Kentico CMS 6.0

Performance Test Report

February 2012

Kentico
unlimited website possibilities

www.kentico.com

Table of Contents

Disclaimer

This report was conducted by Kentico Software with intention to provide customers with information on what performance they can expect from Kentico CMS. Kentico Software put in the best effort to conduct an unbiased test. Still, the performance of the website depends on many parameters, such as computer hardware, network configuration, client configuration, operating system and software configuration, site content, number of items in Kentico CMS database, information architecture, custom code and other factors. Kentico Software doesn't provide any guarantee that the same values will be achieved with other than tested configurations. The reader of this report uses all information in this report at his/her own risk. Kentico Software shall in no case be liable for any loss resulting from the use of this report.

Executive Summary

Kentico CMS for ASP.NET provides excellent performance and scalability. Being built on the Microsoft ASP.NET platform, it leverages all its power.

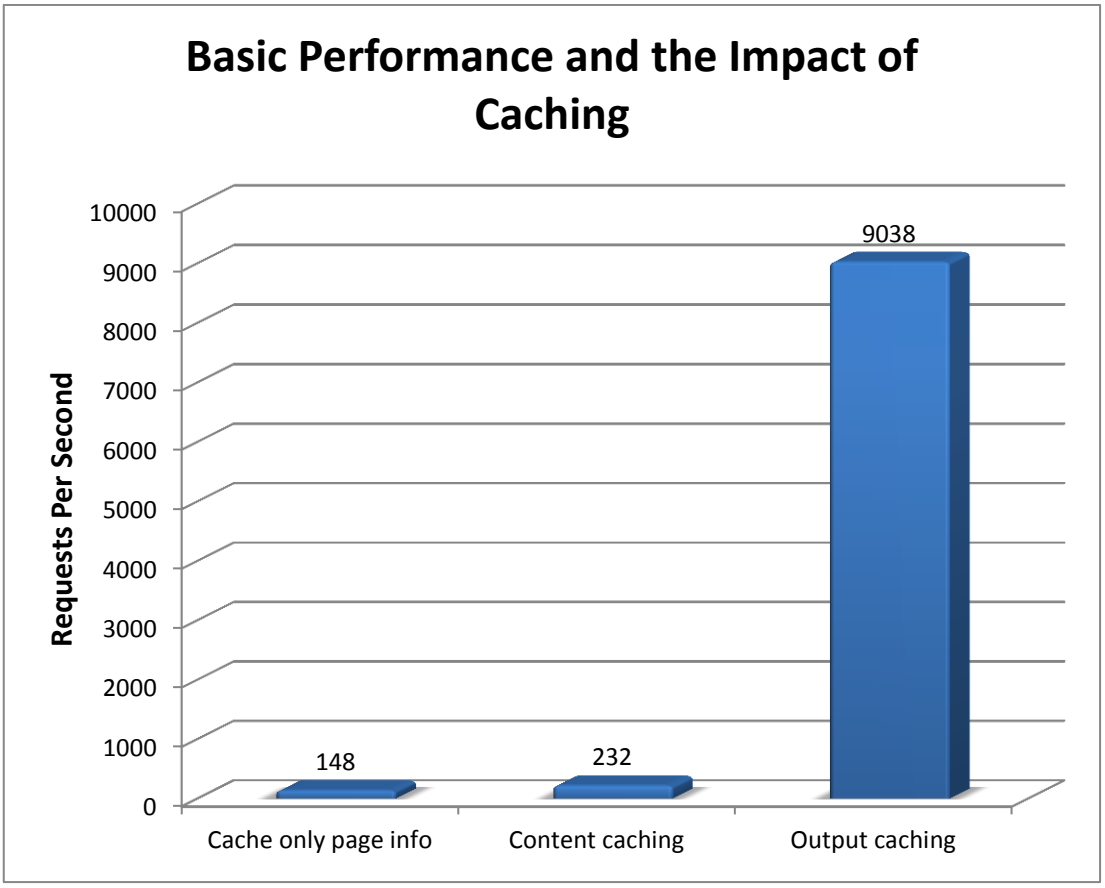
The tests were performed internally by Kentico staff on very common hardware (Intel Core 2 Quad at 2.66 GHz, 10k SATA II disks, 4 GB memory). Kentico has not used any high performance servers, so the results may be even better on more powerful hardware.

Basic Performance and the Impact of Caching

The slowest parts of a web application are typically accessing the database and rendering the content for a web browser. Kentico CMS optimizes the performance by storing content that is often accessed in a server memory. This mechanism is called **caching**. When another visitor comes to the same page, the page is already stored in the very fast computer memory and Kentico CMS can quickly send it to the browser without repeatedly accessing the database and rendering the page. The following graph shows a comparison of how caching influences the performance of Kentico CMS (the test was conducted on a single machine with both web server and database).

Caching can be configured for a particular part of the page - this is called **content caching**. You can also configure **Output caching** that stores the whole page pre-rendered in the memory.

The following figure shows the impact of caching on the overall performance. The values represent the number of requests per second (RPS) which means how many pages can be viewed by the visitors per second.

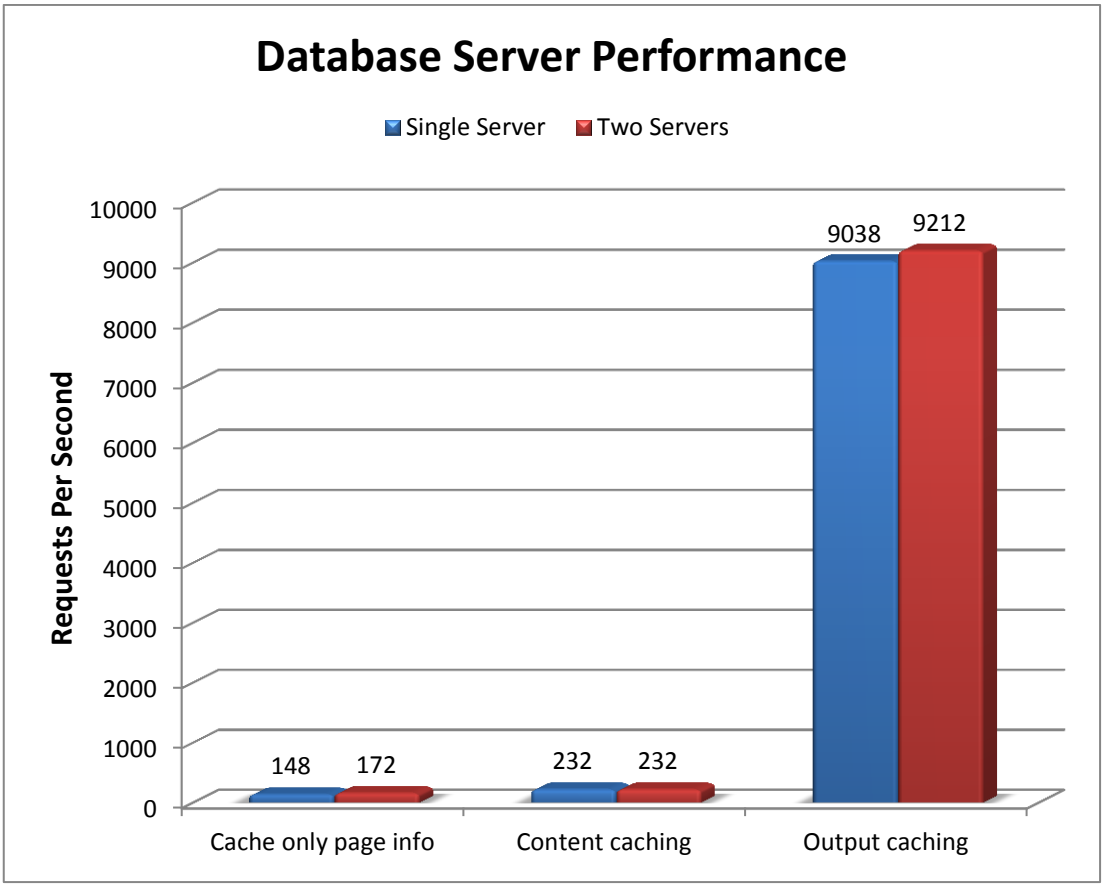


Comments: As you can see, Kentico CMS provides a highly efficient caching mechanism that boosts the performance significantly.

Database Server Performance

If you need high performance, it's recommended that you install the web server and the database server on two different machines which allows you to distribute the computing and achieve shorter response time, especially if you cannot use caching.

The following figure compares the performance of a single server configuration and a two-server configuration (Web Server + Database Server):



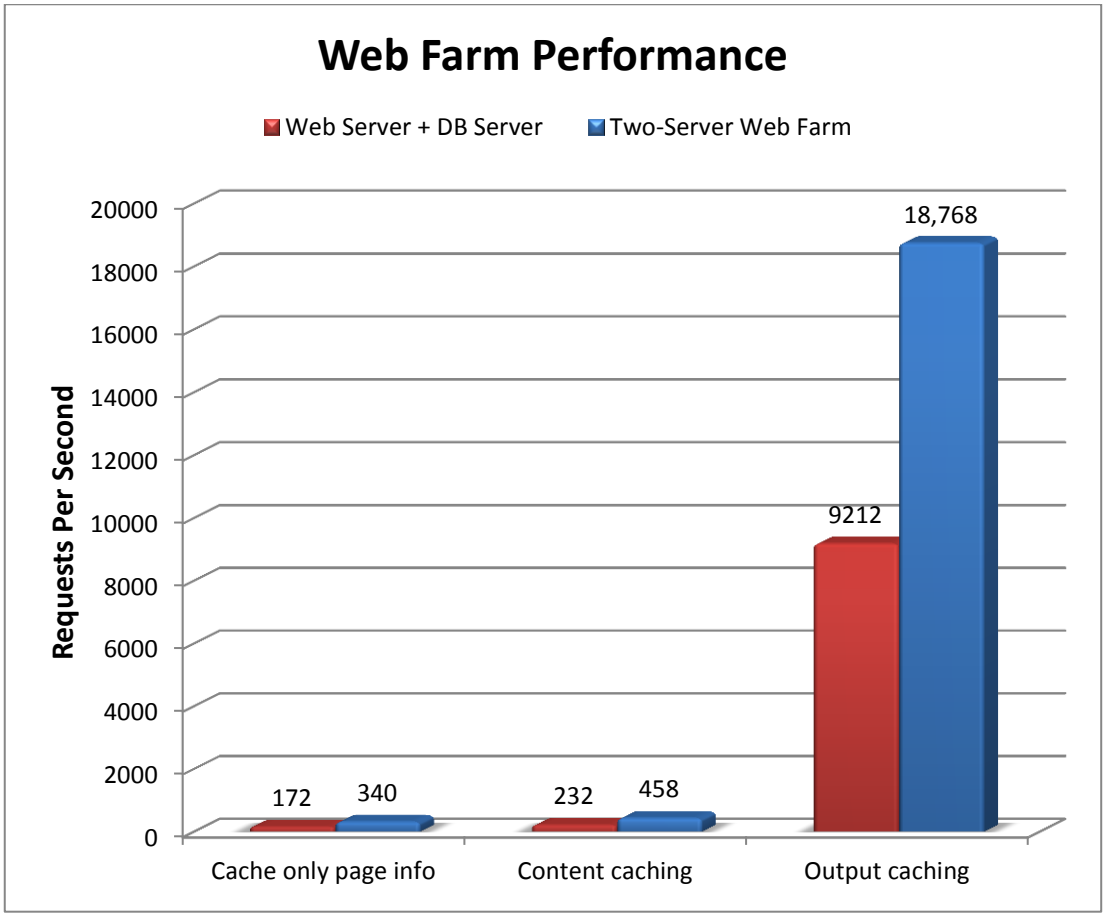
Comments: As you can see, the performance gain from a dedicated database server is most significant when the caching is not used.

Kentico CMS also allows you to distribute the SQL Server database on multiple database servers using **SQL Server Merge Replication** which allows for virtually **unlimited database performance and high availability**.

Web Farm Performance

Web farms allow you to distribute the computing among **multiple web servers** that all provide the same content. It can also be used for achieving **high availability** of your site - if some server in the web farm stops working, the other server(s) will serve the content instead of the broken server.

The following figure compares the performance of a single web server and two web servers (each configuration uses a single dedicated database server):

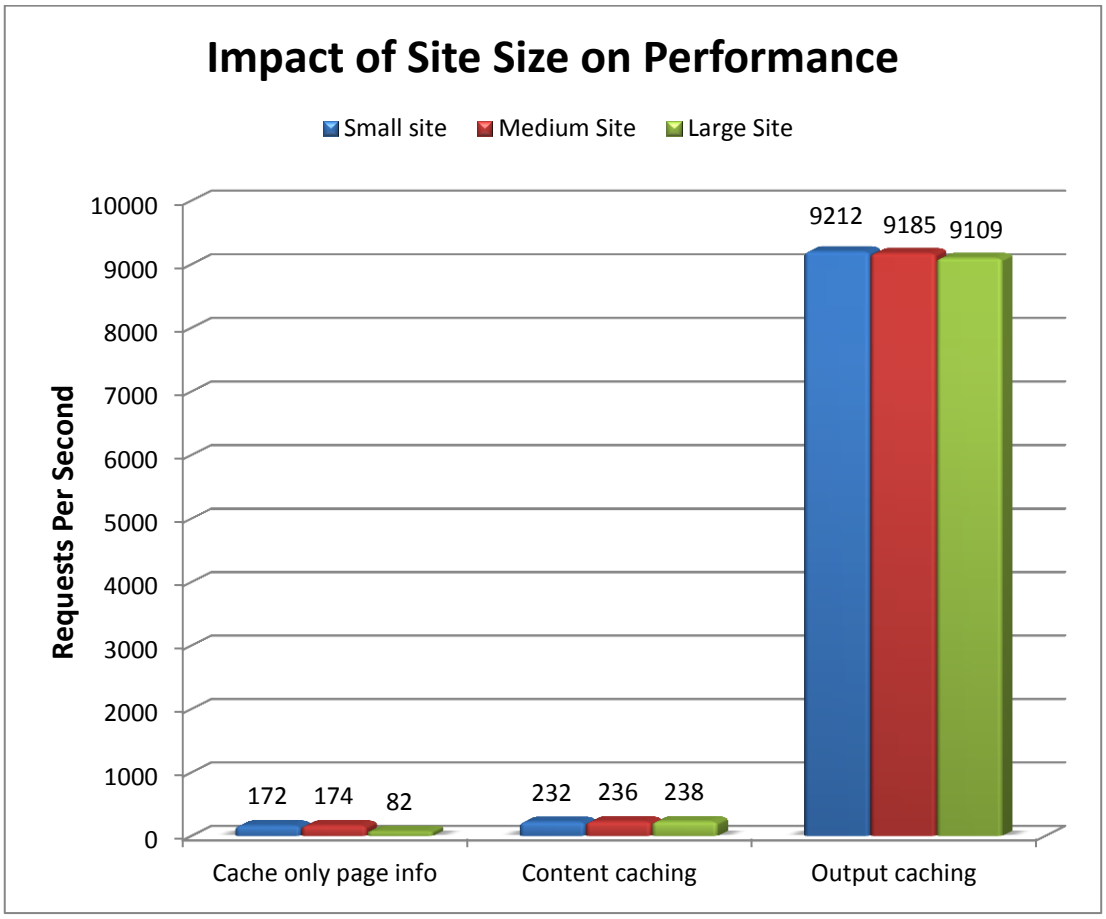


Comments: The results show that Kentico CMS provides excellent scalability. With an additional web server, the performance grows by 100%.

Impact of Site Size on Performance

The website performance depends also on the number of pages and other items in the CMS database. Kentico CMS was optimized for high number of items and it was tested with **100,000 documents and 10,000,000 users** stored in the database. Not only the public website, but also Kentico CMS administration interface can handle this number of items without a negative effect on usability and user interface responsiveness.

The following figure compares the performance of a small site (325 documents), medium site (10,000 documents) and a large site (100,000 documents), both running on a single web server and a dedicated SQL Server:



Comments: the performance without any caching drops with a very large number of documents. This scenario requires site-specific optimization of SQL queries and database indexes. For instance, we were able to improve the performance from 28 RPS to 174 RPS for a website with 10,000 documents. The performance with caching is practically same regardless of the site size.

Cloud Computing

Kentico CMS can be used also with cloud computing platforms that provide both high performance and availability. Kentico CMS supports natively Microsoft Windows Azure (including Windows Azure Storage) and Amazon EC2 (including Amazon S3 Storage) and we have numerous clients who use Kentico CMS on these platforms. Our customers use also other cloud platforms, such as Rackspace Cloud.

Kentico CMS can be run on multiple instances in both Windows Azure and Amazon EC2 which makes you ready for virtually unlimited traffic. You can also leverage their CDN capabilities to optimize the delivery of uploaded files, such as images, video, PDF documents, etc.

Server Hardware Configurations

CLIENT COMPUTERS AND WEB SERVERS

Motherboard: Giga-Byte EP45C-DS3R (Intel® Socket 775, Intel® P45)

Processor: Intel® Core™2 Quad Q9400 BOX (2.66GHz)

Graphics card: MSI R4350-D512H PCIE 512MB DDR2 SDRAM

Memory: Kingston 2G-UDIMM (PC2-6400)

Capacity	4 GB (Kit 2x 2 GB)
Type	DDR2-SDRAM
Memory Bus	1066 MHz

HDD: 1x Western Digital WD1500HLFS VelociRaptor 5RZ

Format	3.5”
Capacity	150 GB
Interface	SATA II
Speed	10000rpm
Access Time	4.2ms / 4.7ms
Cache	16 MB

DATABASE SERVER

Memory: Kingston 4G-UDIMM (PC2-6400)

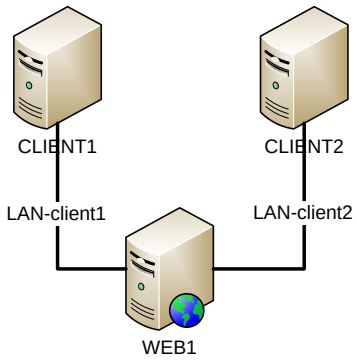
Capacity	16 GB (Kit 4x 4 GB)
Type	DDR2-SDRAM
Memory Bus	1066 MHz

(other parameters of the database server were same as for the other configurations)

Testing Configurations

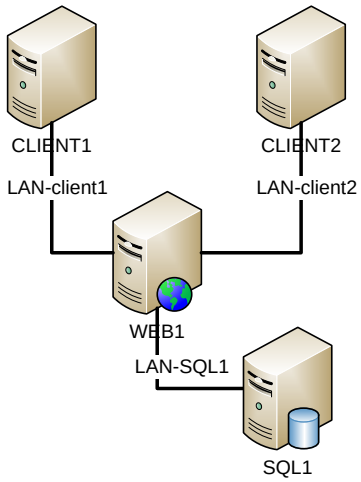
CONFIGURATION A - SINGLE SHARED SERVER (WEB + DATABASE)

Windows Server 2008 Service Pack 2, Microsoft SQL Server 2008



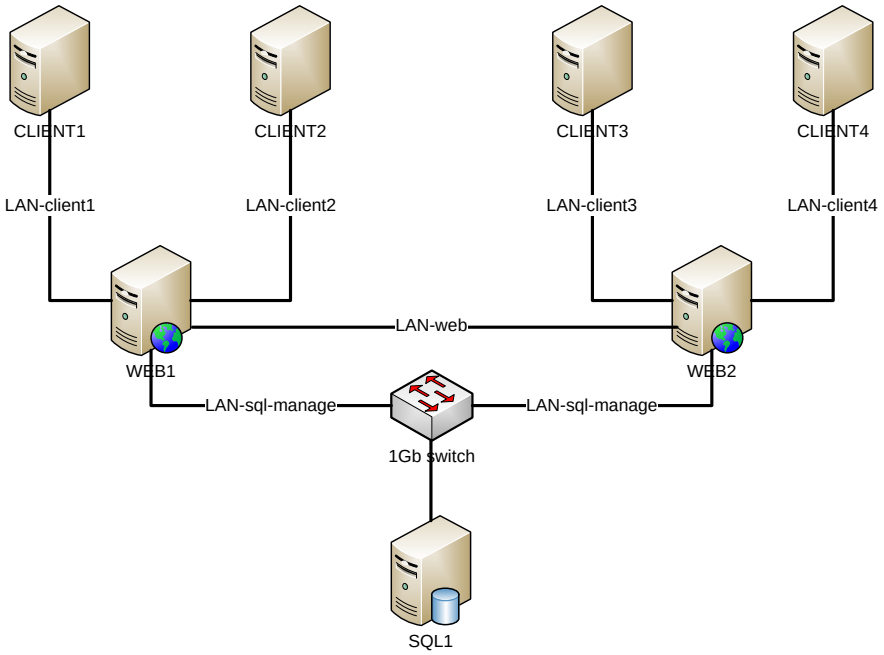
CONFIGURATION B - TWO SEPARATE SERVERS (A WEB SERVER AND A DATABASE SERVER)

Windows Server 2008 Service Pack 2, Microsoft SQL Server 2008 R2



CONFIGURATION C – TWO SERVERS IN A WEB FARM AND A DATABASE SERVER

Windows Server 2008 Service Pack 2, Microsoft SQL Server 2008 R2



Caching Configurations

Cache 1 - Cache only page info: Page info (page data and metadata) cache enabled for 10 minutes

Cache 2 - Content caching: content caching (web part/control-level caching) enabled for 10 minutes, with default filters used[1]

Cache 3 - Output caching: Output caching (whole page cached in memory) enabled for 10 minutes

[1] The use of additional XHTML filters decreases performance by around 30% when using content caching - this can be solved by writing XHTML-compliant code by developers, so that the XHTML filter doesn't have to be used.

How the Tests Were Performed

- All tests were performed for 15 minutes (+1 minute warm-up time)
- Domain name was in the host file (localhost was not used)
- Tests were executed in Visual Studio Enterprise Architect – Application Center Test
- 50 simultaneous browser connections from each client computer (100% new users)
- HTTP, DNS, Socket, Windows or Kentico CMS errors were NOT allowed during tests
- Kentico CMS restart was NOT allowed during one test run
- IIS was restarted before every test execution
- Kentico CMS used GZip compression
- Network traffic was measured on the server side and it was the sum of all interfaces connected to the clients

Which Performance Counters Were Watched

- Processor(_Total)\% Processor Time
- Process(w3wp)\Thread Count
- Process(w3wp)\Handle Count
- Process(w3wp)\Private Bytes
- Process(w3wp)\Virtual Bytes
- .NET CLR Memory(w3wp)\# Bytes in all Heaps
- .NET CLR Memory(w3wp)\# Gen 0 Collections
- .NET CLR Memory(w3wp)\# Gen 1 Collections
- .NET CLR Memory(w3wp)\# Gen 2 Collections
- .NET CLR Memory(w3wp)\% Time in GC
- .NET CLR Exceptions\# Exceps thrown / sec
- ASP.NET\Application Restarts
- ASP.NET\Requests Rejected
- ASP.NET\Worker Process Restarts
- Memory\Available Mbytes
- PhysicalDisk(_Total)\Disk Read Bytes/sec
- PhysicalDisk(_Total)\Disk Write Bytes/sec
- Network Interface\Bytes Received/sec
- Network Interface\Bytes Sent/sec

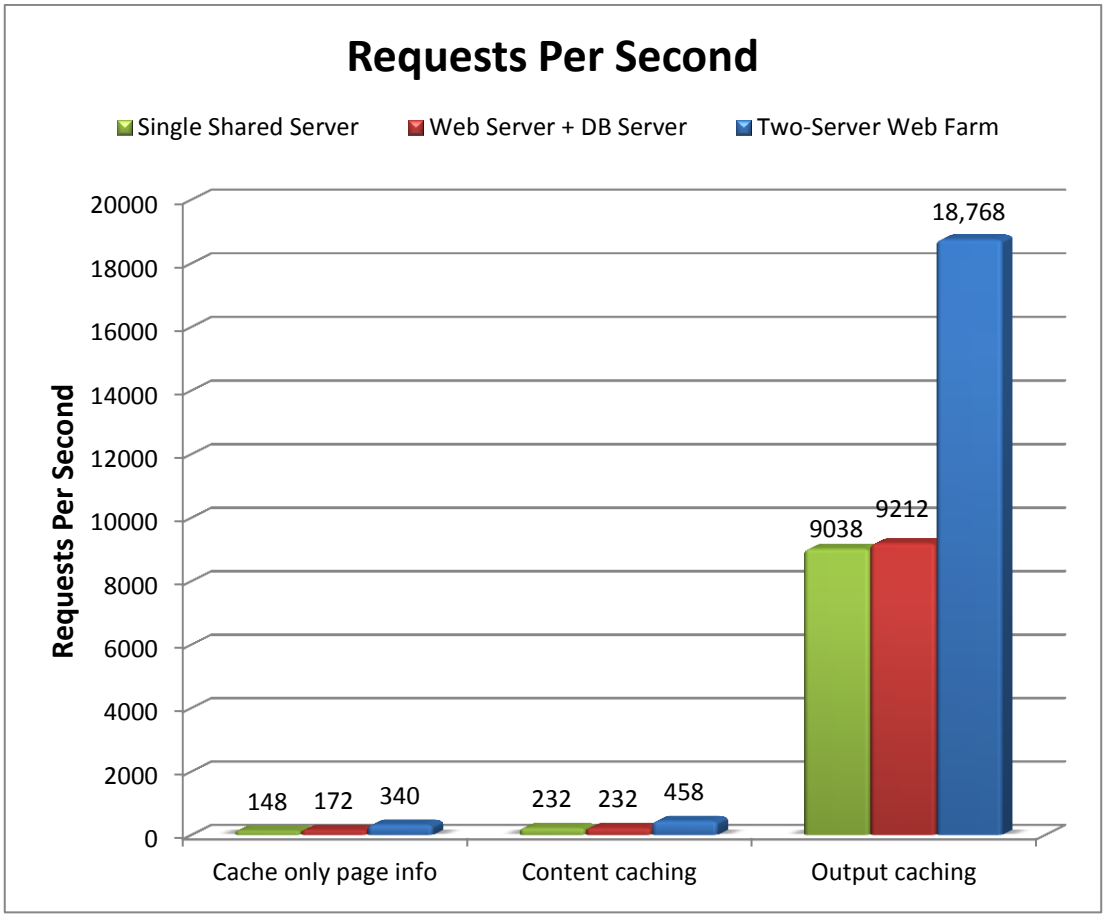
Performance Test Results

The tests were based on visiting the following pages:

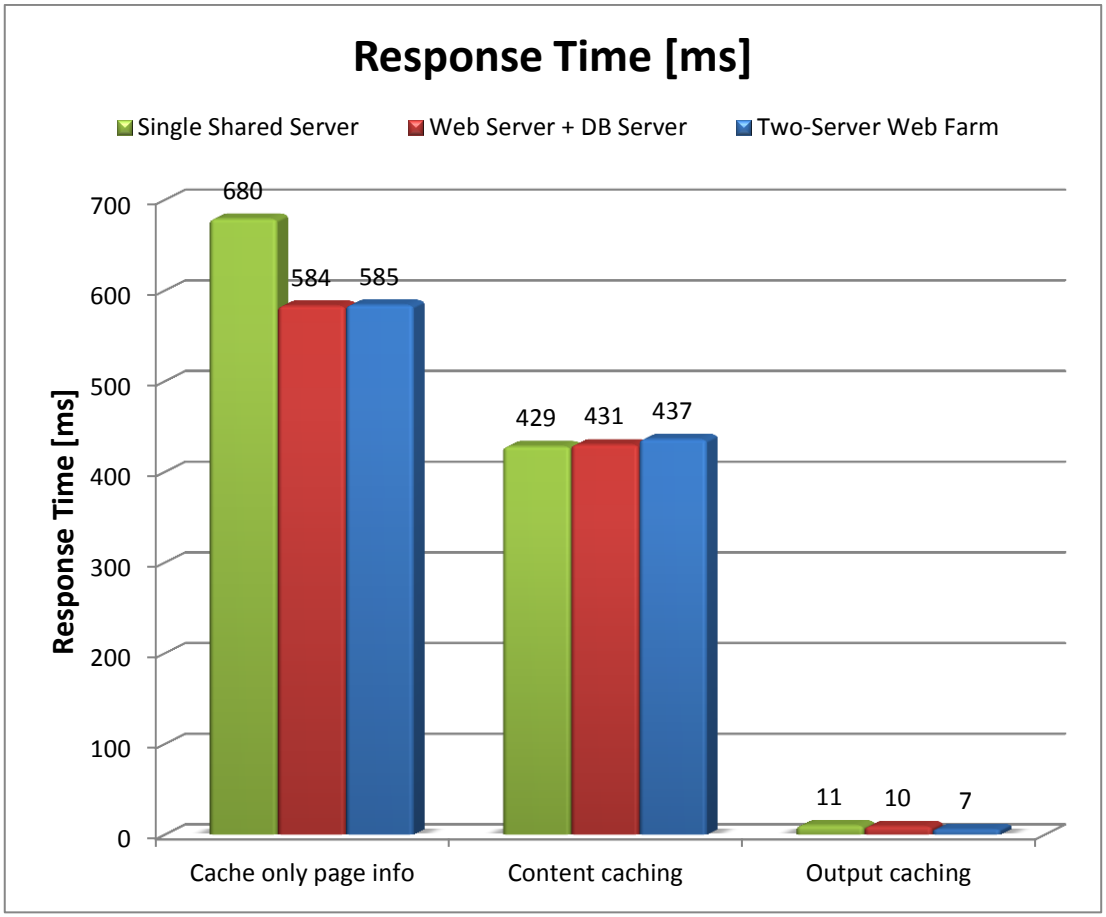
SAMPLE CORPORATE SITE (FROM KENTICO CMS 5.0)

~/Home.aspx
~/Services.aspx
~/Network-administration.aspx
~/Products.aspx
~/Products/Cell-phones/Samsung-SGH-E250.aspx

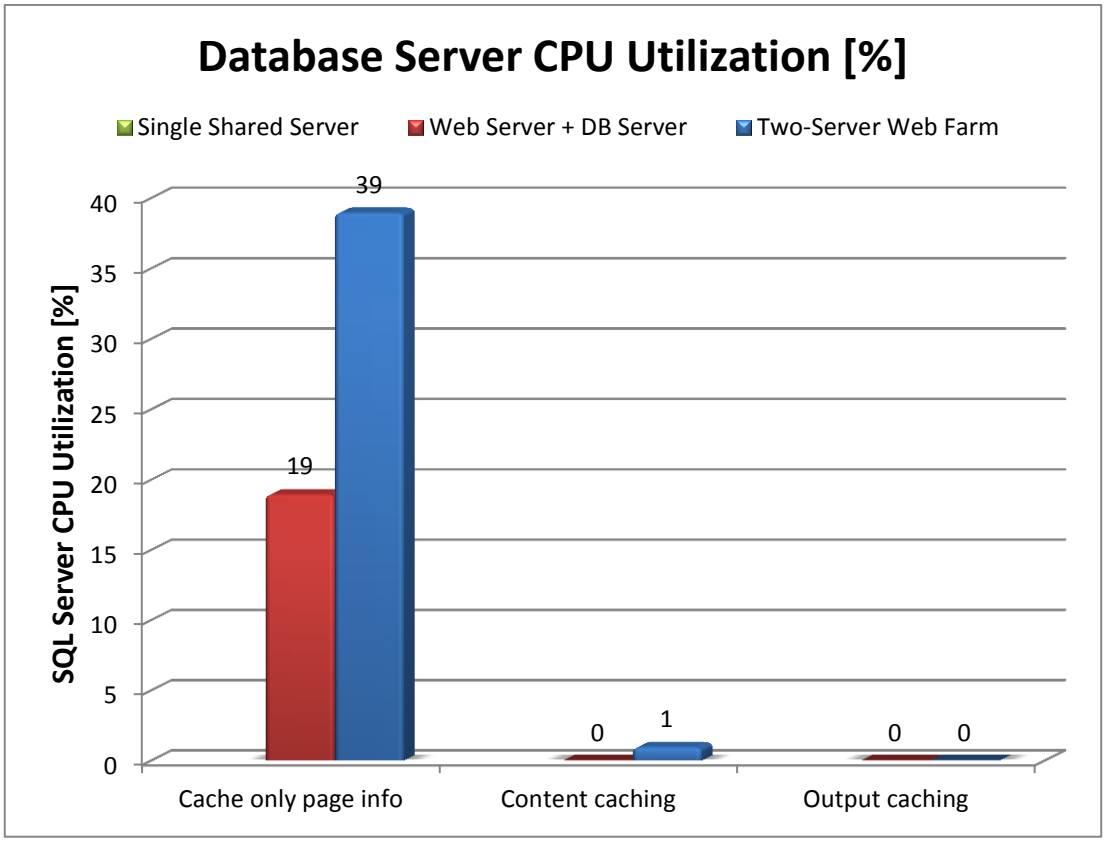
	AVG requests per second	AVG requests per hour	AVG requests per 24 hours
Single Server for Both Website and Database			
Cache only page info	148	496 138	12 534 006
Cache content	232	770 247	19 458 874
Output cache	9 038	30 744 421	776 701 164
Web Server + DB Server			
Cache only page info	172	576 593	14 566 548
Cache content	232	770 247	19 458 874
Output cache	9 212	31 336 314	791 654 251
Two Web Servers in a Web Farm + DB Server			
Cache only page info	338	1 153 185	29 133 096
Cache content	450	1 540 494	38 917 748
Output cache	18 325	62 672 628	1 583 308 502



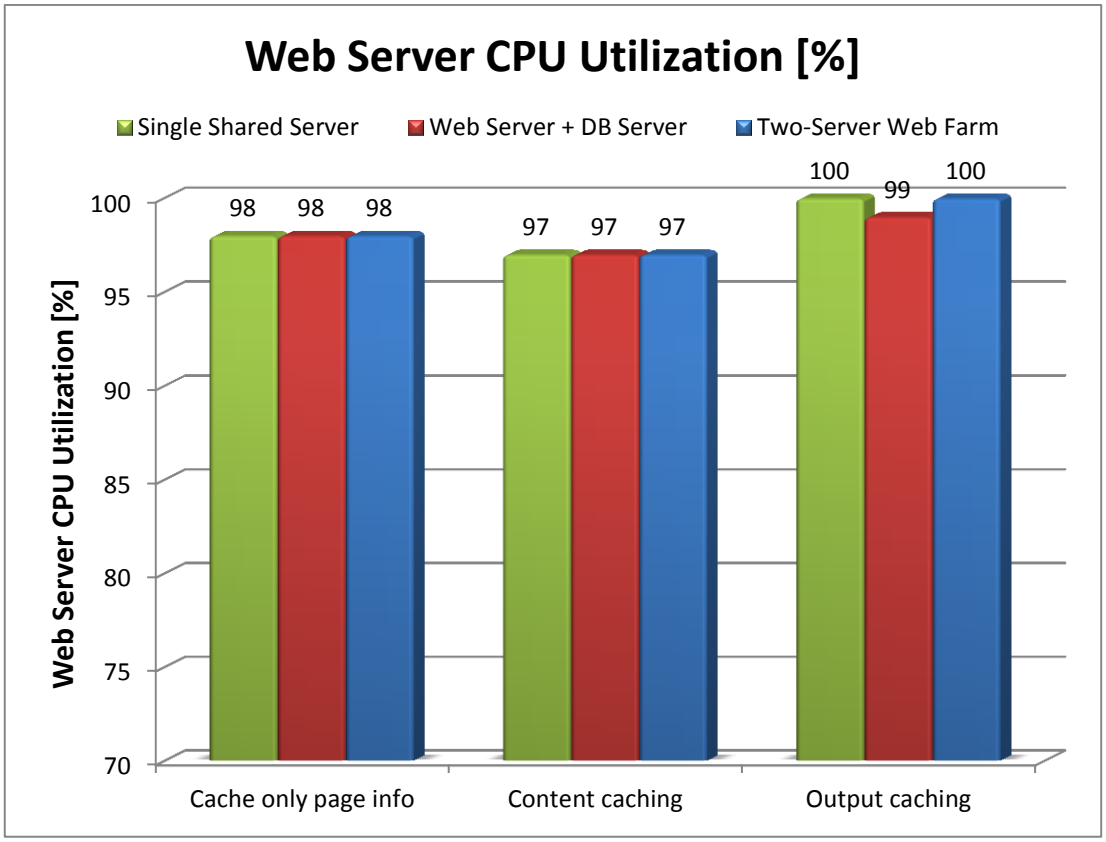
Comments: Here you can see that output caching provides the best possible performance. The graph also shows that using a web farm multiplies the overall performance by the number of web servers.



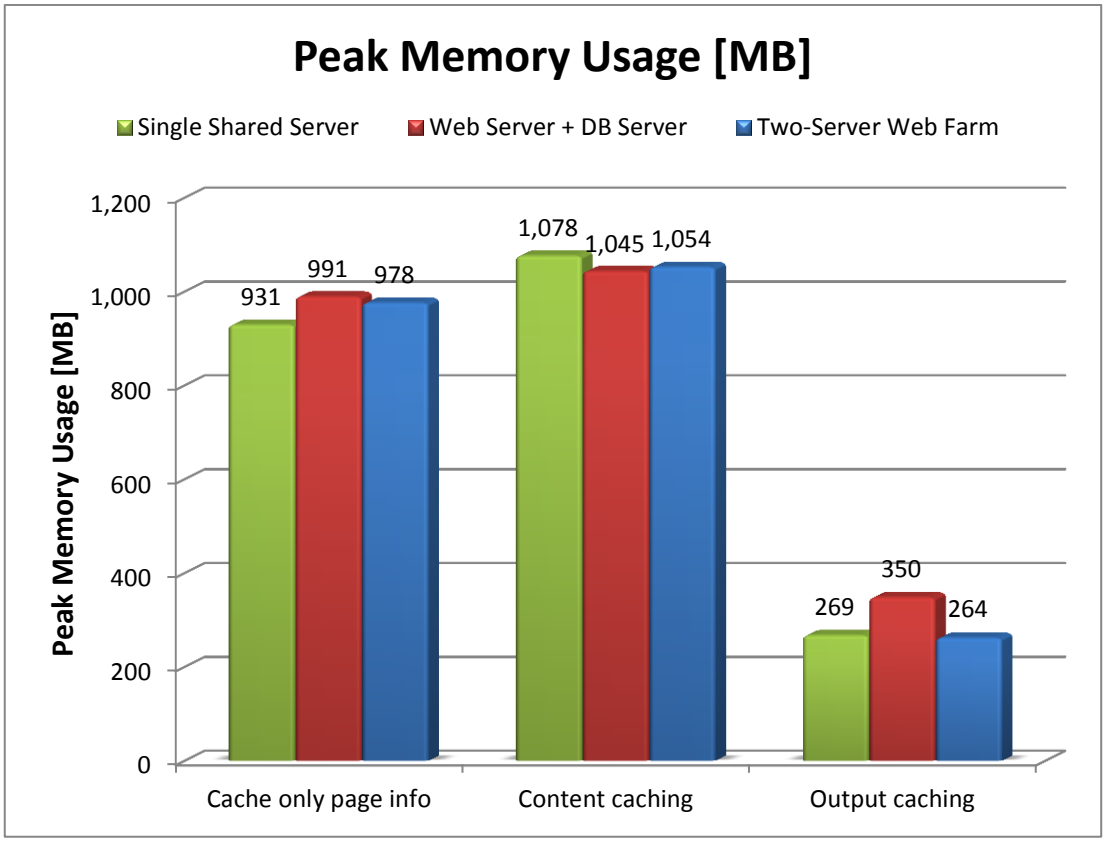
Comments: The response time highly depends on the amount of cycles that need to be done to render the requested page. The Output caching only takes the HTML code cached in the memory and sends it to the browser, so it's extremely fast.



Comments: The graph shows that once caching is used, the database server utilization is very low since the website only accesses the database when the content is requested for the first time. The results for Single Shared Server were not measured since the database server shared the CPU with the web server.

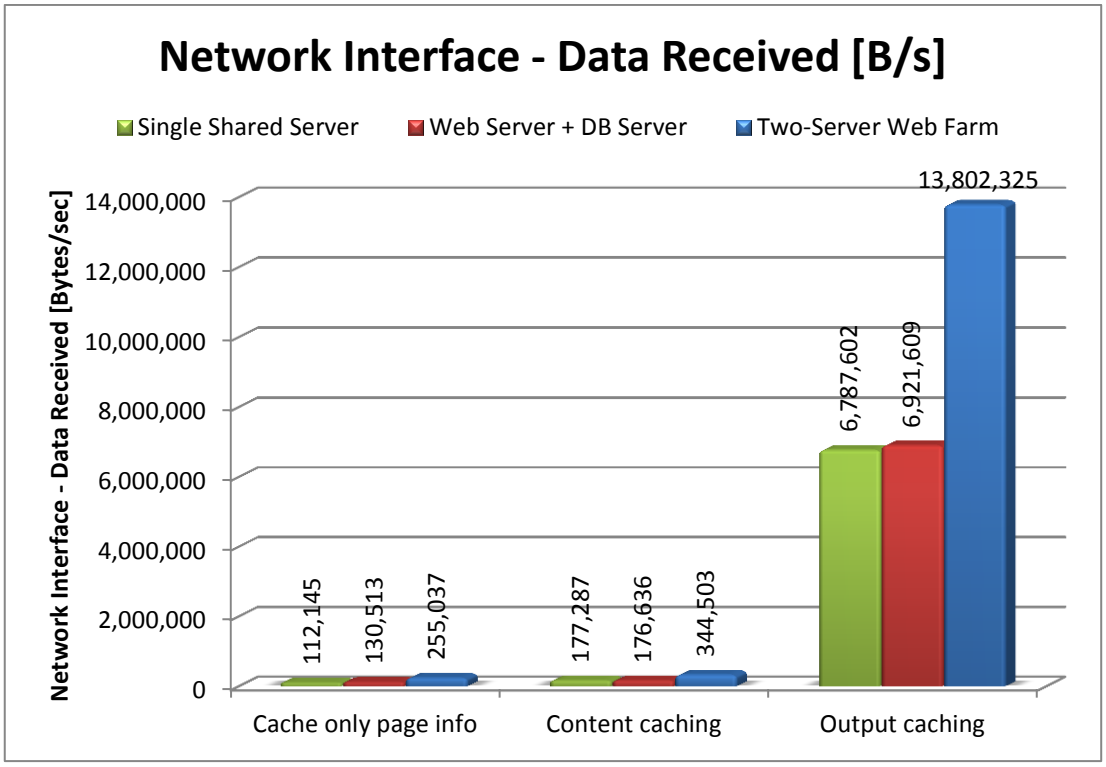


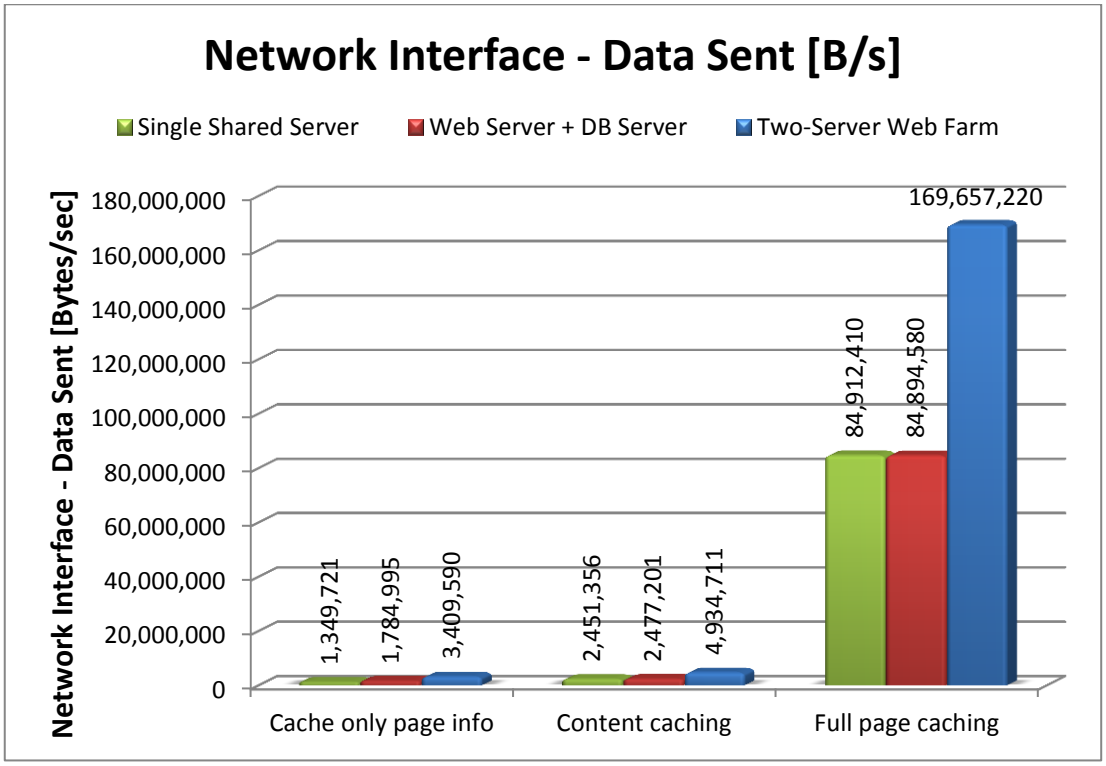
Comments: The graph shows the utilization of the processor by all processes running on the server. The Single Shared Server configuration includes both the web server process and SQL Server process. Since we tested for the highest performance, the utilization tends to reach 100%. The only exception are configurations without caching (Cache only page info) that depend on an external database server that slows down the Web server CPU utilization since the Web server has to wait for the database server.



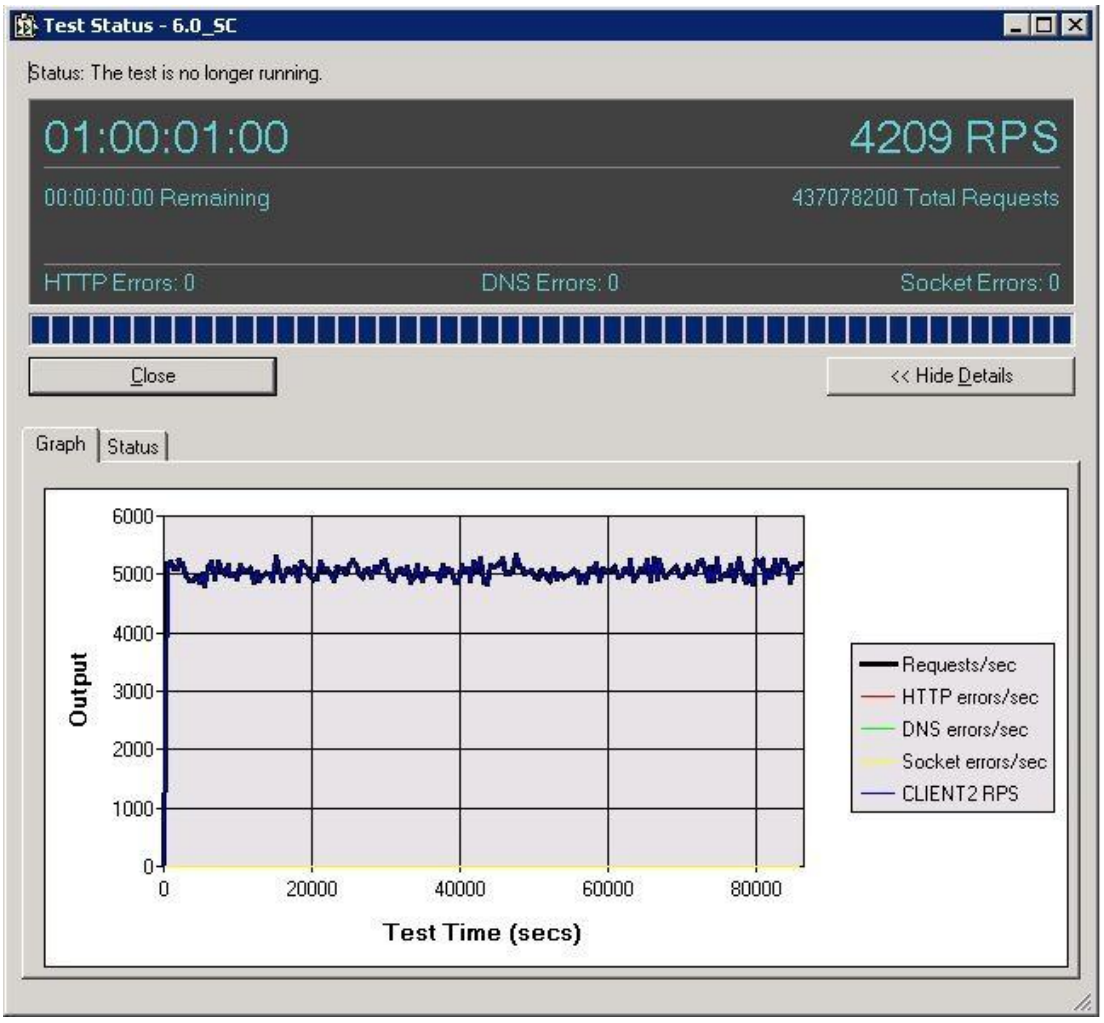
Comments: This test was performed for 24 hours and the web server process didn't required any restarts during that period which shows there were no memory leaks that would crash the process. The Output caching requires less memory since it creates a smaller number of .NET objects in the memory (that are destroyed by the garbage collector then). The results for Output caching may, however, look different if you have a large site with many pages that would be stored in the memory.

Without output caching, ASP.NET session objects occupied more than 60% of the allocated memory because each test created a new session and the garbage collector needed to wait for the session timeout to release the memory.


Network Interface - Data Received [B/s]
Single Shared Server
Web Server + DB Server
Two-Server Web Farm
Network Interface - Data Received [Bytes/sec]
14,000,000
12,000,000
10,000,000
8,000,000
6,000,000
4,000,000
2,000,000
0
112,145
130,513
255,037
177,287
176,636
344,503
6,787,602
6,921,609
13,802,325
Cache only page info
Content caching
Output caching


Network Interface - Data Sent [B/s]
Single Shared Server
Web Server + DB Server
Two-Server Web Farm
Network Interface - Data Sent [Bytes/sec]
180,000,000
160,000,000
140,000,000
120,000,000
100,000,000
80,000,000
60,000,000
40,000,000
20,000,000
0
1,349,721
1,784,995
3,409,590
2,451,356
2,477,201
4,934,711
84,912,410
84,894,580
169,657,220
Cache only page info
Content caching
Full page caching

24-hour Stability Test Results



Comments: The picture shows the Requests per Second (RPS) values on a client computer during the 24-hour stability test of the "Output caching" configuration. The tests were performed on a two-server web farm with a dedicated database server. The results show that Kentico CMS provided a stable performance during the whole testing period, without any crashes, errors or downtimes. The system has shown the same stability also for the "Cache only page info" and "Content caching" configurations.

Performance Test for Files (File, Media File)

Tests were based on:

Server configuration: Two separate servers for Web and database
Image file: Microsoft.jpg – 1024x768px 145.11KB (and its resized versions)

GETFILE - SERVES ALL FILES UPLOADED INTO THE CONTENT REPOSITORY:

~/GetFile/62bafe6e-0e74-4c1c-a319-ce3463c22449/Microsoft.aspx
~/GetFile/62bafe6e-0e74-4c1c-a319-ce3463c22449/Microsoft.aspx?width=320
~/GetFile/62bafe6e-0e74-4c1c-a319-ce3463c22449/Microsoft.aspx?width=640

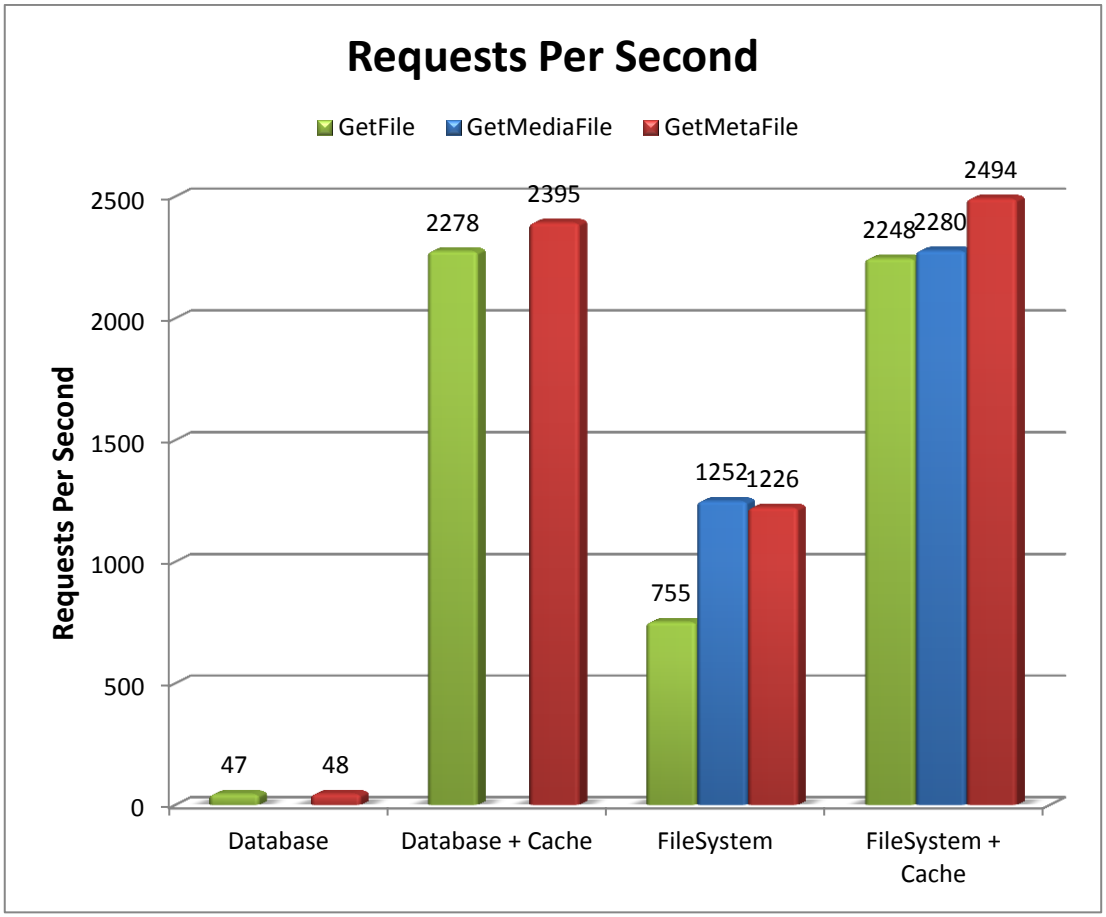
GETMETAFILE - SERVES ALL FILES UPLOADED TO THE PRODUCTS:

~/GetMetaFile/ da2495b2-ff5f-47cb-b463-4b99d308eadd/Microsoft.aspx
~/GetMetaFile/ da2495b2-ff5f-47cb-b463-4b99d308eadd/Microsoft.aspx?width=320
~/GetMetaFile/ da2495b2-ff5f-47cb-b463-4b99d308eadd/Microsoft.aspx?width=640

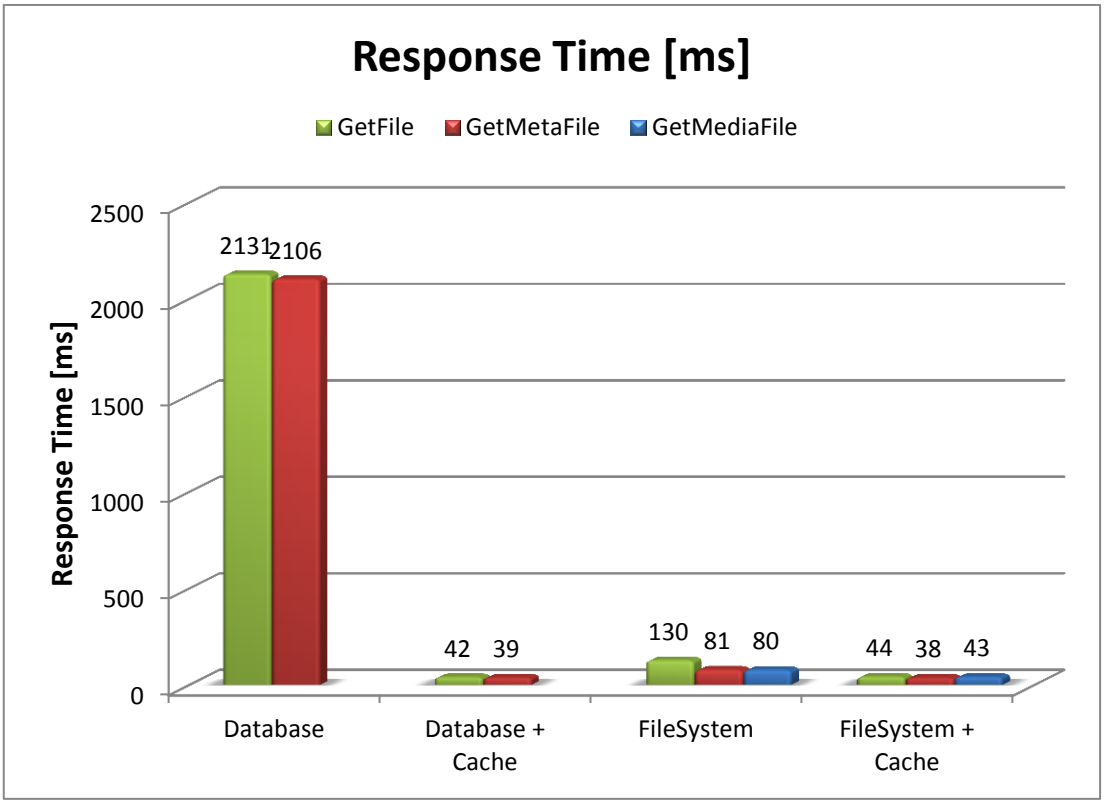
GETMEDIAFILE - SERVES ALL FILES STORED IN THE MEDIA LIBRARY:

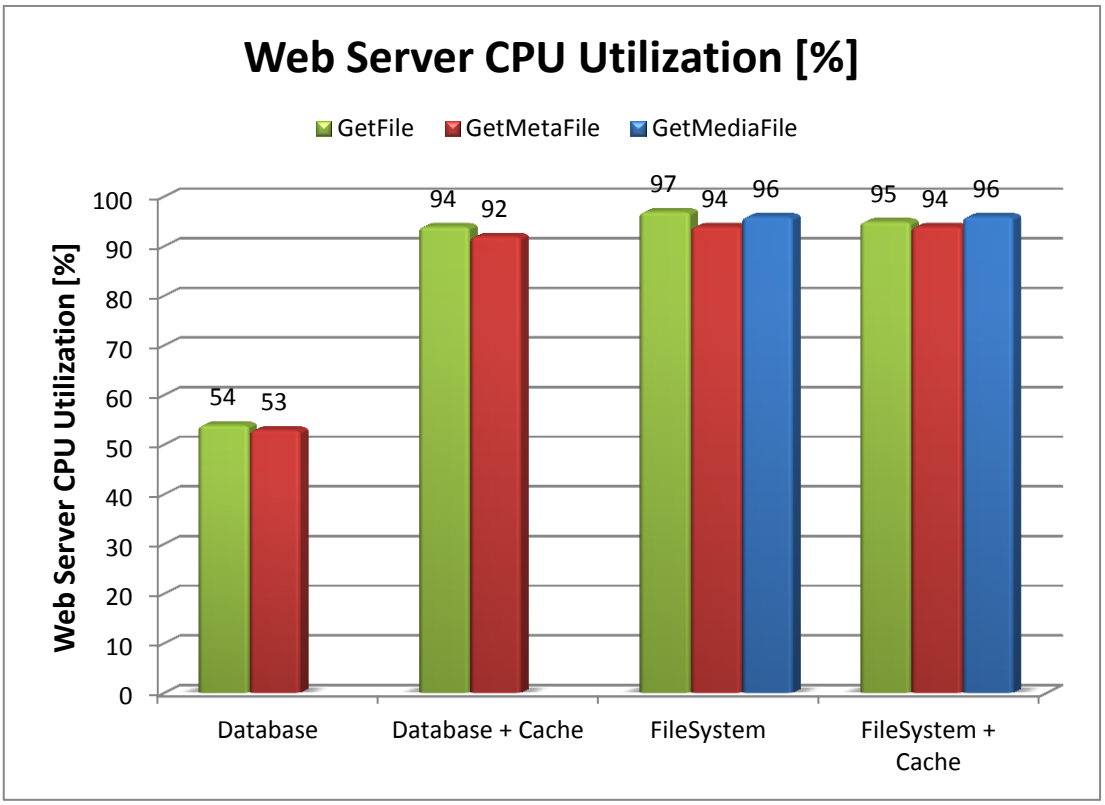
~/GetMedia/20fdd53d-ea61-49a1-94f4-0d7bbdebf559/Microsoft.aspx
~/GetMedia/20fdd53d-ea61-49a1-94f4-0d7bbdebf559/Microsoft.aspx?width=320
~/GetMedia/20fdd53d-ea61-49a1-94f4-0d7bbdebf559/Microsoft.aspx?width=640

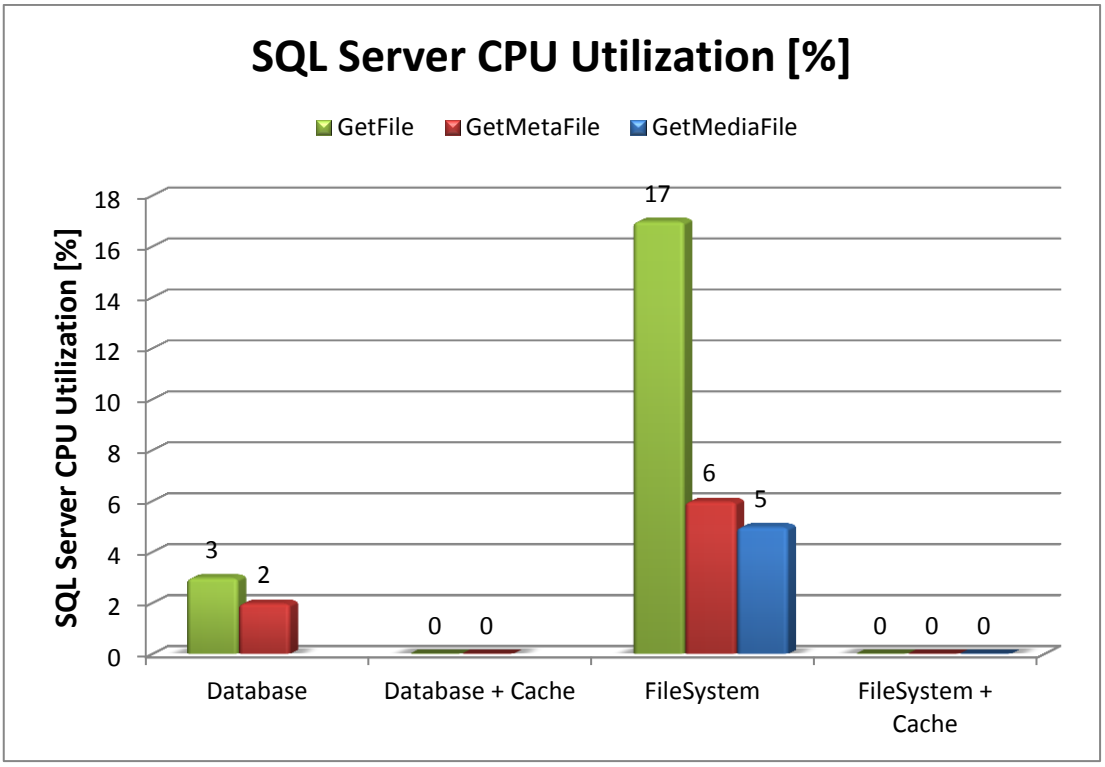
	AVG requests per second	AVG requests per hour	AVG requests per 24 hours
GetFile			
Database	47	167 445	4 018 680
Database + Cache	2 278	7 883 922	199 172 763
File System	755	2 617 653	66 130 172
File System + Cache	2 248	7 437 575	187 896 639
GetMetaFile			
Database	48	170 290	4 086 960
Database + Cache	2 395	8 061 671	203 663 273
File System	1 226	4 109 038	103 807 275
File System + Cache	2 494	8 283 117	209 257 688
GetMediaFile			
File System	1 252	4 058 868	102 539 826
File System + Cache	2 280	7 390 514	186 707 719



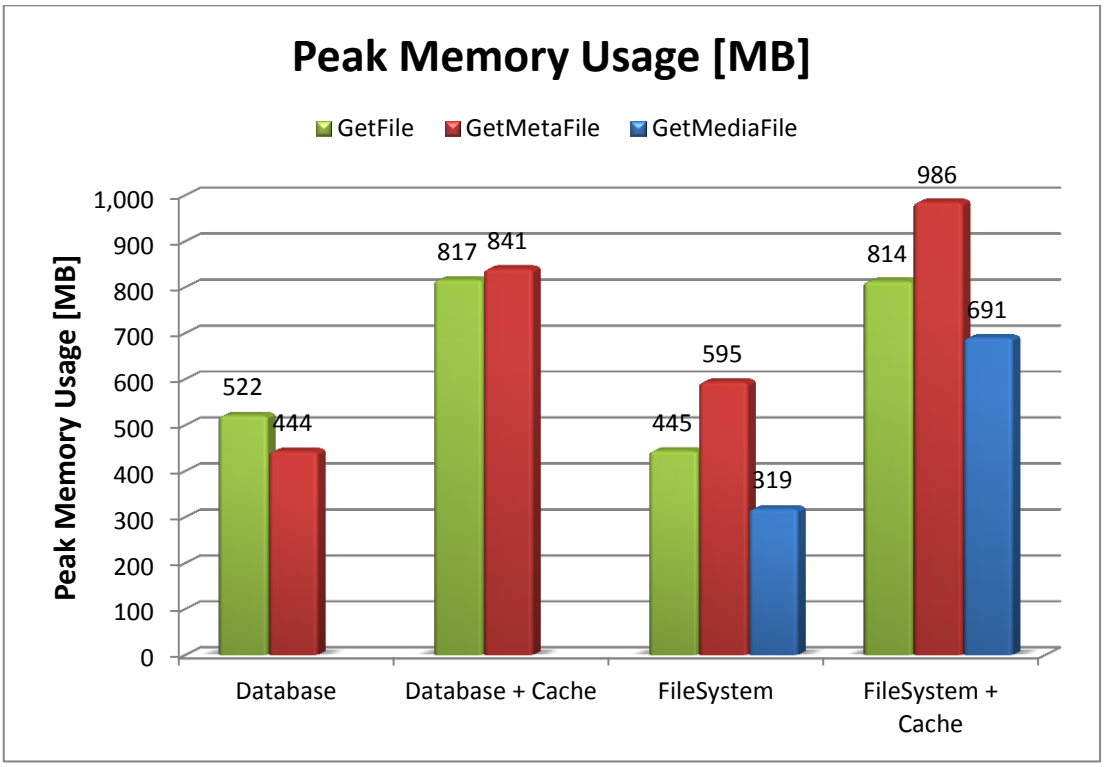
Comments: GetFile is generally slower because of dealing with the related document (not just the file). If you plan to work with many media files or with large files, such as video or music, it's recommended that you use the Media Library instead of storing these files in the content tree. The media files can be stored only in the file system which is why GetMediaFile test wasn't performed for database storage.

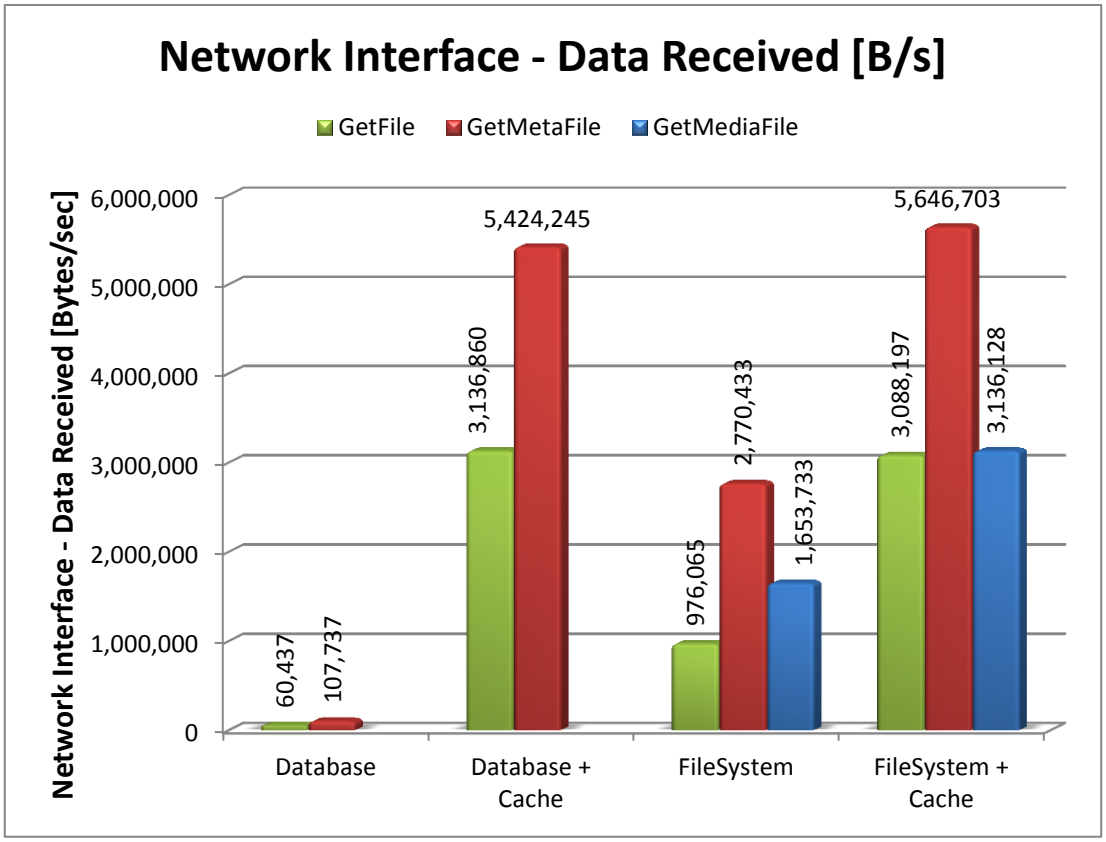

Response Time [ms]
GetFile
GetMetaFile
GetMediaFile
Response Time [ms]
2500
2000
1500
1000
500
0
2131
2106
42
39
130
81
80
44
38
43
Database
Database + Cache
FileSystem
FileSystem + Cache

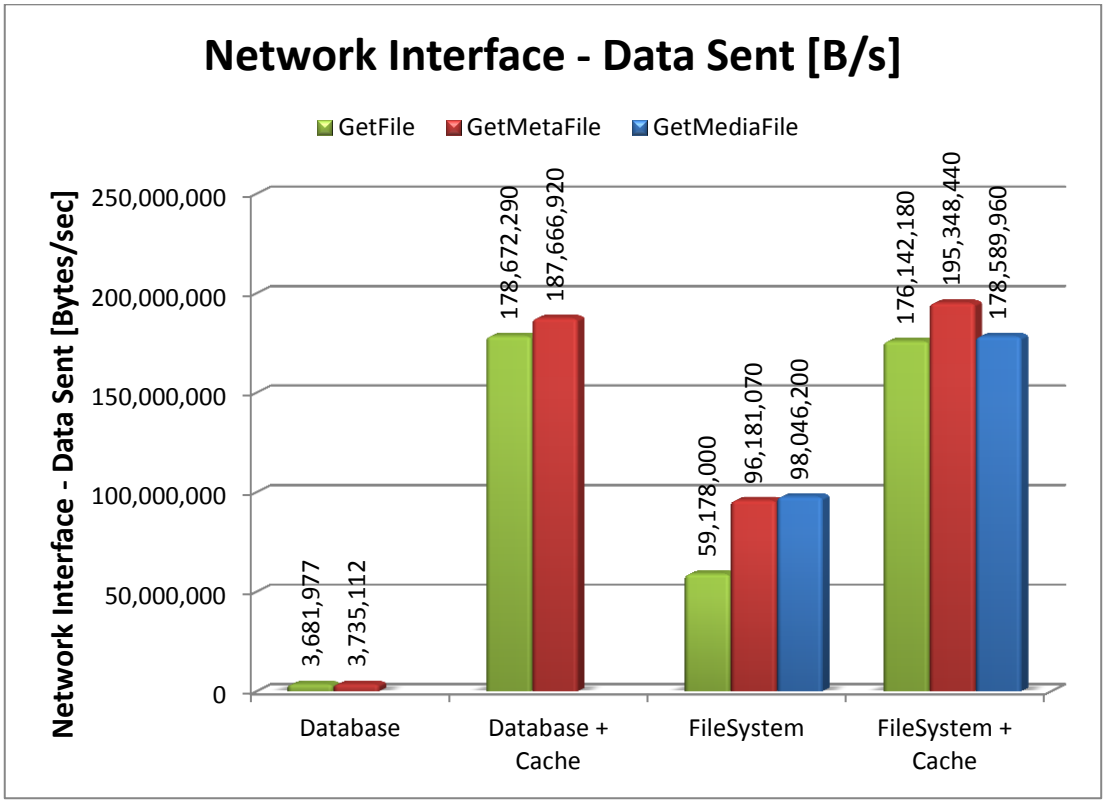

Web Server CPU Utilization [%]
GetFile
GetMetaFile
GetMediaFile
Web Server CPU Utilization [%]
100
90
80
70
60
50
40
30
20
10
0
54
53
94
92
97
94
96
95
94
96
Database
Database + Cache
FileSystem
FileSystem + Cache



Comments: GetFile needs to call the database because of dealing with the related document (not just the file), which leads to higher SQL Server CPU utilization.

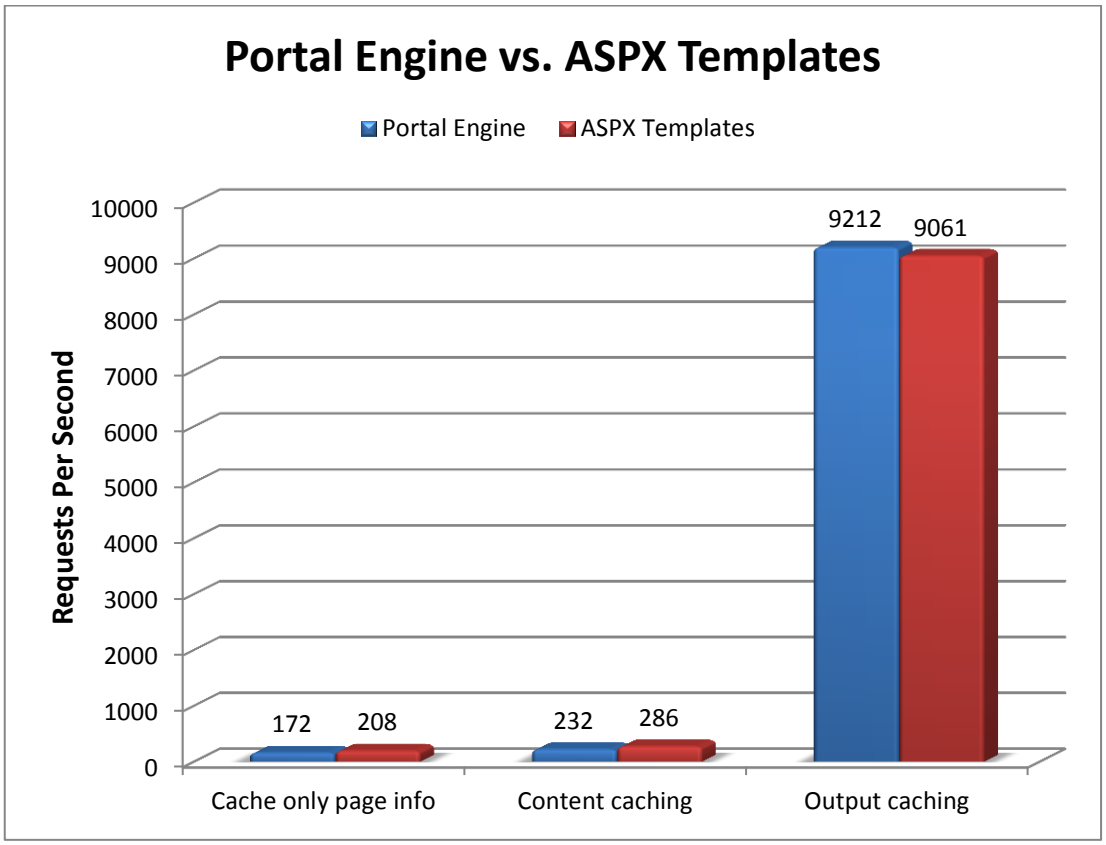



Network Interface - Data Received [B/s]
GetFile
GetMetaFile
GetMediaFile
Network Interface - Data Received [Bytes/sec]
6,000,000
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
60,437
107,737
3,136,860
5,424,245
976,065
2,770,433
1,653,733
3,088,197
5,646,703
3,136,128
Database
Database + Cache
FileSystem
FileSystem + Cache


Network Interface - Data Sent [B/s]
GetFile
GetMetaFile
GetMediaFile
Network Interface - Data Sent [Bytes/sec]
250,000,000
200,000,000
150,000,000
100,000,000
50,000,000
0
3,681,977
3,735,112
178,672,290
187,666,920
59,178,000
96,181,070
98,046,200
176,142,180
195,348,440
178,589,960
Database
Database + Cache
FileSystem
FileSystem + Cache

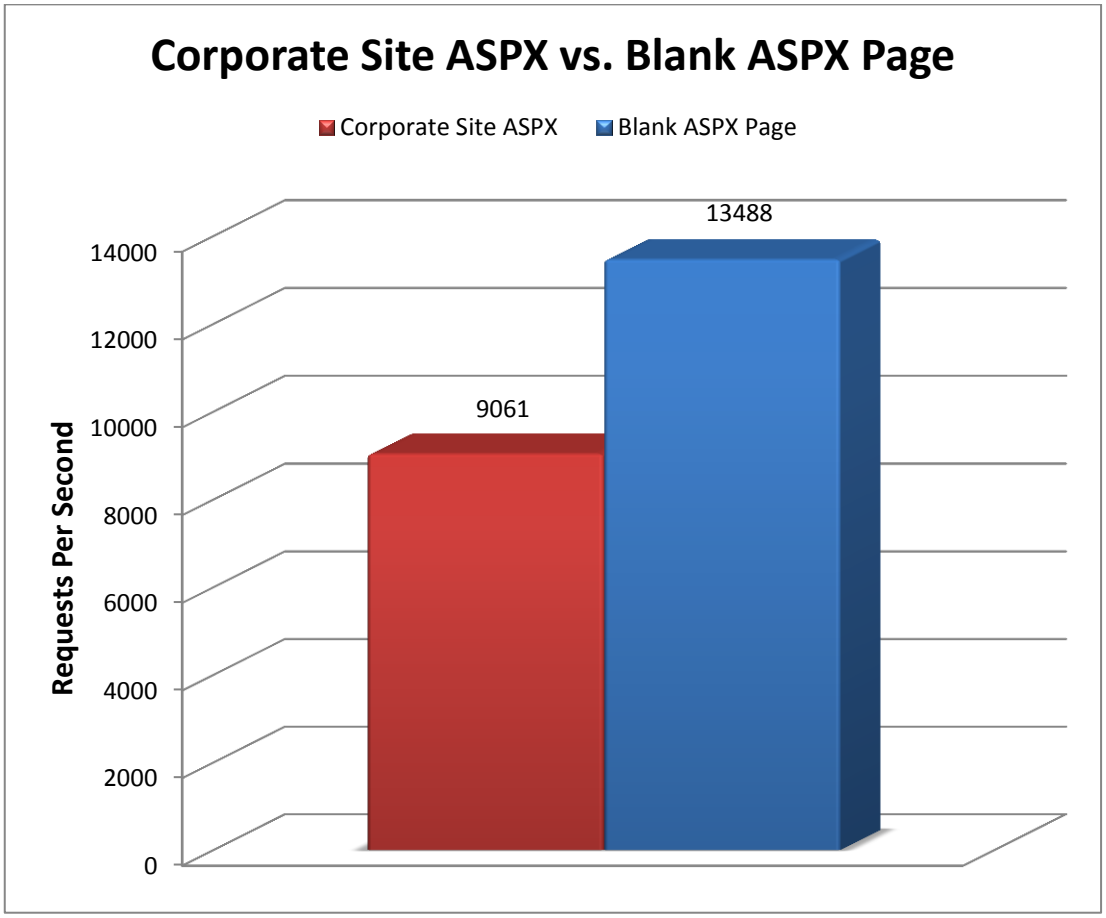
Comparison of Performance for both Development Models

Kentico CMS provides two development models - ASPX templates that are based on standard ASP.NET pages and portal engine that runs on the top of the standard ASP.NET pages and adds an extra, browser-based rapid website development layer.



Comments: As you can see, the ASPX templates provide a better performance since they require less overhead. However, when Output caching is used, there's no difference in performance since the pages are rendered only once. The extra overhead of the portal engine is balanced by easier and faster development with portal engine in comparison to ASPX page templates.

Comparison with a Blank ASPX Page



Comments: Kentico CMS with Output caching provides performance that is not too far from displaying a blank ASPX page which is an excellent result if you realize that it provides dynamic content stored in the database.

When Output Caching Doesn't Help

As you can see from the test results, the use of Output caching may increase the overall performance significantly. It's important to say that in some case the Output caching may not be used:

Content that changes more often than every minute - if you need to display real-time data, you cannot use Output caching.

Personalized content - if you need to display content personalized by the current user, such as user name or if you need to restrict displayed content by current user's permissions, the Output caching cannot be used. Kentico CMS can cache such pages, but since the page will be stored in the memory for every authenticated user, it may consume lots of memory.

You can still use content caching for chosen web parts/controls in such cases, but it provides a significantly lower performance. It's highly recommended that you avoid things like personalized content on the pages that get high traffic.

How to Plan Your Hardware - Server Sizing for Kentico CMS

This chapter will help you plan the configuration of your servers for Kentico CMS so that they can handle the expected load. It will provide you with very rough estimate, but at least, it will give you some guidelines for your decisions.

Please note: if you want to get accurate numbers, the only way is to create the website and run the test on your hardware! There are too many factors that influence the overall performance of your site that it's impossible to calculate the performance upfront, without doing the performance tests.

Also, it's highly recommended that you include performance testing as a part of testing phase of your project and have some reserve for performance optimization. Launching a new site without prior performance testing and optimization often results in bad start and sleepless nights.

Step 1 - Identify the peak load

Identify the peak load, not just the number of visitors per month, because the traffic is not spread equally in time and there may be peaks at specific times. What is the highest number of concurrent visitors on your site now? What number do you expect in the future? What will this number look like if you run a successful advertisement?

Step 2 - Estimate the number of page views per second

Imagine what a typical visitor will do on your site (or better, use some web analytics software to see the current visitor behavior if you already have such site). How many pages does a typical visitor see every minute?

Identify the **pages that are same for all visitors and do not require any personalization per user** and that do not change more often than 1 or 2 minutes. These pages can use Output caching that provides the best performance.

How many pages of this type do you need to serve per second? This number will be called **PVa**.

Identify the **pages that display personalized or frequently changing information**, such as current user's name or content personalized by user's permissions. These pages can use only content caching for chosen web parts or controls.

How many pages of this type do you need to server per second? This number will be called **PVb**.

Step 3 - Calculate the number of web servers and database servers

The number of web servers can then be very roughly calculated as **(PVa + 52*PVb)/9000** if you consider the servers of the same performance level.

The number of database servers highly depends on the caching option you choose. While Output caching requires a single database server only even if you use a large web farm, the content caching requires database servers whose number can be calculated as **PVb/230**.

How to Get More Precise Numbers

As you can see, the numbers are very hypothetical and you still need to consider other factors, such as custom code you wrote, other applications running on the server, ASP.NET start-up and/or compilation time, hardware configuration, network configuration, size of pages and images, speed of communication with client computers, etc. So the only way how to get more real-world numbers is to do the performance tests on the actual website.