Multidisciplinary Research on Geographical Information in Europe and Beyond
Proceedings of the AGILE'2012 International Conference on Geographic Information Science, Avignon, April, 24-27, 2012
ISBN: 978-90-816960-0-5
Editors: Jérôme Gensel, Didier Josselin and Danny Vandenbroucke

Caching techniques for high-performance Web Map Services

Alexander J. Loechel
University of the Bundeswehr
Munich, Germany
Alexander.Loechel@unibw.de

Stephan Schmid
University of the Bundeswehr
Munich, Germany
Stephan.Schmid@unibw.de

Abstract

The demand for digital maps on the Internet has increased considerably in the last few years. Therefore the performance of web mapping services is becoming more and more important. This paper introduces different caching techniques for high performance serving of standardized *OGC WMS*. It describes and examines different caching mechanisms based on tile caching, reverse proxy caching and web application acceleration. Furthermore it demonstrates benefits and problems and data can be modified for different caching techniques. The paper outlines the advantages of WMS caching systems and investigates the behavior of these systems with an increasing number of concurrent requests using benchmarking tests. This includes the examination of applicability of the *INSPIRE* level of service agreement for view services.
Keywords: Caching, INSPIRE, Benchmark, Generic approach, WMS

1 Introduction

The use of maps on web pages has increased in the last few years. Professional mapping services like *Google Maps*, *Bing Maps* and other companies dominate the market. The utilization of these professional service providers has the advantage that a user can receive maps from a high performance server cluster. Of course this professional mapping service cluster uses different techniques to enhance the speed of the transfer. So the bandwidth between the mapping services and the user is the limiting factor on receiving the map [1].

If a Geo Information System (GIS) project uses its own Web Map Service (WMS) server (e.g. *GeoServer)*, loading and rendering of maps takes much longer compared to professional mapping services. This can be a result of limitation on powerful hardware or insufficient knowledge in efficient caching techniques, because those standard installations are normally not optimized. According to the *INSPIRE* directive (Infrastructure for Spatial Information in the European Community), which came into force on 15th May 2007, it is necessary that Spatial Data Infrastructures (SDI) of the member states are compatible and usable in a trans-boundary context [2].

This Paper describes different techniques for caching a WMS. The paper presents a test setup for performance tests of different caching software. The result of each test setup has been analysed in terms of the *INSPIRE* directive. The results will show, how transfer time can be improved and how the *INSPIRE* directive can be fulfilled using caching techniques.

2 INSPIRE Directive

The INSPIRE directive defines the performance of a web service representing how fast the service request should be completed. This proposes a set of items [3]:

- *Response time* is the time required to complete a web service request.
- *Transaction time* represents the time that passes while the web service is completing one complete transaction.
- *Latency* is the round-trip delay (RTD) between sending a request and receiving the response.
- *Execution time* is the time taken by a web service to process its sequence of activities.

The *INSPIRE* directive further describes a service level agreement for the use of WMS services in the field of *INSPIRE*.

> "For a 470 KB image the response time for sending the initial response to a *GetMap* request to a view shall be a maximum of 5 seconds in a normal situation." [4].

3 Theory of caching

In the beginning of the World Wide Web (WWW), static content on web pages was usual. Today the demand for real-time information has increased the amount of dynamic generated content. Just-in-time-generation of dynamic content, like map images, is usually extremely slow, so caching dynamic content has become a common technique [5].

The Hypertext Transfer Protocol (HTTP) defines the communication between clients and servers. To conclude an agreement how the requested resource should respond, this contract is exchanged as HTTP headers, which includes generation date, client and server information, content type specification and caching guidelines [6].

HTTP is defined as stateless, any similar request should result in the same response. Therefore any response to a request can be cached. The protocol defines data transfer on GET and POST request methods. Caches normally ignore all POST-requests and any GET-request that contains a "?" (query-call) in the URI. Browsers neither expect them to be cacheable nor send HTTP headers for cache requests. The *OGC* specifies their web services based on those GET-requests queries, so an OGC Web Services (OWS) request normally would not be cached. But the HTTP standard contains several techniques which allow shared caches and browsers to cache content (static & dynamic).

3.1 Caching of WMS GetMap Requests

A WMS can be very effective in combination with modern AJAX clients (e.g. *OpenLayers*). *WMS GetMap* requests are normal HTTP-Get-requests, structured on the URI with a hostname, WMS service path and a set of query data. The amount of query data is theoretically infinite, but in real world scenarios the requested data is very limited by the used mapping APIs. Common mapping APIs like *OpenLayers*, *Google Maps*, *Bing Maps*, etc. automatically request only tiles of 256x256px with pre-defined boundary-boxes and different scales, which is not reflected in the WMS-GetMap-Request. Those two variables are the most important and most often changing parameters, so the set of possible requests on normal usage is limited to an easily manageable amount and repeatable number of requests.

An example WMS-GetMap Request looks like: http://hostname:port/wms?LAYERS=layer&FORMAT=image/png&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&STYLES=&SRS=EPSG:4326&**BBOX**=11.42578125, 47.98828125,11.6015625,48.1640625&**WIDTH**=256&**HEIGHT**=256, where the bold variables are important.

Several different techniques have been developed to cache WMS-requests, in context of GIS Web Map Tiled Services (WMTS) have been focus of research and are well-known by the community. They provide a standard-based solution for serving digital maps, using predefined image tiles [16]

In the larger web technology community generic HTTP-caches are more common. Both approaches have different focuses and limitations for a web project. Tile caches seed all map-tiles during initialization and just serve them. HTTP-caches take a look-up if the request has already been cached and serve it. If the request has not been cached it will be forwarded to the map-server. For the first request a HTTP-cache must be slower than a tile cache. The advantage of generic HTTP-caches is that all valid WMS-requests can be handled and cached. A second advantage is the acceleration in the miss-case. If a map-server generates a tile, it also needs time to transfer the produced map-image to the client. Therefore a thread of the server is blocked by the transfer, which is bind to the transfer-bandwidth. The use of a HTTP cache which is directly bind to the map-server, can reduce this problem. It always uses the maximum possible bandwidth even if cache and map-server are on different server instances. Therefor the requested map-image can be served to the client more efficiently. A large disadvantage of all current tile cache implementations, but not for generic HTTP-caches, is that changes on the underlying data are not recognized by the tile cache. It is the so called purge problem.

3.2 HTTP cache headers.

HTTP cache headers define exchange information (cacheable content) between client, shared caches and the server. The HTTP protocol version 1.1 defines several HTTP header statements. In the field of caching there are existing several cache headers that are sent as part of the HTTP request (client side values) as well as HTTP response (server side values) [6]. This Paper describes the basic methods for cache manipulation:

- *Expires header:* The *Expires header* is a response only header, that indicates by a RFC 1123 timestamp, when the response should be expected as not fresh [9].
- *If-Modified-Since* header: *If-Modified-Since* header is a request-only header, that a browser or cache can use to request a document, if it has been modified since the last call. If a *HTTP-status code 304 Not Modified* is transferred, it presents the current cache entry, because no changes have been made.
- *Cache-Control* header: The *Cache-Control* header is a bidirectional header; it can be used by client browsers as well as by servers. It defines how the requested document should be handled by caches using different cache control values. For example the max-age=seconds cache control tells the cache to take the response as fresh for the defined time, without revalidation. On Client side it indicates that the client is willing to accept a response whose age is not greater than the specified time in seconds.

Figure **2** shows several stages of caches, marked with orange background-color. Those shared caches can only hold a copy of the requested map images (HTTP documents) if the configuration and cache headers allow it.

Webserver and cache-proxies have the capability to set and modify HTTP-headers. Setting the right headers for caching might increase the speed of a WMS project. How this manipulation can be implemented is out of scope of this paper [7].

3.3 Caching and modifying data - the purge problem

The use of caching leads to the purge-problem, the propagation of modifications of data. This is a very complex topic if map-data changes often over time.

Normally non-cached map servers always render the current available data. Caches try to serve their cache entry. Tile caches ignore the HTTP standard and will always serve the seeded tiles. If no further manipulations of cache-headers have occurred, shared caches will not held any cache-entries itself.

For generic HTTP-caches the purge problem is more complex, as all several stages of caches (Figure **2**) can act like shared caches and proxies on the Internet or like a cache in the administrated area which can hold a cache entry.

The purge effect leads to the problem, that a cache or map-server might respond outdated maps. This is assumed to be critical in two cases:

1. Manually changing data: The map has to reflect the change immediately; otherwise users get confused and try to perform the change again.
2. Automatically changing data: public map layers with up to date information, like traffic conditions or weather data.

There are special techniques to solve this problem, but these cannot be explained in detail here due to space limitations.

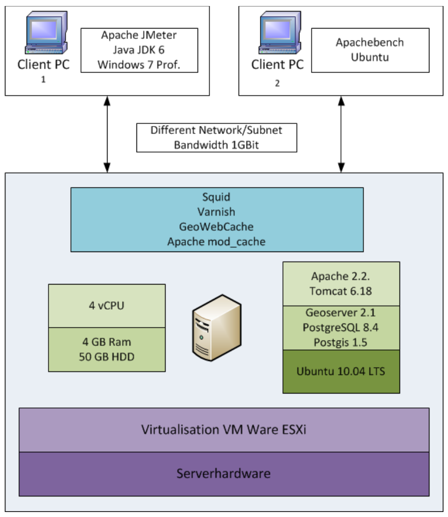
For case 2 a possible solution is a caching time less than one minute.

4 Test procedure

The performance tests described in this section investigate different caching mechanisms in terms of *OGC WMS* services including tile caching, reverse proxy caching and web application acceleration. The tests include the behaviour of WMS caching systems with an increasing number of concurrent requests as well as the examination of applicability of the *INSPIRE* directive.

The configuration of the different cache software follows their standard documentation for HTTP acceleration/caching with binding to the WMS server *GeoServer* [10]. For the tests a test-bed of the second authors institute is used. Figure **1** shows the current configuration of the test-bed. For *Squid* proxy the refresh pattern for *HTTP Get-query-requests* has to be changed. *GeoServer's* general cache headers were set for each layer.

Figure 1: Test-bed for the performance tests



- *Non-cached:* The results of non-cached systems reaction on concurrent requests offer the possibility to appraise performance improvements using different caching systems.
- *Static tile image*: To have reference values, a static version of a disk stored map tile image was included in the performance test. *GeoWebCache* represents a common tile caching software. GeoWebCache produces tiles with a size of 256 x 256 px. Tiles were seeded after setting up the web map service [10]
- *Apache mod cache* represents a traditional configuration as proxy server.
- *Varnish* represents a special caching system on a single machine usable only on Linux operating systems, a so called application accelerator [12].
- *Squid* represents common caching software that is used for large caching and clustering projects. For the Apache Benchmark test two versions of Squid were used, the widespread version 2.7 and the newer reimplementation, version 3.0. [11]

Two tools are used, Apache Benchmark and Apache JMeter [7, 8, 13]. Apache Benchmark is the standard for HTTP benchmarking, supplies all information and even a statement about the quality of services. For a more realistic test on tiling clients additional Apache JMeter tests are performed. Apache JMeter is able to simulate a browser-like behavior, which is important to investigate common traffic conditions. Two test runs were performed with a number of requests with specified concurrency level, with a variety of 1 to 500.000 requests and a maximum of 1000 concurrent requests.

1. The first run requests an image with a file size of 256 x 256 px, the typical WMS tile size.
2. The second run requests an image with a file size of 800 x 600 px. This is the example image size of the INSPIRE directive.

For the tests with GeoWebCache 12 tiles were requested with each a size of 256 x 256 px. All map images were requested in format PNG. Tests were performed in decimal steps from 1 to 100 users. All tests were conducted with each of the four layers.

5 Results

Table 1 reflects the determined performance level of each system as a result of the Apache Benchmark test runs. The tests with Apache Benchmark can be summarized as follows: Caching systems increase the ability to handle *"requests per second"* to an average of 3700 / 1050 r/s compared to 50 / 25 r/s by *GeoServer* itself. This is an increase by the factor of 40-70 and thus reduces the necessary time per request from 21000 ms / 41000 ms to an average of 300 ms / 950 ms, which is a reduction by a factor of 50-100.

Table 1: Results of Apache Benchmark

Cache System	256x256 30 KB PNG WMS		800x600 105 KB PNG WMS	
	request / sec	response time / request [mean]	request / sec	response time / request [mean]
non-cached GeoServer	50	21000 ms	25	41000 ms
GeoWebCache	3700	310 ms	*unable to perform such a request*	
Apache tile static served	3700	290 ms	1070	930 ms
Apache with mod_cache	3600	290 ms	1050	930 ms
Varnish	3700	270 ms	1050	920 ms
Squid 3.0	2550	390 ms	1000	1000 ms
Squid 2.7	2650	380 ms	950	1050 ms

The tests with the static file performs in both tests cases very fast, with a quality of service of 90 % within 65 ms. The very small differences of the systems GeoWebCache, Apache with mod cache and Varnish astound. The large difference in comparison with Squid versions, 1000 request less per seconds and additional 100 ms per request, was not assumed. The configuration of Squid proxy is far more complex than for every other cache system, a non-optimal configuration of the test Squid systems must be supposed. Squid 3.0 is relatively faster serving larger file sizes compared to other caching systems, but compared to Squid 2.7, it is slower (see Figure **3**, Figure **4**).

A reason for this behavior might be the *Squid* reimplementation. *GeoWebCache* transfers only tiles of a specific size. *GeoWebCache* is not able to answer any request, on the second test case, so it just responded an error code. The figures show that the test server fulfills the *INSPIRE* directive of 90% within 1.5 seconds for the defined and generated map sizes. The quality of service requirements of the *INSPIRE* directive is meet by all caching systems in our tests, with the defined image sizes.

The comparison of cache server results to the static served version of the tile shows that these results seem to reflect almost the hardware limits of the test setup, especially the bandwidth between server and test client. This is also indicated by the change of necessary time per request over 90 % which is based on the transfer stack of the ISO-OSI reference model [14].

The package collision on the network layer decreases the possibility to transfer packages successfully. HTTP is a layer 7 protocol, therefore it is not aware of connectivity problems in the transport stack. So a higher bandwidth might increase the number of requests handled per second.

JMeter results (see Figure **5**) also show that caching systems increase the speed of multiple WMS request handling. Squid and *Varnish* are almost equally fast, while *GeoWebCache* and Apache mod cache have slightly less performance. The number of successfully transferred responses is also limited by the bandwidth. 80 threads can be run by every caching system, with a response time that fulfills the *INSPIRE* directive. Only *Apache mod_cache* needs a little longer for the 100 requests with 5121 ms. Every caching system pushes a WMS request to meet the *INSPIRE* directive.

In the following the characteristics of the systems which are reflected by the results, will be set in the context of performance-enhancement. *GeoWebCache* is directly included in *GeoServer* and therefore directly bound to the underlying data. Tile caching systems are aware of the *OGC WMS* standard, respectively there is an own standard for tile caching: *Web Map Tile Service* (WMTS) [15].

For generic HTTP caching systems the response document content type or bounded service is not relevant. They are able to cache everything that is conforming to the HTTP protocol, where the statelessness of the protocol results in same responses for same requests. Caching removes the necessity for a permanent generation of map tile images.

The server can handle more requests concurrently and reduces the necessary amount of hardware infrastructure.

Because of less concurrent requests sent directly to the map server, there is less possibility for failure occurrence. The result of this is a very robust system. Test results show that even with more than 1000 concurrent requests the failure rate and the system response time does not dramatically increase.

6 Conclusion and outlook

As the test results show, caching of WMS services is possible and will increase the performance of a WMS server. Caching has some advantages and limitations that have to be considered for productive use.

The differences between specialized spatial tile caching systems and generic HTTP caching systems are in detail, both have some advantages. The HTTP protocol, together with cache header settings, offers the possibility for standard compliant browsers and shared caches to check the freshness of their local cache entry. Caching is a reduction of network traffic with such an amount of concurrent requests. One common limitation of caching systems is a result of the purge problem. It is necessary that caching software recognizes data changes in an appropriate time. Another limitation of generic HTTP caching systems is the ability to assume content for pre-fetch, content is cached just in time.
Spatial caching systems can generate tiles for the complete bounding box in advance.

We strongly recommend all GIS projects to make them self-aware of caching techniques and how the cache-headers work. Faster WMS project using caching techniques might increase the acceptance of GIS technologies.

Further studies have to analyze the edges where this setup has reached it limits:

- *Squid* cache clusters may perform more efficient in large setups than lightweight single server cache systems like the tested Varnish.
- Another research topic might be to analyze, how prefetching algorithms of spatial caches could be separated and used in generic HTTP caches.

References

[1] Michael P. Peterson. *International perspectives on maps and the Internet.* Springer, Berlin; New York, 2008. ISBN 978-3-540-72028-7.

[2] European Commission. *INSPIRE*. http://inspire.jrc.ec.-europa.eu/.

[3] European Commission. *INSPIRE Network Services Performance Guidelines*. European Commission, 2007.

[4] European Union. *Commission Regulations: 1088/2010*. European Commission, 2010.

[5] Brian Krupp. *Exploration of dynamic web page partitioning for increased web page delivery performance*. 2010.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol HTTP/1.1. RFC Editor*, United States, 1999.

[7] Sascha Kersken. *Apache 2.2.* Galileo Press GmbH, 2008. ISBN 9783836213257.

[8] Apache Foundation. *ab - Apache HTTP server benchmarking tool.* http://httpd.apache.org/docs/2.2/-programs/ab.html.

[9] R. Braden. *Requirements for Internet Hosts Application and Support*. RFC 1123, Internet Engineering Task Force, United States, 1989.

[10] OpenGeo. *GeoServer User Manual.* http://docs.geoserver.org/stable/en/user/index.html.

[11] Dirk Dithardt. *Squid.* dpunkt Verlag, 2006. ISBN 9783898643177.

[12] Varnish. *Varnish Cache*. 2010. https://www.varnish-cache.org.

[13] Emily H. Halili. Apache JMeter: *A practical beginner's guide to automated testing and performance measurement for your websites*. Packt Publishing Limited, Birmingham, 2008. ISBN 9781847192950.

[14] Hubert Zimmermann. *OSI Reference Model: The IS0 Model of Architecture for Open Systems Interconnection.* IEEE Transactions on Communications. IEEE, 1981

[15] Open Geospatial Consortium Inc.; *Web Map Tile service Implementation Standard*, OGC 07-057r7

[16] Dražen Odobašić et.al; *Web mapping is distributed – overview of open source proxy and processing services*, 2011, http://zgis16.plus.sbg.ac.at/gi-forum/images/stories/GIforum2011/ea_odobasic_etal_webmappingisdistributed.pdf

Figure 2: Schema of a WMS GetMap request - response communication path and managing area

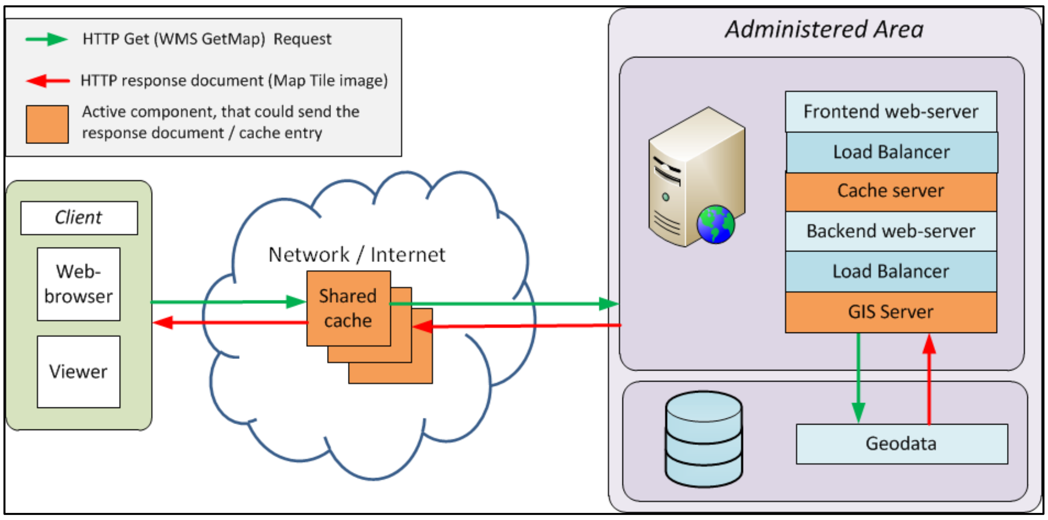


Figure 3: Results of the Apache Benchmark test for tiles

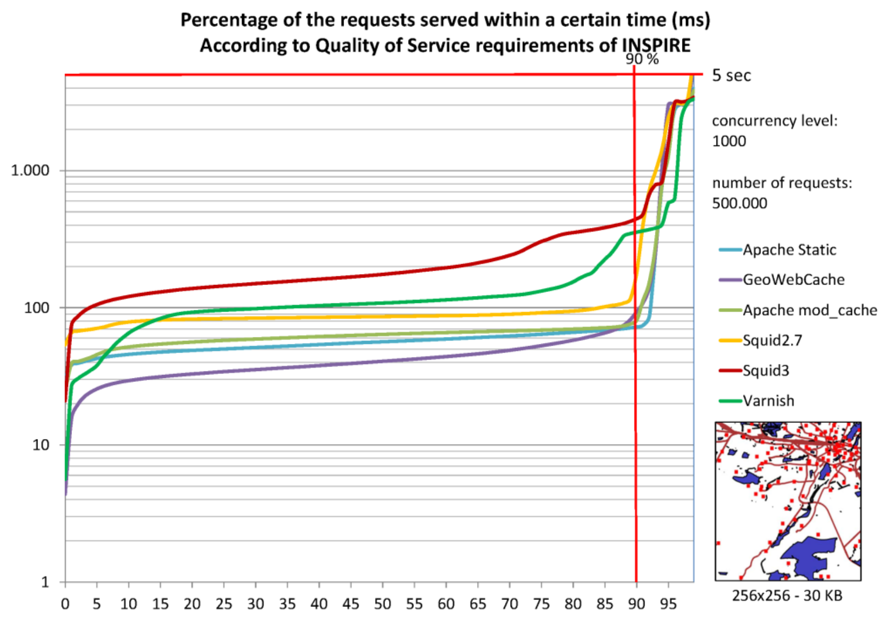


Figure 4: Results of the Apache Benchmark test according to INSPIRE directive

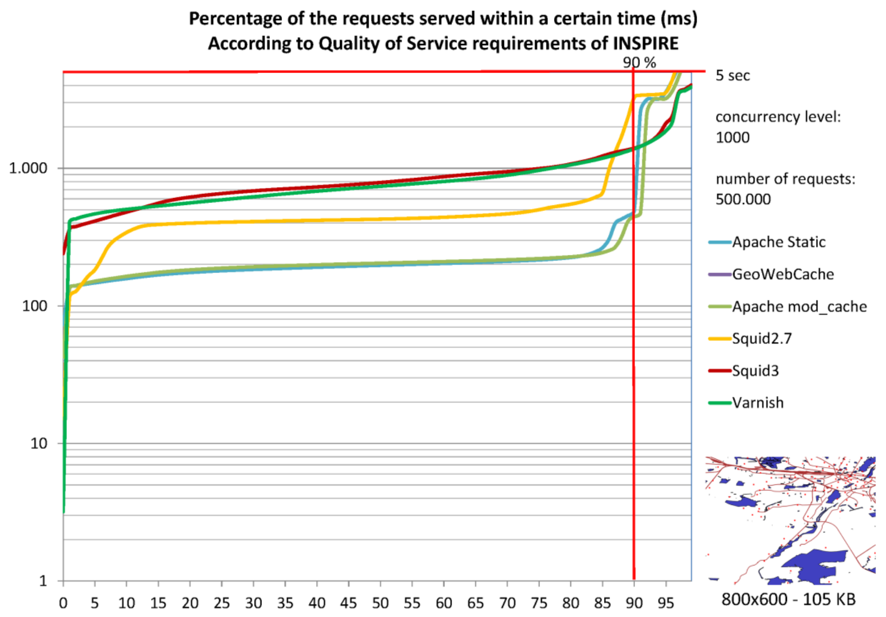
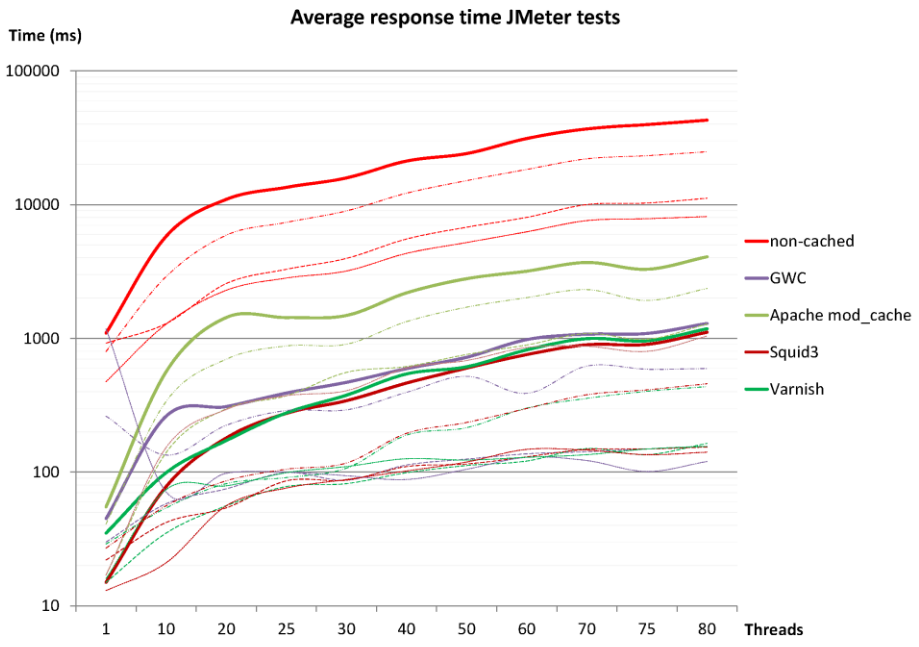


Figure 5: Results of the Apache JMeter tests


Average response time JMeter tests
Time (ms)
Threads
non-cached
GWC
Apache mod_cache
Squid3
Varnish