Serving 4 million page requests an hour with Magento Enterprise

Optimising Magento for Performance

Introduction

In order to better understand Magento Enterprise's capacity to serve the needs of some of our larger clients, Session Digital conducted a series of tests on how Magento performs under load. We wanted to understand Magento's performance and scaling capabilities not with just a few products and categories, but with a much larger catalogue and under the realistic peak load of one of the UK's biggest online retailers. We wanted to test with real user journeys rather than just requests to specific pages or checkout functionality, to mimic as close as is realistically possible the real world conditions of running a leading online store on Magento. It would also give us the opportunity to trial some possible optimisation and tuning options that we have been discussing for some time at Session Digital.

This white paper aims to validate Magento's Enterprise status, proving that it can meet the requirements of a very high-traffic e-commerce site with a large catalogue, even providing additional capacity for future growth. It will help e-commerce managers and technical staff to better understand the performance characteristics of the application and get an idea of the recommended infrastructure to run a large-scale Magento Enterprise store.

Following the guidelines presented in this white paper, online retailers can more accurately plan the correct infrastructure requirements of a large Magento installation. Ultimately, the white paper will help you squeeze out every last ounce of performance from Magento Enterprise.

Tools

In order to place load on the system we needed software that could generate many HTTP requests and follow a set of defined user journeys through the Magento website, building a shopping cart and checking out. The ideal solution to this is JMeter, a Java-based application designed to load test functional behaviour and measure the performance of web applications. JMeter can generate large volumes of traffic and be configured to follow a complete user journey.

So that we could monitor the load on the machines during our testing, we also needed a lightweight monitoring system that would plot the load a machine is under and also lots of additional data about the use of Memcached and MySQL etc. The ideal solution for this was Munin. Munin can be used to survey any number of machines on a network and plot the use of resources over a period of time. It is ideally suited to our needs and should be considered in a production environment to provide information for capacity planning.

Methodology

In order to test Magento we have built a cluster of servers representative of those we would use for our clients. This consists of (see the Appendix for full technical details):

- a load-balancer distributing the HTTP traffic between multiple web servers
- a single master database server with a single slave
- a single Memcached server
- a single NFS mounted asset server

We have not used a Content Delivery Network (CDN) to serve any static assets as that does not directly affect the performance measurement in this instance. JMeter does not load the assets during load testing, as these are not part of the underlying application. However, we would advocate the use of a CDN within a live system under high traffic load.

Onto these servers we deployed Magento 1.9.1.1 (the latest version at the point of testing) using the default enterprise theme and configuration. We populated this install with a single website and store, with three store views for alternate languages. To this we added 500,000 products that we generated with random data such as title, price and inventory.

Once Magento was set up and running, we generated representative traffic for the website using JMeter on a server within the cluster data centre.

Visitor Patterns

Rather than simply testing the performance of Magento Enterprise full-page cache with users hitting the home page, category, and product pages, we took this further and modelled real user journeys based on observed user patterns. We have taken the recorded traffic figures for the busiest hour from 2009 for a major online retailer and created load tests that replicate this. Here is the breakdown of the traffic we used to test Magento Enterprise performance:

Action	Views per hour
Display Product Details page	**58,000**
Homepage	**50,000**
Main Search	**70,000**
Refined Search (via facets)	**20,000**
Add to Basket	**11,000**
Login	**11,000**
Category Browse	**46,000**
Purchase	**6,000**

This peak hour resulted in 4,187 orders being placed with an average of 1.8 products in each basket – a result most online retailers would be very satisfied with.

The question we needed to answer was: can Magento support this level of traffic and how much would it cost to run?

We were confident that by flexing a little of the knowledge and experience within Session Digital we could match and indeed exceed these performance requirements with fairly moderate hardware requirements.

Testing Scenario

In order to compile a full picture of how Magento and the servers would perform under load, we ran all the test scenarios for 60 minutes.

- Run the tests with a basic Magento installation.
- Run the tests with the Magento compiler enabled.
- Run the tests with bespoke enhancements to the autoloader.
- Run the tests with Varnish reverse proxy cache.

We have used a payment method that did not require an external service to authorise the payments. This was to ensure we isolated the performance of Magento from that of any external payment gateway.

Performance Review and Modifications

In order to increase the application throughput, we attempted several code optimisations and platform configuration alternatives. Before applying any change, the application was tested in order to identify the performance of a default installation.

We could measure great improvement by enabling the Magento compiler. Different compiler solutions, such as compiling the filepath of all the included files to cache, or refactoring the autoloader using PHP 5.3 functions, did not have visible effects on performances beyond those of the Magento compiler. We highly recommend the use of the compiler on production servers.

After testing different combinations to identify which approach was providing the best result, we proceeded with a dynamic code analysis looking at how Magento retrieves information, processes data and creates the output.

After adding some more memoisation around the disk access caused by calls to the file_exists() method, we obtained a greater throughput from the application.

Finally we built a Varnish reverse proxy server and used this as a load balancer and proxy cache. Once Varnish is applied the performance that the end user experiences is greatly improved. All the Catalogue and CMS pages are cached and served directly from memory, greatly reducing the load on the web servers, as PHP and Magento are not even being executed to serve these requests.

However, it is not appropriate to cache all of the Magento pages in this way. The checkout pages should always be generated dynamically, as these display the contents of an individual's basket and need to be updated in real time. Also, once a user is logged in, there is a lot of personalised content that should be displayed. This is not appropriate for caching.

We created a simple Varnish configuration file that will respect these requirements and allow the checkout process to remain uncached.

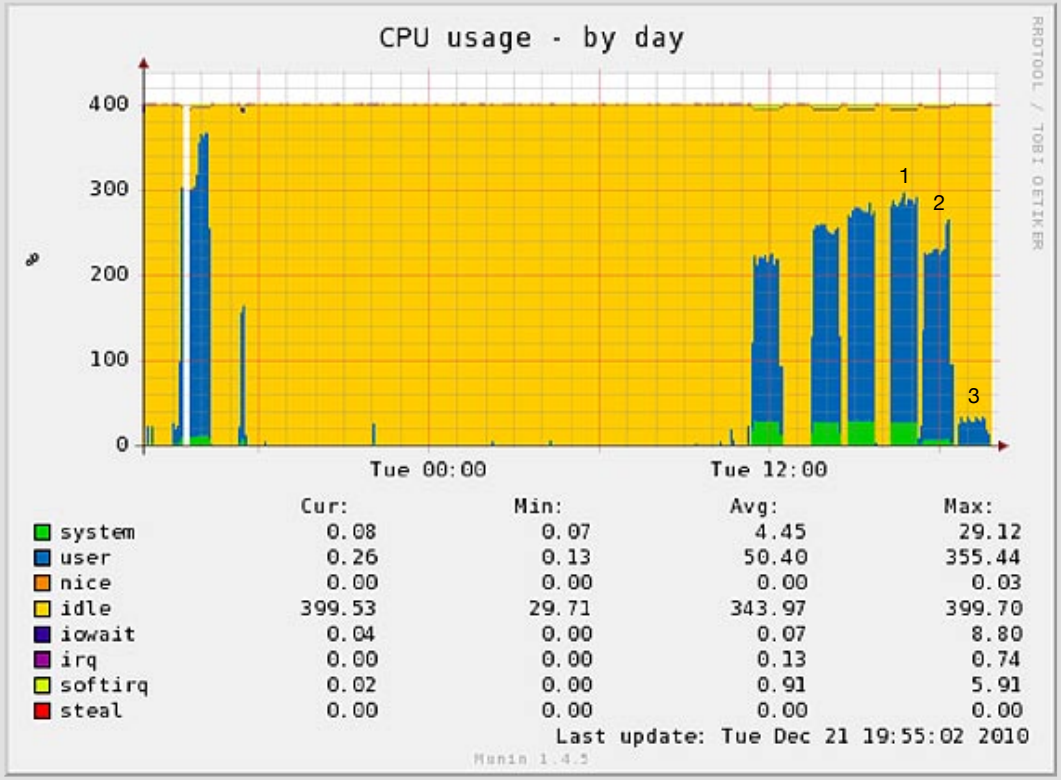
This configuration had the effect of removing nearly all the load, except users following the checkout process from the web servers and the database server.

Performance Results

JMeter captures the results of every request it makes during the hour we are running the tests. After analysing these results it shows a very definite trend. You can see from the table below that for each of the separate user journeys, with each incremental update to the system, we were able to achieve much greater throughput - achieving a total of **4.29 million page requests in an hour.**

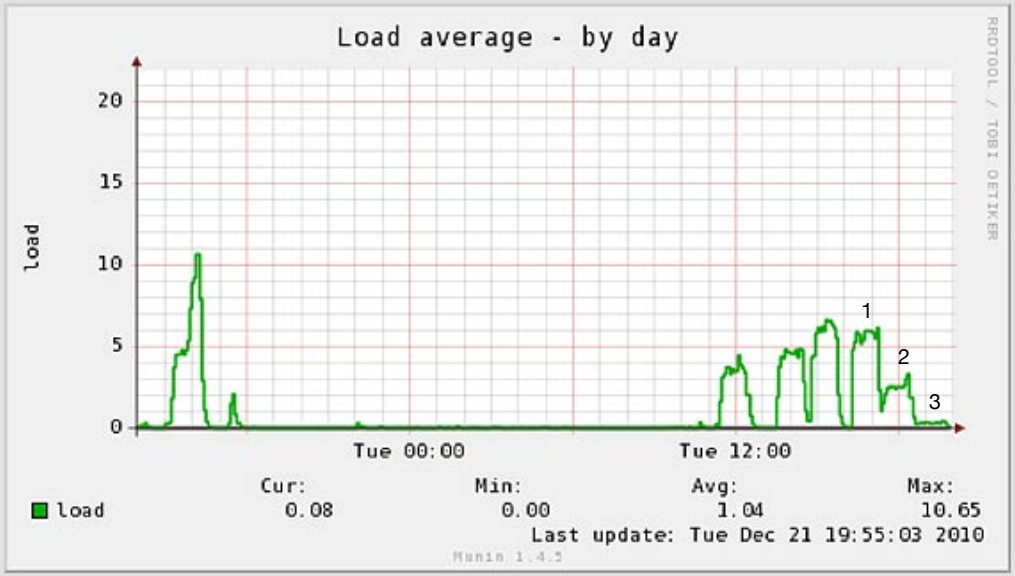
	Homepage	Category Listings	Product Detail	Add to cart	Login	Main Search	Faceted Search	Purchase	Total
Baseline	50,000	46,000	58,000	11,000	11,000	70,000	20,000	6,000	272,000
Magento Baseline	209,848	197,911	232,437	9,845	17361	79,391	0	4,897	751,690
Magento Compiler	367,740	370,167	421,163	19,831	31,034	143,714	0	9,864	1,363,513
Magento + Varnish	940,858	994,290	1,072,797	22,011	37,490	1,218,435	0	10,930	4,296,811

The Munin graph below shows how as we ramp up the traffic to the systems the CPU usage climes consistently until we start to apply some optimisations; at that point, you can see the CPU usage drop and then reach a very low figure when we applied Varnish.



1: Magento Baseline
2: Magento Compiler
3: Magento + Varnish

As expected, the system load average also shows a similar trend:



1: Magento Baseline
2: Magento Compiler
3: Magento + Varnish

CMS and Catalog Page Performance

Even without the use of Varnish, we have shown that Magento is capable when tuned correctly to serve the traffic needs of one of the UK's biggest online retailers. This required some technical know-how and some pretty extensive hardware, but the costs would be insignificant compared to the potential extra revenue it would generate.

Using Varnish reverse proxy cache with Magento allows us to remove the workload of serving the CMS and Catalog pages as they are served directly from RAM, and PHP or Magento is not even executed. Even just caching these pages for a few seconds can dramatically reduce the load on your web servers.

Example:

50,000 requests per hour with a 5 second cache would reduce your server load enormously.

Reducing the load on the hardware means that we can reduce the hardware requirement and therefore the cost.

Checkout Page Performance

Because the web servers and the database are not tied up serving the static content pages, we have seen a big increase in the throughput of the checkout process. Remember, we applied no caching to the checkout process or to users who had logged in.

Magento Admin Performance

During testing we did not focus on the admin interface performance. For a store with this volume of traffic, we would recommend that the Magento administration be run from an individual server separated from those running the public store. This will ensure that administrators can always access the system even when there is peak customer load. We would also recommend that any cron tasks be run on this machine, ensuring that the cron tasks do not slow down the system for customers who are purchasing from the store.

We did, however, feel that it was appropriate to test the performance of the two main areas of Magento administration that are directly affected by the size of the product catalogue: Product Import and Index Processes.

Product Import

The Magento product import is renowned for being a little slow (this has been dramatically improved in the latest 1.10 release). So we have created a number of scripts that convert the import into a single SQL statement that can be executed much more quickly. Creating the SQL for 500,000 products takes approximately 40 minutes depending upon the complexity of the product. Running the SQL to insert the products takes less than 10 minutes.

By using this method to insert products, we have prevented the auto-indexing of products from running. However, we recommend that you turn off auto-indexing ahead of any major product import anyway, running the indexing as a separate process after you have completed your product import.

Index Processes

The indexing process can be very slow initially. In fact, when we ran the indexing the first time with 500,000 products, the process was still running over 24 hours later. We knew it would take a while, but this was far greater than we expected to re-index the entire catalogue for 3 store views. With a small amount of investigation and expertise within Session Digital, we identified the root cause of this performance bottleneck: one of the tables used for the indexing did not have suitable indexes that would support this operation. This was causing MySQL to create a huge temporary table in memory / scratch storage in order to execute the indexing query. The fix to this was simple: by running the sql EXPLAIN statement for some of the queries involved in the indexing we could identify the appropriate indexes to add. After applying this patch to the database schema the indexing process completed in less than 2 hours.

Index	Start Time	End Time	Duration (Minutes)
catalog_product_attribute	17 Dec 2010 13:37	17 Dec 2010 13:48	11
catalog_product_price	17 Dec 2010 13:48	17 Dec 2010 13:56	8
catalog_url	17 Dec 2010 13:56	17 Dec 2010 15:24	88
catalog_product_flat	17 Dec 2010 15:24	17 Dec 2010 15:42	18
catalog_category_flat	17 Dec 2010 15:42	17 Dec 2010 15:42	0
catalog_category_product	17 Dec 2010 15:42	17 Dec 2010 16:01	19
catalogsearch_fulltext	17 Dec 2010 16:01	17 Dec 2010 16:31	30
cataloginventory_stock	17 Dec 2010 13:36	17 Dec 2010 13:37	1
tag_summary	17 Dec 2010 16:31	17 Dec 2010 16:31	0

This patch has been submitted to Magento and should hopefully be part of a future release.

Conclusions

Magento is a large PHP application and requires fairly substantial server resources in order to reach the absolute best performance. It is primarily CPU-bound and requires a large amount of RAM to be available to perform its operations.
This, however, is fairly typical of most large PHP applications, and with appropriate infrastructure and optimised configuration Magento can provide performance that will match most other platforms.

Magento can meet the traffic requirements of an extremely high-traffic e-commerce website when scaled horizontally with hardware or into the cloud. The hardware costs for such infrastructure are relatively small.

The application of appropriate technology, such as a Varnish reverse proxy server, can dramatically reduce the hardware requirements. Although we ran our benchmarks with a cluster of 7 web servers, we believe that we could safely support the same levels of traffic on just 4 web servers. We could probably reduce this even further, but we wish to factor in some redundancy for a site of this nature.

Some appropriate bespoke enhancements can further improve this performance. We are constantly evaluating work from other projects and applying appropriate performance enhancements to the Magento stores we implement.

The Varnish configuration that we used for these benchmarks was fairly rudimentary and could be improved in a number of ways. For example, introducing Edge Side Includes ESI would allow more of the store pages to be cached and remove the requirement to stop caching for logged in users.

Appendix

Key Magento Configuration

Magento flat catalog: Enabled

Magento flat categories: Enabled

Magento compiler: Enabled

Magento Enterprise page cache: Enabled

Hardware Specification

Web Servers

7 Web heads

Memory: 12 GB Memory Processor(s): 1 #Processors 4 #Cores per Proc
Hard Drive(s): 2 x 15000 HDD RPM 146 GB Hard Drive

Linux OS: Red Hat Enterprise Linux 5 - 64 bit RAID Configuration: RAID 1

Database Servers

1 Master and 1 Slave (Only really needed for data integrity and fail over?)

Memory: 24 GB Memory Processor(s): 4 #Cores per Proc 2 #Processors
Hard Drive(s): 4 x 15000 HDD RPM 146 GB Hard Drive

Linux OS: Red Hat Enterprise Linux 5 - 64 bit RAID Configuration: RAID 10

Hardware Costs

This hardware was generously donated by Rackspace for the duration of our testing. Rackspace are our preferred hosting partner for Magento Enterprise customers, as they provide support and technical knowledge that is unparalleled.

This hardware would normally cost on average:

Annual Costs	
Magento License	1 x £8,500 per year 3 x £6,500 per year
Total Annual Cost	***£28,000***
Monthly Hosting Costs	
Web Servers	4 x £822.80 per month
Database Servers	2 x £774.80 per month
Varnish / Load Balancer Box	1 x £1,259 per month
Total Monthly Hosting Cost	***£6,039.80***

About Session Digital

Session Digital works closely with global retailers and emerging brands to deliver complex e-commerce sites on the Magento Enterprise platform that not only look fantastic but also work brilliantly.

With a London-based operation of over 50 – including highly skilled software engineers and an industry recognised creative team – Session Digital has the knowledge, expertise and track record to deliver and support next-generation e-commerce projects.

The team has worked closely with Magento since 2009 and has a prestigious and growing roster of established clients from the world of fashion, music, finance, entertainment and technology.

www.sessiondigital.com

.session