OPTIMIZING MAGENTO WHITE PAPER: BENCHMARKING PHP 5.3 and 5.4

Table of Contents

INTRODUCTION

This white paper compares the performance of the Magento ecommerce platform running under PHP 5.3 and 5.4. In March 2013, PHP 5.3 entered its end of life cycle. Moving forward, the PHP team plans to release only critical fixes for the 5.3 branch[b] and urges all users to upgrade to version 5.4.

These upgrades present challenges to users by introducing bugs, changing application behaviors, and causing unexpected[c] resource use. While the upgrade is inevitable, advance knowledge of these potential challenges can put you in control of the timing and logistics. How do you know when it's time to upgrade? Organizations face risks regardless of the choice made. If you are a Magento user weighing your PHP options, the information available does not paint a clear picture of the new version's benefits. Some users report speed increases and some report decreased memory footprints. However, comprehensive, methodical test results on full enterprise Magento installations have not been available.

Copious uses Magento and PHP a great deal, so we wanted to address the upgrade question for clients. The upgrade is potentially time consuming, so we wanted to know what performance benefits we will see. This paper captures the results from controlled tests of Magento on PHP 5.3 and 5.4 to determine performance differences.

Differences Between PHP 5.3 & 5.4

PHP 5.4 introduced a number of new features. Among them are enhancements to the language such as new array syntax and improvements to variable scoping in closures. The release notes also mention "improved performance and reduced memory consumption."

Some changes will break compatibility with PHP 5.3 applications. The release notes go into more detail, but most of these language features were deprecated in 5.3 and have been removed in 5.4.

BENCHMARKING MAGENTO

We wanted to understand an upgrade's impact on the performance of a Magento application. Through vigorous benchmarking, it is possible to track resource allocation and performance differences and pinpoint trouble areas. The end result will be a comprehensive view of the performance differences between PHP 5.3 and 5.4.

We tested for responsiveness and the number of requests Magento could successfully return because those measurements directly affect the user experience. Magento's value comes from converting users browsing your online store into paying customers. To do this, it needs to quickly and efficiently serve as many product detail pages (PDPs) as possible. Magento's shopping cart also needs to be responsive and fast so we don't keep users waiting. Long waits lead to abandoned carts.

What is Important to Magento

Magento can consume a lot of computing resources. For best performance, your PHP and system configurations need to allow Magento to use the disk and memory it needs. Here are some options we recommend to keep it performing as well as possible.

Disable as many unused modules and features as possible

Enable flat catalogs
Enable CSS and JS compilation
Cache blocks with the core_block_abstract_to_html_before event
Increase the system/page_cache/max_cache_size and system/page_cache/allowed_depth as large as possible
Move as many actions as possible to automation routines, background processing, and out-of-band systems
Minimize the count of active promotional rules
Set used_in_product_listing on attributes necessary for category and product pages
Minimize calls to Mage::getModel('catalog/product')->load()

TESTING ENVIRONMENT CONFIGURATION

The test environment consisted of one web server and one database server. The web server was an 8-core Intel Xeon X3460 with 32GB of RAM, running PHP-FPM through nginx 1.1.19. The database server had four Xeon E5645 processors, for a total of 24 cores, with 32GB of RAM. It ran MariaDB 10.0.10 and Redis 2.8.8.

We tested two PHP versions: 5.3.10 and 5.4.26. Both used the APC opcode cache. These were

used to power Magento Enterprise Edition 1.13.1 with all recommended patches applied. We configured Magento with the default enterprise theme. All default features and all framework caches were enabled. All Magento CSS and JavaScript were compiled and the Magento cron jobs ran at 5 minute intervals. To avoid external network latency, we didn't enable any third party shipping or payment services. HTTPS was also disabled.

The MariaDB database was loaded with a site catalog of 2 million products. Magento's own Expert Consulting Group provided this catalog. For a complete description of the testing environment, please see Appendix A[d].

To simulate real world loads, we developed a custom Bash script with cURL 7.22.0. The script simulated a user adding 5 products to his cart and proceeding through checkout. The simulated user approximated real-world throughput and browsing behavior, including pausing between page loads and actions. The load generated by the test scripts would be slightly greater than the projected load of a well-planned site experiencing their busiest shopping day of the year. It is the load of a very good Cyber Monday.

In addition, we load tested various elements of the Magento site using ApacheBench 2.3. This allows us to measure raw page-load times without simulating an entire shopping cart experience. Though the cart is where customers spend their money, it's the product display pages that transform browsing users into customers. The testing scripts are more fully discussed in Appendix B.

VALIDATION OF THE TESTING ENVIRONMENT

In order to prove that the test environment would produce usable results, we laid out parameters for each test. Each test assertion would be proven by test profiles lasting at least 10 minutes. Final assertions would be backed by constant test profiles running for a minimum of 60 minutes. The length of these testing periods ensures that the final data is not skewed by accidentally allowing the test period to fall within the "burst" capacity of the webserver and database server architecture.

In addition, network errors should account for fewer than 0.001% of all requests. On the database server, the thread cache hit rate should be greater than 99.999% and table lock requests granted without waiting should be greater than 99.9999%. This would ensure that all

hardware and software configuration was operating as intended.

The testing environment was validated using load tests with Siege 2.70 and ApacheBench 2.3. A valid test involved flushing all system caches, generating a sitemap, and hitting each sitemap URL three times with Siege. Finally, the load test would run, generating a valid benchmark.

Testing Process

Each load test followed this procedure.

Generate sitemap

Hit all sitemap URLs three times with Siege

Store current time and current order ID

Run load test

Identify end time and current order ID

COLLECTING THE RESULTS

The validation benchmarks produced by Siege gave interesting numbers. PHP 5.4 showed a 27% improvement in requests per minute over PHP 5.3 and a corresponding drop in average response time. However, as this was only a straight load test without any simulation of actual user activity, the numbers aren't enough to draw useful conclusions. All we can see is that PHP 5.4 appears to be slightly faster. We don't get enough metrics to know if the newer version of PHP will provide noticeable benefits in a full ecommerce website.

Table 1: Benchmarks from Siege load test

PHP Version	Transactions Per Second	Average Response Time
5.3.10	9.08	2.20
5.4.26	11.54	1.73

The simulated user benchmarks, however, produced interesting results that merit more discussion. Some measurements were simply the requests per minute throughput for specific types of pages: a standard PDP, a PDP with a simple promotion attached, a category page, and the home page. Finally, and most importantly, the benchmark measured the number of orders completed per hour.

Table 2: Simulated user benchmark results

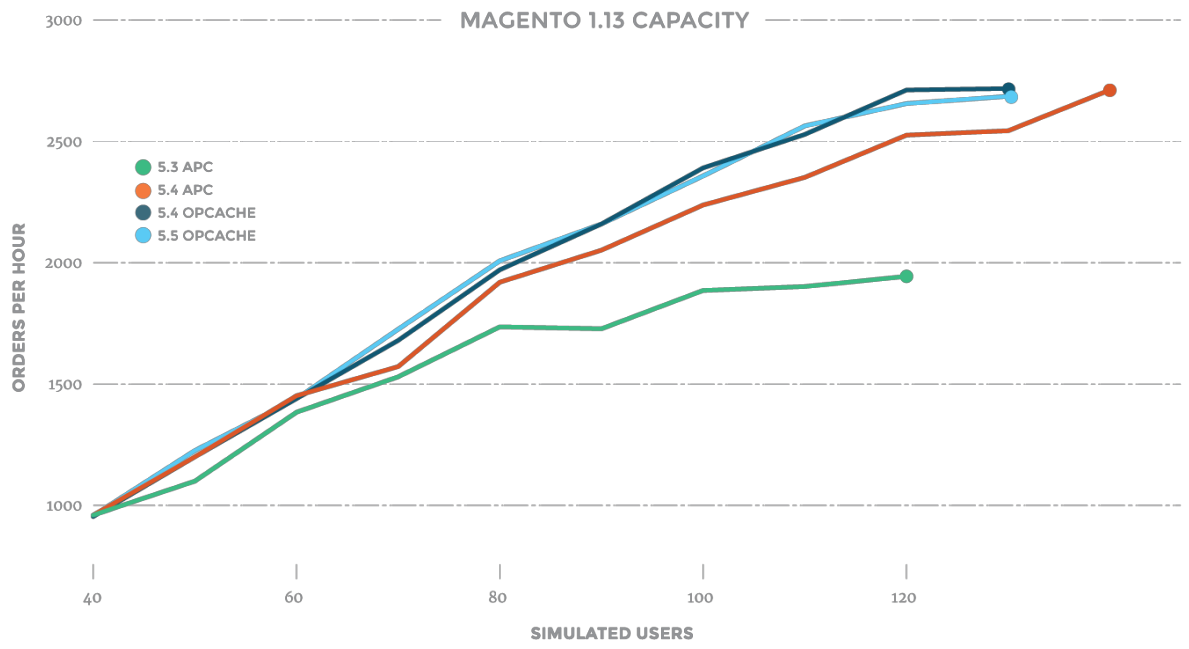
	PHP 5.3.10	PHP 5.4.26
Standard PDP	20,413	22,730
PDP with Promotion	19,760	22,745
Category Page	13,784	15,784
Home Page	18,930	21,530
Orders per Hour	1,944	2,718

PHP 5.4 Wins Every Time

In every case, PHP 5.4 outperformed PHP 5.3 in terms of raw page-loads and orders completed. However, this is not the whole story. In addition, webserver processes running PHP 5.4 used only 85MB of RAM each, beating out PHP 5.3 at 97MB each.

PHP 5.3 was able to handle 8 Mb/s of data on its user-facing network interface and was able to push 50 Mb/s on the network interface linked to the database server. Again, PHP 5.4 outperformed it, with transaction rates of 10 Mb/s on the user-facing interface and 60 Mb/s on the database interface. Because it faced fewer bottlenecks, PHP 5.4 consumed significantly more resources, with 25% more network bandwidth consumed, 25% more disk I/O throughput, and 12.4% less RAM utilization.

The graph below shows performance over time for Magento running under PHP 5.3 and 5.4. The results clearly show the increased throughput and response time of PHP 5.4.



This means that the end-of-life of PHP 5.3 will have a huge impact on development and maintenance time. Though an upgrade to 5.4 might mean some short-term challenges, it means continued support into 2015.

WHAT DID WE LEARN?

PHP 5.4 is faster and leaner. That's great news for everybody. With more efficient resource usage, the same server cluster can handle greater throughput. However, because of the changes in the way resources are used, it is important to evaluate hardware configuration to make sure that heavy load won't cause unexpected resource contention. Heavy load may look better on the webserver end, but additional concurrent threads can lead to resource exhaustion on the database server.

IT'S NOT JUST THE SPEED

Speed, improved throughput, and reduced resource consumption are not the only reasons to upgrade. Future releases of 3rd party libraries and modules will be designed for modern versions of PHP. Important patches for performance, security, and critical bugs might not be available for PHP 5.3. Avoiding the upgrade means that maintenance costs will increase in the future as developers backport patches for compatibility.

This means that the end-of-life of PHP 5.3 will have a huge impact on development and maintenance time. Though an upgrade to 5.4 might mean some short-term challenges, it means continued support into 2015.

New language features will change the way developers address problems. Traits, function array de-referencing, and improved closure bindings all give developers new tools. Source code will become more elegant and easier to understand, but it will be incompatible with versions of PHP older than 5.4.

From a business standpoint, the biggest benefit of upgrading is an increased return on investment. The same amount of hardware can do more work, which means that expanding infrastructure to support future growth won't cost as much. At the same time, lower latency and greater throughput mean more completed orders on a Magento application. Repeated studies show that web application performance heavily impacts conversions.

Technical Benefits of PHP 5.4 over 5.3

Faster software

Less resource usage

Smooth upgrade path for 3rd party modules

Continued support until 2015

New language features

Business Benefits

Better ROI on existing hardware

Improved performance means more conversions

Easier to find new hires

CONCLUSION

Every Magento developer wants his application to attract and engage with customers. Your plans should be converting more users into customers. To reduce website latency and increase customer satisfaction, you need speed and throughput. You need to cut hardware resource consumption in order to increase the ROI of your existing and future server investment.
Our testing revealed that PHP 5.4 outperforms 5.3 in speed, memory usage, and resource utilization. In many cases, such as maximum orders per hour and I/O improvements, the improvements are dramatic.

That means it's not a question of whether or not an upgrade will happen; it's a question of when and how. The numbers show that upgrading as soon as possible is the best path.
A successful upgrade needs testing and evaluation of all system components and a risk analysis. Any upgrade also needs a rollback plan in case elements of a production environment don't match the staging or testing environments.

Finally, if your developers are unfamiliar with the new features of PHP 5.4, now is the time to have them learn about new possibilities. Incorporating these new features can make your software more elegant and attractive and can increase its maintainability.

PHP continues to be a dynamic and relevant technology solution. As Magento continues to evolve, future versions of PHP will become supported. Learning about how these PHP upgrades can affect your technology and business is crucial to staying on the leading edge of ecommerce.

APPENDIX A: DETAILED DESCRIPTION OF TEST ENVIRONMENT

Server Configuration

Web Server

- IBM x3250 type 4252
- two Xeon x3460 CPUs 2.80GHz
- 32GB RAM
- Ubuntu 12.04 LTS 64-bit
- Storage: mechanical disks
- Kernel: 3.8.0-38-generic

Database Server

- IBM x3650 M3 type 7945-AC1
- four Xeon x5645 CPUs 2.40GHz
- 32GB RAM
- Ubuntu 12.04 LTS 64-bit
- Storage: mechanical, RAID 10
- Kernel: 3.8.0-38-generic

Additional Hardware

- Firewall: FortiGate800D (High Availability pairing)
- Switch: Cisco 2960S48TS-L (High Availability pairing and LACP)
- Software Configuration
- Magento configuration
- Version 1.13.1 with recommended patches
- all Enterprise Edition default features enabled
- default theme
- all caches enabled
- CSS and JS compiled
- Cron jobs running at 5 minute intervals

PHP Versions

- 5.4.26-1+deb.sury.org~precise+1
- 5.3.10-1ubuntu3.11 (20090626)
- Webserver software
- nginx version: 1.1.19

Additional Packages

- build-essential
- unzip
- php5-common
- php5-mysqlnd
- php5-fpm
- php5-cli
- autoconf2.13
- autoconf
- automake1.9
- php-pear
- php5-dev
- php5-curl
- php5-gd
- php5-mcrypt
- libhiredis0.10
- libmysqlclient18
- mysql-common
- git
- mysql-client-core-5.5

PHP Modules

- phpredis
- LZF
- APC

- OPcache

Database Server

- MariaDB 10.0.10 Stable 2014-03-31
- Magento's Expert Consulting Group provided magento-vanilla-1.13.0.2-big-simple (2 million SKUs)

Caching Software

- Redis 2.8.8

Testing Tools

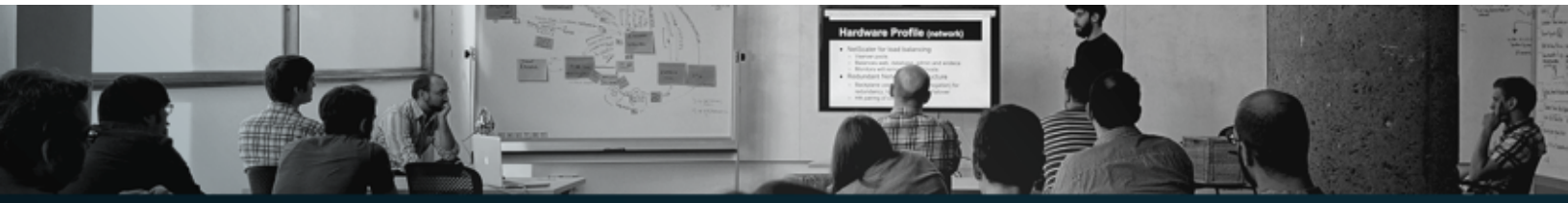
- Siege 2.70
- ApacheBench2.3
- GNU bash 4.2.25(1)-release
- curl 7.22.0

Commands to Validate Environment Behaviors

- siege -v -c 10 -i -b -t 15m -f target-urls.txt
- ab -n 100000 -c 1000 -H 'Accept-Encoding: gzip, deflate' http://example.org/

APPENDIX B: BENCHMARKING SCRIPTS

The script used to generate the transactions per hour measurement is written in Bash. It runs approximately 40 distinct web requests using curl. The script is available in the following Github repository: https://github.com/copious/cop_magento_tests



COPIOUS

Copious is a digital agency that crafts engaging experiences that drive meaningful transactions.

411 SW 6th Avenue Portland, OR 97204 www.copio.us • hello@copio.us • 503-255-1822