Performance in ImageVault

Abstract

This whitepaper discusses performance in ImageVault when it comes to delivering images to the end user/browser.

Background

ImageVault uses a database to store image information like metadata, access rights, conversion formats, etc.. The image data itself is stored on disc. To deliver the images to the end user, ImageVault uses a handler that parses the parameters and produces the requested image.

The performance of this handler is critical to ImageVault. In this study we will compare different storing alternatives and see what performance can be expected from each.

Compared alternatives

Alternative 1, IIS

The fastest way to deliver images would be to put images in a folder in IIS and let IIS handle the delivery. This will be the most ”native” and direct method and should be the fastest alternative.

Alternative 2, EPi

The normal way to store images in EPiServer CMS5 would be to store the images in the page folder for the EPiServer page. The page folder uses (by default) a virtual path provider provided by EPiServer; namely the EPiServer.Web.Hosting.VirtualPathVersioningProvider. The version of EPiServer tested is v5.2.375.276

Alternative 3, IV

Our alternative is the ImageVault handler described above. The version of the tested ImageVault is v3.3.2.666.

The sets

Testing has been done on three different sets of images. Each set contains different image formats (gif, jpg and png). The difference between the sets is the size of the included images.

Set 1, small image set

This set contains jpg, png and gif images. The size of each image is around 15kb.

Set 2, medium image set

This set contains jpg, png and gif images. The size of each image is around 126kb.

Set 3, large image set

This set contains jpg, png and gif images. The size of each image is around 600kb.

The trials

The following tests compare the three different alternatives and examines how they perform in the same tasks. All tests are performed when the system is ”hot” and all caches has been fully populated. All tests are also performed locally where client, web server and SQL server are located on the same physical machine.

Please note that the speeds and timings that were measured in the tests only are relevant to the tested system. Only relative figures between the compared alternatives can be considered as relevant for other systems.

Trial1, single client

This trial tests a single client that downloads all images of the set and measures the download speed.

Trial2, load test

This trial simulates 100 users that simultaneously performs Trial 1. The purpose of this test is to clarify that a heavy load on the server won’t affect the tested source. The trial runs for 60 seconds and each tread will continuously perform trial 1 during this time. The output of this test is also be the speed of the downloaded images, but since this test uses 100 clients the speeds should be considered 1/100 of the measured speeds in trial 1.

The results

Trial 1

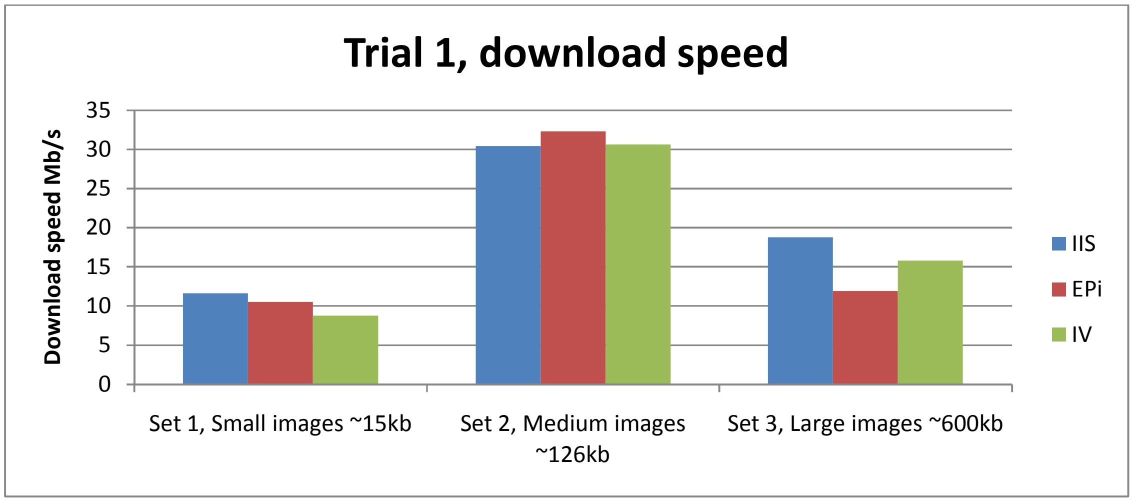

Fig 1 Trial 1, download speed

The result of this trial firstly indicates that difference in image size obviously is a factor to take into consideration when running performance tests.

We can also see that the different alternatives don’t differ very much in speed. We get some sort of indication, but since the system wasn’t under load, we can’t really tell if these figures will be relevant in a runtime system with multiple simultaneous users. That’s up to Trial 2 to determine.

Trial 2

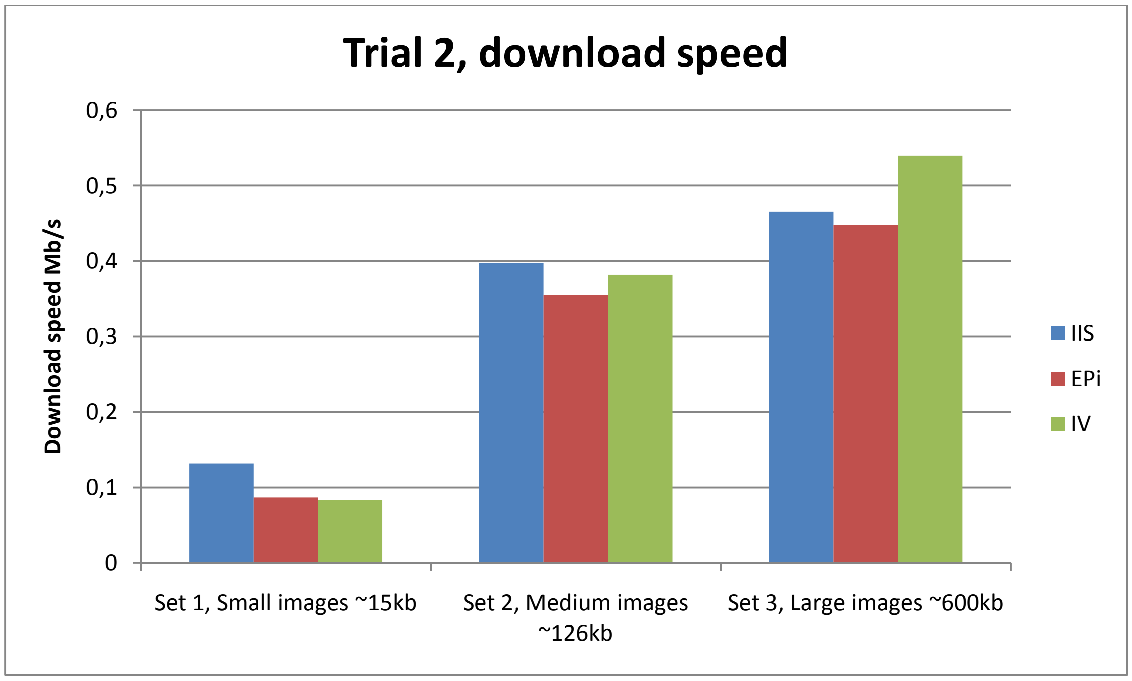

Fig 2 Trial 2, download speed

This second trial will be more reliable since it samples more requests and utilizes the whole processing capability of the system. Here we can draw the following conclusions.

For small size images EPi and IV performs 66% respectively 63% of the IIS performance.

For medium size images difference are much smaller, here EPi and IV performs 89% respectively 96% of the IIS performance.

For large size images IV passes both IIS and EPi. (116% of the IIS performance). This might seem strange, but if we look at the systems memory usage when the tests are performed, both IIS and EPi uses a lot more memory when delivering big images. This is probably the cause for IV's better performance of larger images.

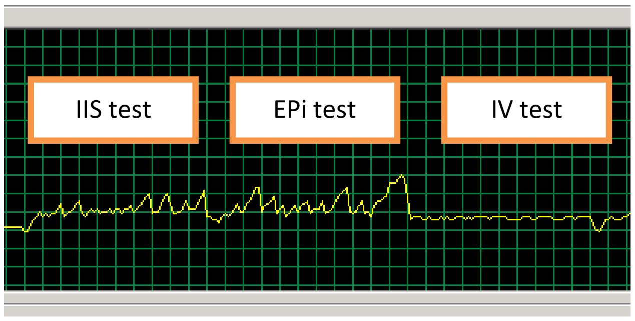

Fig 3 Memory usage for Trial 2, Set 3

Conclusion

ImageVault's performance of delivering small sized images are similar to the one provided in EPiServer. To close in on the IIS performance we have some optimizations left to do.

When it comes to delivering medium sized images ImageVault shows a very good level of performance, reaching 96% of the IIS performance level.

For large sized images, ImageVault outperforms the competitors.