Oracle Forms Services Release 6*i* Capacity Planning Guide

An Oracle White Paper
November 2001

ORACLE

Oracle Forms Services Release 6*i*
Capacity Planning Guide

Oracle Forms Services Release 6*i*
Capacity Planning Guide

INTRODUCTION

This whitepaper will help decision-makers, consultants, and system administrators understand the scalability features of Oracle Forms Services Release 6*i* in order to assist with planning the deployment of critical business systems built using Forms 6*i*.

It is important for companies to know in advance the type of environment and hardware commitment required to deploy their systems. It is equally important that there is confidence that the system can grow in a scalable manner, to meet business growth and future demands.

What's in this guide?

This document, firstly, summarizes the results of Oracle Corporation's own scalability testing of Forms Services. Then scalability is defined, followed by a description of the testing methodology used by Oracle, as well as a discussion on common mistakes that many people make when conducting their own testing. Then the results are examined in depth.

Appendix 1 contains information on one of the applications used for testing by Oracle.

Note: For the sake of brevity, the Forms Listener Servlet has been abbreviated to FLS in this document. Oracle Forms Services Release 6*i* is often abbreviated to Forms Services, or simply Forms.

Summary of Results

To understand the scalability of Forms Services, a number of benchmarks were conducted using popular operating systems and hardware. The results of these benchmarks appear in Table 1. A detailed analysis of the results appears in **Scalability Testing Results**, on page 16.

In summary, the results confirm that Oracle Forms Services is an extremely scalable application deployment platform that scales linearly on various operating systems and hardware.

		Solaris Sun 450R Up to 4 CPUs @ 450 MHz each Up to 4 GB RAM		**Windows** Compaq ProLiant DL380 Up to 4 CPUs @ 864 MHz each Up to 4 GB RAM	
		Small	**Medium**	**Small**	**Medium**
Architecture	**Forms Listener**	220 users / CPU 2.6 Mb / user	180 users / CPU 5.8 Mb / user	420 users / CPU 4.0 Mb / user	300 users / CPU 6.2 Mb / user
	Forms Listener Servlet	200 users / CPU 2.7 Mb / user	150 users / CPU 6.2 Mb / user *	Not yet available	Not yet available

Table 1: Results summary of Forms 6*i* scalability testing

* For now, it is important to realize that the Forms runtime process does not consume more CPU or memory when using the Forms Listener Servlet (FLS). Rather, there is an overhead from the Servlet Engine, which consumes resources, thus leaving less memory and CPU for the Forms runtime processes. This is fully explained the results section on page 16.

1 SCALABILITY DEFINED

Scalability, within the context of this guide, refers to the ability to accommodate an ever-increasing user population on a single system by simply adding more hardware resources and not having to change any of the underlying software.

1.1 Why is this important to me?

A CIO or an Information Systems Manager will benefit greatly from knowing if the investment that they are making in an enterprise system will be technology insurance for the future. In other words, understanding how an enterprise system scales will enable you to make an informed decision about a significant investment.

The most common questions asked when considering a deployment platform are:

1. How much memory will each user consume?
2. How many concurrent users per CPU can I run?
3. What hardware will I need for *n* concurrent users?
4. What happens if I add more users in the future?

Figure 1: Common questions when deploying a Forms application

There is no way to answer those questions with 100% accuracy as there are so many variables. But a scalable architecture allows you to make predictions with confidence because it exhibits a known and reliable behavior.

1.2 Who should read this document?

This paper is aimed at helping business decision makers, consultants, and application or system administrators realistically determine their hardware requirements for Oracle Forms Services, given their user communities, application complexity, etc.

It is assumed that the reader is familiar with the Oracle Forms Services architecture.

2 METHODOLOGY

This chapter describes the methodology used for the Forms scalability testing, the hardware used, plus common mistakes people make when benchmarking.

The aim of the tests was not to "obtain numbers", or see how many users may be "shoe-horned" onto a platform, but rather to establish a pattern of behavior for the scalability of Forms. This allows companies to determine the number of users that may be supported with acceptable performance for a given environment. Any numbers obtained would only be relevant for the specific applications run in the specific environments used. See the results chapter for a full explanation.

2.1 Overview

The following steps are an overview of the methodology used:

1. Define a realistic scenario.
2. Record the scenario.
3. Playback the scenario to simulate concurrent users, each time simulating more users.
4. When performance is degraded beyond usability, you have reached the maximum number of users.

A perfectly scalable architecture should produce a graph similar to Figure 2.

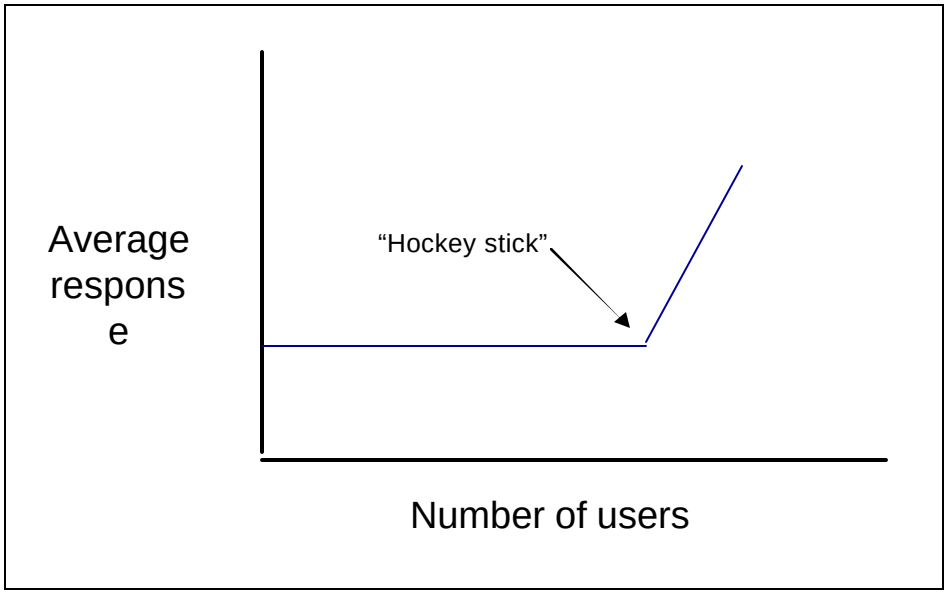


Figure 2: Example graph for a scalable architecture

2.1.1 Performance Degradation

Rather than performance slowly degrading as more users are added, it should remain constant until the machine is saturated: the "hockey stick" or "knee" point on the graph. Once the saturation point is reached, performance degrades drastically.

(A simple analogy might be playing an audio cassette tape using older alkaline batteries versus more modern lithium batteries. With alkaline batteries, the tape would play at normal speed for a while, then it would begin to play slower and slower as the batteries drained, until it eventually stopped. With new lithium batteries, the tape would play at full speed right up until the batteries were drained, and then stop suddenly.)

2.1.2 Number of users

For the scalability tests conducted on Forms , the number of simulated concurrent users run were in steps of 50. The first test was run with 50 users, the second with 100 users, the third with 150, and so on.

2.1.3 Response time

An acceptable average response time was defined as up to twice the average response time for 1 user. For example, if the average response time for 1 user was 30 milliseconds, then an average response time of up to 60 milliseconds was deemed acceptable.

Note: The response time is defined as the time it takes to send a "Forms message" (**not** a TCP/IP packet) from the client to the server, and get a response. It is **not** the time an action takes, as perceived by the user. There is a relationship, though, between the two.

2.1.4 Test Completion

Tests were run until performance degraded sharply, indicating that the server had reached saturation point. The saturation point may or may not be within an acceptable average response time. Table 2 contains an example result set.

Number of Users	Average Response Time (ms)
1	110.97
50	111.12
100	112.25
150	115.22
200	120.73
250	128.49
300	602.87

Table 2: Example table of response times and the saturation point

The saturation point clearly occurred somewhere between 250 and 300 users, since the average response time jumped drastically to 602.87 milliseconds. In this case, the saturation occurred while the response time was within an acceptable limit: less than 2 x 111.12 = 222.24.

As a point of interest, for the test results shown in Table 2, it would be stated that the scalability limit was 250 users. Often, there was no drilling down to determine a more exact number. This means that the results in this document are understated and the actual results would be better.

For instance, in the example of Table 2, the real scalability limit might be 295 users (versus the 250 stated). Or it could be 255 users. Either way, it is better than stated.

When you benchmark your application in your environment, once you broadly work out the saturation point, you may want to conduct finer-grained tests to determine a more accurate figure.

2.2 Two types of tests

In order to determine the memory consumption per user, and the number of concurrent users per CPU that can be run (see Figure 1, on page 5), the following tests were used:

- Memory-bound Test
- CPU-bound Test

The results and experience from running the tests not only answer those questions, but establish a pattern of behavior for the scalability of Forms so that predictions can be made. This will help in answering questions 3 and 4 in Figure 1.

2.2.1 Memory-bound Test

This test is used to determine the amount of memory consumed per user. It is conducted by forcing the memory to be the bottleneck. When the server becomes saturated, then you know that it was because of memory.

In practice, this is done by configuring a machine with far more CPU available than memory. In the case of the tests conducted by Oracle, a machine with 4 CPUs and 1 Gb of memory was used. This ensures that all of the memory will be used before the CPU is fully utilized.

For example, you run tests using the configuration just described, and determine the scalability limit to be 300 users. Then it can be stated that you can run 300 users per Gb. A more useful metric for many people is the amount of memory consumed per user. Divide 300 users by 1 gigabyte and you get 3.4 Mb per user.

2.2.2 CPU-bound Test

This is the opposite of the memory-bound test. In this case, a machine with 4 Gb RAM and only 1 CPU was configured. Thus, the CPU is fully utilized before all of the memory is consumed. When the server becomes saturated, then you know it was because of the CPU.

For example, you run tests using the configuration just described, and determine the scalability limit to be 250 users. Then you can state that you are able to run 250 users per CPU.

2.3 Application Complexity

Two differently sized applications were used in the Forms scalability benchmarking tests: small and medium. For large forms, the results of Oracle's E-Business suite of applications were used since they conduct independent benchmarking.

2.3.1 Small Application

A small application is defined as having

- Less than 1 Mb of files opened concurrently (.FMX, .PLX, etc)
- Less than 10 blocks, 5 canvases, and 3 windows
- Maximum of 2 forms opened concurrently

The small application used is called the Summit application, and is based on the application used by Oracle Education in their Forms training courses. Information on this application may be found in Appendix 1.

2.3.2 Medium Application

A medium application is defined as having

- Between 1-3 Mb of files opened concurrently (.FMX, .PLX, etc)

- 10-20 blocks, 5-15 canvases, and 3-10 windows
- Maximum of 3-5 forms opened concurrently

An Oracle customer (who cannot be named for legal reasons) donated their Forms application to be used as a sample medium application. It is a "real-life" help-desk and customer support application, designed and used by this customer.

2.3.3 Large Application

A large application is defined as having

- Greater than 3 Mb of files opened concurrently (.FMX, .PLX, etc)
- More than 20 blocks, more than 15 canvases, and more than 10 windows
- Greater than 5 forms opened concurrently

2.4 Scenarios

The scenarios used included common actions performed by users, such as:

- Querying data
- Inserting, updating, and deleting data
- Navigating between fields and canvases within a form
- Navigating between forms
- And so on…

In the benchmarking performed by Oracle, the scenario recorded was of a user working relatively intensively. The number of actions performed by users can significantly affect scalability.

2.5 Architecture

The following two architectures were tested:

- Forms Listener in socket mode
- Forms Listener Servlet

2.5.1 Forms Listener

Figure 3 gives an overview of the benchmark architecture used for the Forms Listener.

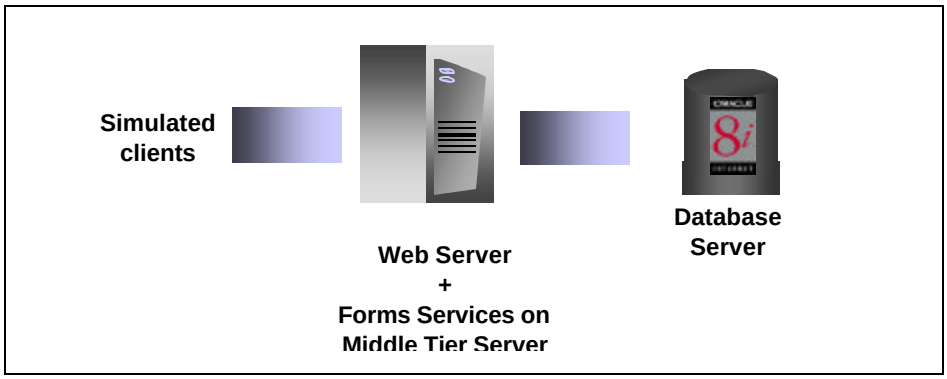


Figure 3: Forms Listener Architecture

The Forms runtime processes ran on one server. An Oracle8*i* RDBMS ran on a separate server. A third machine was used to run the client simulation software.

2.5.2 Forms Listener Servlet

As can be seen in Figure 4, the middle tier has been split into two separate machines: one hosting the Application Server, and the other hosting Forms Services.

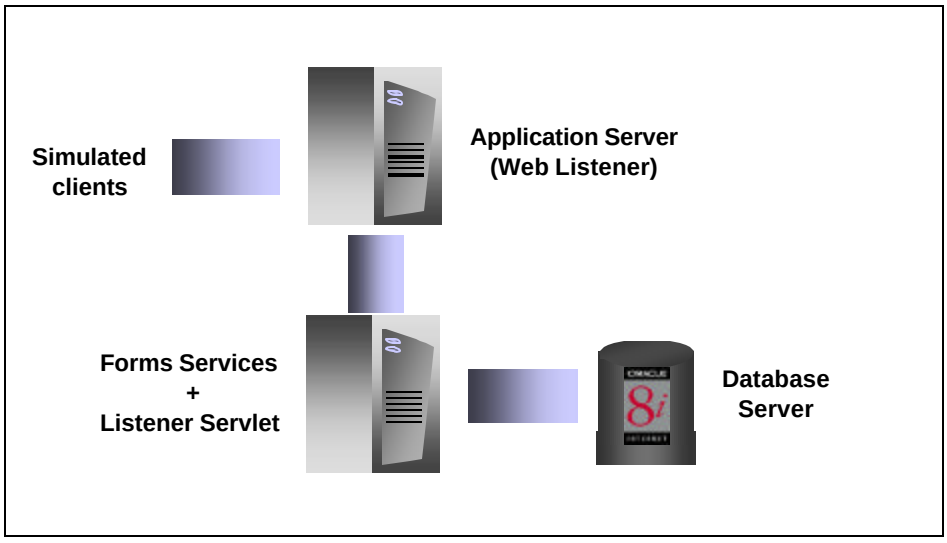


Figure 4: Forms Listener Servlet Architecture

There is no reason why the Application Server and Forms Services cannot be run on a single machine. But by having Forms on a machine by itself, you guarantee that the CPU utilization and memory consumed are due to Forms, which makes for a more reliable benchmark. Moreover, it is representative of many companies, where a machine is dedicated as the Application Server.

2.6 Software and Hardware

The Application Server used was Oracle9*i*AS version 1.0.2.2. The version of Forms Services used was Forms 6*i* (patchset 4). The database used was an Oracle8*i* RDBMS, version 8.1.6.

The Application Server, the database, and the client simulation software were each run on different hosts, which were the following:

- Sun 450R
- Solaris 2.6 operating system
- 4 CPUs @ 450 MHz each
- 4 Gb RAM

When testing on Solaris, Forms Services was installed on an additional Sun 450R with the configuration described above. (For the memory-bound tests, memory was physically removed. For the CPU-bound tests, OS software was used to control the number of CPUs used.)

When testing on Microsoft Windows, the following machine was used to host Forms Services:

- Compaq ProLiant DL380
- Windows 2000 operating system
- 4 CPUs @ 864 MHz each
- 4 Gb RAM

Similarly, not all of the CPUs and memory were necessarily used in all tests.

2.6.1 Client Simulation Software

In a benchmark scenario, it is impractical to configure a number of client machines and end users that accurately represent a live application environment. Therefore, load simulators are used to simulate real users that perform transactions.

The simulation software utilized in these tests is a tool used by Oracle Consulting to conduct benchmarking on Oracle customer systems. Companies that wish to have their own applications tested may contract Oracle Consulting to perform the benchmarking in a professional manner.

Alternatively, if you wish to conduct your own benchmarking, there are third party load simulation products on the market, such as LoadRunner from Mercury Interactive.

2.7 The Wrong Approach

Having a correct testing methodology is essential to obtaining accurate results. The section discusses some common mistakes that people make when conducting their own tests.

2.7.1 Memory Consumption

The only true way to determine the scalability limit is to push the server past the saturation point. Only then can accurate figures be obtained. Here is a real-life example which illustrates mistakes that are commonly made.

Bob sets up Forms Services on a server with 4 CPUs and 1 Gb RAM, running Windows NT – this is the middle tier. The database is on a separate server. On a third machine, Bob launches his (fairly small) Forms application in a browser.

On the middle tier, Bob opens the Task Manager, clicks on the Processes tab, and observes that the ifweb60 process is consuming 15 Mb. His initial thought -- as would be many people's -- is, "What! Forms is using 15 Mb per user? That means my 1 Gb machine can only run 68 users!"

This is misleading and incorrect.

Bob then starts up his simulation software and simulates 50 users. He looks at the memory usage on the middle tier and calculates that Forms is using an average of 6.2 Mb per user.

"With 6.2 Mb per user," he thinks, "I can run 165 users. That's better than 68, but I'm only using a small Forms application."

Again, these results are misleading.

Bob then runs several tests to stress his middle tier server until performance degrades sharply and he has determined the saturation point. **He then calculates that Forms, when the machine was fully utilized, was consuming only 2.8 Mb per user.**

This is the correct result. Why is this?

Simply measuring the memory consumption using the operating system memory tool can lead to misleading results. Often, the tool doesn't even report the correct figure because the OS has not released the memory, or not updated its internal tables, for instance.

More importantly, a lot of work has gone into optimizing Forms -- and the way it manages memory, for example, cannot be determined by simply running one or two processes and looking at memory consumption. Forms may grab more

memory than it really needs in anticipation of needing more in the near future. So while the memory manager reports that 15 Mb is consumed, Forms may only be using 5 Mb of it, say.

In addition, Forms processes are able to share executable code and the application image. If 200 users open the same form, there are many objects in memory that can be shared. Even if different forms are opened concurrently, there are still many objects in memory that can be shared.

Therefore, when you have only one process, there is nothing to share with, so that process appears bloated. Moreover, when Forms is finished with some memory, it may choose not to release it immediately. Allocating memory is a relatively expensive operation so it may retain the memory until it absolutely has to be released.

That is why it is extremely important, when benchmarking, to stress the server past the saturation point. Just running 1, 10, 50, or even 100 users, and extrapolating your figures from that will yield misleading results.

2.7.2 Define a Realistic Scenario

When you record a scenario that is to be played back, there will be potentially hundreds of simulated users playing that scenario concurrently. It is important that the scenario be representative of the actions of your user population. The scenario should perform the same types of operations as your users – this is obvious to most people – but also at the **same speed** your users would.

Often, testers will carefully research what actions their users perform. Satisfied, they will then record a scenario, but record the actions in quick succession, thinking that it's the only the transactions themselves that are important. Equally important is the *think time*, the time between operations.

For example, your application might be a help desk system that your users use while on the phone to customers. An average phone call might last 5 minutes, and the users perform an average of 10 transactions in that time, which is 2 transactions per minute. (A transaction may be a set of operations, such as inserting or updating, navigating between fields, canvases, forms, etc.)

If you then record a scenario that performs those same types of transactions, but instead does 5 transactions per minute, then, when you simulate many users, the Forms server and the database are working harder than they normally would. You will obtain false results.

Even though you may have 200 concurrent users, the Forms server isn't really doing 200 "things" at exactly the same time. People pause when they work; they look something up on a piece of paper; they talk to a customer on the phone; they think. These gaps are a long time to a computer, even for data entry users who type fairly constantly. By recording a scenario without realistic think time – without those gaps – your server is working harder than usual and you will get misleading results.

Therefore, when defining and recording your scenario, it is critical to not only record the type of actions that your users would perform, but also at the same speed. That will yield more accurate benchmarking results.

3 SCALABILITY TESTING RESULTS

This chapter analyzes the results of Oracle's benchmarking of Forms, which are shown in Table 3.

		Solaris		Windows	
		Small	**Medium**	**Small**	**Medium**
Architecture	**Forms Listener**	220 users / CPU 350 Users / GB 2.6 Mb / user	180 users / CPU 160 Users / GB 5.8 Mb / user	420 users / CPU 230 Users / GB 4.0 Mb / user	300 users / CPU 150 Users / GB 6.2 Mb / user
	Forms Listener Servlet	200 users / CPU 340 Users / GB 2.7 Mb / user	150 users / CPU 150 Users / GB 6.2 Mb / user	Not yet available	Not yet available

Table 3: Results summary of Forms 6*i* scalability testing

3.1 Background

The memory per user is calculated from the number of users per gigabyte. For each scenario, it was assumed that 100 Mb was used by the operating system, and therefore not available to Forms.

When testing with the Forms Listener Servlet architecture, separate machines were used for the Application Server and for the Forms runtime processes. See the Methodology chapter for details on the benchmarking setup.

3.2 Disclaimer

It is important to understand that the information and results published in this documents are not to be used to compare the Sun architecture and the Microsoft architecture. This is especially important because the machines used are not comparable. The Sun machines were based on 450 MHz CPUs while the Windows machines were based on 864 MHz machines. We believe that this information only provides the reader with more choices.

The numbers obtained by Oracle's benchmarking pertain to Forms 6*i*, and to the specific scenario and environment used in the tests. **The only true way to test the scalability for your application is to run your own benchmarking tests.** There are so many variables that it is impossible to predict what the results will be from one environment to the next. Here is a list, by no means exhaustive, of some of those variables:

- The client specification: memory and CPU.
- The middle tier specification: memory and CPU.
- The database tier specification: can the database keep up with the requests from all of your concurrent users or is it a bottleneck?
- The network topology.
    - Between the client and the middle tier.
    - Between the middle tier and the database.
- Some Forms applications are more memory-intensive while some are more CPU-intensive, depending on what they do.
    - Do they do a lot of calculations (CPU-intensive)? Do they process large amounts of data (memory-intensive)?
    - Is your logic in Forms or in stored procedures in the database?
- How fast do your users work? Many transactions per minute, or few transactions per minute?
- Does your application have many modules or few? How many are open at one time? Do they have complex GUI screens or simple ones?
- Do your users run the one application all day, or are they stopping and starting it several times a day?

All of these inputs, and many more, combine to produce different actual scalability numbers for each system. However, the way that Forms scales remains consistent, as will be shown in the next section.

The methodology used, while more realistic than purely empirical, does introduce a degree of subjective qualification to the test. What may be acceptable to one group of users may not be acceptable to another. If the complexity of your application matches the small or medium application used in these tests, then you may be able to use the results as a rough estimate. There is no substitute for conducting your own tests, though.

3.3 Analysis

The significance from the benchmarking is not the numbers themselves, but the way that the scalability of Forms behaves.

The scalability testing has shown that Forms scales linearly. If you double the hardware, you roughly double the number of users. Therefore:

- If you are deploying for the first time, you can run benchmarks and then be able to predict with accuracy your hardware requirements.

- If you wish to add more users to your current deployment, you can predict with accuracy your hardware requirements.

This is shown by the graphs, below. Figure 5 shows the results of a CPU-bound test, where the RAM was effectively infinite and hence not a factor. The first CPU was able to support approximately 150 users. Each time a CPU was added, another 150 users were able to run.

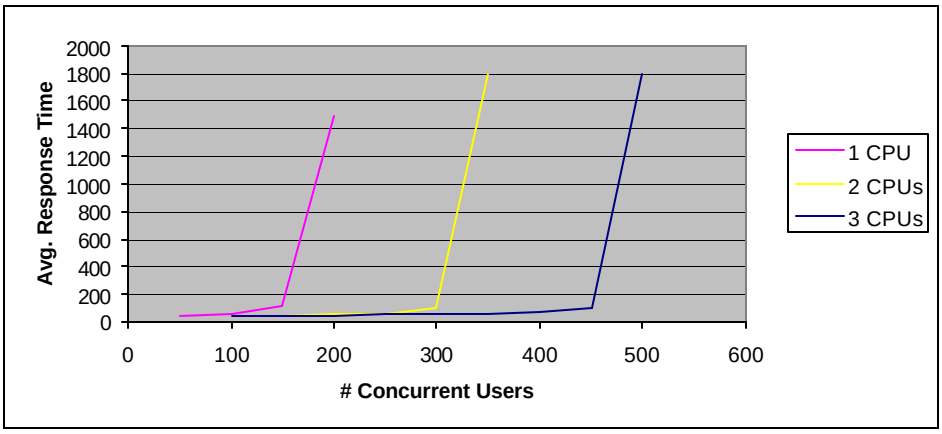


Figure 5: CPU-bound, Solaris, medium application, Forms Listener Servlet

Figure 6 contains the results of a memory-bound test, where the CPU capability was effectively limited. For 1 Gb RAM, around 160 users were able to run. When another gigabyte of RAM was added, roughly 320 users were supported,

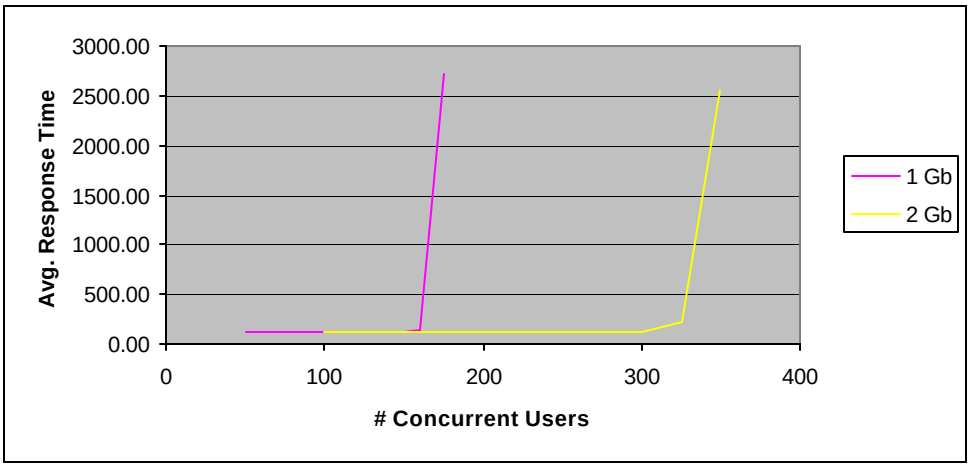


Figure 6: Memory-bound, Solaris, medium application, Forms Listener

In both Figure 5 and Figure 6, notice how the response time stays constant as users are added, right up until and the "hockey stick" is reached and performance degrades.

It is worth reiterating that it is not the actual results that are important, but the scalability behavior of Forms and the fact that it scales linearly.

How is this useful in making predictions? For example, Company 1 and Company 2 both have Forms applications. Company 1, given their hardware, environment, and scenario, are able to run 200 concurrent users. Company 2, with a different system and application, are able to run 900 concurrent users.

If Company 1 and Company 2 double their hardware, then they will be able to run roughly 400 and 1800 users, respectively. i.e. Even though the actual scalability numbers obtained by Company 1 and Company 2 are different, the scalability behavior of Forms remains the same.

3.3.1 Response Time

The tests showed that an acceptable response time was always within plus-or-minus ten percent of the machine's saturation point. This is another indication that Forms scales well because the average response time stayed fairly level until the server was overloaded.

3.3.2 The Forms Listener Servlet

The FLS is a new architecture (introduced in a patchset after the initial release of Oracle Forms 6*i*) designed to let Forms use **standard network communication** so that Forms applications could run on any network. The previous architecture was proprietary; the client communicated directly with the Forms Server (socket, HTTP, and HTTPS modes). Some of the benefits of the Forms Listener Servlet architecture are:

- It is a non-proprietary, standard solution
- HTTP/1.1 or HTTP/1.0 supported
- The web server's certificate is used for SSL communication
- You can use standard load balancing techniques, including hardware load balancing

There are other benefits to using the FLS, which you can read about in the Forms section on http://otn.oracle.com/products/forms/content.html.

Along with the standard architecture and other advantages of the FLS comes a price to pay in terms of overhead. On each server where you run Oracle Forms Services, the Forms Listener Servlet manages the network communication with the client. This Java Servlet consumes memory and CPU, leaving less for the Forms runtime processes, **with no perceivable impact on performance or response times**.

Figure 7 and Figure 8 show the impact of the Forms Listener versus the Forms Listener Servlet, for both CPU-bound and memory-bound.

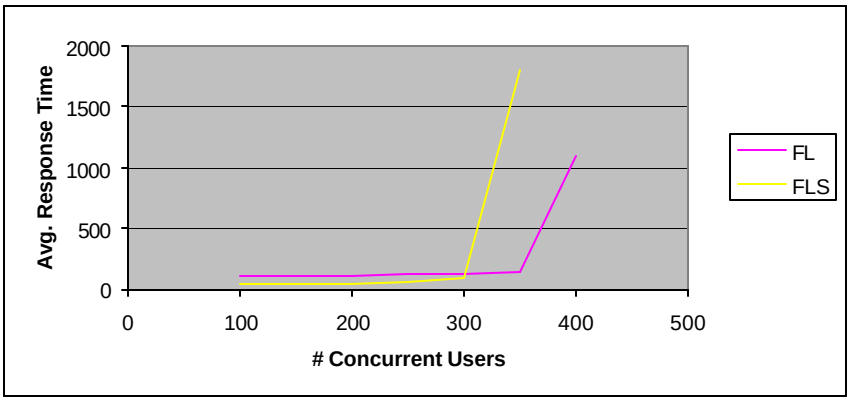


Figure 7: CPU-bound, Forms Listener versus Forms Listener Servlet Medium application on Solaris, 2 CPUs, 4 GB RAM

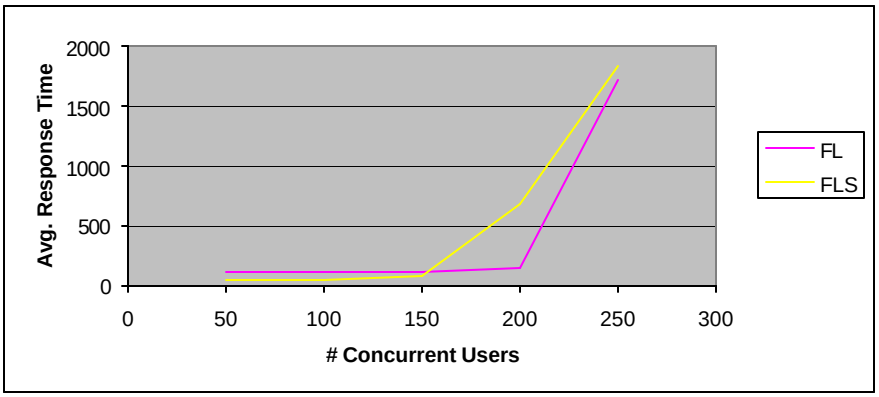


Figure 8: Memory-bound, Forms Listener versus Forms Listener Servlet Medium application on Solaris, 4 CPUs, 1 GB RAM

It is important to realize that the behavior and scalability of Forms is not affected. The same runtime process is used for the both Forms Listener and the Forms Listener Servlet architecture. The only difference is that the servlet engine, which runs on the same host as the Forms runtime processes, consumes CPU and memory, leaving less for Forms.

3.3.2.1 Impact on the Application Server

Using the Forms Listener, the client communicates directly with the Forms runtime process. With the Forms Listener Servlet architecture, all Forms communication goes through the web server, thereby increasing the load on that machine.

In the Forms scalability testing, it was found that there was approximately a 20% increase on CPU usage, and a 25% increase in memory consumption on the machine that hosted the Application Server. There is no impact on performance or response times.

Each environment will have varying figures, depending on the usage of their application.

3.3.3 Large Application

Oracle's E-Business suite of applications, who conduct separate testing, is representative of a large Forms application. **Impressively, they were able to run 16016 concurrent users with an average response time of 1.01 seconds.** The reader is referred to their results, which may be found at the following location:

http://www.oracle.com/apps_benchmark/index.html

3.4 Conclusion

There is absolutely no substitute for conducting your own benchmarking to determine the scalability of your application. But some general comments can be made.

The results from the benchmark tests show that Oracle Forms Services is an extremely scalable application deployment platform which scales linearly on various operating systems and hardware. That is difficult to achieve and few applications in the market today can make such a claim.

Because of the scalability of Forms, Oracle customers can have confidence that Forms will meet their future demands, and business decisions concerning future growth -- such as purchasing hardware, whether to build a new system or expand an existing one, and so on – can be made safely.

APPENDIX A – SMALL APPLICATION

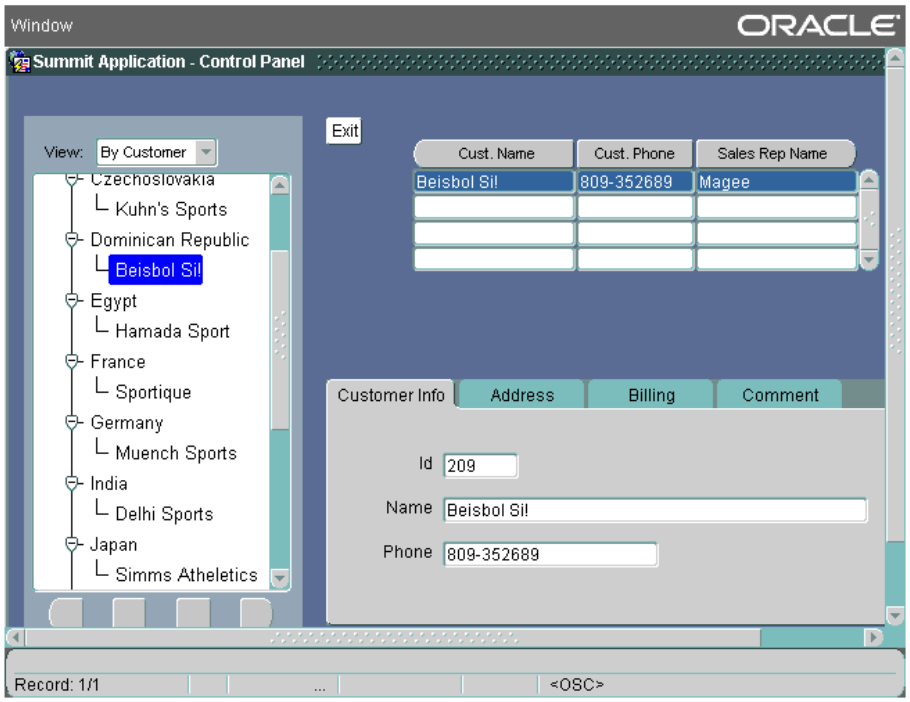
The Summit application was used for the Small Application. Oracle Education originally designed it for use in their training courses.

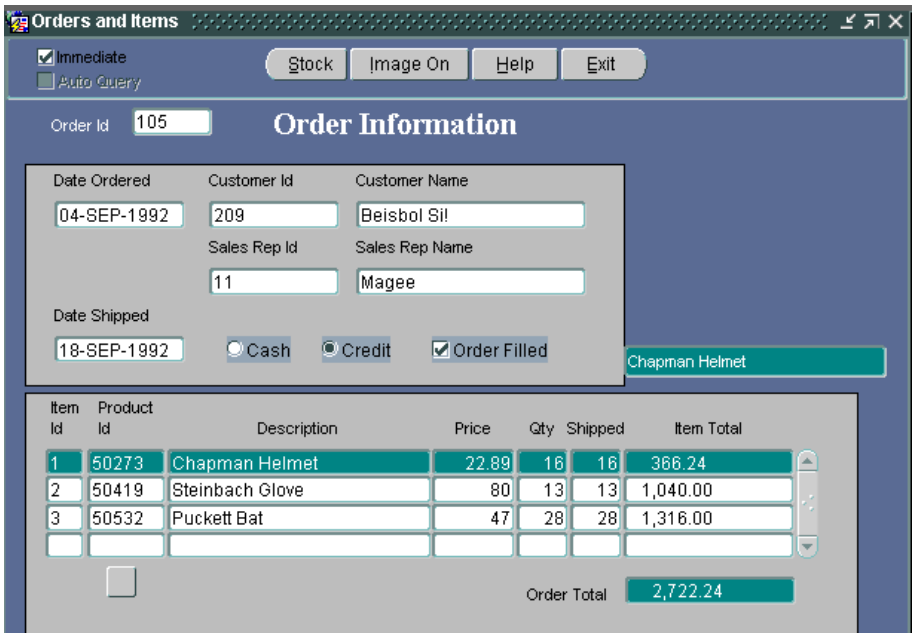
You may download the application from OTN to perform your own benchmarking and compare your results with those in this document. Use a browser and go to the following URL:

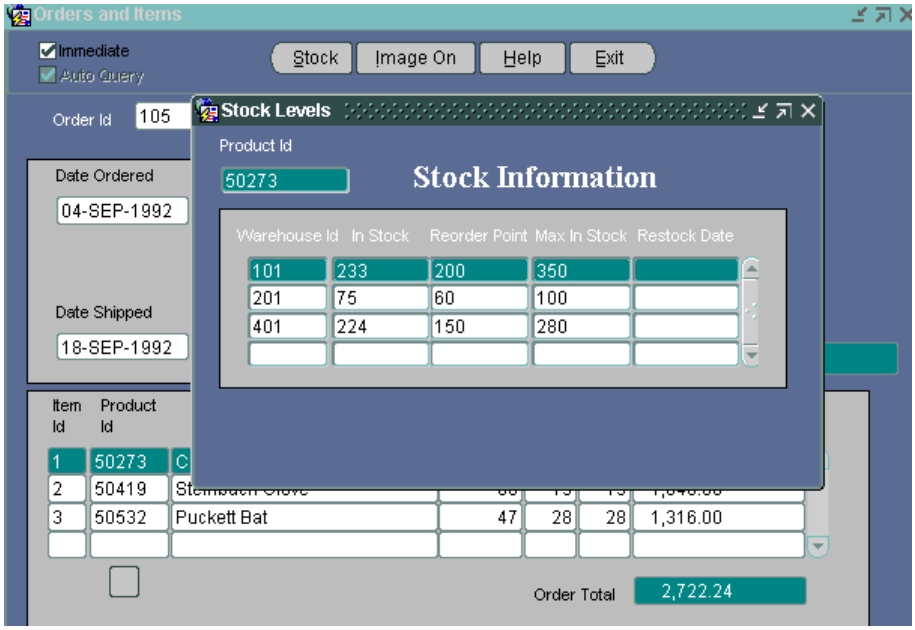
http://otn.oracle.com/products/forms/content.html

From there, you should be able to navigate around and find the Summit application to download it. It will contain the necessary Forms modules, database schema, instructions for installing, plus the scenario used in the scalability testing of Forms.

Some screen shots of the application are provided, below.




Orders and Items
Immediate
Auto Query
Stock
Image On
Help
Exit
Order Id
105
Order Information
Date Ordered
Customer Id
Customer Name
04-SEP-1992
209
Beisbol Si!
Sales Rep Id
Sales Rep Name
11
Magee
Date Shipped
18-SEP-1992
Cash
Credit
Order Filled
Chapman Helmet
Item Id
Product Id
Description
Price
Qty
Shipped
Item Total
1
50273
Chapman Helmet
22.89
16
16
366.24
2
50419
Steinbach Glove
80
13
13
1,040.00
3
50532
Puckett Bat
47
28
28
1,316.00
Order Total
2,722.24


Orders and Items
Immediate
Auto Query
Stock
Image On
Help
Exit
Order Id
105
Stock Levels
Product Id
50273
Stock Information
Warehouse Id
In Stock
Reorder Point
Max In Stock
Restock Date
101
233
200
350
201
75
60
100
401
224
150
280
Date Ordered
04-SEP-1992
Date Shipped
18-SEP-1992
Item Id
Product Id
1
50273
2
50419
3
50532
Puckett Bat
47
28
28
1,316.00
Order Total
2,722.24

White Paper Title
November 2000
Author: Forms Product Management
Contributing Authors: PerfScale Team

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.