

# **Web Application Stress Test Methodology**

*ESRI Systems Integration Technical Brief*

---

June 2005

Copyright © 2005 ESRI  
All rights reserved.  
Printed in the United States of America.

The information contained in this document is the exclusive property of ESRI. This work is protected under United States copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by ESRI. All requests should be sent to Attention: Contracts and Legal Services Manager, ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

The information contained in this document is subject to change without notice.

#### **U.S. GOVERNMENT RESTRICTED/LIMITED RIGHTS**

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

@esri.com, 3D Analyst, ADF, AML, ARC/INFO, ArcAtlas, ArcCAD, ArcCatalog, ArcCOGO, ArcData, ArcDoc, ArcEdit, ArcEditor, ArcEurope, ArcExplorer, ArcExpress, ArcFM, ArcGIS, ArcGlobe, ArcGrid, ArcIMS, ArcInfo Librarian, ArcInfo, ArcInfo—Professional GIS, ArcInfo—The World's GIS, ArcLocation, ArcLogistics, ArcMap, ArcNetwork, ArcNews, ArcObjects, ArcOpen, ArcPad, ArcPlot, ArcPress, ArcQuest, ArcReader, ArcScan, ArcScene, ArcSchool, ArcSDE, ArcSdl, ArcStorm, ArcSurvey, ArcTIN, ArcToolbox, ArcTools, ArcUSA, ArcUser, ArcView, ArcVoyager, ArcWatch, ArcWeb, ArcWorld, Atlas GIS, AtlasWare, Avenue, BusinessMAP, Database Integrator, DBI Kit, ESRI, ESRI—Team GIS, ESRI—The GIS Company, ESRI—The GIS People, FormEdit, Geographic Design System, Geography Matters, Geography Network, GIS by ESRI, GIS Day, GIS for Everyone, GISData Server, *Insite*MAP, JTX, MapBeans, MapCafé, MapObjects, ModelBuilder, MOLE, NetEngine, PC ARC/INFO, PC ARCPLOT, PC ARCSHELL, PC DATA CONVERSION, PC STARTER KIT, PC TABLES, PC ARCEDIT, PC NETWORK, PC OVERLAY, PLTS, Rent-a-Tech, RouteMAP, SDE, SML, Spatial Database Engine, StreetEditor, StreetMap, TABLES, the ARC/INFO logo, the ArcCAD logo, the ArcCAD WorkBench logo, the ArcCOGO logo, the ArcData logo, the ArcData Online logo, the ArcEdit logo, the ArcExplorer logo, the ArcExpress logo, the ArcFM logo, the ArcFM Viewer logo, the ArcGIS logo, the ArcGrid logo, the ArcIMS logo, the ArcInfo logo, the ArcLogistics Route logo, the ArcNetwork logo, the ArcPad logo, the ArcPlot logo, the ArcPress for ArcView logo, the ArcPress logo, the ArcScan logo, the ArcScene logo, the ArcSDE CAD Client logo, the ArcSDE logo, the ArcStorm logo, the ArcTIN logo, the ArcTools logo, the ArcView 3D Analyst logo, the ArcView Business Analyst logo, the ArcView Data Publisher logo, the ArcView GIS logo, the ArcView Image Analysis logo, the ArcView Internet Map Server logo, the ArcView logo, the ArcView Network Analyst logo, the ArcView Spatial Analyst logo, the ArcView StreetMap 2000 logo, the ArcView StreetMap logo, the ArcView Tracking Analyst logo, the Atlas GIS logo, the Avenue logo, the BusinessMAP logo, the Data Automation Kit logo, the ESRI ArcAtlas Data logo, the ESRI ArcEurope Data logo, the ESRI ArcScene Data logo, the ESRI ArcUSA Data logo, the ESRI ArcWorld Data logo, the ESRI Digital Chart of the World Data logo, the ESRI globe logo, the ESRI Press logo, the Geography Network logo, the MapCafé logo, the MapObjects Internet Map Server logo, the MapObjects logo, the MOLE logo, the NetEngine logo, the PC ARC/INFO logo, the Production Line Tool Set logo, the RouteMAP IMS logo, the RouteMAP logo, the SDE logo, The World's Leading Desktop GIS, *Water Writes*, www.esri.com, www.geographynetwork.com, www.gisday.com, and Your Personal Geographic Information System are trademarks, registered trademarks, or service marks of ESRI in the United States, the European Community, or certain other jurisdictions.

Other companies and products mentioned herein are trademarks or registered trademarks of their respective trademark owners.

---

# Table of Contents

Section	Page
<b>1.0 Introduction.....</b>	<b>1-1</b>
<b>2.0 Scope.....</b>	<b>2-1</b>
<b>3.0 Web Stress Testing Primer .....</b>	<b>3-1</b>
3.1 Transaction.....	3-1
3.2 Throughput.....	3-1
3.3 Utilization.....	3-2
3.4 Service Time .....	3-2
3.5 Queue Time.....	3-2
3.6 Response Time.....	3-3
3.7 Scalability.....	3-3
3.8 Performance .....	3-3
<b>4.0 Test Readiness .....</b>	<b>4-1</b>
4.1 Application Center Test .....	4-1
4.2 Develop a Hypothesis .....	4-1
4.3 Analyze the Application Design .....	4-1
4.4 ArcIMS Configuration Tuning.....	4-2
4.5 System Analysis.....	4-2
4.6 Application Work Flow .....	4-2
4.7 System Monitoring.....	4-3
<b>5.0 ArcIMS Stress Testing .....</b>	<b>5-1</b>
5.1 Application Creation and Configuration.....	5-1
5.2 Recording the Script.....	5-1
5.3 Script Validation .....	5-2
5.4 Calculating Transaction Throughput .....	5-2
5.5 Calculating Map Service CPU Time.....	5-3
5.6 Calculating Web Browser Response Time .....	5-4
5.7 Script Execution and Configuration Tuning .....	5-4
<b>6.0 ArcGIS Server Stress Testing.....</b>	<b>6-1</b>
6.1 Application Creation and Configuration.....	6-1

---

<b>Section</b>	<b>Page</b>
6.2 Recording the Script.....	6-3
6.3 Script Validation .....	6-3
6.4 Calculating Transaction Throughput .....	6-4
6.5 Calculating Map Service CPU Time.....	6-4
6.6 Calculating Web Browser Response Time .....	6-4
6.7 Script Execution and Configuration Tuning .....	6-4
<b>7.0 Concluding Remarks .....</b>	<b>7-1</b>
<b>8.0 Support.....</b>	<b>8-1</b>

## 1.0 Introduction

Deploying a Web-based application requires sufficient hardware and infrastructure resources to ensure its performance and success. One of the steps in the implementation process requires some level of system testing to ensure that the system is performing as designed. This technical brief will describe how to use Microsoft's Application Center Test tool (ACT) to stress test a template-based ArcIMS and ArcGIS Server map viewer application. The same test methodology can be applied to custom applications as well but may require a different approach for services other than map services.



## 2.0 Scope

This document contains two major sections. The first section addresses testing ArcIMS, and the second section addresses testing ArcGIS Server. This document is not a step-by-step procedure for stress testing and does assume a certain level of knowledge regarding system performance testing, computer hardware architecture, and ESRI Web-based product configuration and tuning. It is intended for technical personnel involved in the design, configuration, and administration of Web server systems. References will be made to operational laws for calculating performance parameters, but they will not be discussed in detail.



## 3.0 Web Stress Testing Primer

Capacity planning for systems, such as database servers, is typically approached from a concurrent user perspective. That is, the question might be asked, How large should a server be to support 100 concurrent users? The approach for capacity planning of Web server applications is somewhat different because of the work flow differences and the user base. For typical database applications, the user base is mostly understood, and so is the user work flow. For Web applications, the user base can be quite large and the work flow is often quite different. For example, imagine a Web-based restaurant locator application that is exposed to the world on the Internet. A typical user arrives, spends about five minutes looking for a restaurant, then leaves. This work flow is very different from, for example, a GIS desktop analyst that connects to a database server for several continuous hours and performs a variety of tasks. Therefore, the idea of concurrent users is somewhat ambiguous for most Web applications, and a different approach should be taken. This different approach should be based on a transactional model and focused on the transactional throughput of a Web-based system, regardless of the number of users connecting and issuing the requests. The focus of this approach is to determine how many GIS requests, or transactions, in a given time period a system can provide while maintaining reasonable user response time. The answer will vary, depending on the complexity of the application and the processing power of the Web system. Therefore, performance modeling and analysis can be used to predict the throughput of a Web system, and Web stress testing can be used to verify system capacity and performance. Before continuing, it is important that some critical terms be defined, since these terms are often used interchangeably or incorrectly.

### 3.1 Transaction

A transaction is defined as a Web page request that results in a Web browser output display that the user examines before performing the next task. That is, only the final Web product is counted as a transaction, not each individual Web request (GETs, POSTs, etc.) that is required to create the page of output. Typically, a GIS application Web page will contain a map object and potentially several other mapping objects such as a scale bar and a layer table of contents. As an example, panning from one section of data to the next, which causes the map to refresh along with other Web controls, is considered one transaction.

### 3.2 Throughput

Throughput is the rate at which some amount of work is being performed. In a Web environment, one measure of throughput is the transaction rate, or the number of Web transactions performed in a given period of time. Typically, this is reported in transactions per hour (TPH) or transactions per second (TPS).

### 3.3 Utilization

Utilization measures the fraction of time a device is busy servicing a request and is usually reported as a percentage using a scale from 0 to 1, in which 0 is idle and 1 is fully utilized. Utilization is a function of throughput and service time and results in the following Utilization Law:

$$Utilization = Throughput \times Service Time$$

### 3.4 Service Time

Service time is the time required by a device (or server) to service a request. For example, a bank teller may process 30 sequential check cashing requests per hour. During each transaction, the teller is 80 percent utilized and spends 20 percent of the time on other tasks, breaks, and so forth. The Utilization Law can be manipulated and used to determine the average service time for each transaction (single server) as follows:

$$Service Time = \frac{Utilization}{Throughput}$$

In this case, utilization is .8 (80%) and throughput is 30 transactions per hour. Therefore, the service would be .8 divided by 30, or .0267 hours (1.6 minutes). In this example, the average service time for each bank transaction is 1.6 minutes. This same methodology applies to computer system performance analysis in which a request is a Web transaction and the service time of primary concern is the CPU service time. CPU service time is directly related to the processing capability of the CPU and therefore testing to determine performance and throughput should be performed on hardware that is equivalent to the hardware planned for deployment. If similar hardware is not available, results from other hardware can be used in conjunction with SPEC performance benchmarks to extrapolate the results to the target system, which is beyond the scope of this document. Also, it is necessary to adjust the service time equation for the number of servers, or CPUs/CPU cores, as follows:

$$Service Time = \frac{Utilization \times CPUs}{Throughput}$$

### 3.5 Queue Time

Queue time is the amount of time spent waiting to receive service. This could be related to the application queue, the CPU queue, the disk queue, and so on. For example, if a thread that normally takes one second of CPU time is executing concurrently with a similar thread, then the CPU queue time for each thread will be one second as well, because each thread has to share the CPU and execute for a short period of time, then get out of the way to allow the

second thread to execute. In addition, applications such as ArcIMS have their own queues to prevent too many requests from being processed concurrently, and so forth. All the various queues involved with the request add up to total queue time.

### 3.6 Response Time

Response time is the accumulation of time associated with processing requests (service time) and waiting for service (queue time). Response time would also include any other delays such as network latency. Therefore, response time represents the user end-to-end experience. In a Web environment, response time would be the time required to satisfy a Web page request such as a map zoom from the time the user initiated the request to the time the page completed drawing in the web browser. In Web environments, response time can be much longer than the CPU service time required to generate the map page, map image, etc.

$$\text{Response Time} = \text{Service Time} + \text{Queue Time}$$

### 3.7 Scalability

Scalability is the ability of a computer system to adapt to an increasing load demand while providing minimal performance degradation. For a Web system, this could be represented by the ability of a single server to maintain the same service time for requests whether it is processing a single request or multiple requests (assuming the system is not fully utilized). That is, the system should be able to scale up to its peak output with minimal performance degradation. With multiple servers, the system should scale as a function of the aggregate computing capability of the servers. Thus, if two map servers are present and are of the same capability, then both systems should be able to provide nearly double the throughput of a single system. If not, a bottleneck most likely exists and, therefore, the system is not scaling. Other components of the system, such as the data server and the network, must scale as well.

### 3.8 Performance

Performance is related to response time and is based on a user's observation of how well a system is performing a given task. Performance is determined by the speed of a system's components and not necessarily the number of components that are available. For example, a user will observe that a request from a 3.6 GHz dual-CPU server is faster than one from a 2.0 GHz, quad-CPU, although the total computing capability of the two servers may be identical (based on SPEC benchmarks, for example). Therefore, they will both scale identically, but the 3.6 GHz, dual-CPU server will outperform the 2.0 GHz, quad-CPU system. The only time both systems will appear to perform identically is when they are both operating at peak capacity. This is why periodic hardware upgrades are critical for maintaining performance and productivity.



## 4.0 Test Readiness

Prior to performing any tests, some homework should be completed and analysis performed. It is not a good practice to merely configure for a test, run it, collect the results, and then accept the results as fact. The results may appear to be valid, but invalid conclusions could be drawn. Or, the results may be valid but appear invalid because of the impact of a test parameter that has not been considered, again, resulting in incorrect conclusions. The following sections outline several test readiness activities and how to avoid pitfalls.

### 4.1 Application Center Test

Application Center Test is a Web test tool included with certain versions of Microsoft's Visual Studio .NET development environment, for example, the Enterprise Architect Deployment. ACT can be used to record and play back a user's experience with a given Web application. Learning to use ACT to record and play back scripts is a fairly simple task and does not necessarily require a seasoned tester or developer to use. ACT does provide options for scripting capabilities to further enhance the test capability, but that is beyond the scope of this document.

### 4.2 Develop a Hypothesis

A very important test rule to follow is to never accept the results of the test without some sort of pretest and posttest analysis. Prior to testing, it is imperative to analyze the configuration and derive some sort of hypothesis regarding the outcome of the testing. ESRI has developed standard sizing models for both ArcIMS and ArcGIS Server, which are available in the *System Design Strategies* Technical Reference Document available at <http://www.esri.com/systemsint/kbase/strategies.html>. These models can be used as a good starting point for a hypothesis, and adjustments can be made based on the particular application. For example, the design models are based on a set of fairly simple mapping applications retrieving a reasonable amount of features per map display. If the application under test has 50 layers, uses complex symbology, and is retrieving 5,000 features per map display, then chances are, it will perform far differently from what has been established by the design models. So your hypothesis must include those types of considerations. The results of your test will either prove or disprove your hypothesis. If the results are not in line with the hypothesis, further analysis must be performed to determine the cause.

### 4.3 Analyze the Application Design

In parallel with developing a hypothesis, you should consider the application design. Questions should be asked, such as, Does the application really need to query 5,000 features per map display? Or can scale suppression be used to reduce it to 500? Is the use of 50 layers necessary when 15 will work? Does the symbology need to be that complex, or can simple symbology suffice? Does the map image need to be 800 x 1,000 or will 500 x 500 do?

These types of adjustments can have a tremendous impact on the application performance and, therefore, the transaction rates. For example, cutting the service time of an application in half means you need one-half the amount of hardware to achieve the same transaction rates (compared to the application with twice the service time).

#### 4.4 ArcIMS Configuration Tuning

The intent of stress testing is to push the system to its maximum potential to ensure there are no performance issues or bottlenecks. To achieve full performance potential, the configuration needs to be tuned so enough application threads are available to take advantage of the available CPU resources. Tuning threads can be time consuming and can result in trade-offs between individual application response times and full utilization of the CPUs. That is, adding threads to push the CPUs to their maximum utilization during a stress test may not be the best solution for the operational environment because of the impact on individual users' performance caused by longer response times. This topic will be addressed in later sections and is discussed in detail in the ArcIMS Configuration Performance Factors technical brief available at [http://www.esri.com/systemsint/kbase/docs/arcims\\_config\\_factors.pdf](http://www.esri.com/systemsint/kbase/docs/arcims_config_factors.pdf).

#### 4.5 System Analysis

ArcIMS performance is one of several parts that make up a system-level stress test. Often, performance issues are the result of a problem in other areas. So before any test results can be accepted, a high level of confidence must be reached that the system is healthy as a whole. This includes issues that have already been addressed such as the application and ArcIMS tuning along with others such as the network, database, server hardware, operating systems, other compute loads, and firewalls. For example, if you have performed tuning at various system levels and begin to run tests only to find that the map server CPUs can only reach 50 percent utilization, what do you do? The problem could be not enough threads, not enough memory, a network issue or other I/O problem, a database problem, and so forth. Preliminary tests will help to expose some of these issues, which should be addressed before continuing. Once you have a healthy system, you can then run your final stress tests and compare the results with your hypothesis.

#### 4.6 Application Work Flow

The stress test should represent how the application will be used in the real world. A typical work flow needs to be worked out and a representative portion followed through for script recording. The script needs to represent at least 10 different map displays so there is adequate variation in the script, which will better represent real-world application performance. For functional testing, script functions should represent typical tasks such as button clicks, turning on/off layers, page navigation, map pans/zooms, identifies, and so forth. However, many of these functions do not put much, if any, load on the Map Server, which is the main area of concern regarding performance. Map generation and any

geoprocessing functions will dominate the system load and should be the focus of the testing. For the simple map viewer application, recording a variety of pans and zooms at a modest extent should provide a sense of how the application is performing.

## 4.7 System Monitoring

While the stress test is running, various system components should be monitored including the CPUs, disk subsystem, memory, and network. System monitoring applies to all the systems that play a part in the test configuration, including the database server. Details regarding many of these topics can be found in the *System Design Strategies* Technical Reference Document at <http://www.esri.com/systemsint/kbase/strategies.html>.

### CPU Utilization

CPU utilization can approach 100 percent, but achieving this is not necessary to have a productive system and, in fact, can be detrimental to response time performance. A trade-off must be made between throughput, which is directly related to CPU utilization, and individual user experience, or response time. During stress testing, the system should be able to reach 80 percent or greater utilization. If it does not, a bottleneck most likely exists and can be in various locations such as the disks, network, data server, and so forth.

### Disk and I/O

The disks need to be monitored to ensure that they are not causing an I/O bottleneck. This could be the result of paging operations caused by low memory, high activity from output image generation, large amounts of data access, and so forth. If the disks are bottlenecking on the Map Server or on any other server in the system, this will cause wait time that will not allow you to take full advantage of CPU resources, regardless of how many threads are added. Adding more disks, tuning the operations system, and tuning the disk subsystem can help alleviate disk contention issues.

### Memory

Adequate memory should be made available for published map configurations. Be sure to consider all the variant map configurations that will be published and not only the map configuration under test. Memory consumption is based on the number of processes and threads configured for published map configurations. With modern computer systems, memory has become affordable, and several gigabytes are usually enough to support GIS mapping operations.

### Network

Having plenty of CPU resources to generate maps requires adequate network resources to access data and to serve Web pages with images to the user community. Map Servers should be connected to data servers with 100 Mb or faster connectivity. If file-based data is used from a file server, a gigabit connection may be required between the map servers and the data

server. Output image size plays a large role in network performance and should be kept to a manageable size, for example, in the 25–75 KB range. At some point during stress testing, an additional test should be performed that includes the actual user network connecting the Web server to the users. Testing from a 100 Mb internal LAN may work well, but how will it work over a T-1?

## 5.0 ArcIMS Stress Testing

The following section addresses stress testing of an ArcIMS test application. Before testing can commence, a test application must exist. At some point, you will want to test the actual application under development, but prior to that, a template-based application can be built and deployed for test purposes, which will allow you to become familiar with the test methodology, and so on. The following sections will guide you through the application creation, system tuning, and test and system monitoring processes.

### 5.1 Application Creation and Configuration

With ArcIMS Designer, create a Web site using a published map configuration and overview map configuration. This type of application can be said to represent a typical ArcIMS viewer application. Choose the defaults for all the application parameters such as available tools, scale, and so on. Ensure that the application is tuned properly including scale restrictions, proper symbolization, and so forth.

By default, the output image size will vary, depending on the browser display size. Since output image size greatly impacts performance, the output image size should be standardized so it will not vary as different tests are recorded (unless that is the goal of the test). ESRI's Enterprise Systems Lab (ESL) performance testing has standardized based on an image size of 600 x 400, which has been reported to be a fairly typical size. To modify the output image size to a fixed size regardless of the browser display area, perform the following:

Open MapFrame.htm and find the section containing the following parameters:

```
var mWidth = getMapWidth();  
var mHeight = getMapHeight();
```

Change them as follows:

```
var mWidth = 600;  
var mHeight = 400;
```

Test the Web site to ensure the size is fixed at a 600 x 400 size. This can be validated using the operating system to examine the properties of the output image file.

### 5.2 Recording the Script

Prior to recording the script, ensure that the output directory is clear of any previous image files. Also, stop the Tasker service to prevent it from deleting the output images that will be created during script recording. It is a good idea to copy the output images created from the

script recording to a backup location so they can be recopied to the output directory as needed. Further, the output directory should be cleared out occasionally to avoid accumulation of large amounts of image files.

Start ACT and perform the following to record a test script:

1. Choose Action and choose New Test.
2. In the wizard, choose Record a new test.
3. For Script Language, leave the default of VBScript.
4. Choose Start Recording and wait for a Web browser to appear.
5. Enter the URL to the entry point of the Web site.
6. Record the various steps of the work flow (10 pans, for example).
7. Once completed, choose Stop Recording.
8. Enter a test name.

### 5.3 Script Validation

Access the test script properties and adjust the number of users, the test duration, and the warm-up time. It is wise to start with a single user and short durations (one minute) while validating the script. Warm-up time can be 15–30 seconds to start, which allows the system to stabilize while initially processing transactions, and so on.

Initially, set the test duration property to one iteration and run the test. Once the test finishes, examine the output directory and ensure that 20 new images from the primary map configuration were properly created with the correct image sizes and output image type. In ACT, examine the test results and choose Requests from the Report pull-down menu (top right of the ACT results display). Scroll down the list of requests and click the `com.esri.esrimap.Esrimap` POST statement. The number of requests presented should be greater than the number of steps recorded in the script. For example, if 20 map pans were recorded, the number of POSTs might be 24. What is needed here is the difference between the number of steps recorded and the number reported, which, in this case, is four. Note this Single Iteration POST Delta value, because it will be used later. Also check the test results for any errors. Although errors do occasionally occur, they should be less than 1 percent of the total number of transactions.

### 5.4 Calculating Transaction Throughput

Once the script has been validated, set the script property back to run for a specific duration—one minute. Start the script and monitor the CPU utilization on the ArcIMS servers. Chances are, a single user running in batch mode will only utilize a portion of the CPU. This is where tuning comes into play, and the number of users and available Map Server threads need to be adjusted to take advantage of the available CPU resources. Once the test finishes, examine the results for errors and choose Requests from the Report pull-down menu to retrieve the total number of requests. Then use the following equation to determine throughput in transactions per hour.

$$\text{Throughput} = \frac{(\text{Esrimap POSTs} - (\text{Test Iterations} \times \text{Single Iteration POST Delta})) \times 60}{\text{Test Duration (min)}}$$

Note that this method has been shown to work for typical map viewer applications, but another method may be required for other application types. For example, a single-user test was executed for two minutes on a dual-CPU, 3.2 GHz server, which has a SPECint\_rate2000 of 28.2. The POST results were 321, the number of test iterations was 25, the single post iteration delta was three, and the Map Server CPU utilization was 25.4 percent.

$$\text{Throughput} = \frac{(321 - (25 \times 3)) \times 60}{2} = 7,380 \text{ TPH}$$

This method is fairly accurate and is much more appealing than attempting to count the number of images in the output directory, which becomes skewed because of warm-up time and in-process requests once the test stops. It is also more accurate than merely dividing the total number of requests during the test by the number of requests per iteration. It must be stressed that ACT does not always differentiate between a valid response and an error and errors can bias the results making it appear that the system is achieving higher throughput than expected. It is a good practice to cross-check the number of ACT reported requests with the total number of output images as a further test validation step.

## 5.5 Calculating Map Service CPU Time

Next, the Utilization Law can be used to determine the average map service CPU time (MST). This value can then be used to estimate the maximum potential throughput of the Map Server, assuming no other bottlenecks exist. So for the same example and after converting throughput from TPH to TPS (7,380/3,600), use the following equation:

$$\text{Map Service CPU Time} = \frac{.254 \times 2}{2.05} = .248 \text{ seconds}$$

Additionally, dividing 3,600 by the service time, then multiplying by the number of CPUs/Cores should provide an estimated peak throughput for the Map Server, which, in this case, is 28,346 TPH. Chances are, some service time degradation will occur as the system is ramped up and the Map Server threads begin to compete more and more for system resources. In addition, it is not always possible or desirable to reach 100 percent CPU utilization because of other system constraints, and so forth. For these reasons, it is rare to reach the theoretical peak throughput based on extrapolated single-transaction performance. However, in a test environment, aiming for 90 percent of that value is a reasonable expectation. One of the key reasons for measuring map service time at low utilization is to determine how much service time degrades as the system is loaded.

## 5.6 Calculating Web Browser Response Time

Web browser response time per iteration is available from ACT on the overview page of the test results and is identified as "Average time to last byte per iteration (msecs)". To convert to response time per transaction, divide this value by the number of transactions per iteration and move the decimal three places to the left (divide by 1,000) to convert to seconds. For the sample test, ACT reported Average Time to Last Byte per Iteration (msecs) as 4,342.60. Since the test included 10 transactions per iteration, average response time was .434 seconds.

## 5.7 Script Execution and Configuration Tuning

In ArcIMS Administrator, configure the number of Image Server instances (threads) based on two instances per Map Server CPU, which is a good starting point for Map Server tuning. Increase the ACT user count to at least match that value or go beyond it by one or two users to simulate requests waiting in the application queue. The more users waiting in the queue will not necessarily increase throughput but will increase response times. For example, for the dual CPU Map Server system under test, an Image Server Virtual Server with two spatial server processes were used with two instances each (a total of four image service threads) and the number of ACT users was set to four. For the sample, the test results showed 844 POSTs and 67 iterations with 69.0 percent CPU utilization on the Map Server.

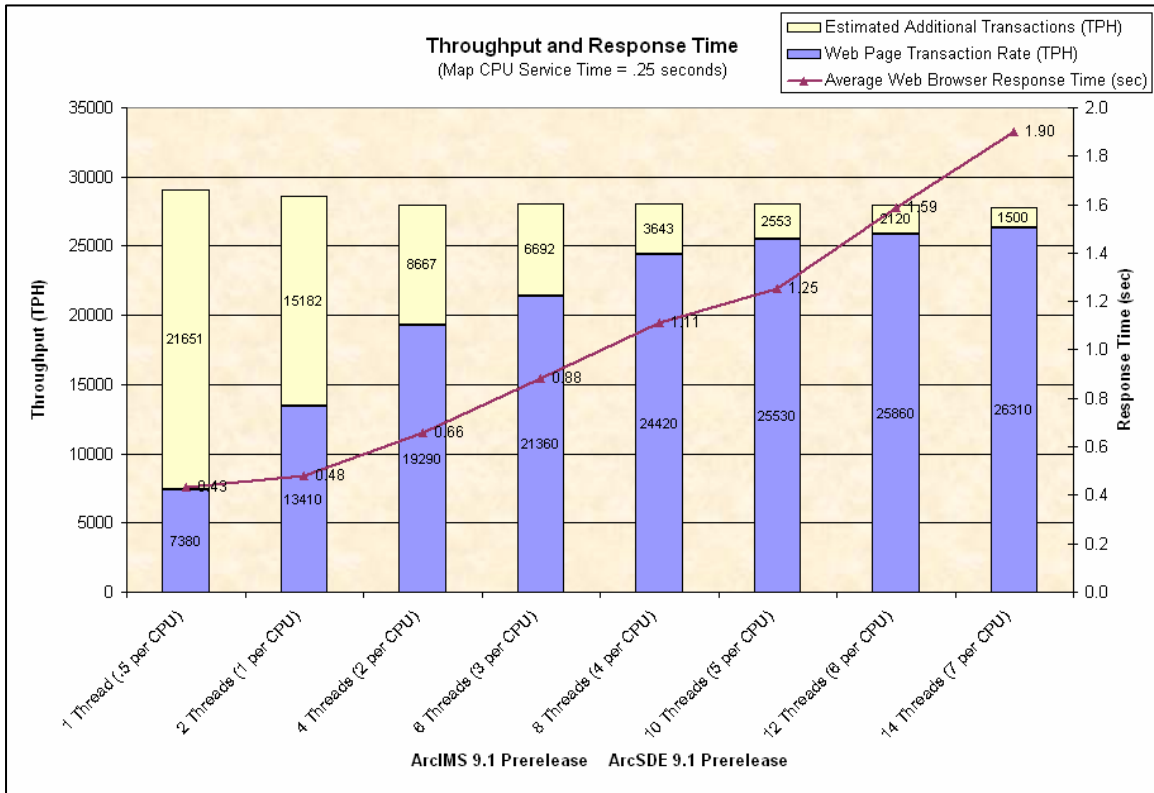
$$\text{Throughput} = \frac{(844 - (67 \times 3)) \times 60}{2} = 19,290 \text{ TPH}$$

$$\text{Map Service CPU Time} = \frac{.69 \times 2}{5.36} = .257 \text{ seconds}$$

$$\text{ACT Response Time} = .656 \text{ seconds}$$

Sixty-nine percent utilization is not bad, and might be acceptable for an operational environment since it represents a good compromise between throughput and individual response times, but with 31 percent remaining CPU capacity, more transactions are available. So the number of concurrent ACT users and threads could be increased to drive additional throughput. However, as you can see from the first two examples, although service time remains flat, response times begin to increase. Figure 1 depicts throughput, estimated additional throughput, and response time for the sample test series. For these tests, the response time is actually conservative, since the user requests were throttled and the application queue was managed at a steady state of zero because the number of users always matched the number of available threads. In a real user environment, users arrive in a random fashion and, therefore, an average queue length is probable during peak loads, which will further elongate response times.

**Figure 1**  
**Throughput and Web Browser Response Time**



The ratio of the blue and yellow stack bars effectively represents Map Server CPU utilization. It is clear that after eight batch users (no thinking time) hitting eight available threads, the system is nearly 90 percent utilized (87.0%, to be exact) and at that point, adding threads provides minimal additional throughput, although response times continue to increase.

In this case, increasing the amount of threads per CPU beyond four is fine for stress testing, but it will begin to severely impact user response times, since the increased number of threads causes more CPU queue time as threads compete for CPU resources. Further, additional applied load will yield little additional throughput and should only be done to push the system as part of the stress test and not to determine the optimum operational level. The balance of peak throughput per system and user response times is an ROI, performance, and productivity balance.

Once tuning is finalized, the script run time should be set to run for at least 10 minutes to minimize performance fluctuations and an adequate warm-up time, for example, 30 seconds, should be used as well.



## 6.0 ArcGIS Server Stress Testing

Many of the concepts that relate to ArcGIS Server testing are identical or similar to ArcIMS stress testing. When appropriate, references will be made to ArcIMS test sections to avoid redundancy. The following discussion relates to testing of the ArcGIS Server 9.1 release. Future releases will require a different test methodology as the technology progresses.

### 6.1 Application Creation and Configuration

In an application development environment, such as Visual Studio .NET, create a new project using a Visual C# map viewer template. Retain all the Web objects within the application. Set the primary map and overview map objects, Host and ServerObject, to point to the appropriate host and published map services.

As with ArcIMS, the output image size should be standardized to 600 x 400. Again, this is not necessary; it is simply a good image size to test with. To change the image size, set the Height parameter in the Layout section to 432 and the Width to 632. These numbers are slightly greater than the dimensions but are required to generate the output image at the correct size. This can be validated by using the operating system to examine the properties of the output images. Changing the output image size will cause the pan buttons, scale bar, and north arrow objects on the display to be in the incorrect position and will require you to readjust them accordingly.

Configure the impersonation object as required to establish security for the application. Also, ensure that the associated map services are tuned properly including scale restrictions, proper symbolization, and so forth.

ArcGIS Server provides a method for using MIME to stream the output image to the client and, therefore, it is not necessary to output the images to an output directory. This is the default configuration for the map templates. However, using output images instead of streaming during stress testing provides a method to validate that the images are being created correctly. This is easily accomplished by changing the UseMIMEData parameter under the Output section to False. Ensure that the associated ArcGIS Server object is configured properly for output images as well.

To ensure proper error reporting when testing ArcGIS Server applications, some changes should be made to the Web site application. You need to change the application code so it will not redirect to file error.aspx. When it redirects, ACT will record it as a successful request when it is actually an error. Sometimes this results in multiple redirects leading to the status code 301. It is clearer to return error status 500 instead of redirecting so you can detect how many true errors are being received.

In Web.config, set the Custom Errors parameter from RemoteOnly to Off (case sensitive), for example:

```
<customErrors mode="Off" defaultRedirect="ErrorPage.aspx">
</customErrors>
```

Find the callErrorPage section in Default.aspx.cs and replace it with the following:

```
private void callErrorPage(string errorMessage, Exception exception)
{
    Session["ErrorMessage"] = errorMessage;
    Session["Error"] = exception;
    Page.Response.StatusCode = 500;
    Page.Response.Output.Write("**Error Message: " + errorMessage + "
Exception: " + exception);
    //Page.Response.Redirect("ErrorPage.aspx",true);
}
```

Find the Application\_Error section in Global.asax.cs and replace it with the following:

```
protected void Application_Error(Object sender, EventArgs e)
{
    Exception exception = Server.GetLastError();
    if (exception.InnerException != null)
        exception = exception.InnerException;
    Server.ClearError();
    Session["ErrorMessage"] = "Application_Error";
    Session["Error"] = exception;
    Response.StatusCode = 500;
    Response.Output.Write("**Error from Global.asax.cs**" + exception);
    //Response.Redirect("ErrorPage.aspx",true);
}
```

If using IIS as the Web server, set the cookieless parameter in the SessionState section in Web.Config to False. The template applications do not use cookies, and the workaround for a test environment is to set this to False.

## 6.2 Recording the Script

Prior to recording the script, ensure that the output directory is clear of any previous image files. It is a good idea to copy the output images created from the script recording to a backup location so they can be recopied to the output directory as needed. To keep the images in the output directory from being deleted by ArcGIS Server, enable the read-only file attribute on the files.

Start ACT and perform the following to record a test script:

1. Choose Action and choose New Test.
2. In the wizard, choose Record a new test.
3. For Script Language, leave the default of VBScript.
4. Choose Start Recording and wait for a Web browser to appear.
5. Enter the URL to the entry point of the Web site.
6. Record the various steps of the work flow (20 pans, for example).
7. Once completed, choose Stop Recording.
8. Enter a test name.

## 6.3 Script Validation

Access the test script properties and adjust the number of users, the test duration, and the warm-up time. It is wise to start with a single user and short durations (one minute) while validating the script. Warm-up time can be 10–15 seconds to start, which allows the system to stabilize while initially processing transactions, and so forth.

Initially, set the test duration property to one iteration, then run the test. Once the test finishes, examine the output directory and ensure that 20 new images were properly created from the primary map service with the correct image sizes and output image type. In ACT, examine the test results and choose Requests from the Report pull-down menu (top right of the ACT results display). Scroll down the list of requests and click the GET statement for default.aspx. There should be a single request for it. Immediately following that should be a POST statement for default.aspx, which should have 19 requests. Adding these two numbers together should equal the total number of steps in the test, or 20, in this case. Also, check the test results for any errors. Although errors occasionally occur, they should be less than 1 percent of the total number of transactions.

## 6.4 Calculating Transaction Throughput

The concepts for determining throughput for ArcGIS Server are the same as with ArcIMS (see section 5.4), except the following equation should be used to determine transactions per hour. Note that this method has been shown to work for typical map viewer applications, but another method may be required for other application types.

$$\text{Throughput} = \frac{(\text{default.aspx GETs} + \text{default.aspx POSTs} \times 60)}{\text{Test Duration (min)}}$$

It must be stressed that ACT does not always differentiate between a valid response and an error and errors can bias the results making it appear that the system is achieving higher throughput than expected. It is a good practice to cross-check the number of ACT reported requests with the total number of output images as a further test validation step.

## 6.5 Calculating Map Service CPU Time

See section 5.5.

## 6.6 Calculating Web Browser Response Time

See section 5.6.

## 6.7 Script Execution and Configuration Tuning

See section 5.7.

## **7.0 Concluding Remarks**

Web stress testing is as much art as it is science. It often takes patience and a little creativity to turn the knobs in the correct direction and to uncover and resolve system bottlenecks. The test methodology presented here should provide a good starting point for evaluating ArcIMS and ArcGIS Server application performance. Performance predictions can be made (hypotheses) as to how the system will perform, and the ACT test results can be used to validate the predictions.



## 8.0 Support

Enterprise GIS system design is addressed in the *System Design Strategies* white paper at <http://www.esri.com/library/whitepapers/pdfs/sysdesig.pdf>. For answers to additional GIS capacity planning and solution questions, contact ESRI Systems Integration at [sihelp@esri.com](mailto:sihelp@esri.com). For technical support, contact ESRI Technical Support at <http://support.esri.com>.