Recommended Magento Configuration for High Order Volumes

Summary

This paper documents the performance that can be achieved by properly configuring Magento Enterprise Edition. An optimized configuration includes the use of common caching schemes such as Varnish and Nginx along with appropriate hardware. With an optimized configuration, Magento is able to support over 350 million catalog views and 487,000 orders per day with improved response times.

Comparing Configurations

With the goal of optimizing Magento Enterprise Edition for peak performance, the Magento team set out to identify the optimal hardware and software configuration for a standard 2+1 (two nodes plus one database) setup.

For comparison, three different configurations are listed below:

- Standard: Apache + MySQL — The most widely used configuration, combining the Web's most popular HTTP server and open-source database.
- Cache Optimized with Varnish: Varnish + Apache + Percona MySQL — An optimized configuration using Varnish (an open-source reverse proxy server), Apache, and Percona's enhanced version of MySQL.
- Cache Optimized with Nginx: Nginx + PHP-FPM + Percona MySQL — An optimized configuration of Nginx (a combined web server, caching proxy, and load balancing solution), PHP-FPM (the PHP FastCGI process manager), and Percona's enhanced version of MySQL.

The results of testing—detailed on the following pages—demonstrate that optimizing 2+1 configurations using Varnish or Nginx can lead to significant increases in performance.

Performance Testing Methodology

This test scenario models the typical behavior of visitors and buyers—viewing products and categories, adding them to their shopping carts, and then submitting orders.

The following eCommerce parameters were set in each test scenario:

- Number of websites: 1
- Number of stores and store views: 1
- Number of categories and products: 50 categories/100,000 products
- Users per simulation: 2,800
- Conversion rate ranging from 3% – 6%
- Concurrency rate: 100 catalog page views/sec

Two test scenarios were used, each of which simulated different user activities.

The first scenario attempted to serve the maximum number of visitors and buyers performing the following activities:

- Navigating the Magento Enterprise Edition catalog and product pages (visitors).
- Opening category pages, adding products to the shopping cart, and placing orders via one-page checkout as non-logged-in users (buyers).

The second scenario served customer visits only. The goal was to serve as many catalog views from one Web server as possible.

Hardware & Software Specifications

To test Magento Enterprise Edition with different hardware/software configurations, four physical servers (each having a similar hardware configuration) were used.

Hardware

- CPU: 2 x Intel® Xeon® CPU E5645 @ 2.40GHz
- HDD: RAID1 - LSI MegaRAID SAS 9260-4i; 2 x SAS 164GB 15,000rpm
- RAM: 24GB ECC
- Network Interface Card: Intel 1GB 82576

Software

The following software was installed on the performance testing servers:

- Magento Enterprise Edition 1.11.2.0
- Apache 2.2.15, mpm_prefork, mod_php
- PHP 5.3.10
- MySQL 5.5
- Pecl memcache 3.0.6
- Varnish 2.1.5
- Nginx 1.0.15
- PHP-FPM 5.3.10
- Percona MySQL 5.5
- Memcached 1.4.4
- Operating system: Red Hat Enterprise Linux version 6.2 (Santiago) x86_64

Testing the Standard Configuration: Apache with MySQL

The combination of Apache and MySQL is the standard Magento Enterprise Edition configuration.

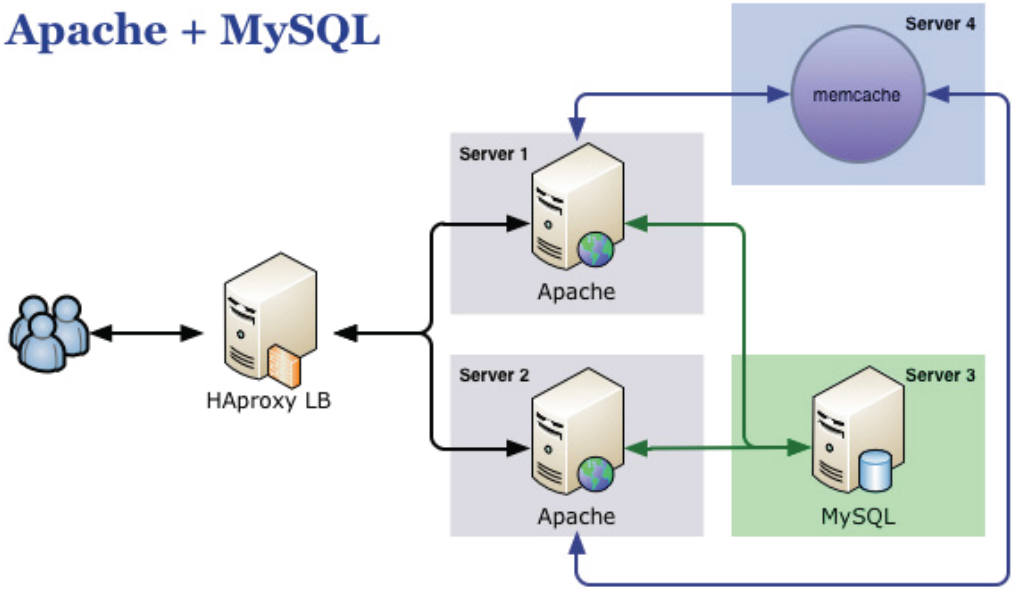

Figure 1: Standard Infrastructure: Apache + MySQL

Figure 1 depicts the following servers and software components:

Servers 1 and 2 — Two application servers; each runs the Apache web server, with mod_php installed; each Apache web server handles both static files and dynamic requests. HAProxy balances requests between each Apache instance

Server 3 — MySQL database server

Server 4 — Memcache key/value storage, which stores session data, temporary data, and rendered blocks

Summary of Results:

- Number of orders (clients): 2,800
- Time required to process all orders: 900 seconds
- Orders/per sec (average): 3.11
- Response time (average): 2.11 sec
- Web server load: 100%
- Concurrent visitors: 100 page views/sec

With the Standard configuration, the system was able to serve approximately 268,700 orders and 79,500,000 catalog views per day.

Testing Cache Optimized with Varnish

To improve the software/hardware architecture, the standard MySQL database server was replaced with Percona MySQL and we included the Varnish caching HTTP reverse proxy.

Varnish accelerates web applications by caching most of the static and dynamic content for them. In the caching architecture, Varnish is installed immediately in front of the application server that “speaks” HTTP. The Varnish Configuration Language (VCL) supports writing flexible rules for handling incoming requests and outgoing responses.

For this configuration, Percona enhances MySQL in the following ways:

- Fine-grained mutex locking
- Lock-free algorithms
- Buffer pool pre-load
- Read-ahead improvements
- Shared-memory buffer pool

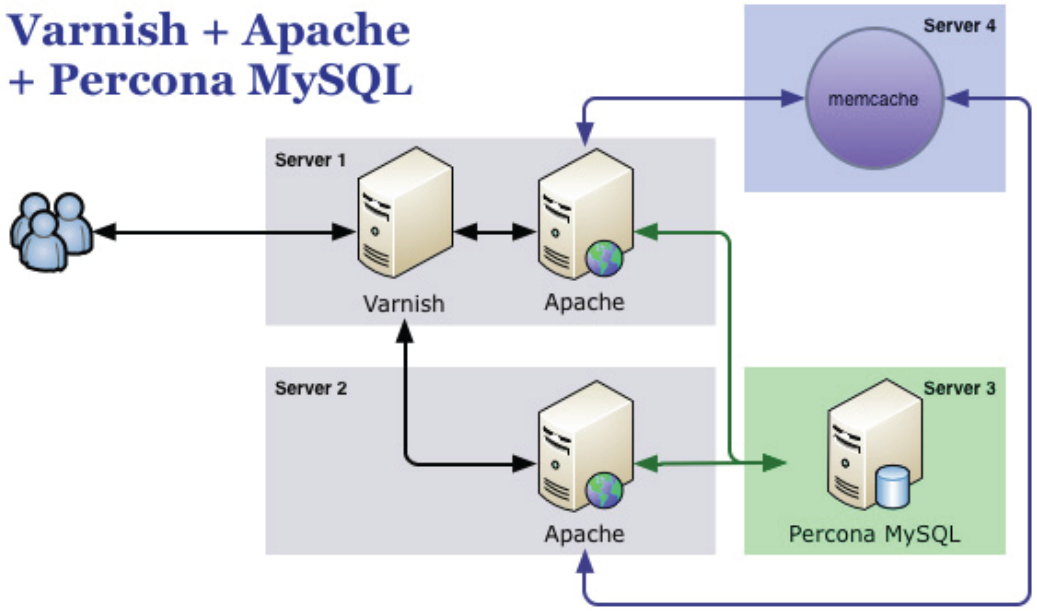

Figure 2: Cache Optimized with Varnish (Varnish + Apache + Percona MySQL)

The servers shown in Figure 4 have the following components installed:

Server 1 — Reverse proxy server running Varnish. It caches both static and dynamic requests and proxies certain dynamic requests to the application servers (in this case, Apache). The server also has one instance of the Apache web server to handle both static files and dynamic requests. Apache uses mod_php.

Server 2 — Apache web server with mod_php, which also handles both static files and dynamic requests.

Server 3 — MySQL database server.

Server 4 — memcache key/value storage, which stores session and temporary data and rendered blocks.

Summary of Results:

- Number of orders (clients): 2,800
- Time required to process all orders: 496 sec
- Orders/per sec (average): 5.65
- Response time (average): 1.052 sec
- CPU load: 100%
- Concurrent visitors: 100 page views/sec

The system was able to serve approximately 487,300 orders or 79,500,000 catalog views per 24-hour period. The CPU load on web servers during the orders scenario test was as high as it was in the Standard Apache + MySQL architecture (close to 100%). However, due to the aggressive caching of static files as well as catalog pages, the bar was raised to handle 1.8 times additional orders. This architecture can easily serve more than 4,000 catalog page views per second from a single server with Varnish — whereby the only limiting factor is the network interface (1Gbit in these tests). To increase the number of catalog pages served to a client, you can add extra network cards or set up a 10Gbit network. For scaling dynamic content, you must add extra application servers.

Testing Cache Optimized with Nginx

An additional cache-optimized configuration, leveraging the synergy of the Nginx web server, the PHP FastCGI process manager and the Percona MySQL database manager was tested. This configuration was chosen because of the following considerations:

- Nginx provides a unique combination of a web server, a caching proxy with a low memory footprint and a load balancing solution. For serving large amounts of static traffic, Nginx requires fewer system resources than the Apache web server.
- PHP-FPM (PHP FastCGI Process Manager) handles dynamic requests as an application server. It communicates with Nginx via the FastCGI protocol. Compared to Apache, PHP-FPM has lower memory footprint requirements.
- MySQL is enhanced by the Percona support with benefits as outlined above.

For detailed information about the configuration of Nginx and PHP-FPM, refer to Appendix D. Nginx and PHPFPM Configuration.

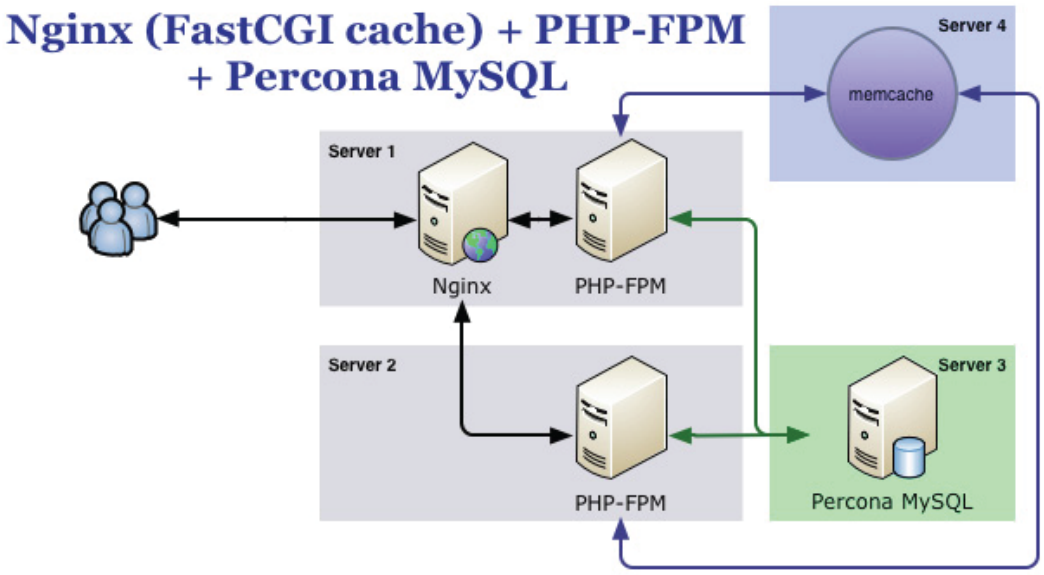

Figure 6: Cache Optimized Infrastructure with Nginx

The Nginx optimized configuration requires the following servers and software components:

Server 1 — This server has Nginx installed. Nginx serves static files directly from the disk and caches dynamic content as much as possible. It proxies certain dynamic requests to the application servers - in this case, PHP-FPM. The server also contains one instance of PHP-FPM which is the FastCGI server for servicing PHP requests.

Server 2 — This server has the second instance of the PHP-FPM application server for servicing PHP requests.

Server 3 — MySQL database server.

Server 4 — This server has the memcache key/value storage, which stores session and temporary data and rendered blocks.

Summary of Results:

- Number of orders (clients): 2,800
- Time required to process all orders: 460 sec
- Orders/per sec (average): 6.09
- Response time (average): 1.237 sec
- CPU load: 100%
- Concurrent visitors: 100 page views/sec

With a configuration optimized by implementing caching inside the FastCGI module of the Nginx server, the system was able to serve approximately 525,300 orders or 373,200,000 catalog views per 24-hour period. During an order scenario test, the CPU load on the web servers was as high as it was on the Standard configuration. However, this configuration managed to offload the system by serving static files to Nginx and aggressively caching catalog pages. This resulted in almost doubling the number of orders created. This architecture can easily serve more than 4,000 catalog page views per second from a single server with Nginx — the only limiting factor is the network connectivity (1Gbit in these tests). To increase the number of catalog pages served to the client, you can add extra network cards or set up a 10Gbit network. For scaling dynamic content, you must add extra application servers.

Summary: Results & Recommendations

Proper configuration of Magento Enterprise Edition with the appropriate hardware capabilities and software configuration can lead to a significant improvement in overall system performance. Using Varnish or Nginx as a caching mechanism greatly offloads the application servers from serving dynamic requests. This results in faster response times and allows more requests to be processed by the application server.

The following points summarize the key findings presented for each configuration and provide additional considerations when optimizing a system:

- Using operation code caching provided by the APC accelerator and memcached for storing sessions and temporary data results in significant improvements in overall system performance.
- Using Nginx with PHP-FPM slightly outperforms Varnish with Apache (by 8%).
- While consuming very little CPU time, both Nginx and Varnish process static files more rapidly compared to Apache. This improvement can be further scaled by adding network cards.
- For the tested scenarios, the database server is not a bottleneck if using one to five application servers. The main bottleneck is the application server CPU bandwidth. With more servers dedicated to handling dynamic requests, more orders can be processed.
- When serving non-cacheable content, the caching layer causes significant delay and decreased response times — by at least 40 times. This results in a net positive outcome.
- By offloading Apache from serving static content, the number of orders handled almost doubles. By using external full-page caching, the number of catalog views can be increased up to 5 times compared to the standard architecture.
- In terms of performance, the caching implementation of Nginx is as efficient as that of Varnish. Thus, it is possible to use Nginx as an alternative to Varnish as a full-page caching solution.
- When deciding between Varnish and Nginx, consider these points: Nginx can be a point of SSL termination; it provides flexible virtual hosting; and preserves cache between restarts. On the other hand, Varnish supports more flexible cache rule definitions due to the Varnish Configuration Language.
- Having a caching layer on two application servers allows the system to survive a single application server failure. If you implement a cache on both servers, use either Active-Passive mode to avoid double caching, or use Active-Active mode behind the load balancer to serve more traffic.