

# Maximizing Performance and Scalability with Magento Enterprise Edition

## Magento White Paper Series

In this Performance White Paper, we provide an overview of architecture and design considerations, optimization guidelines, and performance tuning tips for Magento Enterprise Edition as well as for the underlying hosting environment. We also provide test results from our testing of these optimization techniques in order to provide best practices for maximizing the Magento Enterprise Edition performance.

# Content

## INTRODUCTION 1

## GENERAL CONSIDERATIONS 1

- Environment Components 1
- Magento Enterprise Edition Configuration Components 1

## PERFORMANCE TESTING METHODOLOGY 2

## PERFORMANCE TECHNIQUES AND TEST RESULTS 3

### Database Configuration 3

- Available Memory 3
- Multi-Threading 3
- Built-In Caching 3
- Buffers 3
- Slow Queries Logging 3
- InnoDB Storage 3

### Web Server Configuration 4

### Accelerating PHP 6

- Realpath Cache Configuration 6
- Bytecode Caching 6
- php.ini Configuration 6

### Directory Structure Optimization 7

### Caching in Magento Enterprise Edition 8

- System Cache 8
- Full Page Cache 9

### Handling Sessions 10

### Magento Enterprise Edition cron Scripts 10

### Rebuilding Indexes 10

### Admin Panel Separation 10

## Sales Archive 10

## Checkout Performance Test Results 11

## Frontend Layout Complexity 11

## Number of HTTP Requests per Page 11

## Using Parallel Connections 11

## Media and Static Content Delivery 11

## Additional Performance Gains 12

## Scalability 12

- Scaling Web Nodes 12
- Scaling Database Nodes 13

## Multi-Core Servers 14

## Using Solr as a Search Engine 15

## Summary of Recommendations 16

## APPENDICES 17

### Appendix A: Default Configuration 17

### Appendix B: Optimized Configuration 22

### Appendix C: Software Versions Used 24

### Appendix D: Tests Configuration 25

### Appendix E: Magento Enterprise Edition Sample Databases 26

### Appendix F: Magento Enterprise Edition Sample Configurations 26

## Introduction

Introduced in 2009, the Magento Enterprise Edition is the leading enterprise-grade, feature-rich eCommerce platform built on Open Source technology. The Magento Enterprise Edition provides online merchants with unprecedented flexibility and control over the presentation, content, and functionality of their eCommerce channel, giving merchants the power to create sites that provide an unrivaled and rich online shopping experience for their customers.

Magento Enterprise Edition is an open system designed to be flexible, completely scalable and configurable, easily tailored to merchants' unique technical and business requirements and restraints. However, much like all flexible enterprise-grade software, custom configurations can often result in a number of places where incorrect configuration or insufficient resources can adversely affect performance.

In this Performance Whitepaper, we provide an overview of architecture and design considerations, optimization guidelines, and performance tuning tips for Magento Enterprise Edition as well as for the underlying hosting environment in an effort to provide best practices for maximizing the Magento Enterprise Edition performance.

## General Considerations

When getting ready to deploy and configure Magento Enterprise Edition, there are some general technical environment components that must be taken into consideration when creating an optimized Magento Enterprise Edition setup. These range from physical hardware selection and network throughput to the underlying Open Source stack, which are the key underpinnings in driving Magento Enterprise Edition and its own configuration components.

### *Environment Components*

- **Hardware**  
With a big number of concurrent users, sufficient amount of RAM is highly critical to handle all incoming connections. Faster modern systems with multi-core CPUs, high front side bus speeds, and fast hard drives, preferably at 7200RPM and higher, will generally speed up the entire application.
- **Network**  
Insufficient network I/O throughput and latencies in the internal network can significantly impact the performance of a multi-server setup. Outbound connection latency may cause discomfort for a customer while browsing the store frontend.
- **Software**  
Magento Enterprise Edition is a PHP application that runs on the LAMP stack. Therefore, current, up-to-date, and well-configured versions of the Linux Kernel, Apache, MySQL, and PHP will provide better performance results. A proper configuration of a web server, a database server, and PHP itself is required in order to achieve optimal performance results.

### *Magento Enterprise Edition Configuration Components*

When trying to achieve the optimal setup, take into account the following configuration components:

- Using proper cache backend
- Handling sessions with fast storage
- Optimizing directory structure
- Using Magento Enterprise Edition cron scripts
- Rebuilding indexes
- Using a dedicated server for the Admin backend
- Using sales order archiving
- Simplifying frontend layout
- Reducing the number of HTTP requests on the page
- Using parallel connections
- Delivering media and static content
- Scaling web and database nodes
- Using multi-core servers
- Using Solr module that performs Magento integration with Apache Solr

## Performance Testing Methodology

Tests and metrics are crucial components of measuring and reflecting performance under any given setup. A number of tests have been conducted showing the benefit of optimizing different configurations and setups.

Tests were run on dedicated servers provided by Nexcess.net LLC Hosting — <http://www.nexcess.net/>. Every tested instance was based on the RedHat Enterprise Linux x86\_64 preinstalled by the Nexcess support team with the latest updates included. PHP and MySQL were installed from the Varien RPM repository. The Apache and MySQL configuration files are included in Appendices A through C. Tests were run against three Magento Enterprise Edition installations of different catalog sizes — Magento Enterprise Edition with default sample data of ~100 SKUs, 10,000 SKUs, and 80,000 SKUs catalogs accordingly. The Solr instance used in the corresponding test\ was installed on a dedicated server.

Tests were performed against the following predefined test scenarios:

- CMS Page Test — a default 2 column homepage with static content.
- Shopping Scenario Test - a series of URLs that include categories and product pages, adding product to shopping cart and shopping cart page.
- Browsing Scenario Test — customer activities that are addressed on catalog (category and product) pages browsing
- Order Placing Scenario Test — emulation of adding random product to the shopping cart by user and checking it out using new customer registration. This scenario includes all steps of one page checkout process. It was used to measure ordering performance as described below.



Find the list of URLs in Appendix D of this whitepaper and note that it varies depending on the test and the data used.

The diagrams included in this document were run on three Magento Enterprise Edition installations of different catalog sizes mentioned earlier. These are noted as Magento Enterprise Edition sample data (sd), Magento Enterprise Edition with 10,000 products (10k), and Magento Enterprise Edition with 80,000 products (80k). Most tests were performed with 10, 20, 50, and 100 concurrent connections.



A Full Page Cache test was performed with 50, 100, 200, and 500 concurrent connections respectively.



**Number of concurrent connections** is a number of test threads started consequently with a small delay between them and running simultaneously. Each thread executes the test plan independently of other test threads and simulates an activity of a real Magento Enterprise Edition user.

The majority of figures provided below in this whitepaper (figures [1\](#), [2\](#), [3\](#), [4\](#), [5\](#), [7a\](#), and [8a\](#)) shows both the performance results for CMS Page Test and one of the predefined scenario tests. The performance is calculated as the number of transactions per second.

For the charts in figures [1a\](#), [7b\](#), and [8b](#), the Order Placing Scenario Test results are used, and the performance is calculated as the number of orders per hour.

The rest of the charts provide performance results for one of the predefined scenario tests, and the performance is calculated as the number of transactions per second.

Magento Inc. cooperates with Nexcess.net LLC to provide Magento customers with a fully managed and customizable hosting solution to grow their business on, backed by dedicated account teams providing 24x7 support. Nexcess.net LLC prides itself on its reputation of providing some of the world's best hosting solution coupled with outstanding and unmatched support provided by a top level team.

## Performance Techniques and Test Results

### Database Configuration

Proper MySQL configuration is one of the most important aspects of configuring any Magento Enterprise Edition environment. Optimizing the MySQL configuration can provide up to a 70% performance boost. An incorrect configuration may result in the web server spending more time in idle loops waiting to retrieve data from the database.



The default MySQL installation, even in later versions, is configured to use far less resources than the average hardware can provide.

The calculation formulas that are used in this section can be found in the template file bundled within a MySQL distribution package. Let us go through the most important directives in the MySQL configuration file, **my.cnf**, and their recommended values.

### Available Memory

Magento Enterprise Edition uses InnoDB as its primary table storage engine type. InnoDB, unlike MyISAM, can use the in-memory buffer pool to cache table indexes and data. Less disk I/O is needed to get data from hard drives when the value of the in-memory buffer pool is set higher. A general recommendation is to set this parameter up to 80% of the available RAM for a dedicated database server. In cases where both the web server and the database are running on the same machine, it is recommended to split the entire memory pool into two parts, each having its own primary assigned portion (e.g. on a single server with 6 GB RAM installed, it can be split to have 2-2.5 GB used by MySQL with the rest left for the web server).

The key parameter in this section is **innodb\_buffer\_pool\_size**, which should be set to use as much available memory as possible.

**Table 1. Recommended innodb\_buffer\_pool\_size settings.**

Server Type	innodb_buffer_pool_size
combined web and database server, 6 GB RAM	2-3 GB
dedicated database server, 6 GB RAM	5 GB
dedicated database server, 12 GB RAM	10 GB

### Multi-Threading

Today's industry standard servers typically have more than 1 CPU installed with 2 or more cores each. The InnoDB engine can effectively use multiple threads to serve more concurrent connections.

**innodb\_thread\_concurrency** should be set to a value that equals to or is greater than 8, even for a single

CPU. The recommended value is calculated with the following equation:

$$2 * [\text{numberofCPUs}] + 2$$

**thread\_cache\_size** allows caching of client's threads when a client disconnects and reusing them when new connections are created. The recommended value is within the range from 8 to 64 and it depends on your **max\_connections** number. **thread\_concurrency** can be simply calculated as [number of CPUs] \* multiplier. The multiplier value is between 2 and 4 and is determined by testing different values and benchmarking for the best results in your environment.

### Built-In Caching

**table\_cache** is the number of tables that can be simultaneously opened by MySQL. A value of 1024 will be sufficient for most, if not all, Magento Enterprise Edition sites.

Having the query cache enabled may result in significant speed improvements when you have a large amount of identical queries, which is the case for any eCommerce application frontend. The recommended values for the Magento Enterprise Edition database server are as follows:

- query\_cache\_size 64M
- query\_cache\_limit 2M

### Buffers

A sort buffer is used for optimizing sorting in ORDER BY and GROUP BY queries. 8M is the recommended value for the Magento Enterprise Edition database.

### Slow Queries Logging

Logging slow queries might be useful for debugging purposes, but it should be disabled in production use.

### InnoDB Storage

The InnoDB engine works with a single data storage file which usually grows in time. It is a good idea to have the engine's initial state configured to be at least twice as large as the Magento Enterprise Edition database size, and **innodb\_autoextend\_increment** should be set to a fairly high value in order to avoid frequent data file extending operations.


InnoDB supports transaction operations by using transaction log files. Transaction log files are generally configured in groups of two. The bigger the size of a transaction log file the less often it performs I/O operations on primary storage files. However, in case a database restore is required, more time will be needed. Do not use multiple InnoDB tablespaces unless you are sure you know the benefits of your particular hardware setup.

## Web Server Configuration

The most commonly used Apache configuration provides PHP support with `mod_php`. This Apache configuration loads a large number of modules. However, most of these modules are not necessary for running Magento Enterprise Edition. This becomes more relevant in a multi-server setup where different tasks can be split up into different nodes and each node has to be configured to perform its specific task the best.

The minimum required list of Apache modules includes the following:

- `mod_expires` — generates content expiration and cache control headers
- `mod_deflate` — compresses content before it is delivered to the client
- `mod_mime` — associates the requested file with its type and behavior
- `mod_dir` — serves directory index files
- `mod_rewrite` — supports Search Engine Friendly URL's
- `mod_authz_host` — limits access to specific files
- `mod_authz_user` — might be required in a staging environment to set up password authentication, but on a live site it is not necessary

 If you are running other web applications on the same server, consult which Apache modules are required for them.


With all unused Apache modules disabled by commenting out the corresponding **'LoadModule'** lines in `httpd.conf`, it is possible to cut memory consumed by Apache, which will allow more concurrent connections to be handled with the same amount of RAM.

Another important web server configuration component is setting an optimal number for running Apache processes. The best method to do this is to create the required number of Apache processes when the web server is started. This number should be calculated by measuring the memory amount consumed by Apache under the maximum load. This is currently the best threading method as **mpm\_worker** cannot be safely used with PHP and the process of forking every new Apache child in the **mod\_prefork** mode is an expensive operation.

Also, note that the **ServerLimit** and **MaxClients** values should be specified explicitly to prevent running out of physical memory and going into a swap file which causes a severe breakdown of web server performance. **MaxRequestsPerChild** can be set to the default value of 4000.

When a web server is under a heavy load, keeping persistent connections becomes disadvantageous, thus the **KeepAlive** directive should always be disabled.

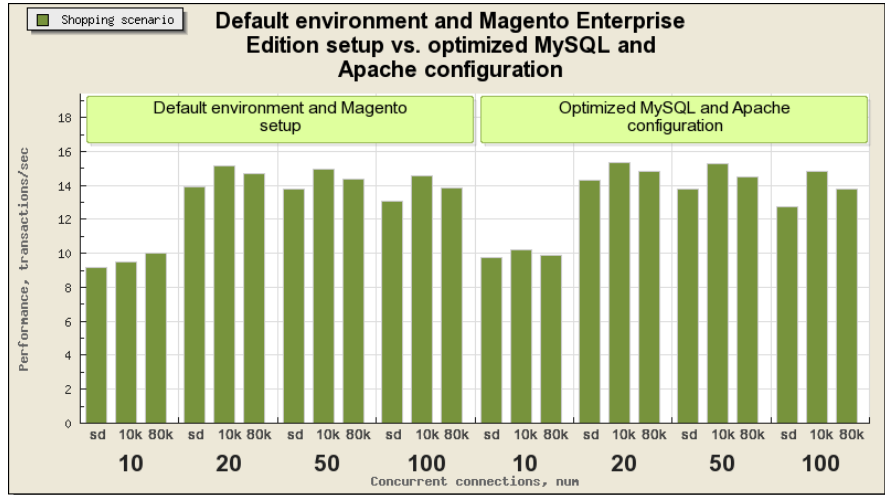
**mod\_deflate** allows compressing the content before sending it to the browser. Magento Enterprise Edition **.htaccess** file already includes necessary settings to enable the compression.

 Make sure to uncomment this section in order to decrease the page load time.

Additionally, you can take advantage of eliminating directory structure scans for the `.htaccess` files by moving all `.htaccess` directives into the appropriate `<Directory>` sections of the main `httpd.conf` file. In order to reduce the I/O throughput on Apache web nodes in a multi-server setup, it is advisable to use a load balancer which can handle all logging activity instead of having the activity handled by the Apache backends.

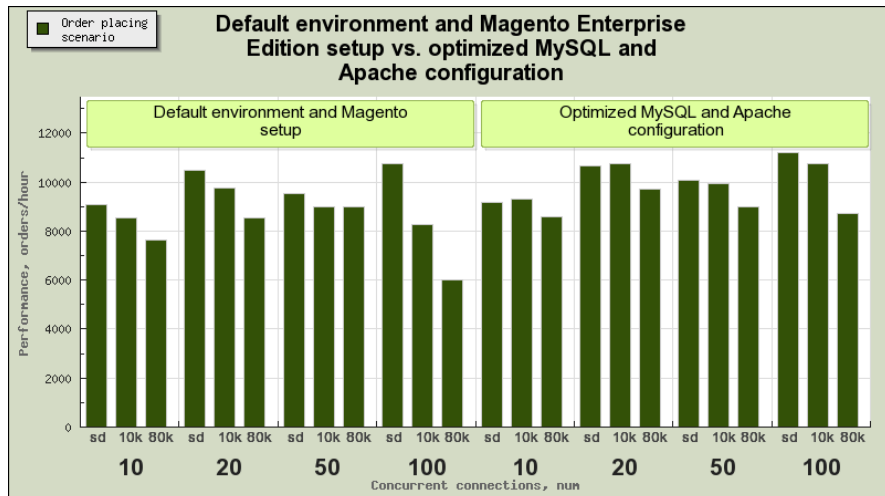
**Figure 1a: Default Environment and Magento Enterprise Edition setup vs. Optimized MySQL and Apache configuration.**

Result: MySQL optimization provides a performance increase.



**Figure 1b: Default Environment and Magento Enterprise Edition setup vs. Optimized MySQL and Apache configuration.**

Result: Similarly to the transaction performance, MySQL optimization provides a checkout performance increase on bigger databases



All the following tests were performed using optimized configurations for both Apache and MySQL

## Accelerating PHP

PHP is an interpreted scripting language. The process of running a PHP script includes the following steps:

- Reading a script file from the hard drive
- Parsing and compiling bytecode
- Running the bytecode

### Realpath Cache Configuration

File I/O optimization is not limited to using faster hard drives only. To provide better optimization, increasing the default `realpath_cache_size` and `realpath_cache_ttl` values in the `php.ini` settings is highly recommended. According to our tests, the recommended values on production servers are as follows:

- `realpath_cache_size=32k`
- `realpath_cache_ttl=7200`

### Bytecode Caching


The process of reading PHP scripts from disk and compiling them can be eliminated by enabling PHP accelerators. PHP accelerators cache compiled bytecode which results in less file and system I/O. The well-known eAccelerator and APC PHP accelerators are tested and partially compatible with Magento Enterprise Edition. Their built-in shared memory can also be used as Magento Enterprise Edition cache storage, which will be covered later in this document.


### php.ini Configuration

In `php.ini`, you can enable only the minimum set of PHP extensions required to run Magento Enterprise Edition to reduce the memory usage and speed up the PHP performance. The necessary extensions are as follows:

- PDO\_MySQL
- simplexml
- mcrypt
- hash
- GD
- DOM
- iconv2
- SOAP (if the Magento Webservices API is to be used)

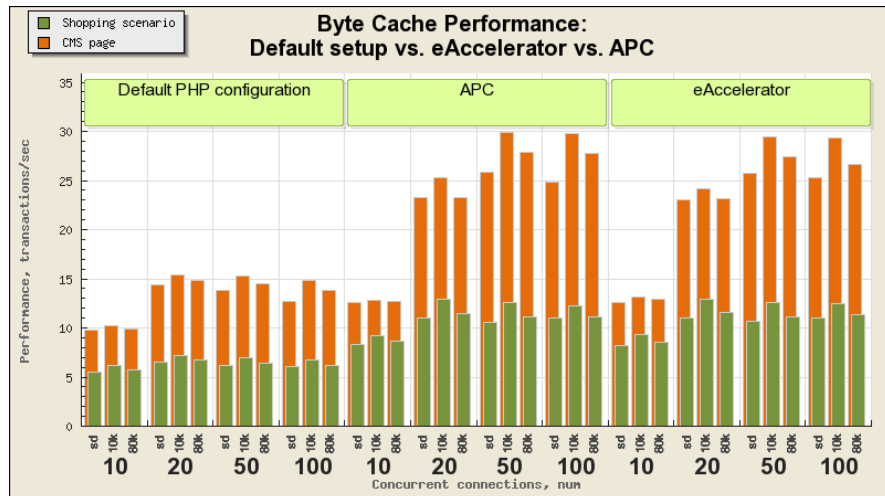
In case of using APC bytecode cache, the modification time check option (`apc.stat`) must be disabled.

 If you are running other PHP applications on the same server, consult which PHP extensions are required for them.

 All the following tests were performed using the APC bytecode cache enabled.


**Figure 2: Default setup vs. eAccelerator vs. APC**

Result: Adding a PHP accelerator provides a performance boost to 100% when different PHP files are used. According to our tests, APC is 3-5% better than eAccelerator in the bytecode caching mode





## Directory Structure Optimization

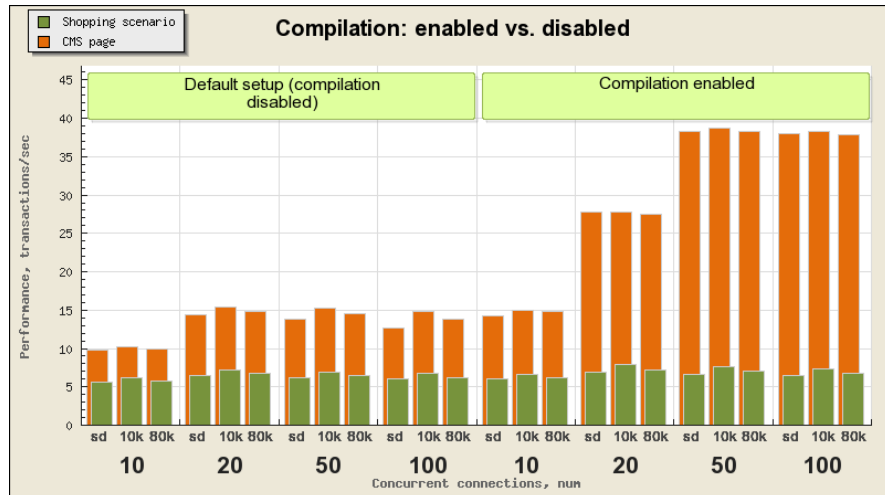
 The following test was performed with APC bytecode cache feature disabled, there is no sense to use both APC bytecode cache and Directory Structure Optimization feature at the same time.

Optimizing directory structure can help fine-tune Magento Enterprise Edition performance in case there is no way to use APC bytecode cache. It is highly recommended to use the Zend Framework distribution bundled within Magento Enterprise Edition as it is adjusted to significantly reduce the number of system calls required to locate a file in the directory structure.

This is accomplished by commenting out all extra **require\_once** directives within the configuration file. Additional **require\_once** calls are not required because Magento Enterprise Edition implements its own **autoload** function which handles all necessary file requests on demand. Recent Magento Enterprise Edition versions (since version 1.3.x) include the Magento Enterprise Edition Compilation Module (**Mage\_Compiler**) which provides extra optimization by placing all files in one directory and combining the most used classes in a few single files.

**Figure 3: Compilation enabled vs. Compilation disabled**

*Result: Enabling Magento Enterprise Edition Compilation Module provides a good additional performance boost*



## Caching in Magento Enterprise Edition

### System Cache

Magento Enterprise Edition can cache frequently used data using various cache backends. When installing Magento Enterprise Edition, the file system is set to be used as a cache backend by default. Using a cache backend will always improve the performance of Magento Enterprise Edition. Still, while the file system cache is the most reliable storage with unlimited size, it does not provide the best performance. Magento Enterprise Edition v. 1.11 can also work with the following cache backends that provide performance better than that of the file system cache backend:

- *APC* — a bytecode cache for PHP which also provides a shared memory storage for application data
- *Memcached* — a distributed high-performance caching system.

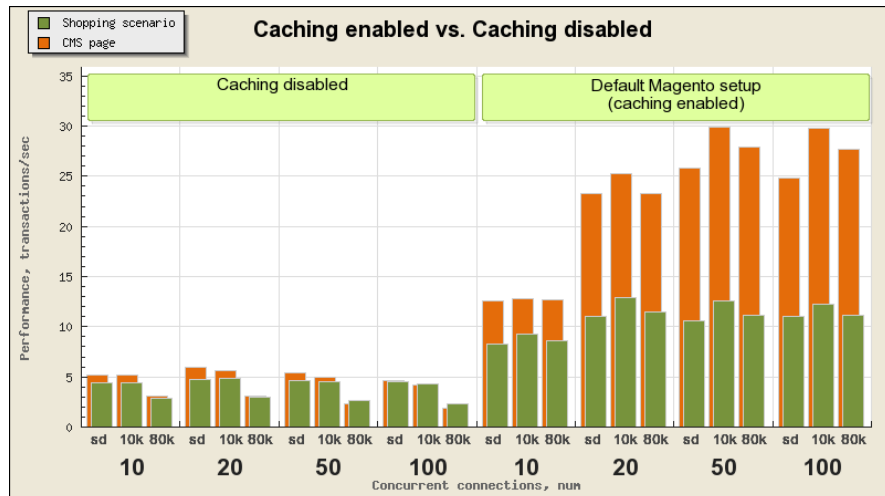
Once the cache is enabled and one of the above-mentioned shared memory backends is used, Magento automatically uses [TwoLevel cache backend](#). In this case, the file system cache backend is used as a slow cache storage for one web node environment, and the MySQL database backend for the environment with multiple web nodes.

This is fully customizable and can be easily set in `app/etc/local.xml`, as shown in Appendix F\.

- ⚠ When using APC, the Source Code Compiler must be disabled.
- ⚠ Make sure that if you are using APC or memcached as backends, you configure each of them with enough memory to include all cached data; otherwise, a backend may purge the required cache hierarchy structure.

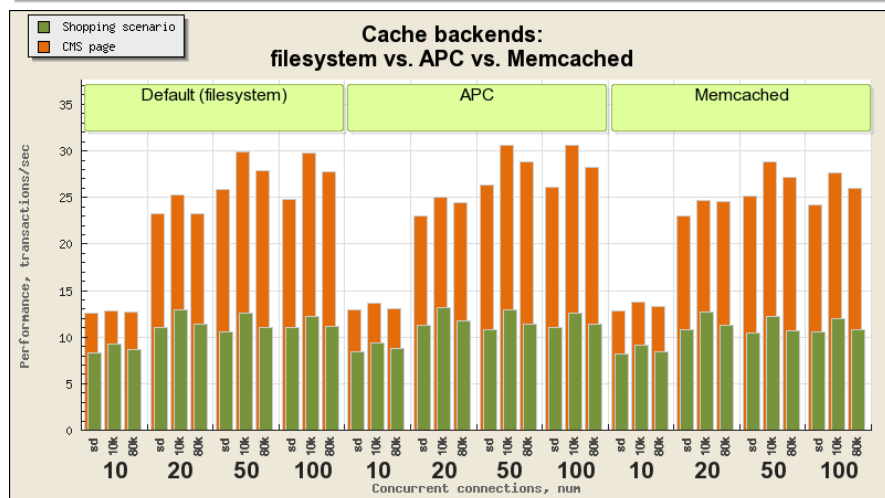
**Figure 4: Caching enabled vs. Caching disabled.**

*Result: It may be required to disable the built-in Magento Enterprise Edition cache (that is, by default, enabled after the installation) during the active development. Make sure that caching is enabled on production sites as disabled cache makes the store frontend 5-6 times slower and less responsive under a load.*



**Figure 5: Cache backends: file system vs. APC vs. Memcached.**

*Conclusion: APC gives the best results, which is 3-5% better than the file system backend results. However, using Memcached is recommended for installations which involve multiple web nodes, because, in case of multiple web nodes, the APC and file system cache backends need additional administration for synchronizing data between nodes*

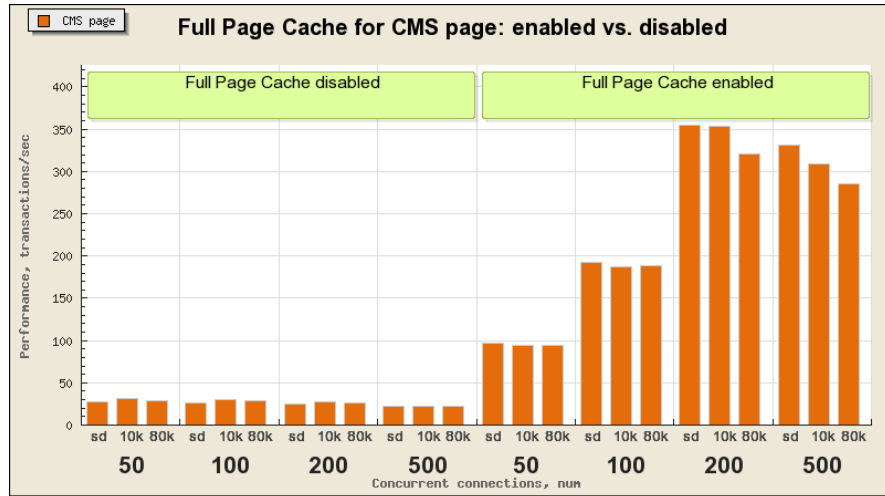


## Full Page Cache

Magento Enterprise Edition can cache entire page contents and return statically stored (X)HTML files rather than build pages dynamically. Only CMS, category, and product view pages support full page caching. Homepage is a CMS page which is accessed most frequently. Full Page Cache allows the web server to significantly increase the performance on pages that can be cached statically.

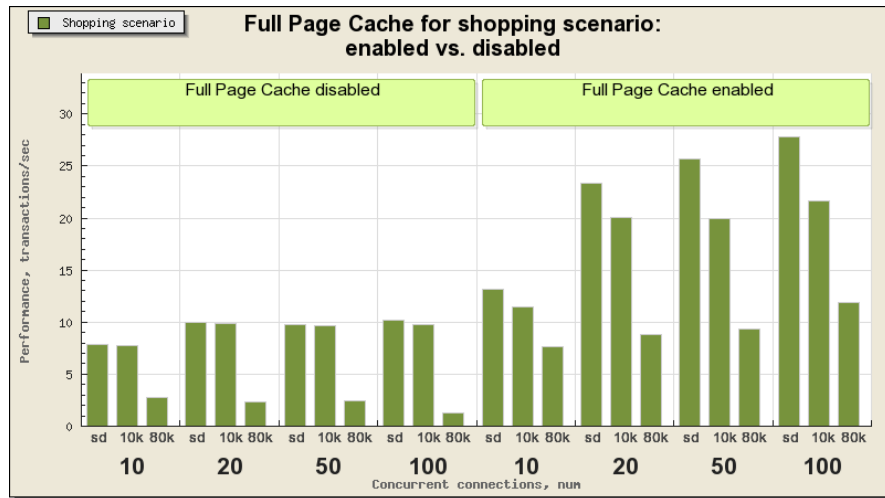
**Figure 6a: Full Page Cache Results For CMS Page Test: enabled vs. disabled.**

Result: Our tests show that with Full Page Cache enabled, the performance increase for the homepage exponentially depends on the number of simultaneous concurrent connections. Increasing the number of concurrent connections from 10 to 100 leads to the homepage performance increase.



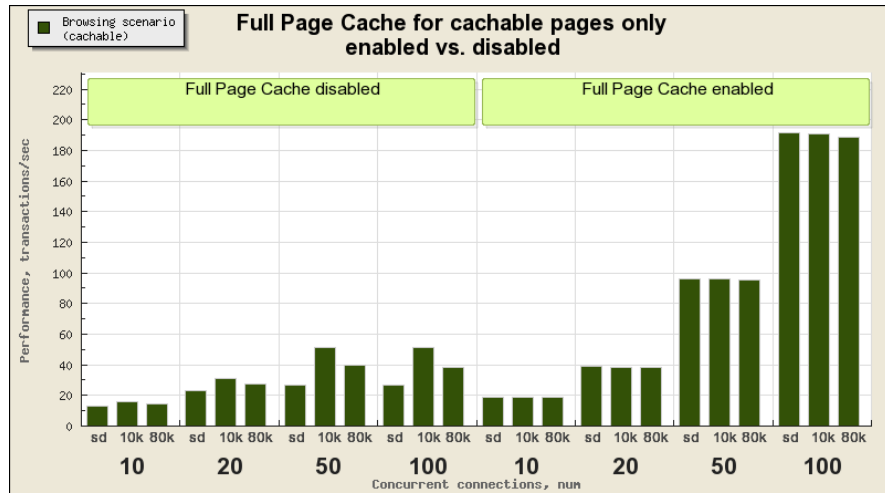
**Figure 6b: Full Page Cache for shopping scenario: enabled vs. disabled**

Result: The performance behavior for the shopping scenario slightly differs from tests for the homepage and cachable URLs, i.e. increasing the number of concurrent connections from 10 to 100 leads to the performance increase up to 150% increase and does not have exponential dependency.



**Figure 6c: Full Page Cache for browsing scenario: enabled vs. disabled**

Result: The performance behavior for the list of cachable URLs resembles tests for the homepage, i.e. increasing the number of concurrent connections from 10 to 100 leads to the performance increase up to 9.5 times; while for the standard set of URLs, the performance gives 150% increase.



Conclusion: Full Page Cache stores and uses generated full web page for forthcoming users. This significantly improves performance of website up to 9.5 times by decreasing load on hardware during regular surfing through the website.

## Handling Sessions

Magento Enterprise Edition uses PHP sessions to store customer session data.

The default method is to use the file system storage which works well if you are using a single web server. Its performance can be improved by configuring the **tmpfs** in-memory partition to avoid extra hard drive I/O activity.

In a clustered environment with multiple web servers, the first option for handling sessions is to use a load balancer capable of associating client requests with specific web nodes based on the client IP or the client cookies. If you are in a clustered environment and not using a load balancer capable of the above-mentioned, it is necessary to share the session data among all web servers. Magento Enterprise Edition supports two additional session storage types that can be used in this case.

Though fully supported, storing session data in the database is not recommended as it puts an additional load on the main database, and therefore, requires a separate database server to efficiently handle multiple connections under load in most cases. However, storing session data in the database gives advantage in case it's important to keep user sessions in spite of any server crashes. It is guaranteed that the database-driven sessions will not be damaged when one or all servers in the cluster are down.

The **memcached** session storage is free of these disadvantages.

The **memcached** service can be run on one of the cluster servers to provide fast session storage for all web nodes of the cluster. Though, because of extra overhead processing compared to raw file system session files, the **memcached** session storage does not show any performance improvements when used in a single server configuration.

## Magento Enterprise Edition cron Scripts

There are some activities in Magento Enterprise Edition that must be scheduled with cron. For example, catalog price rules generate price indexes for three days ahead when a rule is applied from the administrator panel. In order not to make catalog price rules expire in three days, they must be refreshed by the Magento Enterprise Edition cron scripts on a daily basis. There are also other scheduled tasks in Magento Enterprise Edition, and it is important not to configure all of them to run simultaneously as it may overload the master database.



### Note on Scheduled Backups:

It is a general best practice to run important data backup on servers on a daily basis. Though it is not under the

Magento Enterprise Edition scope, make sure that the system backups on your server are run at a time when the number of active customers and the server load are minimal.

## Rebuilding Indexes

The process of rebuilding the Magento Enterprise Edition index tables, for example, layered navigation indexes or catalog price rules indexes, is a resource consuming operation.



Do not rebuild indexes during the period of highest customer activity.

Magento Enterprise Edition includes the content staging functionality that can be used for scheduling necessary catalog updates at an appropriate time.

In the future Magento Enterprise Edition versions, there will be an option to use a separate database for calculating index data in order not to affect the performance of the store.

## Admin Panel Separation

Admin panel operations are in general more resource consuming than the frontend activities. Often they require increasing PHP memory limits or having extra extensions compiled into PHP. Therefore, having a dedicated admin server can help make admin panel operation faster while not impacting the frontend configuration and performance. The separation can be done by specifying different base URL's on a global level, and for the frontend websites and stores. Each separate domain name can then be served by a separate server.

## Sales Archive

As it has been previously noted, generally the backend operations take more server resources, especially when the administrator works with orders. A huge number of orders to be processed by the Magento Enterprise Edition installation may be the cause of server performance failure. However, most of the orders are just listed in the backend; they are not actively processed by the administrator.

Sales archiving introduced in Magento Enterprise Edition v. 1.8 helps the administrator list and work only with those orders that are currently in use. The rest orders are available in the archive; thus, the administrator is not distracted by unimportant data. This lets the administrator perform order management operations faster and reduces the overall server load for administration routines.

Sales archive operation is provided by separating sales entity grid database tables. While the table for archived

orders contains a full set of data, the table for active orders contains only the records corresponding to active orders. This decreases the amount of data in the grid tables and improves the performance of sorting/filtering operations in the admin grids.

## Checkout Performance Test Results

In the previous releases, two big database transactions, one for the customer increment calculation (if a new customer is created during the checkout) and the other for the stock items quantity decrease, were involved in the checkout process. It caused big pauses when customers were simultaneously performing the checkout; only one checkout at the same time was possible. Checkout process has been significantly reworked since Magento Enterprise Edition v. 1.8; so, the stock operations transaction has been separated to smaller quicker database requests, and the customer increment logic has become optional (it is disabled by default now). Due to this change, locks on the database tables used for the checkout are avoided, and simultaneous checkouts for many customers are possible.

Magento Enterprise Edition v. 1.8 has also introduced a flat structure for the Sales module data which results in the additional checkout performance increase.

## Frontend Layout Complexity

The Magento Enterprise Edition page generation process implies building a block object hierarchy specified by the frontend theme layout. The fewer is the number of blocks on a page the less time is required to instantiate block objects.

Default Magento Enterprise Edition themes were designed to represent all Magento Enterprise Edition features. The themes introduce a large number of small blocks on most pages. A specific store will benefit from a custom built theme which facilitates practical customer experience while taking into account business and product specifics.

Customizing a proper design theme, simplifying the layout structure, and combining small blocks can make the page generation time 4-5% faster than using of a complex layout with a lot of blocks on the page.

## Number of HTTP Requests per Page

In order to improve page load and processing time, it is important to reduce the number of HTTP requests per page. Magento Enterprise Edition allows combining multiple JavaScript files and style sheets into a smaller number of files.



This process is fully under the control of a theme developer who implements it through the flexible system of theme layouts instead of directly including the JavaScript files from within the templates.

The following is the example of how the number of JavaScript files may be properly reduced:

```
<action
method="addJs"><script>custom_js/gallery.js</script
></action>
<action
method="addJs"><script>custom_js/intro.js</script><
/action>
</reference>
```

The previous layout file example combines the two scripts in a single file that will be added to the page with one request to `js/index.php?c=auto&f=,custom_js/gallery.js,custom_js/intro.js`.

## Using Parallel Connections

Browsers can load page elements in parallel. Specifying different domains for media, skin, and JavaScript URLs in the Magento Enterprise Edition configuration (**System > Configuration > GENERAL > Web**) will help speed the page rendering time in the browser, as most browsers limit the number of downloads to 2-4 parallel threads per domain name.

Other web page design recommendations are beyond the scope of this document. You can find the detailed list of web site design best practices at the Yahoo Developer Network at <http://developer.yahoo.com/performance/rules.html>.

## Media and Static Content Delivery

Though Apache is a fast and reliable web server, there are other web server options that are known to more efficiently serve static content and media files which consumes less memory and CPU time.

Among the widely used are nginx, lighttpd, and tinyhttpd. These are the multiplexing web servers which do not have built-in scripting languages support but can handle thousands of simultaneous connections per server.

```
<reference name="head">
```

© 2011, Magento, Inc. All rights reserved.

December 1, 2011

## Additional Performance Gains

Static content delivery can be improved by using a caching reverse proxy, such as Squid or an HTTP accelerator like Varnish. A reverse proxy can locally cache the content received from Apache in order to reduce the load on the Apache backends.

Another way to reduce your server load and to get smaller network latencies is the use of content delivery networks (CDN). Most CDNs support pushing media content through a simple API and can be integrated with the Magento Enterprise Edition backend quite easily.

## Scalability

Magento Enterprise Edition is designed to utilize the benefits of running a multi-server setup in a clustered environment. Web nodes are not limited to be of

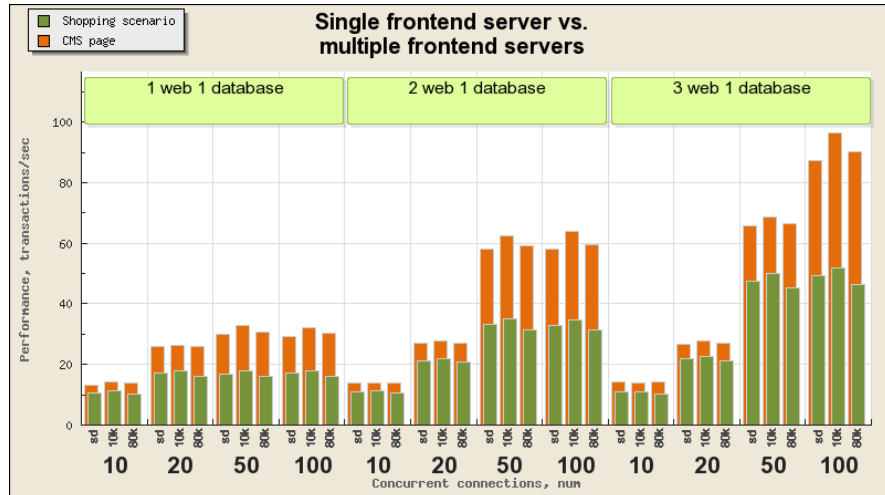
exactly the same type. There may be different nodes, such as frontend servers, static content and media servers, and a separate admin panel server each performing different tasks.

### Scaling Web Nodes

Magento Enterprise Edition can be scaled over any number of additional web servers. This allows handling a bigger number of concurrent requests by simply introducing new web nodes when the number of page views and visitors grows. Doubling the number of web nodes can provide a performance increase of over 90%.

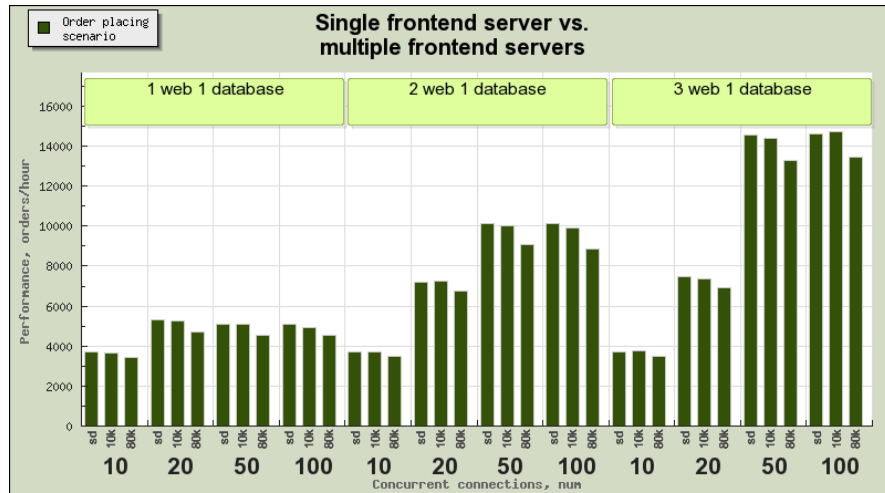
**Figure 7a: Single frontend server vs. multiple frontend servers**

*Conclusion: Enabling a second frontend web server almost doubles the number of requests that can be handled by this setup. Introducing every new web node adds up to a 100% performance increase with higher concurrencies.*



**Figure 7b: Single frontend server vs. multiple frontend servers**

*Result: Enabling a second frontend web server almost doubles the number of requests that can be handled by this setup. Introducing every new web node adds up to a 100% performance increase with higher concurrencies.*



### *Scaling Database Nodes*

Magento Enterprise Edition works with a database in a manner that easily allows separating database connections for read and write activities. Each particular module can use its own connections if needed. This is fully customizable and can be easily set in **app/etc/local.xml**, as shown in Appendix F.

In case of separating read and write database operations, there must be a master database instance that processes only write requests, and a slave database instance that is used for reading data. There must be an instant replication organized between the master and the slave instances.

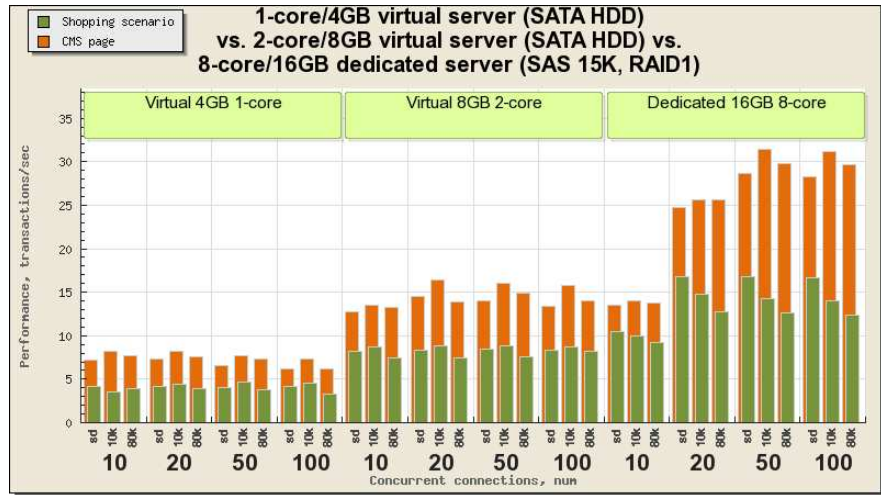
The replication speed is a critical parameter and must be as high as possible

## Multi-Core Servers

As it is expected, the test results show better performance on servers with more CPU cores per server as can be seen from the following figures.

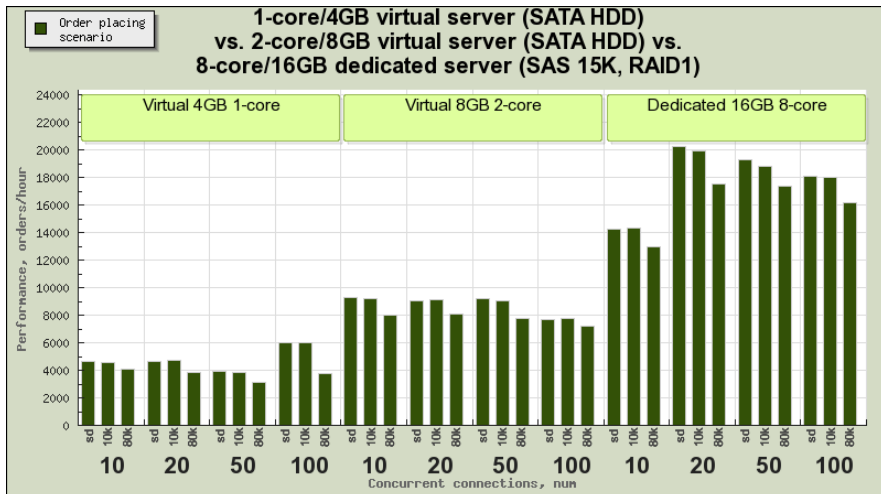
**Figure 8a: 1-core/4GB virtual server (SATA HDD) vs. 2-core/8GB virtual server (SATA HDD) vs. 8-core/16GB dedicated server (SAS 15K, RAID1)**

*Result: A low-level single-core server with a single SATA hard drive and 4GB RAM installed is able to handle about 8 trans/sec on homepage tests and up to 4 trans/sec during an average customer session. Adding more cores and faster hard drives configured in a RAID array allows for the handling of more page requests. Dual-core server with SATA hard drive is able to handle 15 trans/sec on homepage and up to 8 trans/sec on customer session URL test, while 8-core server with SAS 15K hard drives in RAID1 array can handle 29 trans/sec on homepage and about 15 trans/sec on customer session URL test with hundreds of concurrent customer sessions which allows for the serving of more page views for more visitors. In our tests, we were able to get the same requests per second with 400 concurrent sessions, but to keep page response time low, one might consider adding another web node for serving such traffic.*



**Figure 8b: 1-core/4GB virtual server (SATA HDD) vs. 2-core/8GB virtual server (SATA HDD) vs. 8-core/16GB dedicated server (SAS 15K, RAID1)**

*Result: A low-level single-core server with a single SATA hard drive and 4GB RAM installed is able to handle about 4000 orders/hour during an average customer session. Adding more cores and faster hard drives configured in a RAID array allows for the handling of more orders per hour. Dual-core server with SATA hard drive is able to handle 9000 orders/hour on customer session URL test, and, finally, 8-core server with SAS 15K hard drives in RAID1 array can handle up to 19000 orders/hour on customer session URL test.*





## Using Solr as a Search Engine

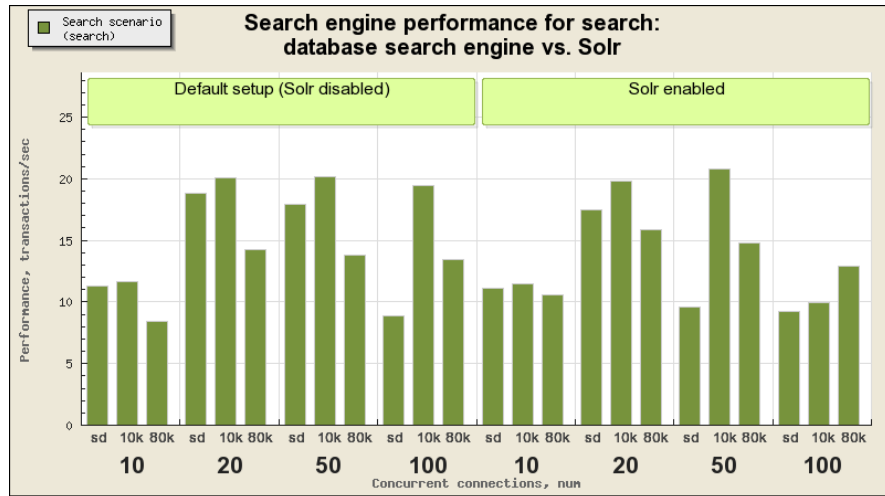
Since v. 1.8, Magento Enterprise Edition supports the Apache Solr search engine and provides the **Enterprise Search** module that allows choosing between the built-in MySQL fulltext search engine and the Solr. Any of these two search engines can be selected in the backend configuration under **System > Configuration > CATALOG > Catalog > Catalog Search**:

- MySQL Fulltext
- Solr

The fulltext search is used by default in case Solr is not available, not effective, or cannot be used in the current Magento configuration. However, Apache Solr provides better performance on a bigger product list, thus it is recommended to use Solr when possible, unless you are not running an electronic store with a small product list.

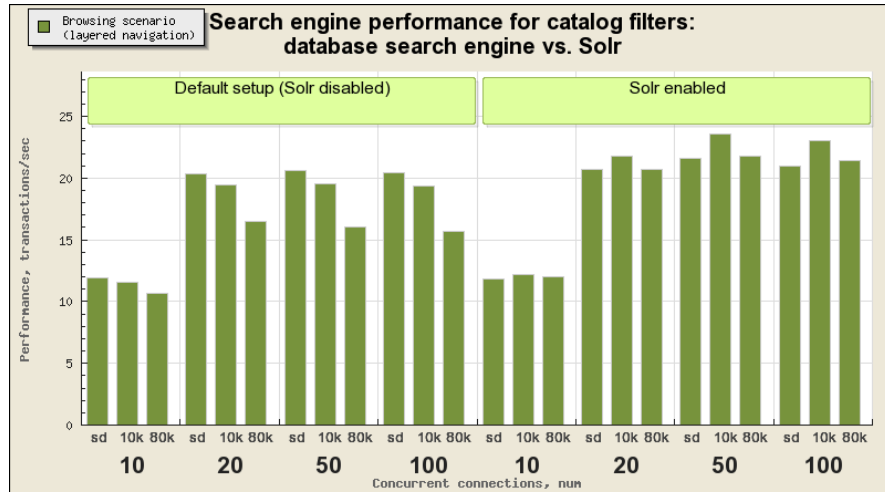
**Figure 9a: Search engine performance for search: database search engine vs. Solr**

*Result: Enabling the Solr integration gives the search performance increase up to 30% on bigger database and higher concurrent connections number. As a side-effect, using Solr significantly increases the search quality.*



**Figure 9b: Search engine performance for catalog filter: database search engine vs. Solr**

*Result: Enabling the Solr integration gives the catalog navigation (filtering) performance increase up to 30% on bigger database and higher concurrent connections number.*



## Summary of Recommendations

Beyond doubt, when planning, deploying, and configuring your Magento Enterprise Edition setup, much attention to detail must be paid in order to get the most significant benefits of it. The considerations on physical platform, network performance, stack configuration and fine-tuning as well as a suitable Magento Enterprise Edition installation play an important role in ensuring that you get the best performance possible out of your unique setup. The following is the review of some key points highlighted in this whitepaper.

- It is strongly recommended to optimize the MySQL and Apache configuration which provides a 40-45% performance increase, especially on dynamic pages. The default MySQL and Apache setup is configured to use far less resources than the average hardware can provide. The default setup is not able to handle a high number of concurrent customer sessions which may result in unpredictable and sometimes erratic server load.
- Adding a PHP accelerator is another important aspect of configuring your Magento Enterprise Edition environment. eAccelerator shows good results with up to 100% increase. According to our tests, APC is even more efficient showing a performance boost with additional 3-5% better than eAccelerator.
- Enabling caching on production sites is vital. The disabled cache can make the web store frontend 5-6 times slower and less responsive under load.
- When configuring Magento Enterprise Edition on a single host, consider using default file system cache backend or APC fast storage cache backend which provide similar results. However, in case of using several web nodes, it is better to switch to the memcached cache backend, as in case of multiple web nodes, APC and file system cache backends need additional administration to synchronize data between nodes.
- Enabling the Full Page Cache feature can improve the Magento Enterprise Edition installation for any page with static content (homepage or other CMS pages), so if you considered having the homepage static and using several landing pages for your website, it is highly recommended to turn Full Page Cache on.
- In a single server setup, there is no need to change the default file system session storage as it shows the best results. However, in clustered environment, if the used load balancer cannot associate client requests with specific web nodes based on the client IP or the client cookies, it may be required to use either the memcached or the database shared session storage. The memcached session storage shows the results which are close to those of the default storage or even 1-2% worse. The database session storage should be used in a clustered environment only if the memcached storage cannot be used for some reasons.
- Installing the Magento Enterprise Edition Compilation Module and enabling compilation can give additional performance boost.
- In case the Apache Solr search engine is available, it is recommended to have the Solr search engine support enabled in the catalog search configuration options. The search quality increases significantly even in case the product list is short and it gives only a small performance increase (not more than 30% on catalog navigation and 30% on search).
- Using modern multi-core CPUs and fast hard drives improves the results even more. A low-level single-core server with a single SATA hard drive and 4GB RAM installed is able to handle about 8 trans/sec on homepage tests and up to 4 trans/sec during an average customer session. Dual-core server with SATA hard drive is able to handle 15 trans/sec on homepage and up to 8 trans/sec on customer session URL test, while 8-core server with 15K hard drives in RAID1 array can handle 29 trans/sec on homepage and about 15 trans/sec on customer session URL test with hundreds of concurrent customer sessions which allows for the serving of more page views for more visitors.

The Magento Enterprise Edition subscription is the leading Open Source eCommerce platform built on solid technology which gives you the flexibility, configurability, and performance you need to develop a unique online marketing and storefront channel. Attention to detail can go a long way and with a little bit of fine-tuning, the Magento Enterprise Edition rock solid stability and well-configured performance give you the ultimate value when it comes to serving as many customers as richly and as cost-effectively as possible, while providing you with the edge necessary to differentiate yourself from and stay ahead of your competitors.

## Appendices

### Appendix A: Default Configuration

#### Apache Web Server Configuration

```

ServerTokens OS
ServerRoot "/etc/httpd"
PidFile run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15

StartServers      8
MinSpareServers  5
MaxSpareServers  20
ServerLimit      256
MaxClients       256
MaxRequestsPerChild 4000

Listen 80

LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_alias_module modules/mod_authn_alias.so
LoadModule authn_anon_module modules/mod_authn_anon.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
LoadModule authn_default_module modules/mod_authn_default.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule authz_owner_module modules/mod_authz_owner.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule authz_dbm_module modules/mod_authz_dbm.so
LoadModule authz_default_module modules/mod_authz_default.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule logio_module modules/mod_logio.so
LoadModule env_module modules/mod_env.so
LoadModule ext_filter_module modules/mod_ext_filter.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule expires_module modules/mod_expires.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule headers_module modules/mod_headers.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule info_module modules/mod_info.so
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule actions_module modules/mod_actions.so
LoadModule spelling_module modules/mod_spelling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule cache_module modules/mod_cache.so
LoadModule suexec_module modules/mod_suexec.so
LoadModule disk_cache_module modules/mod_disk_cache.so
LoadModule file_cache_module modules/mod_file_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so
LoadModule cgi_module modules/mod_cgi.so
LoadModule version_module modules/mod_version.so

User apache
Group apache

```

```

ServerAdmin root@localhost
UseCanonicalName Off

DocumentRoot "/var/www/html"
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

DirectoryIndex index.html index.html.var
AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

TypesConfig /etc/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
#   MIMEMagicFile /usr/share/magic.mime
    MIMEMagicFile conf/magic
</IfModule>

HostnameLookups Off
ErrorLog logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access_log combined

ServerSignature On

Alias /icons/ "/var/www/icons/"
<Directory "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

IndexOptions FancyIndexing VersionSort NameWidth=* HTMLTable

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

```

```

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

DefaultIcon /icons/unknown.gif

ReadmeName README.html
HeaderName HEADER.html

IndexIgnore .??.* *~ *# HEADER* README* RCS CVS *,v *,t

AddLanguage ca .ca
AddLanguage cs .cz .cs
AddLanguage da .dk
AddLanguage de .de
AddLanguage el .el
AddLanguage en .en
AddLanguage eo .eo
AddLanguage es .es
AddLanguage et .et
AddLanguage fr .fr
AddLanguage he .he
AddLanguage hr .hr
AddLanguage it .it
AddLanguage ja .ja
AddLanguage ko .ko
AddLanguage ltz .ltz
AddLanguage nl .nl
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddLanguage pt .pt
AddLanguage pt-BR .pt-br
AddLanguage ru .ru
AddLanguage sv .sv
AddLanguage zh-CN .zh-cn
AddLanguage zh-TW .zh-tw

LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl nn no pl pt pt-BR ru sv zh-CN zh-TW

ForceLanguagePriority Prefer Fallback

AddDefaultCharset UTF-8

AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

AddHandler type-map var

AddType text/html .shtml
AddOutputFilter INCLUDES .shtml

Alias /error/ "/var/www/error/"

<IfModule mod_negotiation.c>
<IfModule mod_include.c>
  <Directory "/var/www/error">
    AllowOverride None
    Options IncludesNoExec
    AddOutputFilter Includes html
    AddHandler type-map var
    Order allow,deny
    Allow from all
    LanguagePriority en es de fr
    ForceLanguagePriority Prefer Fallback
  </Directory>
</IfModule>
</IfModule>

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

BrowserMatch "Microsoft Data Access Internet Publishing Provider" redirect-carefully
BrowserMatch "MS FrontPage" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
BrowserMatch "^gnome-vfs/1.0" redirect-carefully
BrowserMatch "^XML Spy" redirect-carefully

```

```
BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully
```

## MySQL Configuration

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
old_passwords=1
max_connections=100

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

## PHP Configuration

```
[PHP]
engine = On

zend.zend_compatibility_mode = Off
short_open_tag = On
asp_tags = Off

precision = 14
y2k_compliance = On
output_buffering = 4096
zlib.output_compression = Off
implicit_flush = Off

unserialize_callback_func=

serialize_precision = 100
allow_call_time_pass_reference = Off

safe_mode = Off
safe_mode_gid = Off
safe_mode_include_dir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH

disable_functions =
disable_classes =

expose_php = On

max_execution_time = 30 ; Maximum execution time of each script, in seconds
max_input_time = 60; Maximum amount of time each script may spend parsing request data
memory_limit = 128M ; Maximum amount of memory a script may consume (128MB)

error_reporting = E_ALL
display_errors = Off
display_startup_errors = Off
log_errors = On
log_errors_max_len = 1024

ignore_repeated_errors = Off
ignore_repeated_source = Off

report_memleaks = On
track_errors = Off

variables_order = "EGPCS"
register_globals = Off
register_long_arrays = Off
register_argc_argv = Off
auto_globals_jit = On

post_max_size = 8M

magic_quotes_gpc = Off
magic_quotes_runtime = Off
magic_quotes_sybase = Off

auto_prepend_file =
auto_append_file =

default_mimetype = "text/html"

doc_root =
user_dir =
```

```

enable_dl = On
file_uploads = On

upload_max_filesize = 2M

allow_url_fopen = On
allow_url_include = Off

default_socket_timeout = 60

[Syslog]
define_syslog_variables = Off

[mail function]
; For Win32 only.
SMTP = localhost
smtp_port = 25

; For Win32 only.
;sendmail_from = me@example.com

; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
sendmail_path = /usr/sbin/sendmail -t -i

[SQL]
sql.safe_mode = Off

[ODBC]
odbc.allow_persistent = On
odbc.check_persistent = On
odbc.max_persistent = -1
odbc.max_links = -1
odbc.defaultlrl = 4096
odbc.defaultbinmode = 1

[MySQL]
mysql.allow_persistent = On
mysql.max_persistent = -1
mysql.max_links = -1
mysql.default_port =
mysql.default_socket =
mysql.default_host =
mysql.default_user =
mysql.default_password =
mysql.connect_timeout = 60
mysql.trace_mode = Off

[MySQLi]
mysqli.max_links = -1
mysqli.default_port = 3306
mysqli.default_socket =
mysqli.default_host =
mysqli.default_user =
mysqli.default_pw =
mysqli.reconnect = Off

[mSQL]
msql.allow_persistent = On
msql.max_persistent = -1
msql.max_links = -1

[PostgreSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0

[Sybase]
sybase.allow_persistent = On
sybase.max_persistent = -1
sybase.max_links = -1
sybase.min_error_severity = 10
sybase.min_message_severity = 10
sybase.compatability_mode = Off

[Sybase-CT]
sybct.allow_persistent = On
sybct.max_persistent = -1
sybct.max_links = -1
sybct.min_server_severity = 10
sybct.min_client_severity = 10

```

```

[bcmath]
bcmath.scale = 0

[Informix]
ifx.default_host =
ifx.default_user =
ifx.default_password =
ifx.allow_persistent = On
ifx.max_persistent = -1
ifx.max_links = -1
ifx.textasvarchar = 0
ifx.byteasvarchar = 0
ifx.charasvarchar = 0
ifx.blobinfile = 0
ifx.nullformat = 0

[Session]
session.save_handler = files
session.save_path = "/var/lib/php/session"
session.use_cookies = 1
session.name = PHPSESSID
session.auto_start = 0
session.cookie_lifetime = 0
session.cookie_path = /
session.cookie_domain =
session.cookie_httponly =
session.serialize_handler = php
session.gc_probability = 1
session.gc_divisor = 1000
session.gc_maxlifetime = 1440
session.bug_compat_42 = 0
session.bug_compat_warn = 1
session.referer_check =
session.entropy_length = 0
session.entropy_file =
session.cache_limiter = nocache
session.cache_expire = 180
session.use_trans_sid = 0
session.hash_function = 0
session.hash_bits_per_character = 5

url_rewriter.tags = "a:href,area:href,frame:src,input:src,form:fakeentry"

[MSSQL]
mssql.allow_persistent = On
mssql.max_persistent = -1
mssql.max_links = -1
mssql.min_error_severity = 10
mssql.min_message_severity = 10
mssql.compatibility_mode = Off
mssql.secure_connection = Off

[Tidy]
tidy.clean_output = Off

[soap]
soap.wsdl_cache_enabled=1
soap.wsdl_cache_dir="/tmp"
soap.wsdl_cache_ttl=86400

```

## Appendix B: Optimized Configuration

### *Apache Web Server Configuration*

```

ServerTokens OS
ServerRoot "/etc/httpd"
PidFile run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15

StartServers 100
MinSpareServers 100
MaxSpareServers 150
ServerLimit 256
MaxClients 256
MaxRequestsPerChild 4000

Listen *:80

```



```

LoadModule authz_host_module modules/mod_authz_host.so
LoadModule expires_module modules/mod_expires.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule mime_module modules/mod_mime.so
LoadModule dir_module modules/mod_dir.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule log_config_module modules/mod_log_config.so

User apache
Group apache

ServerAdmin root@localhost
#ServerName www.example.com:80
UseCanonicalName Off

DocumentRoot "/var/www/html"
<Directory />
Options FollowSymLinks
AllowOverride None
</Directory>

<Directory "/var/www/html">
Options Indexes FollowSymLinks
AllowOverride All
Order allow,deny
Allow from all
</Directory>

AccessFileName .htaccess

<Files ~ "^\.ht">
Order allow,deny
Deny from all
</Files>

TypesConfig /etc/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
#MIMEMagicFile /usr/share/magic.mime
MIMEMagicFile conf/magic
</IfModule>

HostnameLookups Off
ErrorLog logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common

LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

CustomLog logs/access_log combined

ServerSignature On
AddDefaultCharset UTF-8
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

LoadModule php5_module modules/libphp5.so
AddHandler php5-script .php
AddType text/html .php
DirectoryIndex index.php

```

## MySQL Configuration

```

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
old_passwords=1

# Disabling symbolic-links is recommended to prevent assorted security risks;
# to do so, uncomment this line:
# symbolic-links=0

max_connections = 1000
max_connect_errors = 10
table_cache = 1024
max_allowed_packet = 16M
max_heap_table_size = 64M

```

```

sort_buffer_size = 8M
join_buffer_size = 8M
thread_cache_size = 8
thread_concurrency = 8
query_cache_size = 64M
query_cache_limit = 2M
tmp_table_size = 64M
key_buffer_size = 32M
read_buffer_size = 2M
read_rnd_buffer_size = 16M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover
innodb_additional_mem_pool_size = 16M
innodb_log_buffer_size = 8M
innodb_log_file_size = 512M
innodb_log_files_in_group = 2
innodb_buffer_pool_size = 10G
innodb_data_file_path = ibdata1:1G;ibdata2:512M:autoextend
innodb_autoextend_increment=512

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

```

### *PHP Configuration (complementary to the default configuration)*

#### Required extensions

```

extension=bcmath.so
extension=curl.so
extension=dom.so
extension=gd.so
extension=mcrypt.so
extension=memcache.so
extension=mhash.so
extension=pdo.so
extension=pdo_mysql.so

```

#### Extensions from the default setup that are no longer needed in the optimized configuration

```

;;;extension=dbase.so
;;;extension=json.so
;;;extension=mysqli.so
;;;extension=mysql.so
;;;extension=pdo_sqlite.so
;;;extension=sqlite.so
;;;extension=wddx.so
;;;extension=xmlreader.so
;;;extension=xmlwriter.so
;;;extension=xsl.so
;;;extension=zip.so

```

#### APC-specific configuration used where APC is enabled

```

extension=apc.so
apc.shm_size=256
apc.num_files_hint=10000
apc.user_entries_hint=10000
apc.max_file_size=5M

```

#### eAccelerator-specific configuration used where eAccelerator is enabled

```

zend_extension="/usr/lib64/php/modules/eaccelerator.so"
eaccelerator.shm_size = "256"

```

## Appendix C: Software Versions Used

- CentOS release 5.5 (Final)
- Linux 2.6.18-194.3.1.el5 SMP x86\_64 GNU/Linux
- mysql 5.0.84
- PHP 5.2.13
- php-pecl-apc-3.1.3p1-1
- php-eaccelerator-0.9.5.3
- Apache/2.2.3
- memcached 1.4.5
- SIEGE 2.69

- Magento Enterprise Edition 1.11.0.0

## Appendix D: Tests Configuration

### *Shopping scenario URL List for Testing with Sample Data*

```
$(BASEURL)
$(BASEURL) electronics/cell-phones.html
$(BASEURL) electronics/cell-phones.html?price=2%2C100
$(BASEURL) electronics/cell-phones.html?price=2%2C100&color=23
$(BASEURL) electronics/cell-phones/samsung-mm-a900m-ace.html
$(BASEURL) checkout/cart/add/product/20/
$(BASEURL) apparel.html
$(BASEURL) cn-clogs-beach-garden-clog.html
$(BASEURL) checkout/cart/
$(BASEURL) checkout/onepage/
$(BASEURL) catalogsearch/result/?q=ink&x=0&y=0
$(BASEURL) apparel.html?cat=17
$(BASEURL) apparel.html?price=1%2C100&cat=17
$(BASEURL) the-get-up-kids-band-camp-pullover-hoodie.html
$(BASEURL) checkout/cart/add/product/39/
$(BASEURL) apparel/shirts.html
```

### *Shopping scenario URL List for Testing with 10.000 and 80.000 SKUs*

```
$(BASEURL)
$(BASEURL) category-3.html
$(BASEURL) category-3.html?cat=28
$(BASEURL) category-3.html?cat=28&price=2%2C1000
$(BASEURL) category-3/pr15031-50.html
$(BASEURL) checkout/cart/add/product/2489/
$(BASEURL) category-273.html
$(BASEURL) category-273/100-190-b7h.html
$(BASEURL) checkout/cart/
$(BASEURL) checkout/onepage/
$(BASEURL) catalogsearch/result/?q=345&x=0&y=0
$(BASEURL) category-273.html?cat=427
$(BASEURL) category-273.html?cat=427&ab_host=303
$(BASEURL) category-273/r1256ap.html
$(BASEURL) checkout/cart/add/product/1354/
$(BASEURL) category-273/category-309.html
```

### *Browsing scenario URL List of Cachable Pages Only for Testing with Sample Data*

```
$(BASEURL)
$(BASEURL) electronics/cell-phones.html
$(BASEURL) electronics/cell-phones.html?price=2%2C100
$(BASEURL) electronics/cell-phones.html?price=2%2C100&color=23
$(BASEURL) electronics/cell-phones/samsung-mm-a900m-ace.html
$(BASEURL) apparel.html
$(BASEURL) cn-clogs-beach-garden-clog.html
$(BASEURL) apparel.html?cat=17
$(BASEURL) apparel.html?price=1%2C100&cat=17
$(BASEURL) the-get-up-kids-band-camp-pullover-hoodie.html
$(BASEURL) apparel/shirts.html
$(BASEURL) apparel/shoes.html
$(BASEURL) apparel/shoes.html?price=1%2C1000000
$(BASEURL) apparel/shoes.html?price=1%2C1000000&shoe_type=52
$(BASEURL) kenneth-cole-new-york-men-s-con-verge-slip-on.html
```

### *Browsing scenario URL List of Cachable Pages Only for Testing with 10.000 and 80.000 SKUs*

```
$(BASEURL)
$(BASEURL) category-3.html
$(BASEURL) category-3.html?cat=28
$(BASEURL) category-3.html?cat=28&price=2%2C1000
$(BASEURL) category-3/pr15031-50.html
$(BASEURL) category-273.html
$(BASEURL) category-273/100-190-b7h.html
$(BASEURL) category-273.html?cat=427
$(BASEURL) category-273.html?cat=427&ab_host=303
$(BASEURL) category-273/r1256ap.html
$(BASEURL) category-273/category-309.html
$(BASEURL) category-924/category-925.html
$(BASEURL) category-924/category-925.html?price=1%2C1000000
$(BASEURL) category-924/category-925.html?aahauptkategorie=513&price=1%2C1000000
$(BASEURL) category-924/category-925/cc08-01-002.html
```

## Search scenario URL list for Testing Solr Integration for Search

Automatically generated URL lists for testing Solr vs. MySQL search are attached as separate files:

- [for search with Sample Data](#)
- [for search with 10.000 and 80.000 SKUs](#)

Please note, that the Solr vs. MySQL search performance was tested using slightly different technique, every URL from the list was taken randomly to prevent every concurrent connection to perform the same search request.

## Search scenario URL list Used for Testing Solr Integration for catalog filtering with Sample Data

```
$(BASEURL) apparel/shirts.html
$(BASEURL) apparel/shirts.html?price=1%2C100
$(BASEURL) apparel/shirts.html?price=2%2C100
$(BASEURL) apparel/shirts.html?color=White
$(BASEURL) apparel/shoes/mens.html
$(BASEURL) apparel/shoes/mens.html?price=2%2C100
$(BASEURL) apparel/shoes/mens.html?color=Gray
$(BASEURL) apparel/shoes/mens.html?shoe_type=Running
$(BASEURL) apparel/shoes/womens.html
$(BASEURL) apparel/shoes/womens.html?price=1%2C100
$(BASEURL) apparel/shoes/womens.html?manufacturer=CN+CLogs
$(BASEURL) apparel/shoes/womens.html?color=Brown
```

## Search scenario URL list for Testing Solr Integration for catalog filtering with 10.000 and 80.000 SKUs

```
$(BASEURL) category-736.html
$(BASEURL) category-736.html?aahauptkategorie=515
$(BASEURL) category-736.html?aahauptkategorie=514
$(BASEURL) category-736.html?aahauptkategorie=513
$(BASEURL) category-924.html
$(BASEURL) category-924.html?aahauptkategorie=515
$(BASEURL) category-924.html?aahauptkategorie=514
$(BASEURL) category-924.html?aahauptkategorie=513
$(BASEURL) category-811.html
$(BASEURL) category-811.html?aahauptkategorie=515
$(BASEURL) category-811.html?aahauptkategorie=514
$(BASEURL) category-811.html?aahauptkategorie=513
```



```
$(BASEURL) = http://server.test/
```

## Appendix E: Magento Enterprise Edition Sample Databases

Please contact [enterprise@magento.com](mailto:enterprise@magento.com) for sample data and databases.

## Appendix F: Magento Enterprise Edition Sample Configurations

### Configuration for Two Level Cache

```
<config>
  <global>
    <cache>
      <cache>
        <!-- Can also be either apc, memcached, or file. If empty, file is assumed -->
        <backend>memcached</backend>
        <!-- Used for only memory based storages, can be either database or file. If empty, file is assumed -->
      </cache>
    </global>
  </config>

  <slow_backend></slow_backend>
  <!-- memcached cache backend related config -->
  <memcached>
    <servers><!-- any number of server nodes can be included -->
      <server1>
        <host><![CDATA[localhost]]></host>
        <port><![CDATA[11211]]></port>
        <persistent><![CDATA[1]]></persistent>
        <weight><![CDATA[2]]></weight>
        <timeout><![CDATA[10]]></timeout>
        <retry_interval><![CDATA[10]]></retry_interval>
        <status><![CDATA[]]></status>
      </server1>
    </servers>
  </memcached>
</global>
```

```

<!--
        <server2>
            ...
        </server2>
-->
</servers>
<compression><![CDATA[0]]></compression>
<cache_dir><![CDATA[]]></cache_dir>
<hashed_directory_level><![CDATA[]]></hashed_directory_level>
<hashed_directory_umask><![CDATA[]]></hashed_directory_umask>
<file_name_prefix><![CDATA[]]></file_name_prefix>
</memcached>
</cache>
</global>
</config>

```

### Configuration for Two Database Nodes

The following configuration code sample shows how to set up two separate connections for the master and slave database servers:

```

<config>
  <global>
    <resources>
      <default_setup>
        <connection>
          <host><![CDATA[master]]></host>
          <username><![CDATA[writeuser]]></username>
          <password><![CDATA[writeuserpwd]]></password>
          <dbname><![CDATA[magento]]></dbname>
          <active>1</active>
        </connection>
      </default_setup>
      <default_read>
        <connection>
          <use></use>
          <host><![CDATA[slave]]></host>
          <username><![CDATA[readuser]]></username>
          <password><![CDATA[readuserpwd]]></password>
          <dbname><![CDATA[magento]]></dbname>
          <model>mysql4</model>
          <initStatements>SET NAMES utf8</initStatements>
          <type>pdo_mysql</type>
          <active>1</active>
        </connection>
      </default_read>
    </resources>
  </global>
</config>

```





Magento, Inc. is an experienced player in the eCommerce industry. Since 2001, we have been leveraging the power of Open Source technology to help online merchants fulfill their business goals at a fraction of the cost and time of proprietary eCommerce solutions. Our clients are a new generation of merchants eager to realize their creative ideas and gain an edge in their industry, without technology getting in the way.

Magento is a feature-rich, professional open-source eCommerce solution that offers merchants complete flexibility and control over the look, content, and functionality of their online store. Magento's intuitive administration interface contains powerful marketing, merchandising and content management tools to give merchants the power to create sites that are tailored to their unique business needs. Completely scalable and backed by an extensive support network, Magento offers companies the ultimate eCommerce solution.

For more information, visit us at [www.MagentoCommerce.com](http://www.MagentoCommerce.com).

## Headquarters

Magento, Inc.  
10441 Jefferson Blvd.  
Suite 200  
Culver City, CA 90232  
USA  
[www.MagentoCommerce.com](http://www.MagentoCommerce.com)

## Sales Information and Inquiries

Magento has offices and solution partners worldwide. Contact us for more information about Magento Enterprise Edition at [enterprise@magento.com](mailto:enterprise@magento.com).

