An Esri® White Paper • January 2010

Performance and Throughput Tips for ArcGIS® Server 9.3.1 Cached Map Services and the Apache HTTP Server

Esri, 380 New York St., Redlands, CA 92373-8100 USA
TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

Performance and Throughput Tips for ArcGIS Server 9.3.1 Cached Map Services and the Apache HTTP Server

An Esri White Paper

Contents **Page**

Performance and Throughput Tips for ArcGIS Server 9.3.1 Cached Map Services and the Apache HTTP Server

Executive Summary

Today, it is clear that the caching of static ArcGIS® Server maps for use in Web applications provides the fastest and most optimal throughput for delivering maps over the Web and to the enterprise. However, there are ways to take this optimization and throughput a step further by adding load balancing and tuning the Web tier for better cache handling. This document describes some simple steps for utilizing Apache HTTP Server and the ArcGIS Server REST handler to increase the performance and throughput of cached content for Web applications. It also discusses the results of some test scenarios that involved these tuning steps.

Testing Scenarios

The testing scenarios examined in this paper involve an ArcGIS Server cached map service of the city of Portland, Oregon, from the Portland dataset that comes with the sample data shipped with the ArcGIS Server Software Developer Kit (SDK). The map cache consists of 22,085 individual tiles in JPEG format, organized for eight scale levels.

The Web test involved pulling the pregenerated cache tiles from the dataset indicated above via the ArcGIS Server for Java REST handler endpoint.

The environment used for this benchmark was ArcGIS Server 9.3.1 Advanced Enterprise for the Java Platform, using the hardware configuration as described below:

The load test started with 1 thread and progressively grew to 125 concurrent threads, with a stepping increment of 5 threads every five minutes. Each of these threads pulled various map tiles from multiple levels of detail with no think time in between requests, simulating a heavy load on the server.

Testing Configuration

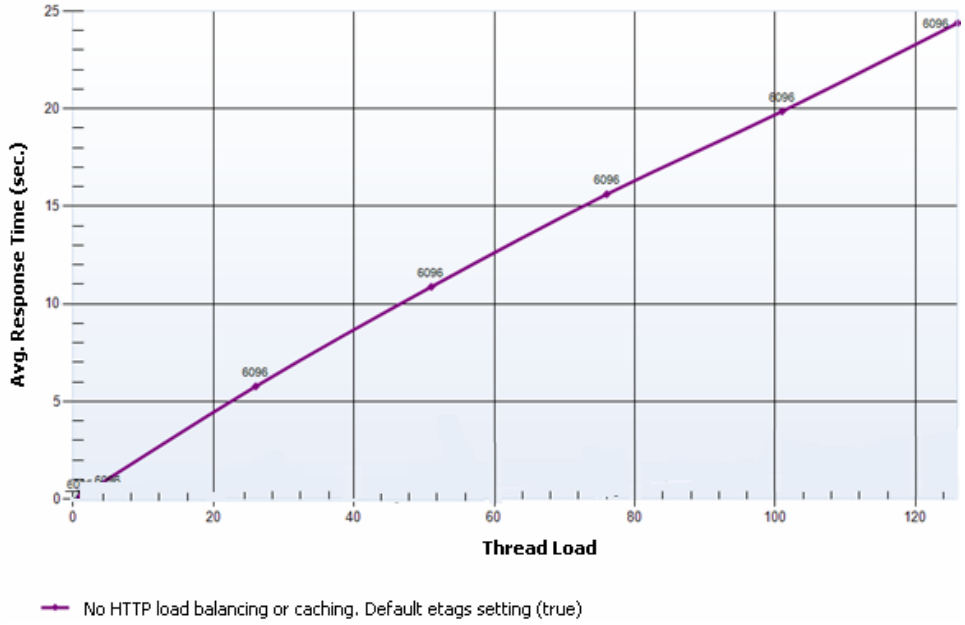
- ArcGIS Server 9.3.1 for the Java Platform
- RHEL 5.3 x 86_64
- Eight cores
- E5450 at 3.00 GHz with 16 GB of RAM
- Apache HTTP Server 2.2.11, Linux
- Internal ArcGIS Tomcat instance memory configuration set to a minimum/maximum of 1 GB

Scenario 1: Pulling Map Tiles from an Out-of-the-Box Cached Map Service

We started the test by running a stress script against an out-of-the-box ArcGIS Server for the Java Platform REST endpoint for the map service.

As seen in figure 1, we observed an increase in response times as additional load was added to the map service. Response time quickly grew beyond the threshold we would expect for a cached map service. In figure 1, the vertical axis represents the response time for map tile retrieval on a local network through the map service REST endpoint. Load was applied to the server by increasingly adding client threads. Each client thread in this exercise retrieves map tiles from the server with no think time in between requests. That is, a client thread requests a map tile, waits for the response, requests another map tile, and so on. In the scenarios analyzed in this paper, a maximum number of 135 simultaneous client threads were applied (see horizontal axis).

Figure 1
Average Response Time vs. Thread Load



Scenario 2: Switching E-tag Configuration to False

In the second scenario, we modified the default configuration of the ArcGIS Server REST endpoint slightly by switching the e-tag's value in the REST handler configuration to false.

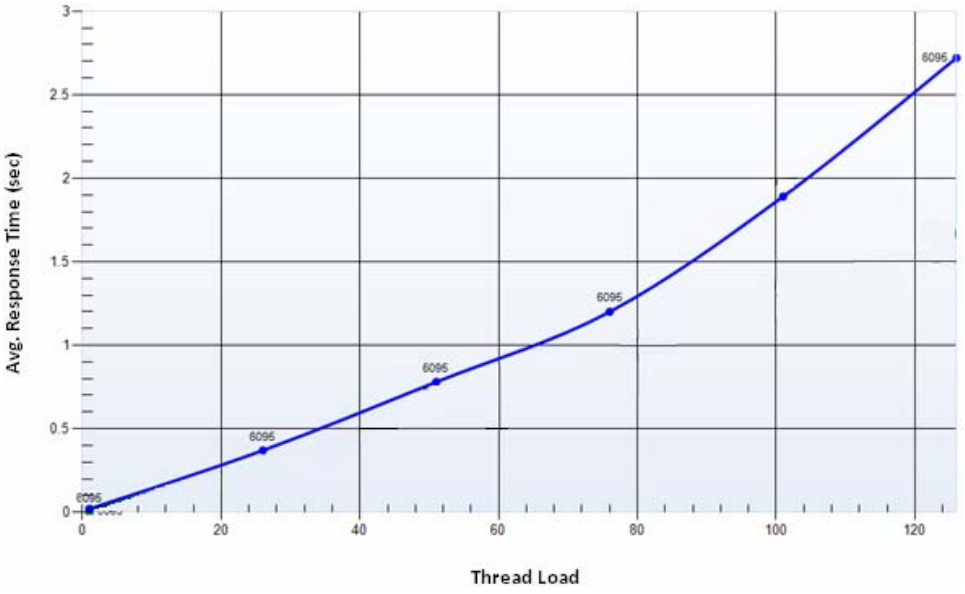
E-tags, or entity tags, are a mechanism that Web servers and browsers use to determine whether a component in the browser's cache matches one on the original server. In this case, a component is the same as an entity; this refers to things from a Web application such as images, scripts, and style sheets. E-tags are used for validating entities and are considered the most flexible way of performing last-modified updates rather than component comparisons.

At ArcGIS Server 9.3.1, the REST handler uses e-tags by default. The properties of a REST handler instance can be modified in the *rest-config.properties* file by using the setting *etags=false*. The rest-config.properties is located in `<ArcGIS_rest_home>/WEB-INF/classes/resources`. To change the etag setting, open this file and add the following line below the "#cache config" entry:

```
config.use-tile-etags=false
```

After applying this change, we observed dramatically decreased response times compared to the results of scenario 1, as shown in figure 2.

Figure 2
Average Response Time vs. Thread Load, with "etags=false"



Switching the *etags* parameter as described above is a workaround that 9.3.1 users can apply to improve performance on cached map services in ArcGIS Server for the Java Platform.

Scenario 3: Incorporating HTTP Server Load Balancing and HTTP Caching on Top of Scenario 2

In our last scenario, we incorporated Apache's HTTP server load balancing and HTTP caching on top of changes described in scenario 2 for the purpose of improving our cached map service even further. Refer to appendix A in this document for details on the changes applied to the Apache Web server configuration.

Load balancing and caching with Apache HTTP server can be an effective way to speed up the response times of dynamic requests, cached tile requests, and certain ArcGIS applications while simultaneously load balancing HTTP traffic. By *load balancing*, we are referring to the use of a load balancer in front of the Web server, which is also known as a reverse proxy or a gateway. Apache HTTP server can be deployed on a Windows, Linux, or Solaris operating system and can be on separate, enterprise- or non-enterprise-grade hardware, or with the ArcGIS Server Web tier.

Load balancing caching with Apache works with the help of several modules. These modules are extensions to the Apache HTTP server. The bulk of the balancing logic is performed via *proxy_module, proxy_balancer_module,* and *proxy_http_module.* Caching is performed via the *mod_cache* and *mod_disk_cache* modules. Collectively, they listen on Apache's port (usually 80) for requests to a predefined back end server or group of servers.

The first time a request is made from a client application, the load balancer determines which of those servers is available and capable of handling the request. Once a server generates the response, it is forwarded back to the load balancer where it is saved to the cache store on disk. Then, the stored cached response is sent back to the client. The next time that same request enters the load balancer, it can be quickly retrieved from its cache, avoiding the network trip of traversing the entire software stack. The functionality of storage and retrieval of files from the cache is through Uniform Resource Identifier (URI)-based keys.

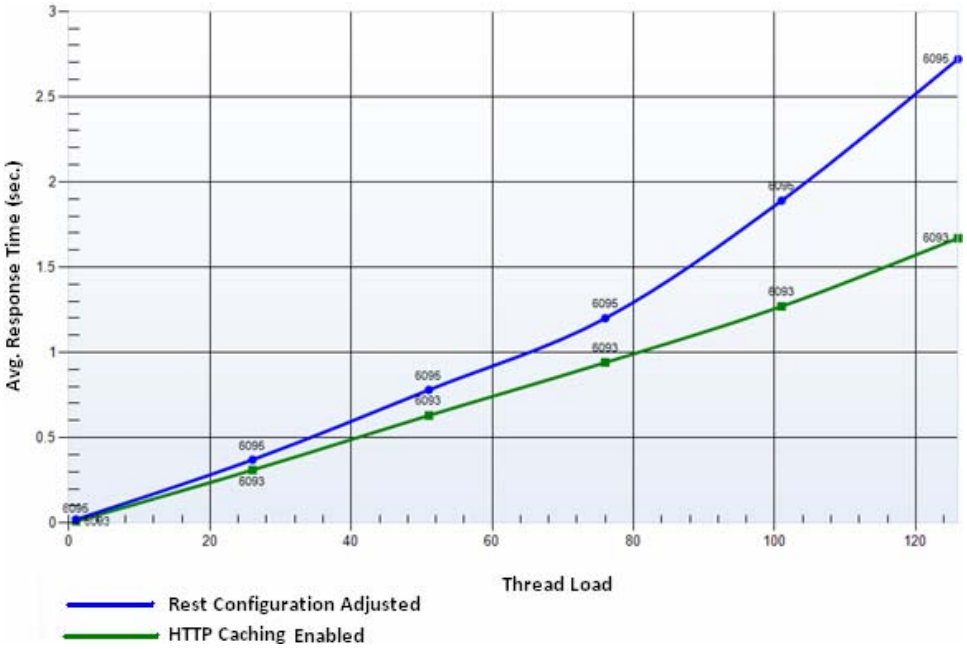
The flexible nature of Apache load balancer caching enables it to cache items that it receives from back end servers. This can offer a speed-up in Java applications, and it off-loads the serving of static content from the Java runtime environment (JRE).

In general, back end servers do not need to know about the load balancers. From a security standpoint, clients should not know about the servers behind the load balancer either. A load balancer can help provide a transparent link between both ends. Because of this, clients no longer enter the ArcGIS Server container through the Web tier universal resource locator (URL) but instead enter through the load balancer URL.

Results from this configuration are shown in figure 3. We observed a slight improvement in overall maximum map throughput and response times over what was observed in scenario 2.

J-9822

Figure 3
Average Response Time vs. Thread Load



Summary and Conclusion

By taking advantage of Apache HTTP Server caching and load balancing capabilities, in conjunction with ArcGIS Server software's REST handler properties modifications for turning off the use of e-tags, throughput and average response times for ArcGIS Server client applications can be dramatically improved.

Note that the Apache HTTP server configuration described in this document may not yield the same results for every situation. This configuration is just one of many. One size does not fit all, but the options here can be used as a starting point. With additional tuning and adjusting, parameters may be found that are better suited for a particular environment.

It is also worth noting that the technique described in this document can be applied not only to cached map services but also to dynamic map services.

Frequently Asked Questions

How will I know if my Apache Load Balancer is balancing requests?

The Apache configuration allows control over the verbosity of the error log.

By default, it is set to "warn," but when set to "debug" (followed by a restart of the Apache HTTP server), you can obtain a wealth of information about the logic the load balancer performs on every request sent to it. When logging is set to debug, the start of a request is marked with the phrase `proxy: BALANCER: canonicalising URL`, while the phrase `proxy: HTTP: has released connection` marks the end of a request. Each request will contain about 14 lines of logic that Apache performed. To see which back-end machine was chosen to serve the requested item, look for the phrase `proxy: by requests selected worker [HOSTNAME]`, where `[HOSTNAME]` is the name of the ArcGIS Server. If you examine this entry for each request, you should find that Apache is cycling through the back-end servers that were defined in the configuration and is balancing the load.

How will I know if my HTTP caching is working?

If you are using HTTP caching, there are three things that you may observe to verify that your configuration is working:

- A significant decrease occurs in the response times of repeated requests, for example, the same resources (CSS style sheets, images, etc.) of a Web application, the same map tiles from a map service cache, or the same map request to dynamic map services.
- The HTTP server's cache store becomes populated.
  - Look in the cache directory for alphanumeric folders. These will contain the cache (and cache headers).
- Significant decrease occurs in server object container (SOC) activity (assuming the same area of interest is used).

How large will my cache store grow?

HTTP Web server cache sizes and growth limits depend entirely on what you are trying to cache. The limit is theoretically bound by hardware resource limits. However, it is important to acknowledge that a huge cache is not necessarily good. If you find your files are out of date (despite quick expiry configuration) and you need to immediately delete a 500 GB cache of many small files, the system performance could be negatively impacted as the disks in the machine churn for several hours. Since there is no limit to set in the *httpd.conf* file, you will need the assistance of the *htcacheclean* program (included with the Apache installation). The *htcacheclean* program runs manually or in daemon mode to keep the cache storage within a certain size.

Why should I use disk cache? Isn't memory cache faster?

Theoretically, yes, but Apache states that for most cases, *mod_disk_cache* is the preferred choice.

J-9822

From a security standpoint, will authenticated items be cached?

Please refer to the resources area for links to Apache's Caching Guide. That document will detail what can be cached and what should not be cached. You can force a virtual path to not be cached by the Apache load balancer using the CacheDisable directive. However, Apache states that headers with access protection or resources requiring authorization will never be cached. This is a good thing.

Can OGC WMS, WCS, or WFS benefit from this type of caching?

Yes. You can cache Web Map Service (WMS), Web Coverage Service (WCS), and/or Web Feature Service (WFS) requests with *mod_cache*. With the assistance of the *mod_headers* Apache module, you can instruct your cache to store Open Geospatial Consortium, Inc. (OGC), requests like WMS, WCS, and/or WFS. The following was inserted into the httpd.conf file between the mod_cache and mod_proxy configurations:

```
<Location /arcgis/services/Portland/MapServer>
Header set Cache-Control "max-age=300"
</Location>
```

These lines change the ability of the request to be cacheable by setting a customized header. The Apache load balancer then takes advantage of this and stores the item in its cache. In this case, the item is any URL with the string `/arcgis/services/Portland/MapServer`.

Can I compress the cache?

There is an Apache module for compressing server data to improve bandwidth, response time, and throughput. Unfortunately, due to the nature of the *mod_deflate* compression module, there are cases where it cannot be used effectively with the *mod_cache* module. The result is the creation of the same file many times in the cache store. This would reduce the chance of a cache hit and negate the potential for the performance increase.

Additional Resources

Apache Module mod_cache
Apache Module disk_cache
Apache htcacheclean
Phil Chen's Blog entry: Some Tuning Tips for Apache mod_cache mod_disk_cache
Apache Module mod_proxy
Apache Module mod_proxy_balancer
Apache Module mod_deflate
Apache Module mod_headers
Digital Sanctuary—Apache mod_deflate and mod_cache issues

Appendix A

Apache on Windows

For Apache on Windows, the installed binaries were deployed unaltered, and the *httpd.conf* file was modified in the following manner:

- In the Dynamic Shared Object (DSO) Support section, the following modules are uncommented:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule info_module modules/mod_info.so
LoadModule status_module modules/mod_status.so
LoadModule cache_module modules/mod_cache.so
LoadModule disk_cache_module modules/mod_disk_cache.so
```

- Next, the text below was added right before the "Main" Server Configuration section.

```
<IfModule mod_cache.c>
  <IfModule mod_disk_cache.c>
  # Ensure that the user who the Apache HTTP Server runs as has
    Full Control over the cache store
  CacheRoot "C:\cache\store1"
  CacheEnable disk /
  CacheDirLevels 2
  CacheDirLength 1
  CacheMaxFileSize 1000000
  CacheMinFileSize 1
  CacheIgnoreCacheControl On
  CacheIgnoreNoLastMod On
  CacheIgnoreQueryString Off
  CacheIgnoreHeaders None
  CacheLastModifiedFactor 0.1
  CacheDefaultExpire 3600
  CacheMaxExpire 86400
  CacheStoreNoStore On
  CacheStorePrivate On
  </IfModule>
  # Disable caching for certain URLs
  # CacheDisable /someapplication/
</IfModule>
<Proxy balancer://loadbalancer.esri.com>
  # To load balance over ArcGIS Java append :8399 to
    ags1.esri.com and ags2.esri.com.
  # You can load balance over just one server and still utilize
    the benefit of the caching
  BalancerMember http://ags1.esri.com route=worker1
  loadfactor=50 keepalive=on
  BalancerMember http://ags2.esri.com route=worker2
  loadfactor=50 keepalive=on
  ProxySet lbmethod=byrequests
</Proxy>
<Location /arcgis/>
 ProxyPass balancer://loadbalancer.esri.com/arcgis/
</Location>
```

```
# The next section defines a useful application for Apache load
  balancer administration
<Location /balancer-manager>
 SetHandler balancer-manager
 Order Deny,Allow
 Deny from all
 Allow from .esri.com
</Location>
```

- Finally, the text below was added to the EnableMMAP and EnableSendfile section.

```
# Achieve better performance with Apache on Windows with the
  following
EnableMMAP off
EnableSendfile off
Win32DisableAcceptEx
```

Apache on Linux

For Apache HTTP server on Linux, the source code was recompiled and configured with specific options as follows (executed on one command line):

```
/configure
--enable-so --enable-rewrite --enable-cgi --enable-http
--enable-proxy --enable-proxy-http --enable-proxy-balancer
--enable-usertrack --enable-info --enable-cache
--enable-mem-cache --with-mpm=worker
--prefix=/usr/local/etc/httpd --enable-disk-cache
--enable-headers
```

Next, additions to the *httpd.conf* file mentioned above for Windows were made for the Linux instance as well. Right before the "Main" Server Configuration section, the following was added:

```
<IfModule mod_cache.c>
 <IfModule mod_disk_cache.c>
  # Ensure that the user who the Apache HTTP Server runs as has
    Full Control over the cache store
  CacheRoot "/cache/store1"
  CacheEnable disk /
  CacheDirLevels 2
  CacheDirLength 1
  CacheMaxFileSize 1000000
  CacheMinFileSize 1
  CacheIgnoreCacheControl On
  CacheIgnoreNoLastMod On
  CacheIgnoreQueryString Off
  CacheIgnoreHeaders None
  CacheLastModifiedFactor 0.1
  CacheDefaultExpire 3600
  CacheMaxExpire 86400
  CacheStoreNoStore On
  CacheStorePrivate On
 </IfModule>
# Disable caching for certain URLs
# CacheDisable /someapplication/
</IfModule>
```

```
<Proxy balancer://loadbalancer.esri.com>
# To load balance over other JREs use their respective port(s)
# If WebLogic were used, the hosts would be ags1.esri.com:7101
  and ags2.esri.com:7101
# You can load balance over just one server and still utilize
  the benefit of the caching
BalancerMember http://ags1.esri.com:8399 route=worker1
  loadfactor=50 keepalive=on
BalancerMember http://ags2.esri.com:8399 route=worker2
  loadfactor=50 keepalive=on
ProxySet lbmethod=byrequests
</Proxy>
<Location /arcgis/>
 ProxyPass balancer://loadbalancer.esri.com/arcgis/
</Location>
# Uncomment the next 3 lines if the rest handler was exported
  into another JRE, like WebLogic
#<Location /rest/>
# ProxyPass balancer://loadbalancer.esri.com/rest/
#</Location>
# Uncomment the next 3 lines if the service handler was
  exported into another JRE, like WebLogic
#<Location /services/>
# ProxyPass balancer://loadbalancer.esri.com/services/
#</Location>
# Uncomment the next 3 lines if "myapplication" is an app that
  needs to be load balanced
#<Location /myapplication/>
# ProxyPass balancer://loadbalancer.esri.com/myapplication/
#</Location>
# Next section defines a useful application for Apache load
  balancer administration
<Location /balancer-manager>
 SetHandler balancer-manager
 Order Deny,Allow
 Deny from all
 Allow from .esri.com
</Location>
```

About Esri

Since 1969, Esri has been helping organizations map and model our world. Esri's GIS software tools and methodologies enable these organizations to effectively analyze and manage their geographic information and make better decisions. They are supported by our experienced and knowledgeable staff and extensive network of business partners and international distributors.

A full-service GIS company, Esri supports the implementation of GIS technology on desktops, servers, online services, and mobile devices. These GIS solutions are flexible, customizable, and easy to use.

Our Focus

Esri software is used by hundreds of thousands of organizations that apply GIS to solve problems and make our world a better place to live. We pay close attention to our users to ensure they have the best tools possible to accomplish their missions. A comprehensive suite of training options offered worldwide helps our users fully leverage their GIS applications.

Esri is a socially conscious business, actively supporting organizations involved in education, conservation, sustainable development, and humanitarian affairs.

Contact Esri

1-800-GIS-XPRT (1-800-447-9778)
Phone: 909-793-2853
Fax: 909-793-5953
info@esri.com
www.esri.com

Offices worldwide
www.esri.com/locations

380 New York Street
Redlands, CA 92373-8100 USA

Printed in USA