

# **KENTICO CMS 5.5**

## **Performance Test Report**

Prepared by Kentico Software in July 2010

## Table of Contents

Disclaimer .....	3
Executive Summary .....	4
Basic Performance and the Impact of Caching .....	4
Database Server Performance .....	6
Web Farm Performance.....	7
Impact of Site Size on Performance.....	8
Cloud Computing .....	9
Server Hardware Configurations .....	10
Testing Configurations .....	11
Caching Configurations.....	12
How the Tests Were Performed.....	13
Performance Test Results.....	14
24-hour Stability Test Results.....	21
Performance Test for Files (File, Media File).....	22
Comparison of Performance for both Development Models .....	27
Comparison with a Blank ASPX Page.....	28
When Full-Page Caching Doesn't Help .....	29
How to Plan Your Hardware - Server Sizing for Kentico CMS .....	30

## 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.

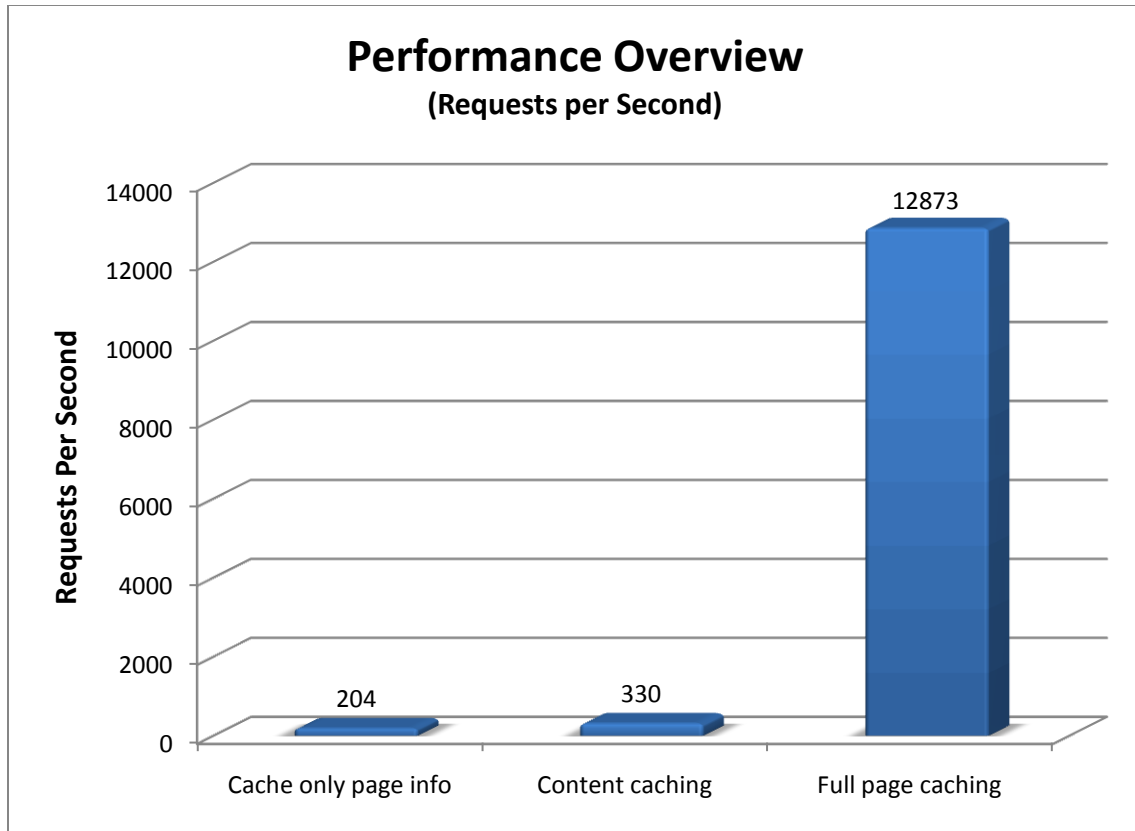
The tests were performed internally by Kentico staff on very common hardware (Intel Core 2 Quad at 2.66 GHz, 10k SATA II disks, 4 GB memory). Kentico has not used any high performance servers, so the results may be even better on more powerful hardware.

## 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 **full-page 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.

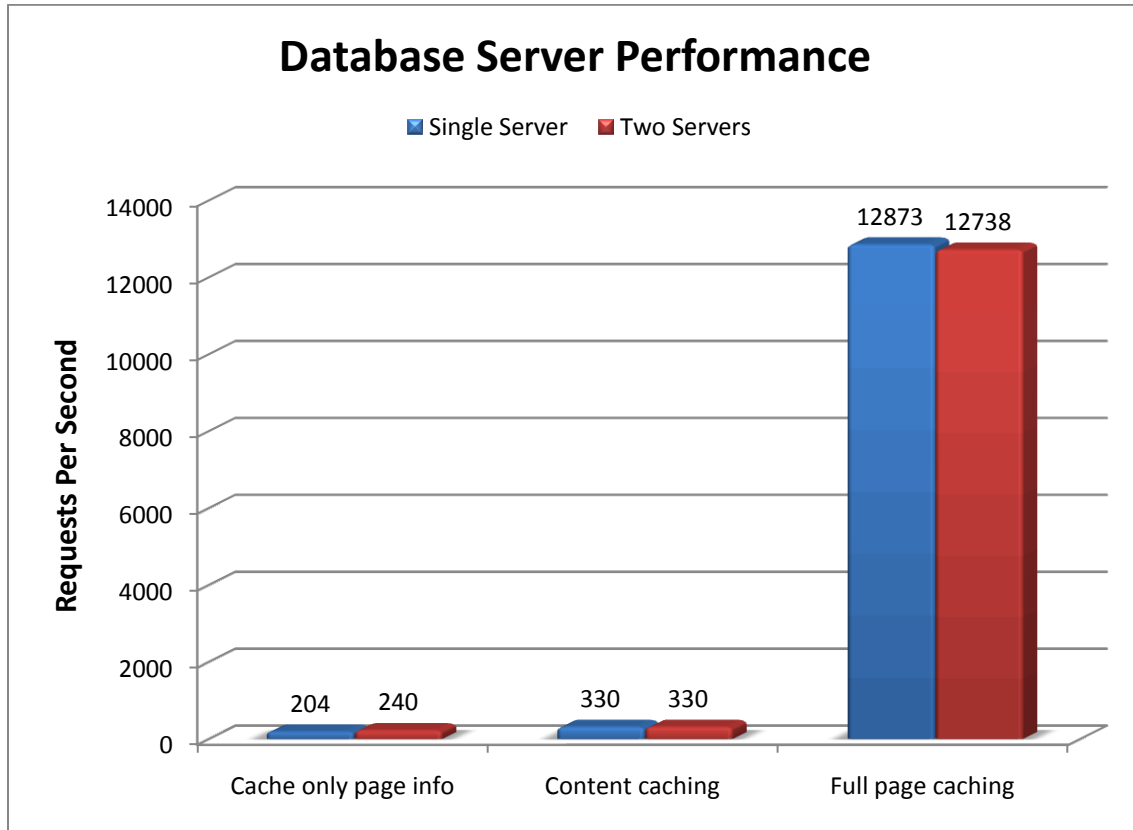


**Comments:** As you can see, Kentico CMS provides a highly efficient caching mechanism that boosts the performance significantly.

## Database Server Performance

If you need to achieve 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):



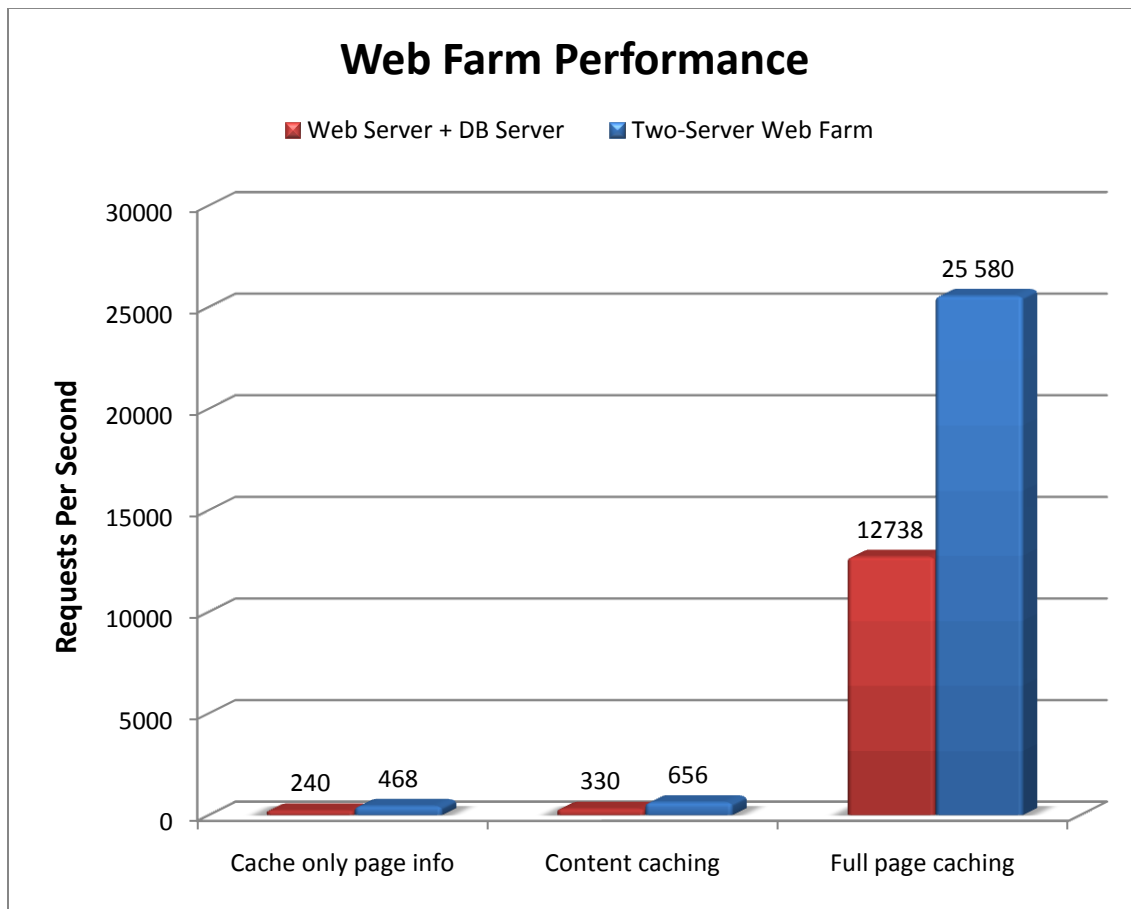
**Comments:** As you can see, the performance gain from a dedicated database server is most significant when the caching is not used.

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):

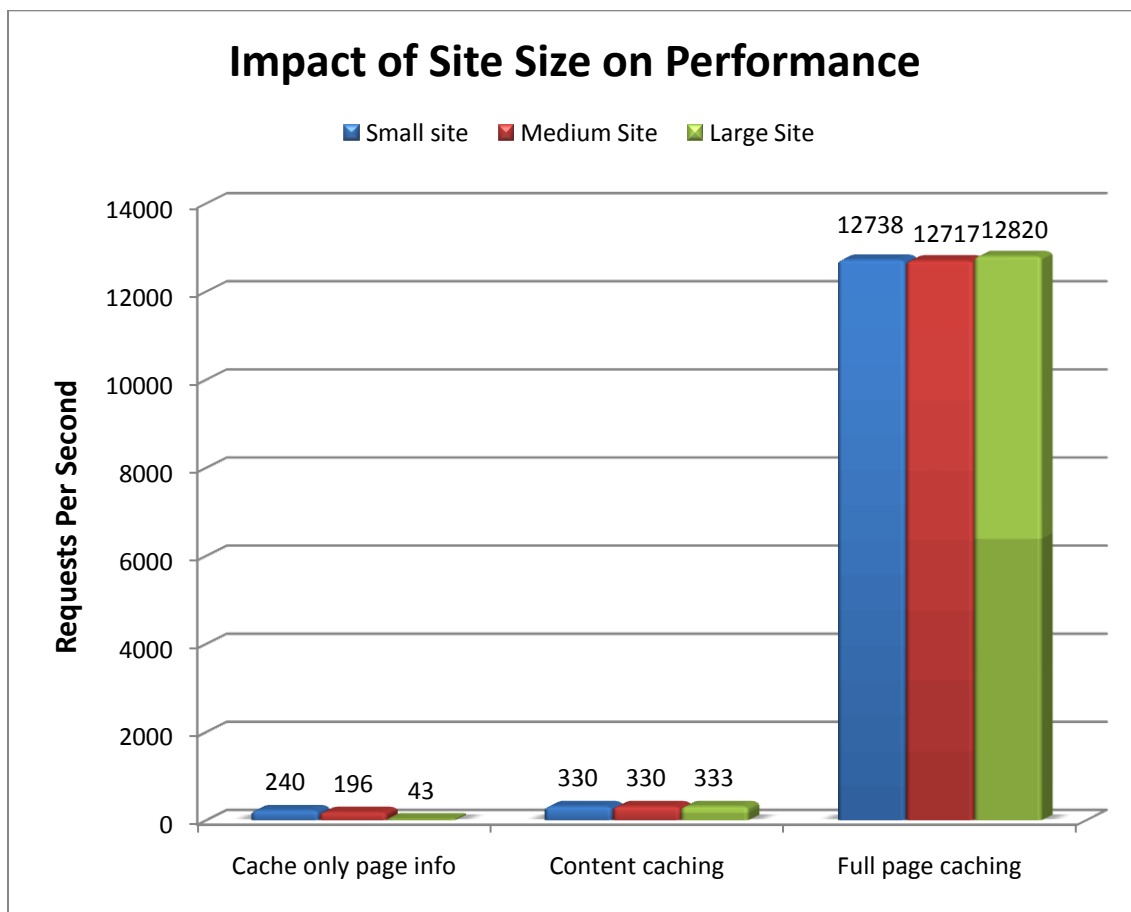


**Comments:** The results show that Kentico CMS provides excellent scalability. With an additional web server, the performance grows by 100%. The result is only slightly lower for configuration without caching since this configuration highly uses database server that was shared between both web servers.

## 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 (284 documents), medium site (10,000 documents) and a large site (100,000 documents), both running on a single web server with a dedicated SQL Server:



Comments: the performance without any caching drops significantly with a very large number of pages. This scenario requires site-specific optimization of SQL queries and database indexes. The performance with caching is practically same regardless of the site size.

## Cloud Computing

Kentico CMS can also be used with cloud computing platforms that provide both high performance and availability. Kentico CMS can work with Rackspace cloud platform (Mosso) and Amazon EC2. Support for Microsoft Azure is planned for Q1/2011. Performance tests were not performed on these platforms.

## 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)

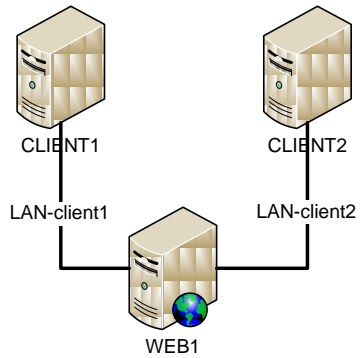
Capacity	16 GB (Kit 4x 4 GB )
Type	DDR2-SDRAM
Memory Bus	1066 MHz

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

## Testing Configurations

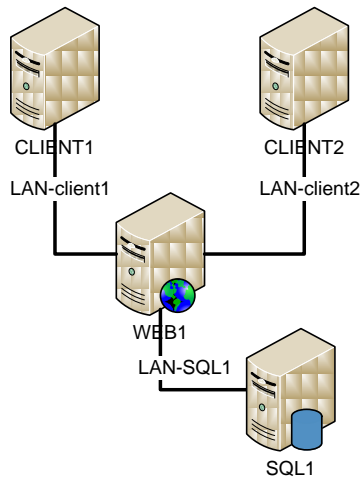
### Configuration A - Single shared server (web + database)

*Windows 2003 R2 Service Pack 2, Microsoft SQL Server 2005*



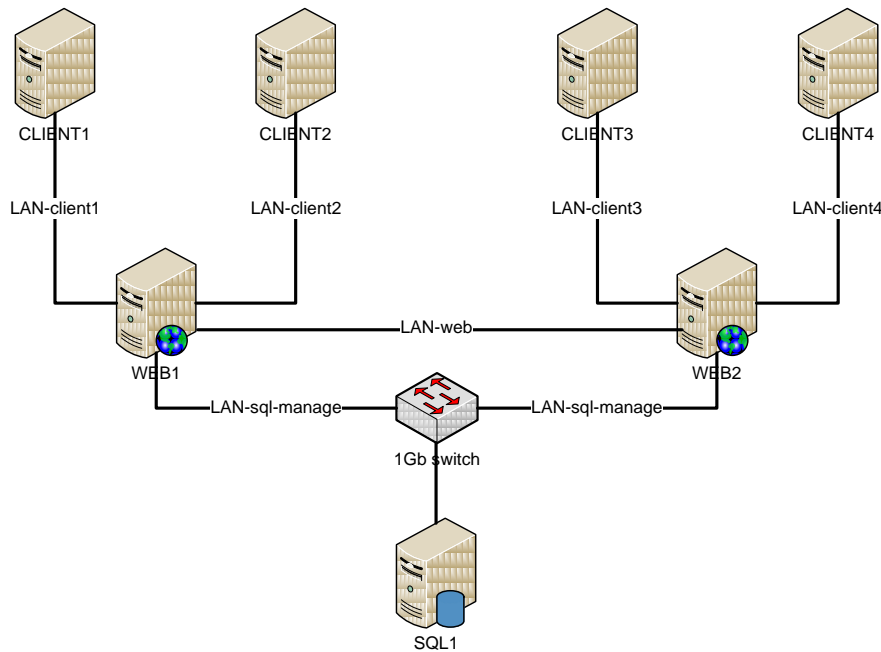
### Configuration B - Two separate servers (a web server and a database server)

*Windows 2003 R2 Service Pack 2, Microsoft SQL Server 2005*



## Configuration C – Two servers in a web farm and a database server

Windows 2003 R2 Service Pack 2, Microsoft SQL Server 2005



## 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<sup>1</sup>
- **Cache 3 - Full-page caching:** full-page caching (whole page cached in memory) enabled for 10 minutes

---

<sup>1</sup> 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 10 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
- HTTP, DNS, Socket, Windows or Kentico CMS errors were NOT allowed during tests
- Kentico CMS restart was NOT allowed during one test run
- Before each test:
  - Restart IIS and delete .NET temporary files
- Kentico CMS used GZip compression
  - Following settings key was added to the web.config section <appSettings>:  
`<add key="CMSAllowGZip" value="true" />`
- Network traffic was measured on the server side and it was the sum of all interfaces

## 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 Exceptions\# Exceps thrown / sec
- ASP.NET\Application Restarts
- ASP.NET\Requests Rejected
- ASP.NET\Worker Process Restarts (not applicable to IIS 6.0)
- 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

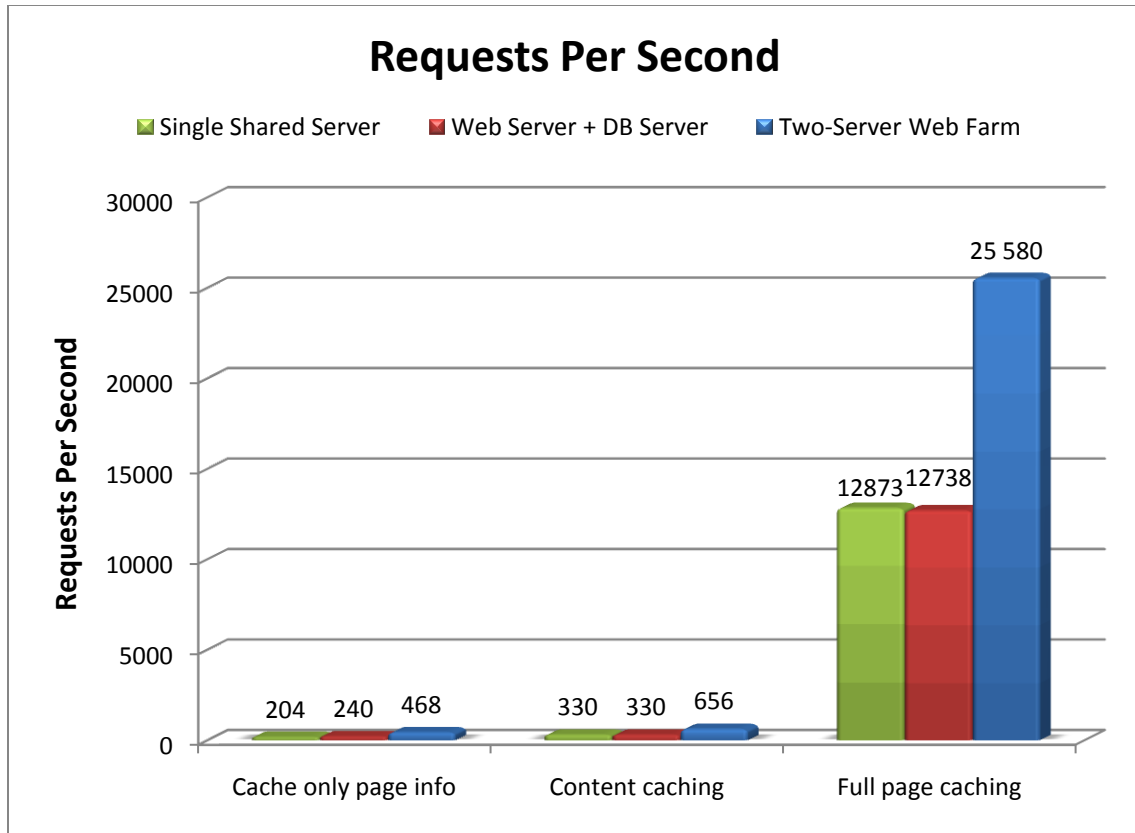
## Performance Test Results

The tests were based on visiting the following pages:

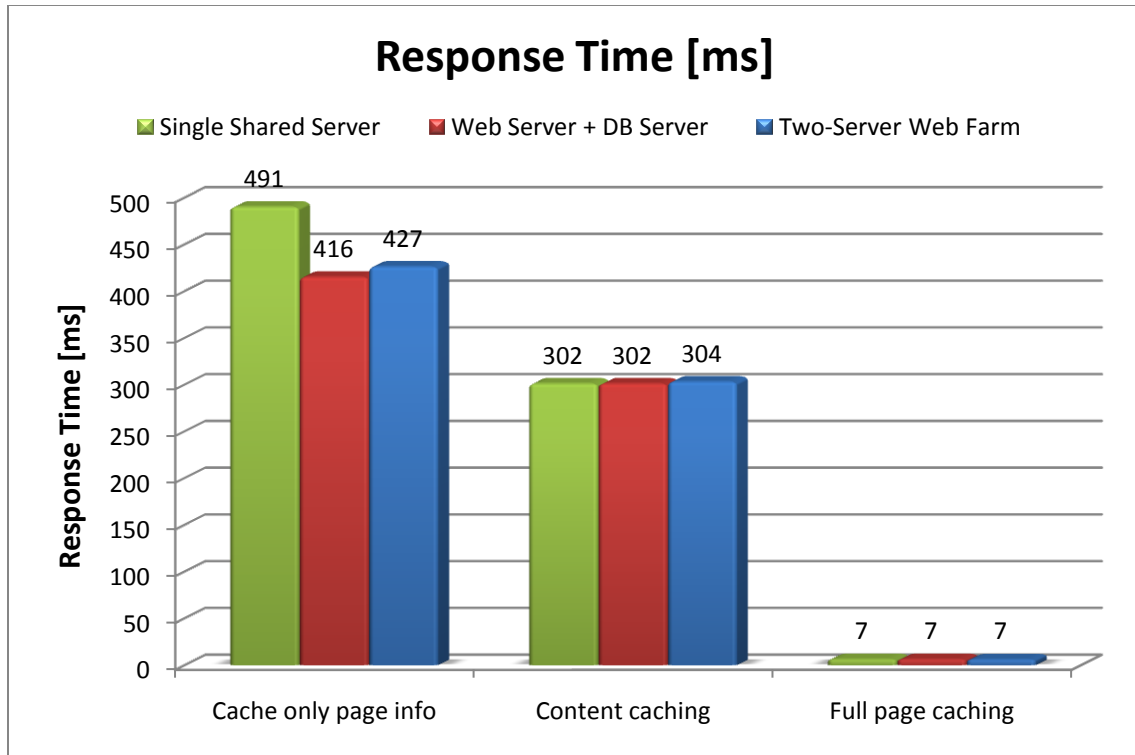
### *Sample Corporate site*

- ~/Home.aspx
- ~/Services.aspx
- ~/Network-administration.aspx
- ~/Products.aspx
- ~/Products/PDAs/HP-iPAQ-114.aspx

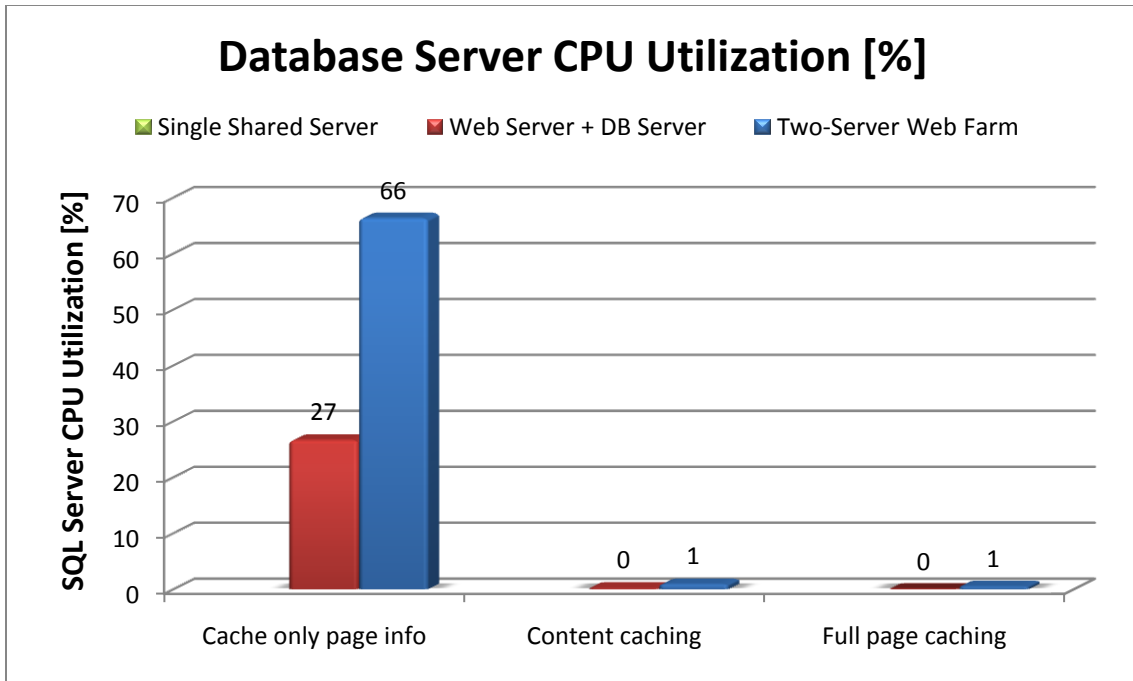
	AVG requests per second	AVG requests per hour	AVG requests per 24 hours
<b>Single shared server</b>			
Cache only page info	204	650 749	17 353 311
Cache content	330	1 053 376	28 090 023
Output cache	12 873	45 654 546	1 217 454 563
<b>Two separate servers</b>			
Cache only page info	240	751 226	20 032 697
Cache content	330	1 030 066	27 468 420
Output cache	12 738	45 960 275	1 225 607 342
<b>Two Servers in a Web Farm + DB Server</b>			
Cache only page info	468	1 502 452	40 065 394
Cache content	656	2 060 132	54 936 840
Output cache	25 580	91 920 551	2 451 214 684



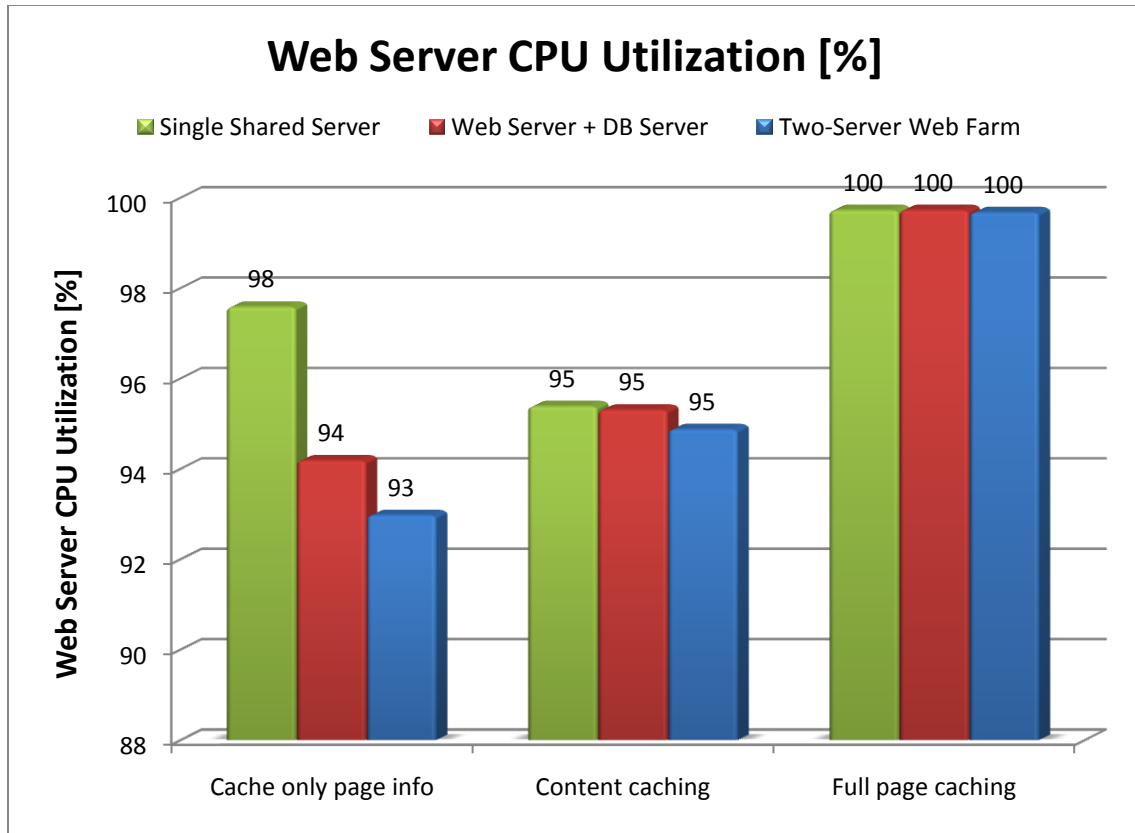
**Comments:** Here you can see that full page caching provides the best possible performance. The graph also shows that using a web farm multiplies the overall performance by the number of web servers.



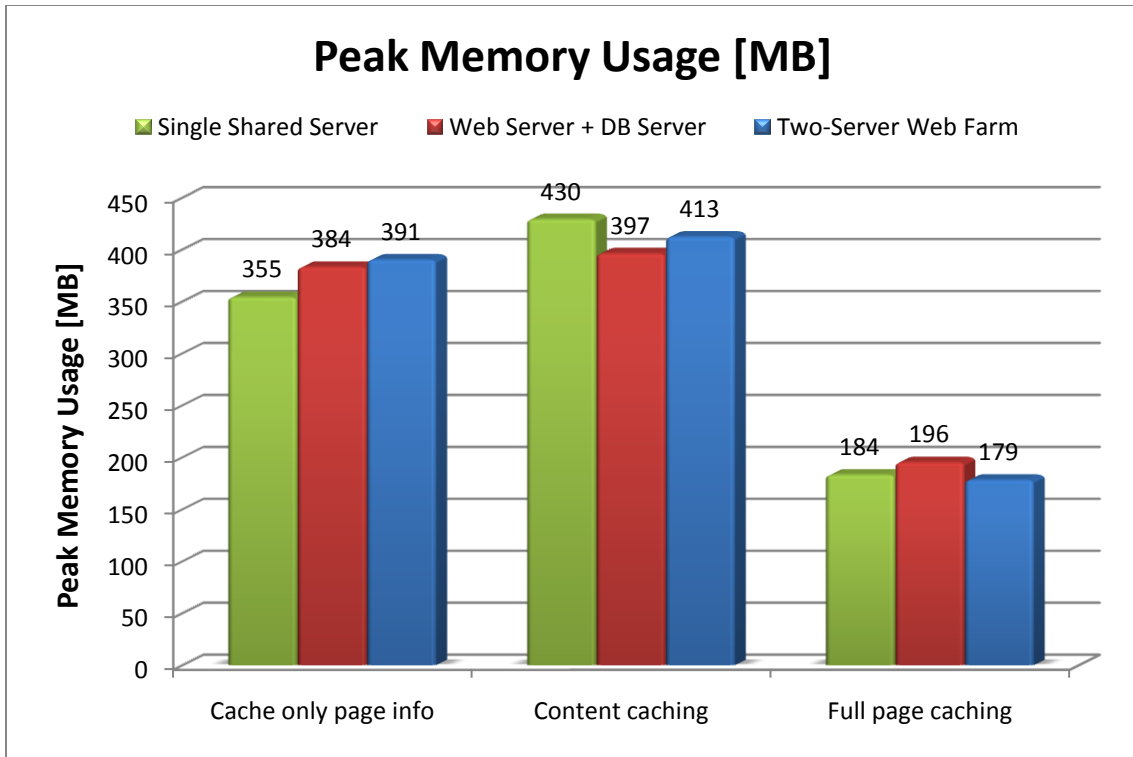
**Comments:** The response time highly depends on the amount of cycles that need to be done to render the requested page. The full-page caching only takes the HTML code cached in the memory and sends it to the browser, so it's extremely fast.



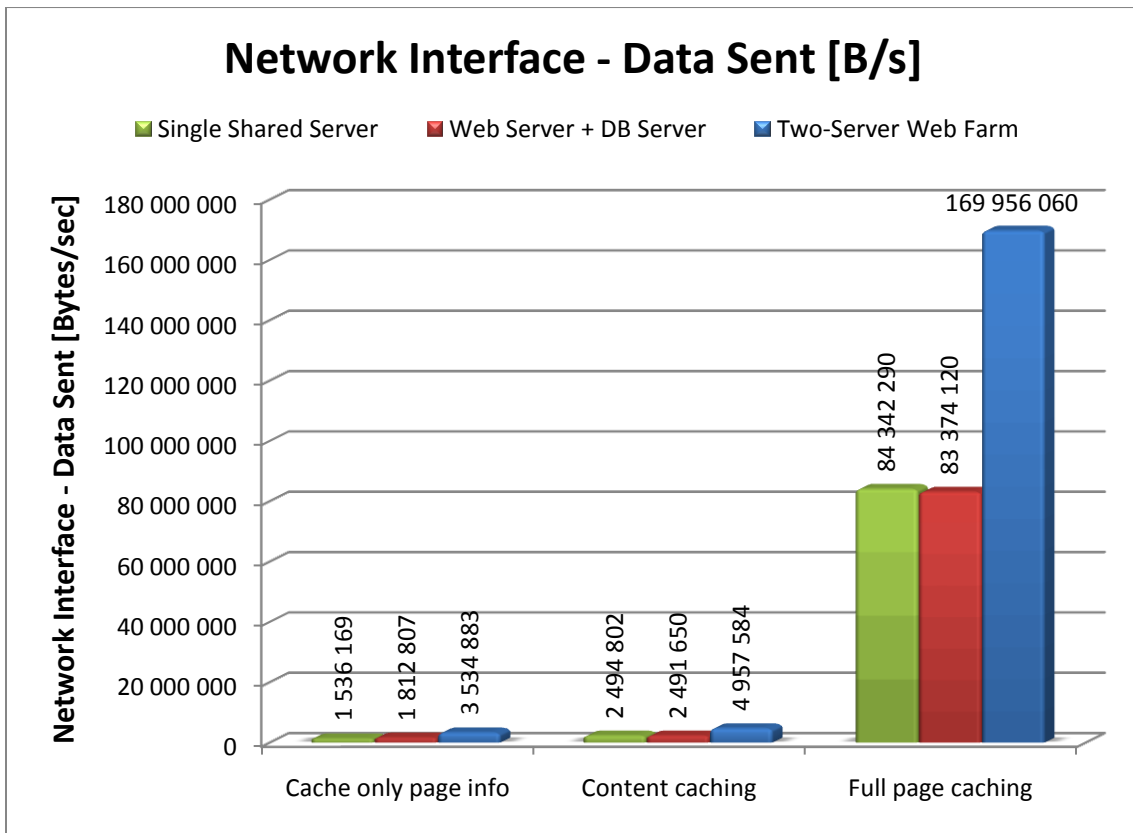
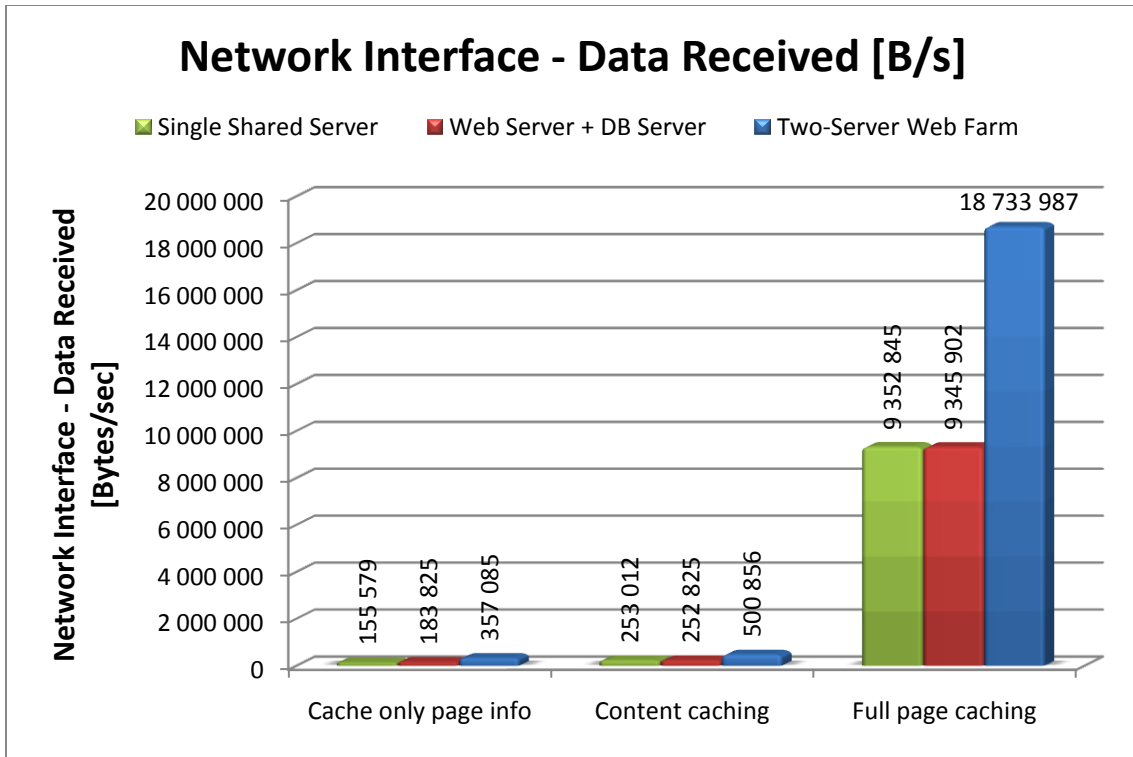
**Comments:** The graph shows that once caching is used, the database server utilization is very low since the website only accesses the database when the content is requested for the first time. The results for Single Shared Server were not measured since the database server shared the CPU with the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



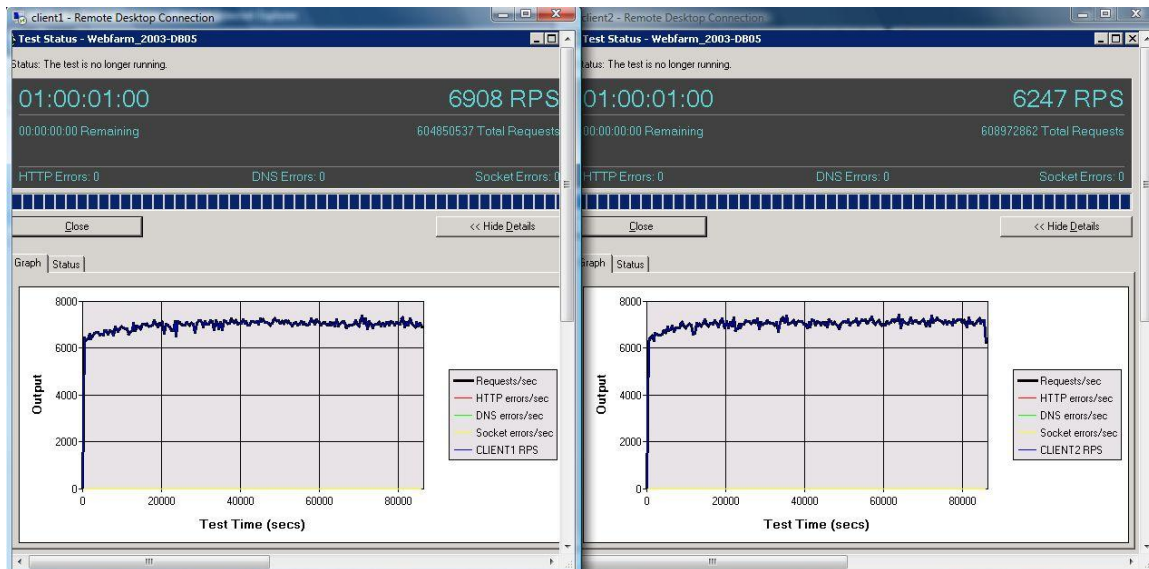
**Comments:** The graph shows the utilization of the processor by all processes running on the server. The Single Shared Server configuration includes both the web server process and SQL Server process. Since we tested for the highest performance, the utilization tends to reach 100%. The only exception 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:** Our tests required no restarts of the web server process during the 24-hour test which also means there were no memory leaks that would crash the process. The full-page 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 full-page caching may, however, look different if you have a large site with many pages that would be stored in the memory.



## 24-hour Stability Test Results



Comments: The pictures show the Requests per Second (RPS) value during the 24-hour stability test. The test was performed on the Two-server web farm with a dedicated database server and with full-page caching. There are two pictures, because we used two client computers to generate the load test traffic for one web 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. All these tests were also performed on a single shared server with expected results.

## 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)

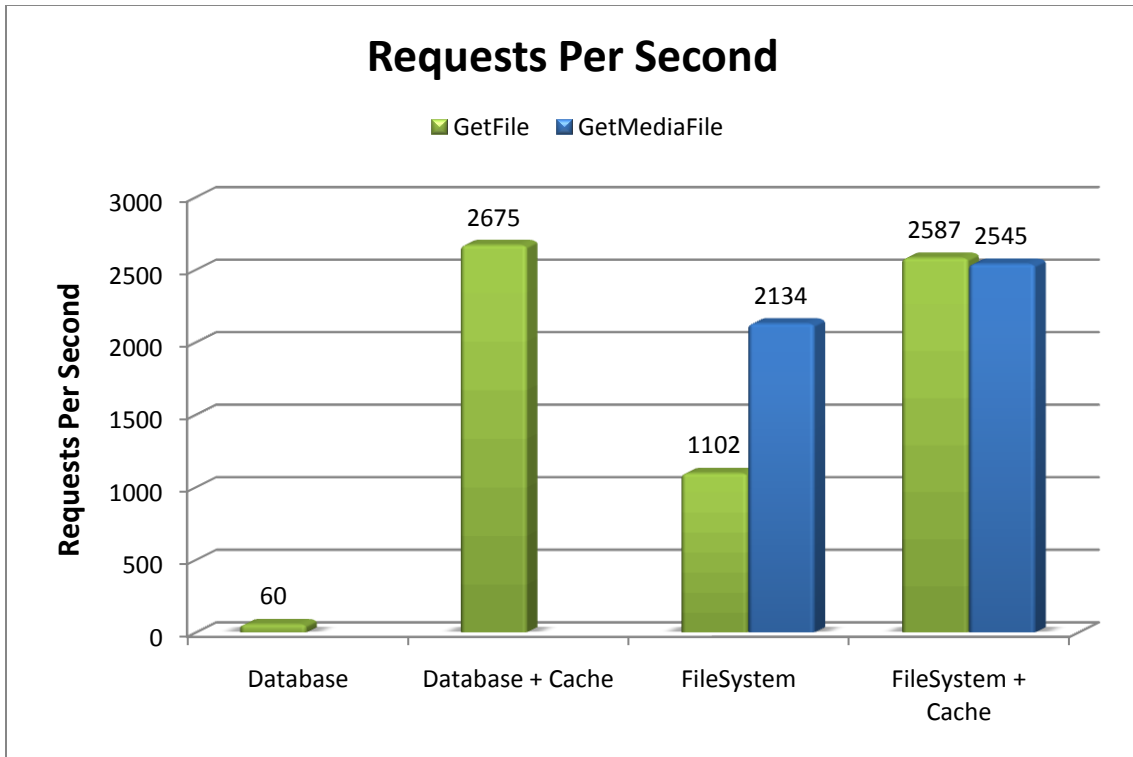
**GetFile - serves all files uploaded into the content repository:**

- ~/getfile/7ad6d7cd-61df-46e1-9009-9430e69e62ff/Microsoft.aspx
- ~/ getfile/7ad6d7cd-61df-46e1-9009-9430e69e62ff/Microsoft.aspx?width=320
- ~/ getfile/7ad6d7cd-61df-46e1-9009-9430e69e62ff/Microsoft.aspx?width=640

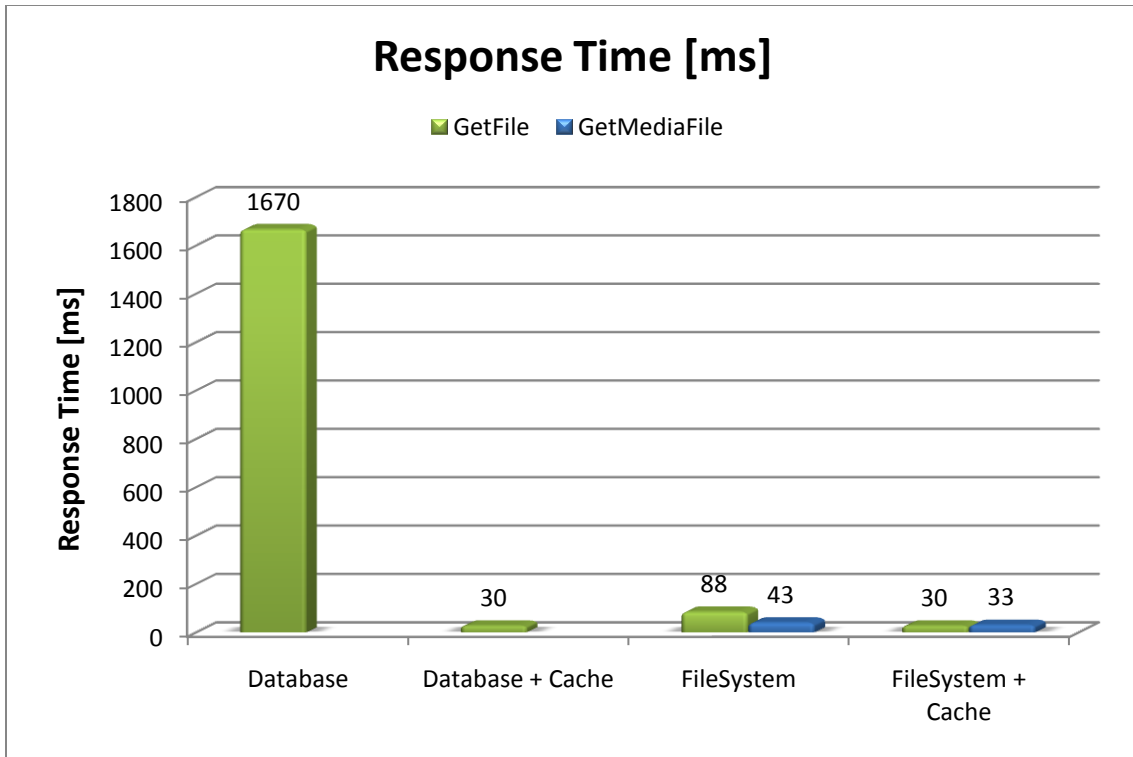
**GetMediaFile - serves all files stored in the media library:**

- ~/ getmedia/8af6e551-5ad9-462b-9e8e-4046004a60ff/Microsoft.aspx
- ~/ getmedia/8af6e551-5ad9-462b-9e8e-4046004a60ff/Microsoft.aspx?width=320
- ~/ getmedia/8af6e551-5ad9-462b-9e8e-4046004a60ff/Microsoft.aspx?width=640

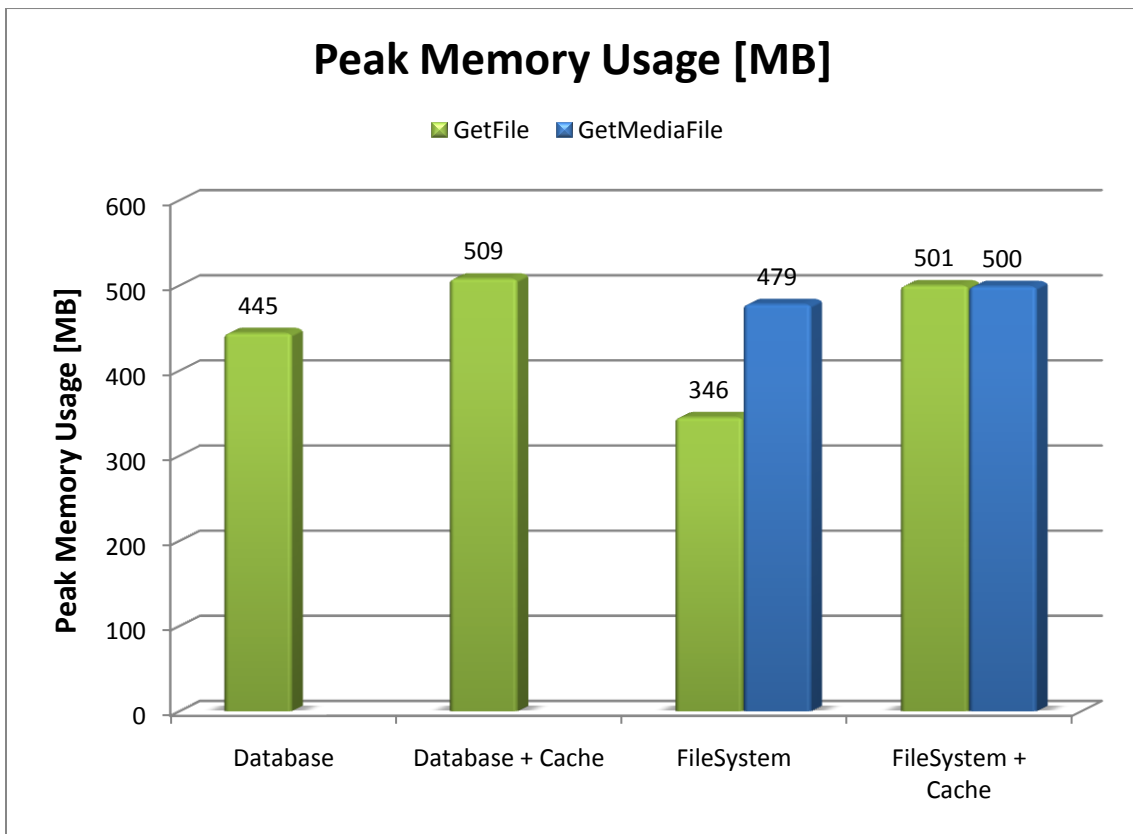
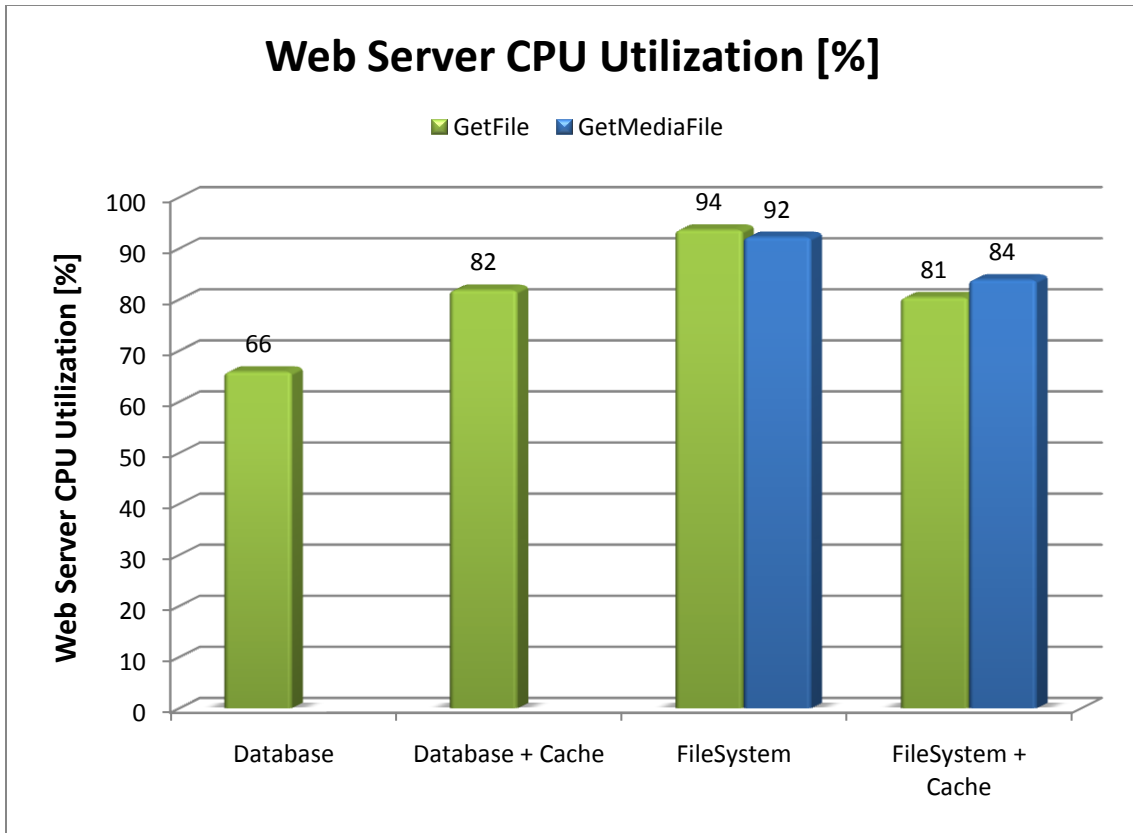
	AVG requests per second	AVG requests per hour	AVG requests per 24 hours
<b>GetFile</b>			
Database	60	194 962	5 198 975
Database + Cache	2675	7 197 870	191 943 203
File System	1102	3 291 368	87 769 812
File System + Cache	2587	7 197 870	191 943 203
<b>GetMediaFile</b>			
File System	2 134	5 389 104	143 709 430
File System + Cache	2 545	6 936 683	184 978 207



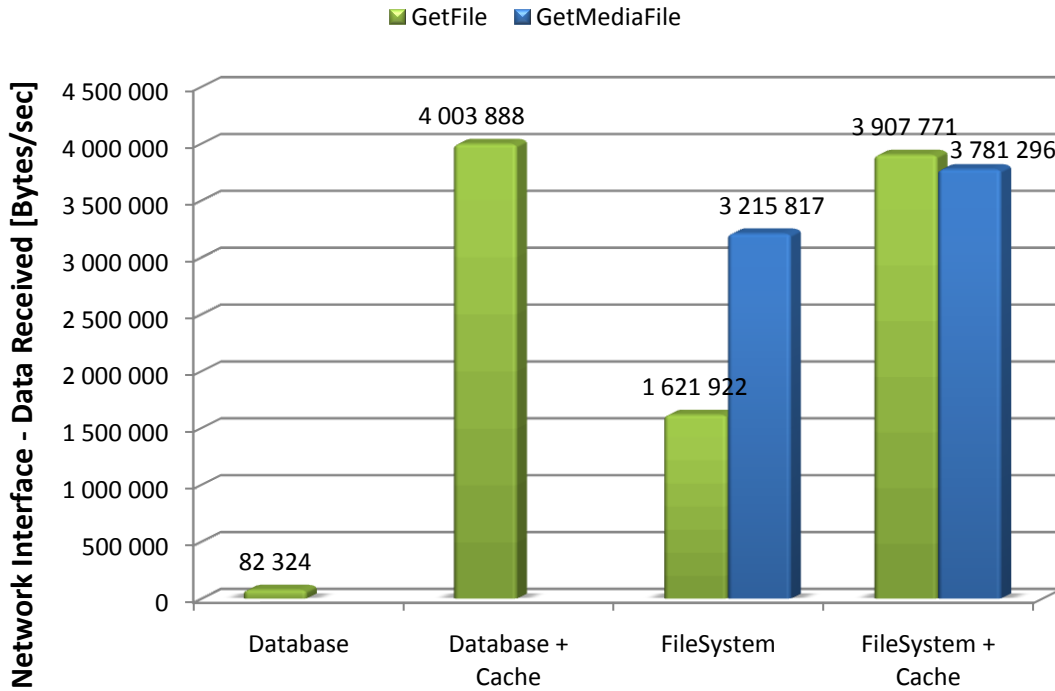
**Comments:** GetFile is generally slower because of dealing with the related document (not just the file). If you plan to work with many media files or with large files, such as video or music, it's recommended that you use the Media Library instead of storing these files in the content tree.



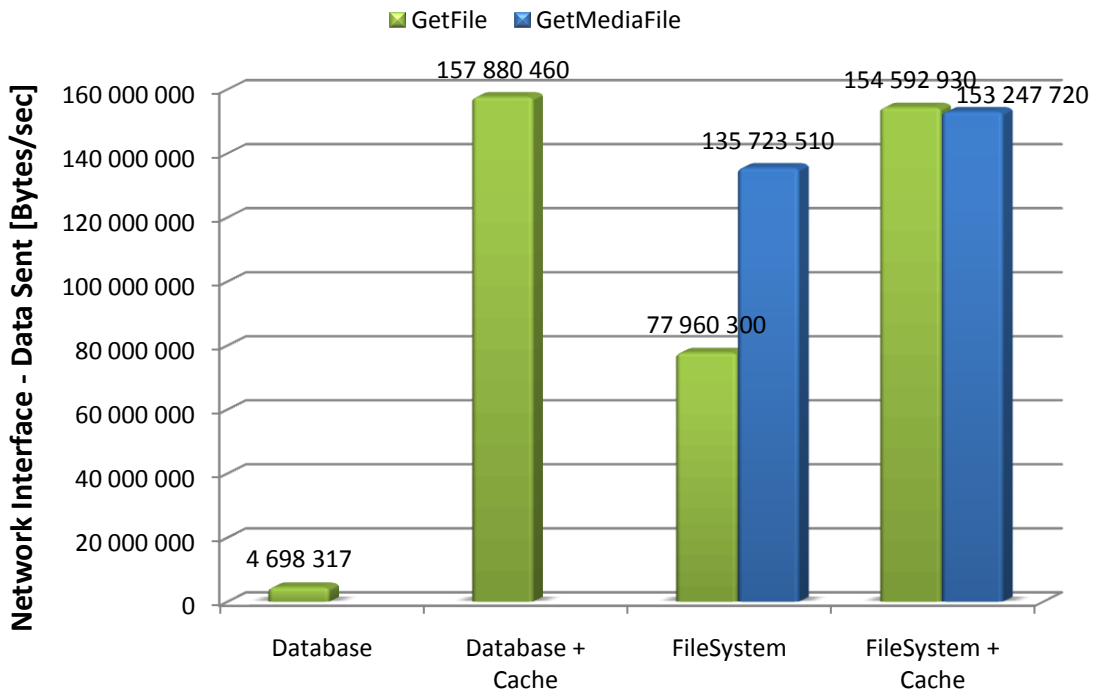
**Comments:** GetFile needs to call the database because of dealing with the related document (not just the file), which leads to higher SQL Server CPU utilization.



## Network Interface - Data Received [B/s]

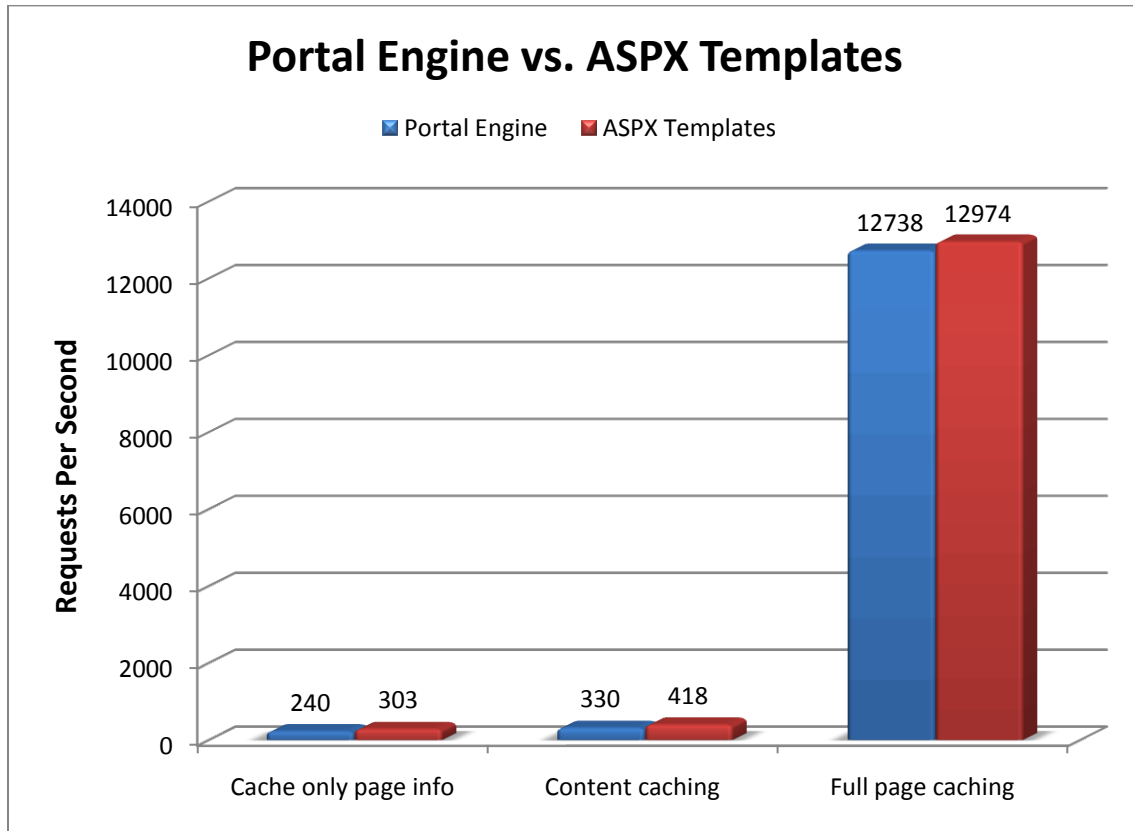


## Network Interface - Data Sent [B/s]



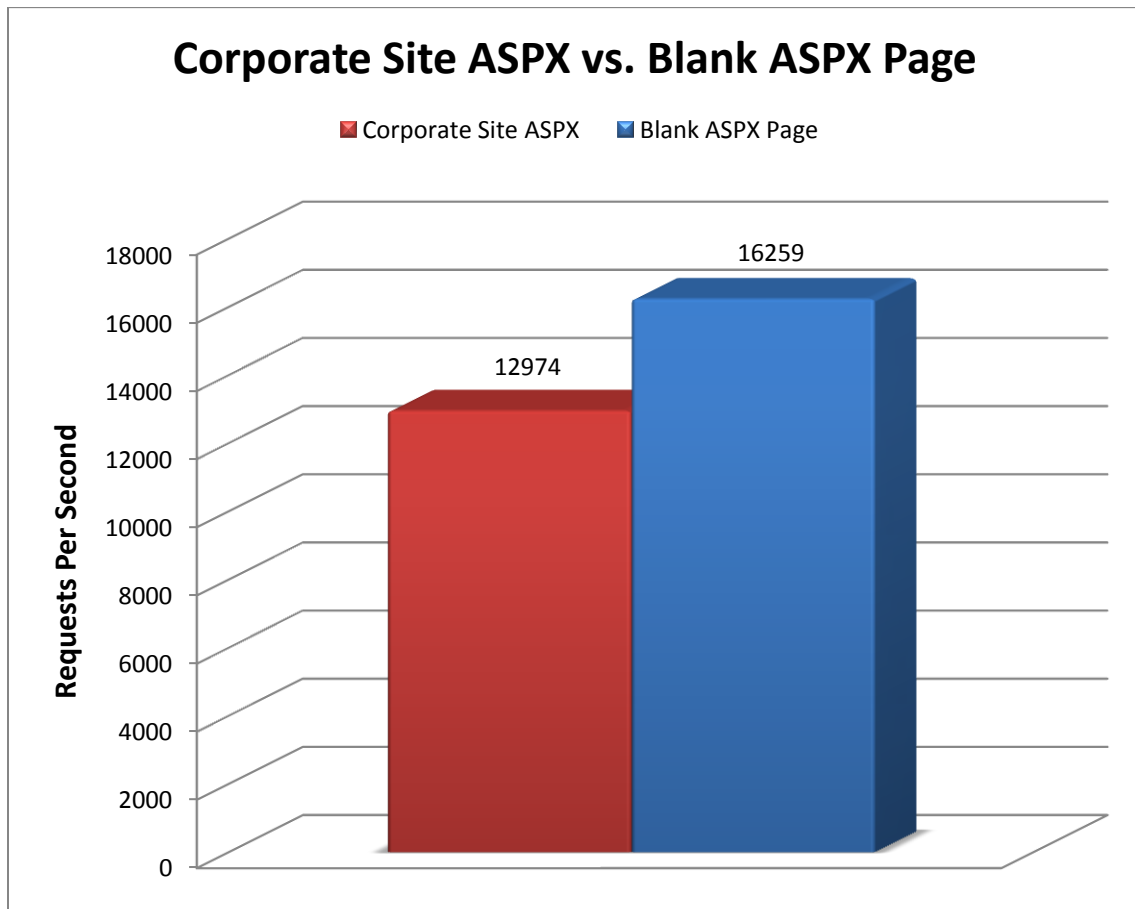
## Comparison of Performance for both Development Models

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 full-page 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 full-page caching provides performance that is not too far to displaying a blank ASPX page which is an excellent result if you realize that it provides dynamic content stored in the database.

## When Full-Page Caching Doesn't Help

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

1. **Content that changes more often than every minute** - if you need to display real-time data, you cannot use full-page caching.
2. **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 full-page 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 are displayed very often.

## 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 full-page 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 4 - Calculate the number of web servers and database servers

The number of web servers can then be very roughly calculated as  $(PVa + 52*PVb)/13000$  if you consider the servers of the same performance level.

The number of database servers highly depends on the caching option you choose. While full-page 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  $PVb/300$ .

#### 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.