BEA WebLogic Portal™®

Capacity Planning Guide

Version 9.2
Document Revised: December 2006

Contents

Capacity Planning for WebLogic Portal

Capacity Planning for WebLogic Portal

BEA WebLogic Portal is an enterprise class portal infrastructure built on a flexible framework designed to meet the highest performance expectations of our customers. Deployments of this product range from smaller departmental applications with a few machines to very large deployments comprising of many machines in a clustered configuration. The architecture and physical deployment of any given application will depend upon several factors which will be explained later in the document. The process of determining what type of hardware and software configuration is required to meet application needs adequately is called capacity planning.

This document covers the steps involved with capacity planning for WebLogic Portal 9.2 and will serve as a baseline set of measurements so that more accurate estimates can be made for capacity planning by our customers.

Capacity planning is not an exact science. Every application is different and every user behavior is different. This document is meant only as a guide for developing capacity planning numbers and will encourage you to err on the side of caution. Before deploying any application into a production environment the application should be put through a rigorous performance testing cycle. For more information on performance testing see this "Approaches to Performance Testing" on the subject.

Note: Any and all recommendations provided in this guide should be adequately verified before a given system is moved into production. As stated above, the data published in this document is meant to represent the specific configuration that was tested. There are a number of factors that come into play when determining how much capacity a system can support and thus there is no substitute for adequately testing a prototype to obtain your own capacity planning numbers.

Capacity Planning Factors to Consider

A number of factors influence how much capacity a given hardware configuration will need in order to support a WebLogic Portal and a given application. The hardware capacity required to support your application depends on the specifics of the application and configuration. You should consider how each of these factors applies to your configuration and application.

The following sections discuss several of these factors. Understanding these factors and considering the requirements of your application will aid you in generating server hardware requirements for your configuration.

Table 1 Capacity Planning Factors and Information Reference

Capacity Planning Questions	For Information, See:
Have you performance tested your application?	"Performance Testing Suggestions" on page 2
Does the hardware meet the configuration requirements?	"Hardware Configuration and Performance Requirements" on page 3
Is WebLogic Portal configured for clustering?	"Clustered Configurations" on page 4
Is the simulated workload adequate?	"Simulated Workload" on page 4
How many users need to run simultaneously?	"Concurrent Sessions" on page 5
Is WebLogic Portal well-tuned?	"Tuning Your WebLogic Portal/Server" on page 6
How well-designed is the user application?	"Application Design" on page 6
Do clients use SSL to connect to WebLogic Portal?	"SSL Connections and Performance" on page 7
What is running on the machine in additional to WebLogic Portal?	"WebLogic Server Process Load" on page 7
Is the database a limiting factor?	"Database Server Capacity" on page 8
Is there enough network bandwidth?	"Network Load" on page 9
What JVM is used and with what parameters?	"Selecting Your JVM" on page 9

Performance Testing Suggestions

Capacity planning is the last step in the performance testing process. Before an application is ready to be sized for a production deployment it must go through an iterative performance testing process to ensure that all of the bottlenecks are out of the system and the application is running as fast as possible.

Running benchmarks against the application will set a baseline set of measurements so that as features are added and removed from the application the impact of those changes can be objectively measured.

Profiling the application during development will help flush out performance problems or performance hotspots that could turn into major issues down the road. Catching these sort of problems early will significantly reduce the overhead in trying to fix them later.

Recommendation

Much has been written on this subject but a good starting place is Approaches to Performance Testing on BEA's dev2dev site.

Hardware Configuration and Performance Requirements

The operating systems and hardware configurations that BEA supports for WebLogic Portal 9.2 are documented at the Supported Configurations for WebLogic Platform 9.2 page.

Often times performance goals for a given WebLogic Portal application are not met because of either slow response time, not enough concurrent users are running, or the application's throughput is too low. The first question that has to be asked in this situation is: What hardware is running the Portal? This is the single most important factor when determining how well the system will scale. During BEA's internal performance testing, WebLogic Portal was CPU bound, so the performance of the system will depend on how fast each CPU is and how many total CPUs there are.

BEA's internal performance testing indicated a direct relationship between the performance of the system and the overall clock-speed of the CPU(s). By adding more CPUs, or faster CPUs the capacity of the system will increase. Additionally, by clustering machines WebLogic Portal will gain additional scalability due to the addition of CPUs to the overall application deployment. Newer processor technology is also a big factor in determining how a system will perform. For instance, in the results section there is a series of data on Sun's UltraSPARC IIIi processors and although the machines have 4 CPUs their performance is not nearly as good as the results on Intel Xeon processors.

Recommendation

Get the fastest CPUs possible and grow the size of the cluster as needed.

Clustered Configurations

Is the WebLogic Portal Server deployment configured to support clusters? Clusters provide session protection and fail over via state replication in addition to spreading out the load across several systems. Customers using clustering should not see any noticeable performance degradation unless their application stores large amounts of data in the session and that session is replicated across the cluster.

If you are using a web server to forward requests to a WebLogic Server cluster, sometimes the bottleneck can be the web server. This can happen when using the supplied HttpClusterServlet and a proxy server, or one of the supported plug-ins. If the response time does not improve after adding servers to the cluster and the web server machine shows a high CPU utilization, consider clustering the web server or running the web server on more powerful hardware. The web server should be largely I/O bound (including disk utilization and network utilization) rather than CPU bound.

Recommendation

Based on capacity tests with tuned applications, WebLogic Portal is typically CPU-bound. When deciding how much hardware to buy for a production environment, the speed of the processor(s) should be the top priority.

In most cases, WebLogic Server clusters scale best when deployed with one WebLogic Server instance for every two CPUs. However, as with all capacity planning, you should test the actual deployment with your target portal applications to determine the optimal number and distribution of server instances.

Simulated Workload

When trying to determine the performance requirements of your system you will need to take into account the expected workload on the application. For example, a typical banking application experiences heavy traffic (a high number of concurrent sessions) during the “peak hours” of 9 AM and 5 PM. So when doing capacity estimates it is best to test with workloads that will closely mimic the anticipated workload.

Several workload factors can influence the overall performance of the system and depending on how these factors are tested, very different results will be produced. The first is the anticipated “think-time” of the users on the system. Think-time is defined as the pause between requests by a user who is active on the system. For example, if a user clicks to see their bank account balance, they may not click again for 30 seconds, thus the think-time is 30 seconds. This think-time should be averaged across all users (because “expert” users will have shorter think-times and “novice” users will have much longer times) and then that value should be used to test the system. Decreasing the think-time will put a higher load on the system and thus require additional hardware resources.

When testing the system the rate at which users are added to the system also can have a dramatic impact on the performance characteristics of the system. For example, if all of the users are added to the system at once, a “wave” effect will occur where the response times will be very high during the initial steps and improve dramatically as users pause, then increase rapidly as users continue to navigate through the system. Adding users in a staggered fashion will prevent this from happening and provide more consistent performance form the system. Putting some degree of randomization in the think-time will also help to decrease the “wavy” behavior and produce more consistent results.

Recommendation

When testing the system to determine capacity requirements, make sure that the workload of the simulated users accurately reflects what the system would experience in the production environment. Pay close attention to excessive simulated workload that is put simultaneously on the system.

Concurrent Sessions

Determine the maximum number of concurrent user sessions for your WebLogic Portal. To handle more users, you will need to have adequate CPU capacity and RAM for scalability. For most supported configurations 1GB of RAM is the minimum configuration and 2GB is recommended in production for each WebLogic Portal instance.

Next, research the maximum number of clients that will make requests at the same time and how frequently each client will be making a request. The number of user interactions per second with WebLogic Portal represents the total number of interactions that should be handled per second by a given Portal deployment.

Consider also the maximum number of transactions in a given period to handle spikes in demand. Ensure that there is enough excess capacity on the system to handle these spikes. If the demand is close to the max capacity for the system then additional hardware should be added to increase the overall system performance and capacity. For capacity information about concurrent users see “Concurrent User Results” on page 17.

Tuning Your WebLogic Portal/Server

Is the WebLogic Portal well-tuned? A WebLogic Server should be tuned using the available tuning guide.

Recommendation

For more information about tuning WebLogic Portal/Server, see

- WebLogic Server Performance and Tuning
- Top Tuning Recommendations for WebLogic Server

Application Design

How well-designed is the application? Badly designed or non-optimized user applications can drastically slow down the performance of a given configuration. The best course is to assume that every application that is developed for WebLogic Portal will have features that will add overhead and will thus not perform as well as benchmark applications. As a precaution, you should take into account these features of the application and add additional capacity to your system.

It is important to note that the size of the portal (based on the taxonomy of the portal which is calculated by adding up the number of distinct books, pages, and portlets) may have a significant impact on the performance and capacity of the system. As the portal size increases so does the control tree that must be rendered and thus the system will not perform as well as a smaller portal.

The use of multi-level menus negates much of the benefits of the Portal Control Tree Optimizations because the tree must be navigated in order to build the menu structure. This is fine with smaller portals, but for larger portals this will have a significant impact on the performance and scalability of the system and will thus require more hardware resources in the deployment.

Recommendation

Breaking large portals into several smaller Desktops is recommended to optimize the performance of the system. Additionally, profiling with either a “heavy-weight” profiler or a run-time “light-weight” profiler is strongly recommended to find non-optimized areas of code in your application. The use of multi-level menus is discouraged for large portals.

For more information about designing portals, see the following books:

- Designing Portals for Optimal Performance

SSL Connections and Performance

Secure sockets layer (SSL) is a standard for secure Internet communications. WebLogic Server security services support X.509 digital certificates and access control lists (ACLs) to authenticate participants and manage access to network services. For example, SSL can protect JSP pages listing employee salaries, blocking access to confidential information.

SSL involves intensive computing operations. When supporting the cryptography operations in the SSL protocol, WebLogic Server cannot handle as many simultaneous connections.

You should note the number of SSL connections required out of the total number of clients required. Typically, for every SSL connection that the server can handle, it can handle three non-SSL connections. SSL reduces the capacity of the server by about 33-50% depending upon the strength of encryption used in the SSL connections. Also, the amount of overhead SSL imposes is related to how many client interactions have SSL enabled.

Recommendation

Implement SSL using hardware accelerators or disable SSL if it is not required by the application.

WebLogic Server Process Load

What is running on the machine in addition to a WebLogic Portal? The machine where a WebLogic Portal is running may be processing much more than presentation and business logic. For example, it could be running a web server or maintaining a remote information feed, such as a stock information feed from a quote service however this is not recommended.

Consider how much of your WebLogic Portal machine's processing power is consumed by processes unrelated to WebLogic Portal. In the case in which the WebLogic Portal (or the machine on which it resides) is doing substantial additional work, you need to determine how much processing power will be drained by other processes.

BEA recommends that the average CPU utilization on the WebLogic Portal server when executing benchmark tests be in the range of 85 to 95% as a cumulative statistic for that machine. For example, if the machine has multiple processors then the average for both processors should be between the above percentages. This allows the machine to operate at near peak capacity, but also allow for other system processes to run and not drive the CPU to 100%. During production additional CPU overhead should be given to the system to accommodate spikes in traffic so that SLAs around response times are maintained.

Additionally, if any third party applications, services, or processes are deployed in addition to WebLogic Portal, BEA recommends deploying those applications, services, or processes on separate hardware machines.

When dealing with a clustered WebLogic Portal deployment a load balancing solution must be considered. With load balancing in a cluster, the user sessions across the nodes should be about even. If the distribution is not even then that points to a problem with either the WebLogic Portal configuration or the load balancer configuration.

Recommendation

If a cluster of servers is required to meet the capacity demands of the system then a load balancer should be implemented to distribute load across the machines.

All third party applications and services should be off-loaded onto separate hardware.

Database Server Capacity

Is the database a bottleneck? Are there additional user storage requirements? Many installations find that their database server runs out of capacity much sooner that the WebLogic Portal does. You must plan for a database that is sufficiently robust to handle the application. Typically, a good application will require a database that is three to four times more powerful than the application server hardware. It is good practice to use a separate machine for your database server.

Generally, you can tell if your database is the bottleneck if you are unable to maintain a high CPU utilization for WebLogic Portal CPU. This is a good indication that your WebLogic Portal is spending much of its time idle and waiting for the database to return.

Some database vendors are beginning to provide capacity planning information for application servers. Frequently this is a response to the 3-tier model for applications. An application might require user storage for operations that do not interact with a database. For example, in a secure system disk and memory are required to store security information for each user. You should calculate the size required to store one user's information, and multiply by the maximum number of expected users.

There are additional ways to prevent the database from being the bottleneck in the system and one of those ways is by implementing caching at the database layer. WebLogic Portal uses many different caches to avoid hitting the database. If during performance testing the database is determined to be a bottleneck then it might be useful to tune the WebLogic Portal caches to take some of the load off the database.

Recommendation

See the WebLogic Portal Database Administration Guide: Performance Considerations and Sizing Considerations documentation for sizing and other performance related considerations.

Review the WebLogic Portal Cache Reference for more information about database caches.

Network Load

Is the bandwidth sufficient? Network performance is affected when the supply of resources is unable to keep up with the demand. WebLogic Server requires a large enough bandwidth to handle all of the connections from clients it is to handle. If you are handling only HTTP clients, expect a similar bandwidth requirement as a web server serving static pages.

In a cluster by default, in-memory replication of session information shares the same network as the HTTP clients. An alternative to the standard network topology would be to change the physical network with a different channel for internal cluster communication and a second channel for external traffic. See Configuring Network Resources for details. Although the WebLogic Portal framework does not create large amounts of session data it is possible for a custom application to add significant overhead in this area. Additionally, a high load of concurrent users with frequent requests will also lead to network saturation. Consider whether your application and business needs require the replication of session information. Finally, the combination of lots of concurrent users and frequent requests to the server should be estimated to determine if the network can handle the anticipated load.

To determine if you have enough bandwidth in a given deployment, you should look at the network tools provided by your network operating system vendor. There are plenty of free and commercial tools available including build-in applications for Windows and Solaris to help measure this. Additionally, most hardware load balancing solutions provide network statistics. If only one load balancer is used it too may become a bottleneck on the system if the load is very high.

Recommendation

BEA recommends running a gigabit LAN and implementing one or more server load balancers to optimize network traffic.

Selecting Your JVM

What JVM will be used? What parameters will be used? How much heap is required to get the best performance out of the application? Different applications may perform better on one JVM or another. WebLogic Portal supports BEA's JRockit and Sun's HotSpot JVMs. In general, BEA's JRockit JVM performed better during "Benchmark" tests on Intel processors with Linux as the OS, however HotSpot performed slightly better as the cluster size increased during "Concurrent User" tests.

The JVM Parameters can have a dramatic impact on the performance of the system. Please see the BEA JRockit's Reference Manual for a list of all of the parameters and where they may be used.

The size of the heap will also impact the performance of the system. Larger applications may need larger heap sizes. Additionally, a high number of concurrent users will require a larger heap size to prevent the system from running out of memory.

In all cases with JRockit it is recommended that `-Xgc:parallel` be used and with HotSpot `-XX:MaxPermSize` with a minimum of 128m be used. Depending on your application the memory requirements may be quite high. In all cases a set of benchmark tests should be run with the different settings to determine what is best for your application.

Performance Results

There are two types of performance test results in the following sections; one test to assess throughput and another to determine the maximum number of concurrent users supported by the system. The differences between these tests are numerous and thus comparing a data-point from one type of test to another is not recommended.

The first set of data is referred to as "Benchmark Results." This set of tests were run to determine a baseline for the throughput of system measured in pages returned per second. The goal of these tests was to determine the maximum throughput of the system in various configurations where the portal size, portlet type, and JVM were varied.

The second set of data is referred to as "Concurrent User Results" because it is more closely related to the sort of tests run in production-like systems. The goal of these tests was to determine the maximum number of concurrent users (actively clicking through the Portal) for a given response time (often referred to as a Service Level Agreement.)

Each test was driven by a script using LoadRunner that allows each user to log-in once and then click through the pages (for all but the Very Small portal, there were 50 page/book clicks) and then repeat at the first page when the last page is reached. The very small portal has 8 pages, so there were 8 clicks. This continued until the duration of the test was complete.

Test Application

The test application is deployed to the cluster as an EAR that contained `.portal` and `.portlet` files. Form-based authentication was used for each Portal so that each user was registered. The portals themselves varied in size and portlet type. Each portal tested includes one portlet type including JSP, PageFlow for "Concurrent User" tests, and JSP, PageFlow, and Struts for "Benchmark" tests and the portlets used are considered simple portlets such as "Hello World"-type portlets. Tree optimization was enabled for all of the portals. No entitlements or user customizations were enabled. Session replication using the flag "replicated_if_clustered" was configured for all tests. Because all of the users were required to log-in and then did not log-out a session was maintained for each user for the duration of the test.

Test Portals Used

The portal sizes vary with the following parameters:

Table 2 Tested WebLogic Portal Sizes

Portal Size	Number of Books	Number of Pages	Number of Portlets
Very Small	1	8	64
Small	5	50	500
Medium	10	100	1000
Large	20	200	2000
Very Large	40	400	4000

With the exception of the Very Small portal (which has 8 portlets per page) each portal has 10 portlets per page.

Benchmark Results

"Benchmark" tests are designed to show the maximum throughput of the system under different conditions. We varied over the type of portlets in the portal and the size of the portal as well as the JVM. For each configuration the goal is to saturate the server to achieve maximum throughput. The WebLogic Portal servers reached between 85 and 95 percent CPU utilization which is the optimal range for maximum throughput.

To achieve maximum throughput, zero seconds of "think-time" was used, which is to say that the time between a response from the server and the subsequent request was 0 seconds. With this type of workload it is very easy to saturate the server and achieve maximum throughput in a short period of time with relatively few users.

For the Benchmark tests a ratio of 10 virtual users (VUsers in LoadRunner) were used per CPU. The Benchmarks were run on two hardware configurations, HP Linux and Sun Solaris. Since all of the Linux machines tested were configured with two CPUs, for each node in the WebLogic Portal cluster, 20 virtual users were used per machine. The Sun Solaris machines tested had 4 CPUs and thus 40 virtual users were used per machine.These users were "ramped-up" (added to the system) over 20 minutes followed by a steady-state (where no additional users were added but the existing users continued to access the system) that lasted an additional 10 minutes.

Note: The baseline numbers produced by the Benchmarks used in this study should not be used to compare WebLogic Portal with other portals or hardware running similar Benchmarks. The Benchmark methodology and tuning used in this study are unique.

This section includes results from the following configurations:

- HP Linux Hardware and Server Configurations
- Sun Solaris Hardware and Server Configurations

HP Linux Hardware and Server Configurations

The HP Linux tests varied over several different cluster configurations in which there were one, two, four, and eight physical machines in a cluster. Each machine had one running managed server, which translates into one portal and one JVM on each physical machine. Each server has two CPUs and the data is presented in the table by CPU count.

- Administration Server: HP ProLiant DL360 G4 -- Dual 3.6 GHz Xeon, 4 GB RAM, 15K RPM SCSI Disks, HyperThreading enabled, RedHat Enterprise Linux AS 3.0 Update 6, Gigabit NIC
- Managed Servers: HP ProLiant DL360 G4 -- Dual 3.6 GHz Xeon, 4 GB RAM, 15K RPM SCSI Disks, HyperThreading enabled, RedHat Enterprise Linux AS 3.0 Update 6, Gigabit NIC
- Database Server: HP ProLiant DL380 G4 -- Dual 3.4 GHz Xeon, 4 GB RAM,15K RPM SCSI Disks, HyperThreading enabled, Windows 2003 Server Enterprise Edition SP1, Oracle 9.2.0.6, Gigabit NIC
- Load Balancer: F5 Networks Big-IP 1500

- LoadRunner Controller: HP ProLiant DL320 G3 -- 3.6 GHz Pentium 4, 2 GB RAM, 15K RPM SCSI Disk, HyperThreading enabled, Windows 2003 Server Enterprise Edition SP1, LoadRunner 7.8, Gigabit NIC
- BEA JRockit JVM with `-Xms1536m -Xmx1536m -Xgc:parallel -XXaggressive -XXlargeObjectLimit:16k -XXtlasize:256k` setting.
- Sun HotSpot JVM with `-server -Xms1536m -Xmx1536m -XX:MaxPermSize=128m` setting.

HP Linux Results

The servers were set to auto-tune which has been a new feature since WebLogic Server 9.0. The JDBC connection pools were set to start at 5 connections with the ability to grow to 25. These tests were run with 0 seconds of think time so that the servers would become saturated quickly.

Table 3 JSP - JRockit JVM - Throughput in Pages Per Second

Portal Size	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Very Small	380	626	1212	2400
Small	294	482	946	1853
Medium	281	466	893	1791
Large	270	449	870	1726
Very Large	243	412	791	1555

Table 4 JSP - HotSpot JVM - Throughput in Pages Per Second

Portal Size	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Very Small	327	578	1148	2262
Small	252	441	874	1739
Medium	244	431	839	1730
Large	235	416	828	1659
Very Large	223	386	775	1549

Table 5 PageFlow - JRockit JVM - Throughput in Pages Per Second

Portal Size	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Very Small	333	487	954	1891
Small	235	332	655	1266
Medium	234	330	642	1293
Large	224	318	626	1228
Very Large	204	289	575	996

Table 6 PageFlow - HotSpot JVM - Throughput in Pages Per Second

Portal Size	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Very Small	259	370	776	1533
Small	185	281	441	1050
Medium	184	270	440	979
Large	185	251	394	826
Very Large	166	214	390	825

Table 7 Struts - JRockit JVM - Throughput in Pages Per Second

Portal Size	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Very Small	327	527	1019	2016
Small	237	388	778	1512
Medium	229	385	742	1464
Large	216	366	717	1426
Very Large	204	343	654	1202

Table 8 Struts - HotSpot JVM - Throughput in Pages Per Second

Portal Size	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Very Small	286	487	986	1961
Small	206	369	723	1449
Medium	201	358	703	1429
Large	196	347	702	1380
Very Large	180	321	652	1305

Sun Solaris Hardware and Server Configurations

The Sun Solaris tests used four and eight CPU configurations in which there were one and two physical machines. Each machine had two running managed servers, which translates into two portals and two JVMs on each physical machine, for a total of two and four managed servers in the cluster. Each server has four CPUs and the data is presented in the table by CPU count.

- Administration Server: Sun Fire v240, 2 x 1.02GHz, 4GB RAM, 10K RPM SCSI Disks, Sun Solaris 10
- Managed Servers: Sun Fire v440, 4 x 1.02GHz, 8GB RAM, 10K RPM SCSI Disks, Sun Solaris 10, Gigabit NIC
- Database Server: HP ProLiant DL380 G4 -- Dual 3.4 GHz Xeon, 4 GB RAM, 15K PRM SCSI Disks, HyperThreading enabled, Windows 2003 Server Enterprise Edition SP1, Oracle 9.2.0.6, Gigabit NIC
- Load Balancer: F5 Networks Big-IP 1500
- LoadRunner Controller: HP ProLiant DL320 G3 -- 3.6 GHz Pentium 4, 2 GB RAM, 15K RPM SCSI Disk, HyperThreading enabled, Windows 2003 Server Enterprise Edition SP1, LoadRunner 7.8, Gigabit NIC
- JVM: Hotspot with `-server -Xms1536m -Xmx1536m -XX:MaxPermSize=128m` setting.

Sun Solaris Results

The servers were set to auto-tune which has been a new feature since WebLogic Server 9.0. The JDBC connection pools were set to start at 5 connections with the ability to grow to 25. These tests were run with 0 seconds of think time so that the servers would become saturated quickly.

Table 9 JSP - HotSpot JVM - Throughput in Pages Per Second

Portal Size	4 CPUs	8 CPUs
Very Small	164	340
Small	127	264
Medium	123	257
Large	121	248
Very Large	114	230

Table 10 PageFlow - HotSpot JVM - Throughput in Pages Per Second

Portal Size	4 CPUs	8 CPUs
Very Small	123	242
Small	90	176
Medium	89	173
Large	85	168
Very Large	80	150

Table 11 Struts - HotSpot JVM - Throughput in Pages Per Second

Portal Size	4 CPUs	8 CPUs
Very Small	147	301
Small	112	224
Medium	109	220
Large	105	215
Very Large	99	200

Concurrent User Results

This set of performance test results are also known as “Capacity Planning” results because they are best suited for determining what the overall capacity of the system is by measuring how many concurrent users can run on a given set of hardware. These tests are designed to mimic real-world user loads and thus show a more accurate representation of the system than the standard “Benchmark” tests.

Based on feedback from our customers the most common SLAs are 2 second and 5 second response times. Our goal was to determine how many users WebLogic Portal could support across various configurations with those SLAs. If your given SLA is higher then the number of supported users will also be higher although estimating that number would be difficult to do without actually running additional tests.

For Capacity Planning tests the think-time is also meant to mimic real-world production systems being accessed by so-called “expert users.” This should be considered a very high workload for the system and in many other configurations the request times by the end users will not be “expert” like. The think-time for these tests was randomized at 5 seconds +/- 25% (between 3.75 and 6.25 seconds.) Whereas a non-export like system might state that the think-time is closer to 30 seconds averaged across all users. The think-time for the system has a dramatic impact on the overall capacity of the Portal. A higher think-time will allow many more users on the system. You can see in the “Benchmark” configuration there was only 10 users per CPU required to saturate the system, but with think-time it could take hundreds if not thousands of users per CPU to have the same impact.

The workload for Capacity Planning tests is vastly different than that of the above “Benchmark” tests. Because the number of users required to meet the minimum SLAs is much higher (due to think-time) the duration of the tests must be extended. The number of users for each configuration was ramped-up over the course of two hours and for each configuration a different number of users was added at a constant rate over one minute. We chose two hours because the system responded better and thus supported more users than with shorter ramp-up schedules. A high number of users was added to the system at a constant rate over 60 seconds until they were all running at roughly the two hour mark.

This test established how many concurrent users the test portal could support with a given response time. Goal response times of two seconds and five seconds were used. The number of concurrent users listed in the table represent the maximum number of running concurrent users under 2 or 5 seconds. This test used the HP Linux configuration, see “HP Linux Hardware and Server Configurations” on page 12. Each server has two CPUs and the data is presented in the table by CPU count.

This section reports the following results:

- JSP Portlet Results
- PageFlow Portlet Results

JSP Portlet Results

Table 12 JSP - JRockit JVM - Two-Second Response Time

Portal Size	2 CPUs	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	1770	3024	5376	8750	11189
Small	1285	2278	4355	6600	8710
Medium	1411	2336	4347	6440	8625
Large	1190	2176	4180	6132	8502
Very Large	1122	1968	3910	5625	7727

Table 13 JSP - JRockit JVM - Five-Second Response Time

Portal Size	2 CPUs	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	2550	4326	7980	12125	16199
Small	1972	3162	6148	9000	12060
Medium	1989	3168	6048	9108	12125
Large	1836	3104	5665	8652	11881
Very Large	1649	2720	5038	8100	10386

Table 14 JSP - HotSpot JVM - Two-Second Response Time

Portal Size	2 CPUs	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	2037	2856	6048	8875	12024
Small	1564	2550	4711	7200	9648
Medium	1462	2368	4672	6900	9125
Large	1462	2420	4372	6888	8829
Very Large	1292	1920	4015	5850	7848

Table 15 JSP - HotSpot JVM - Five-Second Response Time

Portal Size	2 CPUs	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	2497	3948	8400	12500	16967
Small	1972	3332	6365	10100	13802
Medium	1904	3225	6615	9752	13500
Large	1955	3104	5830	9338	12140
Very Large	1802	2976	5680	8100	11445

PageFlow Portlet Results

Table 16 PageFlow - JRockit JVM - Two-Second Response Time

Portal Size	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	1500	3050	4650	6052
Small	715	1450	1806	3418
Medium	666	1032	1940	3455
Large	567	1347	2068	3266
Very Large	627	1098	2131	3089

Table 17 PageFlow - JRockit JVM - Five-Second Response Time

Portal Size	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	1925	3949	5900	7807
Small	742	1456	2200	3553
Medium	720	1170	2111	3392
Large	644	1403	2140	3270
Very Large	783	1229	2141	3060

Table 18 PageFlow - HotSpot JVM - Two-Second Response Time

Portal Size	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	1525	3375	5000	6566
Small	715	1484	2268	3127
Medium	688	1470	2163	2950
Large	729	1280	2040	2944
Very Large	657	1305	2057	3120

Table 19 PageFlow - HotSpot JVM - Five-Second Response Time

Portal Size	4 CPUs	8 CPUs	12 CPUs	16 CPUs
Very Small	2150	4625	6868	9179
Small	792	1687	2464	3345
Medium	792	1590	2340	3304
Large	798	1477	2363	3290
Very Large	747	1469	2271	3158

Other Resources

Remember that WebLogic Portal uses many components from WebLogic Platform. See the following documentation for more information about tuning WebLogic Portal.

- Designing Portals for Optimal Performance
- WebLogic Server Performance and Tuning Guide
- WebLogic Server Capacity Planning Guide
- Tuning WebLogic JRockit JVM
- BEA's dev2dev Website