

SharePoint Server 2010 Capacity Management for Web Content Management Deployments

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft Corporation. All rights reserved.

SharePoint Server 2010 Capacity Management for Web Content Management Deployments

Authors: Joshua Stickler, Tyler Butler

Testers: Zhi Liu, Cheuk Dong, Philippe Flamm

Microsoft Corporation

April 2010

Summary: This white paper provides guidance on capacity management for a SharePoint Server 2010 Web Content Management (WCM) solution

This white paper addresses the following scenarios:

* Internet publishing site
* Intranet publishing site
* Enterprise wiki

Contents

[Prerequisite information 3](#_Toc258507840)

[Introduction 4](#_Toc258507841)

[Web Content Management Deployments 4](#_Toc258507842)

[What to optimize 5](#_Toc258507843)

[Throughput is the key metric 5](#_Toc258507844)

[Bottlenecks and remediation 5](#_Toc258507845)

[Front-end Web Server CPU Utilization 6](#_Toc258507846)

[Other bottlenecks 7](#_Toc258507847)

[Caching helps 8](#_Toc258507848)

[Test Results and Recommendations 8](#_Toc258507849)

[Effect of enabling the output cache 8](#_Toc258507850)

[Output Cache Hit Ratio 8](#_Toc258507851)

[Conclusions and Recommendations 10](#_Toc258507852)

[Anonymous versus Authenticated 10](#_Toc258507853)

[Cache Profiles 10](#_Toc258507854)

[Conclusions and Recommendations 11](#_Toc258507855)

[Scale-out characteristics of read/write operations 11](#_Toc258507856)

[Conclusions and Recommendations 13](#_Toc258507857)

[Output cache caveats 14](#_Toc258507858)

[Data freshness 14](#_Toc258507859)

[Memory constraints 16](#_Toc258507860)

[Conclusions and Recommendations 16](#_Toc258507861)

[Effect of read volume on CPU, response time 17](#_Toc258507862)

[Conclusions and Recommendations 17](#_Toc258507863)

[Impact of write operations on throughput 17](#_Toc258507864)

[Conclusions and Recommendations 22](#_Toc258507865)

[Impact of content deployment 22](#_Toc258507866)

[Effect of Content Deployment Import on Front-end Web Server CPU Utilization 24](#_Toc258507867)

[Conclusions and Recommendations 25](#_Toc258507868)

[Impact of database snapshots during content deployment export 25](#_Toc258507869)

[Conclusions and Recommendations 26](#_Toc258507870)

[Content characteristics 26](#_Toc258507871)

[Number of pages 26](#_Toc258507872)

[Multivalued lookup fields 27](#_Toc258507873)

[Effect of usage reporting 28](#_Toc258507874)

[Appendix 29](#_Toc258507875)

[Test details and approach 29](#_Toc258507876)

[Hardware 29](#_Toc258507877)

[Web Servers and Application Servers 29](#_Toc258507878)

[Database Server 30](#_Toc258507879)

[Dataset 30](#_Toc258507880)

## Prerequisite information

Before reading this document, it is important that you understand the key concepts behind Microsoft® SharePoint® Server 2010 capacity management. The following documentation will help you learn about the recommended approach to capacity management and provide context for helping you understand how to make effective use of the information in this document.

For more conceptual information about performance and capacity that that you might find valuable in understanding the context of the data in this white paper, see the following documents:

* [Capacity Planning and Sizing for SharePoint Server 2010](http://technet.microsoft.com/en-us/library/cc261700(office.14).aspx)
* [Technical Case Study – Enterprise Intranet Publishing Environment](http://technet.microsoft.com/en-us/library/cc261716(office.14).aspx)

# Introduction

This white paper addresses the following scenarios:

* **Internet publishing site**
  + Example: Corporate presence site
* **Intranet publishing site**
  + Example: Internal news site
* **Enterprise wikis**
  + Example: Knowledge repository

By reading this document, you will understand the following concepts:

* Throughput is the key metric to maximize to support high volumes of read operations
* Various potential bottlenecks relevant to a Web Content Management SharePoint Server 2010 deployment
* The importance of the output cache in maximizing throughput
* The effect of write operations on the end-user read experience

# Web content management deployments

This white paper provides guidance on capacity management relevant to SharePoint sites that have the Publishing Infrastructure enabled. There are two models by which content is authored in SharePoint publishing sites that can affect your choice of server farm topology:

In the **author-in-place** model, a single site collection is shared by authors and visitors. Authors can create and update content at any time, leading to variable distributions of read and write operations throughout a given day. This server farm typically experiences a high number of reads and a moderate number of writes.



Internet/Intranet Visitors

Authors

In the **content deployment** model, multiple site collections separately and exclusively support content authoring and publishing. Content is created and updated in the authoring environment and then deployed to the publishing environment on a scheduled basis to be read by visitors. The publishing environment primarily serves read requests except when content is being deployed from the authoring environment. Unlike the author-in-place model, the server load exerted by content deployment can be adjusted to scheduled intervals.



Internet/Intranet Visitors

Authors

Content

Deployment

These content authoring models are mutually exclusive. While Internet publishing sites and Intranet publishing sites can use either author-in-place or content deployment, enterprise wikis work best with the author-in-place model.

An **enterprise wiki** is a single-farm site that grows organically as contributors create new pages and link them to other pages that might or might not exist yet. This site allows people across a company or organization to capture and share knowledge using a solution that is integrated into and enhanced by their SharePoint environment. An enterprise wiki typically experiences a higher volume of write operations relative to read operations because a larger proportion of users can edit pages. Enterprise wiki pages are different from publishing article pages and exhibit different performance characteristics.

# What to optimize

## Throughput is the key metric

Throughput and response time are the most important metrics to optimize when you conduct capacity planning for a SharePoint Server 2010 WCM deployment. Throughput is the number of operations that a server farm can perform per second, measured in requests per second (RPS).

## Bottlenecks and remediation



Figure

A bottleneck is the system resource that, when fully utilized, prevents the server farm from serving additional requests.

### Front-end Web Server CPU utilization

The front-end Web server CPU should be the bottleneck of a well-tuned topology because it is the most readily scalable component. The load balancer routes requests among front-end Web servers and ensures that no single server is significantly more utilized than its peers.

Although additional users can visit the site after front-end Web server CPU utilization has maxed out, the server response time that these users experience increases. This behavior is useful for managing spikes in request volume; however, sustained load beyond a server farm’s capacity eventually results in a large enough backlog of requests to exceed the waiting requests threshold. At this point, front-end Web servers throttle requests and respond with HTTP error 503. In Figure 2, server response time decreases after the waiting requests threshold because only HTTP errors are served.

Figure

1. Front-end Web server CPU utilization approaches 100%
2. The request waiting threshold has been exceeded and additional requests are served with errors

### Other bottlenecks

If the front-end Web server CPU is not the bottleneck, either the farm topology, the farm configuration, or the content of the pages being served should be investigated. Some potential bottlenecks include:

1. Network: in situations of high throughput, the network might be saturated either between the front-end Web server and the database server or between the front-end Web server and end users. To avoid this situation, it is recommended that front-end Web servers use dual gigabit NICs.
2. Database server CPU: If the database server CPU becomes the bottleneck, adding front-end Web servers to your server farm does not increase the maximum throughput your farm can support. This can reflect two situations:
   1. Poor cache settings or excessively slow pages, particularly those that are not output cached. This is characterized by high database server CPU utilization but low or medium throughput and low or medium front-end Web server utilization.
   2. The database server might have reached full capacity for the throughput required for the farm. This is characterized by high front-end Web server and database server CPU utilization at high throughput.

## Caching helps

SharePoint Server 2010 uses three types of caching. The common goal of these caches is to improve efficiency by reducing calls to the database for data that is frequently requested. Subsequent requests for a page can be served from the cache on the front-end Web server, resulting in significantly reduced resource utilization on the front-end Web servers and database servers.

* Output cache
  + Stores requested page content in the memory of the front-end Web server
  + See also: [Output Caching and Cache Profiles](http://msdn.microsoft.com/en-us/library/aa661294(office.14).aspx)
* Object cache
  + Stores SharePoint objects, such as Web and list item metadata, in the memory of the front-end Web server
  + See also: [Object Caching](http://msdn.microsoft.com/en-us/library/aa622758(v=office.14).aspx)
* Disk-based cache for Binary Large Objects (BLOBs)
  + Stores image, sound, and video files on disk
  + See also: [Disk-Based Caching for Binary Large Objects](http://msdn.microsoft.com/en-us/library/aa604896(v=office.14).aspx)

Each of the caches is important for sustaining high throughput; however, output caching has the largest effect and is discussed in depth in this white paper.

# Test results and recommendations

## Effect of enabling the output cache

The output cache is a valuable feature to optimizing a SharePoint Server 2010 solution for high volumes of read operations.

To determine maximum RPS, the number of active users making requests on the farm was increased until CPU utilization of either the database server or front-end Web servers reached 100 percent and became a bottleneck. The test was conducted on 1x1, 2x1, 4x1, and 8x1 farm topologies to demonstrate the effect of scaling out front-end Web servers at different output cache hit ratios.

### Output cache hit ratio

The output cache hit ratio is a measure of output cache hits to misses.

* A cache **hit** occurs when the cache receives a request for object data already stored in the cache.
* A cache **miss** occurs when request for object data that is not stored in the cache. When a cache miss occurs, the cache will attempt to store the requested object data so that subsequent requests for that data result in a cache hit if that request can be cached.

There are several reasons why a page request might result in a cache miss.

* Pages configured not to use the output cache
* Personalized pages, for example pages that have data specific for the current user
* First time browse per output cache variation key
* First time browse after cached content expired

Figure 3

The data point for maximum RPS on a 4x1 server farm with a 100 percent output cache hit ratio is extrapolated and was not observed. The server farm request volume reached the network bandwidth limit; that is, the data transfer rate approached 1 gigabit per second. In all cases, the front-end Web server CPU utilization is 100 percent.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Cache Hit Ratio** | **Measure** | **1x1** | **2x1** | **4x1** |
|
| **100%** | Maximum RPS | 3,463 | 7,331 | 11,032 |
| SQL Server CPU utilization | 0% | 0% | 0% |
| **95%** | Maximum RPS | 2,137 | 3,945 | 5,937 |
| SQL Server CPU utilization | 5.93% | 12.00% | 21.80% |
| **90%** | Maximum RPS | 1,518 | 2,864 | 4,518 |
| SQL Server CPU utilization | 7.12% | 14.40% | 28.00% |
| **50%** | Maximum RPS | 459 | 913 | 1,596 |
| SQL Server CPU utilization | 9.86% | 19.50% | 42.00% |
| **0%** | Maximum RPS | 172 | 339 | 638 |
| SQL Server CPU utilization | 9.53% | 19.00% | 36.30% |

Table 1

### Conclusions and recommendations

Higher output cache hit ratios yield significant increases in maximum RPS. Therefore, we recommend enabling output caching to optimize your SharePoint Server 2010 publishing solution. The output cache can be configured in Site Collection Administration->Output Cache Settings.

In tests that had output caching enabled, the first request that caches a page was excluded; that is, a certain percentage of pages are already stored in the cache. When a user first requests an uncached page, the page is added to the cache. If the cache has reached or is approaching capacity, the cache trims the data that was least recently requested.

The 0% cache hit ratio simulates both an environment in which output caching is not enabled and an environment in which caching is enabled, but the short duration after the cache has been flushed while the cache fills. For example, this would be observed every day in a real world environment when the application pool recycles, so it is important to scale your hardware appropriately to accommodate a situation in which there is a 0% cache hit ratio for the brief period of time between the application pool recycle and subsequently requesting and thereby caching pages.

## Anonymous vs. authenticated

The previous test assumes that all requests to the site are made by anonymous readers. However, in your site, some or all users might be authenticated. Examples of authenticated read scenarios include a corporate intranet publishing site and members-only content on an Internet site.

You can specify different output cache behavior for authenticated users versus anonymous users with output cache profiles.

### Cache profiles

**Cache profiles** aggregate settings that you can apply to pages, page items, content types, and levels of scale in your site deployment. A cache profile defines the following types of cache behavior:

* The length of time that items should be held within the cache
* The security trimming policy
* The expiration of settings, such as duration and changes
* The variations of cached contents, for example based on user permission, user rights, and other custom variables

Any change to a cache profile immediately affects all applicable content on the site. You can set different cache profiles for anonymous and authenticated users.

For anonymous users, the Public Internet (Purely Anonymous) output cache profile was used and for authenticated users, the Extranet (Published Site) output cache profile was used.

The authentication model was Windows Basic Authentication. While we do not recommend using Windows Basic Authentication for Internet sites, this authentication method was chosen to demonstrate a minimum overhead imposed by authentication. The size of this overhead varies by your specific authentication mechanism. When testing your deployment, be sure to take the effect of your authentication mechanism into account. For more information about the authentication mechanisms supported by SharePoint Server 2010, refer to [Plan authentication methods (SharePoint Server 2010)](http://technet.microsoft.com/en-us/library/cc262350%28office.14%29.aspx).

### Conclusions and recommendations

Authenticated users experience lower RPS and less scale-out potential because the additional work of validating credentials exerts load on the database server. As evident from the test results, the maximum RPS observed when users are authenticated is significantly lower than that of an anonymous access farm.

## Scale-out characteristics of read/write operations

Our tests were constructed to record Writes Per Hour (WPH). In this white paper, a write is defined as either the creation and check-in of a new Publishing Page or the editing and check-in of an existing Publishing Page.

For the following tests, readers were added to the system until front-end Web server CPU utilization reached approximately 80-90 percent, then write operations were performed in the environment using artificial delay. The total WPH for the test was approximately 500. We used a 90 percent output cache hit ratio for all tests. We performed the same test on a 1x1, 2x1, and 4x1 farm and observed the front-end Web server and Microsoft SQL Server® CPU usage as well as the overall read throughput for each configuration. In addition, we tested an anonymous read-only configuration as a baseline, and also tested a configuration with authenticated readers using Windows Basic Authentication.

While the front-end Web server CPU was maxed out at 100 percent usage during the read-only scale out tests, we held the front-end Web server CPU between 80-90 percent for the scale out tests with writes. This was done to leave room for additional CPU utilization when performing write activity.

Figure 4 depicts the overall read RPS observed during each test. The read RPS scales linearly as additional WFEs are added, even with write activity. However, there is an RPS loss when incorporating writes.

Figure 4

Database server CPU usage increased as the number of front-end Web servers increased. Figure 7 shows the growth pattern of SQL Server CPU usage in the various configurations. As observed in Anonymous versus Authenticated tests above, authentication affects database server CPU utilization, and this becomes more pronounced as write activity is added (which likewise affects database server CPU utilization).

The extrapolated trend in SQL Server usage demonstrates that SQL Server will become the bottleneck with 6 front-end Web servers with authenticated read requests. In the anonymous read case, however, scaling out to a larger number of front-end Web servers is feasible.

It is important to note that additional factors in a typical deployment affect load on the database server that are important to take into account when conducting capacity planning. To learn more about determining a green zone for typical database server CPU utilization, see Capacity Management Overview.

Figure 5

### Conclusions and recommendations

Our data shows that scaling out the number of front-end Web servers is an effective strategy for increasing throughput as long as the database server does not become the bottleneck. On average, the anonymous read/authenticated writes test mix exerted a 52% increase in front-end Web server CPU utilization as compared to an anonymous read/no writes test mix. In addition, authenticated reads add a large additional SQL Server load, because each request incurs additional authentication checks, which requires a round trip to SQL Server.

Figure 6

## Output cache caveats

If the only concern in capacity planning were to maximize RPS, these tests would suggest that the optimal cache hit ratio is 100 percent. However, it might not be feasible or desirable to enable output caching of any or all pages due to data freshness requirements or memory constraints.

### Data freshness

Data that is served from the output cache might not contain recent updates that have been made to the original content. In the source of content deployment or (for authenticated authors) in an author-in-production scenario, authors might want to see the most recent changes immediately after they update existing content.

This is generally mitigated by setting the **Duration** property in the cache profile, which specifies the amount of time that a cached page persists in the output cache before it expires. When a page expires, it is removed from the cache and a subsequent request results in a cache miss that refreshes the page content.

The **Check for Changes** cache profile property can also be set so that the server compares the time at which a page was cached with the time at which content was last modified in a site collection. A request for a page with unmatched time stamps causes cache invalidation for all pages in the site collection. Because “Check for Changes” affects all pages in a site collection, we recommend enabling this option only in the case of an authenticated author-in-place solution that is infrequently updated and essentially static. Combining this option with a long duration allows all pages to be served from the cache until an update is made to the site.

By default, Check for Changes is not enabled. This means that the front-end Web server serves requests for a page that has not yet expired with data from the output cache regardless of whether the underlying, original ASPX page has been modified.

In this test, conducted on a 1x1 server farm, all variables are the same as in the tests for Scale-out characteristics with write operations with the exception of enabling Check for Changes. With Check for Changes on, the cache is flushed more often, resulting in a lower output cache hit ratio.

We recommend avoiding the Check for Changes output cache profile setting except in specific cases. A site using the author-in-place model and experiences infrequent changes in content might benefit from this setting for authenticated users in conjunction with a long cache duration, but other types of sites will most likely see a degradation in RPS.

Depending on your requirements, you might want time-sensitive content to go live instantly or more rapidly than the default settings allow. In this case, you should decrease the duration or not enable output caching.

### Conclusions and recommendations

Output caching does not solve all problems related to capacity management. There are some situations in which output caching is not appropriate, and you should consider these when enabling the output cache and configuring output cache profiles.

## Effect of read volume on CPU, response time

This test was conducted on a 1x1 farm with anonymous access and output caching enabled.

### Conclusions and recommendations

As discussed in Bottlenecks and remediation, server response time will stay generally constant until the front-end Web server receives sufficient request volume to fully utilize its CPU. As front-end Web server CPU utilization maxes out, response time will increase significantly; however, the server farm will still be able to handle some additional request volume.

## Effect of write operations on throughput

The ratio of creation to editing operations is distributed evenly through the course of the tests. WPH values were tested up to approximately 500, because creating/editing more than 500 pages per hour (excepting automated processes such as Content Deployment, which is discussed in Effect of content deployment) is outside the range of what most SharePoint deployments would operate in. These create/edit operations might result in multiple SQL Server operations, so it is important to note that the write’s measured in these tests are not at the SQL Server level, but rather represent what a user would consider a write operation. All RPS vs. WPH tests were conducted on a 1x1 farm.

We first added read operations to the test until front-end Web server CPU was between 60 and 80 percent to leave a buffer for traffic spikes and maintained this average utilization level throughout the course of the test. We then introduced writes, using an artificial delay to control the number of write operations. However, there were spikes of increased front-end Web server and SQL Server CPU usage while the writes were occurring. Some of these spikes for some cache hit ratios exceeded the green zone threshold (see Figure 5 for an example), though the average stayed within the green zone.

Figure 7

As seen in Figure 7, there is a minor reduction in throughput as writes are added to the environment. The graph demonstrates the change in throughput between a read-only scenario and read operations during ~500 writes per hour. Two data points were recorded for each output cache hit ratio; therefore, the relationship between data points is not necessarily linear.

The percentage reduction is more pronounced for lower cache hit ratios, as shown in Figure 8. Overall read RPS remains largely dependent on the cache hit ratio regardless of the writes.

Figure 8

Figure 9 demonstrates that the database server CPU utilization did not increase appreciably when writes were added to the system. Note that the vertical scale is from 0-10 percent of CPU capacity.

Figure 9

Additional SQL Server load was observed during the write operations, which is expected; however, the largest increase was an additional **2.06%**, which is negligible. Average database server CPU utilization stayed below 10% during the course of all tests. Keep in mind that this test was performed on a 1x1 farm. Database server CPU usage will increase as the number of front-end Web servers is scaled out. This is discussed further in *Scale-out characteristics with write operations*.

Front-end Web server CPU utilization was also measured during the tests. Figure 10 demonstrates that average front-end Web server CPU usage remained in the 60-80% range throughout the course of the tests, even as the writes per hour approached 500.

Figure 10

However, the actual measured CPU utilization spikes when the writes occur, as demonstrated in Figure 11. In general, these CPU spikes do not represent a problem; the goal of the green zone is to provide CPU “head room” to absorb some spikes in CPU load. Also, as additional front-end Web servers are added, the effect of the spikes will be distributed across these servers, so the effect to a single front-end Web server CPU will be lessened. It is important to note, however, that such spikes would be expected in a real deployment; write activity is not uniformly distributed though it does tend to happen in bursts.

Figure 11

It is important to note that a 90% cache hit ratio is actually quite low for a typical deployment. Most SharePoint deployments with high volumes of read operations will have an output cache hit ratio of 95% or more.

### Conclusions and recommendations

The data presented indicates that write operations will not have a large adverse effect on the overall throughput of the system for readers. It is important to recognize that write activity can cause spikes in CPU usage and to plan your typical configuration to anticipate these spikes. A strategy for leveling these spikes is to scale out to multiple WFEs. This has the advantage of both spreading out the write load to multiple WFEs, thus smoothing the overall spikes, while also providing higher read RPS, especially at high output cache hit ratios.

## Effect of content deployment

An alternative to a the author-in-place model, in which a single environment used for editing and reading, is to split into two separate environments, an authoring environment and a production environment, and use Content Deployment to copy content from the authoring environment to the production environment.

In this configuration, the production environment has little to no write activity, except when Content Deployment is importing content. For these tests, reads were added until the front-end Web server CPU usage entered the range of 70-80 percent. The content deployment job then exported 10 sites with 500 pages each from the authoring site collection as a package and imported this package into the publishing site collection. The size of the deployed package is larger than typically observed in real world environments in order to sufficiently extend the duration of the content deployment job to observe test results. For additional information regarding characteristics of the deployed content, refer to the Dataset section.

When export was complete, we imported the content into a separate site collection and measured the application server and SQL Server load as well as the throughput while import was in progress. The import test was conducted for several different output cache hit ratios.

The key observation is that import has only a minor effect on overall read RPS, as shown in Figure 12. We also observed that import did not have any appreciable effect on the front-end Web server CPU utilization, as shown in Table 2, regardless of cache hit ratio. However, there was a more noticeable effect on SQL Server CPU, shown in Figure 13. This is expected, because the database server will experience additional load while content is imported in it. However, the SQL Server CPU stayed below 12% usage at all cache hit ratios tested, even during import.

Figure 12

### Effect of content deployment import on front-end Web server CPU utilization

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cache Hit | 100% | 99 | 98 | 95 | 90 | 50 | 0 |
| Baseline | 72.32% | 73.26% | 71.28% | 73.53% | 71.79% | 68.05% | 63.97% |
| With Import | 70.93% | 74.45% | 69.56% | 74.12% | 70.95% | 67.93% | 63.94% |

Table 2

### Effect of content deployment import on database server CPU utilization

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cache Hit | 100% | 99% | 98% | 95% | 90% | 50% | 0% |
| Baseline | 1.19% | 1.64% | 2.01% | 3.00% | 3.73% | 5.40% | 6.82% |
| With Import | 6.03% | 6.82% | 6.97% | 7.96% | 8.52% | 10.29% | 10.56% |

Table 3

### Conclusions and recommendations

The results from our tests show that performing content deployment operations during normal site operations does not pose a significant performance concern. These results show that this is not strictly necessary to deploy content during low-traffic periods to minimize the effect of the operation on overall performance and that deployment times can be driven primarily by business needs rather than performance requirements.

## Effect of database snapshots during content deployment export

In SharePoint Server 2010, content deployment can be configured to create a snapshot of the source content database prior to exporting content from it. This effectively shields the export process from any authoring activity that might be occurring while export is taking place. However, snapshots can affect the write performance of the database server, because the snapshot acts as a multiplier for the writes. For example, if you have two snapshots of a source database, then you write to the source database, database server copies the existing data to each of the snapshots, and then writes the new data into the source database. This means that a single write to the source database incurs three actual writes: one to the source database, and one additional for each of the database snapshots.

To determine the effect of a snapshot on the authoring environment, we measured the write RPS, response time, and the CPU utilization of the front-end Web servers, database server, and application server during an export operation while write activity was also occurring. The results can be seen in Table 3.

|  |  |  |
| --- | --- | --- |
|  | **4 WPH**  **No Snapshots** | **4 WPH**  **With Snapshot** |
| **Write RPS** | 0.2 | 0.16 |
| **Response Time** | 0.13 | 0.15 |
| **Front-end Web server CPU %** | 0.42 | 0.27 |
| **Application Server CPU%** | 8.67 | 8.98 |
| **Database Server CPU %** | 3.34 | 2.41 |
|  |  |  |
|  | **8 WPH**  **No Snapshots** | **8 WPH**  **With Snapshot** |
| **Write RPS** | 0.44 | 0.44 |
| **Response Time** | 0.13 | 0.13 |
| **Front-end Web server CPU %** | 0.61 | 0.73 |
| **Application Server CPU%** | 14.6 | 12 |
| **Database Server CPU %** | 7.04 | 7.86 |

Table 4

### Conclusions and recommendations

The results of our tests showed no significant effect on any measured data points with database snapshots. All variance recorded was within the margin of error. This confirms that database snapshots can be used without strong performance considerations.

## Content characteristics

The tests were conducted on a single dataset that was designed to answer a specific set of questions. Your dataset will differ and will change over time. This section investigates how content characteristics, such as the number of pages in the page library and the features included on pages, can inform capacity management decisions.

### Number of pages

Maximum RPS across a variety of page library sizes were tested. This test was conducted on a 4x1 topology with output caching disabled and anonymous access. All pages were 42 KB uncompressed, 12 KB compressed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Pages** | 3 | 1,000 | 20,000 |
| **Maximum RPS** | 860 | 801 | 790 |

Increasing the number of pages to 20,000 did not have a significant effect on maximum RPS.

### Multivalued lookup fields

A multivalued lookup field is a column on a list that references one or more items in another list, such as columns using enterprise managed metadata. These fields are generally used as search keywords for a page and are not necessarily rendered. In some cases, for example enterprise wikis, it makes sense to render these field values into the contents of pages. For instance, pages might be filed into Categories when created (e.g. World News, Human Interest, Sports on a news site) and the master page includes a placeholder that will show what categories the page was tagged with to the end user.

Under the hood, multivalued lookup fields cause more data to be fetched every time a page is requested, so having many multivalued lookup fields on a page can affect performance. This scenario is tested in detail below:

Maximum RPS degradation occurs as the number of multivalued lookup fields increases due to the effect on the network between the front-end Web server and the database server.

### Effect of usage reporting

Usage reporting is a service that helps administrators monitor statistics about the use of their sites. For more information about SharePoint Foundation usage reporting, see [Configure usage reporting](http://technet.microsoft.com/en-us/library/cc262541.aspx).

We tested the effect of usage reporting timer jobs on maximum RPS on a 1x1 farm.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Usage DB on | Usage DB off | Difference |
| Output cache on | 3459 RPS | 3463 RPS | 4 RPS |
| Output cache off | 629 RPS | 638 RPS | 9 RPS |

The results show that enabling usage logging does not significantly affect RPS in a read-only scenario.

# Appendix

# Test details and approach

* Site Collection Features
  + SharePoint Server Publishing Infrastructure

Output caching is only available when the SharePoint Server Publishing Infrastructure is enabled. By default, Publishing Portals have this feature enabled.

In each test, variables that might be present in the real world have been abstracted to illustrate specific recommendations. Therefore it is critical to test and monitor on your own environment to ensure you have scaled correctly to meet the request volume you expect. To learn more about capacity management concepts, you can refer to the Capacity Management Overview.

## Hardware

### Front-end Web servers and application servers

The number of Web servers in the farm varied by test, but each had identical hardware.

|  |  |
| --- | --- |
| Web Server | WFE |
| Processor(s) | 2 quad core @ 2.33 GHz |
| RAM | 8 GB |
| Operating System | Windows Server® 2008, 64 bit |
| Size of the SharePoint drive | 300 GB |
| Number of NICs | 2 |
| NIC Speed | 1 gigabit |
| Authentication | Windows Basic |
| Load balancer type | Hardware load balancing |
| Software version | SharePoint Server 2010 (pre-release version) |
| Services running locally | Central Administration  Microsoft SharePoint Foundation Incoming E-Mail  Microsoft SharePoint Foundation Web Application  Microsoft SharePoint Foundation Workflow Timer Service |

### Database Server

|  |  |
| --- | --- |
| Database Server | DB1-2 |
| Processor(s) | 4 quad core @ 3.19 GHz |
| RAM | 16 GB |
| Operating System | Windows Server 2008, 64-bit |
| Storage | 15 disks of 300 GB @ 15,000 RPM |
| Number of NICs | 2 |
| NIC Speed | 1 |
| Authentication | NTLM |
| Software version | SQL Server 2008 |

## Dataset

The tests were conducted by using a dataset that shares common characteristics with actual WCM deployments. Although load was constant, different pages were requested.

|  |  |
| --- | --- |
| **Object** | **Publishing Site** |
| Size of content databases | 2.63 GB |
| Number of content databases | 1 |
| Number of site collections | 1 |
| Number of Web applications | 1 |
| Number of sites | 50 |
| Number of Pages | 20,000 pages, divided into 20 folders with 1,000 pages each |
| Composition of Pages | Article pages in basic HTML and with references to two images |
| Page Size | 42 KB uncompressed; 12 KB compressed |
| Images | 3,000, 30 KB to 1.3 MB each |

We recommend configuring Internet Information Services (IIS) to always compress files instead of the default setting to dynamically compress files. When dynamic compression is enabled, IIS compresses pages until CPU utilization exceeds a certain threshold, at which point IIS ceases to compress pages until utilization drops below the threshold. The tests in this white paper were conducted with compression always on.

This test dataset used only default, out-of-box SharePoint features. Your site probably includes customizations in addition to these basic features, so it is important to test on your own solution.