helpful were all the changes we made? We get the answer by looking at the numbers. Before our changes, loading our site took an average of **13.71 seconds** from different parts of the world. After the optimization changes, the average load time is **1.31 seconds**. We have decreased the initial page load time by **90%**. The server load could take up to 20 RPS before. Now the server can take 60 RPS, which means that all the optimizations we did increased **three times** the amount of load that this virtual server can take.

After reviewing the results, we can see that without adding new hardware, we can speed up our site and the server can process more requests per second. Before expanding on infrastructure, a site should be completely optimized. In some cases, that may be enough to meet the traffic demand.

### Have CMS Needs?

Call us at  
1 (888) 483-1770 ext 100  
or email [newbusiness@oshyn.com](mailto:newbusiness@oshyn.com)  
to discuss your current CMS  
business needs and how  
Oshyn can help you achieve them.



## Drupal Success Story

### The Challenge

Mallika Chopra, the daughter of Deepak Chopra and Sal Taylor Kydd a former Yahoo! executive, wanted to launch Intent.com with the aim of it becoming the most trusted wellness destination for capturing and sharing people's personal, social, spiritual, and environmental intentions.

### The Solution

Armed with an outline of a preliminary visual design and required functionality, Intent.com selected Oshyn to design and build out the rest of the plan and execute the launch of the website.

In addition to the consumer facing website, the business development team at Intent also needed a pre-launch demo of the website to assist them in signing deals with content syndication partners, sponsors, and advertisers. Oshyn rapidly built a beta version of Intent.com to give the business development team the opportunity to showcase the website's functionality from any location while they pursued revenue and content opportunities.

Collaboration was at the heart of the

*"The customized CMS allows us to constantly deliver great content that is increasing registered members."*



Intent.com website. Therefore, Oshyn built a website that offered visitors with many ways to contribute, collaborate, and connect: user accounts, social networking platform, blog, quizzes and polls, newsletters, invitation-only feature, content syndication, podcast, and video. Intent.com needed to have the capabilities to handle many registered user bloggers and guest bloggers, most of whom were not technically savvy. Intent.com planned to aggregate content from various online sources, and to syndicate original content to many other websites. Oshyn selected the Drupal open source framework for building a Content Management System (CMS) that would allow non-technical editors to easily manage all content and syndication

### The Results

The customized CMS allows us to constantly deliver great content that is




www.oshyn.com 1-888-483-1770 ext 100 newbusiness@oshyn.com 523 W 6<sup>th</sup> Street, Suite 330, Los Angeles, CA 90015

increasing registered members.

Oshyn delivered Intent.com, a highly engaging website designed for social networking. Since the launch, Intent.com has rapidly expanded its membership base and has received much publicity. Intent.com has continued to sign new content syndication partners, sponsors, and advertisers. Intent.com is well on its way to becoming the most trusted wellness destination.

### Healthy Living




How to Be Happier  
By: Intent.com

Getting Fit 101  
By: Intent.com


How to Sleep Better  
By: Intent.com

12 Steps to Relaxation  
By: Anne Naylor | Consultant

### Intent's Guide to Sleep



Posted Tue, 06/09/2009 - 11:39  
"Tell Me Your Story" this Saturday at 3pm pdt  
From: Richard Dugan | Tags: Success  
This Saturday ... Read full Post »  
0 Comments




Posted Tue, 06/09/2009 - 11:16  
No -- Just Say It  
From: Claire Shipman & Katty Kay | Tags: Balance, Personal Space  
Once you've tamed your inner-guilt monster, you are ready to welcome that most wonderful of words into your vocabulary. We're certain you barely use it.  
... Read full Post »

Search


### Hot Topics

Happiness • Euna Lee • Laura Ling • Barack Obama • North Korea • Free Will • Freedom


### Latest Intents




from: TOWANDA ALLEN 16 min ago  
My Intent is to continue to thank God for Deepak! For the thought to create Intent.com and the love & support we provide.....  
Comments 0




from: Rajesh Sharma 30 min ago  
My Intent is god  
Comments 0



from: TOWANDA ALLEN 46 min ago  
Peace is possible if I have peace in my heart.  
Comments 2



from: vargucci 48 min ago  
My Intent is to provide my family with the love and security needed in these difficult times.  
Comments 0



from: Nancy Davis 49 min ago  
My Intent is to provide my family with the love and security needed in these difficult times.  
Comments 0

## The Technology

- Drupal Content Management System (CMS)
- PHP5
- MySQL 5
- Apache 2.0
- Red Hat Enterprise 4 Linux
- Rich content syndication model through;
- RSS badges
- AddThis
- Send to a Friend

Now imagine what **Oshyn** can do for you!

Additional Drupal Resources from Oshyn

www.oshyn.com 1-888-483-1770 ext 100 newbusiness@oshyn.com 523 W 6<sup>th</sup> Street, Suite 330, Los Angeles, CA 90015



- [Drupal Development at Oshyn](#)
- [Drupal Blog](#)
- [ICON 4x4 Drupal Case Study](#)
- [Drupal Multisite Options white paper](#)
- [Drupal Multilingual white paper](#)
- [Enterprise Drupal: Social Media, Mobile and Rich Media](#)
- [Drupal Social Media](#)

## About German Villacreces

During his 6 years in enterprise technology, Germán Villacreces has performed work for clients in the Enterprise Content Management, Digital Media, Automotive, Social Networking and Advertisement industries. His expertise ranges from rich internet web technologies with particular emphasis in content management system implementations. He has managed small teams of technology professionals to deliver medium-scale solutions in today's demanding timelines. German has a Bachelor's Degree in Computer Science from University of San Francisco of Quito.

## About Oshyn

Oshyn, Inc. is an Enterprise Technology Agency that has earned a reputation for delivering innovative business solutions for the web, mobile devices and enterprise technology platforms. Headquartered in Los Angeles, Oshyn's growing client list includes Best Buy/Geek Squad, Coca-Cola, Electronic Arts, Epson, Graduate Prospects, Fordham University, Harbor Capital, Lexus, Mars, Miramax, National Education Association, Oliver Wyman, Sapient, Scripps, Southern California Edison and Volkswagen. Oshyn, Inc. is partnered with some of the most respected agencies and technology providers such as 72andSunny, Ascentium, Crispin Porter + Bogusky, Ogilvy & Mather, Saatchi & Saatchi, Springbox, The Groop and Team One. Oshyn, Inc. partners and maintains certifications with leading technology providers such as EPiServer, Microsoft, Jahia, Microsoft, OpenText, Oracle and Sitecore. For more information please visit us at [www.oshyn.com](http://www.oshyn.com). Follow us on Twitter @Oshyn\_Inc.

