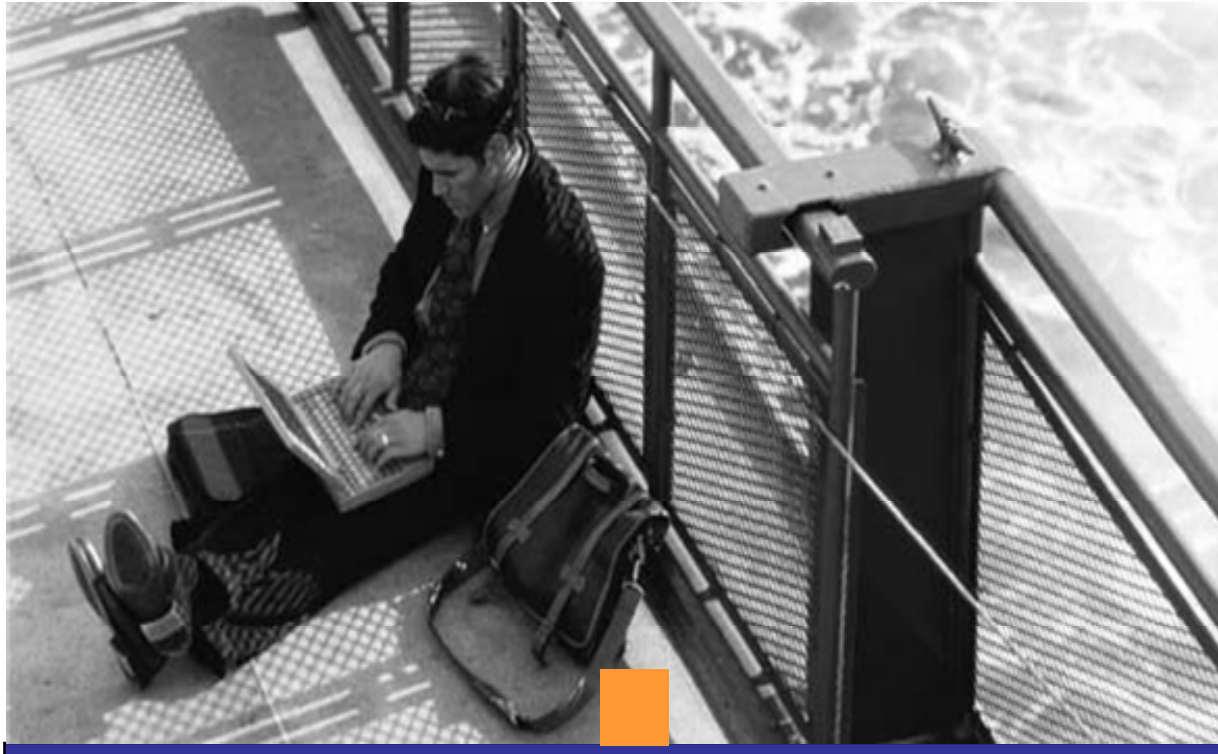


SAP NetWeaver

SAP® ENTERPRISE PORTAL

Scalability Study - Linux

ABOUT SAP® ENTERPRISE PORTAL

SAP® Enterprise Portal is a key component of the SAP NetWeaver™ platform. SAP Enterprise Portal is the industry's most comprehensive portal solution, providing a complete portal infrastructure along with bundled knowledge management and collaboration capabilities. It provides people-centric integration of all types of enterprise information, including SAP and non-SAP applications, structured and unstructured data, and Web content. It is based on open standards, such as Web services and supports Java 2 Platform, Enterprise Edition (J2EE) and Microsoft .NET technology.

Business content delivered with SAP Enterprise Portal speeds portal implementation and reduces the cost of integrating existing IT systems.

SAP Enterprise Portal provides employees, supply chain partners, customers, and other user communities with immediate, secure, and role-based access to key information and applications across the extended enterprise. Because information and applications are unified through the portal, users can identify and address business issues faster, more effectively, and at lower cost – creating measurable benefits and strategic advantages.

ABOUT THIS STUDY

A major strength of SAP Enterprise Portal lies in its ability to handle large numbers of users in enterprise-scale implementations.

This study is comprised of tests to check the scalability and stability of the portal on Linux RedHat.

The aim of this study is to demonstrate the linear scalability capabilities of SAP Enterprise Portal.

Performance tests of the portal on a given hardware platform is based on the following assumptions: availability of specific content, and user behavior to obtain information about CPU usage, server response times, and cluster scalability.

TEST CONFIGURATION

CLUSTER CONFIGURATION

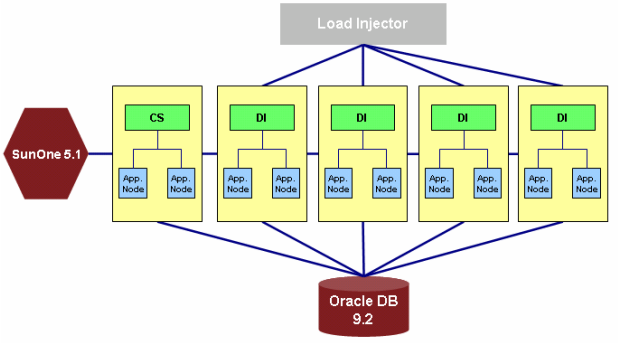
Hardware Configuration

System:	HP ProLiant BL20p G3 Blade servers with dual 3.2 Intel® Xeon™ processors and ,4GB RAM. System landscape components are installed on the same platform. For more information please see : http://h18004.www1.hp.com/products/servers/proliant-bl/p-class/20p/specifications-g3.html
Network:	Portal , Database and Directory Server : 1-Gbit LAN Clients : 100-Mbit LAN
Comments	No firewalls or reverse proxies are used. The portal is accessed via HTTP. No load balancer is used. Load is generated by Mercury Interactive Load Runner addressed directly to blade units.

Software Configuration

Operating System:	Linux RedHat EL 3 (IA32)
Database:	Oracle 9.2.0.5 on Microsoft Windows 2003
Application Server:	SAPJ2EE 6.40 SR1 (NW04 SP9)
Portal	SAP Enterprise Portal 6.0 SR1 (NW04 SP9)
Directory Server:	Sun One 5.1 on Microsoft Windows 2003 server , 20000 named users in test group
JDK:	Sun j2SDK 1.4.2_06
Load Generator	Mercury Interactive LoadRunner 8.0

System landscape contains one central system, four dialog instances ,each containing 2 server nodes, one oracle DB and Directory Server (7 blade server altogether), all servers with 4GB RAM.



1 Central system – Linux two server node machine - will not be tested under load

4 Dialog instances - Linux two server nodes machine

1 Oracle DB - Windows2003 server

1 Directory Server - Sun One 5.1 - Windows2003 server

SAP Web Application Server (JAVA) Configuration

Each server node per dialog instance is configured with the following JVM settings:
See SAP Note **723909** for information about the JVM settings.

- *-XX:SoftRefLRUPolicyMSPerMB=1*
- *-verbose:gc*
- *-XX:+UseParNewGC*
- *-XX:+PrintGCDetails*
- *-XX:+PrintGCTimeStamps*
- *-XX:MaxNewSize=170M*
- *-XX:NewSize=170M*
- *-XX:+DisableExplicitGC*
- *-XX:MaxPermSize=192M*
- *-XX:PermSize=192M*
- *-XX:SurvivorRatio=2*
- *-XX:TargetSurvivorRatio=90*
- *-Xmx1024M*
- *-Xms1024M*

The following are the configurations to optimize performance:

- The log level for all log files is configured to "error" level. The default is "ALL"
- The DSR service is disabled using the offline configuration editor.

It is very important to optimize the log, and trace levels, as well as the DSR service, as these influence the overall system performance.

Additional optimization (which wasn't applied) that can significantly improve system performance :

- SAPJ2EE compression
- SAPJ2EE content expiration settings
- JavaScript and CSS optimization.
- Keep Alive configuration

ORACLE Database Configuration

It is very important to optimize the Oracle database. Before running the real scenarios, preliminary test executed in oracle database in order to check if the configuration is optimal for running 2000 concurrent users. The preliminary test performed showed that, there is the need to perform additional configuration to optimize performance of Oracle and to prevent other bottlenecks across the landscape before running the real scenario.

- *Tuned for 2000 users*
- *Shared Pool 500MB*
- *Buffer Cache 160MB*
- *PGA 300MB*
- *Processes 600*
- *Sessions 650*
- *dml_locks 10000*
- *open_cursors 10000*

LINUX CONFIGURATION

The operating system is configured as follows:

- Shared memory size increased.
  /dev/shm might be too small so "defaults" value in /etc/fstab is replaced with "size=6G".
- Some NFS entries in /etc/fstab do not have the correct format, they are missing the **fs_mntops, fs_freq** and f**s_passno** columns, see "man 5 fstab" for details.
- All NFS entries should have the following options instead of defaults:
  hard,intr,bg,rsize=8192,wsize=8192"
  After setting the correct values, unmount and mount the directories again.
- Remove the host name alias.
  /etc/hosts has the host name as an alias for the IP address "127.0.0.1" as well as the real IP address.
- Change the comment character in **etc/sysctl.conf** to "*" from the character "#". The comment character "#" caused errors.
- Run the kernel in -27.Elsmp. The kernel package is at: **/vol/linuxlab/redhat/testing/3AS/U4/x86/**
- Disable the **cron** daemon so that the test results are not skewed. For example, **updatedb** catalogs the whole file system during the test.
  To stop it: **"/sbin/service crond stop"**
  To make it permanent "**/sbin/chkconfig crond off**".
  You can enable both settings again, "**/sbin/service crond start**" and "**/sbin/chkconfig crond on**" respectively after the test so that the logs get rotated.
- **syslogd** is configured to write to the log files asynchronously.
  Append a hyphen ("-") to the names of the log files in the directory **/etc/syslog.conf,** if a file does not have one. Then restart the service ("**/sbin/service syslog restart**")

GLOSSARY

- **Response time** – is a key performance indicator, the time it takes a system to react to a given input. The response time includes the transmission time, the processing time, the time for searching records and the transmission time back to the originator**.**
- **Scalability** – is the capability to increase resources to yield a linear (ideally) increases in service capacity. The key characteristic of a scalable application is that additional load only requires additional resources rather than extensive modification of the application itself.
- **Scale out scenario** – testing scalability of hardware resources while adding more dialog instances on additional machines
- **Scale in scenario** – testing scalability of hardware resources while adding more dialog instances on the same machine
- **Think Time** – The time between a page being returned and the next user page request.
- **Transaction** – A group of processing steps that are treated as a single activity to perform a desired result. The steps in transaction can be of various types (Database query, client-server events and etc.)
- **GC –** Garbage Collection is a thread that runs to reclaim the memory by destroying the objects that cannot be referenced anymore. Garbage collection is performed by a garbage collector which recycles memory in young generation.
- **FGC –** Full Garbage collection, Garbage collection is performed by a garbage collector who recycles memory in both young and old generations. FGC is usually much longer than GC.

TEST SEQUENCE

The Employee Self-Service Portal Scenario (based on the ESS benchmark package)

The *Business Package for Employee Self-Service* (ESS) is used as the test scenario. It has a number of iViews that allow employees to create, display and change their own data, such as department, name and address.

This scenario focuses on concurrent users with the employee role (workset Employee Self-Service), and their top level navigation behavior in the portal.

From a technical point of view, this scenario enables us to examine the performance of the portal platform (running on the NetWeaver Java stack), while launching business transactions. The backend influence on the performance has been omitted, as there is no connection to a backend system

Interaction Steps

1. Log on to the portal - Welcome Page (includes 3 iViews)
2. Choose Working Time → Record Working Time
3. Choose Leave Request
4. Choose Leave Request Overview
5. Choose Personal Information → Personal Data
6. Choose Address
7. Choose Bank Information
8. Choose Benefits and Payment → Paycheck Inquiry
9. Log off from portal

After the user logs on to the portal, a series of transactions are with specific think time (a pause between transactions) are started. At the end of the test sequence, the user logs off.

Transaction in the tests can be divided into two types :

- Business transactions, consisting of 3 sets of seven different business transactions are launched per loop (steps 2 to 8 * 3). Transactions simulating navigation in ESS workset.
- Login/Logoff transactions are carried out once per loop. The login process is more complicated than ESS business transactions because it consists of user authentication, static and dynamic content delivery from the server to the client, portal navigation, and the Welcome page with 3 iViews.

Directory server contains a pool of 20,000 different users, randomly loaded using the load runner script.

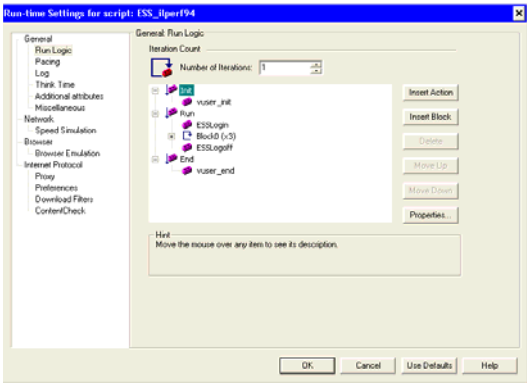
Think time 10 seconds:

Real think time is randomly spread between 5 and 15 seconds, thus one loop contains 2+7*3 transactions and takes between 23*5 and 23*15 seconds (assuming that transaction response times are less than think time).

Think time 60 seconds:

The real think time is randomly spread between 30 and 90 seconds, thus one loop contains 2+7*3 transactions and takes between 23*30 and 23*90 seconds (assuming that transaction response times are less than think time).

Two users are added every 7 seconds during the user ramp-up stage (for instance: it takes 10 minutes to load 170 users).



TESTS OVERVIEW

This document describes the tests performed in the portal using the Employee Self-Service (ESS) scenario based on two different think-times: think time 10 seconds and 60 seconds and the heap size 1024K.

Each test consisted of measurements for the following scenarios: scalability, and stability.

Scalability

The measurements for the scalability scenario simulated the logon activity of users till the CPU usage reached 60-70 percent. At a think time of 10 seconds, 170 users logged on, and at a think time of 60 seconds, 500 users logged on per dialog instance.

At this level of CPU usage, another dialog instance is added to accommodate more users logging into the system. This simulation was repeated until the portal landscape consisted of four dialog instances with eight server nodes. Both tests lasted for about 2-3 hours (till users completed logging on to the fourth instance).

The tests demonstrate system scalability by adding extra dialog instances (each dialog instance has the same number of users).

The scalability scenario contains the following information :

- Cluster Scalability (System Resource vs. test time)
- Consolidation of system resources per dialog instance
- Average CPU utilization by dialog instances
- User scalability on a single dialog instance
- Scalability per dialog instance

Stability

The test measures the overall system behavior overtime, when the average CPU usage is ~60 percent.

Gradual increase in the number of users on all application nodes was executed in parallel. The test lasted for 6 hours.

The stability scenario contains the following information :

- Average response times for transactions
- Transactions behavior (response times) during 6 hours load test.
- JVM memory handling
- Consumption of system resources

Why Measure Both Scalability and Stability?

These two tests have different meaning in the performance world.

Scalability test shows the ability to increase the dialog instances on additional machines to meet the needs of users as the demand for portal resources increases.

Stability tests focuses on how steady is the overall system (response times, VM memory behavior and System resources).

Generally, stability tests last longer (~6 hours in our case) than scalability tests (~2-3 hours in our case), in order to check the behavior of the system over time.

Preliminary Tests

Preliminary test was performed using different number of users and content (internal SAP content: - consisting of different types of iViews, navigation structures, and pages).

The preliminary test was performed in order to evaluate the system setup, and to check for possible bottlenecks.

In addition, the preliminary test helped us to obtain specific settings to configure in order to optimize the performance of the database (DB), the operating system (OS), and the J2EE engine.

Capacity Tests

A preliminary test was carried using a set of different number of users, in order to obtain the suitable number of users that should run on each dialog instance, until the CPU utilization reaches ~60-65 percent.

In our test landscape, and on specific hardware, the tests showed that 170 concurrent users should run on each dialog instance, if the think time 10, and 500 concurrent users should run on each dialog instance, if the think time is 60 seconds.

Load runner scripts were changed accordingly, in order to run the exact number of users on each dialog instance.

Summary

This document describes two tests, each of which consisted of two scenarios. It provides information about the results and their interpretation for the following tests:

1. Think Time 10 seconds, HeapSize 1024K
    1.1. Scalability
    1.2. Stability
2. Think Time 60 seconds, HeapSize 1024K
    1.1. Scalability
    1.2. Stability

The following sections provide the results, and show how linear is the scalability of the portal, and how the system behaves under load.

TEST RESULTS

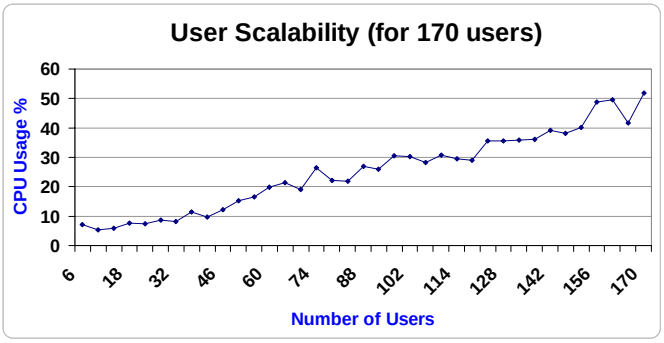
Scenario 1.1 - Scalability Test , ESS Scenario with Think Time 10 seconds, Heap size 1024K

Scalability on a Single Dialog Instances (DI)

The result of this test on a single DI shows the user scalability. With 170 users, the maximum CPU utilization reaches around 60 percent. The graph below shows a single instance example where CPU utilization reaches 56.6 percent.

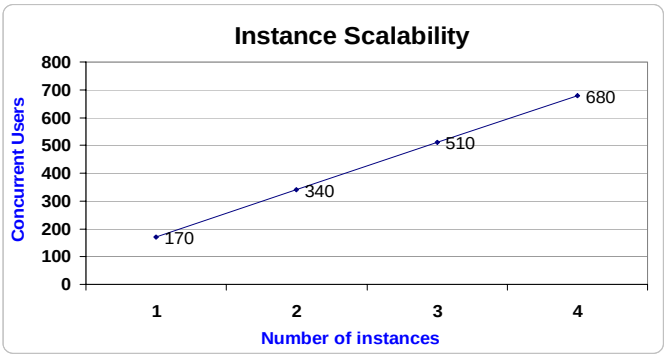
Note that the deeps in the slope is attributed to the fact that there are transactions that are "heavier" in system resource consumption then others (such as Login) and due to the think time variation (as described above) which might cause heavy load at certain point of time.

The curve is relatively flat up to the point of 150 users, with CPU utilization below 40 percent. Then at 156 users the curve becomes steeper with a deep at 166 users.



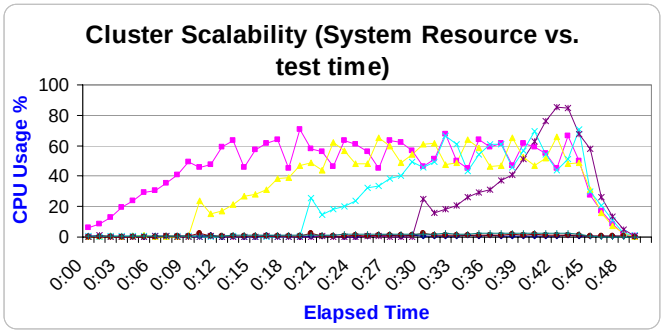
Scalability on a Cluster with 4 Dialog Instances

The scalability on a four (4) DI with regards to number of users is linear with 170 users per DI. This figure should be viewed in conjunction with cluster scalability, DI system resources consolidation and average CPU utilization graphs (right hand side) to see that at 170 users per DI, all DI behave the same over a period of time with CPU utilization in a closely tide band, except for the forth (4) DI which break above the rest after nine (9) Min. of the test. Further, the CPU utilization of the Central instance, DIRECTORY SERVER and Oracle DB remains very low at the bottom, about four (4) percent utilization during the test. Note that the linear scalability can be achieved due to lack of bottlenecks in resources described above.



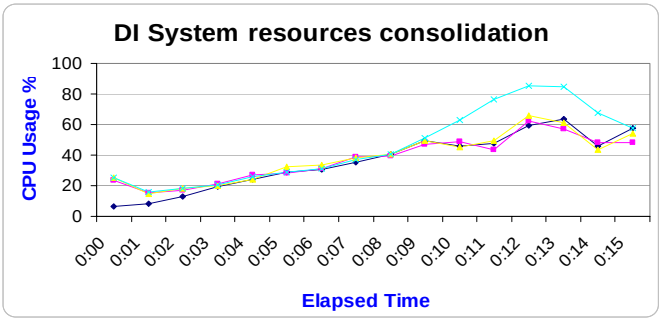
Resources VS. Users

This graph describes the behavior of four (4) dialog instances (DI) with 170 concurrent users on each DI. During this test, each DI was loaded with users, and once it reached 170 users the next DI was launched. The CPU utilization for each DI is between 45 to 60 percent. Note that the forth (4) DI behave a bit different and reaches close to 85 percent utilization for a short period of time . There is no significant impact or degradation in CPU utilization on any DI when a new DI is up and running with 170 users (except for the forth DI). At the bottom of the graph the CPU utilization of the Central Instance, Directory Server and Oracle DB are shown. The level of utilization of the system resources is insignificant – bellow four (4) percent.

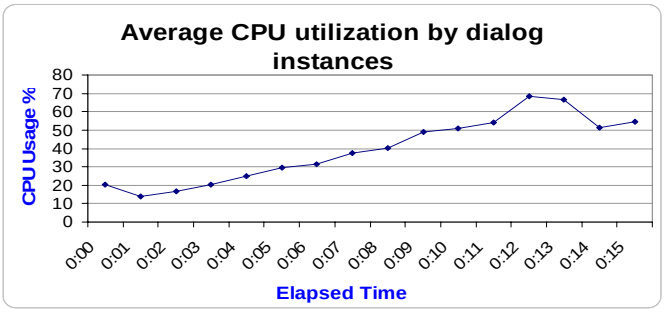


The consolidate graph (from the graph above) of the cluster CPU utilization provides more clear view of the behavior of each DI during the ramp-up. There is a peak up to 85 percent

(fourth dialog instance) for duration of two minutes and then it goes down to a normal pattern.



Finely the average CPU utilization of all dialog instances in the cluster provides the ability to zoom and examine the behavior of the DI scalability. At its peak, the CPU utilization is at 68.2 percent



Scenario 1.2 - Stability Test , ESS Scenario with Think Time 10 seconds, Heap size 1024K

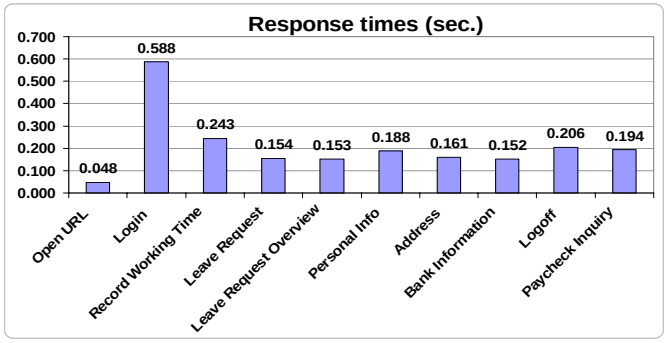
Response Times

The server response time average is differs from transaction to transaction (see the figure below). Transaction in this test can be divided into two types :

- Login to the portal – 0.59 seconds. Response times here are longer than in the other transactions because it is complicated process consisting of user authentication, static and dynamic content delivery from the server, portal navigation and welcome page loading with 3 iViews.
- Portal Navigation (business transactions) - less than 0.3 second. Basic navigation in ESS workset.

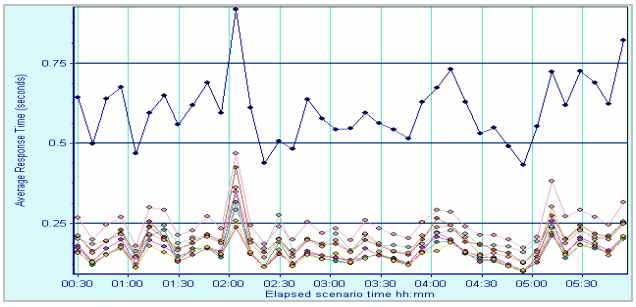
Optimizing the portal by configuring SAPJ2EE compression, content expiration timeout, Keep Alive and JavaScript files can significantly decrease both response times, network traffic and might have impact especially on “login” transaction.

There are several other parameters that influence the marshalling of the data and hence the response time. These include the dependencies on network latency and bandwidth, connection to back end systems and etc. The numbers will differ when taking into consideration all these parameters.



Average transactions response times

The figure below shows response times of transactions versus test time (6 hours not including ramp-up). Server response times remain flat across the entire test. You can realize scaling at some points but response time remains under 0.5 seconds. The spike might happen because of VM memory behavior (GC and FGC occurrences) and other system resources such as database, directory server and etc...

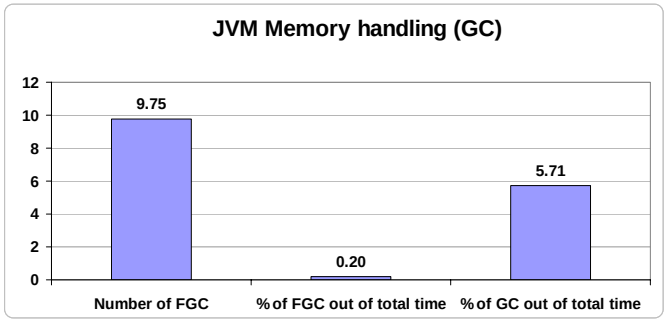


Transactions response times during 6 hours test

System Resources

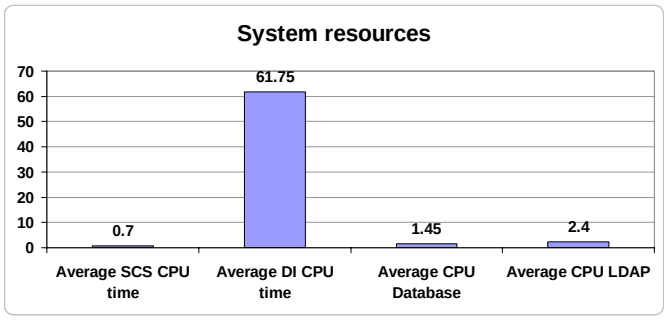
Garbage Collector has major impact on overall application performance and can become a bottleneck; **pauses** (because of FGC) are the times when an application appears unresponsive because garbage collection is going on. The average number of Full GCs per each server node (we have 8 nodes) in this test are 9.75. Average Full GC time is 5.33 seconds while maximum minor GC time is only 0.7 seconds.

Total time for all Full GCs occurred during 6 hours load test is 51.3 seconds in average (0.2 percent of total time). Total GC time is 1439 seconds and this is 5.7percent of total test time (7 hours including users ramp-up).



JVM memory handling

When measuring application performance it is very important to have a broad vision and monitor all components in the system landscape. You can see in figure below that average database, directory server and central instances CPU are not under load and remain very low during 6 hours load test.



CPU utilization in system landscape components

Here are the average CPU utilization results for 4 dialog instances :

Dialog Instance 1 – 61.6 percent

Dialog Instance 2 – 62.4 percent

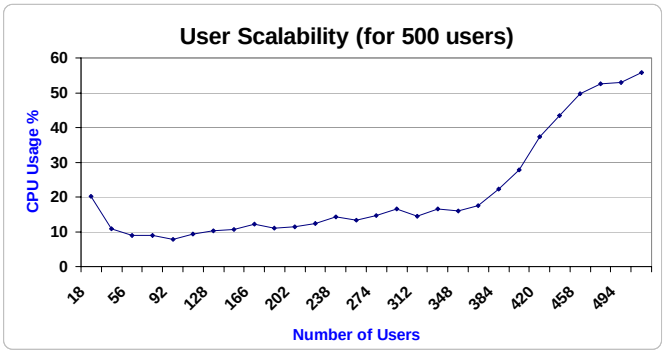
Dialog Instance 3 – 63 percent

Dialog Instance 4 – 60 percent

Scenario 2.1 - Scalability Test, ESS Scenario with Think Time 60 sec. and Heap size 1024K

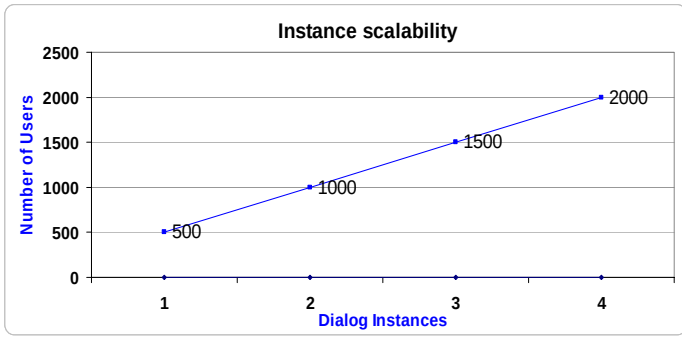
Scalability on a Single Dialog Instances (DI)

The result of this test on a single DI shows that at around 55 percent CPU utilization the number of users is 500. The graph below shows a single instance example where the slope up to 350 users is flat. It becomes steeper from 350 – 500 users, where for each 50 added, the CPU utilization consumed was raised about 45 percent over the previous level (of 300 users), it continues in the same behavior up to 500 users to its peak point.



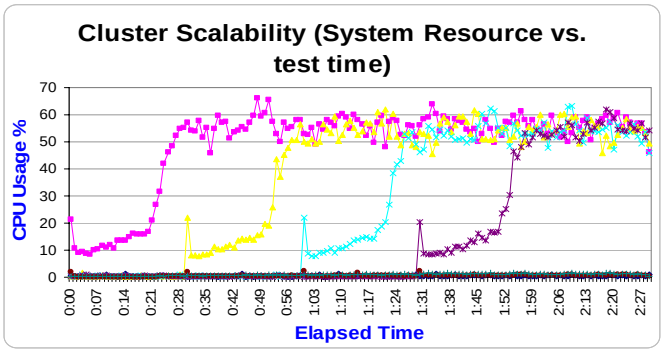
Scalability on a Cluster with 4 Dialog Instances

The scalability on a four (4) DI with regards to number of users is linear with 500 users per DI. The max CPU utilization is at around 60 percent. In this example it shows the max CPU utilization is at 56.9 percent. This figure should be viewed in conjunction with cluster scalability, DI system resources consolidation and average CPU utilization graphs (right hand side) to see that at 500 users per DI, all DI behave the same over a period of time with CPU utilization in a closely tide band. Further, the CPU utilization of the Central instance, Directory Server and Oracle DB remains very low at the bottom, which is below 2.5 percent utilization during the test. Note that the linear scalability can be achieved due to lack of bottlenecks in resources described above.

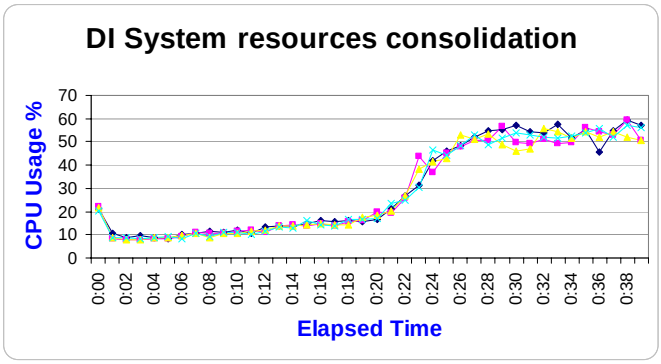


Resources VS. Users

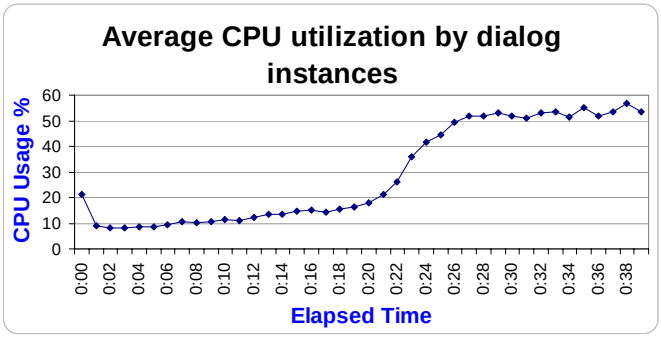
This graph describes the behavior of four (4) dialog instances (DI) with 500 concurrent users on each DI. During this test, each DI was loaded with users, and once it reached 500 users the next DI was launched. The CPU utilization for each DI is between 50 to 60 percent. Note that all DI behave in the same manner. There is no significant impact or degradation in CPU utilization on any DI when a new DI is up and running with 500 users. At the bottom of the graph the CPU utilization of the Central Instance, Directory Server and Oracle DB are shown. The level of utilization of the system resources are insignificant – bellow 2.5 percent for the entire duration of the test.



The consolidate graph (from the graph above) of the cluster CPU utilization provides more clear view of on the behavior of each DI during the ramp-up. All DI have the same pattern of behavior.



Finely the average CPU utilization of a DI in the cluster provides the ability to zoom and examine the behavior of the DI scalability. At its peak, the CPU utilization average is at 56.9 percent.



Scenario 2.2 - Stability Test, ESS Scenario with Think Time 60 sec. and HeapSize 1024K

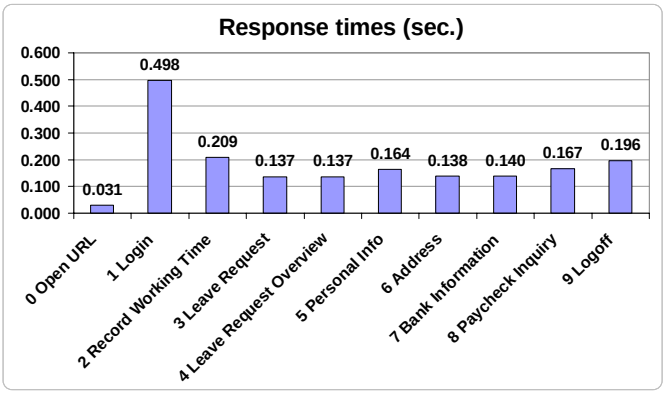
Response Times

The server response time average is differs from transaction to transaction (see the figure below). Transaction in this test can be divided into two types :

- Login to the portal – 0.5 seconds. Response times here are longer than in the other transactions because it is complicated process consisting of user authentication, static and dynamic content delivery from the server, portal navigation and welcome page loading with 3 iViews.
- Portal Navigation (business transactions) - less than 0.2 second. Basic navigation in ESS workset.

Optimizing the portal by configuring SAPJ2EE compression, content expiration timeout, Keep Alive and JavaScript files can significantly decrease both response times, network traffic and might have impact especially on “login” transaction.

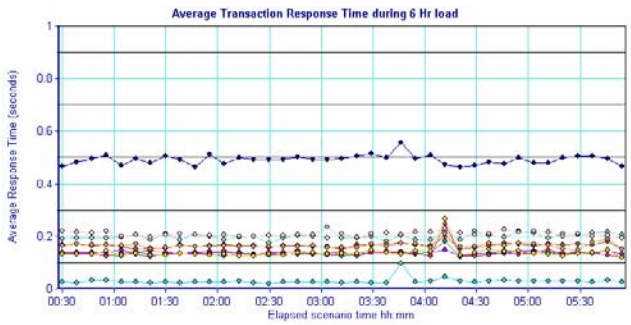
There are several other parameters that influence the marshalling of the data and hence the response time. These include the dependencies on network latency and bandwidth, connection to back end systems and etc. The numbers will differ when taking into consideration all these parameters.



Average transactions response times

The figure below shows response times of transactions versus test time (6 hours not including ramp-up). Server response times remain flat across the entire test. You can realize scaling at some points but response time remains under 0.5 seconds. The spike might happen because of VM memory behavior (GC and

FGC occurrences) and other system resources such as database, directory server and etc...

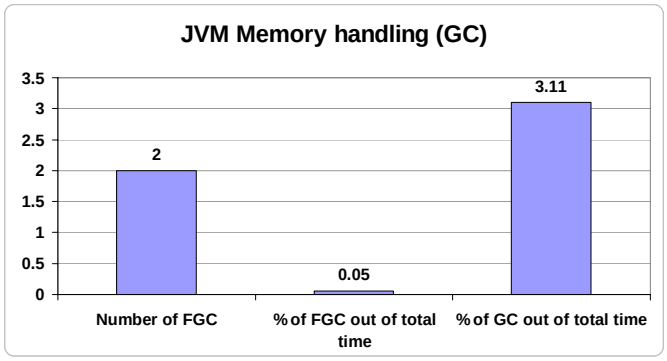


Transactions response times during 6 hours test

System Resources

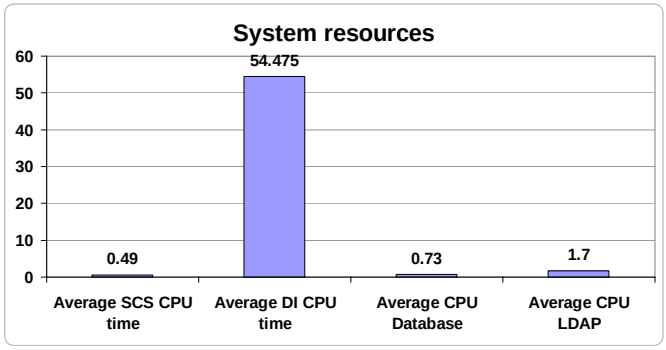
Garbage Collector has major impact on overall application performance and can become a bottleneck; **pauses** (because of FGC) are the times when an application appears unresponsive because garbage collection is going on. The average number of Full GCs per each server node (we have 8 nodes) in this test are only 2. Average Full GC time is 6.8 seconds while maximum minor GC time is only 0.67 seconds.

Total time for all Full GCs occurred during 6 hours load test is 13.6 seconds in average (0.05 percent of total time). Total GC time is 783 seconds and this is 3.11 percent of total test time (7 hours including users ramp-up).



JVM memory handling

When measuring application performance it is very important to have a broad vision and monitor all components in the system landscape. You can see in figure below that average database, directory server and central instances CPU are not under load and remain very low during 6 hours load test.



CPU utilization in system landscape components

Here are the average CPU utilization results for 4 dialog instances :

Dialog Instance 1 – 51.4 percent

Dialog Instance 2 – 55.6 percent

Dialog Instance 3 – 54.5 percent

Dialog Instance 4 – 68.4 percent

INTERPRETATION OF THE RESULTS AND COMMENTS

Linear Scalability

The testing results indicated that adding dialog instances provided a **linear scalability**. System maintained stabled behavior when adding additional instances (CPU utilization was equal on all the instances during all test phases like ramp-up, long load and etc.).

No scalability bottlenecks – in order to support increasing user demand, J2EE instances are easily and dynamically added to the configuration as needed. Resources are optimally utilized by distributing the overall request load among the servers.

Traffic is distributed across multiple SAPJ2EE Instances in order to balance the load placed on the SAPJ2EE engine. This allows the capacity of a Web site to be scaled by simply adding more dialog instances.

What is reasonable think time ?

It can be 3 sec or 3 hours depending on context (for example, a data operator typing a few numbers into a simple form and then save it in the first case and a data analytic getting data from a database and then analyzing them offline in the second case). In our test we have decided to use two think times. One of 10 seconds and the other of 60 seconds. 10 seconds think time described very high user activity and this is unlikely possible scenario. 60 seconds think time represent high user activity on one hand and makes it possible to ramp-up couple of hundreds of users per instance on the other hand. This may seem to be a long time; however one must consider an average for an entire work day.

Think time 10

Based on the measurements provided in the study, the tested 4-server configuration supported 680 concurrent users and CPU utilization reached 68 percent.

Think time 60

Based on the measurements provided in the study, the tested 4-server configuration supported 2000 concurrent users and CPU utilization reached 56.9 percent

Transaction response times are very similar in both tests.

System resources

The table below summarize the differences in system resources usage between two tests:

	Think Time 10	Think Time 60
Number of FGC	9.75	2
Average FGC time	5.33s	6.8s
Max GC time	0.7s	0.67s
Total FGC time	51.3s	13.6s
Total GC time	1439s	783s

The main difference is in the number of FGC occurrences and total GC time. Heavy load executes on the system during the first test, a lot of new object created and this is the reason why the GC and FGC times are greater than in the second test.