International Journal of Spatial Data Infrastructures Research, 2013, Vol. 8, 43-73

Comparison of Different Caching Techniques for High-Performance Web Map Services*

Alexander Loechel[1], Stephan Schmid[1]

[1]University of the Bundeswehr Munich; alexander.loechel@unibw.de; 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 article introduces different caching techniques for high performance transfer of data using standardized Open Geospatial Consortium (OGC) Web Map Services (WMS). It describes and examines different caching mechanisms based on tile caching, reverse proxy caching and web application acceleration. Furthermore it demonstrates benefits, problems and how data needs to be modified for different caching techniques. The article outlines the advantages of WMS caching systems and investigates the behaviour of these systems with an increasing number of concurrent requests using benchmark tests. This includes the examination of applicability of the INSPIRE service level agreement for view services.

Keywords: Benchmark, Caching, INSPIRE, OGC, 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 services from other companies dominate the market. The use of these professional service providers

DOI: 10.2902/1725-0463.2013.08.art3

has the advantage that users can receive maps from a high performance server cluster which uses different techniques to enhance the speed of data transfer, bandwidth between the mapping services and the user is the limiting factor on receiving the map (Peterson, 2008).

If a Geographical Information System (GIS) project uses its own server (e.g. GeoServer) for Web Map Services (WMS) data transfer takes much longer compared to professional mapping services. The server needs to load, render and serve the maps. Usually these standard installations are not optimized, caused by limitation on powerful hardware or insufficient knowledge about efficient caching techniques (Krupp, 2010).

In some projects the WMS server needs to serve the response in a comparable performance to the professional data providers. As hardware might not be available or is too expensive, caching is a cheap and effective way to increase performance. Performance is a general requirement for all IT services. To compare several caching techniques we either compare them to each other (benchmark approach) or define standard requirement (standardization approach) or define limit of user acceptance (empiric approach).

A standardization approach is the Infrastructure for Spatial Information in the European Community (INSPIRE) directive. According to the INSPIRE directive, which came into force on 15th May 2007, it is necessary that the Spatial Data Infrastructures (SDI) of the member states are compatible and usable in a trans-boundary context (European Commission, 2013). The INSPIRE directive defines a maximum acceptable loading time of 5 seconds for web services.

As sometimes standardization commissions just describe the state-of-the-art of their technology, we looked for a general empiric approach, too. Nah (2004) has published a general study on user experience and acceptable waiting time. Her study indicates that waiting times longer than 2 seconds without any visual feedback may confuse the user and lead to an abandonment of the service.

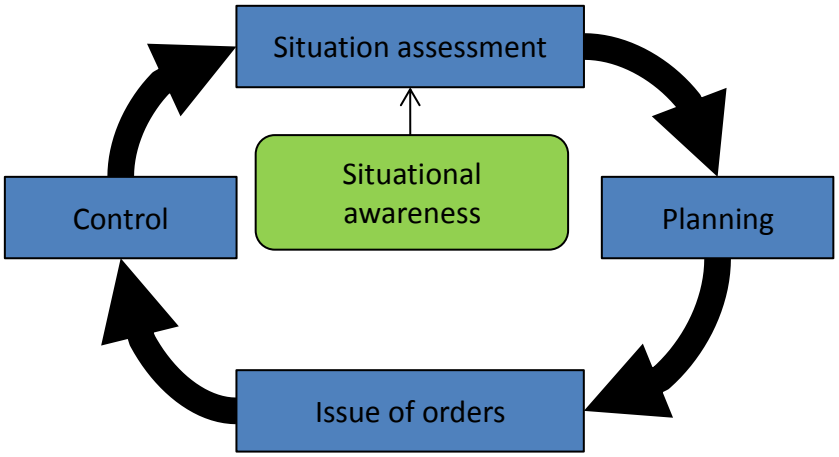
We conclude that a high performance response to a web request is a mandatory requirement for good web services. The two limits, of maximum 5 seconds for a full map load (INSPIRE) and a necessary visual feedback within 2 seconds (Nah, 2004), are sufficient for an assessment of different caching techniques.

This article extends the AGILE 2012 Short Paper Caching techniques for high-performance Web Map Services (Loechel and Schmid, 2012).

2. BACKGROUND OF THIS STUDY

The ongoing PhD thesis of Alexander Loechel (Loechel, unpublished) and several papers (e.g. Loechel et. al., 2012) focus on an open-source approach for military situational awareness system. Figure 1 shows the military command & control (C2) process placing the term “situational awareness” in a military context. Situational awareness is a specific understanding of current tactical situation in mission theatre. As visualization is one key to improve quick recognition and information retrieval, a tactical map is the main visualization instrument.

Figure 1: Cycle of Military Command & Control Process



Source: Loechel et al, 2012

The major requirement for a modern military situational awareness system can be described as: “*Providing all necessary information to the military decision makers to make well-formed decisions.”*

A tactical map is the major overview tool for military decision makers; providing documentation of current and historical status, communication and planning capabilities. A military map typically provides information about:

- Tactical terrain;
- Position of troops (own and enemy);
- Reported incidents.

The military situational awareness system aggregates information in context with position, time and quantity (e.g. division; see Figure 2). An interactive military

map which provides and presents all relevant information in a clear and concise manner, is a successful step in support of military leaders.

Figure 2: Example of a Comupter Generated Tactical Map

Source: Taktiklehrer im Reservistenverband, 2010

Military forces have used maps since the beginning of human culture. The use of maps will continue and therefore some of the primary requirements for a modern military situational awareness system need to be determined. The requirements and capabilities for the C2 tool (command & control) are the following:

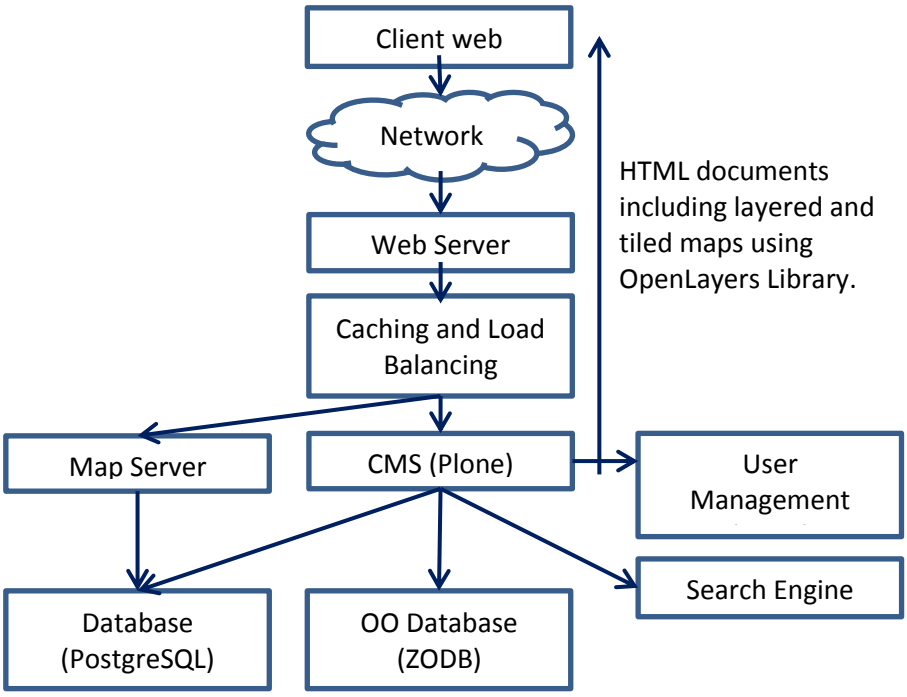
- Using military symbols (NATO Joint Symbology; NATO Standardization Agency, 1986);
- Creating high resolution maps, with different styling for different zoom levels;
- Differentiating physical elements by type and status (e.g. roads by significance and capability for different kind of vehicles or damage status);
- Up to date information presentation with minimized load time;
- A system with a maximum availability.

For a military application all fundamental key-points of IT-security need to be fulfilled; confidentiality, integrity, authenticity and availability have to be met. Availability and a guaranteed Quality of Service (QoS) is the most important

precondition for the success of a mission. The requirements defined by the INSPIRE directive should be transferred as one of the minimum requirements for a military C2 tool. The study of Nah (2004) even “suggests that the presence of feedback prolongs users’ tolerable waiting time is approximately 2 seconds” and that the “continuity of human thought processes is necessary for effective problem solving and a delay of more than 2 seconds may lead to psychological step-down discontinuities, which divert one’s attention from the thought processes.” Continuous interrupting of thought processes or binding attention to a system for a longer time than necessary could result in either refusal of the system or in the military context of a combat support system even to serious consequences as injuries or death. User experience is a major quality factor for such an environment. Load time has to be considered as critical as availability and up-to-dateness. Caching is belongs to the easiest and reliable solution to reduce load time on generated content. Implementing caching has been considered on design of the system.

As the developed C2 tool has to run distributed, with no potential uplink and close to forces in mission theatre, it has been designed as an integrated system that can be executed on standard mobile PC equipment. The system stack (see Figure 3) includes a Content Management System (CMS) that is able to organize all relevant information, including changes on terrain data, a map server that generates the map tiles, a map tiling library and a caching component that reduces the actual load on the system.

Figure 3: Component Architecture of the Implemented C2 System



Source: Loechel et al, 2012

A tactical map is one of the most important components of the application; it is also the most critical point for server performance. Dynamic generation of all map tiles for every request will produce such a load on the server that neither an acceptable QoS can be reached nor soldiers benefit from the application. The idea that resolved in this study was to distinguish between several layers of a map (see Figure 4), the acceptable time of staleness of these layers and how to avoid multiple generations of tiles. Caching seems to be a good possibility to increase the performance. The next step was to analyze if it is necessary to add a specialized Web Map Tile caching solution or if it can be handled by a more generic HTTP caching method, which can also be used for performance optimizations on the CMS and webserver sides of the application.

Adding additional components to system architecture will increase base load and complexity, especially on failure prevention and availability. Assumed hardware setup for system stack is sufficient for all software components in sum, without exceeding RAM or storage quota necessary for caching. This assumption is necessary for our comparison and system design. If available RAM is limited, cache servers will not be able to hold all requested map tiles in memory and will forward requests to the map server which needs to regenerate the tiles. In worst

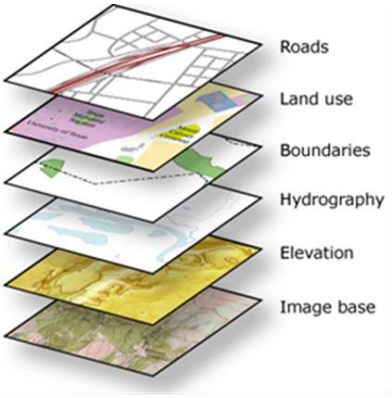
case this means that the amount of tiles in cache is so low that every request causes the caching system to invalidate a tile, pick up the currently request tile and continue for all its entries and requests, so that all tiles will be regenerated and none gets served from cache.

Trying to reduce the necessity of serving generated content on time or even make this generation quicker, system design is to split map into several independent layers which have different cache limits. The map layer in Figure 4 shows an example of several independent layers that all in combination represent a map.

Even if we expect available mobile hardware as powerful as a server, we still have to avoid tile generation where possible. It is like a financial business case where deliver of existing tiles is cheaper than generating new tiles. Storing any information (tiles) in an organized system and reviewing this is cheap, generation of new information (tiles) is expensive.

Any map tile that is stored in a cache reduces the necessary calculation processes. An evaluation layer, boundaries of political districts and land use don't change often or quickly, so these layers might be cached for more than one week. Roads and building layers usually would be considered as a more long term existing object, so that they can be cached for a longer period, but assuming that damage status of a road or building changes in very few seconds it is considered to set a maximum of 5 minutes cache time. Troop positions and movements never ought to be cached or even less than one minute, depending on the resulting server load.

Figure 4: Example Layer Stack, Layer Concept of Maps



Source: ESRI, 2010

3. WMS & CACHING THEORY

A WMS is a special form of a web service. A WMS renders maps with spatial content dynamically from geographic information. Map in this context is understood as the portrayal of geographic information as a digital image file which is suitable to be displayed on a computer screen. There are several methods available for caching a WMS. The following section introduces the WMS standard and the theory of caching.

3.1. Web Map Service and its Regulations

The international WMS standard is developed and maintained by the Open Geospatial Consortium (OGC), which is a non-profit organization founded in 1994. Today, the OCG counts more than 442 active members and universities, GIS software vendors and administrative bodies (OGC, 2013).

Further the WMS standard was approved by the International Organization for Standardization (ISO) in 2005 (OGC, 2005). In the military community the Defence Geospatial Information Working Group (DGIWG) maintains and adjusts the standard to special military needs, like military coordinate reference systems. DGIWG developed a profile, the DGIWG WMS 1.3 Profile and systems requirements for interoperability for the use within a military environment, which is combat ready at the moment (DGIWG, 2008).

The WMS standard defines three operations which can either be invoked using a standard web browser or by using common GIS software like ArcGIS or Quantum GIS.

The WMS standard consists of three operations:

1. GetCapabilities;
2. GetMap;
3. GetFeatureInfo.

1. The GetCapabilities operation returns the service-level metadata. This is a machine readable and human understandable description of the server's information content.

2. The GetMap operation returns a map whose geographic and dimensional parameters are well defined. If the request is not well defined the response of the WMS is an exception.

3. The GetFeatureInfo operation returns information about particular features shown on the map. This operation is optional. It can only be used for layers which support the parameter queryable=1. The GetFeatureInfo operation provides more

information for the user. The user receives the response of the GetMap request and can then obtain more information about separate features selected by a point (I,J).

The use of a WMS in a productive environment needs to define the QoS. Therefore organizations (e.g. W3C, ISO) define quality elements. Today the quality focus is on compliance and on performance. For testing the compliance of the WMS several engines exist, e.g. OGC TEAMengine. For other quality elements the INSPIRE directive has been entering in force in May 2007, whose aim is to establish an infrastructure for spatial information in Europe. The directive is operated by the 27 member states of the European Union (European Commission, 2007).

The directive ensures that spatial data infrastructures of the member nations are compatible and that data is useable in a trans-boundary context. For the transfer of spatial information the member states adjust the OGC WMS standard for their specific needs to cover all 34 spatial data themes.

To use the data in a trans-boundary context it is essential that the transfer is performant and the service is highly available.

The INSPIRE directive defines performance of a web service as representing how fast the service request shall be completed. This proposes a set of performance items (European Commission, 2007) 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. This transaction time depends on the definition of web service 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.

Further the INSPIRE directive describes a Service Level Agreement (SLA) for the use of WMS in the field of INSPIRE (European Commission, 2007).

"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. A normal situation represents a period out of peak load, which is set at 90 % of the time"

This SLA can be reached and optimized within the use of caching for the WMS (see Chapter 6).

3.2. Caching Theory –The Idea

In the beginning of the World Wide Web (WWW), static content on web pages was usual. The commercial internet, started by the transition of Arpanet to a provider based commercial solution in 1990. Bandwidth and connection time was very expensive. The multiple transmission of static content should be avoided. The first caching-servers were proxy caches that limited access to the global network.

The early development was influenced by the new emerging standards Hyper Text Markup Language (HTML), Hyper Text Transfer Protocol (HTTP) (see RFC 2616, 1999) and Uniform Resource Locator (URL). The HTTP standard defines in its initial version all necessary information to check if a HTTP-Document can be cached:

- Date – used to identify cached documents;
- Expires – indicates how long a document should appear as valid;
- Last-Modified - indicates the date and time at which the sender believes the resource was last modified.

With the emergence of dynamic content in the application field, caching has been reused for performance optimization on server side. On modern web and application servers the maximum of simultaneous handled request is limited by the available CPU cores, available processes and threads. Experience showed that normal number of simultaneous handled requests is between 5-250 requests per second depending on the used platform, implementation and complexity of generated content. To handle a large amount of traffic (for example 200.000 request per second) there is either the need of a huge farm of interconnected servers or a strategy to reduce the load on the existing servers (for further information see Jay, 2012) Caching is, beside load balancing one of the easiest ways to reduce load on servers.

3.3. General HTTP Request Rules

The OGC standard defines an implementation of the WMS using the HTTP. Thus, the online resource of each operation supported by a server is a HTTP URL. That means the URL can be different for every request and every operation, but the URL has to be conformant to the description in IETF RFC 2616 (RFC 2616, 1999).

HTTP supports ten request methods, the two most common used are: GET and POST. A WMS server needs to support the GET method (mandatory) while the support for the POST method is optional. IETF RFC 2396 (1998) reserves some characters as significant and requires that these are escaped from requests as there might be a conflict with their usage. The standard reserves several

characters for use in the query portion of WMS requests. If the characters “?”, “&”, “=”, “,” and “+” appear in one of the roles defined in Table 1, they shall appear literally in the URL. If these characters appear elsewhere (for example in the value of a parameter), they shall be encoded as defined in IETF RFC 2396 (1998; OGC, 2005).

Table 1: Reserved Characters in WMS Query String

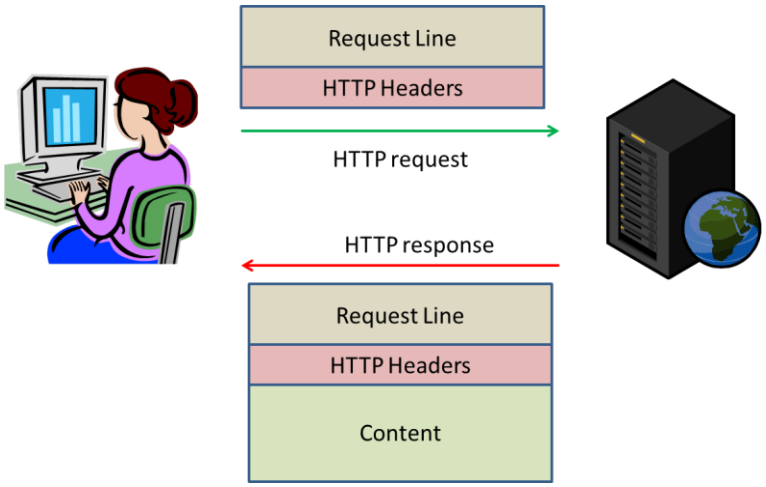
Character	Reserved usage
?	Separator indicating start of query string
&	Separator between parameters in query string
=	Separator between name and value of parameter
,	Separator between individual values in list-oriented parameters (such as BBOX, LAYERS and STYLES in the GetMap request)
+	Shorthand representation for a space character

Source: OGC, 2005

3.4. Basic Caching Techniques

The HTTP is an application level protocol and defines the communication between clients and servers (see Figure 5). 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 (RFC 2616, 1999).

Figure 5: Principle of HTTP Communication



Source: Loechel and Schmid, 2012

The abstract of the HTTP standard describes some of the main attributes of the protocol:

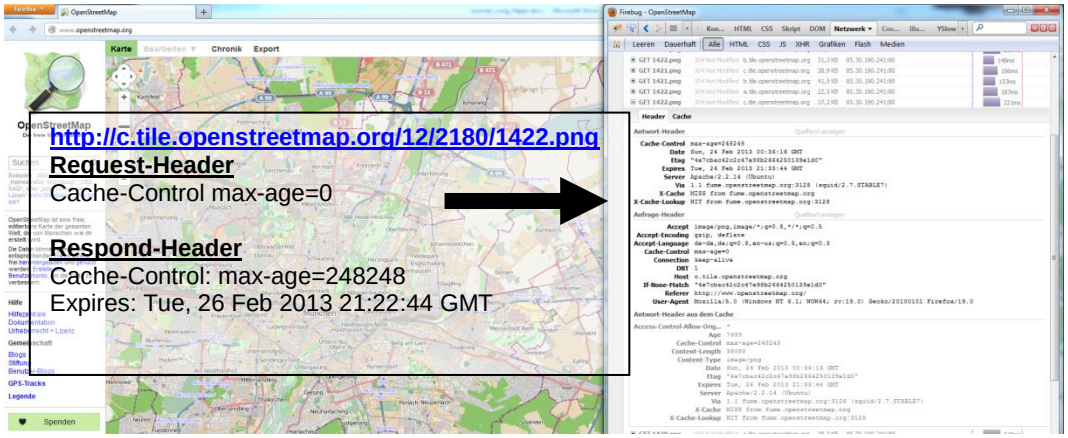
- generic - not bound to a special content type (e.g. HTML, text, images) or use-case;
- stateless - a GET request should always deliver the same response.

The HTTP protocol defines ten request methods (e.g. GET, POST, PUT, TRACE). For common work with web services the request methods GET and POST make up more than 95 % of all used HTTP-methods.

- GET requests a document from the server specified by the Uniform Resource Identifiers (URI).
- POST requests are usually a client response which were performed on huge data transfer or on submitting by a data form.

HTTP is defined as stateless, any similar request should result in the same response. Therefore any response to a request can be cached, but caches usually ignore all POST-requests and any GET-request that contain a "?" (query-call) in the URI. Browsers neither expect them to be cacheable nor send HTTP headers for cache requests (max-age=0; see Figure 6). The OGC specifies their web services based on those GET-requests queries, so an OGC Web Services (OWS) request would usually not be cached. But the HTTP standard contains several techniques which allow shared caches and browsers to cache content (static & dynamic; RFC 2616, 1999).

Figure 6: Relevant Cache-Header on a Map Tile Request



Source: Adopted after OSM, 2012

3.5. 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 parameters. The amount of query parameters 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 256x256 pixels with pre-defined bounding-boxes and different scales, which are not reflected in the WMS GetMap-request. Those two parameters 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 of a WMS GetMap request looks like the following, where the bold variables are important for caching:
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

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

In the larger web technology community generic HTTP-caches are more common. Both approaches have a different focus and limitations for a web project. Tile caches seed all map-tiles during initialization and then transfer tiles to the client. 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 is 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 bound to the transfer-bandwidth. The use of a HTTP-cache which is directly connected to the map-server can reduce this problem. It always uses the maximum possible bandwidth even if cache and mapserver 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 all generic HTTP-caches, is that changes on the underlying data are not recognized by the caching server. This problem is known as the purge problem.

The focus of this article is the comparison of different caching techniques for a WMS-Server setup. Therefore a discussion of both fundamental ideas shows why different techniques are suitable and may vary in the results.

WMTS is a specialized caching standard for WMS Tiles. As described above WMTS seeds the tiles on startup and may regenerate all tiles on scheduled events. Generic HTTP-caching does not know anything about the underlying system, it only caches requested resources, and revalidates them on request level.

Choosing one of those ideas depends on the use-case. For the context of the above described military situational awareness system, a caching technique that is quick in response, can handle different layers independent and caching times within small minute ranges. The amount of data that is necessary for a full tactical map of 100 x 100 square kilometer in tiled SVG-images, with several zoom levels is within a 1 GB range and can easily be stored on a mobile device or cache-server. The scenario does not compare a full to a partial caching solution, both approaches can handle full caching of tiles. In this case, the advantage of generic caching techniques depends on the very quick invalidation of the cached tiles and that those are only regenerated on request level. The overall server load is minimized without losing the advantage of caching.

3.6. HTTP Cache Header

The HTTP protocol version 1.1 defines several HTTP header statements. For caching only a few are sent as part of the HTTP request (client side values) as well as HTTP response (server side values) (RFC 2616, 1999).The article describes some methods for cache manipulation:

- Expires-header: The Expires header is a response only header, that indicates by a RFC 1123 timestamps, when the response should be expected as not fresh (Braden,1989)
- 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 present the current cache entry, 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.

Table 2 defines the Cache-Control values on client side, which allows the browser to inform the server what kind of caching constraints, are desired. A server should take into account the clients constraints and create its response with similar constraints. A server can supply additional or different caching options, which are defined as Cache-Control-Values in Table 3.

Table 2: Possible Values for Cache-Control on Client Requests

Cache-Control-Values	Description
`no-cache`	Indicates that the client requests a non-cached response from the origin server. A cache proxy has to revalidate its stored cached entity with the origin server.
`no-store`	The client indicates that no copy of the response should be stored by any cache.
`max-age=seconds`	Indicates that the client is willing to accept a response whose age is not higher than the specified time in seconds. A max-age=0 indicates that caching is not willing to accept and force a check with the origin server (end-to-end revalidation).
`max-stale[=seconds]`	Indicates that the client is willing to accept a response that has exceeded its expiration time.
`min-fresh=seconds`	Indicates that the client is willing to accept a response whose freshness lifetime is not less than its current age plus the specified time in seconds. That is, the client requests a response that will still be fresh for at least the specified number of seconds.
`only-if-cached`	A very rare used header for extremely poor network connectivity purposes, that indicates that the client is willing to receive the response only if the cache already holds a copy of the response.

Source: Kersken, 2008

Table 3: Possible Values for Cache-Control on Server Response

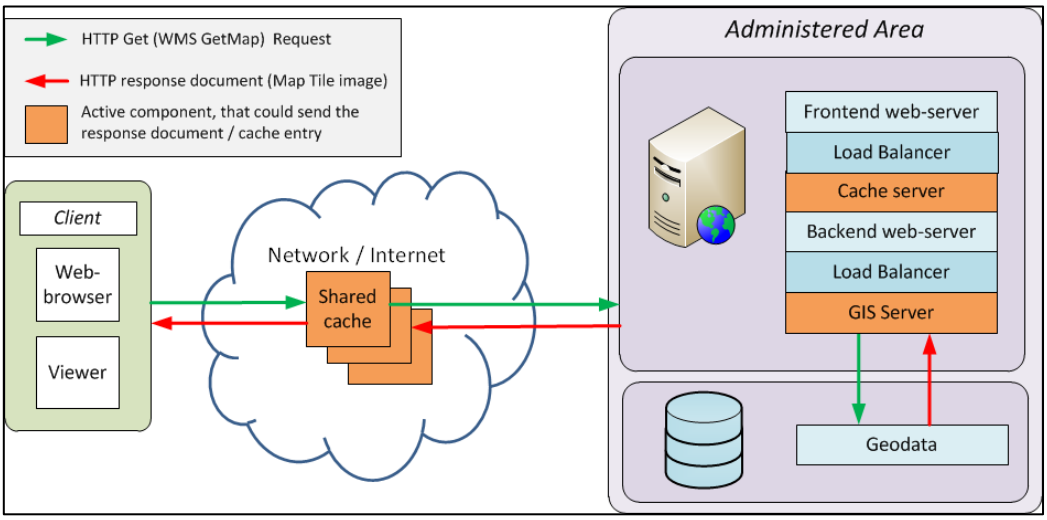
Cache-Control-Values	Description
`public`	Indicates that the response may be cached by any cache, even if it would normally be non-cacheable or cacheable only within a non-shared cache.
`private`	Indicates that the response message is intended for a single user and MUST NOT be cached by a shared cache. A client MAY cache the response.
`no-cache`	A cache MUST NOT use the response to satisfy a subsequent request without successful revalidation with the origin server. This allows an origin server to prevent caching even by caches that have been configured to return stale responses to client requests.
`no-store`	A cache MUST NOT store any part of this response. This directive applies to both non-shared and shared caches, but a client is able to explicitly store the response outside a caching system.
`no-transform`	A cache is not allowed to perform any transformation on response body and header fields (e.g. convert between image formats, due to reduce cache size).
`must-revalidate`	A cache must respect expiration time and max-age directive and is not allowed to use a stale entry.
`proxy-revalidate`	Same as must-revalidate, but only for shared caches. Client caches can ignore this directive.
`max-age=seconds`	A cache can take the response as fresh for the defined time, without revalidation.
`s-maxage=seconds`	If a response includes a s-max-age directive, then for a shared cache (but not for a private cache), the maximum age specified by this directive overrides the maximum age specified by either the max-age directive or the Expires header.

Source: Kersken, 2008

Figure 7 shows several stages of caches, marked with orange background-colour. Those shared caches can only hold and serve 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. Modifying request-headers violates the HTTP-standard, but might be necessary to use specific cache solutions.

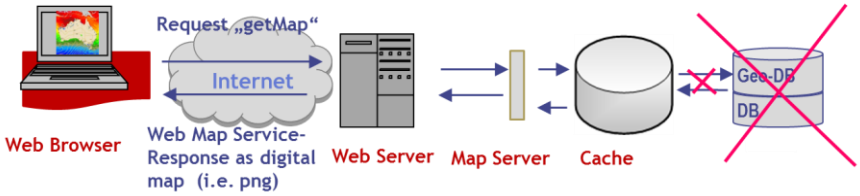
Figure 7: Schema of WMS GetMap Request - Response Communication Path and Managing Areas



Source: Loechel and Schmid, 2012

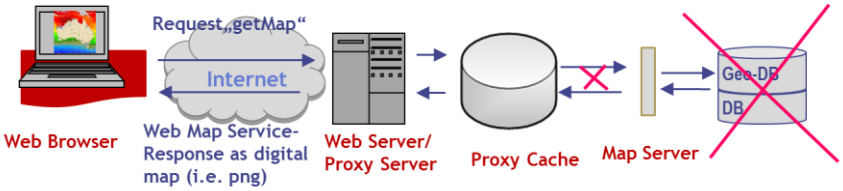
The frontend and backend server might be used to set, modify or delete specific headers especially the Cache-Control-Values no-cache, max-age on request and respond, depending on capabilities of cache servers. Typical modifications are to delete request header Cache-Control: max-age=0 to serve cached map tiles. Backend modifications between cache server and map server are only necessary if the map server is not capable to set cache headers on its own. Figure 8 and Figure 9 show where down-stream communication in request process is interrupted and where the map tile is served from.

Figure 8: Caching Chain with a Tile Cache Instance



Source: Loechel and Schmid, 2012

Figure 9: Caching Chain with a Generic HTTP Proxy Cache Instance



Source: Loechel and Schmid, 2012

How this manipulation can be implemented is out of scope of this article but further information can be found in the specific documentations of used web-servers or cache-proxies. For Apache Web Server the mod_header extensions and its directives are recommended.

3.7. 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.

The purge effect leads to the problem, that a cache or map-server might respond outdated maps even if the underlying geo data base has new or modified features. 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. Dynamically changing data: Public map layers with up to date information.

Both cases differ in the way how to handle this problem and it depends on the system stack. Normally non-cached map servers always render the current available data. Caches try to serve their cache entry. Standard tile cache implementations ignore the HTTP standard and will always serve the seeded tiles.

If no further manipulations of cache-headers have occurred, shared caches will not hold any cache-entries itself.

4. EXPERIMENTS

It is possible to cache WMS-requests; there is the need to know the advantages and disadvantages of the different caching techniques. In a productive environment it is essential to know the performance values of different caching techniques to balance peak load in an adequate way.

4.1. Experiment Setup

4.1.1. Caching Software

A WMS in combination with a modern AJAX client can produce a compelling geo web application using WMS tiles. For each tile, the server has to fetch the required data from various databases, fetch the rendering parameters (which might be cached), possibly perform a projection transformation, and then finally render a graphic from this information (raster image). The following section introduces caching software used for the performance tests.

4.1.2. Tile Caching Software

The easiest way to improve the WMS performance for the end users is to implement a tile cache. GeoWebCache is an open source tile caching service middleware developed by OpenGeo. It supports different OGC standards, including WMTS, WMS-C, KML. By its tiling on demand strategy and OGC WMS seamless integration it has a good prospect to share spatial information. GeoWebCache is implemented using Java and distributed under the GNU General Public Licence (GPL; OpenGeo, 2013).

4.1.3. Generic HTTP Web Caching Services and WMS

From a simple web traffic analytic point of view WMS requests are still HTTP requests. For caching HTTP there are a lot of approaches and developed caching systems. In this paper three commonly used caching solutions will be introduced and used in the test setup for performance benchmarking.

Squid is a well-known proxy server that can be used for caching. It was developed by Duane Wessels of the University of Colorado for the "Harvest object cache" project. A large scale project that uses Squid is Wikipedia (Wikipedia, 2012). This shows how scalable and performance driven a web project can be. The Squid proxy server is a de facto standard for large caching setup. The Squid project has established several common standards for caching clusters, like the Internet Cache Protocol (ICP) and is available for all major platforms. It is published under GPL. The configuration of a Squid proxy is very

complex but it is capable to perform a lot of additional functionality (Dithardt, 2006).

Varnish is a lightweight HTTP application accelerator designed for content heavy dynamic web sites. It acts like other proxy servers that are used for caching. It is much easier to configure than Squid, but only available for Unix/Linux systems. Varnish is open-source and available under BSD license. Varnish was released in 2006, developed by Paul-H. Kamp, a known FreeBSD core developer (Varnish, 2010)

Apache mod_cache is an extension module to the Apache httpd web-server. It implements an HTTP compliant content cache that can be used to cache either local or proxied content. Apache mod_cache requires the services of one or more storage management modules. Two storage management modules are included in the base Apache distribution, so the administrator can choose between disk storage and memory storage. Apache mod_cache is not able to cache content with access protection (King, 2008).

4.2. Performance Measurement Software

In order to obtain reliable performance values, the measurements must be repeatable and traceable. The following benchmark tools are often used for measuring the performance of web servers:

Apache Benchmark (ab) is a command-line benchmarking tool and part of the Apache httpd web server project. Today it is still the de facto standard for benchmarking HTTP requests on a web server. It is a very simple benchmarking tool that analyses the HTTP headers and summarizes information about performance factors, requests per second, mean time per request, transfer rate, failed requests etc. A limitation is that Apache Benchmark is only capable to perform tests on a single resource URI (Apache Foundation, 2013).

Apache JMeter is a GUI based benchmarking tool. It is part of the Apache Jakarta Project, hence it is open source. Apache JMeter is a 100 % Java desktop application and is highly extensible through its provided API. Apache JMeter is used to test the performance both on static and dynamic resources. It can be used to simulate a heavy load, using multiple threads, on the server or on the network (Halili, 2008).

These tools are able to test and measure HTTP request response times. This is the time taken from sending the request until the HTTP response document is loaded. They do not intend to analyze, or render a web page from the HTTP response document. A WMS service presenting implementation in web pages is typically made with JavaScript frameworks like OpenLayer, but requesting the

HTML webpage loads only a HTML document but not the sub requested resources. Therefore a measurement of Web Map Service performance must be made on request level.

4.3. Testsetup/Testbed

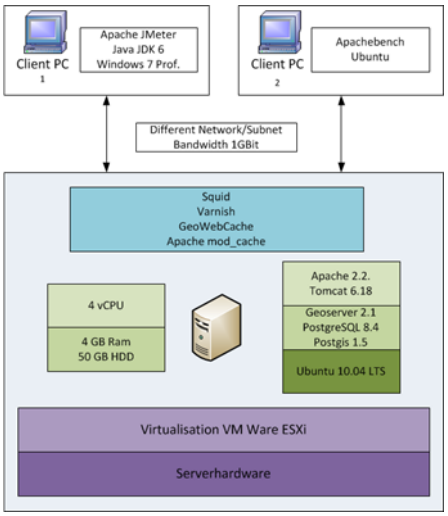
The following section describes the hard/software specifications of the test system, preparation of test data and test procedures for performing the benchmark tests.

For the benchmark test, the testbed of the University of the Bundeswehr Munich is used. A testbed is a platform for technical and scientific experiments. With a testbed different software can be evaluated in a safe and secure environment. For the evaluation of GeoWebServices it is necessary to develop general approaches (e.g. usability). Appropriate test procedures and test criteria need to be devolved. The main goal is to increase the interoperability within the service-oriented architecture. The testbed is based on a three layer model (Loechel and Schmid, 2012):

- Layer 1: The testbed is based on an high performance server hardware;
- Layer 2: Virtualization (VMWare ESXi) allows different test systems to be deployed easily, quickly and cost effectively;
- Layer 3: Due to the provision of geospatial data as web services, various clients can connect to the test bed and perform tests with different systems.

Figure 10 shows the hard and software components of the testbed.

Figure 10: Testbed with Test Arrangement



Source: Loechel and Schmid, 2012

The test system for the benchmarks is established according to a common service-oriented architecture for production use. For Squid proxy the refresh pattern for HTTP Get-query-requests have to be changed.

4.4. Test Data

For the tests open-source data from OpenStreetMap is used (OSM, 2012). Data are extracted from the planet.osm file and imported into three database tables with different geometry types (point, line and polygon). Data of each type is limited to 6500 features. In total four layers for testing have been allocated, which are provided by GeoServer: one layer for each geometry type and a group layer containing all three geometry types.

4.5. Test Procedure

The configuration of the different cache software follows their standard documentation for HTTP acceleration/caching with binding to the WMS server GeoServer (OpenGeo, 2013). GeoServer's general cache headers are set for each layer.

For Squid proxy the refresh pattern for HTTP get-query-requests has to be changed.

- 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: In order to have reference values, a static version of a disk stored map tile image was included in the performance test.
- Tile Cache: GeoWebCache represents a common tile caching software. GeoWebCache produces tiles with a size of 256x256 pixel. Tiles were seeded after setting up the Web Map Service.
- Proxy Cache: Apache mod cache represents a traditional configuration as proxy server.
- Web Acceleration: Varnish represents a special caching system on a single machine usable only on Unix/Linux operating systems, a so called application accelerator.
- Proxy Cache: Squid represents common a 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.

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 256x256 pixel, the typical WMS tile size.
2. The second run requests an image with a file size of 800x600 pixel. This is the example image size of the INSPIRE directive.

For the tests with GeoWebCache 12 tiles with a size of 256x256 pixel each were requested. 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 AND DISCUSSION

This section analyses and discusses the results of the performance and caching tests.

Table 1 reflects the determined performance level of each system as a result of the Apache Benchmark test runs which can be summarized as follows: Caching systems increase the ability to handle more requests per second. The average requests per second increased from 3316 r/s compared to 50 r/s by Geoserver itself for the 256x256 pixel PNG. For the 800x600 pixel PNG the average requests per second increased from 1114 r/s compared to 25 r/s by Geoserver itself. This is an increase by the factor of 40-70 and thus this 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.

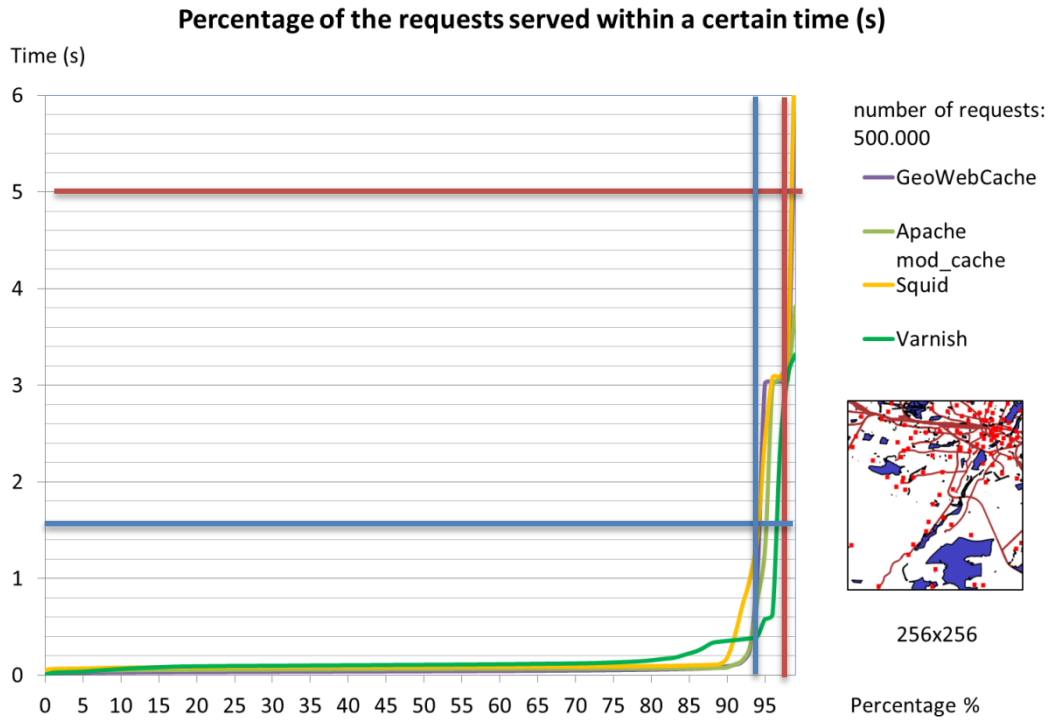
Table 1: Results of Apache Benchmark

Cache System	256x256 pixel - 30 KB PNG WMS response		800x600 pixel - 105 KB PNG WMS response	
	Request/sec	Time/request	Request/sec	Time/request
Non-cached Geoserver	50	21000	25	41000
GeoWebCache	3700	310	Unable to perform such a request	
Apache tile static served	3700	310	1070	930
Apache with mod_cache	3600	290	1050	930
Varnish	3700	270	1050	920
Squid 3.0	2550	390	1000	1000
Squid 2.7	2650	380	950	1050

Source: Loechel and Schmid, 2012

In both test cases, the tests with the static file performs very fast, with a quality of service of 90 % within 65 ms. The very small differences of the systems GeoWebCache, Apache_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 expected. The configuration of Squid proxy is far more complex than for every other cache system, a non-optimal configuration of the Squid test systems must be suspected (red line in Figure 11). Squid 3.0 is relatively faster in serving larger file sizes compared to other caching systems, but compared to Squid 2.7, it is slower. A reason for this behavior might be the Squid reimplementation.

Figure 11: Comparison of Software for 256x256 px PNG

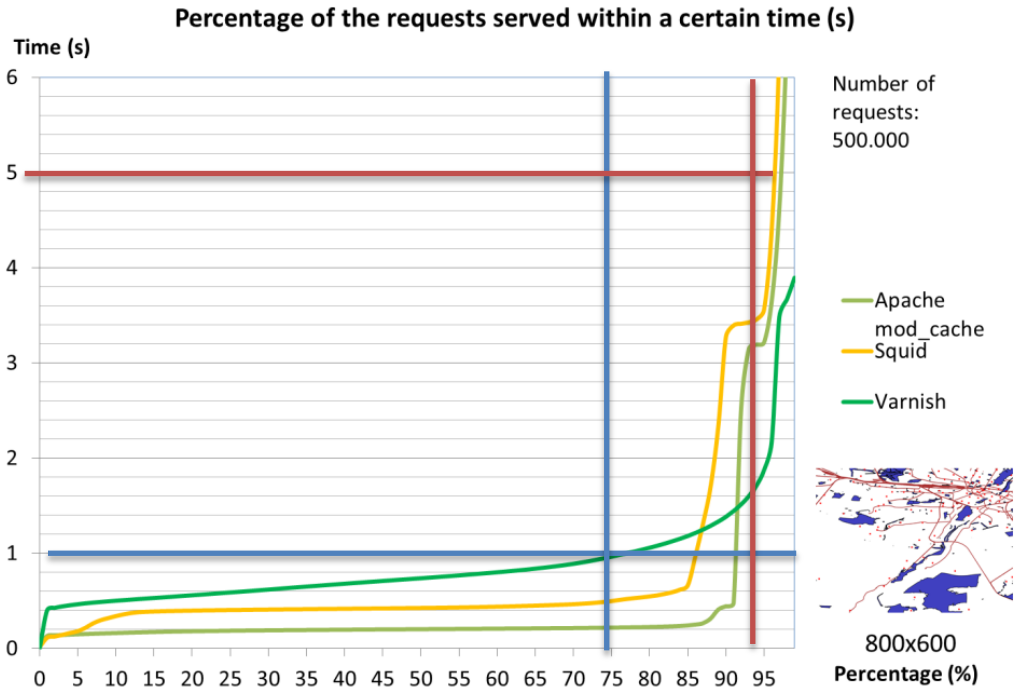


Source: Loechel and Schmid, 2012

GeoWebCache transfers only tiles of a specific size. Figure 11 (blue line) show that the test server fulfills the INSPIRE directive of 93% 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.

For bigger PNGs the tests show equivalent results. GeoWebCache is not able to transfer tiles of the defined image size. All other caching software fulfill the INSPIRE directive within 96 % of the requests (red line in Figure 12) within five seconds. 73% off all requests can be handled within 1 second (blue line in Figure 12). Further the results show that the time limit stated by Nah (2004) of 2 seconds can easily be handled for 80% of the requests.

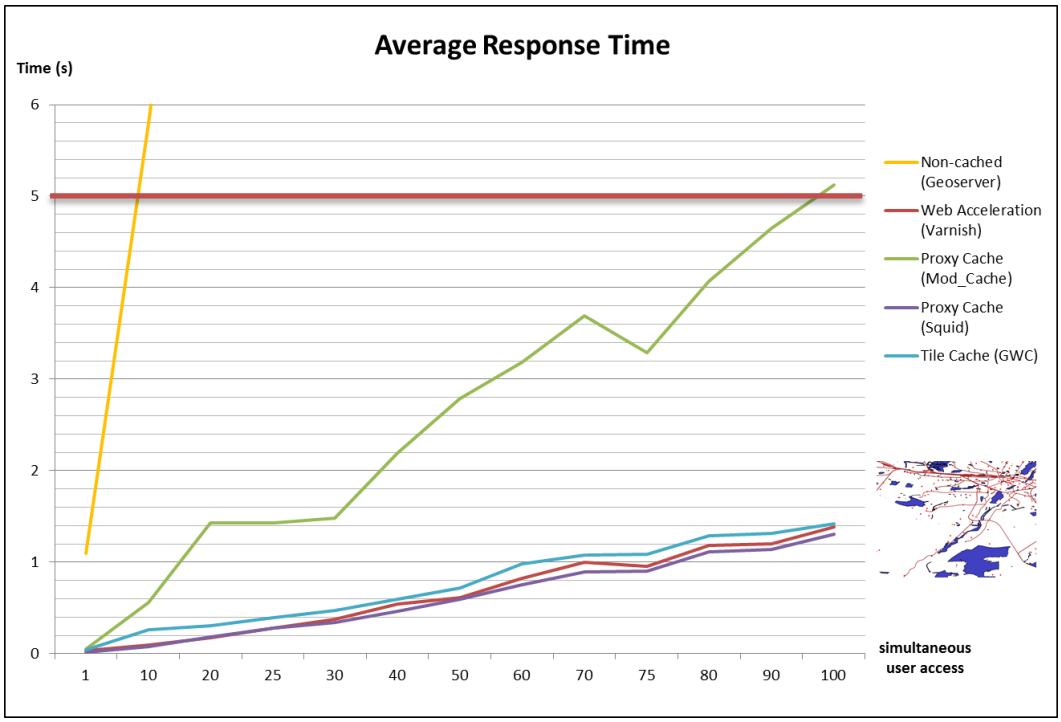
Figure 12: Comparison of Software for 800x600 px PNG



Source: Loechel and Schmid, 2012

The JMeter results (Figure 13) 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.

Figure 13: Response Time for the Grouplayer



Source: Loechel and Schmid, 2012

The comparison of cache server results and the static served version of the tile show 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 increase of necessary time per request over 90 % which is based on the transfer stack of the ISO-OSI reference model (Zimmermann, 1981). The package collision on the network layer decreases the possibility to transfer packages successfully. HTTP is a layer 7 protocol and therefore not aware of connectivity problems in the transport stack. So a higher bandwidth might increase the number of requests handled per second.

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: WMTS (King, 2008). Tile caching systems are fast in providing the tiles. Tiles can be generated in advance so that they just need to be served.

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, the failure rate decreases. This results in 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 increase dramatically.

6. CONCLUSION AND FURTHER WORK

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 timeliness of their local cache entry. Caching is most efficient on handling a huge amount of concurrent request, which call the same resource. A cache will respond its cache entry and thereby reduce network traffic to the generating system and its server load. A common limitation of caching systems is a consequence of the purge problem. It is necessary that caching software recognizes data changes in an appropriate time. Another limitation of generic HTTP caching systems comparing to specialized tile caches is that they do not know anything about the specific protocol or data conventions. A generic HTTP caching system cannot seed or pre-fetch any tiles, content is cached just in time. Spatial tile caching systems can generate tiles for the complete bounding box in advance.

We strongly recommend all GIS project managers to make themself aware of caching techniques and how the cache-headers work. WMS projects, which are using caching techniques and therefore process requests faster, might increase the acceptance of GIS technologies.

Further studies have to analyze the edges where this setup reaches its 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 analyze, how prefetching algorithms of spatial caches could be separated and used in generic HTTP caches.

REFERENCES

Apache Foundation (2013). *ab - Apache HTTP server benchmarking tool*, at http://httpd.apache.org/docs/2.2/programs/ab.html [accessed 26 February 2013].

Braden, R. (1989). *Requirements for Internet Hosts Application and Support*, at http://www.rfc-base.org/txt/rfc-1123.txt [accessed 26 February 2013]

DGIWG – Defence Geospatial Information Working Group (2008). *DGIWG Standards and Implementation Profiles,* at http://www.dgiwg.org/dgiwg/htm/documents/standards_implementation_profiles.htm [accessed 26 February 2013].

Dithardt, D. (2006). *Squid: Administrationshandbuch zum Proxyserver*, Dpunkt Verlag.

European Commission (2013). *INSPIRE*, at http://inspire.jrc.ec.europa.eu/ [accessed 26 February 2013].

European Commission (2007). INSPIRE *Network Services Performance Guidelines*, at http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/Network_Services_Performance_Guidelines_%20v1.0.pdf [accessed 26 February 2013].

ESRI (2010). *ArcGIS desktop help*, at http://resources.arcgis.com/ [accessed 26 February 2013].

Halili, E. (2008). *Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites*, Birmingham: Packt Publishing Limited.

Jay, D. (2012). *What does an earthquake look like*, at http://de.slideshare.net/djay/surviving-earthquake-15132841 [accessed 26 February 2013].

Kersken, S. (2008). *Apache 2.2*, Galileo Press GmbH.

King, A. (2008). *Website Optimization: Speed, Search Engine and Conversion Rate Secrets,* Cambridge: O'Reilly Media, Inc.

Krupp, B. (2010). *Exploration of dynamic web page partitioning for increased web page delivery performance*, at https://etd.ohiolink.edu/ap:10:0::NO:10:P10_ETD_SUBID:53291 [accessed 26 February 2013].

Loechel, A. (unpublished). *Ein Open Source Ansatz für Führungsinformationssysteme;* PhD Thesis, University of the Bundeswehr Munich.

Loechel, A., Mihelcic, G. and S. Pickel (2012). "An open-source approach for a military situational awareness system", *Proceedings of the 45th Hawaii International Conference on System Sciences, Maui, HI, USA, pp. 1462-1471 Washington DC, USA: IEEE Computer Society.*

Loechel, A. and S. Schmid (2012). "Caching techniques for high-performance Web Map Services", *Proceedings of the 15th AGILE Conference on GIScience, April 24-27, 2012, Avignon, France.*

Nah, F. (2004). A study on tolerable waiting time: how long are Web users willing to wait?, *Behaviour & IT*, 23(3): 153-163.

NATO Standardization Agency (1986). *APP-6B Military Symbols for Land Based System; at* http://www.military.com/ResourcesSubmittedFiles/Military_Symbols_Guide.pdf [accessed 26 February 2013].

OpenGeo (2013). *GeoServer User Manual*, at http://docs.geoserver.org/stable/en/user/index.html [accessed 26 February 2013].

OGC – Open Geospatial Consortium, Inc. (2005). *The Open Geospatial Consortium Web Map Service (WMS) Approved as International Organization for Standardization (ISO) Standard*, at http://www.opengeospatial.org/node/436 [accessed 26 February 2013].

OGC – Open Geospatial Consortium, Inc. (2013). *About OGC*, at http://www.opengeospatial.org/ogc [accessed 26 February 2013].

OSM – Open Street Map; FOSSGIS e.V. (2012). *OpenStreetMap-Project*, at http://www.openstreetmap.org [accessed 26 February 2013].

Peterson, M. (2008*). International perspectives on maps and the Internet*, Berlin, New York: Springer.

RFC 2616 (1999). *Hypertext Transfer Protocol HTTP/1.1*, at http://www.ietf.org/rfc/rfc2616.txt [accessed 26 February 2013].

RFC 2396 (1998*). Uniform Resource Identifiers (URI),* at http://www.ietf.org/rfc/rfc2396.txt [accessed 26 February 2013].

Taktiklehrer im Reservistenverband (2010). *Image of a tactical map*, at http://www.taktiklehrer.de/bericht.html [accessed 26 February 2013] (in German).

Varnish (2010). *Varnish Cache*, at https://www.varnish-cache.org [accessed 26 February 2013].

Wikipedia Inc. (2012). *Wikipedia article: Wikipedia, section Software and hardware*, at http://en.wikipedia.org/wiki/Wikipedia#Software_and_hardware [accessed 26 February 2013].

Zimmermann, H. (1980). OSI Reference Model: The ISO Model of Architecture for Open Systems Interconnection, *IEEE Transactions on Communications*, 28(4): 425-432.