Title of Paper
Building a High-Availability ArcIMS Hosting Infrastructure for NYC

Author's Name
Mario Gouvea

Abstract

New York City's Department of Information Technology and Telecommunications (DoITT) hosts a variety of e-government web based applications to serve the public, some of those are ArcIMS applications that DoITT hosts for city agencies: My Neighborhood, My Neighborhood Statistics and EMOLs. Due to the increasing demand for an ArcIMS hosting infrastructure for New York City agencies and the need for a robust environment to host emergency applications, DoITT ventured in upgrading its aging equipment and moving it in-house. The result is a 24x7 scalable environment consisting of more than 25 Sun Microsystems servers, 10 of which are in production and the remaining are part of the development, test, staging and disaster recovery environments. This paper describes the process of designing a high-availability ArcIMS hosting infrastructure for New York City starting from the architecture design to the sizing of servers, integration, configuration, and the deployment of applications with zero downtime.

Introduction

The process of designing a new ArcIMS hosting environment for the City of New York was triggered by issues encountered with the previous environment which hosted applications such as EMOLS (Emergency Management Online Locator System), My Neighborhood, My Neighborhood Statistics and the Department of Sanitation Collection Schedule Application.

That environment was originally designed to host the EMOLS set of applications for the Office of Emergency Management but once it was realized that GIS could be used to effectively aid New York City's e-government initiatives other applications were developed and later deployed into those same servers.

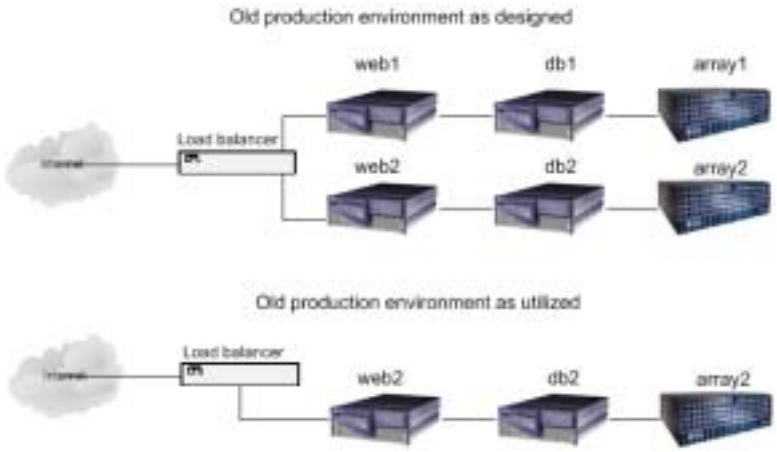

Figure 1 - Old ArcIMS hosting environment

The system was architected to be somewhat redundant, it consisted of five servers, four of which were production servers and one used for development and test. Figure 1 depicts the system architecture and Table 1 describes the software installed in those boxes. Due to the fact that the development server was lost during the September 11 attacks (it was located at World Trade Center 7) two of the remaining four servers had to be used as development servers leaving only two servers to be used as part of the production environment.

Table 1 - Old ArcIMS hosting environment configuration

Item	Software	Hardware
Web Server/ArcIMS Server (2)	Solaris 7, Sun iPlanet 4.1, ArcIMS 3.1	Sun Enterprise 420R 4-450MHz CPU 2 GB RAM
Database Server/ArcSDE Server (2)	Solaris 7, ArcSDE 8.1, Oracle enterprise Edition 8.1.7.4	2-450MHz CPU 4 GB RAM
Storage Array(2)		Sun StorEdge A1000 6x18.2GB

The system operated in that crippled configuration for several months until a new application was released and received a lot of media attention. During that time the map/application server's utilization reached 100% for several hours and triggered the upgrade of the database server from 2 CPUs to 4CPUs which alleviated the load on the system slightly, the bulk of the load was still on the map server. It was clear that it was time to redesign the hosting environment and address several of the deficiencies of the existing system.

Defining the requirements

The design of the new ArcIMS hosting environment started in the end of the first quarter of 2003 and was completed in the beginning of the fourth quarter of the same year. The requirements for the project were clear: to design a robust, highly available and scalable ArcIMS hosting environment that could be used to host the existing applications during normal utilization, peak emergency mode as well as accommodate the needs of New York City agencies wanting to publish their own ArcIMS application on the Internet. What is more, the environment had to be set up at one of DoITT's data centers instead of a commercial hosting provider. Another requirement was to support a full software development life-cycle, which demanded the build of the following environments: development, test, staging, production and disaster recovery.

DoITT was already planning to migrate the NYC.GOV portal infrastructure from a hosting provider to one of its data centers; it had started to build all the necessary network infrastructure and physical rack space for the new hardware. The GIS migration project was combined with the existing portal migration project at a very early stage which led to a collaborative effort and the successful completion of the joint project. The joint project allowed the organization to effectively share hardware and human resources.

Sizing the servers

ESRI consultants were brought on site to provide other consultants and DoITT staff members with a two day training on systems design for ArcIMS. The training provided DoITT with the necessary tools to size and design a highly available and scalable system. The session also served as a forum for the discussion on horizontal versus vertical scalability and how the components of an ArcIMS/ArcSDE installation should scale.

Information collected from past utilization of the applications being hosted at the existing environment proven to be a very valuable resource in estimating the capacity of the new system. Data from the peak utilization days, past statistics from the web server log files, results from application stress test along with information obtained during the two day system design training and the ESRI's System Design white paper are examples of information utilized to size the new servers.

The existing environment was built using Sun Microsystems hardware and the Solaris operating system, in order to support the existing applications and leverage the existing server management infrastructure, software licenses and human capital the natural path chosen was the adoption of the same hardware platform and operating system for the new build (utilizing the current version of the processor and operating system). Table 2 summarizes the hardware and software utilized at the time the environment was built.

Table 2 - New ArcIMS hosting environment configuration

Item (quantity)	Software	Hardware
Reverse Proxy Server (2)	Solaris 8, Sun ONE Web Proxy Server	Sun Fire v210 1 1GHz CPU 1GB RAM
Web Server (2)	Solaris 8, Sun ONE Web Server v6.0, ESRI ArcIMS 4.01	Sun Fire v210 2 1GHz CPU 2GB RAM
Map Server (4)	Solaris 8, ESRI ArcIMS 4.01	Sun Fire v240 2 1GHz CPU 4GB RAM
Database Server (2)	Solaris 8, ESRI ArcSDE 8.3, Oracle enterprise Edition 9iR2	Sun Fire 15K 8 1GHz CPU 16GB RAM
Storage Array(1)		Sun StorEdge 6910

The new ArcIMS hosting environment architecture is derived from one of the high availability ArcIMS designs available through ESRI's consulting services and the Systems Design whitepaper. The differences between the original generic design and the modified version are described below.

Fulfilling the high availability and scalability requirements

One of the key requirements of a high availability environment is to facilitate the management of applications, ensure system functionality during planned or unplanned downtime and ensure fast recoverability in the case of a hardware failure. By adopting a highly available architecture an organization can automate the process of testing, deploying and managing the load on different components of a system.

The main purpose of the ArcIMS hosting environment of New York City is to deliver spatial information services to a wide user audience, some of the information services delivered are of an emergency nature which means that ensuring the integrity and availability of the underlying data is extremely important. Data integrity can be achieved at different levels: at the relational database management system, at the hardware level, at the storage level through the use of RAID (Redundant Array of Inexpensive Disks), at the network level through the use of redundant network cards and load balancers and at the application level as well through the use of application level load balancing and error checking.

Database tier (production environment)

The solution architected at DoITT combined several of the high availability components described above. The database tier consists of a two node cluster set up in an active/passive configuration with a shared storage array (Sun StorEdge 6910). Each of the cluster's nodes consists of a Sun Fire 15K domain running Solaris 8, Oracle 9iR2 Enterprise Edition, ESRI ArcSDE 8.3 and Veritas Cluster Server. While the active node is being utilized by the ArcIMS hosting environment the passive node works as the active node for other projects ensuring no resource is idle or underutilized. Each of the nodes is fully redundant: it has two network cards and is fully scalable, additional CPU resources can be made available dynamically if needed. All systems have redundant power supply.

In order to enable SDE to work with Veritas Cluster Server the SDE startup script had to be slightly modified and some of the SDE settings had to be changed in order to enable clients to transparently recover their connections in the case of a failover from one node to another. The database connection pool settings of some of the existing web applications had to be modified to automatically reestablish the connection to the database server in the case of a failover. During integration tests full database and SDE failover took less than a minute to complete.

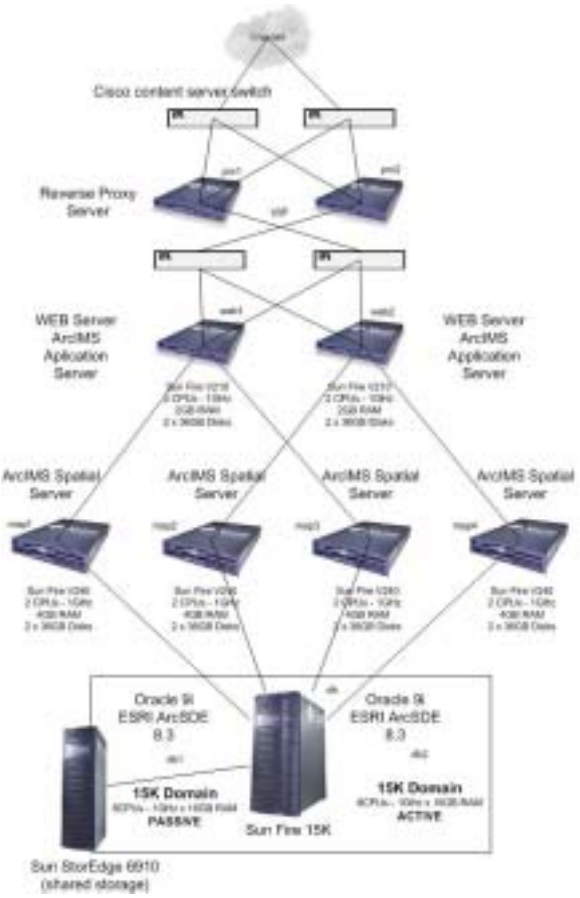

Figure 2 - PRODUCTION Environment

Application/Map server tier (production environment)

The map server tier, consists of four map servers hosts, each configured to be able to run two instances of ArcIMS each instance running at its own port. These ArcIMS map servers are managed by two ArcIMS application servers running on two hosts that also run Sun ONE Web server, a web server and Java servlet container where the actual applications are hosted. Each of the map servers are configured with 4GBs of RAM and 2 CPUs to maximize the processing power needed to generate map images via an ArcIMS image service.

Web server tier (production environment)

The web server tier consists of two hosts running Sun ONE Web Server v6.0 as the web/java application server and ArcIMS 4.1 as ArcIMS application server. Each ArcIMS Application Server communicates with two map servers instead of all the four map servers as it would normally be expected in order to achieve a highly-available architecture. We tested different configurations during the stress test of the new environment and found that when each ArcIMS application server communicates with all the four map servers the system is not able to reach its peak performance. On that configuration each map server host has two independent instances of the ArcIMS map server running that are not aware of each other and they usually compete for CPU resources on a non efficient manner, what is more, on that configuration, each of the ArcIMS application servers is not aware of the amount of requests the other application server is sending to the same map servers it is utilizing. That configuration works under normal load levels but it causes errors on the web tier when we simulated a large number of concurrent users hitting both ArcIMS application servers. We then tried a different configuration (depicted on Figure2) where each ArcIMS application server has two map servers fully dedicated to itself so there is no competition for CPU resources at the map server level. We re-executed the same stress test and achieve much better results without errors.

Network

The system was designed to be fully redundant in terms of network access in order to enable it to recover from an isolated network failure transparently. All the hosts used for the production environment are configured with two network cards each with its own IPMP address and a logical IP address for the host.

In addition to the network card level redundancy the system was designed with dual load balancers at the proxy server and web server levels. The load balancers were configured with session stickiness enabled in order to ensure visitors to the applications remain connected to the same web server on their subsequent connections after they first connect. Without session stickiness site visitor's would not be able to see the map image generated by ArcIMS in a consistent manner. If a user connects to 'web1' via the load balancer's virtual IP 'web' and the load balancer is not configured to support session stickiness the

web application will send a request to ArcIMS for a map image, the image will be generated in web1's output directory but when the user's web browser request the image generated it will be pointing to the virtual IP address of the web load balancer 'web' which may be translated to either 'web1' or 'web2' depending on the algorithm used by the load balancer, therefore there is a chance that the browser will look for the map image on the wrong web server, which is unacceptable.

Supporting a full life-cycle software development group

The process of updating a web application on the old ArcIMS hosting environment was cumbersome and required a lot of manual steps in order to ensure the application was available during the update cycle. Due to the fact that the development server was lost on September 11, 2001, two of the four production servers were used for application development and test. Whenever a new application was launched, it would first be deployed and tested on the 'development' environment which would then become a production system while the other set of servers would become 'development' servers and go thorough the application update process. The switching from development to production was performed manually by updating the pages containing links to the applications. The whole process was strictly controlled by a rigorous change management protocol and it was only performed if change requests were approved by the management group.

In order to improve the process of developing testing and deploying new applications a set of environments were designed: development, test and staging.

Development environment (DEV)

The development environment was designed to provide developers with a lot of flexibility in developing and unit testing their applications. The environment consists of two servers, one running Sun ONE Web Server, ArcIMS application server and ArcIMS map server and another running Oracle 9iR2 and ArcSDE 8.3.

Developers have full permission to deploy WAR files, restart the web server, create image services, restart ArcIMS and restart ArcSDE. The creation of database users and schema is restricted to DBAs following a documented process.

Test environment (TST)

The test environment is an exact replica of the development environment, it consists of a server running Sun ONE Web Server, ArcIMS application server and ArcIMS map server and it shares the same database server hardware. There is a dedicated Oracle instance and a dedicated ArcSDE instance for the test environment which is completely independent of the instances for development running on the same server. ArcSDE runs on port 5151 on the development environment and port 5161 on the test environment.

The test environment is used to test the application deployment and to write and record the functional test scripts. The functional test scripts are later executed against the staging environment to catch application errors related to the underlying differences between the test environment and staging environment.

Staging environment (STG)

The staging environment closely mimics the configuration of the production environment; the only difference is that it utilizes two map servers instead of four map servers. Staging is used to certify that an application is ready for deployment in production. See Figure 3.

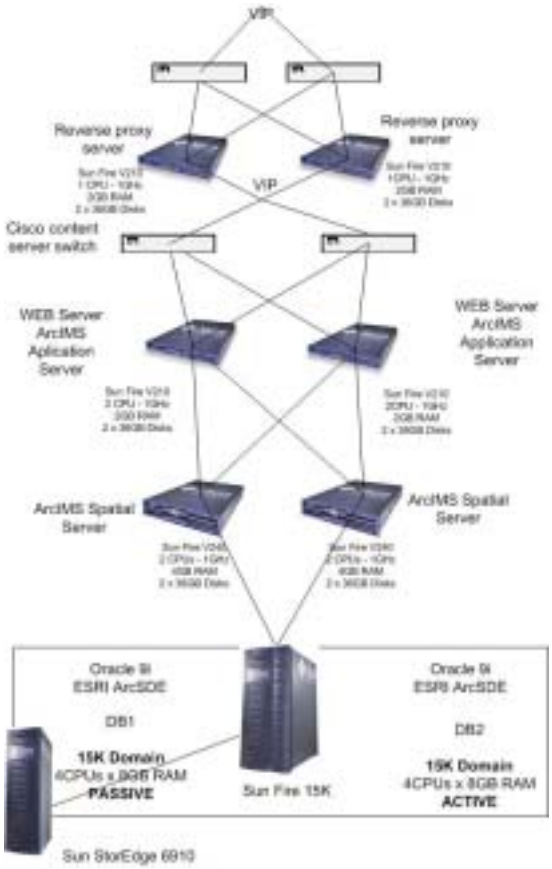

Figure 3 - Staging environment

Staging is the only pre-production environment that enables analysts to detect problems related to the application configuration. Because the ArcIMS application server runs on different servers than the map server it is possible to catch defects that wouldn't be noticed on test or dev. One example of such issues is the location of icons within an application, developer may choose to place them on the web server or on the map server depending on how their code work, but because the map server and web server reside on the same server on dev and test the developer could eventually place the icons on a location where they would be inaccessible by the map server on the staging and production environment once the application in migrated. All these issues can be easily

avoided by properly setting up automated application configuration and deployment processes that take the differences of all the environments in consideration.

Applications that are updated on a regular basis utilize the staging environment for content staging and approval before data is updated in production.

This environment is also utilized to perform several different types of tests and security scans on an application before it is certified to go into the business owner review phase and subsequently into production.

Disaster Recovery (DR)

The disaster recovery environment is a scaled down version of the production environment, it consists of one web/ArcIMS server, two map servers, a reverse proxy server and a database server. It is located on a different physical location than the other environments and is activated in case of a disaster or maintenance of the production environment. The disaster recovery environment can be enabled by following detailed a dr cutover procedures document which contains the details on how to make the necessary DNS changes to point to the virtual IP address of its reverse proxy server.

Systems Integration

The process of building the new environment consisted of multiple tasks being performed by several different groups, some were responsible for the overall project management others for procurement, network design and setup, server hardware setup, OS configuration, software installation, system configuration, system testing, application migration, tuning, application testing, security scans and many other tasks.

Once the servers were procured, setup, configured, connected to the network and individually tested the process of installing the application software and migrating existing applications from the old ArcIMS hosting environment to the new system started.

Testing

Testing was essential in validating the system configuration, performance tuning the system, adjusting memory settings, verifying application functionality and determining the system's maximum capacity. The following types of tests were executed before the system went live:

- Stress tests – system tuning
- Baseline tests – determine baseline response times
- Performance tests – determine application response times under load
- Breakpoint tests – determine application break point
- Functional tests – application functionality

Stress tests – system tuning

The first set of tests was executed in order to tune the web server, the ArcIMS application server configuration, SDE and Oracle settings.

Several cycles of stress tests and tuning were executed; they were very useful in tuning the web server's JVM memory settings, session timeout settings, deciding on the best way of connecting the ArcIMS application server to the map servers and configuring the ArcIMS virtual server settings. Each change of configuration was documented prior to rerunning the same tests to determine the impact of the changes.

The tests were executed utilizing Mercury Interactive's LoadRunner, an extremely powerful testing tool which is capable of emulating hundreds of concurrent users hitting a web application following a pre-determined set of steps through the application. The software also collected statistics from different counters through agents running on different servers. While running stress tests we were able to collect several sets of statistics on counters from Oracle, iPlanet (Sun ONE Web Server) and generic Unix counters. These statistics were used on analyzing the results from tests and determining the best plan of action for optimizing the system configuration.

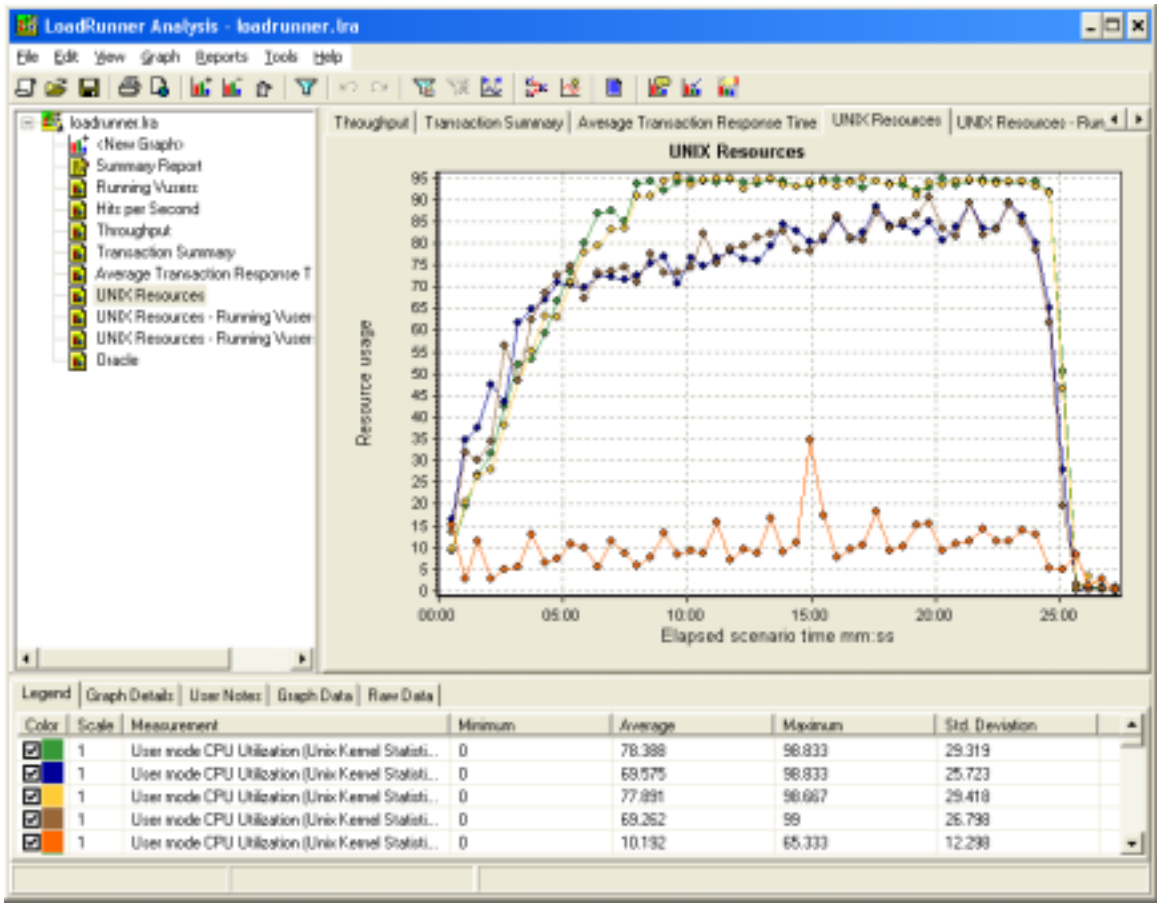

Figure 4 - CPU Utilization - 150 virtual users

Figure 4 shows the results of one of the early stress tests executed during the tuning stage. The test simulated 150 virtual users hitting the My Neighborhood Statistics application for 27 minutes, the ramp up rate was 4 users for every 30 seconds. The test was executed against one web server connected to four map servers hitting one database server. The chart shows the CPU utilization for the four map servers (close to 100%) and the web/ArcIMS application server (below 20% for most of the test). The cpu utilization for the database server was below 30% during the test (not depicted on the chart). The application response time varies a lot depending on what is displayed on the map, whether the layers were generalized or not, the image format, whether labels are anti-aliased or not, etc. Map generation response times vary a lot within the same application so we defined fixed steps to be able to replicate the same map operation every time the test was executed. During that test several different parts of the application were tested, a pan operation under normal conditions took 4.6 seconds (includes web server processing time, map generation and page download time). During the 150 virtual users test the average response time for the same operation was 11.8 seconds.

The tuning stress tests were executed internally from DoITT's local area network, its purpose at this point was to generate enough load on the server to determine the impact of the load on each of the system's components and tune them accordingly.

After tuning the server and modifying the configuration so each ArcIMS application server connects to two map servers the response time for the same pan operation was reduced from 4.6 seconds to 2.3 seconds under normal conditions (without modifying the application).

The following is a summary of one of the tuning test runs:

Duration: 27 minutes and 14 seconds.
Maximum Running Vusers: 150
Total Throughput (bytes): 1,576,100,267
Average Throughput (bytes/second): 963,976
Total Hits: 523,259
Average Hits per Second: 320.036

Baseline tests

Once the system was tuned we ran baseline tests for each of the applications on the system in order to establish baseline response times that could be used for comparison purposes.

Performance/stress tests

Performance tests were executed from DoITT's LAN to determine the response time of different components of several web applications under different loads. We performed several different test scenarios to collect statistics on how the servers were responding.

The following are some of the test scenarios executed:

- 25 concurrent users, 2 new users every 10 seconds for 1 hour
- 50 concurrent users, 2 new users every 10 seconds for 1 hour
- 75 concurrent users, 2 new users every 10 seconds for 1 hour
- 100 concurrent users, 2 new users every 10 seconds for 1 hour

In addition to running internal performance tests we also performed tests utilizing a remote load injector test server farm hitting the servers via the Internet to determine the actual response time an Internet users would experience. We re-ran several of the same scenarios.

The following is the summary of one of the stress tests executed against My Neighborhood Statistics application (executed from a server farm on the Internet):

Duration: 1 hour, 46 minutes and 50 seconds.
Maximum Running Vusers: 87
Total Throughput (bytes): 1,220,434,376
Average Throughput (bytes/second): 190,366
Total Hits: 172,961
Average Hits per Second: 26.979

The average response time for the pan map operation under this load was 10.179 seconds. The following chart summarizes the data throughput for this test:

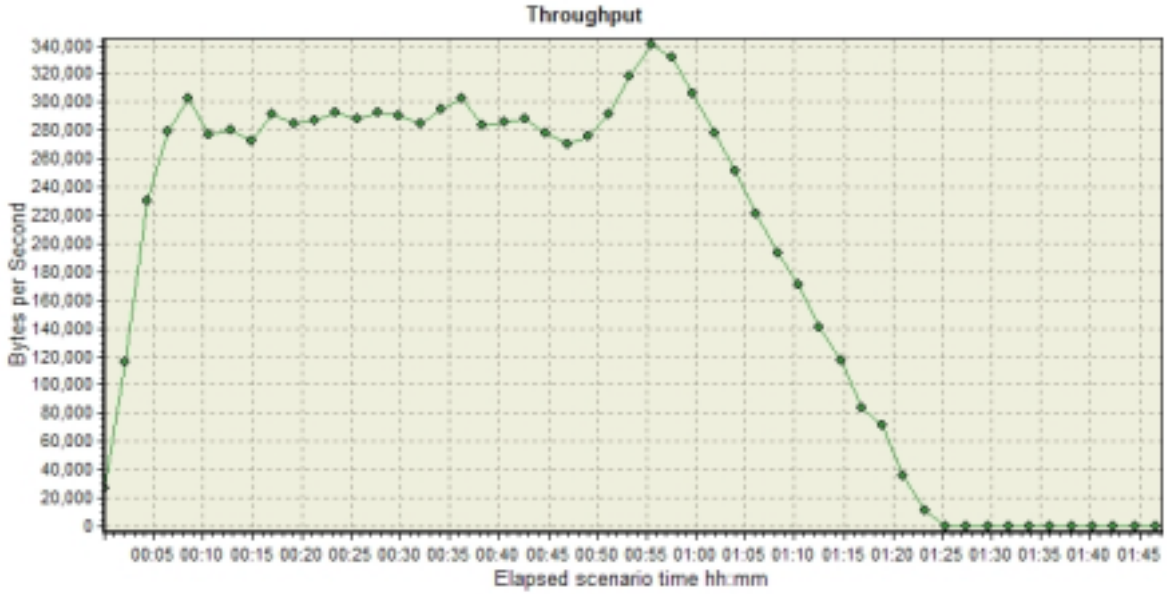

Figure 5 - stress test throughput

Breakpoint tests

Breakpoint tests were executed against the production environment to determine which components of the system would fail under extreme user load. The exit criterion of those tests was to stop the tests as soon as a transaction failed. The following is a summary of the breakpoint test for My Neighborhood Statistics executed from a remote server farm:

Duration: 55 minutes and 47 seconds.
Maximum Running Vusers: 401
Total Throughput (bytes): 2,631,197,263
Average Throughput (bytes/second): 785,901
Total Hits: 374,279
Average Hits per Second: 111.792

The chart below summarizes the throughput during the breakpoint test:

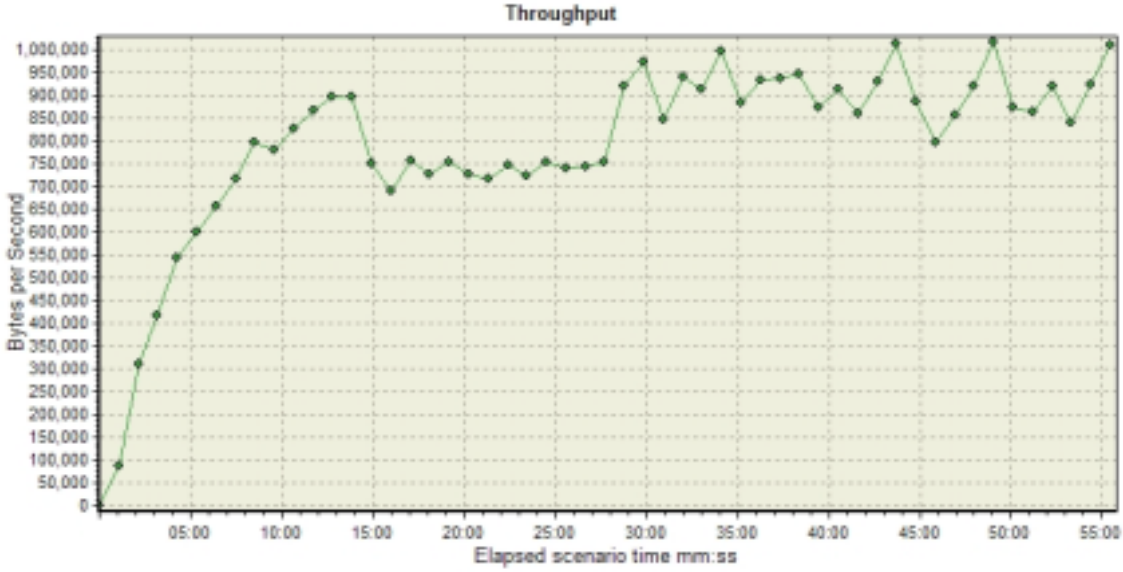

Figure 6 - Breakpoint test throughput

Automated functional testing

Functional tests are crucial in determining whether an application is working as designed after it is migrated from one environment to another. As the number of applications being hosted on the environment increases the process of testing an application manually becomes increasingly more complicated. Fortunately is a wide range of tools can assist on the automation of functional tests. We utilized a tool called QuickTest Professional from Mercury Interactive which can be used to record the steps a user would take when using a web application and record the expected results for each step. Once a test has been recorded in one environment it can be tested against all the other environments.

One of the most useful features we encountered in the functional testing tool utilized is that it supports the creation of bitmap checkpoints which capture a copy of an image during the recording of the script so it can be compared with the image generated at the

test runtime. That feature is especially useful for testing ArcIMS applications and comparing maps generated by one environment with maps generated on another environment. The map image is compared pixel by pixel and any difference will be detected by the testing tool. This feature will detect differences in the configuration of an ArcIMS image services. If one server is generating JPG and the other is generating PNG the tool will display the test results and show the two map images side by side. Figure 7 illustrates a map image from an application, note the missing icons on the right.

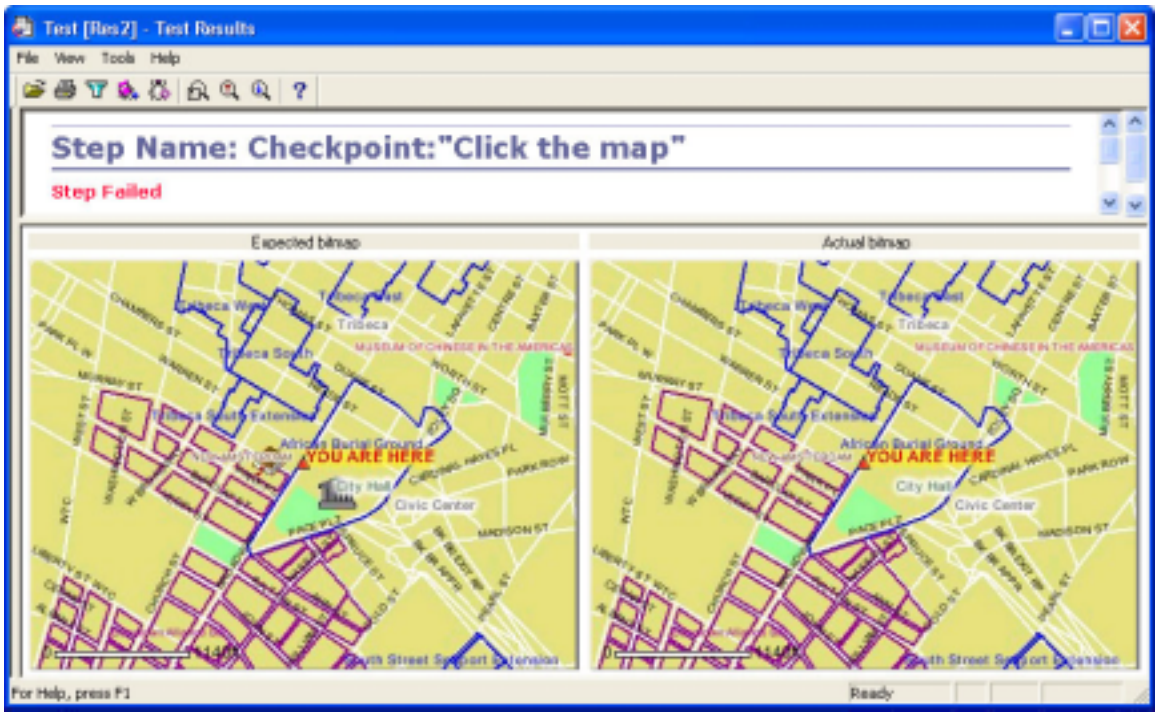

Figure 7 - bitmap checkpoint

Table 3 – Testing software

Purpose	Software
Functional application tests	Mercury Interactive QuickTest Professional
Performance, stress and load tests	Mercury Interactive LoadRunner

Automated application deployment and configuration management

Deploying applications on the original hosting environment was a task that was performed manually by outside consultants; it was susceptible to human errors due to the fact that several application configuration files had to be changed on the two web servers. Different applications were sharing the same servlet container context, the same Java classes' folder and the same database schema. Some applications implemented their own database connection pool. Most of those configuration files were application specific and did not follow any standard which made the task of maintaining them very difficult.

After moving the management of application in-house we started the development of common Ant build scripts to automatically create a deployment WAR file for each web

server on each of the different target environments. These build scripts expect each application web.xml file to follow a standard naming convention for variables that changed from one environment to another. Examples of such variables are the hostname of the ArcIMS Application Server, the hostname of the geocoding server (non-ArcIMS) and ArcIMS port number. The scripts can also automatically copy each of the generated war files to the target environment utilizing scp and deploy them to Sun ONE Web Server utilizing its wdeploy command via sshexec. This simple process increases productivity and reduces the amount of defects drastically, increasing the overall quality of the services delivered via the hosting environment.

Deploying applications with zero down time

The production environment's architecture allows the deployment of applications with zero down time. Once the application has been deployed to staging environment successfully, the process of deploying it to production can be accomplished by transferring the traffic at the web server's load balancer level to only one of the web servers leaving half of the environment (with the exception of the database server) available for the update or maintenance. See figure 7 below. This whole process is strictly controlled via a change management system. Deployments to the production environment can only be performed once change requests have been approved by business owners.

Once the traffic is redirect to web2 as depicted below, changes can be made to the applications on web1 and ArcIMS configuration changes can be performed on map1 and map3. The web server and ArcIMS Application Server can be restarted if necessary without impacting users ensuring business continuity.

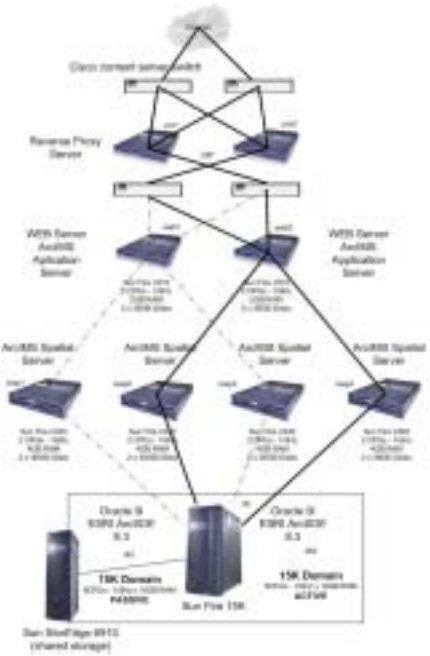

Figure 8 - Application deployment/update mode

Maintenance and update tasks on the production environment that may impact users have to be performed after the traffic is redirect to the disaster recovery environment making the production environment fully available for maintenance work.

Conclusion

The implementation of a high-availability ArcIMS hosting environment enabled DoITT to move out of a reactive mode of managing application availability to a more proactive mode which enabled system administrators, developers and managers to focus their time on providing citizens with a higher quality level of information services.

Acknowledgements

Several individuals and organizations were involved in implementing different aspects of this project.

End Notes

This paper provides a brief overview of the process of setting up a high availability ArcIMS hosting infrastructure for New York City. It does not contain information on the security implementation, configuration details, backup strategy and daily system management.

References

System Design Strategies, An ESRI White Paper – March 2004, Dave Peters, Systems Integration

Author Information (name, title, organization, address, telephone and fax numbers, e-mail address)

Mario Gouvea
Manager, Enterprise Spatial Systems
DoITT – Department of Information Technology and Telecommunications
Citywide Geographic Information Systems
59 Maiden Lane, 33 fl
New York, NY 10038
USA

Tel: (212) 232-1138
Fax: (212) 232-1180

E-mail: mgouvea@doitt.nyc.gov