Kentico 10

Performance

February 2017

www.kentico.com

Table of Contents

Disclamer

This report was conducted by Kentico Software with the intention of informing customers about performance in Kentico 10. Kentico has done its best to ensure an unbiased test.

However website performance depends on many things, such as computer hardware, network configuration, client configuration, operating system and software configuration, site content, number of items in the Kentico database, information architecture, custom code, and other factors.

Kentico Software cannot, therefore, guarantee that the same results will be achieved with any other than the tested configurations. The reader of this report uses all information contained within at their own risk. Kentico Software shall in no case be liable for any loss resulting from the use of this report.

Executive summary

Kentico 10 provides excellent performance and scalability. Being built on the Microsoft ASP.NET platform, it leverages all its power. Tests were performed internally by Kentico staff on virtual machines (4 virtual processors at 2.39 GHz, SSD disk with 4000 IOPS, 16 GB memory).

Kentico has not used any high performance servers, so results may be even better on more powerful hardware.

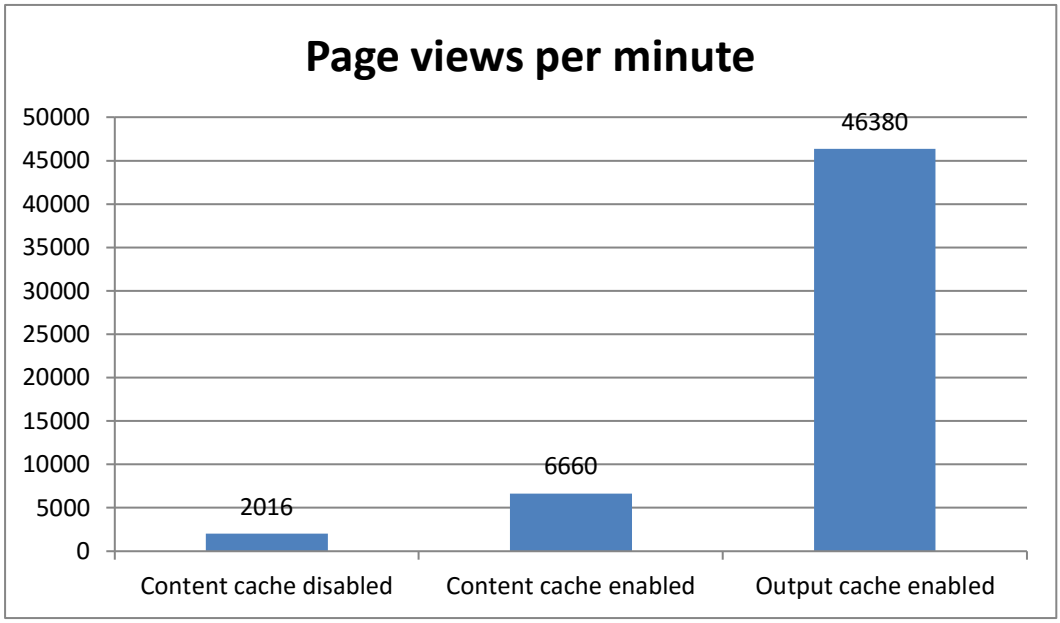
Basic performance and the impact of caching

The slowest parts of a web application are typically accessing the database and rendering the content for a web browser. Kentico 10 optimizes performance by storing content that is frequently 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 10 can quickly send it to the browser without repeatedly accessing the database and rendering the page.

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

The following graph shows the impact of caching on the overall performance. The values represent the number of pages viewed by virtual users per minute.



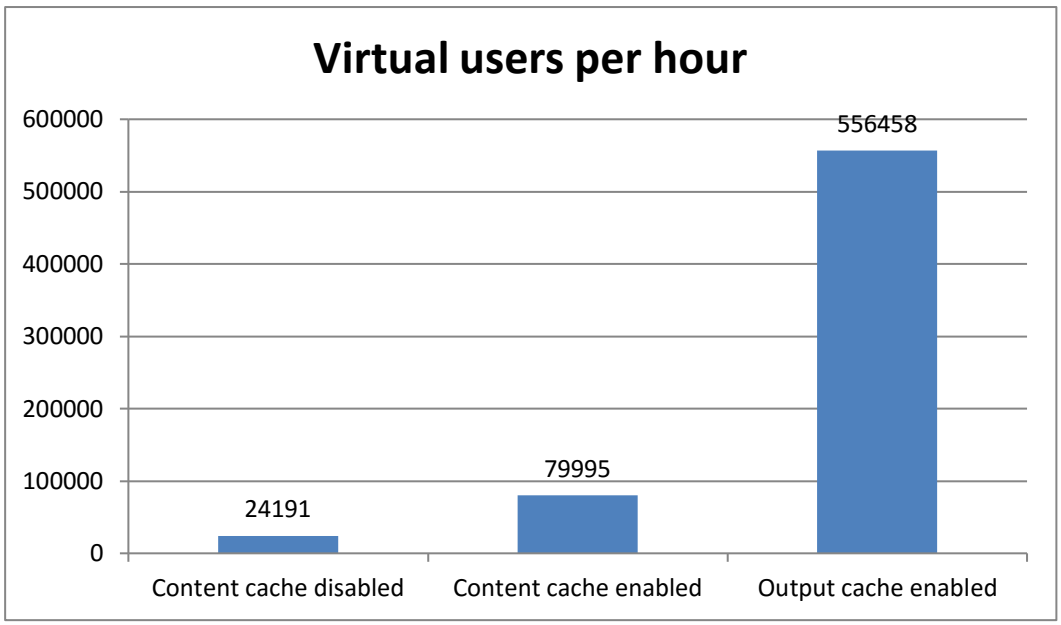
A total of 2016 page views per minute with content cache disabled means about 120 thousand page views per hour, which means almost 3 million page views per day. That's about **87 million page views per month**.

If you enable the content cache, you can handle more than 3 times more traffic, i.e. almost **288 million page views per month**.

In this scenario, with the output cache enabled, Kentico 10 can handle **2 billion page views per month**.

Of course, this depends heavily on the content of your website, the features you use, and the hardware you have available.

To improve performance, you can use CDN (Content Delivery Network) services to serve static files like images and relieve the web server of client requests. You should also use client caching which enables the user's browser to cache static resources so it doesn't have to send requests for these files to the web server at all.



As you can see in the graph above, during our **one-hour test** with the output cache enabled, more than **556,000 unique visitors** each visited 5 pages of our Sample Corporate site News section, which meant serving more than **2.7 million page views per hour**.

Database server separation

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

Web farm usage

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

Impact of site size on performance

Website performance also depends on the number of pages and other items in the Kentico database. Kentico 10 was optimized for a high number of items and was tested with **100,000 pages** stored in the database.

Not only can public-facing websites handle this number of items without negative effect on usability or user interface responsiveness, but so can the Kentico 10 **administration interface**.

Of course, Kentico must be properly set up and optimized for such scenarios. As each project is unique, this could require some customization and optimization of the SQL queries and database indexes. Performance of the live site with caching enabled is practically the same regardless of site size, except for the first page load.

Cloud computing

Kentico 10 can also be used with cloud computing platforms, which provide both high performance and availability. Kentico 10 natively supports Microsoft Azure (including Microsoft Azure Storage) and Amazon EC2 (including Amazon S3 Storage), and we have numerous clients who use Kentico on these platforms. Our customers also use other cloud platforms, such as Rackspace Cloud.

Kentico 10 can be run on multiple instances in both Microsoft 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.

How tests were conducted

Hardware configuration

As you can see in the following image, we used 4 computers dedicated to performance testing. Client1 and Client2 were the computers that generated the load to the web server, called Web1. Sql1 was the database server for storing databases used by Web1.

Client1

Processor	4 virtual processors at 2.39 GHz
Memory	3.81 GB
SSD	160 GB, 4000 IOPS
LAN	1 Gbit/s

Client2

Processor	4 virtual processors at 2.39 GHz
Memory	2 GB
SSD	120 GB
LAN	1 Gbit/s

Web1

Processor	4 virtual processors at 2.39 GHz
Memory	16 GB
SSD	160 GB, 4000 IOPS
LAN	1 Gbit/s

SQL1

Processor	4 virtual processors at 2.39 GHz
Memory	16 GB
SSD	160 GB, 4000 IOPS
LAN	1 Gbit/s

System configuration

- Microsoft Windows Server 2016
- Microsoft Internet Information Services 10
- Microsoft SQL Server 2016
- Microsoft Visual Studio 2013 Ultimate

Kentico 10 settings

Configuration for all tests

- Enable output compression: **True**
- Use progressive caching: **True**
- Client cache (minutes): **100**
- Allow JavaScript minification: **True**
- Allow CSS minification: **True**
- Disable debugging: **True**
- Enable on-line marketing: **False**
- Enable A/B testing: **False**
- Enable multivariate testing: **False**
- Enable content personalization: **False**
- Enable web analytics: **False**

Caching configurations

- **Content cache disabled**: Page info (page data and metadata) cache enabled for 100 minutes, content caching set for 0 minutes
- **Content cache enabled**: Content cache (web part/control-level caching) enabled for 100 minutes[1]
- **Output cache enabled**: Output cache (whole page cached in memory) enabled for 100 minutes

Load test settings

Tests were based on visits to the News section of our Sample Corporate site. There were more than 100,000 pages in the database to demonstrate the power of caching.

- All tests were performed for a duration of 1 hour (+ 1 minute warm-up time).
- Load Tests were executed in Visual Studio 2013 (Ultimate Edition).
- The constant load of 50 simultaneous virtual users was used (100% new users).
- When a virtual user finished visiting their 5 page requests, a new virtual user was created right away.
- There was no "Think Time" between requests.
- Dependent requests were also parsed.
- No HTTP, DNS, Socket, Windows or Kentico 10 errors were allowed during tests.
- Kentico application restarts were not allowed during tests.
- IIS was restarted before every test execution and the application was warmed up with a few requests.

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

Kentico 10 – Sample Corporate site

~/News.aspx
~/News/Apple-iPad-2-In-Stock.aspx
~/News/Community-Website-Section.aspx
~/News/Company-Growth-Exceeds-Expectations.aspx
~/News/New-Consulting-Services.aspx

Performance counters watched

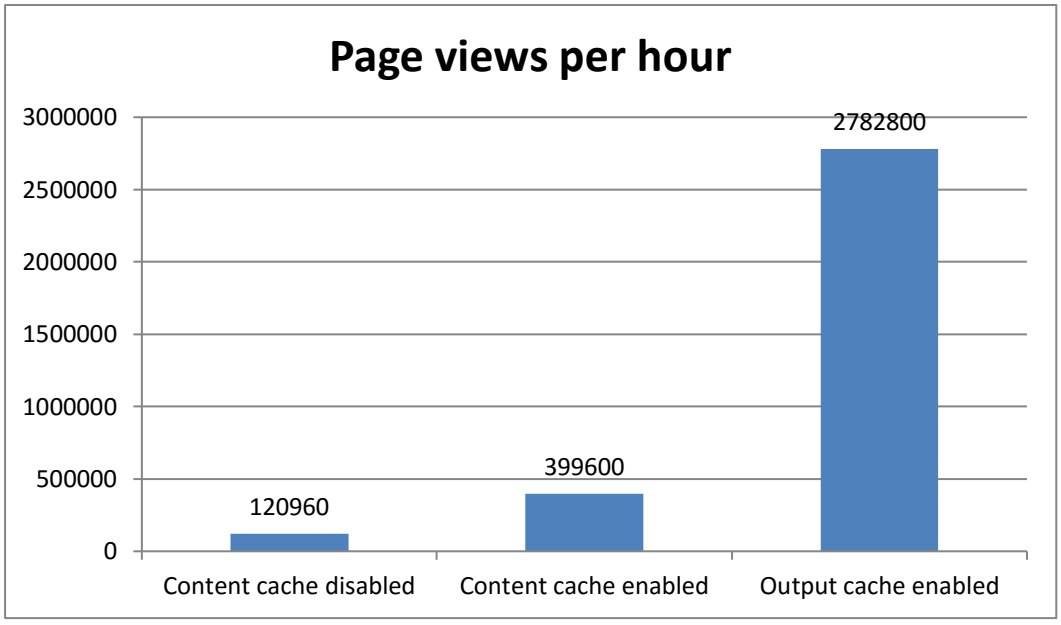
Processor(_Total)\% Processor Time
Process(w3wp)\Private Bytes
.NET CLR Memory(w3wp)\# Bytes in all Heaps
.NET CLR Memory(w3wp)\# Gen 0 Collections
.NET CLR Memory(w3wp)\# Gen 1 Collections
.NET CLR Memory(w3wp)\# Gen 2 Collections
.NET CLR Memory(w3wp)\% Time in GC
.NET CLR Exceptions\# Exceps thrown / sec
ASP.NET\Requests Queued
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

Page views per hour

The "Page View" statistic shows how many complete pages are loaded, including dependent requests, such as images, javascript files, stylesheets, etc. So, the results for Requests per Second (RPS) would be much higher (proportional to the number of dependent requests).

On the other hand, some of these files could be cached on the client side and therefore wouldn't be requested more than once per user session. In the following image, you can see the number of page views per hour for our three cache scenarios.

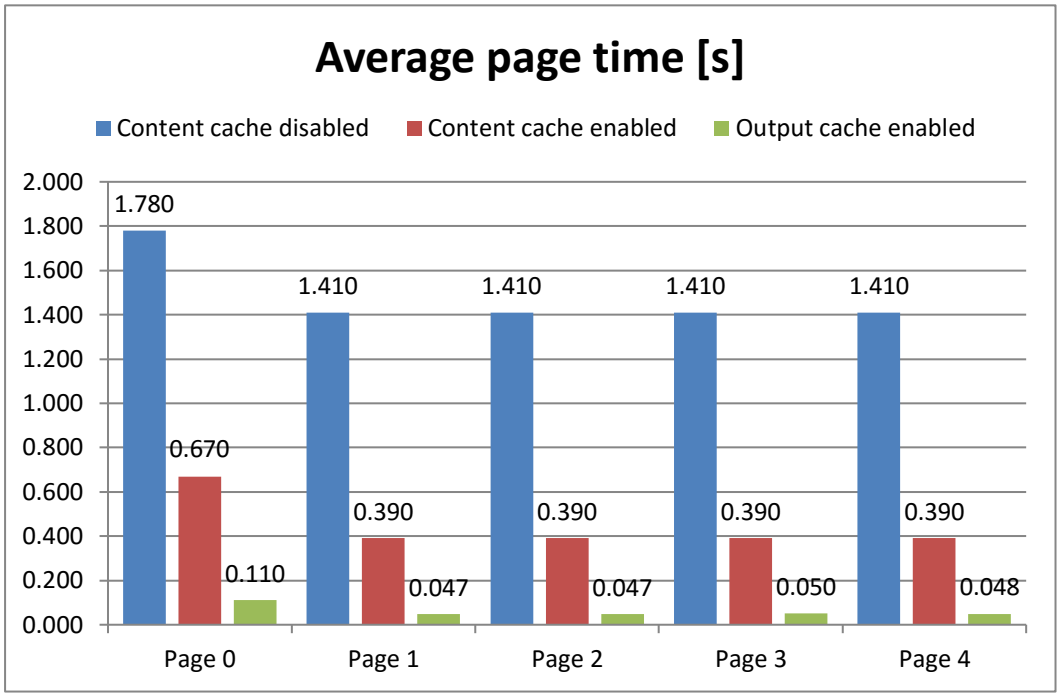


In one hour, with output cache enabled, besides the **2.78 million main requests**, there were another **14.5 million dependent requests** performed and another **22.3 million dependent requests cached**.

Average page time

The "Page Time" statistic includes the response time of all dependent requests. As you can see in the following graph, the loading of pages that display pages with caching disabled is very slow – it takes more than a second for a high number of pages stored in the database.

In contrast, output caching only has to take the HTML code cached in the memory and send it to the browser, so it is extremely fast.

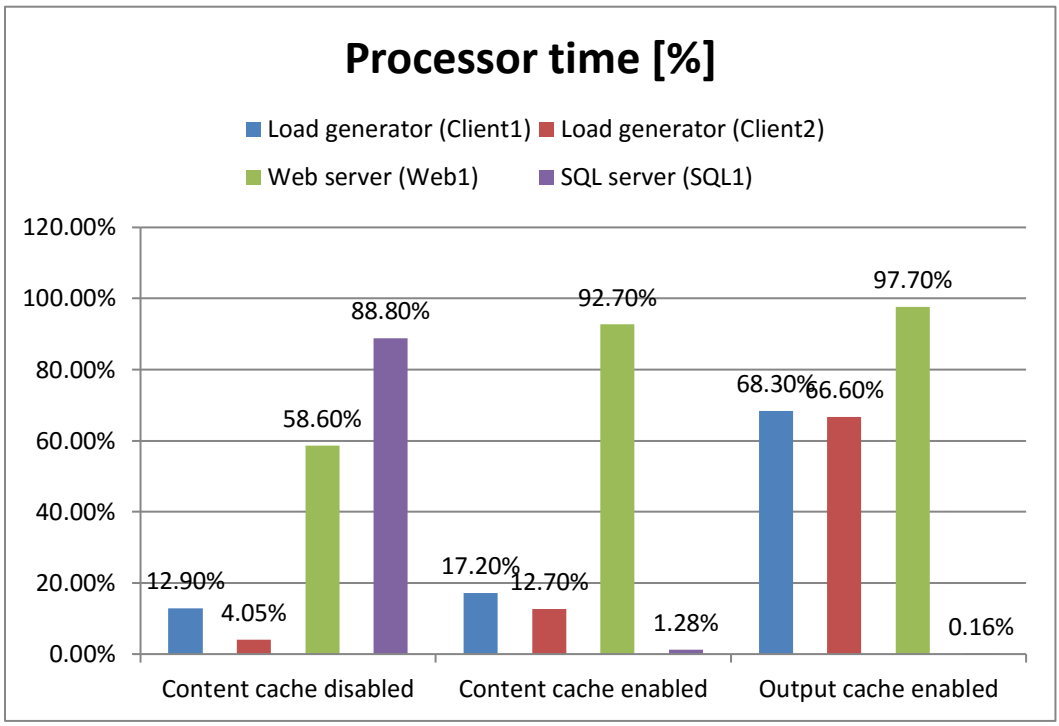


There were more than 100,000 pages in the database. The News.aspx page displayed teasers of the top four news pages, and the other pages displayed a specific news text.

Page 0	~/News.aspx
Page 1	~/News/Apple-iPad-2-In-Stock.aspx
Page 2	~/News/Community-Website-Section.aspx
Page 3	~/News/Company-Growth-Exceeds-Expectations.aspx
Page 4	~/News/New-Consulting-Services.aspx

Processor time

The "Processor Time" performance counter shows the hardware limitation of both the system and the tests. As you can see in the following graph, the processor time is a bottleneck in each scenario but on different machines.



With content cache disabled, the load generators are almost idle (do not have to generate more requests), the web server has plenty of resources available, while the SQL server is overloaded. This is the scenario in which it makes perfect sense to separate the database from the web server. If it was a shared server, with both website and database, the performance would be significantly lower.

With content cache enabled, the load generators generate more requests to the web server because the web server is able to process more of them. Notice that the SQL server is idle, since the content is cached in the application memory. The web server is now a bottleneck.

When the output cache is enabled, the load generators dramatically increase the demand for CPU power. The web server now cannot handle more requests while SQL server is completely idle.

When output caching doesn't help

As you can see from the test results, the use of the output cache may increase the overall performance significantly. But it is important to say that in some cases, the output cache should not be used:

- **Content that changes more often than every minute** – if you need to display realtime data, you cannot use output caching.
- **Personalized content with a high number of users** – if you need to display content personalized to the current user, such as the user name, or if you need to restrict the displayed content by the current user's permissions, the output cache cannot be used without some drawbacks. Kentico 10 can cache such pages, but since the page is stored in the memory for each authenticated user, it may consume lots of memory.

You can still use content caching for chosen web parts or controls in such cases, but this provides significantly lower performance.

Further reading

You can find more information about tweaking Kentico performance in the following materials:

- https://docs.kentico.com/k10/configuring-kentico/optimizing-website-performance
- https://docs.kentico.com/k10/configuring-kentico/optimizing-website-performance/configuring-caching
- https://docs.kentico.com/k10/configuring-kentico/optimizing-website-performance/loading-data-efficiently
- http://devnet.kentico.com/articles/troubleshooting-kentico-performance