

Rod D. Kuhns

ArcIMS Web Site Performance Measurement and Tuning

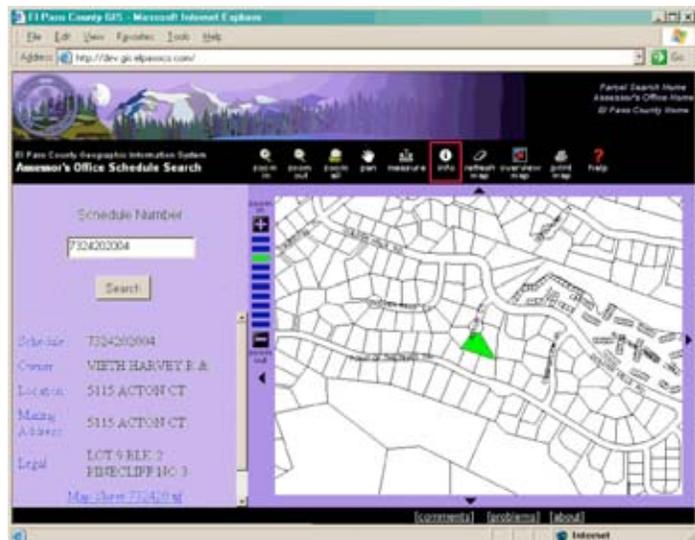
Performance tuning can be performed on many aspects of an ArcIMS web site to increase its performance. The results of extensive testing to determine precise performance bottlenecks in a typical ArcIMS architecture will be presented from a Case Study performed by [Kuhns & Associates](#).

This paper outlines tools and techniques used to design and automate a set of repeatable performance benchmark tests for an ArcIMS site using the Microsoft Web Application Stress Tool (available via download or [email request](#)). Both packet size and time can be measured throughout the many requests that pass to/from the web server and the end user browser in a typical user interaction session with an ArcIMS web site. By establishing a set standard benchmark tests, the impacts of tuning various components (SDE, Web Server, Network, Server side vs. Client side scripts, ArcIMS settings, etc.) can be precisely measured.

Background

A popular beginning to most ArcIMS implementations starts with utilizing or customizing the “out of the box” HTML viewer provided with ArcIMS. The HTML viewer utilizes client-side JavaScript extensively to process user requests, and to both formulate XML and parse XML used for interaction with the ArcIMS Server.

In 2001, Kuhns & Associates was contracted by the El Paso County, Colorado, Information Technologies Department. The contract was to develop an Assessors Parcel Search web application in ArcIMS. The ArcIMS HTML Viewer was selected to serve as the basis for the site implementation, with customization efforts targeted toward developing custom look and feel (branding) as well as to trim down the generic viewer functionality and provide only the essential tools.



The site was implemented using ArcIMS 3.1, Windows NT, on a server class machine with two 1 GHz processors, 1.3 Gb of RAM and 36 Gb of disk storage. Spatial data for the site is served by a separate computer dedicated to SDE running on SQL*Server. Attribute data for the site is served by the County Assessor database that resides on a separate system on the internal network running SQL*Server.

Upon site implementation, testing quickly revealed that although performance was acceptable on the Counties internal network, performance was unacceptably poor for external users with 28.8K and 56K modem access via the internet. For example, the initial load time for the site was approximately 3 minutes for a user connected to the Internet with a 28.8K modem. Since the average user of the El Paso County web site connects via 28.8K or 56K modem, this was determined to be an unacceptable level of performance. As a result, much research was performed and learned in the area of performance measurement and tuning for ArcIMS.

Performance Measurement

Before starting ArcIMS performance tuning it is a ~~good idea to~~, no, it is a requirement to establish a performance benchmark. A performance benchmark provides a standard measurement methodology for performance. A repeatable benchmark test is needed to compare performance results before and after changes or tuning is performed.

The goal of benchmark definition is to design a repeatable test and methodology that assesses the performance of the web site under various conditions. In addition the initial "benchmark" testing will establish a starting performance level for the website's performance before any changes are made to the website. This starting point will also result in a test design and will provide a methodology to compare the results against other tests that will be conducted in the future.

Thus, the goals of creating a system of standard performance measurements has to be ease of performing the test and repeatability, accuracy of measurement, and establishment of a method of collecting and compiling performance data. The performance measurements should be applicable to the site and any possible changes that will be made to the site.

ArcIMS site performance can be measured in units of both time and data volumes.

Case Study

For our ArcIMS Performance Tuning case study, the performance test was comprised of a set of steps that a typical user might perform on the El Paso County web site. The test analyzed the web site's performance for a 28.8 Kbps modem, a 56 Kbps modem, a cable modem, 5 concurrent users, 10 concurrent users, 20 concurrent users, and 50 concurrent users. Tuning would be targeted to the dial up users which the County determined was their average users mode of accessing the site.

Steps representing typical actions a user might perform with the site:

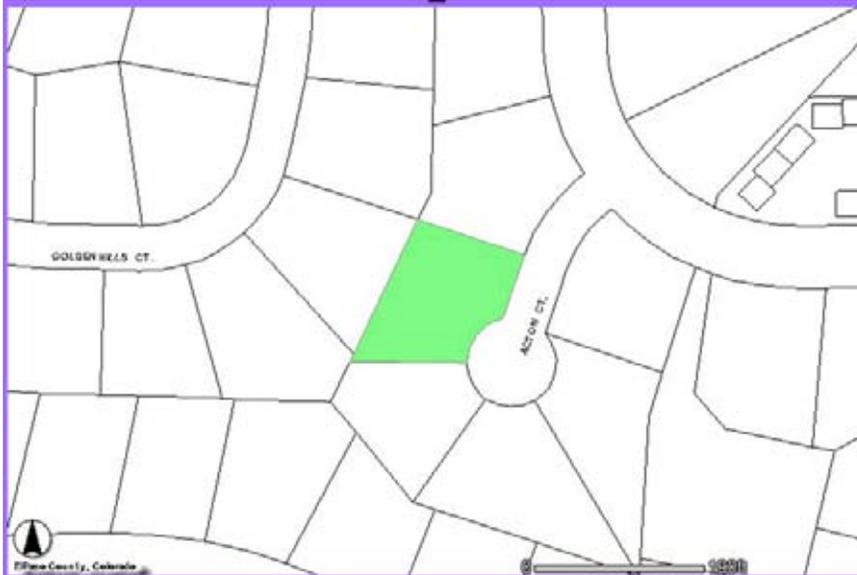
Step ID	Description
1	Load the ArcIMS website home page
2	Search for a parcel by ID number (1)
3	Search for a parcel by ID number (2)
4	Pan the map
5	Zoom out (1)
6	Select a parcel with the info tool (1)
7	Select a parcel with the info tool (2)
8	Zoom out (2)
9	Zoom in

Example (Step ID 2):

Type *7324202008* into the schedule number text box and click on the *Search* button.

To establish the benchmark baseline three scenarios of stress tests were established to emulate the typical range of user connection speed as well as both a load and no load situation on the server. The three test scenarios were:

1. [28.8K – 1 User] Typical external user connecting with 28.8K modem
2. [100 mb/s – 1 User] Typical internal user connecting via 100 mbs network
3. [100 mb/s – 20 Users] High vol. (20) internal users, simultaneous requests



So how does browsing to a web site and performing 9 steps representing typical user actions make it a performance benchmark? Well, in order for this to be a useful performance benchmark, it needs to be measured accurately with time and data volume. You also need to be able to precisely repeat these exact user actions in future tests. Using a stopwatch might come to mind as an acceptable tool for measurement of time. But, timing these user actions would be too dependent on the time the user takes to click their mouse in the right locations. Thus a better and more precise tool and method for measurement of time and data volume is needed.

Software Tools

Several commercial tools are available for precise performance measurement. We selected one that not only worked well but also was free.

Our case study utilized Microsoft's Web Application Stress Tool (henceforth, WAS). This program was found on the *Microsoft Internet Information Server Resource Kit*, and was available for download from <http://homer.rte.microsoft.com>.



The tool records the user interaction with a web site into a script that can be run again and again or simultaneously and can simulate various user loads, various Internet bandwidths and conditions. WAS allows you to program the number of users to emulate the typical actions for and the volume of requests sent to the

web site. The tool, when run also records very precisely the time to first byte (TTFB) and time to last byte (TTLB) for every request made to a web server for the benchmark steps designed. In addition, precise information is recorded as to the volume of data being sent and returned from a web server.

Once a test script is established and you have set the test parameters, you can run the test. After the test concludes, the Web Application Stress Tool will provide a detailed report about the test. Some of the details provided in this report include the following:

- Total number of hits
- Hits per page
- Number of socket errors
- Number of requests per second
- Any performance counters you specified prior to running the test
- Time To First Byte (TTFB) and Time To Last Byte (TTLB) for each item on a page

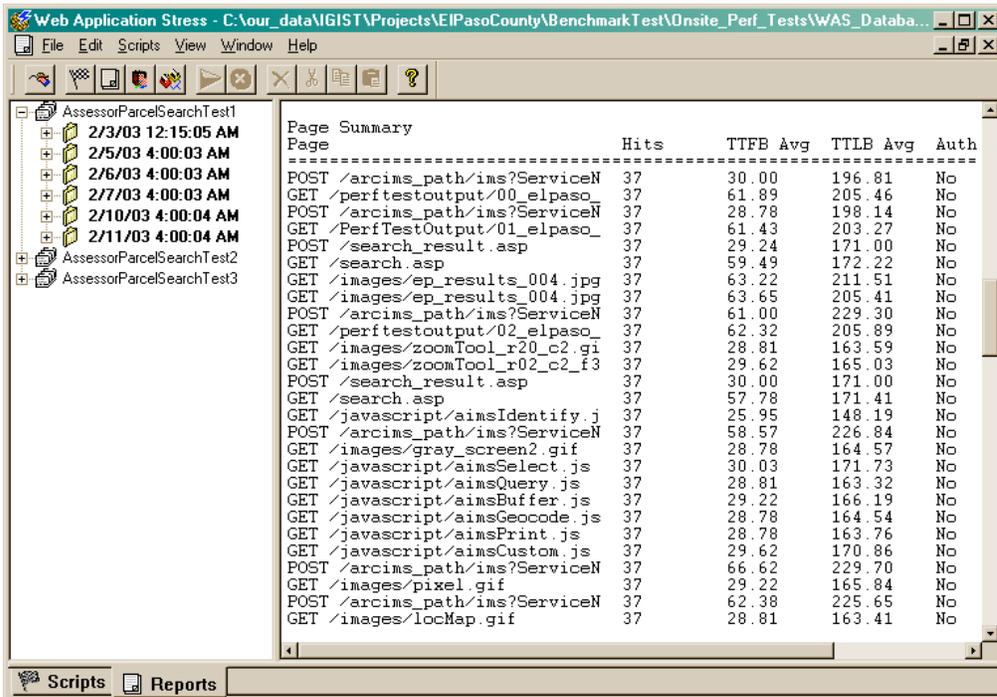
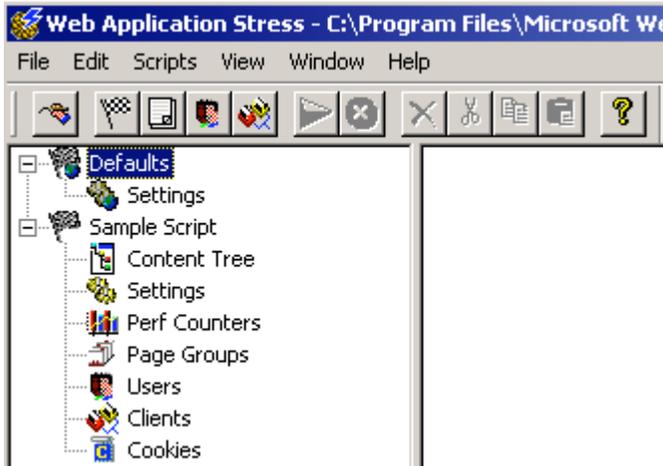
So does this measure ArcIMS performance or the web server performance? The answer is both ArcIMS and web server performance are combined to measure the effective throughput to the end users browser. It's a measurement of both.

It appears that the WAS tool is no longer available for download from the site referenced above, so I would be glad to send a copy to anyone that would like to license it. Distribution and use is subject to a Microsoft License and Use agreement upon installation. Just send an [email request](mailto:rod@kuhnsandassociates.com) to rod@kuhnsandassociates.com and request the Microsoft Web Application Stress Tool.

After you complete the download, run *setup.exe* to install WAS. Note that you will need Administrator rights to perform the installation. By default, the setup installs the tool in the directory *C:\Program Files\Microsoft Web Application Stress Tool* and adds an entry for the tool on the *Programs* menu of the Windows *Start* button. The setup program also installs an Access database, *was.mdb*, which WAS uses to store test scripts and test results data.



WAS works with Microsoft Internet Explorer to create the stress tests. After the tests are run, WAS creates reports, which can be exported to ASCII text files in comma-separated value (CSV) format. These reports can be inserted into Microsoft Excel for further analysis and compilation.



Methodology used for compiling test results:

The raw data gathered by the WAS tool was divided into the major steps in the test. Each step was then organized into substeps. Principally, the substeps break a step down into things like downloading interface images (icons, etc.), downloading the map images, sending AXL requests, and the like.

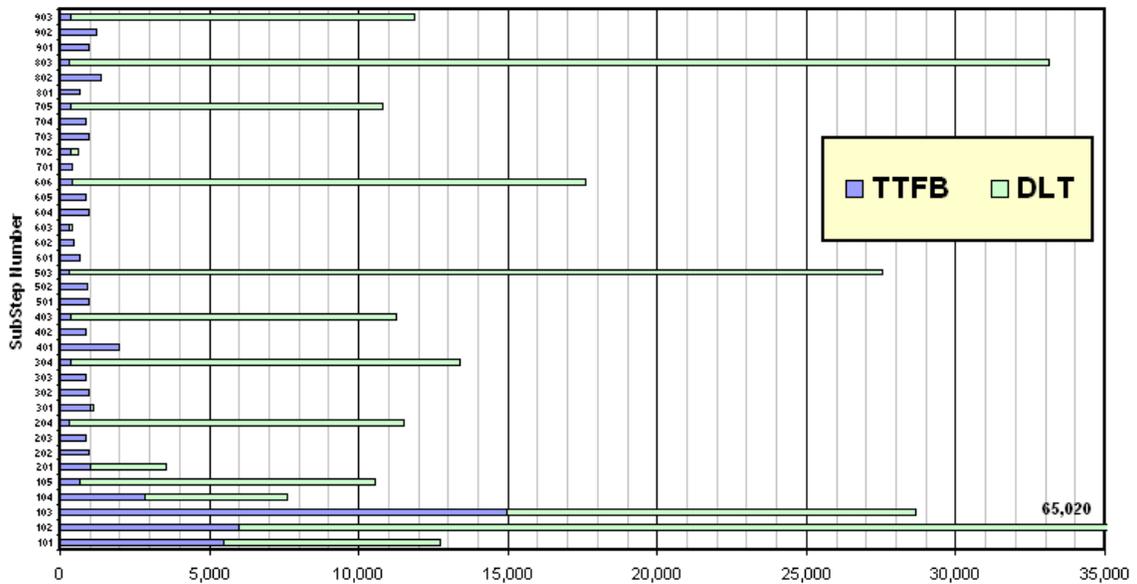
Each substep was assigned a unique three digit ID number. The following table shows a complete list of the substeps, their IDs and the step that they belong to:

Description	SubStep	Step
Download HTML	101	1
Download JavaScript	102	1
Download Images	103	1
ArcIMS Request	104	1
ArcIMS Map download	105	1
Parcel search 1 - form posts	201	2
Parcel search 1 - get feature AXL request	202	2
Parcel search 1 - parcel search AXL request	203	2
Parcel search 1 - download map image	204	2
Parcel search 2 - form posts	301	3
Parcel search 2 - get feature AXL request	302	3
Parcel search 2 - parcel search AXL request	303	3
Parcel search 2 - download map image	304	3
Download Images	401	4
Pan map AXL request	402	4
Pan map image	403	4
Download Images	501	5
Zoom out AXL request	502	5
Download zoom out image	503	5
Download Images	601	6
AXL request based on mouse click	602	6
Attributes listing	603	6
AXL GET_FEATURES request (by ID)	604	6
AXL request for map	605	6
Download info tool map	606	6
AXL request based on mouse click	701	7
Attributes listing	702	7
AXL GET_FEATURES request (by ID)	703	7
AXL request for map	704	7
Download info tool map	705	7
Download Images	801	8
Zoom out AXL request	802	8
Download zoom out image	803	8
Download Images	901	9
Zoom in AXL Request	902	9
Download zoom in image	903	9

In summary the compilation of results involved the following:

- ❑ Assignment of each Request to a SubStep
- ❑ Extracting the Bytes Uploaded and Downloaded for Each Request
- ❑ Calculating the Totals for the Steps and SubSteps
- ❑ Creating "Conditional Sum" Formulas in Excel
- ❑ Calculating the Byte Totals for Each SubStep
- ❑ Calculating the Byte Totals for the Steps
- ❑ Calculating the Download Times for the SubSteps
- ❑ Calculating the Download Times for the Steps
- ❑ Creating Summary Graphs
- ❑ Creating Comparison Graphs

The following graph shows a compilation of resultant time data collected by WAS



ArcIMS Log Analyzer

Another useful tool for analyzing ArcIMS performance is the ArcIMS log Analyzer produced by Björn Svensson. The log Analyzer is a perl script (aimslogs.pl) that compiles and presents ArcIMS log statistics in an HTML or .xls format. The script analyzes your ArcIMS FeatureServer, ImageServer, and QueryServer log files for performance statistics and calculates average times, and total requests and responses for each map service and layer.

Example ArcIMS Log Analyzer output:

SUMMARY of El Paso County ArcIMS log file analysis

9/12/2002 - 11/19/2002

Kuhns & Associates, Inc.

			Second		Kilobyte	
2 dif	M					
FEA	M					

Most popular IMAGE MapServices (out of 2 different MapServices)

	M					
	e					
	e					

Interpreting the Results of the ArcIMS Log Analyzer

Statistics by MapService

AXL Parse Time - total time required to complete the 3 steps in axl parsing:
building a tree from the AXL, creating objects AXL needs, and
interpret the AXL objects.

OUTPUT TIME - time spent to generate the image file.

TOTAL PROCESSING TIME - total time to prepare, send, fetch, and draw the
data.

Statistics by Layer

DATA SEARCH TIME - time to prepare and send the query with its attribute and
spatial constraints.

DATA RETRIEVAL TIME - time to fetch and draw the data.

Stress Testing with WAS and Performance Analysis

Stress testing is the process of simulating a large number of users and
generating a heavy load on the web server. This section provides some general
information and advice for stress testing an ArcIMS web site and analyzing its
performance.

There are two main categories of stress testing:

Performance testing

Stability testing

Stress testing is done to measure the maximum requests per second that the
web server can handle before it starts to fail. This is a quantitative measurement.
The Microsoft Web Application Stress Tool allows you to set and increase the
number of requests per second sent to a web site.

Stability testing is place a website under a long term load to test if continued high
volumes of stress can be maintained or handled without a crash or failure.

Stability testing is done by utilizing WAS to generate an above average load for the site over a period of say 24 hours.

Performance analysis involves determining which resource prevents the requests per second from going higher, such as CPU, memory, ArcIMS Spatial Server, ArcIMS Application server or other backend dependencies such as data access / SDE etc. We refer to this as determining the bottleneck and is much more of an art than a measurement.

To determine the maximum overall web server processor processing capacity, you could increase the stress to the point where the requests per second start to decrease, then back the stress off slightly. This is the maximum performance that the web site can achieve. Increasing stress is accomplished by increasing the stress level (threads) and/or sockets in Web Application Stress, and increasing the number of Web Application Stress client machines.

To help determine if the Web server is the bottleneck, use Performance Monitor to watch the processor utilization and "Web Service: Connection Attempts/sec" as well as "Requests Queued" counters. If the processors are running at 80 to 85%, then they are most likely the bottleneck. If the Requests Queued fluctuate considerably during the stress and the processor utilization remains relatively low, this is an indication that the script is calling a Server Side Object that is receiving more calls than it can handle. In this case, the Server Side Object could be the bottleneck.

One approach to utilizing the WAS tool to analyze if your sites performance is adequate would be to determine a reasonable peak ArcIMS site load that meets your business needs. Then create a test that uses enough threads in WAS to reach that criteria. This will reveal the max request rate supported for the web application and confirm that it is in line with your expected peak load.

Interesting Example provided by WAS documentation:

"On a single processor web server, Internet Information Server (IIS) can handle 10 requests at one time when the web server is configured with the default settings. If requests are incoming at a higher rate than they are being processed, then all 10 threads dedicated to handling requests will be busy. After that, additional requests are placed in the queue, which can hold up to 490 pending requests. If the queue fills up to 490, then the web server returns the error message "HTTP/1.0 Server Too Busy".

OK – now we know about stress testing and finding bottlenecks, so how then do we tune an ArcIMS site to reduce its bottlenecks and increase its performance?

Performance Tuning Methodology

Now that we have a tool to measure our ArcIMS site performance accurately, we can perform tuning in the cyclical fashion of test a little, tune a little, test a little etc. Its important to make only one performance altering change at a time so that conclusive evidence can be generated as to the performance impact of each change.

First, consider what options you have for tuning. Once bottlenecks have been determined, often times it takes resources (a.k.a. money to change or add resources). The best “bang for the buck” for additional hardware and software upgrades is:

- 1) Spatial Data Server – i.e. SDE (for sites supporting large datasets)
- 2) Memory vs. Processor upgrades
- 3) Add Web servers / Spatial servers / Application Servers
- 4) Web site Technology (Server side environment vs. client side processing)

However, much performance tuning can be performed to increase performance of an ArcIMS site without adding additional expensive hardware and software upgrades.

ArcIMS Web site Performance Tuning should focus on:

- Data tuning
- AXL configuration tuning
- Application tuning
- ArcIMS Spatial Server tuning
- ArcIMS Map Service tuning

Data Tuning

Performance tuning for data can be effective by implementing ways to reduce the amount (volume) of data that needs to be processed by the spatial server. For example data can be generalized to remove unnecessary vertices, or combining or eliminating features that are not required at the scale the data will be presented at on the Web Site. If your ArcIMS map service utilizes shape files, make sure that the spatial index files are implemented and are in sync with the spatial data. (.sbn and .spx files). If data is served by ArcSDE to your ArcIMS web site, data tuning consists of RDBMS tuning and configuring appropriate grid sizes.

AXL Configuration Tuning

The following AXL tuning tips and recommendations were suggested by Bart Killpack, ESRI Denver:

- 1) Tune scale dependencies
- 2) Use Where clauses instead of valumaps in Query and SpatialQuery statements
- 3) Use SPATIALQUERY search order "attributefirst" clause (ArcSDE)
- 4) Use Strings and Integers in Valuemaps
- 5) Use Featurelimits for large layers (in query, get_features, spatialquery)
- 6) Minimize the use of antialiasing and transparency
- 7) Minimize multilayers symbols (highway shields etc.)
- 8) Select the appropriate client for the application
 - a. Thick Client – Java, ActiveX, ArcExplorer, ArcGIS
 - b. Thin Client – JavaScript DHTML
 - c. Server Side – HTML from ColdFusion, JSP, ASP, etc.
 - d. Select appropriate output image type (GIF, JPG, PNG-8, PNG-24);
 - e. JPG, PNG-24 for > 256 colors
 - f. GIF, PNG-8 for < 256 colors
- 9) Use raster types with a pyramid structure (MrSID, ArcSDE) or uncompressed imagery (TIFF)
- 10) Avoid projection on the fly (store data in geographic coordinates)

Application Tuning

For serving users with low bandwidth connections use a server side application (i.e. thin client model)

For serving users with DSL, Cable or Internet connections a thick client is acceptable/preferred

For simple applications (pan, zoom, id) use server side / thin client model

For applications requiring heavy geoprocessing, use distributed or thick client model.

For HTML Viewer based ArcIMS web sites, chop out un-used code. (This could prove to be challenging and time consuming)

For sites requiring significant customization, consider using server side connectors (Java, ActiveX, or ColdFusion). The programming and debugging environments for Server side programming are a huge benefit in development.

Minimize the client – server round trip requests required as much as possible.

ArcIMS Spatial Server Tuning

The ArcIMS Spatial Server is responsible for carrying out the requests for spatial information (generate image, return features, perform geocoding or extract shapefiles). The Spatial Server runs as two background processes Monitor and Tasker (windows services). Monitor tracks the status and keeps things running, and Tasker cleans up old images.

Most ArcIMS installations utilize a single Spatial Server. In some cases, it is beneficial to have more than one Spatial Server. It is most advantageous to performance to add additional Spatial Servers across multiple CPU's in a distributed fashion.

The following definitions and spatial server performance tuning tips were provided by ESRI and compiled from the ArcIMS on-line help:

Each Spatial Server is assigned to one or more Virtual Servers. This association defines the types of requests that are processed by the Spatial Server. For example, a Spatial Server associated with an ArcMap Virtual Server processes requests against ArcMap Services. It cannot also be associated with a Metadata Virtual Server, performing Metadata Service related work. The Spatial Server Properties window details any current Virtual Server associations and the number of instances used.

Each ArcIMS map service must be assigned to a single Virtual Server but each Virtual Server can have more than one map service assigned to it.

Additionally, each Virtual Server is associated with one or more Spatial Servers. When more than one Spatial Server is used, this association allows you to provide uninterrupted access to the Virtual Server's assigned services.

To improve system performance, consider changing the number of instances of a particular Virtual Server. The number of instances corresponds to the number of simultaneous hits your Web site can handle.

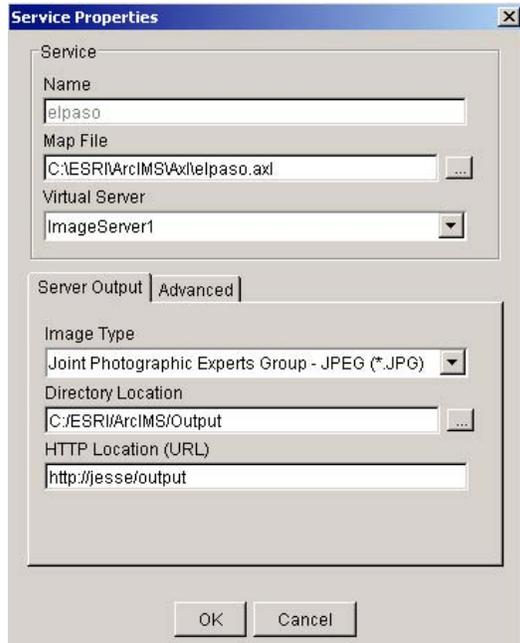
By default, each Virtual Server, except ArcMap, utilizes two instances per Spatial Server. ArcMap Server uses only one instance per Spatial Server. In addition, an instance is utilized by ArcIMS Monitor and another by ArcIMS Tasker.

You can change or reallocate the number of instances used by a particular Virtual Server. For example, if one Virtual Server is used more heavily than the others, you might consider allocating more instances to it.

To monitor system usage and performance, periodically check Virtual Server statistics.

ArcIMS Map Service Tuning

ArcIMS Administrator allows several configuration settings. This section will show how to determine the performance impact of various ArcIMS Map service property settings through examples conducted in our case study.



In our case study we initially invested in the least cost tuning activities to attempt to get the site performing to an acceptable level with configuration tuning:

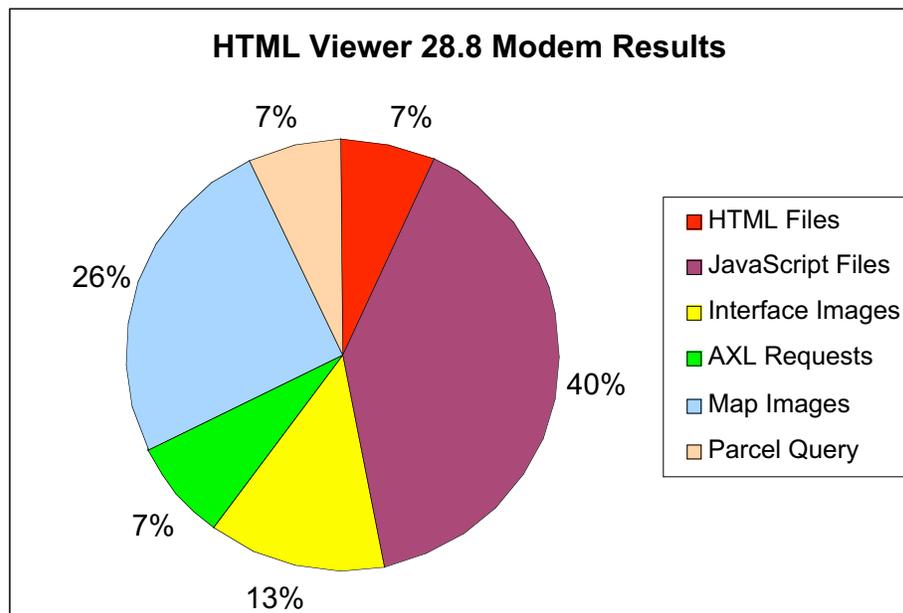
For our case study we utilized the performance measurement tools and methodology to test the following ArcIMS Map service configuration state settings:

- A. *Benchmark Basis:*
- B. *PNG 8 Bit Map Images:*
- C. *PNG 24 Bit Map Images:*
- D. *Three Server Instances:*
- E. *Four Server Instances:*
- F. *Scale Dependencies - AXL File Changes:*
- G. *Font Characteristics - AXL File Changes:*
- H. *Browser Size (optional):*

Testing showed that the optimum configuration for the El Paso site was to use three server instances, and 8 bit png images. Scale dependencies also had a significant impact on performance. Again, anything that produced a larger file download such as font characteristics and browser size had an impact on performance. Our tuning was performed to maximize the performance for the remote modem users.

Case Study Results and Findings

The results of the baseline performance test show that the initial download of the website can be expected to take over two minutes on a 28.8 modem. Downloading the JavaScript files and the images for the web interface make up the majority of the initial load. Once the initial page is loaded, the subsequent substeps mostly loaded in less than one second, except for the map JPG images. The TTFB for those images was quick, and the majority of the time was spent transmitting the file itself. The TTLB was greater than 10 seconds for every one of the map JPG images. Thus, the three principal factors in the delays experienced by 28.8Kbps modem users are in downloading the JavaScript files, the images and icons in the website's interface, and the map JPG images. This study did not find that there are significant delays caused by the web server or ArcIMS.



The biggest performance issue in the case study was found to be:

- 1) The large amount of JavaScript required to be downloaded initially to the end user browser associated with the HTML Viewer based site.

Large map images and their download time over slow bandwidth connections.

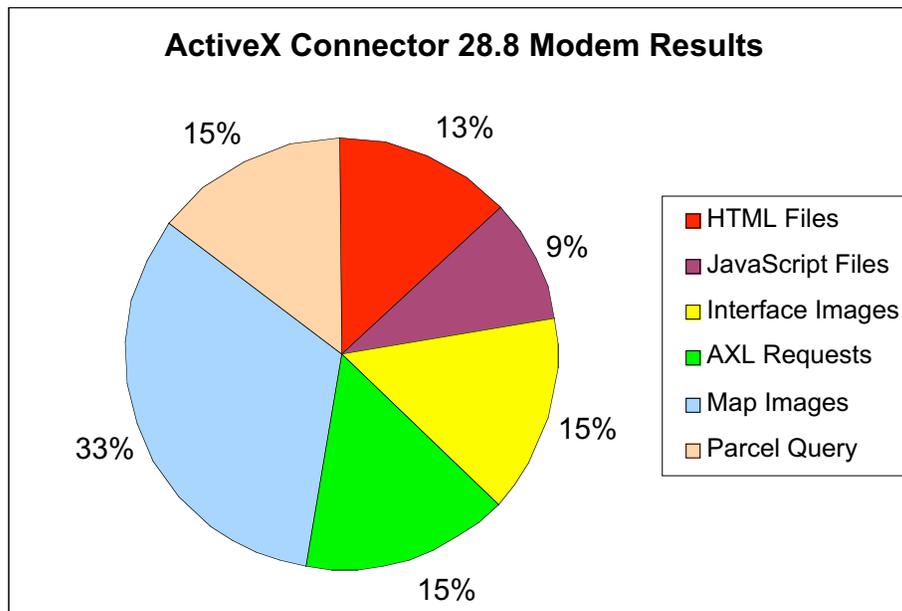
- 2) The large amount of round trip requests to the server and back to the user browser which is typical of the client processing model with the HTML Viewer based site.

In addition to performance, it was also learned that the HTML Viewer technology based on the default connector used by Servlet Exec and the HTML Viewer, exposes a data security problem. All data is wrapped in XML packages and delivered to the spatial server for XML parsing and image creation. Other internet based clients such as ArcGIS have the ability to connect to this type of ArcIMS service through the default connector and download the source data to the external internet based system in the same fashion.

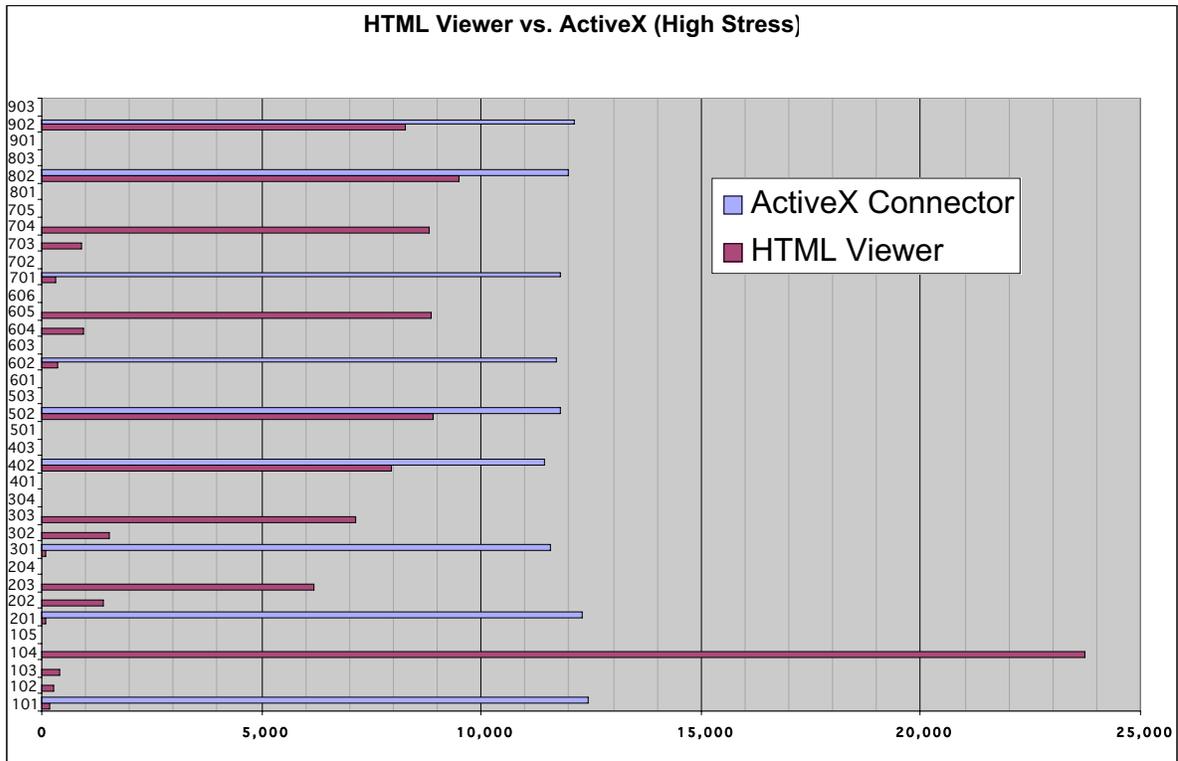
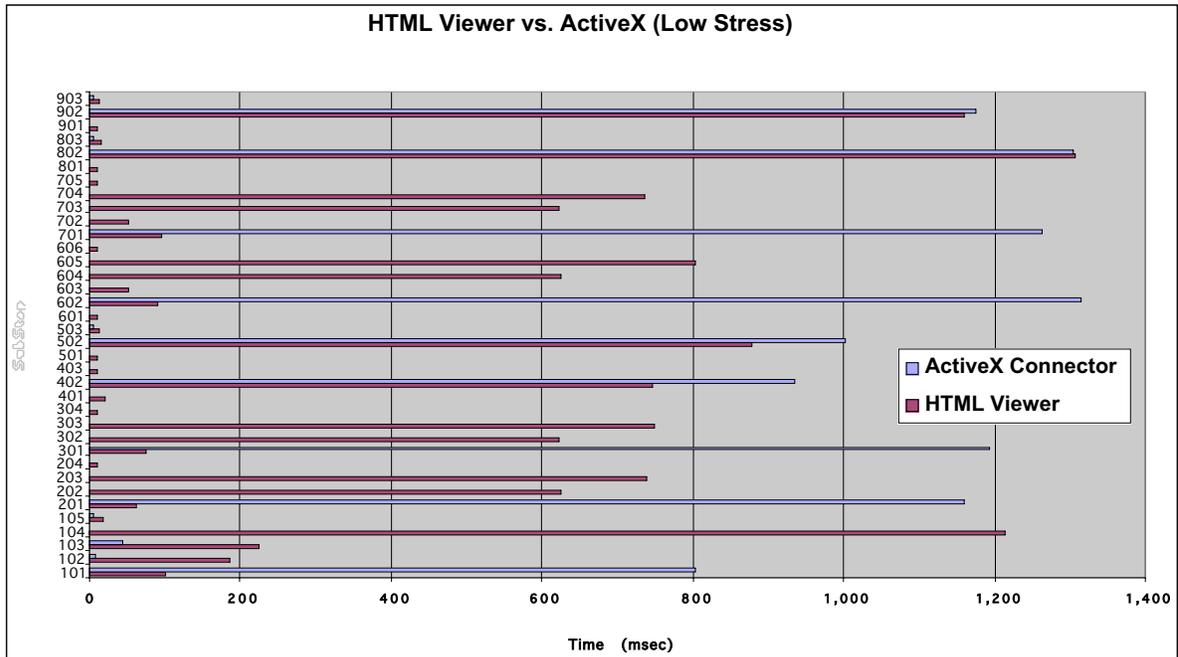
Thus, the poor performance for slow bandwidth connection users and the security issue prompted El Paso County to migrate the site to Active-X ASP.NET.

Advantages of Active-X Connector and Server Side ASP.NET

- Much better programming environment (Visual Studio.NET with full debugging and tracing capabilities)
- Code is compiled and not interpreted
- Data Security
- Performance: Initial JavaScript Download time greatly reduced (120% performance improvement)
- Ease of customization and maintenance



HTML Viewer vs. Active-X Performance Results



* Note that Active-X perform in one step what the HTML Viewer often performs in many steps

ACKNOWLEDGMENTS

Eckhart, Jeff, GIS Manager El Paso County Information Technologies

Killpack, Bart, Colorado ArcIMS User Group presentation and additional technical support: ESRI-Denver, 2003.

Askov, David, – Performance test development and results analysis, Co-author

ESRI, ArcIMS Guides

Björn Svensson, ArcIMS log Analyzer produced by

Rod D. Kuhns
President, Kuhns & Associates, Inc.
P.O. Box 4863
Englewood, Colorado 80155

<http://www.kuhnsandassociates.com>

Email: rod@kuhnsandassociate.com