Portal & Gateway Performance

SunPS eUniversity Performance Escalation Team*

Sun™ONE
Open Net Environment

Sun microsystems

Summary

Hypothesis: The Portal/Gateway is the bottleneck in the eUniversity production system.

This paper sets out findings from investigations conducted to test this hypothesis, to provide some ideas on how that might be rectified (if true) and to determine how capacity might sensibly be added going forward.

We conducted a number of tests using a benchmarking environment designed to emulate the eUniversity production environment in as far as this is possible. The tests were designed to examine the effects on Portal/Gateway perfomance of different hardware configurations, operating system versions and the effect of tuning system parameters.

We concluded that

- The current Portal/Gatway configuration is **not** an inhibitor to current system performance and there is headroom within the current configuration.
- There is a performance benefit to be gained in upgrading the operating system to the latest version.
- There is little performance benefit in tuning system parameters at present.
- There is a linear relationship between the amount of Gateway servers that can be added to the system and the number of users that can be sustained giving a clear upgrade path for the future. A bottleneck will then appear in the networking infrastructure.
- The use of dedicated Secure Socket Layer (SSL) processing hardware is likely to provide additional headroom in the future.
- The Portal/Gatway configuration may warrant further investigation in the context of a more performant version of the application it serves.

The body of this paper covers the details of the tests undertaken and their results and the conclusions that can be drawn from them. This material is presented in a form that should be understood by any technically literate reader. Additional technical material of specialist interest is at the back of the document.

*The team are Nick Cushway, Julz Dawson, Paul Eggleton, Andy Head, Dominic Kay, Tony Marshall.

CONTENTS

Summary 1

Introduction 3

Assumptions & Constraints 10

Splitting Portal and Gateway 12

Libthread, Solaris 9 & GC Tuning 22

Secure Socket Layer (SSL) 25

Conclusions and Further Work 27

References 30

Appendix A - LoadRunner Scripts 32

Introduction

Introduction

This chapter provides a basic outline of what a Portal/Gateway is. Interested technical readers are referred to the manual set which is listed in the References section at the end of this document.

We then briefly reiterate the design of eUniversity's Portal/Gateway infrastructure, again avoiding the level of detail of the project design documentation listed in the references.
The chapter then ends with a brief description of our load generation tool (Loadrunner) and the design of the end-user scenario we used during the tests.

What Is a Portal?

A Portal is the entry point to a set of resources that an enterprise wants to make available to the Portal's users. For some consumer Portals, the set of resources includes the entire World-Wide Web. For most enterprise Portals, the set of resources includes information, applications and other resources that are specific to the relationship between the user and the enterprise. For service providers, the Portal provides a point of entry to customer service applications as well as a controlled content aggregation service.

In general, a Portal enables users to:
- Customize the available data content
- Change how channels are generated
- Control how data is displayed
- Create and manage links to other permitted web sites

Resources can include the use of provider applications and utilities such as mail, file management, and storage facilities.

SunONE Portal

The Portal Server provides a single point of entry to the site for all users. It is at this point that users authenticate with the system. Once identified to the

system the appropriate content and functionality is delivered to the user's “desktop”, based on their role membership and personal preferences. The desktop consists of a number of channels which are distinct regions of the screen, each delivering content from different sources or providing links to different activities.
The desktop and channels are built with JSPs and HTML templates based on well defined
APIs that are supplied with the Portal. This allows rapid development of channels drawing
content from a variety of streams.

The iDSAME APIs allow access to user profile information, stored in the Directory Server.
The Portal can aggregate user roles such that users who have a number of levels of access
rights can receive all the content to which they are entitled without one role overriding
another.

What Is a Gateway?

A Gateway offers browser-based secure remote access to Portal content and services from any remote browser. It is a cost-effective, secure access solution that is accessible to users from any Java™ technology-enabled browser, eliminating the need for client software. Integration of a Gateway with a Portal Server ensures that users receive secure encrypted access to the content and services that they have permission to access.

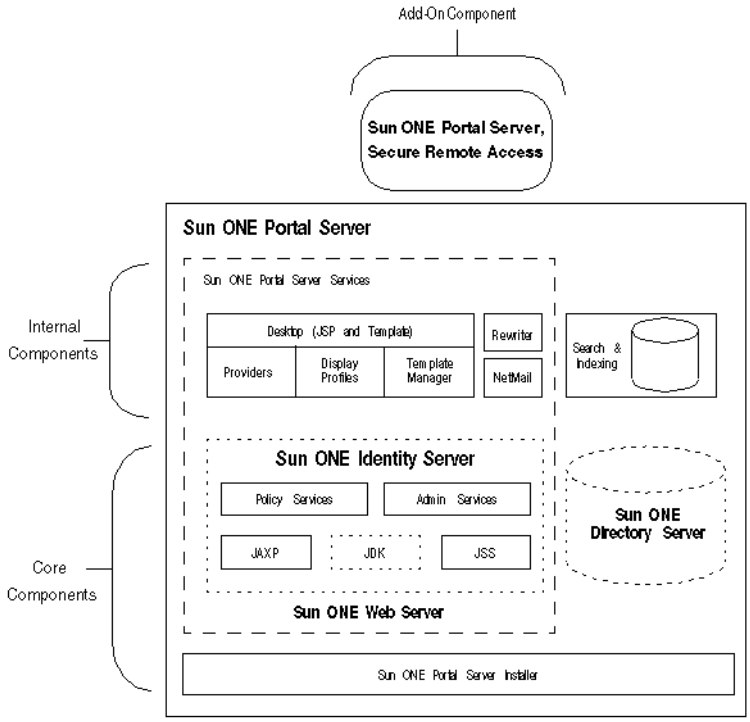
Sun Gateway (Secure Remote Access)

The Sun Gateway is a component, not a product and is known simply as Secure Remote Access (SRA) It acts as a secure reverse proxy to the Portal server. All requests to the Portal server are made through the Gateway and it encrypts all data between the browser and the Portal server. This means that only the SRA and the Mail Multiplexor machines need to have public addresses, increasing the security of the system by reducing the number of externally facing entry points. The SRA can be installed on a separate machine to the Portal server, but currently in the case of eUniversity, due to a lack of machines the SRA is on the same machine as the Portal server.

Portal Gateway Description

Portal Server is a Java application that runs within the SunONE Web Server web container. It is configured using a variety of properties and xml files and administered using the Identity Server console. The Identity Server Console is another Java application that runs in the same web container. Users, roles and services, one of which is the Portal Server, are administered via this console. The Directory Server is an LDAP server and is the persistent store for all the configuration and user data required by the Portal.

The diagram below shows the overall architecture of the system.



eUniversity Design Principles

To meet the solution objectives of the project, some core design principles were established which we outline here.

To achieve the availability and reliability requirement, the architectural approach adopted mandates that there must be no single point of failure in the solution infrastructure. Where feasible and cost-effective, users (particularly paying users) will not be aware of any component failure in the system. At worst, following a failure a user may need to reload a page in a browser window, or re-authenticate to a service. This is particularly true when a user is connecting through an application which maintains stateful data i.e. data which is unique to the user/session.

Each major architectural component is isolated on a separate host system (either physical or virtual) that may be scaled according to the type of service offered by the component. This strategy offers the greatest flexibility in responding to additional user and service demand during the life of the system. Also, solution components can be maintained, upgraded and even replaced in an independent manner.

Scaling of components is generally approached vertically or horizontally based on the scaling capability of the underlying application:

Vertical scaling is the increase of resources within a single or clustered system e.g. CPU, memory, disk, and I/O. This is appropriate for applications which scale well relative to the system kernel. All applications scale vertically to some point beyond which the cost/benefit becomes prohibitive.

Horizontal scaling is the use of multiple systems performing the same function in parallel. This is appropriate for applications which have limitations in a single kernel environment.
Single threaded applications typically scale horizontally whilst multi-threaded applications generally scale vertically, although they can also benefit from horizontal scaling.

The use of horizontal scaling requires a mechanism to distribute the load across the multiple systems. A key benefit of load distribution is that “N+1” high availability becomes straightforward to implement (“N+1” is a strategy for high availability; N is the number of servers required to support anticipated workload; N+1 servers are provisioned to provide a resilient solution that copes with a single server failure without degrading performance).

High availability of services is achieved by examining each component in turn and either
employing load distribution techniques across N+1 systems or utilising Sun Cluster to monitor availability and invoke automatic failover.

The architecture is designed to accommodate the addition of future services by being modular so that complex interactions and dependencies between components are minimised.

Applying the Architecture

The Access Zone is responsible for controlling access to the eUniversities network. It consists of firewalls for security and Sun ONE services for authentication and authorisation. Users (both Students and eLearning Content Providers) access via JANET or the Internet, by connecting through a firewall layer to the Sun ONE Gateway Servers. The firewall servers are responsible for providing edge security by packet filtering.

The Gateway Servers provide basic authentication services (via the Identity Servers) to incoming users sessions and then route requests through to the Portal Servers. ThePortal Servers display the eUniversities common look

and feel, allows user personalisation and manage requests to the services provided by eUniversities.

Users can only gain access to eUniversities via the gateways which, once the users have been authenticated, route all requests through to the Portal Servers. Enrolled student data is held in a distributed LDAP directory that contains specific user profile information that is used for authentication. The Portal Servers and Identity Servers use the LDAP Server to determine the format and presentation of the web pages depending on the user information stored in the profile directory. Through this mechanism, different organisations or groups can be presented with tailored pages.

The LDAP service is not discussed here.

Server platforms are installed as follows for the Access Zone:

- Multiple (N+1) firewalls
- Multiple (N+1) Sun ONE Gateway Servers (Portal Remote Access Pack)
- Multiple (N+1) Sun ONE Portal Servers

[In practice, for reasons of economy the Portal and Gateway are cohosted on two single servers]

The resilience and scalability strategy for each type of server is as follows:

For the Firewall Server, the load balancing (network) switch detects the failure and routes requests through to the surviving firewalls. The scaling is horizontal.

For the Gateway Server the load balancing (network) switch detects the failure and routes requests through to the surviving gateways. The scaling is horizontal.

The Portal Servers are currently cohosted with the Gateway Servers but the Architecture envisages that these can be hosted on servers of their own in which case the load balancing (network) switch detects the failure and routes requests through to the surviving Portals. Scaling is therefore horizontal.

The conclusion to this discussion is that it was always intended that we add servers and scale horizontally either with multiple Portal/Gateways or multiple Portals and multiple Gateways. The latter approach is more resilient.

Load Generation using Loadrunner

Mercury Interactive LoadRunner is a suite of tools for measuring and analyzing the behavior and performance of enterprise applications. We use it to exercise our benchmarking infrastructure by emulating a population of users ("Virtual Users" or "Vusers"). It employs performance monitors to identify and isolate problems. Through this process, it enables us to obtain an accurate picture of end-to-end system performance and identify and eliminate performance bottlenecks through a process of iterating through modification and test.

LoadRunner creates Vusers to put an application through the rigors of real-life user loads. We can stress an application across all architectural tiers applying consistent, measurable and repeatable loads, then use the data to identify scalability issues. It measures the end-user response times of transactions, uncovering end-to-end performance problems. It employs monitors that capture and display performance data from every component. Its' Analysis module then automatically correlates this performance data with the end-user response times, so the we can determine the impact and source of bottlenecks. The module also automatically generates standard and custom management reports, examples of which are included in this document.

The LoadRunner Scenarios Used

Readers interested in the scenarios that we used and the code for them are referred to Appendix A.

Variables under Test

The chapters that follow this one focus on individual areas of investigation. However in summary we have concentrated our testing on;

- Splitting the Portal from the Gateway.
- Adding Gateways to the Portal.
- Using Solaris alternate libthread.
- Upgrading to Solaris 9.
- Switching off Secure Socket Layer (SSL).
- Observing Java Virtual Machine Garbage Collection (GC).
- Tuning operating system parameters.

These were chosen as the most likely areas for improvement. Rather than run every combination which would have yielded over 50 tests, we "pruned the tree" of possibilities as soon as it became apparent that there were no appreciable gains to be made; examples of this approach were GC and OS tuning which were observed to be non-issues from the outset of their tests

and multiple gateway configurations which yielded interesting and encouraging results throughout.

Assumptions & Constraints

Production Platform & Benchmarking Platform Differences.

The benchmarking environment differs from Production in several ways.

- We model “one half” of the production platform. In the UK eUniversity environment there is a redundant path comprising dual Portal/Gateways which are load-balanced.
- Our equipment (440 MHz US-II/US-IIi CPUs) pre-dates that in production (550 MHz US-IIi CPUs).

We ignore these factors for the purpose of the test and re-address them in the Conclusion.

Response Times & SLA

The criterion for performance is set out in the Non-Functional Requirements. The maximum page-load response time will be 3 seconds under controlled circumstances [Non Functional Requirements (SPS16980), section 3.2.2.1].

Scalability criteria are derived in the High Level Infrastructure Architecture (SPS17900) section 10.3.

	Year 1	Year 2	Year 3
Concurrent Users (ave)	1,200	3,078	6,665
Concurrent Users (peak)	3,600	9,233	19,995

Real Users and Virtual Users

In most benchmarks there is a 1:1 relationship between the real end-user of the system and the Vuser used to model their access to the system. One of the key attributes that the vuser must emulate is the amount of time spent between interactions with the system; i.e clicking a mouse button. This is known as the “think time”. Typical values for most business applications are in the range of 5 to 15 seconds. This variable is key because the smaller the think time, the harder the application has to work.

This is problematic in the case of eUniversity whose site delivers learning content. The access pattern for this type of content is completely different to any commercial web-application. The real user will spend minutes or hours in one screen with no interaction with the system. However the tests we run must complete in bounded time if reasonable progress is to be made and sufficient load must be applied to the platform to stress it. Therefore we have resorted to using think time in the range of 1 to 40 seconds.

The result of this compromise is that we have informally agreed that a reasonable conversion factor between virtual users and real users is a factor of 5.

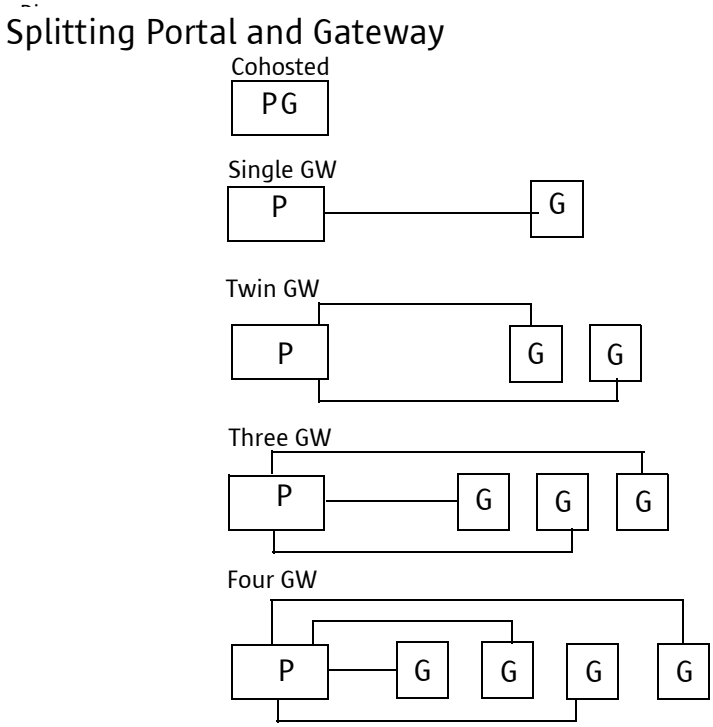
Splitting Portal and Gateway

Introduction

The next stage of our testing involved investigating the horizontal scalability of the Portal/Gateway functions within the system. Concerns had previously been raised about the possibility of Portal/Gateway being a bottleneck. The goals of the tests detailed in this chapter were to investigate Portal/Gateway limitations and to measure the benefit of splitting the services between hosts and scaling the number of Gateway machines.

The initial configuration consisted of Portal and Gateway coexisting on a single host (*cohosted*). The test plan was first to split the services on to physically separate hosts and then scale the number of Gateway machines upwards to two, three and finally four Gateway hosts, as shown in diagram 1.

Diagram 1:



The machine specifications were identical across all tests. The specification for the initial Portal host was:

- Netra t1405
- 1 x 440MHz UltraSPARC II
- 4 Gb RAM

The specification for the Gateway machines:

- Netra T1
- 1 x 440MHz UltraSPARC IIi
- 1 Gb RAM

Across all tests in this section the operating system used was Solaris 8. The LoadRunner tests logged 5 users in every minute through the Portal until the system resource usage hit and remained above 95%; the measure of success being the number of running Vusers by this point.

Portals typically make extensive use of cached static data. Dynamic data can not be cached within the Portal. For this reason tests were run against both the dynamic content of the application and static HTML content delivered from the Application Server. Using static content also enabled us to ensure that any bottlenecks occurred were in the Portal and not the Application Server.

Reports and Data

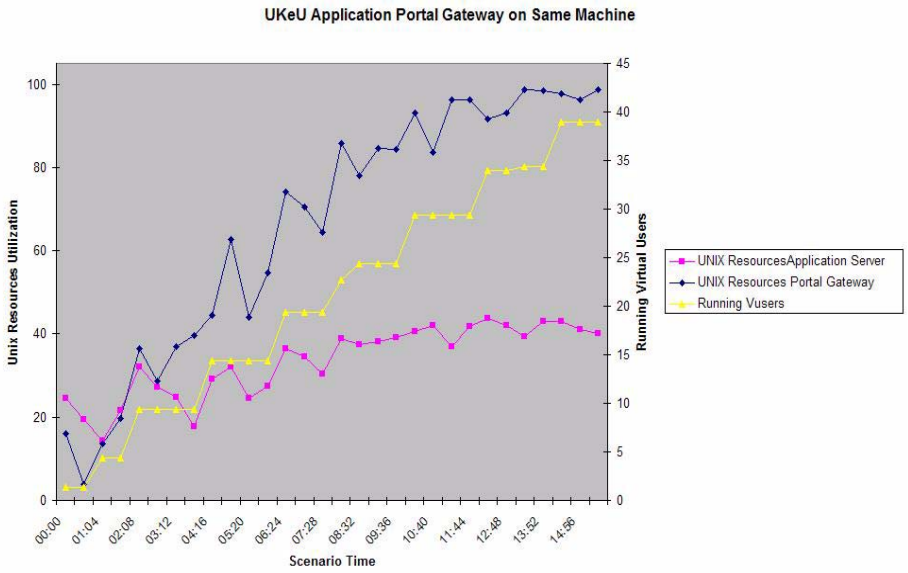
The results for each of the ten tests presented in this section are averaged from 3 separate runs to ensure we generated valid data.

Dynamic Content

Cohosted Portal and Gateway

The results detailed here show the outcome of testing the dynamic application content from the Application Server through a cohosted Portal/Gateway machine. The maximum number of supported Vusers was 35, see Figure 1.

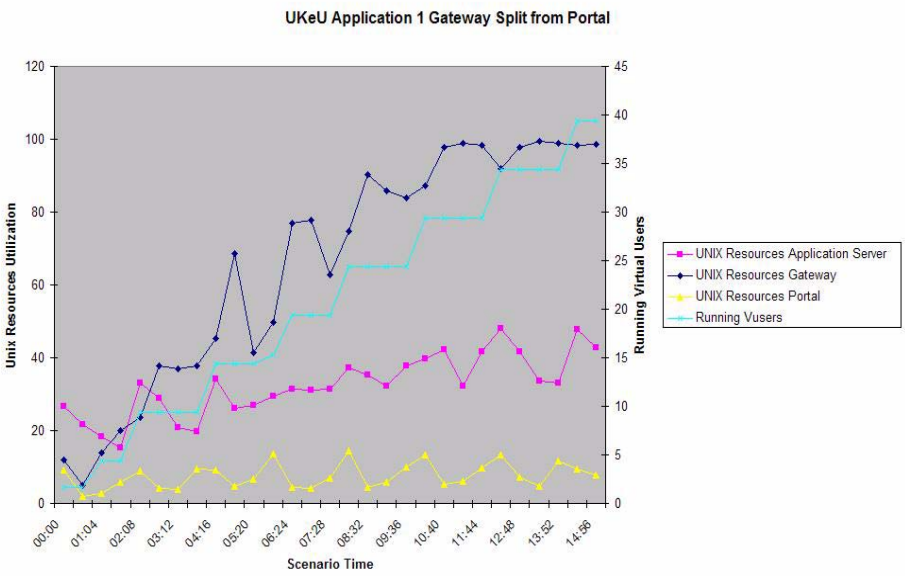
Figure 1:



Portal and Single Separate Gateway

Next we separated the Portal and Gateway functions out onto two machines (see Figure 2). This yielded a Vuser limit of 30 Vusers before the resource limits were reached on the Gateway. The drop in supported load can be explained by the overhead of introducing a network link (physically separate hosts as opposed to cohosted) between the Portal and Gateway service. The number of Vusers continued to rise to 39 but at the expense of response time.

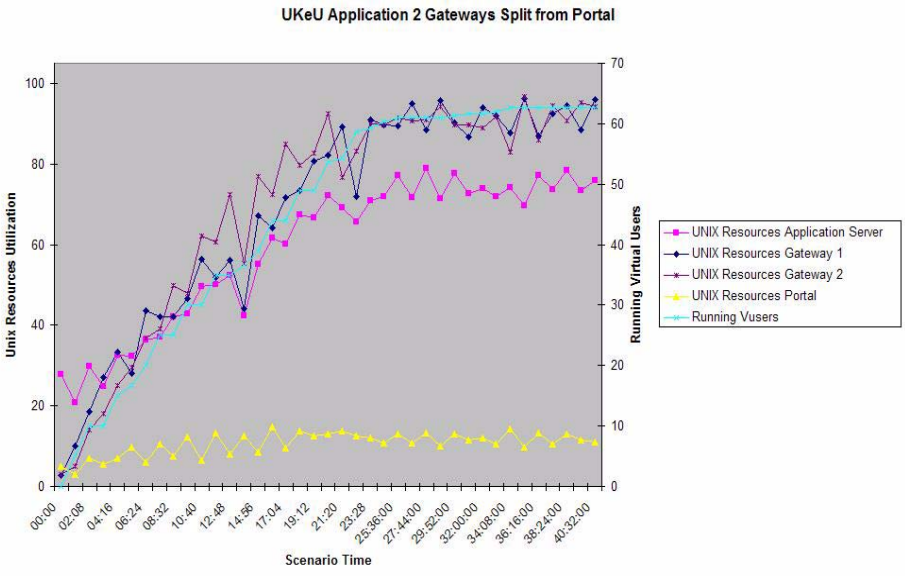
Figure 2:



Portal and Two Separate Gateways.

When we added a second Gateway, the number of Vusers increased from 30 to 62. See Figure 3.

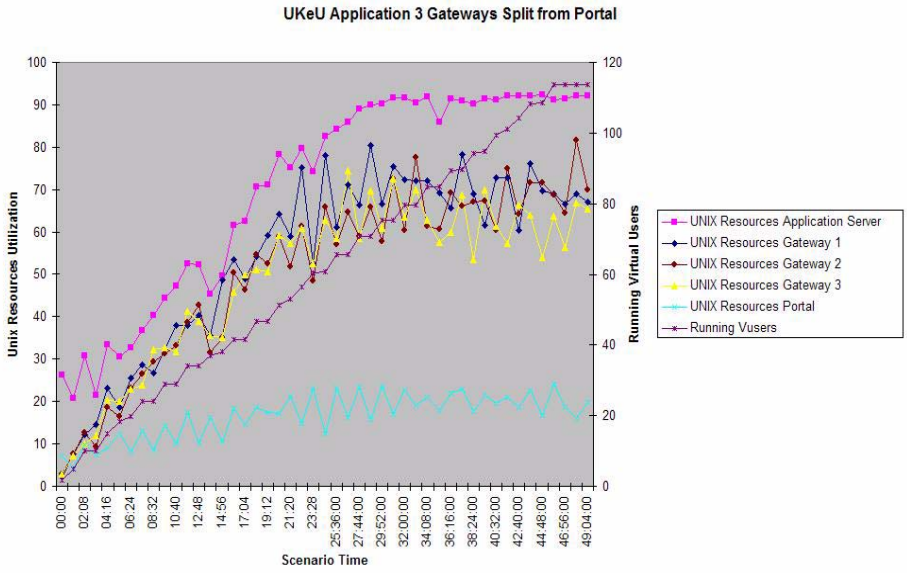
Figure 3:



Portal and Three Separate Gateways.

Next we increased the number of Gateways to three. This time the Vuser total rose to 70, from 62. At 70 Vusers the application server reached peak utilisation.
We were actually able to add more users up to 113 at the expense of response time. None of the Gateway machines exceeded 80% utilisation. Figure 4 shows the data.

Figure 4:



Portal and Four Separate Gateways

Due to that fact that we had identified the Application as a bottleneck with 3 Gateway machines it was not worthwhile running dynamic content tests against 4 Gateways.

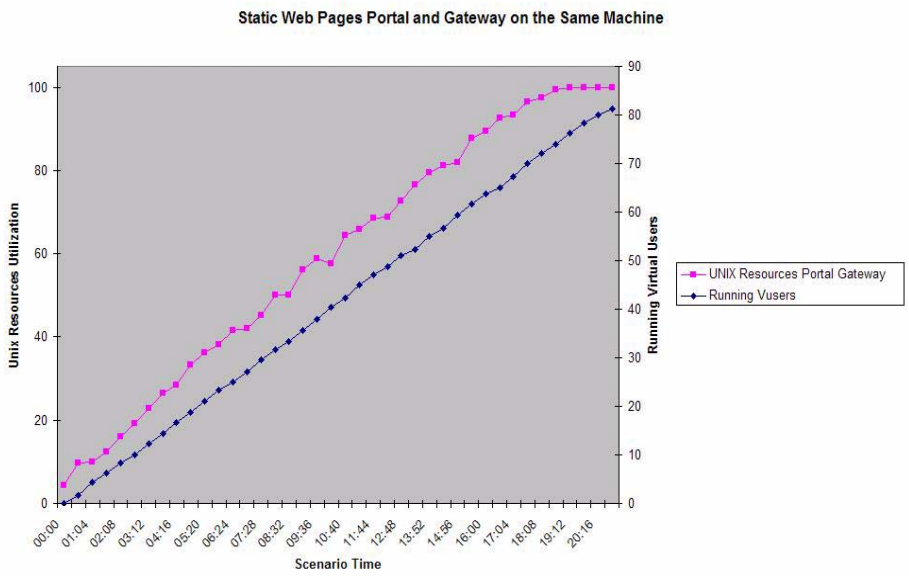
Static Content

Cohosted Portal and Gateway

We tested the system with Static content in order to further stress the Portal/Gateway system without bottlenecking at the application server. On a cohosted Portal/Gateway we achieved 86 Vusers before reaching resource limits.

Figure 5 shows the results for a single Portal machine and cohosted Gateway.

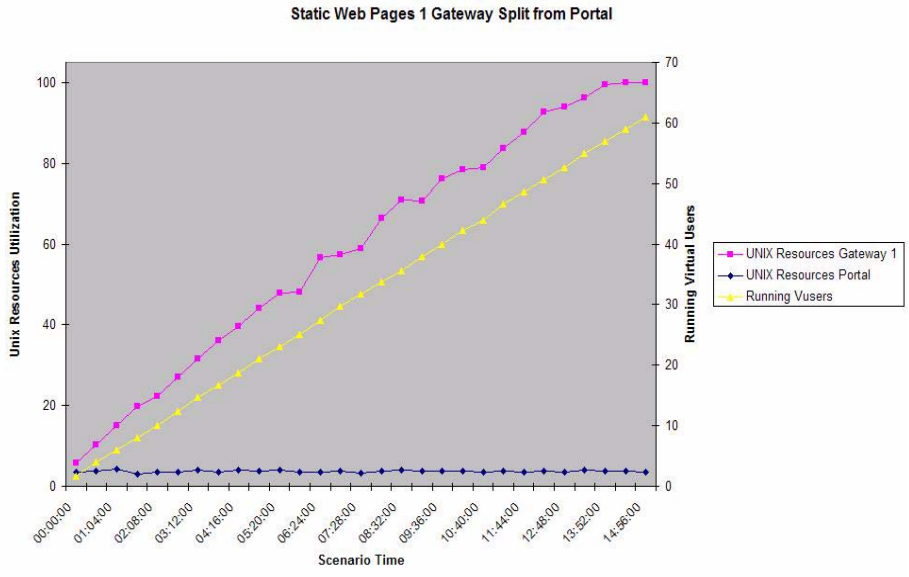
Figure 5:



Portal and Single Separate Gateway

Test results from this scenario showed that we were able to support 61 Vusers on a single Gateway machine before the resource limits were reached. See Figure 6 below for the results.

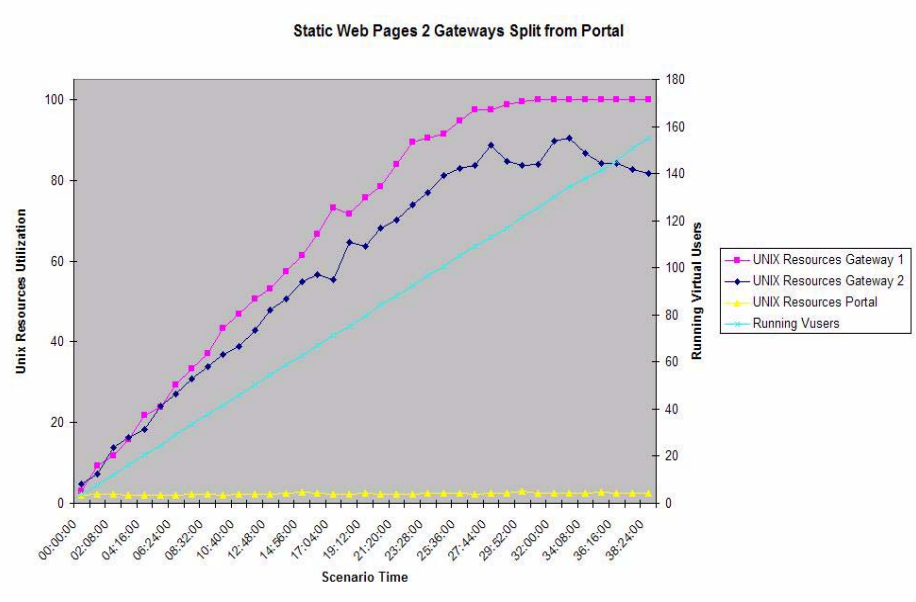
Figure 6:



Portal and Two Separate Gateways.

The results from the test using two separate gateways showed that we were able to support 157, an increase in supported Vusers of 96. See Figure 7 below for the data.

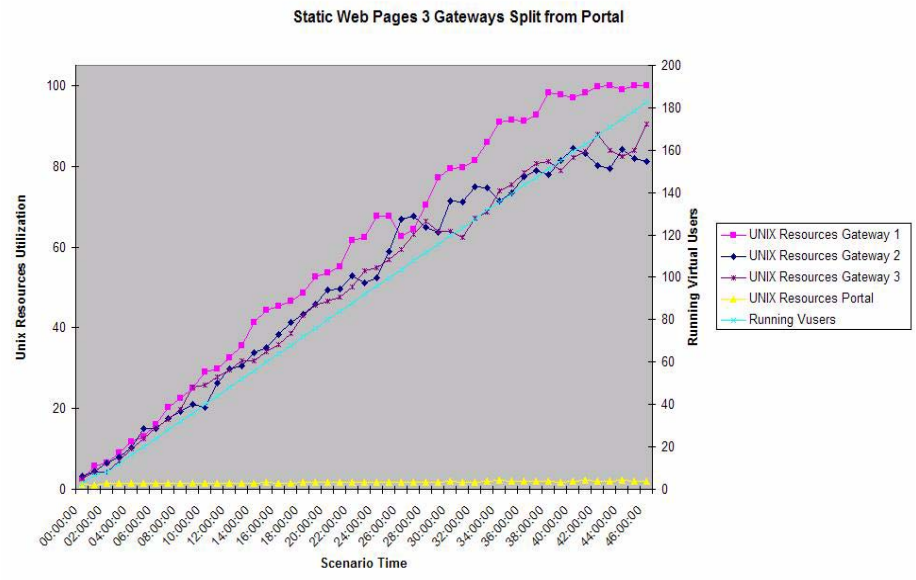
Figure 7:



Portal and Three Separate Gateways.

The maximum number of supported Vusers for three separate Gateways was 182, an increase of 24 over two Gateway machines. See Figure 8 below for the data.

Figure 8:



Portal and Four Separate Gateways.

This test supported ~300 running Vusers, but is not really a valid measure because we were limited by network bandwidth. Consequently these users would have been experiencing slow response times long before the resources of the Gateway hosts were a limitation. We were running on 100M/bit network giving a theoretical maximum of 11.92MBytes a second. In reality though the maximum is closer to 8 MBytes a second. See Figure 9 below for the data. This finding correlates with those of the Portal product team.

Figure 9:

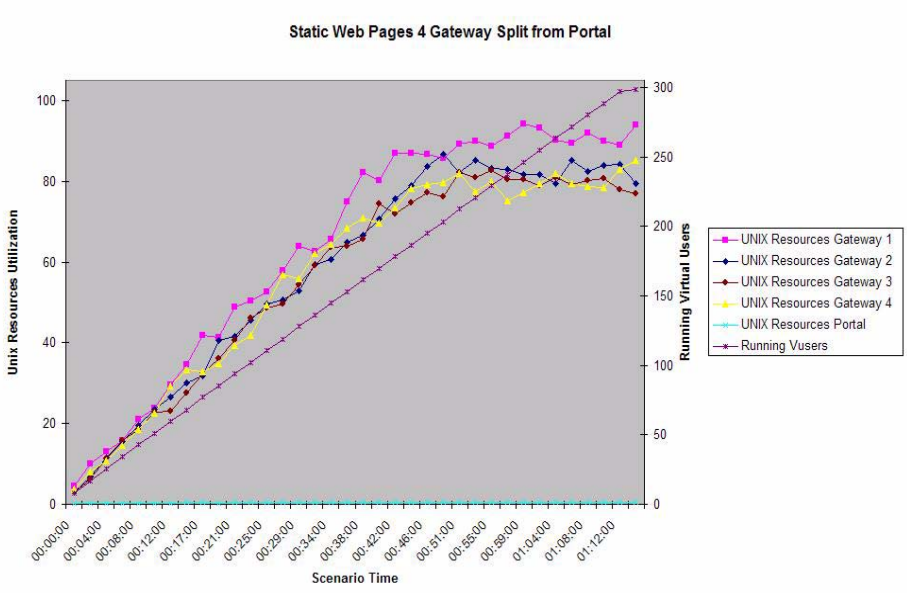
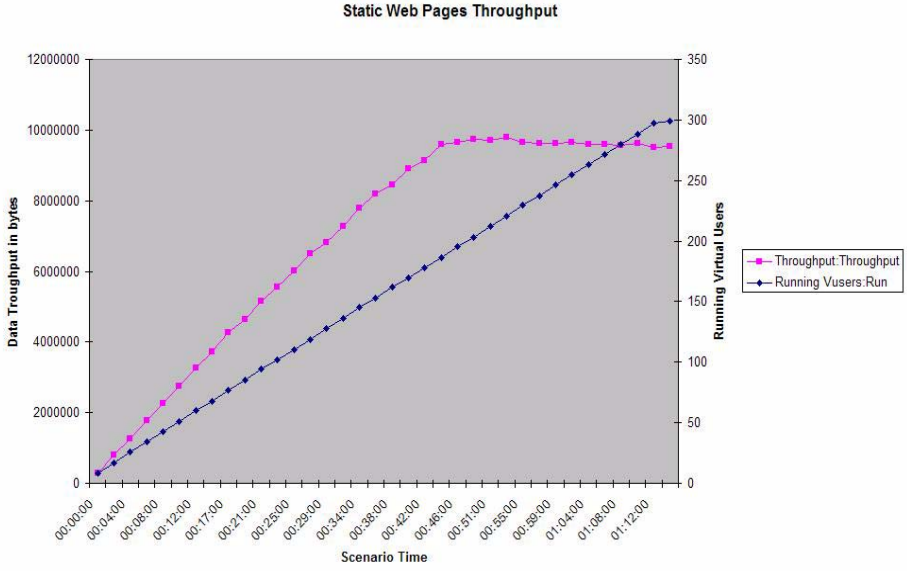


Figure 10 below shows the throughput tail off against the number of running Vusers.

Figure 10:



Interpretations and Summary

Figure 10 shows the throughput elbows at 8.7MB/s with a Vuser population of 178.

For static content the maximum supported load increases almost linearly with the addition of Gateway hosts after an initial drop seen when moving from a cohosted system to separate single Portal and Gateway hosts. At 4 separate Gateway hosts the limiting factor is the underlying 100Mbit network. The final numbers for supported users are:

Number of Gateways	Supported Vusers
Cohosted	86
1	61
2	157
3	182
4	178 (network limitations)

For dynamic content a similar pattern emerged:

Number of Gateways	Supported Vusers
Cohosted	35
1	30
2	62
3	70 (application limitations)
4	Not run.

Our static tests used a large web page of 600kb. Some brief research has shown that the maximum learning object size is not going to exceed 150kb (see Assumptions). In order to get an indication of how our data might relate to the production system we have to make allowance for the difference in data size. To achieve this we have applied a scaling factor of 4. The scaling factor is discussed in Conclusions and Further Work.

The Vuser to real user ratio is 5 to 1; see Assumptions. Lets take an example:

3 Gateway's with 50 Vusers scaled by 4 gives 600 Vusers. Applying the Vuser ratio gives an estimated 3000 supportable concurrent users with 3 Gateway machines per Portal.
Applying this method to our static data gives:

Gateways	Real Users

cohosted	1720
1	1220
2	3140
3	3640

Conclusions

Testing with dynamic content from the application prevented us from stretching the Portal/Gateway system(s) to their maximum limits. The Application Server became the primary restraint on increases in throughput and load in this case. For this reason the static content proved to be a useful measure of the true limitations of Portal/Gateway.

The data shows that the Portal server is under utilised and has significant headroom for growth. This underutilisation could only be observed once the Gateway functionality was removed from the Portal system to a separate host and the Portal resources could be properly monitored.

The data in figure 3 demonstrates the requirement for a one to many relationship between a Portal and Gateways; SSL encryption is far more CPU intensive than HTML rendering. The ability of the system to continue scaling as we added Gateways whilst still only having a single Portal host reinforced this conclusion.

Portal/Gateway is not a bottleneck, it is scalable within the current system up to the limits of the underlying network infrastructure. Further Portal/Gateway capacity investigations with dynamic application content cannot be carried out due to bottlenecking on our current Application Server.

Removing the Gateway from the Portal to a configuration with a single, separate Gateway produced a drop in performance, this can be explained by the increased latency introduced between the two services by the addition of a network link (see Diagram 1).

Libthread, Solaris 9 & GC Tuning

Introduction

Solaris 9 introduced many performance enhancements over previous versions of Solaris, the most notable of which was to make what had previously been the “alternate” libthread implementation in Solaris 8 the default and to further enhance it.

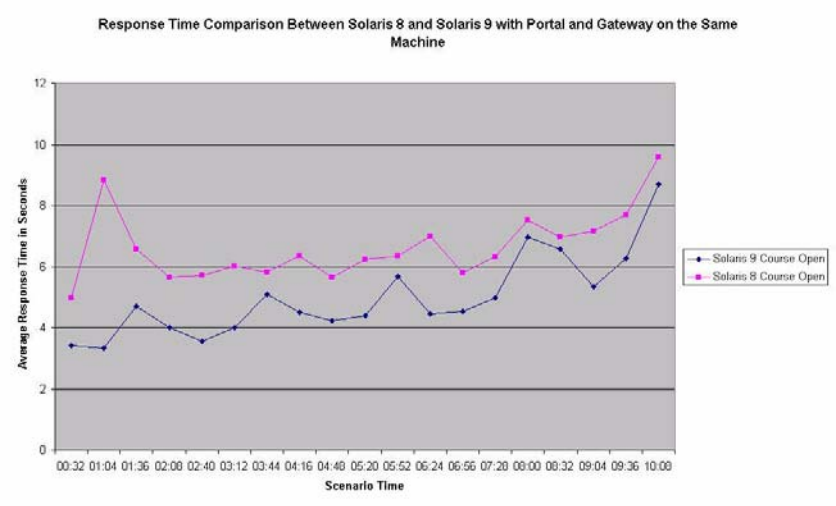
This library was first introduced as an alternative in Solaris 8 02/02 and gave a significant improvement on multithreaded applications over previous implementations of libthread. The performance impact on the majority of applications has been dramatic. As part of our investigations we compared the performance of the Portal/Gateway running with Solaris 8 alternate libthread library to Solaris 9. The more detailed technical description of libthread can be found in eUniversities OS & JVM Tuning SPS31808 1 Sept. 03.

We repeated a subset of the tests run previously in investigating horizontal scaleability. We compared both a cohosted Solaris 8 Portal/Gateway with a cohosted Solaris 9 Portal/Gateway and a separated single Solaris 8 Gateway with a separated single Solaris 9 Gateway.

Reports and Data

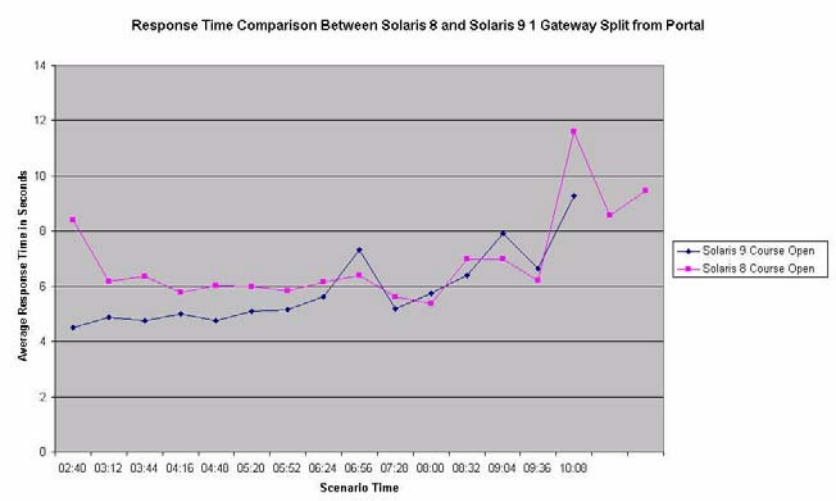
Cohosted Portal and Gateway

Figure 11:



Portal and Single Separate Gateway

Figure 12:



Interpretations and Summary

The comparison between Figure 11 and Figure 12 shows the negative effect of network latency as seen in Splitting Portal & Gateway.

Figure 11 shows a decrease in Vuser response time in the region of 1 second due to the upgrade to Solaris 9. This represents a 33% improvement against the Non-Functional Requirement of a 3 second response time outlined in Assumptions & Constraints. The improvements in response time can also be observed in Figure 12 but are less pronounced due to the effect of network latency.

JVM Tuning

We monitored JVM Garbage Collection during various test runs. No significant GC overhead was observed. "Significant" is generally defined as greater that 5% of application time. Here we have 1.5% (0.6 second) GC stall every 40 seconds.

Perftune

There is a tuning script supplied with the Portal product that automates the tuning of a number of Solaris kernel variables in accordance with “best practice” as defined by the product group. We performed a test to see if an installation modified using this script ran any better. There was no effect. It is likely that the tuning in this script is relevant to much larger (4-8 CPU and >4GB) Portals.

Secure Socket Layer (SSL)

Introduction

The eUniversity application makes heavy use of secure transmissions using the TCP/IP secure sockets layer (SSL). SSL encrypts all traffic between the client and server, causing considerable CPU overhead. It is vital that load balancers use affinity routing in a distributed SSL environmen. Affinity routing ensures that after an initial connection is established between client and server all subsequent requests are routed to the same server, preventing the need to repeat the costly SSL handshake sequence. It is possible to off load the overhead of SSL processing to dedicated hardware known as SSL accelerator cards.

With SSL having such a CPU utilisation cost, it was a candidate for investigation. In the absence of real SSL hardware accelerator cards we performed tests that would compare the systems performance with and without SSL. The results here can only accurately show the benefit of turning SSL off completely; not the benefit of using dedicated hardware. Whilst these results may be indicative of the outcome of testing using SSL accelerator hardware, it cannot provide a firm metric.

These results may be seen as an indicator of the value of a full SSL hardware investigation.

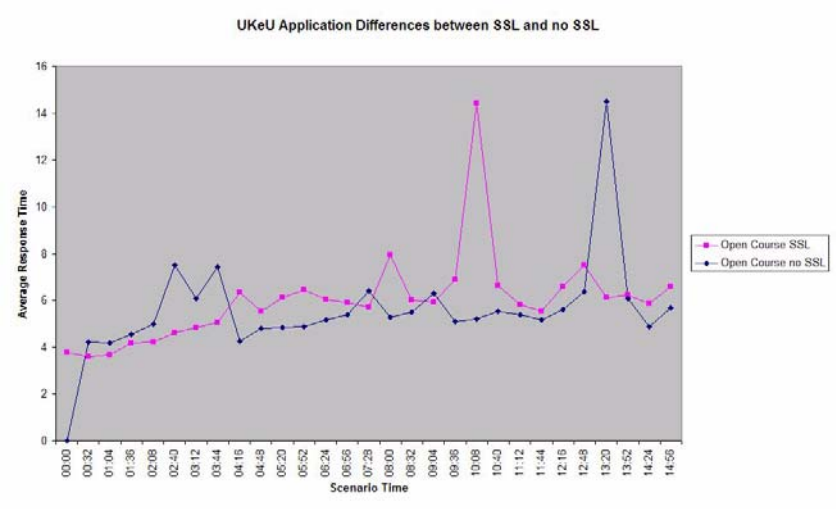
Reports and Data

SSL vs. Non-SSL

Portal and Single Separate Gateway

Figure 13 below shows the difference in response time with and without SSL.

Figure 13:



The measurement unit we used to ascertain the overhead of SSL was response time of opening a course while the system was loaded with 20 concurrent Vusers. Response times varied from 4 seconds to 8 seconds.

On average, response times for this test showed that Non-SSL requests took 0.32 seconds less than SSL requests. The average response time for SSL requests was 6.08secs and 5.76secs for non-SSL.

Conclusion

Our brief SSL investigations have proven what we expected, that without SSL processing taking place between the client and server we observe an improvement in response times. This will not have an impact on the maximum supported load, but instead will result in faster response times and improved user experience. There may be some value in investigating the effect of SSL accelerator hardware on system performance in the future.

Conclusions and Further Work

Conclusions

In this section we revisit the figures from the Non-Functional Requirements and our results for splitting Portal/Gateway. Then the conclusions outlined in the Summary on page 1 are reiterated and justified. Then we list further investigations that might be worthwhile, subject to funding.

Non-Functional Requirement for user load:

	Year 1	Year 2	Year 3
Concurrent Users (ave)	1,200	3,078	6,665
Concurrent Users (peak)	3,600	9,233	19,995

Result data for additional Gateway hardware:

Gateways	Real Users (test)	Real Users (eUniversity)
cohosted	1720	3440
1	1220	2440
2	3140	6280
3	3640	7280

The eUniversity production network consists of double the test network hardware, i.e. where we have a single Portal/Gateway the production network is load balanced across two. The second column above shows the number of real users achievable utilising the test system. The third column shows the number of real users achievable with eUniversities production network.

We continue to discount the increase in performance in the eUniversity environment due to the differences in hardware between that environement and the test environment (See Assumptions & Constraints).

- The current Portal/Gateway configuration is not an inhibitor to current system performance and there is headroom within the current configuration. The original technical design is validated. As far as Portal/Gateway is concerned, there is a clearly definable hardware upgrade path to support the average concurrent user load defined by the Non-Functional Requirement.

- We are confident that Portal/Gateway will scale to meet the year 3 Non-Functional Requirement for peak load if steps are taken to improve the network infrastructure and given a more performant application.

- Our tests bear out our expectations of the platform and also broadly correlate to benchmarks carried out by internal Sun product groups. We know from other tests we have performed in parallel to those described above, that the current performance bottleneck is in the application code - principally the User Management Module (UMM). Our focus moving forward will be to systematically test the ELP1h code base, commencing with the rewritten version of UMM to establish where future bottlenecks may lie.

- There is a significant performance benefit in upgrading the current version of the operating system.

- There is a linear relationship between the amount of Gateway servers that can be added to the system and the number of users that can be sustained giving a clear upgrade path for the future. A bottleneck will then appear in the networking infrastructure. The appearance of this bottleneck in our testing bears out the findings of the internal Sun product teams who have had difficulty in stress testing Portal/Gateway 6.0 without resorting to gigabit networking hardware. This is particularly the case in the context of multi-cpu servers.

- The use of dedicated Secure Socket Layer (SSL) processing hardware is likely to provide additional headroom in the future. All of the site's traffic is encrypted. Although we haven't tested this, eUniversity may take the view that it is more cost-efficient to use this route to extend the capability of their servers rather than simply adding them.

- Another possibility we have only touched on is having less than 100% of the site content transmitted to users in encrypted form. We have not pursued this because it would involve revisiting fundamental design decisions.

- The Portal/Gateway configuration may warrant further investigation in the context of a more performant version of the application it serves. As we have indicated previously, the tests we can presently carry out are somewhat constrained by the amount of throughput we can coax out of the current application. The performance of the application will improve with future releases and it is then that the characteristics of Portal/Gateway may change.

Recommendations

- Upgrade Portal/Gateway to currently shipping version of Solaris.
- Consider the purchase of SSL Accelerator hardware.
- Audit Production network to ensure affinity routing is used.

Further Work

Further validation of the scaling factor used to compensate for difference between page size used in test and production.

The possibilities for further work that might be undertaken, subject to funding are:

- Validating the tests we have used in the context of a future application release.
- Providing metrics on performance improvements due to SSL cards to assist in any purchasing decision.
- GC Tuning if appropriate. This is especially relevant if a move is made to multi-CPU hardware

References

Manuals

These are available at http://docs.sun.com
Deployment Guide Sun ONE Portal Server Release 6.0 816-6363-10 March 2003

Installation Guide for UNIX Sun ONE Portal Server 6.0 816-6358-10 August 2002

Administrator's Guide Sun ONE Portal Server 6.0 816-6359-10 March 2003
Desktop Customization Guide Sun™ ONE Portal Server Release 6.0 816-6361-10 Aug 2002

Developer's Guide Sun ONE Portal Server Release 6.0 816-6362-10 February 2003

Installation Guide Sun ONE Portal Server Communication Channels Version 6.0 816-6720-10 June 2003

Installation Guide, Sun ONE Portal Server, Secure Remote Access Release 6.0 816-6455-10 September 2002

Administrator's Guide Sun ONE Portal Server, Secure Remote Access 6.0 816-6421-10

eUniversities Project Documents

eUniversities High Level Infrastructure Architecture SPS17900 13th September 2002
eUniversities Portal Architecture Document SPS22836 18th September 2002
eUniversities OS & JVM Tuning Document SPS31808 1st September 2003

JVM Tuning

Java Performance Tuning 2nd Edition Jack Shirazi (Oreilly)
This textbook covers JVM tuning generally, see especially Chapters 2-3.
Tuning Garbage Collection with the 1.3.1 Java Virtual Machine
http://java.sun.com/docs/hotspot/gc/index.html
This is the reference for the specific version of JVM used.

Appendix A - LoadRunner Scripts

Introduction

This appendix contains a selection of the LoadRunner scripts used during testing. A complete LoadRunner program should consist of an init file, an action file and an end file. The init code will start and initialise the Vuser and perform one time tasks such as logging in. The action code should be used to do work related to the functionality to be tested. In our case this may have been choosing courses, selecting learning objects, reading mail or other user functions. The end code should clean up and exit the Vuser. Logging out is done here.

Appserver Tests

This Script logs a users into the eUniversity application and selects the course that our test users are enrolled on. Once in this course it selects the first module. The script will then randomly select a sub module, and then a learning object from that sub module. Once the learning object has been loaded the script will then return to the main portal home page for that user and then execute the loop again. Between each step there is a random think time which is designed in an attempt to stop all of the users getting into a lock step. Each step also has a transaction response time associated with it that is saved and has produced some of the results seen throughout this document.

vuser_init.c

```
#include "web_api.h"
#include "lrw_custom_body.h"
#include "../common/functions.c"

vuser_init()
{
                    int rc;

          /*Login to the UKeU Application*/
          if ( ( rc = ukeulogin("{username}") ) != 0 )
          {
                    lr_error_message("Error in login return
code %d", rc);
          lr_abort();
```

```
     }
     return 0;
}
```

action.c

```
#include "as_web.h"


Action()

{

     int i=0,j=0,rc;

     char *linksel;

     char *position_of_search;


     /*Initial think time after login*/
     if ( ( rc = think ( 20, 1) ) != 0 )

     {

          lr_error_message("Error in think time");

     }


     /*Registers a find in the next web page for the text
Your course*/
     if ( ( rc = web_reg_find("Text=Your course", "Save-
Count=coursecount",

          LAST) ) != 0 )

     {

          lr_error_message("Web Reg Find failed");

          return 2;

     }


     /*Registers a save for all the available modules in
the course to an array*/
     if ( (rc = web_reg_save_param("savedparameter",
"Lb=class=\"paraTextSize\">", "RB=</a>", "ORD=ALL", LAST)
) != 0 )

     {

                    lr_error_message("Web Reg Save Param
failed");

                    return 8;

     }
```

```
    lr_start_transaction("course_open");


    /*Opens the main course page and checks the return
codes to ensure the function completed correctly*/
    if ( ( rc = web_url("iassso",

         "URL=https://portal.portal.private/http://por-
tal.portal.private/sso/iassso?tar-
get=appserver&request=http%3A%2F%2Fas1.portal.private%2Fe
learn-
ing%2Flearn%2Flearn%2FgetProgrammeOffering.do%3FnamePath%
3DUKeU%2FPERFORMANCE+TEST+1%2FEnrolled",

         "TargetFrame=",

         "Resource=0",

         "RecContentType=text/html",

         "Referer=https://portal.portal.private/http://
portal.portal.private/portal/dt",

         "Snapshot=t21.inf",

         "Mode=HTML",

         EXTRARES,

         "Url=/http://as1.portal.private/elearning/
learn/images/backgrounds/topnav_back2.gif", "Ref-
erer=https://portal.portal.private/http://as1.portal.pri-
vate/elearning/learn/learn/
getProgrammeOffering.do?namePath=UKeU/PERFORM-
ANCE%20TEST%201/Enrolled", ENDITEM,

         "Url=/http://as1.portal.private/elearning/
learn/images/backgrounds/mus_spine_mod1.gif", "Ref-
erer=https://portal.portal.private/http://as1.portal.pri-
vate/elearning/learn/learn/
getProgrammeOffering.do?namePath=UKeU/PERFORM-
ANCE%20TEST%201/Enrolled", ENDITEM,

         LAST) ) != 0 )

    {

         lr_end_transaction("course_open", LR_AUTO);

         lr_error_message("Error in opening page");

         return 3;

    }

    lr_end_transaction("course_open", LR_AUTO);


    /*Think time between 1 and 20 seconds*/
    if ( ( rc = think ( 20, 1) ) != 0 )

    {

         lr_error_message("Error in think time");

    }
```

```
     /*Randomly selects a module to open*/
     if ( (linksel = random_select()) == "6" )

     {

          lr_error_message("saved parameter failure");

          return 6;

     }



/*   if ( ( rc = lr_save_string(lr_eval_string(link-
sel),"moduleselect") ) != 0 )

     {

          lr_error_message("Error in saving a string");

          return 1;

     }
*/

     /*Due to the course that is currently in use only
haveing module 1 populated
       the string that is returned in ignored and module 1
is entered into the
       parameter.  The code is left in to allow changes to
be made once more
        course data is available*/
     lr_save_string("1 Induction", "moduleselect");

     lr_output_message("x= %s", lr_eval_string("{mod-
uleselect}"));


     /*Registers a find for the next page*/
     if ( ( rc = web_reg_find("Text=Your course",

          LAST) ) != 0)

     {

          lr_error_message("Error in Web Reg find");

          return 2;

     }


     /*Registers a save for all the sub modules that are
available in the unit*/
     if ( ( rc = web_reg_save_param("savedparameter",
"LB=Unit\" class=\"paraTextSize\">", "RB=</a>",
"ORD=ALL", LAST) ) != 0 )

     {

          lr_error_message("Error in Web Reg Save Param");

          return 8;

     }
```

```
     lr_start_transaction("moduleselect");


     /*Opens the module that was selected earlier*/
     if ( ( rc = web_link("moduleselect", "Text={mod-
uleselect}", "Snapshot=t3.inf", LAST) ) != 0 )

     {

          lr_end_transaction("moduleselect", LR_AUTO);

          lr_error_message("Error opening web link");

          return 9;

     }


     lr_end_transaction("moduleselect", LR_AUTO);


     /*Randomly selects a link from the sub modules that
are available*/
     if ( (linksel = random_select()) == "6" )

     {

          lr_error_message("saved parameter failure");

          return 6;

     }


     /*Saves the string so that it is available for use in
the LoadRunner framework*/
     if ( ( rc = lr_save_string(lr_eval_string(link-
sel),"submoduleselect") ) != 0 )

     {

          lr_error_message("Error in saving a string");

          return 1;

     }



     /*Registers a find for the next page*/
     if ( ( rc = web_reg_find("Text=Your course",

          LAST) ) != 0 )

     {

          lr_error_message("Error in Web Reg Find");

          return 2;

     }
```

```
     /*Registers a save for all of the units that are
available in the submodule*/
     if ( ( rc = web_reg_save_param("savedparameter",
"LB=onkeypress=\"if (event.keyCode == 13) { reposition-
Win();newWin=window.open('','learningObject", "RB=</a>",
"ORD=ALL", LAST) ) != 0 )

     {

          lr_error_message("Error in Web reg save param");

          return 8;

     }


     /*Random think time of between 1 and 20 seconds*/
     if ( ( rc = think ( 20, 1 ) ) != 0 )

     {

          lr_error_message("Error in think time");

     }




     lr_start_transaction("unitselect");


     /*Opens the sub module*/
     if ( ( rc = web_link("unitselection", "Text={submod-
uleselect}", LAST) ) != 0 )

     {

          lr_end_transaction("unitselect", LR_AUTO);

          lr_error_message("Error in web link");

          return 9;

     }



     lr_end_transaction("unitselect", LR_AUTO);

     /*Random think time between 1 and 20 seconds*/
     if ( ( rc = think ( 20, 1 ) ) != 0 )

     {

          lr_error_message("Error in think time");

     }


     /*Randomly select a learning object*/
     if ( (linksel = random_select()) == "6")

     {

          lr_error_message("saved parameter failure");
```

```
          return 6;

     }


     /*Saves the string into a varible that LoadRunner can
use*/
     if ( ( rc = lr_save_string(lr_eval_string(link-
sel),"learnobjectselect") ) != 0)

     {

          lr_error_message("Error in saving string");

          return 1;

     }


     /*Search the string to find the text for the link to
click on. This is due to the amount of data
       that needs to be searched in order to only save the
strings of the learningobjects not the modules and
        sub modules.*/
     if ( ( position_of_search = (char
*)search_for_token( lr_eval_string("{learnobjectse-
lect}"), ">", 2 ) ) == "NULL" )

     {

          lr_error_message("Error finding the correct
link");

          return 7;

     }


     /*Saves the string into a varible that LoadRunner can
use*/
     if ( ( rc = lr_save_string(position_of_search,
"searchlearningobject") ) != 0)

     {

          lr_error_message("Error in saving string");

          return 1;

     }



     lr_start_transaction("learningobjectselect");


     /*Opens the learning object*/
     if ( ( rc = web_link("learningobject",
"Text={searchlearningobject}", LAST) ) != 0)

     {

          lr_end_transaction("learningobjectselect",
LR_AUTO);

          lr_error_message("Error in opening web link");
```

```
		return 9;

	}


	lr_end_transaction("learningobjectselect", LR_AUTO);

	/*Random think for 2 to 40 seconds*/
	if ( ( rc = think ( 40, 2 ) ) != 0 )

	{

		lr_error_message("Error in think time");

	}




	lr_start_transaction("returnhome");


	/*Open the portal home page*/
	if ( ( rc = web_url("dt",

		"URL=https://portal.portal.private/http://por-
tal.portal.private/portal/dt",

		"TargetFrame=",

		"Resource=0",

		"RecContentType=text/html",

		"Referer=https://portal.portal.private/http://
as1.portal.private/elearning/learn/learn/navigateOffer-
ing.do?nodeName=4100%3Acom.sun.elearn-
ing.lms_.learningplatformprovider_.Unit",

		"Snapshot=t44.inf",

		"Mode=HTML",

		LAST) ) != 0 )

	{

		lr_end_transaction("returnhome", LR_AUTO);

		lr_error_message("Error in web url");

		lr_abort();

		return -1;

	}




	lr_end_transaction("returnhome", LR_AUTO);
```

```
     return 0;

}
```

vuser_end.c

```
#include "as_web.h"


vuser_end()
{
     int rc;

     /*Logout of the UKeU Application*/
     if ( ( rc = ukeulogout()) != 0 )
     {
          lr_error_message("Error in logging out of the
application return code %d", rc);
          lr_abort();
     }

     return 0;
}
```

StaticPages Test

This Script logs the users into the system and saves the list of pages that are available. It will then randomly select one of those pages to open and save the list at the top of the page for the next selection. There is a random think time which is designed in an attempt to stop all of the users getting into lock step. This is then looped to keep randomly selecting pages from the list that appears at the top of each page. A transaction time is recorded for each page that is loaded and this is used as the response time for the scenario.

vuser_init.c

```
vuser_init()
{
     int rc;

     if ( ( rc = web_reg_find("Text=UKeU Login", LAST) )
!= 0 )
     {
          lr_error_message("Error Registering Web Reg
Find");
          return 2;
     }

     if ( ( rc = web_url("portal.portal.private",
          "URL=https://portal.portal.private/",
          "TargetFrame=",
          "Resource=0",
          "RecContentType=text/html",
          "Referer=",
          "Snapshot=t1.inf",
          "Mode=HTML",
          EXTRARES,
```

```
          "Url=/redirect/http://portal.portal.private/
portal/images/images/buttons/button_register_r.gif",
"Referer=https://portal.portal.private/http://portal.por-
tal.private/amserver/login?gw=portal.portal.pri-
vate&org=ukeu.ac.uk", ENDITEM,
          LAST) ) != 0 )
     {
          lr_error_message("Error Opening page");
          return 3;
     }

     if ( ( rc = think(50, 30) ) != 0 )
     {
          lr_error_message("Error in think time ignored");
     }

     if ( ( rc = web_reg_save_param("savedparameter",
"LB=.html\">Click here for ", "RB=</a>", "ORD=ALL", LAST)
) != 0 )
     {
          lr_error_message("Error in saving parameter");
          return 8;
     }

     if ( ( rc = web_reg_find("Text=Test page 1", LAST) )
!= 0 )
     {
          lr_error_message("Error in registering Web reg
find");
          return 2;
     }


     if ( ( rc = web_submit_data("login",
          "Action=https://portal.portal.private/http://
portal.portal.private/amserver/login?module=UkeuLDAP",
          "Method=POST",
          "RecContentType=text/html",
          "Referer=https://portal.portal.private/http://
portal.portal.private/amserver/login?gw=portal.por-
tal.private&org=ukeu.ac.uk",
          "Snapshot=t2.inf",
          "Mode=HTML",
          ITEMDATA,
          "Name=REQ_uid", "Value={username}", ENDITEM,
          "Name=REQ_userPassword", "Value=netscape",
ENDITEM,
          "Name=password", "Value=REQ_userPassword",
ENDITEM,
          "Name=userName", "Value=REQ_uid", ENDITEM,
          "Name=Submit", "Value=Login", ENDITEM,
          "Name=Submit.x", "Value=12", ENDITEM,
          "Name=Submit.y", "Value=8", ENDITEM,
          LAST) ) != 0 )
     {
          lr_error_message("Error submitting data to
login");
          return 4;
     }
     else
     {
          lr_vuser_status_message("Username %s",
lr_eval_string("{username}"));
          return 0;
     }
}
```

Action.c

```
Action()
{
     int rc;
     char *linksel;

     lr_think_time( 7 );
     lr_think_time( 4 );

     if ( ( linksel = random_select() ) == "6" )
     {
          lr_error_message("Error in randomly selecting
link");
          lr_abort();
     }

     if ( ( rc = lr_save_string(lr_eval_string(link-
sel),"selectedlink") ) != 0 )
     {
          lr_error_message("Error in saving string");
          return 1;
     }

     if ( ( rc = web_reg_save_param("savedparameter",
"LB=.html\">Click here for ", "RB=</a>", "ORD=ALL", LAST)
) != 0 )
     {
          lr_error_message("Error in saving parameter");
          return 8;
     }

     if ( ( rc = web_reg_find("Text={selectedlink}",
          LAST) ) != 0 )
     {
          lr_error_message("Error in registering find");
          return 2;
     }


     lr_start_transaction("Load_page");

     if ( ( rc = web_link("Click here for {selectedlink}",
          "Text=Click here for {selectedlink}",
          "Snapshot=t4.inf",
          LAST) ) != 0 )
     {
          lr_end_transaction("Load_page", LR_AUTO);
          lr_error_message("Fatal Error in loading page");
          lr_abort();
     }


     lr_end_transaction("Load_page", LR_AUTO);

     return 0;
}
```

vuser_end.c

```
vuser_end()
{
     int rc;
```

```
     /*Logout of the UKeU Application*/
     if ( ( rc = ukeulogout()) != 0 )
     {
          lr_error_message("Error in logging out of the
application return code %d", rc);
          lr_abort();
     }

     return 0;
}
```

functions.c

The functions.c holds routines that are used in a number of scripts. This functions file is downloaded on to all LoadRunner load injectors. Once this file is downloaded to the injectors the functions are then available for all scripts, to save time in rewriting code for each script.

```
#include "as_web.h"

/*This function looks at the LoadRunner saved parameter
calculates how many times it has matched
  the left and right brackets and then randomly selects
one of those and returns that to the main
  program. if there are no parameters saved the system
will return an error code of 6*/
char * random_select ( void )
{
     int i,j;
     char linksel[50];

     /*Evaluate the number of saved parameters*/
     i = atoi(lr_eval_string("{savedparameter_count}"));

     /*If the number of saved parameters is equal or less
than 0 error and return to the main program*/
     if ( i <= 0 )
     {
          lr_error_message("Saved Parameter has no val-
ues");
          return "6";
     }

     /*Randomly select a value between 0 and the number of
saved parameters and add 1 to the value
       as LoadRunner evaluates the saved parameter between
1 and max value rather than 0 and
       maxvalue - 1*/
     j = rand() % i + 1;

     /*Assign the value that is stored in the saved param-
eter at position j to a variable*/
     sprintf(linksel,"{savedparameter_%d}", j);

     /*return the selected string*/
     return linksel;
}

/*This function calculates a random think time between 2
values maxtime and mintime, and then
  executes the think time.  It will return a 0 if every-
thing is passed and a 5 if the maxtime is
  less than the mintime.*/
int think ( int maxtime, int mintime )
{
     int j,delta;
```

```
     delta=maxtime-mintime;

     if ( delta < 0 )
     {
          lr_error_message("Think Time Error due to less
than zero delta");
          lr_output_message("Think Time Ignored");
          return 5;
     }
     else
     {
          j = rand() % delta + mintime;
          lr_think_time (j);
          return 0;
     }
}

/*This function will select and execute a random think
time depending on a minimum value that
  is passed to the function.  The randomness of the think
time is goverened by the minimum value
  passed the max think time is 1.5 * mintime.*/
int think_random_1_5 ( int mintime )
{
     int j;

     j = rand() % ( mintime / 2 ) + mintime;
     lr_think_time ( j );
}

/*This function will search a string for a token for a
defined number of times. The string is
  passed to the function must be first and the token sec-
ond.  The number of iterations is the
  number of time the strtok is called on that string so if
the second occurance of the token is
  required then the number if iteration must be 2.  The
string that is output form the strtok is
  passed back as the return value.  If the output of the
strtok is NULL then a NULL string is
  returned.*/
char * search_for_token ( char string1[50], char
token[2], int iterations )
{
     char *string2;
     int i=1;


     /*Run the first iteration of the strtok command and
check the return value is not NULL*/
     if ( ( string2 = (char *)strtok( string1, token )) ==
NULL )
     {
          lr_error_message("Error obtaining string after
strtok returned string is NULL");
          return "NULL";
     }

     /*Execute the number of iterations required and check
the return value each time*/
     while ( i < iterations )
     {
          if ( ( string2 = (char *)strtok( NULL, token ))
== NULL )
          {
               lr_error_message("Error obtaining string
after strtok returned string is NULL");
```

```
                return "NULL";
          }
          i++;
     }

     /*Return the output of command after the required
number of iterations*/
     return string2;
}

/*Execute the login to the UKeU application.  The user-
name to logon as is passed to the function
  the password used is the default for all test users cre-
ated on the UKeU directory*/
int ukeulogin ( char user[] )
{
     int rc;

     /*Saves the string that is passed to the function so
that loadrunner can use it*/
     if ( (rc =
lr_save_string(lr_eval_string(user),"username")) != 0 )
     {
          lr_error_message("Error Saving username for
LoadRunner to use");
          return 1;
     }

     /*Registers a find for the UKeU Login text to ensure
the front page is found.*/
     if ( (rc = web_reg_find("Text=UKeU Login", "Save-
Count=Count_ukeu_login", LAST)) != 0 )
     {
          lr_error_message("Error in registering find for
UKeU Login");
          return 2;
     }

     /*Opens the login page for the UKeU application*/
     if ( (rc = web_url("portal.portal.private",
          "URL=https://portal.portal.private/",
          "TargetFrame=",
          "Resource=0",
          "RecContentType=text/html",
          "Referer=",
          "Snapshot=t1.inf",
          "Mode=HTML",
          EXTRARES,
          "Url=/redirect/http://portal.portal.private/
portal/images/images/buttons/button_register_r.gif",
"Referer=https://portal.portal.private/http://portal.por-
tal.private/amserver/login?gw=portal.portal.pri-
vate&org=ukeu.ac.uk", ENDITEM,
          LAST)) != 0 )
     {
          lr_error_message("Error opening URL for UKeU
Login Page");
          return 3;
     }

     /*Waits for 30 seconds before logining in*/
     lr_think_time(30);

     /*Submits the login information for the use to login
to the portal*/
     if ( (rc = web_submit_data("login",
```

```
          "Action=https://portal.portal.private/http://
portal.portal.private/amserver/login?module=UkeuLDAP",
          "Method=POST",
          "RecContentType=text/html",
          "Referer=https://portal.portal.private/http://
portal.portal.private/amserver/login?gw=portal.por-
tal.private&org=ukeu.ac.uk",
          "Snapshot=t2.inf",
          "Mode=HTML",
          ITEMDATA,
          "Name=REQ_uid", "Value={username}", ENDITEM,
          "Name=REQ_userPassword", "Value=netscape",
ENDITEM,
          "Name=password", "Value=REQ_userPassword",
ENDITEM,
          "Name=userName", "Value=REQ_uid", ENDITEM,
          "Name=Submit", "Value=Login", ENDITEM,
          "Name=Submit.x", "Value=22", ENDITEM,
          "Name=Submit.y", "Value=11", ENDITEM,
          LAST)) != 0 )
     {
          lr_error_message("Error submitting login infor-
mation");
          return 4;
     }
     else
     {
          /*if the login is successful the change the
vuser status message to show the
            username*/
          lr_vuser_status_message("User Running %s",
lr_eval_string("{username}"));
          return 0;
     }
}

/*This function will log a user out of the ukeu applica-
tion*/
int ukeulogout()
{
     int rc;

     if ( (rc = web_reg_find("Text=UKeU Login",
          LAST) ) != 0 )
     {
          lr_error_message("Error registering web reg
find");
          return 2;
     }

     if ( ( rc = web_url("Logout",
          "URL=https://portal.portal.private/http://por-
tal.portal.private/portal/dt?action=logout",
          "TargetFrame=",
          "Resource=0",
          "RecContentType=text/html",
          "Referer=https://portal.portal.private/http://
portal.portal.private/portal/dt",
          "Snapshot=t45.inf",
          "Mode=HTML",
          LAST) ) != 0 )
     {
          lr_error_message("Error opening web url");
          return 3;
     }

     return 0;
```

```
}
```

Sun Microsystems, Limited Java House, Guillemont Park, Minley Road, Blackwater, Camberley, Surrey UK, GU17 9QG **Web** sun.co.uk

Sun Worldwide Sales Offices: Africa (North, West and Central) +33-13-067-4680, Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55 11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India–Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +822-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand–Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China–Beijing +86-10-6803-5588; Chengdu +86-28-619-9333; Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Singapore +65-6438-1888, Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland–German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44-1-276-20444, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800