

How To Effectively Size Hardware for Your Portal Implementation

An Oracle White Paper
December 2004

How To Effectively Size Hardware for Your Portal Implementation

Overview.....	3
Introduction.....	3
Architectural Overview	4
Capacity Planning and Performance Overview.....	5
Performance Targets.....	5
Sizing the Portal System.....	8
Other Performance Factors.....	9
Sizing & Estimation Methodology	11
Section Summary	12
Benchmark Overview.....	13
What is GlobalXChange ?.....	13
GlobalXChange Metrics.....	14
Simulation	15
Stress Testing.....	18
Test Hardware.....	18
Software.....	19
Workload Description.....	19
Sizing metrics derivation	21
Input Variables.....	21
Output Parameters	21
Tests and Results.....	22
Initial Results	23
Normal Load Scenario.....	24
Hit/s per User Ratio	24
CPU Utilization and Memory.....	26
Heavy Load Scenario	27
Impact of Login Percentage.....	29
Quick Reference.....	29
Detail Calculations.....	29
Infrastructure Memory	30
Midtier Memory.....	30
Conclusion.....	31
Appendix 1	32

How To Effectively Size Hardware for Your Portal Implementation

OVERVIEW

This paper explains the overall capacity planning methodologies available for Oracle Portal, and explains the calculations used to obtain metrics for estimating and sizing a portal.

INTRODUCTION

Oracle Portal meets the three critical requirements for scaling performance:

- The underlying architecture is cross-platform, allowing the portal to take advantage of the full range of available hardware resources. Portal administrators can choose the best platform for each portal component, allowing optimization of both performance and total cost of ownership.
- The architecture supports load distribution and parallel execution of portal components across multiple servers.
- The portal implements intelligent caching of dynamically generated portal pages. Intelligent caching ensures that information remains fresh and timely while minimizing the cost of regenerating content from databases and back-end services. Intelligent caching supports user-level customizations and is also modular, so that entire pages or just page components can be stored in a cache and refreshed as necessary. Finally, intelligent caching protects security by ensuring that cached content is only accessible by authorized users.

The first part of this document examines the elements to be considered when performing an architectural estimation for Oracle Portal offers recommendations and further considerations.

You can obtain further information and assistance with sizing your hardware requirements from Oracle Technology Network (<http://otn.oracle.com>)

ARCHITECTURAL OVERVIEW

At the heart of the Oracle Portal architecture is the Parallel Page Engine, a multithreaded servlet deployed on Oracle Containers for J2EE (OC4J), Oracle's highly scalable J2EE framework. Multiple Parallel Page Engines deployed in a server farm work together to retrieve content from portlet providers, manage caching, and assemble and deliver pages.

Oracle Portal is also fully integrated with Oracle Web Cache, Oracle's patented caching technology. Unlike legacy cache servers, which only handle static data, Oracle Web Cache combines caching, compression, and assembly technologies to accelerate the delivery of both static and dynamically generated Portal content. Oracle Web Cache also provides back-end Web server load balancing, failover, and surge protection features which ensure blazing performance and rock solid up time. With Oracle Web Cache, Oracle Portal can now serve rich content faster, to more users, using fewer computing resources than ever before.

Without the unique combination of caching and scalability features offered by Oracle Portal, it would be impossible to scale a portal site in a cost-effective manner. Oracle Portal's architecture truly minimizes the hardware resources required to deploy a high-traffic portal.

Measuring the scalability of a portal requires a common denominator. For example, for database performance metrics we can refer to TPCC benchmarks. For J2EE application server performance, we can refer to ECPeef transaction figures. Unfortunately, there is no ECPeef for portals. Until portal development and deployment standards are truly unified, through the efforts of JSR168, WSRP, and other open portal development standards, it will be impossible to develop a Portal ECPeef. There are simply too many differences among the variety of implementation methods employed by portal vendors in the marketplace.

CAPACITY PLANNING AND PERFORMANCE OVERVIEW

As with any Web portal, the server and database capacity needed to deploy a portal built using Oracle Portal largely depends on the number of anticipated user requests for a given page. Displaying a single page to a user may require many separate transactions, from verifying whether the user has permission to view the page, to loading the images that appear on the page, to calling a style sheet that contains formatting information for the page.

The upper and lower limits of what is needed are determined by how users are expected to use the portal. At a minimum, enough server capacity to satisfy the average load during a work day will be required, with response times that are acceptable to the user base. If possible, strive to satisfy the volume of page requests anticipated during peak intervals of high user activity. Hardware resources such as CPU, memory, I/O capacity, and network bandwidth are key to reducing response times. Unless you install Oracle Portal on a server or group of servers that can handle a large number of transactions, users are probably going to experience slow response times.

The same is true of the database. If many applications compete for the same database resources, Web portal performance may suffer. It is possible to install multiple instances of Oracle Portal in the same database. For example, you could have a development instance for developing new pages and portlets, and a separate instance for deploying the finished web portal. Consider whether the database can satisfy requests from both instances in a timely manner.

Adding more servers and database capacity will certainly improve the web portal's performance, but unless there are unlimited funds available, balancing good performance against the costs associated with each new piece of hardware and software will become key.

Performance Targets

Whether designing or maintaining a system, you should set specific performance goals for optimization. Altering parameters without a specific goal in mind can waste tuning time for the system without a significant gain.

An example of a specific performance goal is an order entry response time under three seconds. If the application does not meet that goal, identify the cause (for example, I/O contention), and take corrective action. During development, test the application to determine if it meets the designed performance goals.

Tuning usually involves a series of trade-offs. After determining the bottlenecks, performance in some other areas may need to be modified to achieve the desired results. For example, if I/O is a problem, purchasing more memory or more disks may resolve that. If a purchase is not possible, limiting the concurrency of the system to users may achieve the desired performance. However, if there are clearly defined goals for performance, the decision on what to trade for higher performance is simpler because the most important areas will have been identified.

User Expectations

Application developers, database administrators, and system administrators must be careful to set appropriate performance expectations for users. When the system carries out a particularly complicated operation, response time may be slower than when it is performing a simple operation. Users should be made aware of which operations might take longer.

Performance Evaluation

With clearly defined performance goals, determining when performance tuning has been successful becomes a simple matter of comparison. Success depends on the functional objectives established with the user community, the ability to measure whether or not the criteria are being met, and the ability to take corrective action to overcome any exceptions.

Ongoing performance monitoring enables maintenance of a well-tuned system. Keeping a history of the application's performance over time enables useful comparisons to be made. With data about actual resource consumption for a range of loads, objective scalability studies can be undertaken and from these predict the resource requirements for anticipated load volumes.

Performance Terms

concurrency

The ability to handle multiple requests simultaneously. Threads and processes are examples of concurrency mechanisms.

contention

Competition for resources.

cluster

A group of machines that handle workload in a distributed manner, providing redundancy and failover.

failover

A method of allowing one machine or set of machines to provide an alternative execution arena for a task, should the original machine(s) fail.

hit

The subsequent request for a snippet of content from either the Portal Parallel Page Engine, or the client browser (such as images, javascript libraries, cascading style sheets, etc.) It is reasonable to expect ~30 hits from a single page request.

latency

The time that one system component spends waiting for another component in order to complete the entire task. Latency can be defined as wasted time. In networking contexts, latency is defined as the travel time of a packet from source to destination.

page request

The unique request for a page defined inside the Portal repository. A figure specifying page requests per second is the measurement of the load expected for the architected solution given a common element of portal content. i.e. given identical portal content and structure, the page request rate is a suitable measurement of specific portal hardware. One page request is likely to result in one or more (~30) hits for subordinate content.

response time

The time between the submission of a request and the receipt of the response.

scalability

The ability of a system to provide throughput in proportion to, and limited only by, available hardware resources. A scalable system is one that can handle increasing numbers of requests without adversely affecting response time and throughput service time

The time between the receipt of a request and the completion of the response to the request.

think time

The time the user is not engaged in actual use of the processor.

stream time

The time taken to transmit the response to the requestor

throughput

The number of requests processed per unit of time.

wait time

The time between the submission of the request and initiation of the request.

SIZING THE PORTAL SYSTEM

Consider the following elements of page generation when planning for a system sizing.

- Peak page throughput required
- Page cache hit rate
- Peak login rate

Also, be sure to consider other performance factors final portal, including portlet cache hit rate, portlet execution speed, page complexity, page security, available network bandwidth and load distribution, other portal activity, available hardware resources, amount and type of content, and the impact of using SSL.

Peak Page Throughput

Peak Page Throughput is the peak number of pages/second requested by portal users. For example, assume a portal serves a total population of 10,000 users, of which 10% are active (a user may be logged in but not active) at peak times. Assume an active user makes 3 requests per minute. The peak throughput requirement is: $((10000 \times 0.10) \times 3) \div 60 = 50$ pages/sec

Page Cache Hit Rate

The Page Cache Hit Rate is the number of page definitions that can be retrieved from the cache compared to the number of pages that must be regenerated during peak load times. To estimate the PageCHR consider

- How often pages are modified by their owners
- How often pages are customized by end users
- Whether you will be using validation, invalidation, or expiry based caching

The aim with PageCHR is to get it as close to 100% as possible. Judicious use of the correct caching policy within each unique page and portlet, when weighed against the the relative dynamic nature of the data, ensures that the most up to date content possible is delivered in a timely and performant fashion. Building page content from cached content (both in-memory and on-disk) is much less expensive than retrieving the content from the metadata repository.

Peak Login Rate

The Peak Login Rate is the rate at which users login to the portal, thereby placing a load on the SSO and OID servers. For example, assume a portal serves a total population of 10,000 users and 20% of those users login during a 15-minute period at the start of the business day. The peak login rate is:

$$10000 \times 0.20 \div 15 = 133.33 \text{ logins/min (2.22 logins/sec)}$$

Explicit logouts also place a load on the servers and may need to be considered also.

Other Performance Factors

Portlet Cache Hit Rate (PortletCHR)

The PortletCHR is the number of portlet requests that can be satisfied from the cache compared to the number of the portlet requests that must be handled by a provider during peak load times

Portlet Execution Speed

The Portlet Execution Speed is the average time required to execute all (uncached) portlets on a page. Since portlets execute in parallel, this measure is equal to the execution speed of the slowest portlet, plus any page assembly overhead. The portlet execution speed may differ from site to site if each site has a differing mix of content, caching policies and hardware for their portal. Estimating this number can only be achieved through a proof of concept that accurately reflects the eventual target data and page design. In general, the speed of page assembly is limited by the execution speed of its slowest portlet.

Page Complexity

Page security and the number of tabs and portlets on a page affect the time it takes to generate page metadata. The number of portlets on a page affects the page's assembly time, especially if each page must be generated or contacted for a validity check.

Network Bandwidth

The speed of the network that connects interacting portal components affects response times but does not affect in-machine throughput. Bandwidth issues are a large concern for portal implementations with a geographically dispersed user base. The further the content must travel, the more latency sensitive the delivery mechanism.. Largely distributed systems over higher latency networks suffer from poor performance, which is exacerbated by dynamic portlet content.

Load Distribution

The distribution of system load across servers affects overall system performance. The normally accepted method of dealing with load distribution and scalability is to place each AS component on a separate machine or machines, depending on how much scalability is required. This distributed load architecture also assists when dealing with the issues of High Availability.

Other Portal Activity

The impact of the other users of the portal affects overall performance response. Content managers, developers, and monitoring overhead can all consume valuable processing resource that could be applied to page generation. This is a normal situation, as the nature of the portal provides for multiple concurrent usage models. However, from a pure performance point of view, the execution models for page generation differ widely from that for application development. Given that, doing both

activities simultaneously reduces the overall systems resource available for one specific task.

Hardware Resources

Both page generation from the PPE and caching through Web Cache are memory sensitive, in-memory operations that are orders of magnitude faster than those involving I/O bound disk caching or swap files. Providing suitable quantities of memory for the portal servers is a critical factor in your machine configuration.

CPU Performance

Page generation is a CPU-intensive process. Therefore, the speed of the available CPU speed and quantity of those CPUs is another critical factor in the machine configuration for a portal server

Type of Content

The amount and type of content that is served could affect system throughput. Multimedia content could place an additional load on the OHS, network bandwidth, file system, memory cache, and DB processes.

SIZING & ESTIMATION METHODOLOGY

Estimating anything can be a complex and error-prone process. That's why it's called an 'estimation', rather than a 'calculation'.

There are three primary approaches to sizing a portal implementation:

- Algorithm, or Calculation Based
- Size By Example Based
- Proof of Concept Based

Algorithm, or Calculation Based

An algorithm or process that accepts input from the customer (e.g. user count, page count, hits, latency, doc size, etc.) and attempts to deliver a processing requirement, is probably the most commonly accepted tool for delivering sizing estimations. Unfortunately, this approach is generally the most inaccurate. When considering a logical n-tier enterprise class portal implementation, the number of variables involved in delivering a calculation that even approaches a realistic sizing response requires input values numbering in excess of one hundred, and calculations so complex and sensitive that providing an input value plus or minus 1% of the correct value results in wildly inaccurate results.

The other approach to calculation-based solutions is to simplify the calculation to the point where it is simple to understand and simple to use.

This paper shows how this kind of simplification can provide us with a sizing calculator.

Size-By-Example Based

A size-by-example (SBE) approach requires a set of known samples to use as data-points along the thermometer of system size. The more examples available for SBE, the more accurate the intended implementation will be.

Oracle has the ability to deliver targeted SBE sizing solutions for our prospective portal customers through reference implementation documents that outline both our internal deployments and customer's external deployments.

By using these real world examples, both customers and Oracle can be assured that the configurations proposed have been implemented before and will provide the performance and functionality unique to the proposed implementation.

You can find examples of real-world Size-By-Example documents on OTN.

Proof of Concept Based

A proof of concept (POC), or pilot based approach, offers the most accurate sizing data of all three approaches.

A POC you to do the following:

- Test your portal implementation design
- Test your chosen hardware platform
- Test your caching strategy
- Simulate projected load
- Validate design assumptions
- Validate Oracle Portal
- Provide iterative feedback for your implementation team
- Adjust or validate the implementation decisions made prior to the POC

There is, however, two downsides to a POC based approach, namely time and money. Running a POC requires the customer to have manpower, hardware, and the time available to implement the solution, validate the solution, iterate changes, re-test, and finally analyze the POC findings. A POC is always the best and recommended approach for any sizing exercise. It delivers results that are accurate for the unique implementation of the specific customer that are as close to deploying the real live solution as possible, without the capital outlay on hardware and project resources.

Section Summary

The process of sizing and estimating a new portal installation is a non-trivial task with many elements to consider, all of which affect the performance of a given system. The rest of this document introduces the new portal capacity planning formula, describes its derivational method, and examines how to interpret the data.

BENCHMARK OVERVIEW

As discussed earlier, it is difficult for portal vendors to compare their products on a performance basis. Each vendor's architecture is different and it is impossible to develop a generic benchmark application (like J2EE's ECPeef) that can be used by all vendors to compare their relative performance characteristics.

With this in mind, Oracle Portal Development designed and developed a generic benchmark package to allow the benchmarking and modeling of systems using the data from the generic benchmark as a model.

What is GlobalXChange ?

GlobalXchange (GXC) is Oracle's premier internal knowledge management portal. This portal provides the necessary infrastructure, including processes and technologies, to facilitate information sharing, knowledge exchange, and reuse by embedding knowledge management practices into the Oracle culture. It is a way of combining people, processes and tools to make Oracle's collective knowledge accessible by anyone in the Oracle organization. GXC is the gateway to Oracle's knowledge - from quality content to e-collaboration. It is the place for communities and organizations to share what they have and what they know by collaborating on-line, marketing "what sells," and sharing "how to get the job done." The portal provides views of qualified and managed information relevant to a particular individual or group and encourages knowledge exchange among our greatest assets - our staff.

GlobalXChange is utilized by every employee in Oracle, which means the amount of content it serves exceeds that of some of Oracle's externally facing sites. Its size and usage models lend themselves to duplication as part of a benchmark.

As Oracle Development worked to create the benchmark, they carefully analyzed GlobalXChange to determine various sizing factors.

GLOBALXCHANGE METRICS

The following table gives you an idea of the structure of the content stored within GXC:

Page Group	Folder Count	SubFolder Count per Folder	Item Count per Folder	Min/Max Items per Folder	Avg Docs per Folder	Min/Max Docs per Folder
Shared Objects	3257	6.67	4.33	1/374	4.25	1/348
GlobalXChange	45503	4.25	6.698	1/410	6.5	1/410

Further analysis of the constituent document elements yields the following information:

Documents	Count	Avg Doc Size	% of Total
Total	183708	500K	4.33
Images of Total	36554	32K	20%
Documents of Total	147327	580K	80%
Public/Private Ratio	129688/53056		71%/29%
Item types per Folder	1.4		

The GXC portal predominantly uses page groups and content pages. The data above shows that all the content for this portal is in two page groups: Shared Objects and lobalXChange.

The public home page for this portal contains both portlet and tabbed regions. The layout is shown in Figure 1:

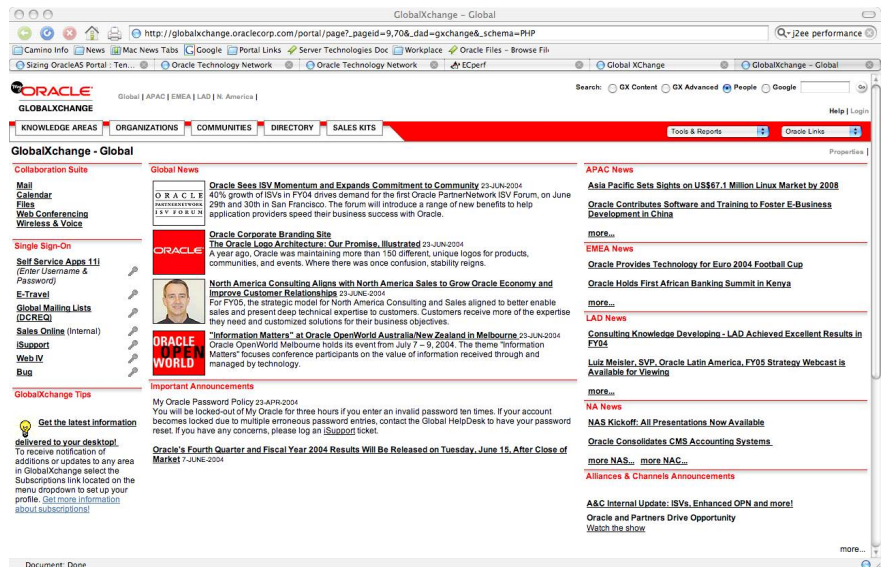


Figure 1 : GlobalXChange Home Page

From the statistics, the GXC page group contains about 45000 folders and Shared Objects about 32000 folders. Shared Objects contains only the personal pages for authenticated users of the system.

The item types in the item regions are mainly files (documents), images, URL items, and folder links. Some of the items display the Author, Create Date, and Document Size attributes following the item. Items in a region are grouped by category.

Simulation

To perform the benchmark, the GXC page makeup and structure as well as the user community had to be reproduced

Portal page groups and pages are defined in the portal repository stored in an Oracle database. The seeding of the page structure was therefore possible via a PLSQL package.

Such a package was developed, called WWTST_POB_GXC, providing the following mechanisms:

```
/**
 * Create a site to simulate Global Exchange.
 *
 * This creates the site and all the pages in the site.
 *
 * If the number of pages is not specified, this will be based
 * on data from the Global Exchange site. Otherwise, it will create
 * the number of pages specified. This site creates both public
 * and secure pages based on the number specified.
 *
 * If the site already exists, then the procedure is a no op, but no
 * exception is raised.
 *
 * @param p_start_public_page_num the start number for the public pages
 * @param p_num_public_pages number of public pages to create
 * @param p_start_secure_page_num the start number for the secure pages
 * @param p_num_secure_pages number of secure pages to create
 * @param p_doc_dir the directory path where the files
 * and images for the items are located
 */
procedure create_gxc_site(
    p_start_public_page_num in integer default null,
    p_num_public_pages in integer default null,
    p_start_secure_page_num in integer default null,
    p_num_secure_pages in integer default null,
    p_doc_dir in varchar2 default null
);

/**
 * Delete a site which simulates Global Exchange.
 *
 * This deletes the site and all the pages in the site.
 */
procedure delete_gxc_site;
```

The WWTST_POB_GXC package allows an administrator to create large quantities of representative pages using a script similar to the following:

```
declare
    DOC_DIR constant varchar2(200) := '/net/export/home/oracle/perffiles/';
    l_start_page_num integer := 0;
begin
    --
    -- Get the timing and gather statistics for the first
    -- 1000 pages upto 10,000 pages.
    --
    for i in 1..10 loop
        wwtst_common.heading2('Start Page Num: ' || l_start_page_num);

        wwtst_pob_gxc.create_gxc_site(
            p_start_public_page_num => l_start_page_num,
            p_num_public_pages      => 500,
            p_start_secure_page_num => l_start_page_num,
            p_num_secure_pages      => 500,
            p_doc_dir               => DOC_DIR
        );

        l_start_page_num := l_start_page_num + 500;

        --
        -- Gather Stats
        --
        wwsbr_stats.gather_stats;
    end loop;
```

Executing this script creates 10,000 GXC simulated pages, with 50% created as publicly accessible and 50% created with secure access for members of random test groups.

The users and groups are created programmatically using another PLSQL script designed to take a given number of users and groups and randomize their memberships into those groups. In turn, the groups are randomly granted access to the secure pages created by the page creation script.

The end result of running the page creation script is a page group that looks similar to Figure 2:

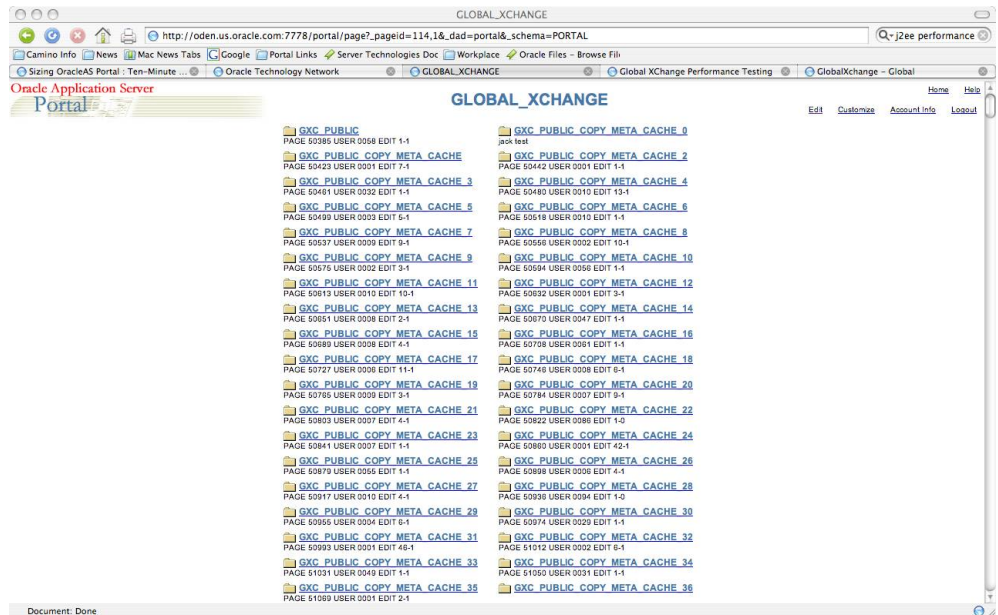


Figure 2 : GlobalXChange simulation root page

Each page that is a sub page of the root folder contains a mixture of these elements: (shown in Fig 3).

- 3 JPDK portlets with non-cacheable characteristics
- 1 link item
- 1 zip file
- 5 MS Word docs
- 1 image file (gif)

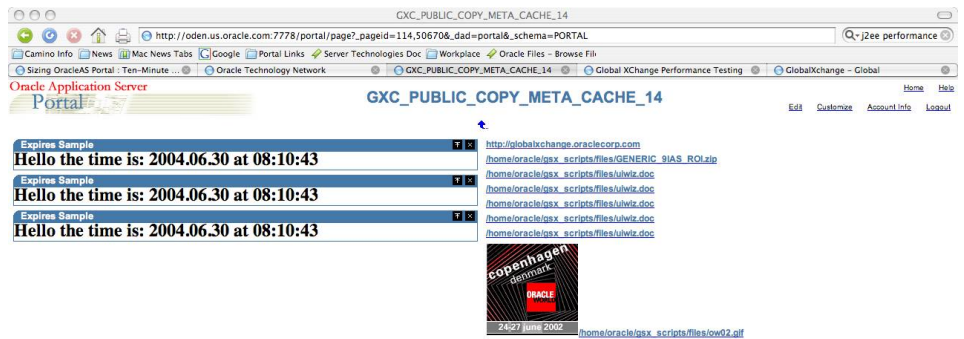


Figure 3 : Sample of GXC Seed Pages

This distribution of file items and portlets is proportional to the values obtained from the live instance of GlobalXChange.

Stress Testing

Once the site simulation was set up, it was necessary to develop a method for loading the running system. For this task, Mercury Loadrunner was selected.

A Java utility was created to allow dynamic creation of Loadrunner testing scenarios. The use of this Java utility replaces the labor-intensive process of creating a scenario for each unique parameter change.

Given certain parameters, such as login percentage and user think time, the Java utility provided us with two output files (parameters.dat and vuser_init.c) which were used within the Loadrunner controller to create a new load scenario.

Test Hardware

The following hardware was commissioned to act as test servers throughout the benchmark period. The Loadrunner controller was deployed on another separate machine.

Host	CPU	MHz	RAM	OS
Infra & DB	4	2400	6 GB	Red hat AS 2.1
Midtier	4	900	4 GB	Red hat AS 2.1

Software

The following software was loaded on the test servers

Product	Version	JVM	JVM Args (2 OC4J_Portals midtier)
Oracle Portal 10g	9.0.4	1.4.1_03	-server -Xms950m -Xmx950m
LoadRunner	LoadRunner 7.6	N/A	N/A

Workload Description

The workload had predefined/encoded GXC content and a default profile, which can be broken down as:

- Login percentage: 10%
- Portlets per page: 3
- Think time:~ 30 seconds
- Measured page type:
 - Anonymous_Public_Item_100k
 - Anonymous_Public_Item_18k
 - Anonymous_Public_Page
 - Auth_Public_Page
 - Auth_Secure_Item_100k
 - Auth_Secure_Item_18k
 - Auth_Secure_Page
 - Login

Term	Definition
Login percentage	Starting percentage of the total user base that has a valid authenticated session in the portal repository
Portlets per page	Count of non-cacheable JPDK portlets deployed on the test pages
Think time	Elapsed time between each virtual user mouse click. If think time = 30 secs, then each user clicks a mouse button to request a new page twice a minute. This is an averaged amount based on real world usage models. A user may click 5 times in quick succession but then read the page displayed for 3 minutes, resulting in an average click rate of 1.6 clicks/minute.
Measured page type	Type of pages that are available for random requests within the Loadrunner script. An Anonymous_Public_Item_100k refers to a public document of size 100K, whereas Auth_Secure_Item_100k refers to the same document but one for which a valid authenticated session with permissions must exist. The difference is that the workload required of the portal to retrieve the secured request is much greater than that of the unsecured request.

Number of Users and Concurrency

For testing purposes, the users' activity concurrency was a critical factor that determines the page hit rate (Req/Sec). Concurrency cannot be measured directly from within Loadrunner, but can be controlled by changing the think time for each virtual user.

For example, in the portal DB test, results show that a 10 user, high concurrency scenario generates similar CPU% and Average Response Time (ART) as a 300 user, low concurrency scenario.

Concurrency	# of User	UserCPU%	Hit/s	ART (s)
Zero ThinkTime	10	90	44	1.16
Default ThinkTime	300	88	37	0.93

SIZING METRICS DERIVATION

To provide a serviceable calculation model when benchmarking concludes, it is important that the numbers of inputs and outputs be kept to the most relevant bare minimum set of factors that affect the load on a given system.

Before recording any tests, several iterative tests were performed to provide feedback on the health of the system. Based on the measurements of these iterative tests, several tuning exercises were carried out within both the middle tier and the infrastructure. The tuning parameters changed are documented in Appendix 1.

Input Variables

There is a relationship between page requests and hits. The portal considers a page request as a unique request for a single page in the repository. When returned to the browser, the page HTML may result in subsequent browser requests for further content like images, JS files etc. Therefore, there is a ratio between page reqs and hits. In the case of the benchmark, this ratio was 1 to 4.5.

Essential Input

- Number of users (U) – Total user population
- User activity intensity, in one of following formats:
 - Hit/sec – respective hits seen by Web Cache for all content
 - Page request/time - 1 page view = 4.3 hit/s for default scenario

Optional Profile Detail

- Average think time
- Login percentage

Output Parameters

- CPU utilization
- Memory usage
- Average response time (ART)
- Transaction Throughput

TESTS AND RESULTS

To produce a valid resource consumption profile for the baseline running system, certain measurements were taken to provide CPU and memory baselines.

The measurements were based upon the following formulas:

CPU Consumption

$$\begin{aligned} \text{CPU MHz Needed} &= (\text{\#Hits/s}) * (\text{Cost per hits/s}) \\ \text{\#CPU} &= \text{CEILING}((\text{CPU MHz required}) / (\text{MHz per CPU})) . \end{aligned}$$

Memory Consumption

$$\text{Memory Required} = \text{Base AS Memory} + \text{Incremental Memory}$$

The base application server memory for the midtier was 313MB, which included the following processes:

- OC4J_Portal
- OC4J_home
- WebCacheAdmin
- WebCache
- HTTP_Server

The base application server memory for the infrastructure was 352 MB, which included the following processes:

- Metadata and Portal DB
- TNS Listener
- OID
- OC4J_Security
- HTTP_Server

The incremental memory required for the running portal applications was determined by observing various test results and deriving a mathematical equation to extrapolate it.

Initial Results

Default workload test results (incremental usage at a stable duration of 30 minutes):

# Users	Hit/s	Mid mem(MB)	Mid CPU%	Infra Mem (MB)	Infra CPU%
100	15.1	162.2	3.3	183.3	2.2
200	30.0	182.7	5.0	229.5	4.3
300	45.6	188.2	6.5	237.2	6.3
400	60.5	177.3	8.9	273.7	8.8
500	76.9	192.8	10.7	299.7	11.7
600	91.2	228.4	13.0	344.9	14.8
700	106.6	202.7	15.5	310.1	18.1
800	121.5	228.5	18.0	371.3	18.8
900	135.9	244.5	20.3	421.7	25.3

CPU Utilization

Test results show that CPU utilization is mostly determined by hits/second. It was found that the test results could be approximated by following the formula for CPU utilization in terms of Hit/s (H):

Before reaching the limits (75% utilization), CPU MHz per Hit/s can be treated as a constant.

For Midtier, CPU MHz per Hit/s = 5.31

For Infrastructure, CPU MHz per Hit/s = 15.04

Memory Usage

Memory usage involves many parameters. The following sections describe the differences between normal load and peak load.

Normal Load Scenario

Active Concurrent Process Memory Usage

Within a safe zone (that is, CPU utilization of less than 80%), the Apache forked processes (httpd and java) have a near-zero time lifespan. Because the load test simulates continuous concurrent incoming requests, this causes a couple of hundred forked processes to be kept active. So, the memory usage (or System Baseline) is determined by the number of active processes. The number of active processes is determined by the following factors:

- P: Configuration parameters setting in httpd.conf and opmn.xml;
- U: Number of users (browser clients). This factor can be controlled and observed;
- C: Concurrency or intensity of user activities. This factor is again determined by:
 - ThinkTime or waiting interval, this can be controlled and observed;
 - Response Time, this can be observed but not controllable; it is activity-specific and content-specific.

Hit/s per User Ratio

For any given workload within this benchmark, the Hit/s per User is a constant. In the field, however, this ratio is a variable that is specific to each customer installation.

At different ratios, the memory usage characteristics differ. Given time, it would be possible to derive a sophisticated formula to generalize these differences; however, time constraints force us to identify three distinct ranges that simplify the formula:

$$R = \text{Server Hit/sec Per Active User}$$

Value of R	Type of Scenario	An increase of 1 unit of user load ...
$R > 1$	Heavy hit scenario	Increases the unit of hit/s on the system by a value larger than 1
$R < 1$	Large user case scenario	Increases the unit of hit/s on the system by a value less than 1. This was our scenario during the benchmark test; our default workload was $R=0.15$.
$R = 1$	High stress scenario	Proportionally increases 1 unit of hit/s onto the system

CPU Utilization and Memory

When CPU utilization rises above 80%, the system overhead increases significantly to handle other tasks. The lifespan of each child process is longer and, as a result, the memory usage supporting those active concurrent processes increases significantly.

At stable load, 10% login, and CPU utilization below 80%, the memory usage formula is as follows:

$$\text{MemoryUsage} = \text{SystemBaseline} + \text{ASBaseline} + (nU) \times R$$

For Midtier:

$$\text{MemoryUsage} = \text{System Baseline} + 313\text{MB} + 2U \times R \quad (R < 1)$$

$$\text{MemoryUsage} = \text{System Baseline} + 313\text{MB} + 3U \times R \quad (R = 1)$$

$$\text{MemoryUsage} = \text{System Baseline} + 313\text{MB} + U \times R \quad (R > 1)$$

For Infrastructure:

$$\text{MemoryUsage} = \text{System Baseline} + 352\text{MB} + 1.8U \times R \quad (R < 1)$$

$$\text{MemoryUsage} = \text{System Baseline} + 352\text{MB} + 2.5U \times R \quad (R = 1)$$

$$\text{MemoryUsage} = \text{System Baseline} + 352\text{MB} + 0.8U \times R \quad (R > 1)$$

Example

For a midtier machine with 1000 users, with default workload where R is 0.1, and system baseline (that is, memory consumed by the O/S) is 150MB: .)

$$\text{Memory usage} = 150 + 313 + 2 * 1000 * 0.1 = 663\text{MB}$$

Avoiding the Linux Apache Fork Bomb

When system load generates a high CPU utilization (>90%) some of the constituent processes do not have enough CPU resource to complete within a certain time and remain "active".

Because of this, Apache forks out more processes to handle incoming requests.

Apache keeps doing this until, in a matter of minutes, the number of httpd and java processes grows exponentially.

if there is enough memory and available CPU, some of the existing processes are completed and memory de-allocated: thus, fork-bomb is avoided.

if there is not enough memory and the minimum 512MB free memory limit in Linux OS is exceeded, memory swapping occurs.

The system keeps repeating this process, resulting in hanging processes that are unresponsive. OPMN (not receiving a response) will try to restart Apache and/or OC4J_Portal, thus worsening the situation.

The system then crashes, requiring a physical power cycle.

Heavy Load Scenario

The heavy load scenario was designed as a series of tests to find the breaking point of the benchmark system. The goal was to ultimately discover the maximum throughput obtainable with the given hardware, tuning parameters, and installed portal model.

The system was loaded until an element of it crashed (see sidebar).

Result Data

Fixed Hit/s, only changing the number of active users (stable duration 10 minutes, default parameters):

# users	Hit/s	Mid CPU %	Mid Mem (MB)	Infra CPU %	Infra Mem (MB)
100	153	21	36	25.63	94
200	150	22	31	28.25	48
300	150	23	51	31.4	38
400	144	21	32	32.35	55

Fixed Number of Users, only change Hit/s (stable duration 10 minutes, tuned parameters):

# users	Hit/s	Mid CPU %	Mid Mem (MB)	Infra CPU %	Infra Mem (MB)
300	12.0	5.4	0.0	11.0	178.0
300	45.6	6.5	188.2	6.3	237.2
300	180	31	283	40	499

Test results shows that memory usage is determined by both hit/second and number of users. The test results could be approximated by following the formula for Hit/s (H) and number of users (U). Further, it was observed that with a longer running time, more memory is used. The formulas are based on a 30-minute stable run. Many other configuration parameters, such as HTTP_KEEP_ALIVE, MIN_SPARE_SERVERS, etc., also affected memory usage.

Average Response Time (ART)

The test page Auth_Secure_Page, being proportionally the most expensive page to render, was a good indicator for system responsiveness.

$$\text{ART (ms)} = 70 + 10000 / (320 - \text{Hit/s})$$

Transaction Throughput

Tests shows that the 320 Hit per second is the maximum the system can support.

Because the workload is predefined, the average transaction throughput is observed as proportional to Hit/sec.

$$\text{Average Transaction Throughput (Kbytes/s)} = 12 \times \text{Hit/s}$$

Final Page Overhead Calculation

The culmination of the test model is the calculation below, which has been proven to deliver accurate sizing results, assuming the use of the GXC simulation model as well as the GXC simulation hardware and O/S.

$$\text{Tier \# Chips Req'd} = (((((\text{Active users} * \text{click rate}) / 60) * \text{Hit Ratio}) * \text{MHz Hit}) / \text{Chip Speed})$$

$$\text{Tier Memory Usage} = \text{System Baseline} + \text{AS Baseline} + (nU) \times R$$

Impact of Login Percentage

Obviously there are many possible login percentages. However, when running a benchmark, it is difficult to cover all possible eventualities. The benchmark testing used four values for a logged in percentage, ranging from zero to 100, including the GXC default of 10% logged in users.

For each login scenario, 700 users were used as the active user base, based around a percentage of the total user base of GXC and the disparity between the hardware being used for the benchmark. i.e. the user base was reduced on a pro-rata basis proportional to the reduction in available hardware.

Quick Reference

The following table gives a good rule of thumb for estimating the effect of the logged in percentage.

Login Sizing Reference Table				
Login %	0	10	50	100
Rounded CPU# Utilized (Infrastructure)	1	1	2	3
Infra Mem used (MB)	80	300	420	600
Rounded CPU# Utilized (Midtier)	1	1	1	1
Midtier Mem used (MB)	270	200	300	400

The table shows that taking 10% login as a baseline, if the login ratio is increased by 5 times, then an increase is seen in resource requirements. The infrastructure requires an increase of 1 * 2.4GHz CPU and 50% more memory; the midtier requires 50% more memory.

Detail Calculations

Infra CPU

With the 10% login as a baseline, if login percentage increases L percent, then the infrastructure CPU utilization increases $\sim \sqrt{L}$ percent.

For example, if at 10% login the infrastructure CPU is 20%, then at 50% login, the infrastructure CPU is about $20 + \sqrt{400} \sim 40\%$.

No effect on midtier CPU was observed.

Infrastructure Memory

With the 10% login as a baseline, if the login percentage increases L percent, then the infrastructure memory usage increases about $5L^{1/3}$ percent.

For example, if at 10% login the infrastructure memory usage is 300MB then at 50% login, the infrastructure memory usage is about $300 + 300 \times 5 \times 400^{1/3} \% \sim 410\text{MB}$.

Midtier Memory

With the 10% login as a baseline, if the login percentage increases L percent, then the midtier memory usage increases about $7L^{1/3}$ percent.

For example, if at 10% login the midtier memory usage is 200MB, then at 50% login, the midtier memory usage is about $200 + 200 \times 7 \times 400^{1/3} \% \sim 303\text{MB}$.

CONCLUSION

This paper has shown that sizing any form of portal solution is a complicated process. By creating a benchmark, Oracle provides a static reference point that allows customers and partners to compare their own hardware and portal installations against a benchmark.

If you decide to implement the practices described in this white paper to help size your hardware needs, it is recommended that you validate your conclusions with Oracle Consulting. You can do this by contacting your Account Manager for assistance.

The sizing algorithms provided in this paper serve as a point of reference for the given simulation on the given hardware with the given load and O/S. Results with other portal installations or hardware may deviate from those recorded in this paper.

APPENDIX 1

The following parameters were tuned for running tests.

Web Cache Configuration	
LOADLIMIT	1024
MAXINBOUNDCONNECTIONS	1024
OHS Configuration	
KeepAlive	OFF
MaxClient	1024
OC4J Configuration	
#Portal OC4J Instances	2
DB Server	
#Processes	3000
#Sessions	3305
sga_max_size	800MB



How to effectively size hardware for your Portal implementation.

December, 2004

Authors: Jason Pepper, Jack Sun, Biswajit Nayak

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2004 Oracle Corporation
All rights reserved.