An Oracle White Paper
First Published October 2007

Optimizing Oracle's Siebel Application on Oracle Servers with CoolThreads Technology

ORACLE

Introduction

In an effort to ensure the most demanding global enterprises can meet their deployment requirements, Oracle engineers continue to work to improve Oracle's Siebel Server performance and stability on the Oracle Solaris operating system (OS).

This technical white paper is an effort to make available tuning and optimization knowledge and techniques for Oracle's Siebel 7.x eBusiness Application Suite on the Oracle Solaris platform. All the techniques discussed in this document are lessons learned from a series of performance tuning studies conducted under the auspices of the Siebel Platform Sizing and Performance Program (PSPP). The tests conducted under this program are based on real world scenarios derived from Oracle's Siebel customers, reflecting some of the most frequently used and critical components of the Oracle eBusiness Application Suite. Tips and best practices guidance based on Oracle’s experience is provided for field staff, benchmark engineers, system administrators, and customers interested in achieving optimal performance and scalability with Siebel on Oracle’s Sun server installations. The areas addressed include:

- The unique features of Oracle Solaris that reduce risk while helping to improve the performance and stability of Oracle's Siebel applications
- The optimal way to configure Oracle's Siebel applications on Oracle Solaris for maximum scalability at a low cost
- How Oracle's UltraSPARC® T1 and UltraSPARC IV+ processors and chip multithreading (CMT) technology benefit Oracle Database Server and Oracle's Siebel software
- How transaction response times can be improved for end users in large Siebel deployments
- How Oracle database software running on Oracle’s Sun storage systems can be tuned for higher performance for Oracle's Siebel software

The performance and scalability testing was conducted at Oracle's Enterprise Technology Center (ETC) in Menlo Park, California, with assistance from Siebel Systems. The ETC is a large, distributed testing facility that provides the resources needed to test the limits of software on a greater scale than most enterprises ever require.

Price/Performance Using Oracle Servers and Oracle Solaris 10

Oracle's Siebel CRM application is a multithreaded, multiprocess and multi-instance commercial application.

- *Multithreaded applications* are characterized by a small number of highly threaded processes. These applications scale by scheduling work through threads or Light Weight Processes (LWPs) in the Oracle Solaris operating system. Threads often communicate via shared global variables.
- *Multiprocess applications* are characterized by the presence of many single threaded processes that often communicate via shared memory or other inter-process communication (IPC) mechanisms.
- *Multi-instance applications* are characterized by having the ability to run more than one set of multiprocess or multithreaded processes.

Chip multithreading technology brings the concept of multithreading to hardware. *Software multithreading* refers to the execution of multiple tasks within a process. The tasks are executed using software threads which are executed on one or more processors simultaneously. Similar to software multithreading techniques, CMT-enabled processors execute many software threads simultaneously within a processor on cores. In a system with CMT processors, software threads can be executed simultaneously within one processor or across many processors. Executing software threads simultaneously within a single processor increases processor efficiency as wait latencies are minimized.

Oracle's UltraSPARC T1 processor-based systems offer up to eight cores or individual execution pipelines per chip. Four strands or active thread contexts share a pipeline in each core. The pipeline can switch hardware threads at every clock cycle, for an effective zero wait context switch, thereby providing a total of 32 threads per UltraSPARC T1 processor. Siebel's highly threaded architecture scales well and can take advantage of these processor characteristics. Extreme scalability and throughput can be achieved when all Siebel application worker threads execute simultaneously on UltraSPARC T1 processors.

The UltraSPARC T1 processor includes a single floating point unit (FPU) that is shared by all processor cores. As a result, the majority of the UltraSPARC T1 processor's transistor budget is available for Web facing workloads and highly multithreaded applications. Consequently, the UltraSPARC T1 processor design is optimal for the highly multithreaded Siebel application, which is not floating-point intensive.

Sun Fire servers from Oracle incorporating the UltraSPARC T1 processor provide SPARC Version 7, 8, and 9 binary compatibility, eliminating the need to recompile the Siebel application. Lab tests show that Siebel software runs on these systems right out of the box. In fact, Oracle servers with UltraSPARC T1 processors run the same Oracle Solaris 10 OS as all SPARC servers, with compatibility provided at the instruction set level as well as across operating system versions.

Oracle's Siebel software can benefit from the several memory related features offered by Oracle's UltraSPARC T1 processor-based systems. For example, large pages reduce translation lookaside buffer (TLB) misses and improve large data set handling. Each core in an UltraSPARC T1 processor-based

system includes a 64-entry instruction translation lookaside buffer (iTLB), as well as a data translation lookaside buffer (dTLB). Low latency Double Data Rate 2 (DDR2) memory reduces stalls. Four on-chip memory controllers provide high memory bandwidth, with a theoretical maximum of 25 GB per second. The operating system automatically attempts to reduce TLB misses using the Multiple Page Size Support (MPSS) feature. In Oracle Solaris 10 update 1/06, text, heap, anon and stack are all automatically placed on the largest page size possible. There is no need to tune or enable this feature. However, earlier versions of Oracle Solaris require MPSS to be enabled manually. Testing at Oracle shows a 10 percent performance gain when this feature is utilized.

Oracle's UltraSPARC T1 processor-based systems require Oracle Solaris 10 or later versions. Siebel CRM applications have been successfully tested on these systems, revealing a 500 percent improvement in performance over other enterprise-class processors. Oracle has certified Siebel 7.8 on Oracle Solaris 10.

The UltraSPARC IV processor is a two core, chip multiprocessing (CMP) processor derived from previous generation UltraSPARC III processors. Each core in the UltraSPARC IV processor includes an internal L1 cache (64 KB data, 32 KB instruction) and a 16 MB external L2 cache. The L2 cache is split in half, with each core able to access 8 MB of the cache. The address and data bus to the cache are shared. The memory controller resides on the processor, and the path to memory is shared between the two processor cores. Each core can process four instructions per clock cycle. The cores include a 14 stage pipeline and use instruction-level parallelism (ILP) to execute four instructions every cycle. Together, two cores can execute a maximum of eight instructions per clock cycle.

Similar to the UltraSPARC IV processor, UltraSPARC IV+ processors incorporate two cores. Key improvements in the UltraSPARC IV+ processor include:

- Higher clock frequency (1.8 GHz)
- Larger L1 instruction cache (64 KB)
- On-chip 2 MB L2 cache, shared by the two cores
- Off-chip 32 MB L3 cache

Oracle Solaris is the cornerstone software technology that enables Oracle servers to deliver high performance and scalability for Siebel applications. Oracle Solaris contains a number of features that enable organizations to tune solutions for optimal price/performance levels. Several features of Oracle Solaris contributed to the superior results achieved during testing efforts, including:

- **Siebel process size.** For an application process running on Oracle Solaris, the default stack and data size settings are unlimited. Testing revealed that Siebel software running with default Oracle Solaris settings resulted in a bloated stack size and runaway processes which compromised the scalability and stability of Siebel applications on Oracle Solaris. Limiting the stack size to 1 MB and increasing the data size limit to 4 GB resulted in increased scalability and higher stability. Both of these adjustments let a Siebel process use its process address space more efficiently, thereby allowing the Siebel process to fully utilize the total process address space of 4 GB available to a 32-bit application process. These changes significantly reduced the failure rate of transactions — only eight failures were observed out of 1.2 million total transactions.

- **Improved threads library in Oracle Solaris 10.** The Oracle Solaris 9 operating system introduced an alternate threads implementation. In this one level model (1x1), user-level threads are associated with LWPs on a one-to-one basis. This implementation is simpler than the standard two-level model (MxN) in which user-level threads are multiplexed over possibly fewer LWPs. The 1x1 model used on Oracle's Siebel Application Servers provided good performance improvements to Siebel multithreaded applications. In Oracle Solaris 8, this feature was enabled by users. It is the default for all applications as of Oracle Solaris 9.
- **Use of appropriate Oracle hardware.** Pilot tests were conducted to characterize the performance of Web, application, and database applications across the current Oracle product line. Hardware was chosen based on price/performance, rather than pure performance characteristics. Servers differing in architecture and capacity were chosen deliberately so that the benefits of Oracle's hardware product line could be discussed. Such information provides customers with data that aid the selection of a server solution that best fits their specific needs, including raw performance, price/performance, reliability, availability, and serviceability (RAS) features, and deployment preferences such as horizontal or vertical scalability. Oracle Database 10*g* files were laid out on a Sun StorEdge 3510 FC array. I/O balancing based on Siebel workload was implemented, reducing hot spots. In addition, zone bit recording was used on disks to provide higher throughput to Siebel transactions. Direct I/O was enabled on certain Oracle files and the Siebel file system.
- **Oracle's database Connection Pooling.** Oracle's database Connection Pooling was used, providing benefits in CPU and memory. In fact, 20 end users shared a single connection to the database.

Oracle's Siebel Application Architecture Overview

Oracle's Siebel Server is a flexible and scalable application server platform that supports a variety of services operating on the middle tier of the Siebel N-tier architecture, including data integration, workflow, data replication, and synchronization service for mobile clients. Figure 1 provides a high level architecture of the Siebel application suite.

The Siebel server includes business logic and infrastructure for running the different CRM modules, as well as connectivity interfaces to the back-end database. The Siebel server consists of several multithreaded processes, commonly referred to as Siebel Object Managers, which can be configured so that several instances run on a single Solaris system. The Siebel 7.x server makes use of gateway components to track user sessions.

Siebel 7.x has a thin client architecture for connected clients. The Siebel 7.x thin client architecture is enabled through the Siebel Webserver Extension (SWSE) plugin running on the Web server. It is the primary interface between the client and the Siebel application server. More information on the individual Siebel components can be found in the Siebel product documentation at *www.oracle.com/applications/crm/index.html*

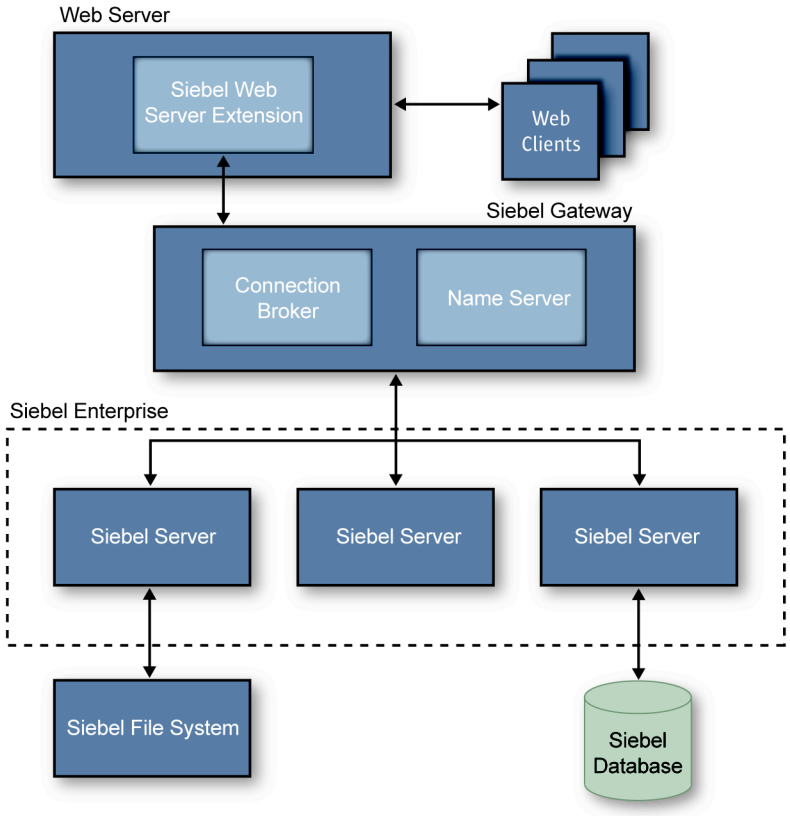

Figure 1. A high-level view of Oracle's Siebel application architecture.

Optimal Architecture for Benchmark Workload

Oracle offers a wide variety of products ranging from hardware, software and networking, to storage systems. In order to obtain the best price/performance from an application, the appropriate Oracle products must be determined. This is achieved by understanding application characteristics, picking Oracle products suitable to application characteristics, and conducting a series of tests before finalizing the choice in machines. Pilot tests were performed to characterize the performance of Web, application, and database applications across the current Oracle product line. Hardware was chosen based on price/performance rather than pure performance, since many organizations are simply not satisfied with a fast system — they want it to be fast and cheap at the same time. Figures 2 and 3 illustrate the hardware configurations used during the testing effort.

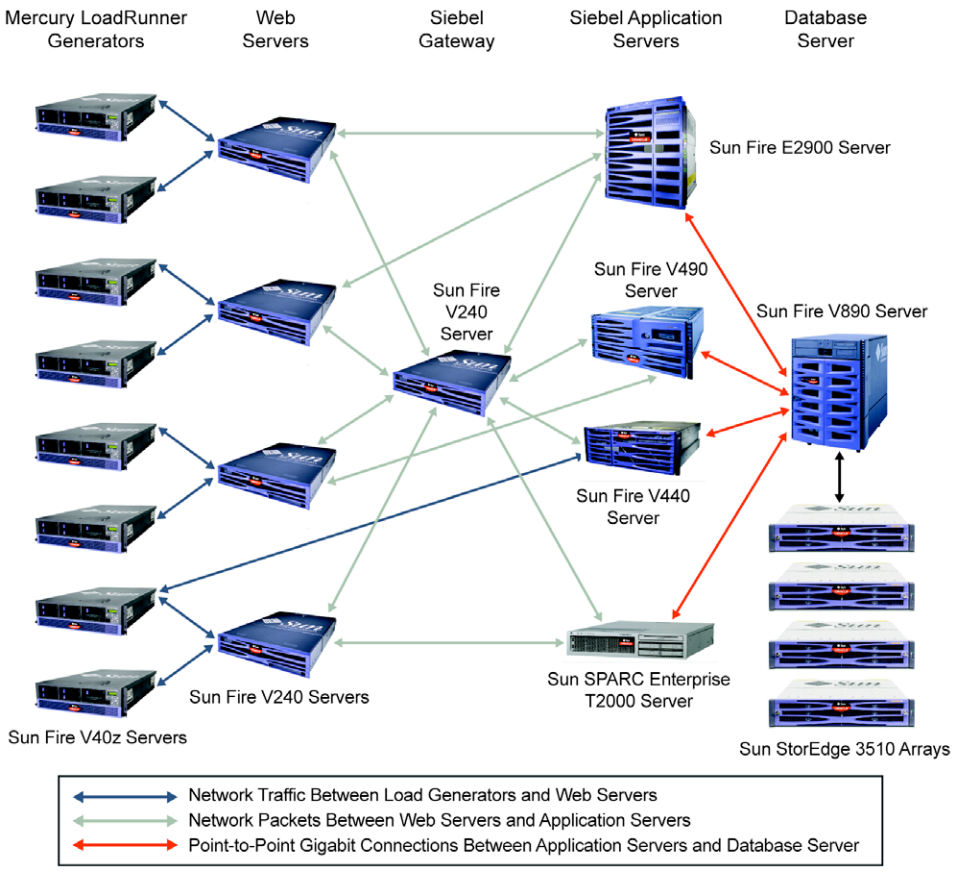

Figure 2. Topology diagram for 8,000 Oracle Siebel users.

The hardware and network topologies depicted in Figures 2 and 3 were designed applying the detailed knowledge of Siebel's performance characteristics. The Siebel end user online transaction processing (OLTP) workload was distributed across three nodes, including a Sun SPARC Enterprise T2000 server, Sun Fire v490 server, and Sun Fire E2900 server from Oracle. Each node ran a mix of Siebel Call Center and Siebel Partner Relationship management. The fourth Siebel node, a Sun Fire v440 server, was dedicated to the EAI-HTTP module. Each system had three network interface cards (NICs), which were used to isolate the main categories of network traffic. The exception was the Sun SPARC Enterprise T2000 server, where on-board Gigabit Ethernet ports were used.

- End user (load generator) to Web server traffic
- Web server to gateway to Siebel applications server traffic
- Siebel application server to database server traffic

The networking was designed using a Cisco Catalyst 4000 router. Two virtual LANS (VLANs) were created to separate network traffic between end user, Web server, gateway, and Siebel application traffic, while Siebel application server to database server traffic was further optimized with individual

point-to-point network interfaces from each application server to the database. This was done to alleviate network bottlenecks at any tier as a result of simulating thousands of Siebel users. The load generators were all Sun Fire V65 servers running Mercury LoadRunner software. The load was spread across three Sun Fire V240 Web servers by directing different kinds of users to the three Web servers.

All Siebel application servers belonged to a single Siebel Enterprise. A single Sun Fire v890+ server hosted the Oracle Database, and was connected to a Sun StorEdge 3510 FC array via a fibre channel host adapter.

The Oracle Database and the Sun SPARC Enterprise T2000 server running the Siebel application ran on Oracle Solaris 10. All other systems ran the Oracle Solaris 9 operating system.

A second testing configuration was built for 12,500 users (Figure 3). The key differences between it and the 8,000 user configuration were that the 12,500 user setup used additional application servers, and had a Sun Fire V890+ server running the Siebel software.

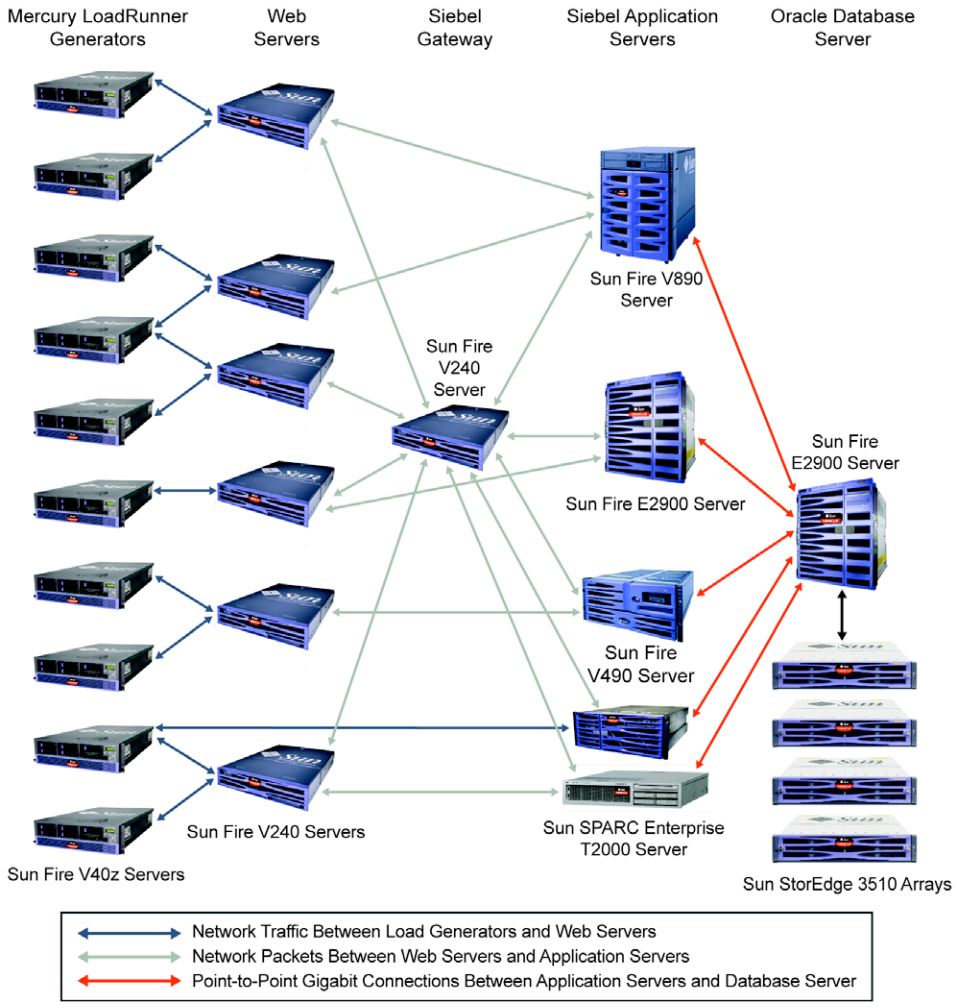

Figure 3. Topology diagram for 12,500 concurrent Siebel users.

Hardware and Software Users

Table 1 summarizes the hardware and software used during benchmarking efforts.

TABLE 1. HARDWARE AND SOFTWARE CONFIGURATION

COMPONENT	SERVER	CONFIGURATION
Gateway Server	Sun Fire V240 Server (1)	• 1.35 GHz UltraSPARC IIIi processors (2), 8 GB RAM • Oracle Solaris 9 OS, Generic • Siebel 7.7 Gateway Server
Application Servers	Sun Fire E2900 Server (1)	• 1.35 GHz UltraSPARC IV processors (12), 48 GB RAM • Oracle Solaris 9 OS, Generic • Siebel 7.7
	Sun SPARC Enterprise T2000 Server (1)	• 1.2 GHz UltraSPARC T1 processor (1), 32 GB RAM • Oracle Solaris 10 OS, Generic • Siebel 7.7
	Sun Fire V490 Server (1)	• 1.35 GHz UltraSPARC IV processors (4), 16 GB RAM • Oracle Solaris 9 OS, Generic • Siebel 7.7
Database Server	Sun Fire V890+ Server	• 1.5 GHz UltraSPARC IV+ processors (8), 32 GB RAM • Oracle Solaris 10 OS, Generic • Oracle 9.2.0.6, 64-bit • Sun StorEdge 3510 FC Array with 4 trays of twelve 73 GB disks running at 15K RPM
Mercury LoadRunner Drivers	Sun Fire V65x Servers (8)	• 3.02 GHz Xeon processors (4), 3 GB RAM • Microsoft Windows XP, SP1 • Mercury LoadRunner 7.8
Web Servers	Sun Fire V240 Servers (4)	• 1.5 GHz UltraSPARC IIIi processors (2), 8 GB RAM • Oracle Solaris 9 OS, Generic • Oracle iPlanet Web Server 6.1, SP4 • Siebel 7.7 SWSE
EAI Server	Sun Fire V440 Server (1)	• 1.2 GHz UltraSPARC IIIi processors (8), 16 GB RAM • Oracle Solaris 9 OS, Generic • Siebel 7.7

Workload Description

All of the tuning discussed in this document is specific to the PSPP workload as defined by Oracle. The workload was based on scenarios derived from large Siebel customers, reflecting some of the most frequently used and most critical components of the Oracle eBusiness Application Suite. At a high level, the workload for these tests can be grouped into two categories: online transaction processing and batch workloads.

Online Transaction Processing

The OLTP workload simulated the real world requirements of a large organization with 8,000 concurrent Siebel Web thin client end users involved in the following tasks and functions in a mixed ratio.

- **Siebel Financial Services Call Center.** The Siebel Financial Service Call Center application is used by over 6,400 concurrent sales and service representative users. The software provides the most complete solution for sales and service, enabling customer service and tele-sales representatives to provide world-class customer support, improve customer loyalty, and increase revenues through cross-selling and up-selling opportunities.
- **Siebel Partner Relationship Management.** The Siebel Partner Relationship Management application is used by over 1,600 concurrent eChannel users in partner organizations. The software enables organizations to effectively and strategically manage relationships with partners, distributors, resellers, agents, brokers, and dealers.

All end users were simulated using Mercury LoadRunner version 7.8 SP1, with a think time in the range of five to 55 seconds, or an average of 30 seconds, between user operations.

Batch Server Components

The batch component of the workload consisted of Siebel EAI HTTP Adapter.

- **Workflow.** This business process management engine automates user interaction, business processes, and integration. A graphical drag-and-drop user interfaces allows simple administration and customization. Administrators can add custom or pre-defined business services, specify logical branching, updates, inserts, and subprocesses to create a workflow process tailored to unique business requirements.
- **Enterprise Application Integration.** Enterprise Application Integration allows organizations to integrate existing applications with Siebel. EAI supports several adapters.

All tests were conducted by making sure that both the OLTP and batch components were run in conjunction for a one hour period (steady state) within the same Siebel Enterprise installation.

Business Transactions

Several complex business transactions were executed simultaneously by concurrent users. Between each user operation, the think time averaged approximately 15 seconds. This section provides a high-level description of the use cases tested.

Siebel Financial Services Call Center — Create and Assign Service Requests

- Service agent searches for contact
- Service agent checks entitlements
- Service request is created
- Service agent populates the service request with appropriate detail
- Service agent creates an activity plan to resolve the issue
- Using Siebel Script, the service request is automatically assigned to the appropriate representative to address the issue

Siebel Partner Relationship Management — Sales and Service

- Partner creates a new service request with appropriate detail
- A service request is automatically assigned
- The saving service request invokes scripting that brings the user to the appropriate opportunity screen
- A new opportunity with detail is created and saved
- The saving opportunity invokes scripting that brings the user back to the service request screen

Siebel Enterprise Application Integration — Integrate Third-Party Applications

EAI requests are made with a customized account-integration object. The requests consist of 80 percent selects, 10 percent updates, and 10 percent inserts.

The use cases are typically considered heavy transactions. For example, the high-level description of the sequential steps for the Create and Assign Service Requests use case follows:

1. Enable Siebel Search Center.
2. Search for a contact.
3. Review contact detail, and create a new service request.
4. Add details to the service request.
5. From the service request view, search for an account.
6. Select an account, and associate it with the service request.

7. Navigate to the Verify tab, and select entitlements.
8. Verify entitlements, and continue service request investigation.
9. Search for insurance group, and select the appropriate policy and product.
10. Create a new contact, entering information into all the fields in the list view.
11. Complete service request details, and save the service request.
12. Select the activity plan option, and automatically generate an activity plan for the service request.
13. Scripting automatically assigns the service request.
14. Summarize the service request with the customer.

Test Results Summary

The following sections describe the results of the testing effort.

8,000 Concurrent Users Test Results

The test system demonstrated that the Siebel 7.7 Architecture on Sun Fire servers and Oracle Database 9i easily scales to 8,000 concurrent users.

- **Vertical scalability.** The Siebel 7.7 Server showed excellent scalability within an application server.
- **Horizontal scalability.** The benchmark demonstrates scalability across multiple servers.
- **Low network utilization.** The Siebel 7.7 Smart Web Architecture and Smart Network Architecture efficiently managed the network, consuming only 5.5 Kbps per user.
- **Efficient use of the database server.** The Siebel 7.7 Smart Database Connection Pooling and Multiplexing enabled the database to service 8,000 concurrent users and supporting Siebel 7 Server application services with 800 database connections.

The actual results of the performance and scalability tests conducted at the Sun ETC for the Siebel workload are summarized below. "Performance Tuning" presents specific tuning tips and the methodology used to achieve this level of performance.

Response Times and Transaction Throughput

Table 2 and Table 3 summarize the test components and workload performance results.

TABLE 2. TEST COMPONENTS.

TEST COMPONENT	SOFTWARE	VERSION	HARDWARE	OPERATING SYSTEM
Database Server	Oracle	9.2.0.6, 64-bit	Sun Fire V890+ Server	Oracle Solaris 10 OS Generic_118822-11
Application Server	Siebel 7.7.1	Build 18306	Sun SPARC Enterprise T2000 Server	Oracle Solaris 10 OS, Generic
			Sun Fire E2900 Server	Oracle Solaris 10 OS, Generic
			Sun Fire V890 Server	Oracle Solaris 9 OS, Generic
			Sun Fire V490 Server	Oracle Solaris 9 OS, Generic
Web Server	Siebel 7.7.1 Oracle iPlanet Web Server	-	Sun Fire V240 Server	Oracle Solaris 9 OS, Generic
EAI Application Server	Siebel 7.7.1 Oracle iPlanet Web Server	-	Sun Fire V440 Server	Oracle Solaris 9 OS, Generic

TABLE 3. WORKLOAD RESULTS.

WORKLOAD	USERS	AVERAGE OPERATION RESPONSE TIME (SECONDS)[a]	BUSINESS TRANSACTIONS (THROUGHPUT PER HOUR)[b]
Siebel Financial Services Call Center	6,400	0.14	58,477
Siebel Partner Relationship Management	1,600	0.33	41,149
Siebel Enterprise Application Integration	N/A	N/A	514,800
Totals	8,000	N/A	614,426

a. Response times are measured at the Web server instead of the end user. The response times at the end user can depend on network latency, the bandwidth between the Web server and browser, and the time for the browser to render content.

b. A business transaction is defined as a set of steps, activities, and application interactions used to complete a business process, such as Create and assign Service Requests. Searching for a customer is an example of a step in the business transaction. For a detailed description of business transactions, see “Business Transactions”.

Server Resource Utilization

Table 4 summarizes the resource utilization statistics of the servers under test.

TABLE 4. SERVER RESOURCE UTILIZATION STATISTICS.

NODE	USERS	FUNCTIONAL USE	CPU UTILIZATION (PERCENT)	MEMORY UTILIZATION (GB)
Sun Fire E2900 Server (1)	4,100	Application Server 3,280 PSPP1, 820 PSPP2 Users	73	26.3
Sun SPARC Enterprise T2000 Server (1)	2,150	Application Server 1,720 PSPP1, 1,430 PSPP2 Users	85	15.4
Sun Fire V490 Server (1)	1,750	Application Server 1,400 PSPP1, 1,350 PSPP2 Users	83	11.8
Sun Fire V240 Servers (4)	8,000	Web Server	64	2.1
Sun Fire V440 Server (1)	—	Application Server — EAI	6	2.5
Sun Fire V240 Server (1)	8,000	Gateway Server	1	0.5
Sun Fire V890+ Server (1)	8,000	Database Server	29	20.8

12,500 Concurrent Users Test Results

The test system demonstrated that the Siebel 7.7 Architecture on Sun Fire servers and Oracle Database 9i easily scales to 12,500 concurrent users.

- **Vertical scalability.** The Siebel 7.7 Server showed excellent scalability within an application server.
- **Horizontal scalability.** The benchmark demonstrates scalability across multiple servers.
- **Low network utilization.** The Siebel 7.7 Smart Web Architecture and Smart Network Architecture efficiently managed the network consuming only 5.5 Kbps per user.
- **Efficient use of the database server.** The Siebel 7.7 Smart Database Connection Pooling and Multiplexing enabled the database to service 12,500 concurrent users and supporting Siebel 7 Server application services with 1,250 database connections.

The actual results of the performance and scalability tests conducted at the Sun ETC for the Siebel workload are summarized below. "Performance Tuning" presents specific tuning tips and the methodology used to achieve this level of performance.

Response Times and Transaction Throughput

Table 5 and Table 6 ummarize the test components and workload performance results.

TABLE 5. TEST COMPONENTS.

TEST COMPONENT	SOFTWARE	VERSION	HARDWARE	OPERATING SYSTEM
Database Server	Oracle	9.2.0.6, 64-bit	Sun Fire V2900 Server	Oracle Solaris 10 OS Generic_118822-11
Application Server	Siebel 7.7.1	Build 18306	Sun SPARC Enterprise T2000 Server	Oracle Solaris 10 OS, Generic
			Sun Fire E2900 Server	Oracle Solaris 10 OS, Generic
			Sun Fire V890 Server	Oracle Solaris 9 OS, Generic
			Sun Fire V490 Server	Oracle Solaris 9 OS, Generic
Web Server	Siebel 7.7.1 Oracle iPlanet Web Server	-	Sun Fire V240 Server	Oracle Solaris 9 OS, Generic
EAI Application Server	Siebel 7.7.1 Oracle iPlanet Web Server	-	Sun Fire V440 Server	Oracle Solaris 9 OS, Generic

TABLE 6. WORKLOAD RESULTS.

WORKLOAD	USERS	AVERAGE OPERATION RESPONSE TIME (SECONDS)[a]	BUSINESS TRANSACTIONS (THROUGHPUT PER HOUR)[b]
Siebel Financial Services Call Center	10,000	0.25	90,566
Siebel Partner Relationship Management	2,500	0.61	63,532
Siebel Enterprise Application Integration	N/A	N/A	683,314
Totals	12,500	N/A	837,412

a. Response times are measured at the Web server instead of the end user. Response times at the end user can depend on network latency, the bandwidth between the Web server and browser, and the time for the browser to render content.

b. A business transaction is defined as a set of steps, activities, and application interactions used to complete a business process, such as Create and assign Service Requests. Searching for a customer is an example of a step in the business transaction. For a detailed description of business transactions, see “Business Transactions”.

Server Resource Utilization

Table 7 summarizes the resource utilization statistics of the servers under test.

TABLE 7. SERVER RESOURCE UTILIZATION STATISTICS.

NODE	USERS	FUNCTIONAL USE	CPU UTILIZATION (PERCENT)	MEMORY UTILIZATION (GB)
Sun Fire E2900 Server (1)	4,100	Application Server 3,280 PSPP1, 820 PSPP2 Users	72	24.5
Sun SPARC Enterprise T2000 Server (1)	2,150	Application Server 1,720 PSPP1, 1,430 PSPP2 Users	84	12.8
Sun Fire V890+ Server (1)	4,500	Application Server 3,600 PSPP1, 1,900 PSPP2 Users	76	25.2
Sun Fire V490 Server (1)	1,750	Application Server 1,400 PSPP1, 1,350 PSPP2 Users	83	10.6
Sun Fire V240 Servers (6)	12,500	Web Server	68	2.3
Sun Fire V440 Server (1)	—	Application Server — EAI	6	2.2
Sun Fire V240 Server (1)	12,500	Gateway Server	1	0.5
Sun Fire E2900 Server (1)	12,500	Database Server	58	28.6

Siebel Scalability on Oracle Hardware Platforms

Oracle's Siebel application scales to a large number of concurrent users on Oracle platforms. Due to Siebel's flexible, distributed architecture and Oracle's varied server product line, an optimal deployment can be achieved either with several small systems with one to four CPUS, or with a single large server such as the Sun Fire E25K server or Sun Fire E6900 server. The graphs below illustrate Siebel scalability on different Oracle system types used in this experiment. All the tuning used to achieve these results can be applied to production systems, and is documented in the next section.

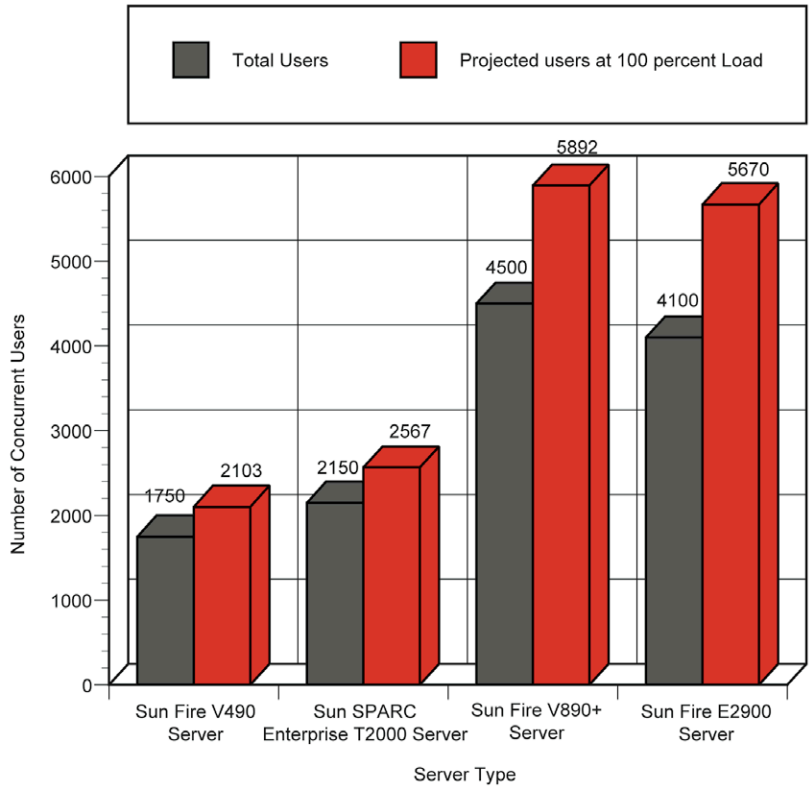

Figure 4. Siebel 12,500 user benchmark results.

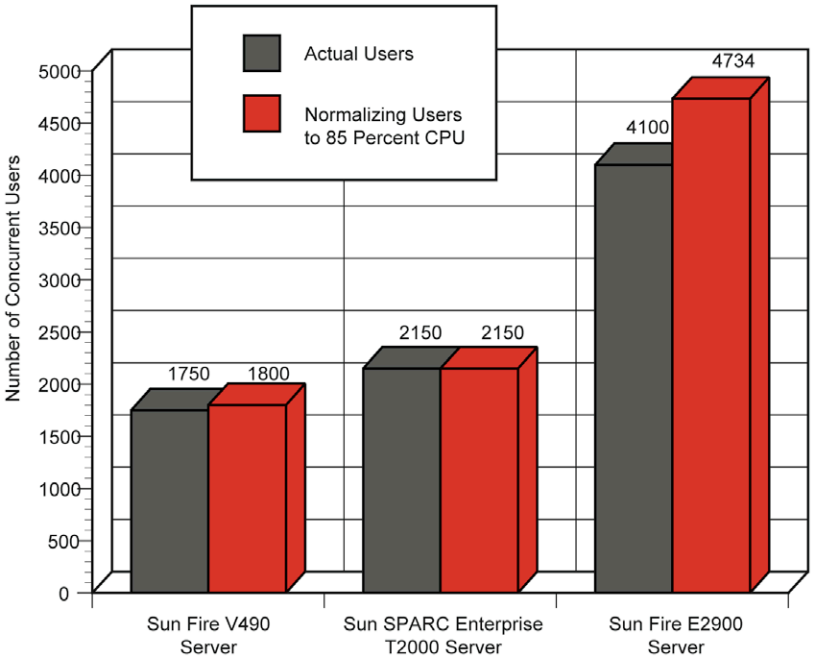

Figure 5. Siebel 7.7.1 8,000 user benchmark results.

Figure 6 shows the number of users per CPU on various application server machines at 100% load.

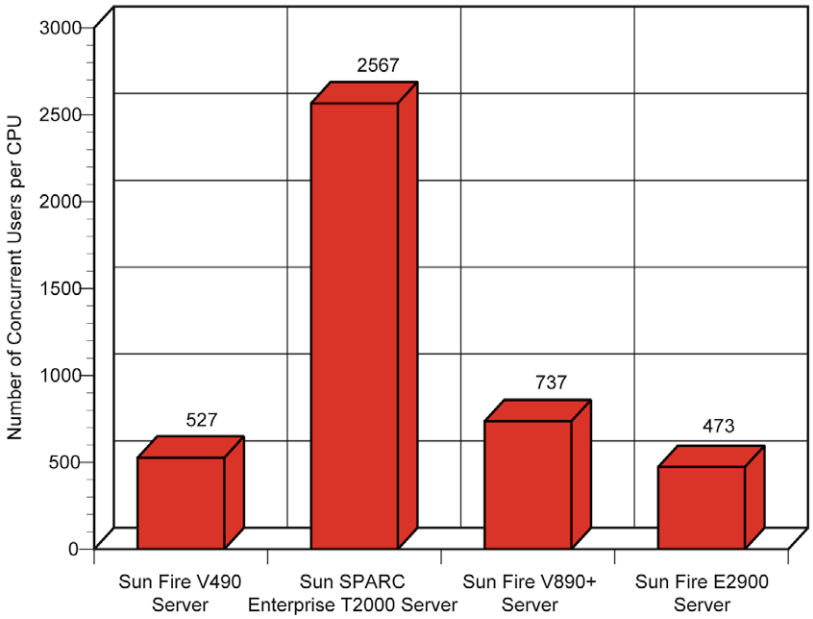

Figure 6. Siebel 7.7.1 12,500 user benchmark results showing the number of CPUs per user at 100 percent load.

The Sun SPARC Enterprise T2000 server incorporates UltraSPARC T1 processors, while the Sun Fire V490 and Sun Fire E2900 servers use UltraSPARC IV processors. Figure 7 shows the difference in scalability between Oracle systems. Customers can use the data presented here for capacity planning and sizing of real world deployments. It is important to keep in mind that the workload used for these results should be identical to the real world deployment. If the workloads are not identical, appropriate adjustments must be made to the sizing of the environment.

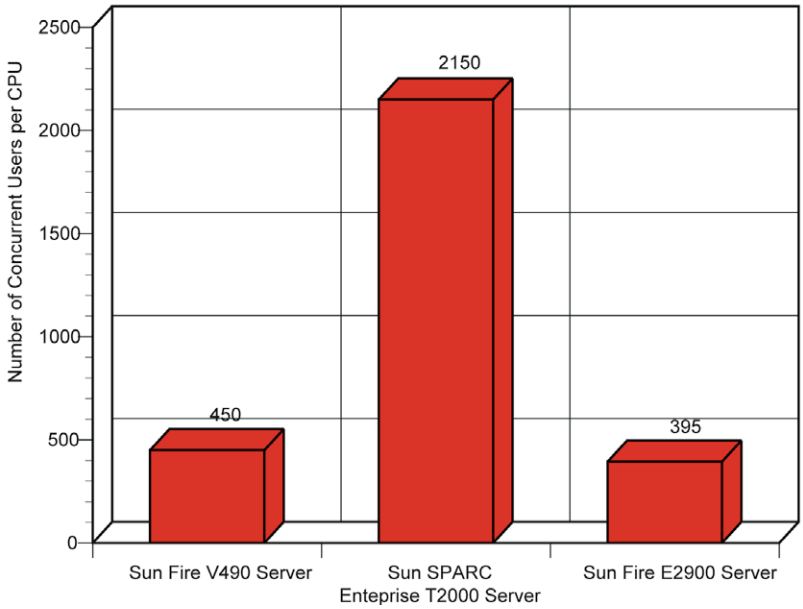

Figure 7. Siebel 7.7.1 8,000 user benchmark results showing the number of users per CPU at 85 percent load.

Figure 8 and Figure 9 illustrate the cost per user on Oracle systems[1]. Oracle servers provide the optimal price/performance for Siebel applications running in UNIX® environments. These graphs describe the cost per user on various models of tested Oracle servers.

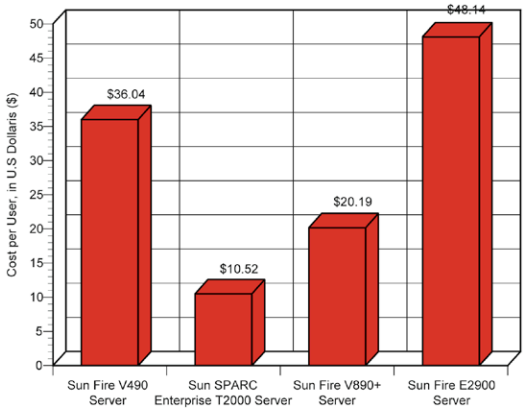

Figure 8. Siebel 12,500 user benchmark results, showing the cost per user.

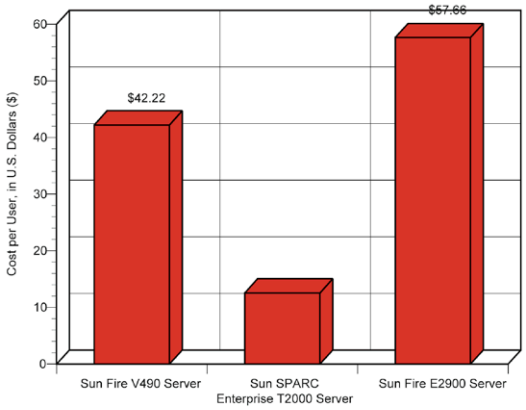

Figure 9. Siebel 8,000 user benchmark, showing the cost per user.

Table 8 summarizes the price/performance per tier of the deployment.

TABLE 8. PRICE/PERFORMANCE PER TIER.

TIER	SERVER	USERS PER CPU	COST PER CPU
Application	Sun SPARC Enterprise T2000 Server	2,150	$12.56
	Sun Fire V490 Server	450	$42.22
	Sun Fire E2900 Server	395	$57.66
	Sun Fire V890+ Server	737	$20.19
Database	Sun Fire V890+ Server	2,851	$5.22
Web	Sun Fire V240 Server	1,438	$3.93
Average Response Time	From 0.126 seconds to 0.303 seconds (component specific)		
Success Rate	> 99.999 percent (8 failures out of approximately 1.2 million transactions		

[1] $/user is based purely on hardware cost and does not include environmental, facility, service or management costs.

Oracle's Sun Blade Server Architecture for Oracle's Siebel Applications

While this paper investigates the use of Sun Fire servers for an Oracle deployment, Oracle's Sun Blade 6000 Modular System with Sun Blade T6300 Server Modules offers an additional choice of server based on the UltraSPARC T1 processor with CoolThreads technology. By utilizing a blade server architecture that blends the enterprise availability and management features of vertically scalable platforms with the scalability and economic advantages of horizontally scalable platforms, application tier servers in an Oracle implementation can harness more compute power, expand or scale services faster, and increase serviceability and availability while reducing complexity and cost.

Sun Blade T6300 Server Module

Following the success of Oracle's Sun SPARC Enterprise T1000 and T2000 servers, Oracle's Sun Blade T6300 Server Module brings chip multithreading to a modular system platform. With a single socket for a six or eight core UltraSPARC T1 processor, up to 32 threads can be supported for applications requiring substantial amounts of throughput. Similar to the Sun SPARC Enterprise T2000 server, the server module uses all four of the processor's memory controllers, providing large memory bandwidth. Up to eight DDR2 667 MHz DIMMs can be installed for a maximum of 32 GB of RAM per server module.

Sun Blade 6000 Modular System

The Sun Blade 6000 Modular System is a 10 RU chassis supporting up to ten full performance and full featured blade modules. Using this technology, enterprises can deploy multi-tier applications on a single unified modular architecture and consolidate power and cooling infrastructure for multiple systems into a single chassis. The result — more performance and functionality in a small footprint.

Figure 10 illustrates how the architecture can be simplified using Sun Blade technology in the Web and application tiers. If a fully configured Sun Blade 6000 Modular System is used, one blade server module can act as the gateway, while the remaining nine blade server modules can each run Web and application server. By enabling the consolidation of the Web and application tiers, the architecture is expected to handle up to 20,000 concurrent Siebel application users and provide faster throughput while reducing power and space requirements. Note that the architecture was not tested using Sun Blade systems. The expected results are projections based on the similarity in processor and compute power of the Sun Blade T6300 Server Module and the Sun SPARC Enterprise T2000 server. Actual testing was performed on Sun SPARC Enterprise T2000 servers.

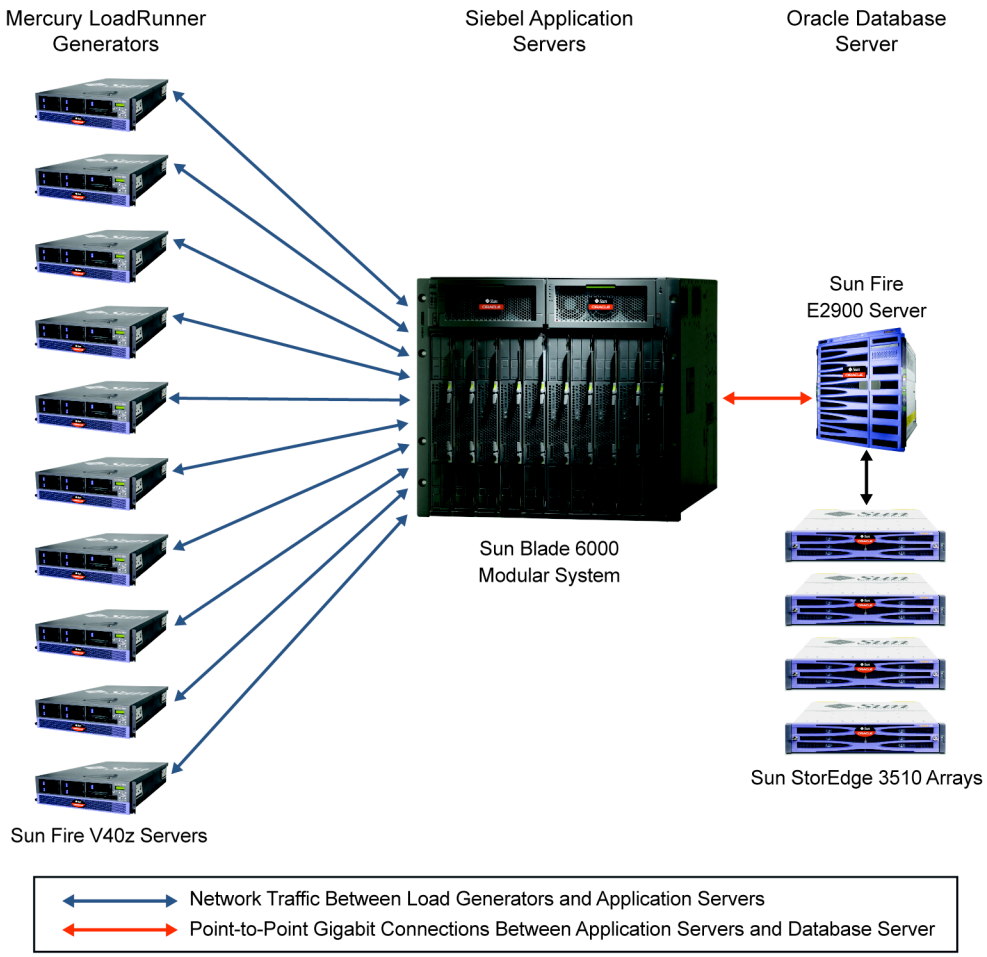

Figure 10. Use of the Sun Blade 6000 Modular System enables as many as 20,000 users to be supported.

Performance Tuning

The following sections provide detail on performance tuning opportunities.

Tuning Oracle Solaris OS for Siebel Server

Information on tuning the Oracle Solaris OS for Siebel Server can be found below.

Oracle Solaris Tuning for Siebel using libumem

`Libumem` is a standard feature in Oracle Solaris OS as of Oracle Solaris 9 Update 3. `Libumem` is an alternate memory allocator module built specifically for multithreaded applications such as Siebel running on Oracle's symmetric multiprocessing (SMP) systems. The `libumem` routines provide a faster, concurrent malloc implementation for multithreaded applications. Observations based on tests of 400 Siebel users, with and without using `libumem`, follow.

- With `libumem`, results indicated a 4.65 percent improvement in CPU utilization.
- Use of `libumem` results in approximately an 8.65 percent increase in per user memory footprint.
- Without `libumem`, Siebel threads spend more time waiting, as evidenced by an increase in the number of calls to `lwp_park`.

The use of `libumem` resulted in lower CPU consumption, although memory consumption increased as a side effect. However, the overall price/performance benefits are positive. The benefits of `libumem` increase on SMP systems with more CPUs. `Libumem` provides faster and more efficient memory allocation by using an object caching mechanism. Object caching is a strategy in which memory that is frequently allocated and freed is cached, so the overhead of creating the same data structure is reduced considerably. In addition, per CPU sets of caches, called Magazines, improve `libumem` scalability by enabling it to have a less contentious locking scheme when requesting memory from the system. The object caching strategy enables applications to run faster with lower lock contention among multiple threads.

`Libumem` is a page-based memory allocator. Therefore, if a request is made to allocate 20 bytes of memory, `libumem` aligns it to the nearest page (at 24 bytes on the SPARC platform) and returns a pointer to the allocated block. (The default page size is 8K on SPARC systems running Oracle Solaris.) Continuing requests can lead to internal fragmentation, resulting in extra memory allocated by `libumem` but not requested by an application to be wasted. In addition, libumem uses eight bytes of every buffer it creates to store buffer metadata. Due to the reasons outlined here, a slight increase in the per process memory footprint may result. How is `libumem` enabled?

1. Edit the *$SIEBEL_ROOT/bin/siebmtshw* file.
2. Add the line `LD_PRELOAD=/usr/lib/libumem.so.1` to the file.
3. Save the file and bounce the Siebel servers.
4. After Siebel restarts, verify `libumem` is enabled by executing the `pldd` command.

```
$%pldd -p <pidofsiebmtshmw> | grep -i libumem
```

Multiple Page Size Support Tuning for Siebel

A standard feature available as of Oracle Solaris 9 gives applications the ability to run with more than one page size on the same operating system. Using this Oracle Solaris library, a larger heap size can be set for certain applications, resulting in improved performance due to reduced TLB miss rates. Multiple Page Size Support (MPSS) improves virtual memory performance by enabling applications to use large page sizes, improving resource efficiency and reducing overhead. These benefits are all accomplished without recompiling or re-coding applications.

Enabling MPSS for Siebel processes provides performance benefits. Using a 4 MB page for the heap, and a 64 KB page size for the stack of a Siebel server process resulted in approximately 10 percent CPU reduction during benchmark tests. Further improvements are found in Oracle Solaris 10 OS 1/06 and later releases — text, heap, anon, and stack are automatically placed on large pages. There is no

need to tune or enable this feature, providing another reason for organizations to upgrade to Oracle Solaris 10. However, earlier Oracle Solaris versions require MPSS to be enabled manually. The benefit of enabling MPSS can be verified by running the following command on a system running Siebel server. Users may notice a decrease in TLB misses from the output of this command when MPSS is enabled.

```
trapstat -T 10 10
```

The steps needed to enable MPSS for Siebel processes follow. It is important to note these steps are only required for operating system versions prior to the Oracle Solaris 10 OS 1/06 release. Since MPSS is enabled on Oracle Solaris 10 by default, simply run the `pmap -xs pid` command to see the page size used by an application.

1. Enable kernel cage if the system is not a Sun Enterprise E10K or Sun Fire 15K server, and reboot the system. Kernel cage can be enabled by the following setting in the */etc/system* file.

   ```
   set kernel_cage_enable=1
   ```

   Why is kernel cage needed? Kernel cage helps reduce memory fragmentation. When a system is in use for several months, memory fragments due to a large number of calls to allocate and free memory. If MPSS is enabled at this stage, the application may not be able to get the required large pages. Immediately after a system boot, a sizeable pool of large pages is available, and applications can get all of their `mmap()` memory allocated from large pages. This can be verified using the `pmap -xs pid` command. Enabling kernel cage can vastly minimize fragmentation. The kernel is allocated from a small contiguous range of memory, minimizing the fragmentation of other pages within the system.

2. Find all possible hardware address translation (HAT) sizes supported by the system with the `pagesize -a` command.

   ```
   $ pagesize -a
   8192
   65536
   524288
   4194304
   ```

3. Run the `trapstat -T` command. The value shown in the `ttl` row and `%time` column is the percentage of time spent in virtual to physical memory address translations by the processor(s). Depending on the `%time` value, choose an appropriate page size that reduces the iTLB and dTLB miss rate.

4. Create a file containing the following line. The file can have any name. The `desirable_heap_size` and `desireable_stack_size` size must be a supported HAT size. The default page size for heap and stack is 8 KB on all Oracle Solaris releases.

```
siebmts*:desirable_heap_size:desirable_stack_size
```

5. Set the `MPSSCFGFILE` and `MPSSERRFILE` environment variables. `MPSSCFGFILE` should point to the configuration file created in step 4. MPSS writes any errors during runtime to the `$MPSSERRFILE` file.

6. Preload MPSS interposing the `mpss.so.1` library, and start the Siebel server. It is recommended to put the `MPSSCFGFILE`, `MPSSERRFILE`, and `LD_PRELOAD` environment variables in the Siebel server startup script. To enable MPSS for the Siebel application, edit the *$SIEBEL_ROOT/bin/siebmtshw* file and add the following lines.

```
#!/bin/ksh
. $MWHOME/setmwruntime
MWUSER_DIRECTORY=${MWHOME}/system
MPSSCFGFILE=/tmp/mpss.cfg
MPSSERRFILE=/tmp/mpss.err
LD_PRELOAD=mpss.so.1
export MPSSCFGFILE MPSSERRFILE LD_PRELOAD
LD_LIBRARY_PATH=/usr/lib/lwp:${LD_LIBRARY_PATH}
exec siebmtshmw $@
$ cat /tmp/mpsscfg
siebmtsh*:4M:64K
```

7. Go back to step 3 and measure the difference in `%time`. Repeat steps 4 through 7 using different page sizes, until noticeable performance improvement is achieved.

More details on MPSS can be found in the `mpss.so.1` man page, as well as the *Supporting Multiple Page Sizes in the Solaris Operating System* white paper located at *http://www.solarisinternals.com/si/reading/817-5917.pdf*

Oracle Solaris Kernel and TCP/IP Tuning Parameters for Siebel Server

The System V IPC resource limits in Oracle Solaris 10 are no longer set in the */etc/system* file, but instead are project resource controls. As a result, a system reboot is no longer required to put changes to these parameters into effect. This also lets system administrators set different values for different projects. A number of System V IPC parameters are obsolete with Oracle Solaris 10, simply because they are no longer necessary. The remaining parameters have reasonable defaults to enable more applications to work out-of-the-box, without requiring the parameters to be set.

For Oracle Solaris 9 and earlier versions, kernel tunables should be set in the */etc/system* file. By upgrading to Oracle Solaris 10, organizations can take advantage of the ease of changing tunables on the fly.

During benchmark testing, certain network parameters were set specifically for the Sun SPARC Enterprise T2000 server to ensure optimal performance. The parameter settings are listed below.

Sun SPARC Enterprise T2000 server `ipge` settings:

```
set ip:ip_squeue_bind = 0
set ip:ip_squeue_fanout = 1
set ipge:ipge_tx_syncq=1
set ipge:ipge_taskq_disable = 0
set ipge:ipge_tx_ring_size = 2048
set ipge:ipge_srv_fifo_depth = 2048
set ipge:ipge_bcopy_thresh = 512
set ipge:ipge_dvma_thresh = 1
set segkmem_lpsize=0x400000
set consistent_coloring=2
set pcie:pcie_aer_ce_mask=0x1
```

TABLE 9. GENERIC SETTINGS FOR ALL SERVERS.

PARAMETER	SCOPE	DEFAULT VALUE	TUNED VALUE
shmsys:shminfo_shmmax	/etc/system		0xffffffffffffffff
shmsys:shminfo_shmmin	/etc/system		100
shmsys:shminfo_shmseg	/etc/system		200
shmsys:shminfo_semmns	/etc/system		12092
shmsys:shminfo_semmsl	/etc/system		512
shmsys:shminfo_semmni	/etc/system		4096
shmsys:shminfo_semmap	/etc/system		4096
shmsys:shminfo_semmnu	/etc/system		4096
shmsys:shminfo_semopm	/etc/system		100
shmsys:shminfo_semume	/etc/system		2048
msgsys:msginfo_msgmni	/etc/system		2048

msgsys:msginfo_msgtql	/etc/system		2048
msgsys:msginfo_msgssz	/etc/system		64
msgsys:msginfo_msgseg	/etc/system		32767
msgsys:msginfo_msgmax	/etc/system		16384
msgsys:msginfo_msgmnb	/etc/system		16384
lp:dohwcksum (for resonate gbic)	/etc/system		0
rlim_fd_max	/etc/system	1024	16384
rlim_fd_cur	/etc/system	64	16384
sq_max_size	/etc/system	2	0
tcp_time_wait_interval	ndd /dev/tcp	240000	60000
tcp_conn_req_max_q	ndd /dev/tcp	128	1024
tcp_conn_req_max_q0	ndd /dev/tcp	1024	4096
tcp_ip_abort_interval	ndd /dev/tcp	480000	60000
tcp_keepalive_interval	ndd /dev/tcp	7200000	900000
tcp_rexmit_interval_initial	ndd /dev/tcp	3000	3000
tcp_rexmit_interval_max	ndd /dev/tcp	240000	10000
tcp_rexmit_interval_min	ndd /dev/tcp	200	3000
tcp_smallest_anon_port	ndd /dev/tcp	32768	1024
tcp_slow_start_initial	ndd /dev/tcp	1	2
tcp_xmit_hiwat	ndd /dev/tcp	8129	32768
tcp_fin_wait_2_flush_interval	ndd /dev/tcp	67500	675000
tcp_recv_hiwat	ndd /dev/tcp	8129	32768

Tuning Siebel Server for Oracle Solaris

The key factor in tuning Siebel server performance is number of threads or users per Siebel object manager (OM) process. The Siebel server architecture consists of multithreaded server processes servicing different business needs. Currently, Siebel is designed such that one thread of a Siebel OM

services one user session or task. The ratio of threads or users per process is configured using the following Siebel parameters:

- `MinMTServers`
- `MaxMTServers`
- `Maxtasks`

Results from several tests reveal that on the Oracle Solaris platform with Siebel 7.7 the following users/OM ratios provided optimal performance.

- Call Center: 100 users/OM
- eChannel: 100 users/OM

The optimal ratio of threads/process varies with the Siebel OM and the type of Siebel workload per user. `Maxtasks` divided by `MaxMTServers` determines the number of users per process. For example, 300 users requires `MinMtServers=6`, `MaxMTServers=6`, and `Maxtasks=300`. These settings direct Siebel to distribute the users across the six processes evenly, with 50 users on each process. The notion of anonymous users must be factored into this calculation, and is discussed in the section on each type of workload. The prstat -v or top command shows how many threads or users are being serviced by a single multithreaded Siebel process.

```
   PID    USERNAME    SIZE    RSS   STATE   PRI  NICE      TIME     CPU    PROCESS/NLWP
  1880    pspp        504M   298M   cpu14   28     0     0:00.00     10%    siebmtshmw/69
  1868    pspp        461M   125M   sleep   58     0     0:00.00    2.5%    siebmtshmw/61
  1227    pspp        687M   516M   cpu3    22     0     0:00.03    1.6%    siebmtshmw/62
  1751    pspp        630M   447M   sleep   59     0     0:00.01    1.5%    siebmtshmw/59
  1789    pspp        594M   410M   sleep   38     0     0:00.02    1.4%    siebmtshmw/60
  1246    pspp        681M   509M   cpu20   38     0     0:00.03    1.2%    siebmtshmw/62
```

A thread count of more than 50 threads per process is due to the fact that the count also includes some administrative threads. If the `maxtasks/MaxMTServers` ratio is greater than 100, performance degrades and results in longer transaction response times. The optimal users per process setting depends on the workload.

Tuning the FINS Call Center and eChannel Siebel Modules

Call Center or `FINSObjMgr_enu` specific tuning and configuration information is presented below. A setting of 100 users/process was found to be optimal. It should be noted that the settings that follow are required to be `TRUE` for scenarios where these functions are required. The benchmark setup required these settings to be disabled.

1. Add the following to the `[SWE]` section of *$SIEBEL_ROOT/bin/enu/fins.cfg* (FINS).

```
EnableReportsFromToolbar = TRUE
EnableShuttle = TRUE
EnableSIDataLossWarning = TRUE
EnablePopupInlineQuery = TRUE
```

2. Set `EnableCDA = FALSE` to disable invoking CDA functionality.

To determine the `maxtasks`, `mtservers`, and `anonuserpool` settings, use the following example outlined in Table 10.

TABLE 10. PRICE/PERFORMANCE PER TIER

Target Number of Users	4000
Anonuserpool	400
Buffer	200
Maxtasks	4600
MaxMTserver=MinMtserver	58

If the Target Number of Users is 4,000, then `AnonUserPool` is 10 percent of 4000, or 400. All allow 5 percent for the buffer. Add all values to determine the `Maxtasks` setting (4000+400+200=4600). To run 80 users/process, set the `MaxMTServer` value to 4600/80=57.5 (rounded to 58).

The `AnonUserPool` value is set in the *eapps.cfg* file on the Siebel Web Server Engine. If the load is distributed across multiple Web server machines or instances, simply divide this number by the number in use. For example, if two Web servers are used, set `AnonUserPool` to 200 in each Web server's *eapps.cfg* file. A snapshot of the file and the settings for Call Center are displayed below.

```
[/callcenter]
AnonSessionTimeout  = 360
GuestSessionTimeout = 60
SessionTimeout      = 300
AnonUserPool  = 200
ConnectString = siebel.tcpip.none.none://19.1.1.18:2320/siebel/SCCObjMgr
```

Latches are similar to mutual exclusion objects (mutexes), which are used for communication between processes. Be sure to configure items properly using the formula shown below.

1. Set the following environment variables in the *$SIEBEL_HOME/siebsrvr/siebenv.csh* or *siebenv.sh* file.

```
export SIEBEL_ASSERT_MODE=0
export SIEBEL_OSD_NLATCH = 7 * Maxtasks + 1000
export SIEBEL_OSD_LATCH = 1.2 * Maxtasks
```

2. Set `Asserts OFF`, by setting the environment variable `SIEBEL_ASSERT_MODE=0`.
3. Restart the Siebel server after setting these environment variables.

The tuning mentioned in the FINS call center section applies to most of the Siebel object managers, including eChannel.

1. Add the following to the `[SWE]` section of the *$SIEBEL_ROOT/bin/enu/scw.cfg* file (eChannel).

```
EnableReportsFromToolbar = TRUE
EnableShuttle = TRUE
EnableSIDataLossWarning = TRUE
EnablePopupInlineQuery = TRUE
CompressHistoryCache = FALSE
```

2. Edit the following parameter.

```
EnableCDA = FALSE
```

Siebel Enterprise Applications Interface HTTP Adapter

The Siebel Enterprise Applications Interface HTTP (EAI-HTTP) Adapter Transport business service lets users send XML messages over HTTP to a target URL or Web site. The Siebel Web Engine serves as the transport to receive XML messages sent over the HTTP protocol to Siebel.

The Siebel EAI component group is enabled on one application server, and the HTTP driver is run on a Sun Fire v65 server running Mercury LoadRunner software on the Windows XP Professional operating system.

To achieve the result of 514,800 business transactions/hour, six threads were run concurrently, with each thread working on 30,000 records. The optimum values for the number of Siebel servers of type EAIObjMgr for this workload are listed below.

```
maxmtservers=2 minmtservers=2 maxtasks=10
```

The other Siebel OM parameters for EAIObjMgr should use default values. Changing `SISPERSISSCON` from the default setting of 20 does not help. All Oracle Database optimizations explained under that section in this paper helped achieve the throughput as measured in objects/second.

The HTTP driver system was utilized to the fullest, slowing performance. I/O was the cause — the HTTP test program generated approximately 3 GB of logs during the test. Once logging was turned off as shown below, performance improved. The `AnonSessionTimeout` and `SessionTimeout` values in the `SWSE` were set to 300.

```
driver.logresponse=false
driver.printheaders=false
```

HTTP adapter (`EAIObjMgr_enu`) tunable parameters include:

- `MaxTasks`, with a recommended value of 10
- `MaxMTServers`, `MinMTServers`, each with a recommended value of 2
- `UpperThreshold`, with a recommended value of 25
- `LoadBalanced`, with recommended value of true
- `Driver.Count`, with a recommended value of 4 (at client)

Making Siebel Use the Server to Maximum Capacity

Several items can help the server to be used to maximum capacity.

The Siebel MaxTasks Upper Limit Debunked

In order to configure Siebel server to run 8000 concurrent users, the Siebel maxtasks parameter must be set to 8800. This setting caused the Siebel object manager processes (Siebel servers) to fail to start with the error messages listed below. This was not due to a resource limitation in Oracle Solaris. Ample amounts of CPU, memory, and swap space were available on the system, a Sun Fire E2900 server with 12 CPUs and 48 GB RAM.

Siebel Enterprise server logged the following error messages and failed to start:

```
GenericLog      GenericError    1       2004-07-30 21:40:07  (sissrvr.cpp 47(2617)
err=2000026 sys=17) SVR-00026: Unable to allocate shared memory
GenericLog      GenericError    1       2004-07-30 21:40:07  (scfsis.cpp 5(57) err=2000026
sys=0) SVR-00026: Unable to allocate shared memory
GenericLog      GenericError    1       2004-07-30 21:40:07  (listener.cpp 21(157)
err=2000026 sys=0) SVR-00026: Unable to allocate shared memory
```

Explanation

This section provides background information on the Siebel `siebsvc` server process. The `siebsvc` process runs as a system service that monitors and controls the state of every Siebel server component operating on that Siebel server. Each Siebel server is an instantiation of the Siebel Server System Service (`siebsvc`) within the current Siebel Enterprise Server. The Siebel server runs as a daemon process in a UNIX environment.

During startup, the Siebel Server System Service performs the following sequential steps:

- Retrieves configuration information from the Siebel Gateway Name Server.
- Creates the *Enterprise_Server_Name.Siebel_Server_Name.shm* shared memory file located in the admin subdirectory of the Siebel Server root directory on UNIX.

The size of the **.shm* file is directly proportional to the maxtask setting. The higher the number of concurrent users, the higher the `maxtasks` setting. The Siebel Server System Service deletes the *.shm* file when it shuts down.

Investigating the Reason for the Failure

The *.shm* file created during startup is memory mapped and is a part of the heap size of the `siebsvc` process. As a result, the process size of the `siebsvc` process grows proportionally with the increase in the size of the *.shm* file.

With 9,500 `maxtasks` configured, a *.shm* file of size 1.15 GB was created during server startup.

```
%ls -l /export/siebsrvr/admin/siebel.sdcv480s002.shm
-rwx------ 1 sunperf other 1212096512 Jan 24 11:44 /export/siebsrvr/admin/siebel.sdcv480s002.shm
```

The `siebsvc` process died abruptly. The process size just before termination was measured to be 1.14 GB.

```
  PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
25309 sunperf  1174M 1169M sleep   60    0   0:00:06 6.5% siebsvc/1
```

A truss of the process reveals it was trying to `mmap` a 1.15 GB file and failed with status code `ENOMEM`. The truss output is listed below.

```
8150:   brk(0x5192BF78)                                 = 0
8150:   open("/export/siebsrvr/admin/siebel_siebapp6.shm", O_RDWR|O_CREAT|O_EXCL, 0700) = 9
8150:   write(9, "\0\0\0\0\0\0\0\0\0\0\0\0".., 1367736320) = 1367736320
8150:   mmap(0x00000000, 1367736320, PROT_READ|PROT_WRITE, MAP_SHARED, 9, 0) Err#12 ENOMEM
```

If the `mmap` operation succeeds, the process may have a process size greater than 2 GB. Siebel is a 32-bit application, and can have a process size up to 4 GB on Oracle Solaris. In this case, however, the process failed with a size slightly larger than 2 GB. The system resource settings on the failed system are detailed below.

```
%ulimit -a
time(seconds)        unlimited
file(blocks)         unlimited
data(kbytes)         unlimited
stack(kbytes)        8192
coredump(blocks)     unlimited
nofiles(descriptors) 256
vmemory(kbytes)      unlimited
```

Even though the maximum size of the datasize or heap is reported as unlimited, by the `ulimit` command, the maximum limit is 2 GB by default on Oracle Solaris. The upper limit as seen by an application running on Oracle Solaris platforms can be obtained by using a simple C program that calls the `getrlimit` API from the *sys/resource.h* file. The following program prints the system limits for data, stack, and virtual memory.

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <sys/time.h>
#include <sys/resource.h>
static void showlimit(int resource, char* str)
{
  struct rlimit lim;
  if (getrlimit(resource, &lim) != 0) {
     (void)printf("Couldn't retrieve %s limit\n", str);
     return;
  }
 (void)printf("Current/maximum %s limit is \t%lu / %lu\n", str, lim.rlim_cur, lim.rlim_max);
}
 int main() {
   showlimit(RLIMIT_DATA,  "data");
   showlimit(RLIMIT_STACK, "stack");
   showlimit(RLIMIT_VMEM,  "vmem");
   return 0;
}
```

Output from the C program on the failed system is listed below.

```
%showlimits
Current/maximum data  limit is   2147483647 / 2147483647
Current/maximum stack limit is      8388608 / 2147483647
Current/maximum vmem  limit is   2147483647 / 2147483647
```

Based on the above output, it is clear the processes were bound to a maximum data limit of 2 GB on a generic Oracle Solaris platform. This limitation is the reason for the failure of the `siebsvc` process as it was trying to grow beyond 2 GB in size.

Solution

The solution to the problem is to increase the default system limit for datasize and reduce the stacksize value. An increase in datasize creates more room for process address space, while lowering stacksize reduces the reserved stack space. These adjustments let a Siebel process use its process address space more efficiently, thereby allowing the total Siebel process size to grow to 4 GB, the upper limit for a 32-bit application.

- What are the recommended values for data and stack sizes on Oracle Solaris while running the Siebel application? How can the limits of datasize and stacksize be changed?

  Set `datasize` to 4 GB, the maximum allowed address space for a 32-bit process. Set `stacksize` to any value less than 1 MB, depending on the stack's usage during high load. In general, even with very high loads the stack may use 64 KB. Setting `stacksize` to 512 KB should not harm the application. System limits can be changed using the ulimit or limit user commands, depending on the shell. Example commands to change the limits are listed below.

```
ksh:
ulimit -s 512
ulimit -d 4194303

csh:
limit -stacksize 512
limit -datasize 4194393
```

- How should the commands above be executed?

  The commands can be executed from `ksh` or `csh` immediately prior to running Siebel. The Siebel processes inherit the limits during shell forking. However, the *$SIEBEL_ROOT/bin/start_server* script is the recommended place for the commands.

```
/export/home/sunperf/18306/siebsrvr/bin/%more start_server
...
USAGE="usage: start_server [-r <siebel root>] [-e <enterprise>] \
[-L <language code>] [-a] [-f] [-g <gateway:port>] { <server name> ... | ALL }"
ulimit -d 4194303
ulimit -s 512
##  set variables used in siebctl command below
```

- How can the system limits for a running process be checked?

  The `plimit pid` command prints the system limits as seen by the process. For example:

```
%plimit 11600
11600:  siebsvc -s siebsrvr -a -g sdcv480s002 -e siebel -s sdcv480s002
  resource              current         maximum
  time(seconds)         unlimited       unlimited
  file(blocks)          unlimited       unlimited
  data(kbytes)          4194303         4194303
  stack(kbytes)         512             512
  coredump(blocks)      unlimited       unlimited
  nofiles(descriptors)  65536           65536
  vmemory(kbytes)       unlimited       unlimited
```

The above setting lets the `siebsvc` process mmap the .shm file. As a result, the `siebsvc` process succeeds in forking the rest of the processes, and the Siebel server processes start up successfully. This finding enables the configuration of maxtasks to be greater than 9,000 on Oracle Solaris. While the scalability limit of 9,000 users per Siebel server was overcome, it can be raised to 18,000. If `maxtasks` is configured to be greater than 18,000, Siebel calculates the **.shm* file size to be approximately 2 GB. The `siebsvc` process tries to `mmap` the file and fails when it hits the limit for a 32-bit application process.

A listing of the Siebel *.shm* file with `maxtask` set to 18,000 is listed below.

```
/export/siebsrvr/admin/%ls -l *.shm ; ls -lh *.shm
-rwx------   1 sunperf  other    2074238976 Jan 25 13:23 siebel.sdcv480s002.shm*
-rwx------   1 sunperf  other       1.9G Jan 25 13:23 siebel.sdcv480s002.shm*
%plimit 12527
12527:  siebsvc -s siebsrvr -a -g sdcv480s002 -e siebel -s sdcv480s002
  resource              current         maximum
  time(seconds)         unlimited       unlimited
```

```
  file(blocks)          unlimited       unlimited
  data(kbytes)          4194303         4194303
  stack(kbytes)         512             512
  coredump(blocks)      unlimited       unlimited
  nofiles(descriptors)  65536           65536
  vmemory(kbytes)       unlimited       unlimited
%prstat -s size -u sunperf -a
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12527 sunperf  3975M 3966M sleep   59    0   0:00:55 0.0% siebsvc/3
 12565 sunperf  2111M  116M sleep   59    0   0:00:08 0.1% siebmtsh/8
 12566 sunperf  2033M 1159M sleep   59    0   0:00:08 3.1% siebmtshmw/10
 12564 sunperf  2021M   27M sleep   59    0   0:00:01 0.0% siebmtsh/12
 12563 sunperf  2000M   14M sleep   59    0   0:00:00 0.0% siebproc/1
 26274 sunperf    16M   12M sleep   59    0   0:01:43 0.0% siebsvc/2
```

In the rare case of needing to run with more than 18,000 Siebel users on a single Siebel server node, install another Siebel server instance on the same node. Such a configuration works well and is a supported on the Siebel and Oracle platform. Users are not advised to configure more maxtasks than needed, as this could effect the performance of the overall Siebel Enterprise.

Dealing with Lost User Connections

During some of the large concurrent user tests, such as 5,000 users on a single Siebel instance, hundreds of users were observed to stall suddenly and transaction response times skyrocketed. Some Siebel server processes servicing these users had either hung or crashed. This problem is commonly — and mistakenly — reported as a memory leak in the `siebemtshmw` process. Close monitoring and repetition of the scenario revealed that the siebmtshmw process memory rose from 700 MB to 2 GB.

The Sun Fire E2900 server can handle up to 30,000 processes and approximately 87,000 LWPs, or threads. Clearly, there was no resource limitation imposed by the server. The Siebel application ran into 32-bit process space limits. The underlying question centered on determining why the memory per process rose from 700 MB to 2 GB. Further debugging indicated the stack size was the issue.

The total memory size of a process is the summation of the heap size, stack size, and the size of the data and anon segments. In this case, the bloating in size was rooted in the stack. The stack grew from 64 KB to 1 GB, an abnormally large amount. The `pmap` utility in Oracle Solaris provides a detailed breakdown of memory sizes per process. Output from the `pmap` command showing the stack segment rising to 1 GB follows.

```
siebapp6@/export/pspp> grep stack pmap.4800mixed.prob.txt (pmap output unit is in Kbytes)
FFBE8000      32      32       -      32 read/write/exec     [ stack ]
FFBE8000      32      32       -      32 read/write/exec     [ stack ]
FFBE8000      32      32       -      32 read/write/exec     [ stack ]
FFBE8000      32      32       -      32 read/write/exec     [ stack ]
FFBE8000      32      32       -      32 read/write/exec     [ stack ]
C012A000 1043224 1043224       - 1043224 read/write/exec     [ stack ]
```

Reducing the stack size to 128 KB fixes the problem. With a reduced stack size, a single Siebel server configured with 10,000 `maxtasks` starts up successfully. The software was failing to start up with a default stack size setting, as it was breaking at the `mmap` call. The default stack size in Oracle Solaris is 8 MB. Reducing and limiting the stack size to 128 KB removed the instability during high load tests. Once the stack size was changed, the 5,000 user tests ran consistently without errors. CPU utilization on the 12-way Sun Fire E2900 server increased to 90 percent.

Changing the mainwin address `MW_GMA_VADDR=0xc0000000` to other values did not seem to make any observable difference. Adjusting this value yields no benefit.

The Solution

Changing the stack size hard limit of the Siebel process from unlimited to 512 bytes solves the problem.

The stack size in Oracle Solaris has two limits: a hard limit (unlimited default), and a soft limit (8 MB default). An application process running on Oracle Solaris can have a stack size anywhere up to the hard limit. Since the default hard limit on Oracle Solaris is unlimited, the Siebel application processes could grow their stack sizes up to 2 GB. When this happens, the total memory size of the Siebel process hits the maximum limit of memory addressable by a 32-bit process, thereby causing the process to hang or crash. Setting the limit for the stack size to 1 MB or lower resolves the issue. The following example shows how this can be done.

```
%limit stack
stacksize 8192 kbytes
%limit stacksize 1024
%limit stacksize
stacksize 512 kbytes
```

A large stack limit can inhibit the growth of the data segment, as the total process size upper limit is 4 GB for a 32-bit application. Data, stack, and heap need to co-exist within this 4 GB address space. Even if the process stack does not grow to a large extent, virtual memory space is reserved for it based

on the default limit setting. The recommendation to limit the stack size to 512 bytes worked well for the workload defined in this paper. Note that this setting may need to be modified for different Siebel deployments and workloads. The range can vary from 512 to 1,024 bytes.

Tuning the Oracle iPlanet Web Server Software

The three main files where tuning can be done are the *obj.conf*, *server.xml*, and *magnus.conf* files.

1. Edit the *magnus.conf* file.

```
Set the RqThrottle=4028 in the magnus.conf file under the Web Server root directory
ListenQ 16000
ConnQueueSize 8000
KeepAliveQueryMeanTime 50
```

2. Edit the *server.xml* file. Replace the host name with the IP address in the *server.xml* file.
3. Edit the *obj.conf* file.
   - Turn off access logging.
   - Turn off `cgi`, `jsp`, and `servlet` support.
   - Remove the following lines from the *obj.conf* file since they are not used by Siebel.

```
###PathCheck fn="check-acl" acl="default"
###PathCheck fn=unix-uri-clean
```

4. Set content expiration to seven days. Doing so improves client response time, particularly in high latency environments. (It eliminates HTTP 304s.) Without content expiration, the login operation performs approximately 65 HTTP GETS. With content expiration, this number is reduced to 15.

Tuning parameters used for high user load with Oracle iPlanet Web Servers are listed in Table 11.

TABLE 11. PARAMETER SETTINGS.

PARAMETER	SCOPE	DEFAULT VALUE	TUNED VALUE
shmsys:shminfo_shmmax	/etc/system		0xffffffffffffffff
shmsys:shminfo_shmmin	/etc/system		100
shmsys:shminfo_shmseg	/etc/system		200
shmsys:shminfo_semmns	/etc/system		12092
shmsys:shminfo_semmsl	/etc/system		512

shmsys:shminfo_semmni	/etc/system		4096
shmsys:shminfo_semmap	/etc/system		4096
shmsys:shminfo_semmnu	/etc/system		4096
shmsys:shminfo_semopm	/etc/system		100
shmsys:shminfo_semume	/etc/system		2048
msgsys:msginfo_msgmni	/etc/system		2048
msgsys:msginfo_msgtql	/etc/system		2048
msgsys:msginfo_msgssz	/etc/system		64
msgsys:msginfo_msgseg	/etc/system		32767
msgsys:msginfo_msgmax	/etc/system		16384
msgsys:msginfo_msgmnb	/etc/system		16384
rlim_fd_max	/etc/system	1024	16384
rlim_fd_cur	/etc/system	64	16384
sq_max_size	/etc/system	2	0
tcp_time_wait_interval	ndd /dev/tcp	240000	60000
tcp_conn_req_max_q	ndd /dev/tcp	128	1024
tcp_conn_req_max_q0	ndd /dev/tcp	1024	4096
tcp_ip_abort_interval	ndd /dev/tcp	480000	60000
tcp_keepalive_interval	ndd /dev/tcp	7200000	900000
tcp_rexmit_interval_max	ndd /dev/tcp	240000	10000
tcp_rexmit_interval_min	ndd /dev/tcp	200	3000
tcp_smallest_anon_port	ndd /dev/tcp	32768	1024
tcp_slow_start_initial	ndd /dev/tcp	1	2
tcp_xmit_hiwat	ndd /dev/tcp	8129	32768
tcp_fin_wait_2_flush_interval	ndd /dev/tcp	67500	675000
tcp_recv_hiwat	ndd /dev/tcp	8129	32768

Tuning the Siebel Web Server Extension

In the Siebel Web Plugin installation directory, go to the *bin* directory. Edit the *eapps.cfg* file and make the following changes.

- Set `AnonUserPool` to 15 percent of `target #users`. The `AnonUserPool` setting can vary based on different types of Siebel users, such as `callcenter` and `eales`.
- Set the following settings in the default section.
  - Set `GuestSessionTimeout = 60`, for scenarios where the user is browsing without logging in.
  - Set `AnonSessionTimeout  = 300`.
  - `Set SessionTimeout = 300`.
- Set appropriate `AnonUser` names and passwords.
  - Set `SADMIN/SADMIN` for eChannel and Call Center (Database login).
  - Set `GUEST1/GUEST1` for eService and eSales (LDAP login).
  - Set `GUESTERM/GUESTERM` for ERM (Database login).

The individual *eapps.cfg* settings for each type of Siebel application, including Call Center and eChannel, used in the 8000 users test are shown in Table 12. More information on tuning Siebel can be found on the SWSE statistics Web page.

TABLE 12. PARAMETER SETTINGS FOR THE 8,000 USER TEST.

PARAMETER	FINSOBJmgr_enu	prmportal_enu
AnonUserNme	SADMIN	GUESTCP
AnonPassword	SADMIN	GUESTCP
AnonUserPool	CC1=420, CC2=525 CC3=315, CC4=210	320
GuestSessionTimeout	60	60
SessionTimeout	300	300

Tuning Siebel Standard Oracle Database and Oracle's Sun Storage

The size of the database used for testing was approximately 240 GB. The database was built to simulate users with large transaction volumes and data distributions representing the most common customer data shapes. Below is a sampling of record volumes and size in database for key business entities of the standard Siebel volume database.

TABLE 13. PARAMETER SETTINGS.

BUSINESS ENTITY	DATABASE TABLE NAME	NUMBER OF RECORDS	SIZE (KB)
Accounts	S_ORG_EXT	1,897,161	1,897,161
Activities	S_EVT_ACT	8,744,305	6,291,456
Addresses	S_ADDR_ORG	3,058,666	2,097,152
Contacts	S_CONTACTS	3,366,764	4,718,592
Employees	S_EMPLOYEE_ATT	21,000	524
Opportunities	S_OPTY	3,237,794	4,194,304
Orders	S_ORDER	355,297	471,859
Products	S_PROD_INT	226,000	367,001
Quote Items	S_QUOTE_ITEM	1,984,099	2,621,440
Quotes	S_QUOTE_ATT	253,614	524
Service Requests	S_SRV_REQ	5,581,538	4,718,592

Optimal Database Configuration

Creating a well-planned database from the start requires less tuning and reorganizing during runtime. Many resources, including books and scripts, are available to facilitate the creation of high-performance Oracle databases. Most database administrators find themselves adjusting each table and index out of the thousands used, based on use and size. This is not only time consuming but prone to error. Eventually the entire database is often rebuilt from scratch. The following steps present an alternate approach to tuning a pre-existing, pre-packaged database.

1. Measure the exact space used by each object in the schema. The `dbms_space` packages provide the accurate space used by an index or table. Other sources, such as `dba_free_space`, only indicate how much is free from the total allocated space, which is always more. Next, run the benchmark test and measure the space used. The difference in results is an accurate report of how much each table or index grows during the test. Using this data, all tables can be rightsized — capacity planned for growth during the test. Furthermore, it is possible to figure out the hot tables used by the test and concentrate on tuning only those tables.
2. Create a new database with multiple index and data tablespaces. The idea is to place all equi-extent sized tables into their own tablespace. Keeping the data and index objects in their own tablespace reduces contention and fragmentation, and also provides for easier monitoring. Keeping tables

with equal extent sizes in their own tablespace reduces fragmentation, as old and new extent allocations are always of the same size within a given tablespace. This eliminates empty, odd-sized pockets in between, leading to compact data placement and a reduced number of I/O operations performed.

3. Build a script to create all of the tables and indexes. This script should result in the tables being created in the appropriate tablespaces with the right parameters, such as `freelists`, `freelist_groups`, `pctfree`, `pctused`, and more. Use this script to place all tables in their tablespaces and then import the data. This results in a clean, defragmented, optimized, and rightsized database.

The tablespaces should be built as locally managed. The space management is done locally within the tablespace, whereas default (dictionary managed) tablespaces write to the system tablespace for every extent change. The list of hot tables for Siebel can be found in Appendix B at the end of this document.

Properly Locating Data on the Disk for Best Performance

To achieve the incredible capacity on disk drives, disk manufacturers implement zone bit recording. With zone bit recording, the outer edge of the disk drive has more available storage area than the inside edge of the disk drive. The number of sectors per track decreases toward the center of the disk. Disk drive manufacturers take advantage of this by recording more data on the outer edges. Since the disk drive rotates at a constant speed, the outer tracks have faster transfer rates than inner tracks.

Consider a Seagate 36 GB Cheetah[2] disk drive. The data transfer speed of this drive ranges from 57 MB/second on the inner tracks to 86 MB/second on the outer tracks, a 50 percent improvement in transfer speed!

For benchmarking purposes, it is desirable to:

- Place active large block transfers on the outer edges of disks to minimize data transfer time.
- Place active, random, small block transfers on the outer edges of the disk drive only if active, large block transfers are not in the benchmark.
- Place inactive, random, small block transfers on the inner sections of disk drive. This is intended to minimize the impact of the data transfer speed discrepancies.

Furthermore, if the benchmark only deals with small block I/O operations, like the SPC Benchmark 1[3], the priority is to put the most active logical unit numbers (LUNs) on the outer edge and the less active LUNs on the inner edge of the disk drive.

---

[2] The Cheetah 15 KB RPM disk drive datasheet can be found at *seagate.com*

[3] More information regarding the SPC Benchmark 1 can be found at *StoragePerformance.org*

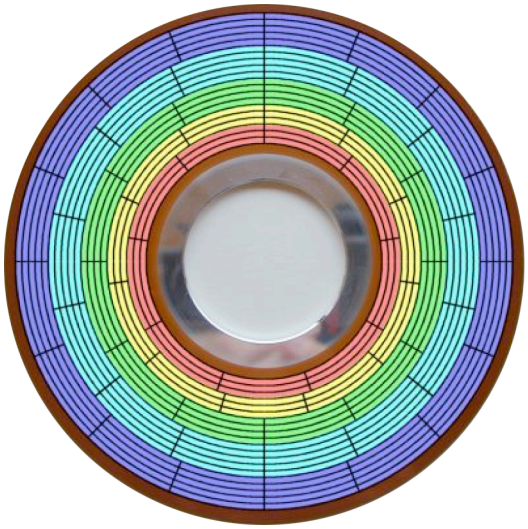

Figure 11. Zone bit recording example with five zones. The outer edge holds the most data and has the fastest transfer rate.

Disk Layout and Oracle Data Partitioning

An I/O subsystem with less contention and high throughput is key for obtaining high performance with Oracle Database Server. This section describes the design choices made after analyzing the workload.

The I/O subsystem consisted of a Sun StorEdge 3510 FC array connected to the Sun Fire v890+ database server via fibre channel adapters. The Sun StorEdge 3510 FC array includes seven trays driven by two controllers. Each tray consists of fourteen 36 GB disks at 15,000 RPM. Hence, the array includes 98 disks providing over 3.5 TB of total storage. Each tray has a 1 GB cache. All trays were formatted using RAID-0, and two LUNs per tray were created. Eight striped volumes, each 300 GB in size, were carved. Each volume was striped across seven physical disks with a 64 KB stripe size. Eight UNIX file systems (UFS) were built on top of the following striped volumes: `T4disk1`, `T4disk2`, `T4disk3`, `T4disk4`, `T4disk5`, `T4disk6`, `T4disk7`, `T4disk8`.

Since Oracle database writes every transaction to redolog files, typically the redo files have higher I/O activity compared to other Oracle datafiles. In addition, writes to Oracle redolog files are sequential. The Oracle redolog files were situated on a dedicated tray using a dedicated controller. Additionally, the LUN containing the redologs was placed on the outer edge of the physical disks. The first file system created using a LUN occupies the outer edge of the physical disks. Once the outer edge reaches capacity, the inner sectors are used. Such a strategy can be used in performance tuning to locate highly used data on outer edges and rarely used data on the inner edge of a disk.

The data tablespaces, index tablespaces, rollback segments, temporary tablespaces and system tablespaces were built using 4 GB datafiles spread across the remaining two trays to ensure there would be no disk hot spotting. The spread makes for effective usage of the two controllers available with this setup. One controller and its 1 GB cache was used for the redolog files, while the other controller and cache were used for non-redo data files belonging to the Oracle software.

The 2,547 data and 1,2391 index objects of the Siebel schema were individually sized. The current space usage and expected growth during the test were accurately measured using the `dbms_space` procedure. Three data tablespaces were created using the locally managed (bitmapped) feature in the Oracle Database. Similarly, three index tablespaces were also created. The extent sizes of these tablespace were `UNIFORM`, ensuring fragmentation does not occur during the numerous deletes, updates and inserts. The tables and indexes were distributed evenly across these tablespaces based on their size, and the extents were pre-created so that no allocation of extents took place during the benchmark tests.

Data partitioning per Oracle tablespace:

```
RBS - All rollback segment objects
DATA_512000K - contained all of the large Siebel tables
INDX_51200K- tablespace for the indexes on large tables
INDX_5120K - tablespace for the indexes on medium tables
DATA_51200K - tablespace to hold the medium Siebel tables
INDX_512K - tablespace for all the indexes on small tables
DATA_5120K - tablespace for the small Siebel tables
TEMP - Oracle temporary segments
TOOLS - Oracle performance measurement objects
DATA_512K - tablespace for Siebel small tables
SYSTEM - oracle system tablespace
```

Tablespace to logical volume mapping:

```
DATA_512000K            /t3disk2/oramst/oramst_data_512000K.01
                        /t3disk2/oramst/oramst_data_512000K.04
                        /t3disk2/oramst/oramst_data_512000K.07
                        /t3disk2/oramst/oramst_data_512000K.10
                        /t3disk3/oramst/oramst_data_512000K.02
                        /t3disk3/oramst/oramst_data_512000K.05
                        /t3disk3/oramst/oramst_data_512000K.08
                        /t3disk3/oramst/oramst_data_512000K.11
                        /t3disk4/oramst/oramst_data_512000K.03
                        /t3disk4/oramst/oramst_data_512000K.06
DATA_51200K             /t3disk2/oramst/oramst_data_51200K.02
                        /t3disk3/oramst/oramst_data_51200K.03
                        /t3disk4/oramst/oramst_data_51200K.01
                        /t3disk4/oramst/oramst_data_51200K.04

DATA_5120K              /t3disk3/oramst/oramst_data_5120K.01
```

```
DATA_512K               /t3disk2/oramst/oramst_data_512K.01

INDX_51200K             /t3disk5/oramst/oramst_indx_51200K.02
                        /t3disk5/oramst/oramst_indx_51200K.05
                        /t3disk5/oramst/oramst_indx_51200K.08
                        /t3disk5/oramst/oramst_indx_51200K.11
                        /t3disk6/oramst/oramst_indx_51200K.03
                        /t3disk6/oramst/oramst_indx_51200K.06
                        /t3disk6/oramst/oramst_indx_51200K.09
                        /t3disk6/oramst/oramst_indx_51200K.12
                        /t3disk7/oramst/oramst_indx_51200K.01
                        /t3disk7/oramst/oramst_indx_51200K.04
                        /t3disk7/oramst/oramst_indx_51200K.07
                        /t3disk7/oramst/oramst_indx_51200K.10

INDX_5120K              /t3disk5/oramst/oramst_indx_5120K.02
                        /t3disk6/oramst/oramst_indx_5120K.03
                        /t3disk7/oramst/oramst_indx_5120K.01

INDX_512K               /t3disk5/oramst/oramst_indx_512K.01
                        /t3disk5/oramst/oramst_indx_512K.04
                        /t3disk6/oramst/oramst_indx_512K.02
                        /t3disk6/oramst/oramst_indx_512K.05
                        /t3disk7/oramst/oramst_indx_512K.03

RBS                     /t3disk2/oramst/oramst_rbs.01
                        /t3disk2/oramst/oramst_rbs.07
                        /t3disk2/oramst/oramst_rbs.13
                        /t3disk3/oramst/oramst_rbs.02
                        /t3disk3/oramst/oramst_rbs.08
                        /t3disk3/oramst/oramst_rbs.14
                        /t3disk4/oramst/oramst_rbs.03
                        /t3disk4/oramst/oramst_rbs.09
                        /t3disk4/oramst/oramst_rbs.15
                        /t3disk5/oramst/oramst_rbs.04
                        /t3disk5/oramst/oramst_rbs.10
                        /t3disk5/oramst/oramst_rbs.16
                        /t3disk6/oramst/oramst_rbs.05
                        /t3disk6/oramst/oramst_rbs.11
```

```
                        /t3disk6/oramst/oramst_rbs.17
                        /t3disk7/oramst/oramst_rbs.06
                        /t3disk7/oramst/oramst_rbs.12
                        /t3disk7/oramst/oramst_rbs.18

SYSTEM                  /t3disk2/oramst/oramst_system.01
TEMP                    /t3disk7/oramst/oramst_temp.12
TOOLS                   /t3disk2/oramst/oramst_tools.01
                        /t3disk3/oramst/oramst_tools.02
                        /t3disk4/oramst/oramst_tools.03
                        /t3disk5/oramst/oramst_tools.04
                        /t3disk6/oramst/oramst_tools.05
                        /t3disk7/oramst/oramst_tools.06
```

With the configuration outlined above — Oracle using the hardware level striping and placing Oracle objects in different tablespaces — an optimal setup was reached as there were no I/O waits noticed and no single disk went over 20 percent busy during the tests. The VERITAS File System (VxFS) was not used, since the setup described above provided the required I/O throughput.

Output from the `iostat` command is below, including two snapshots at five second intervals taken during steady state of the test on the database server. There are minimal reads (r/s), and writes are balanced across all volumes, except for `c7t1d0`, the dedicated array for Oracle redologs. It is quite normal to see high writes/second on redologs — it indicates that transactions are getting done rapidly in the database. The reads/second output is abnormal. The volume is 27 percent busy, which is considered borderline high. Fortunately service times are very low.

```
Wed Jan  8 15:25:20 2003
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
    0.0    1.0    0.0    5.6  0.0  0.0    0.0    1.0   0   1 c0t3d0
    1.0   27.6    8.0  220.8  0.0  0.0    0.0    1.3   0   2 c2t7d0
    0.0   26.0    0.0  208.0  0.0  0.0    0.0    0.5   0   1 c3t1d0
    0.6   37.4    4.8  299.2  0.0  0.0    0.0    0.8   0   2 c4t5d0
    0.0   23.4    0.0  187.2  0.0  0.0    0.0    0.6   0   1 c5t1d0
    0.0   10.2    0.0   81.6  0.0  0.0    0.0    0.5   0   0 c6t1d0
    3.8  393.0 1534.3 3143.8  0.0  0.2    0.0    0.6   0  22 c7t1d0
    0.0   28.2    0.0  225.6  0.0  0.0    0.0    0.6   0   1 c8t1d0
Wed Jan  8 15:25:20 2003
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
```

```
    0.0    1.0    0.0    5.6  0.0  0.0    0.0    1.0   0   1 c0t3d0
    1.0   27.6    8.0  220.8  0.0  0.0    0.0    1.3   0   2 c2t7d0
    0.0   26.0    0.0  208.0  0.0  0.0    0.0    0.5   0   1 c3t1d0
    0.6   37.4    4.8  299.2  0.0  0.0    0.0    0.8   0   2 c4t5d0
    0.0   23.4    0.0  187.2  0.0  0.0    0.0    0.6   0   1 c5t1d0
    0.0   10.2    0.0   81.6  0.0  0.0    0.0    0.5   0   0 c6t1d0
    3.8  393.0 1534.3 3143.8  0.0  0.2    0.0    0.6   0  22 c7t1d0
    0.0   28.2    0.0  225.6  0.0  0.0    0.0    0.6   0   1 c8t1d0
```

Oracle Solaris Multiple Page Size Support Tuning for the Oracle Database Server

Available as a standard feature as of Oracle Solaris 9, Multiple Page Size Support (MPSS) enables a program to use any hardware supported page sizes to access portions of virtual memory. This feature Improves virtual memory performance by allowing applications to use large page sizes, thereby improving resource efficiency and reducing overhead. This is accomplished without recompiling or recoding applications.

Enable MPSS for Oracle Server and shadow processes on Oracle Solaris 9 to reduce the TLB miss rate. It is recommended to use the largest available page size if the TLB miss rate is high. For Oracle Solaris 10 OS 1/06 and later versions, MPSS is enabled by default, and none of the steps described below need to be done.

On Oracle Solaris 10 OS 1/06, simply run the `pmap -xs pid` command to see the actual page size being used by the application. The benefit of upgrading to Oracle Solaris 10 OS 1/06 is that applications use MPSS (large pages) automatically. No tuning or setting is required.

To enable MPSS for Oracle processes:

1. Enable kernel cage if the machine is not a Sun Enterprise E10K or Sun Fire 15K server, and reboot the system. Kernel cage can be enabled by the following setting in the */etc/system* file. Kernel cage should be enabled for the reasons explained earlier in the Siebel server tuning section.

```
set kernel_cage_enable=1
```

2. Find all possible hardware address translation (HAT) sizes supported by the system with the `pagesize -a` command.

```
$ pagesize -a
8192
65536
524288
4194304
```

3. Run the `trapstat -T` command. The value shown in the `ttl` row and `%time` column is the percentage of time spent in virtual to physical memory address translations by the processor(s). Depending on `%time`, choose a page size that helps reduce the iTLB/dTLB miss rate.

4. Create a configuration file for MPSS containing the following line. The desirable heap size and desireable stack size must be one of the supported HAT sizes. By default, all Oracle Solaris releases use an 8 KB page size for heap and stack.

```
oracle*:<desirable heap size>:<desirable stack size>
```

5. Set the `MPSSCFGFILE` and `MPSSERRFILE` environment variables. `MPSSCFGFILE` should point to the configuration file that was created in step 4. MPSS writes any errors during runtime to the `$MPSSERRFILE` file.

6. Preload MPSS interposing the mpss.so.1 library, and start up the Oracle server. It is recommended to put the `MPSSCFGFILE`, `MPSSERRFILE`, and `LD_PRELOAD` environment variables in the Oracle startup script. With all the environment variables mentioned above, a typical startup script may look like the following.

```
echo starting listener
lsnrctl start
echo preloading mpss.so.1 ..
MPSSCFGFILE=/tmp/mpsscfg
MPSSERRFILE=/tmp/mpsserr
LD_PRELOAD=/usr/lib/mpss.so.1:$LD_PRELOAD
export MPSSCFGFILE MPSSERRFILE LD_PRELOAD
echo starting oracle server processes ..
sqlplus /nolog <<!
connect / as sysdba
startup pfile=/tmp/oracle/admin/oramst/pfile/initoramst.ora
!
$ cat /tmp/mpsscfg
oracle*:4M:64K
```

7. Go back to step 3 and measure the difference in `%time`. Repeat steps 4 through 7 until there is a noticeable performance improvement.

More information can be found in the mpss.so.1 man page, as well as the *Supporting Multiple Page Sizes in the Solaris Operating System* white paper located at *solarisinternals.com/si/reading/817-5917.pdf*

Hot Table Tuning and Data Growth

During the four hour test of the 8,000 user OLTP and server component workload, the data in the database grew by 2.48 GB. In total, 256 tables and indexes had new data inserted. Table 14 provides a list of the top 20 tables and indexes based on growth in size. The complete list of tables and indexes that grew can be found in Appendix B.

TABLE 14. TABLE AND INDEX GROWTH STATISTICS.

SIEBEL OBJECT NAME	TYPE	GROWTH (BYTES)
S_DOCK_TXN_LOG	TABLE	1,177,673,728
S_EVT_ACT	TABLE	190,341,120
S_DOCK_TXN_LOG_P1	INDEX	96,116,736
S_DOCK_TXN_LOG_F1	INDEX	52,600,832
S_ACT_EMP	TABLE	46,202,880
S_SRV_REQ	TABLE	34,037,760
S_AUDIT_ITEM	TABLE	29,818,880
S_OPTY_POSTN	TABLE	28,180,480
S_ACT_EMP_M1	INDEX	25,600,000
S_EVT_ACT_M1	INDEX	23,527,424
S_EVT_ACT_M5	INDEX	22,519,808
S_ACT_EMP_U1	INDEX	21,626,880
S_ACT_CONTACT	TABLE	21,135,360
S_EVT_ACT_U1	INDEX	18,391,040
S_ACT_EMP_M3	INDEX	16,850,944
S_EVT_ACT_M9	INDEX	16,670,720
S_EVT_ACT_M7	INDEX	16,547,840
S_AUDIT_ITEM_M2	INDEX	16,277,504
S_ACT_EMP_P1	INDEX	15,187,968
S_AUDIT_ITEM_M1	INDEX	14,131,200

Oracle Parameters Tuning

The key Oracle *init.ora* parameters that were tuned for the Oracle 9.2.0.6 64-bit version on the Oracle Solaris 10 OS are described below.

- **db_cache_size=3048576000.** This parameter determines the size for Oracle's Shared Global Area (SGA). Database performance is highly dependent on available memory. In general, more memory increases caching, which reduces physical I/O to the disks. Oracle's SGA is a memory region in the application which caches database tables and other data for processing. With 32-bit Oracle software running on the 64-bit Oracle Solaris OS, SGA is limited to 4 GB.

  Oracle comes in two basic architectures: 64-bit and 32-bit. The number of address bits determines the maximum size of the virtual address space. For the Siebel 8,000 concurrent users PSPP workload, 6 GB SGA was used. As a result, the 64-bit Oracle server version was used.

```
32-bit = 2^32 = 4 GB maximum
64-bits = 2^64 = 16777216 TB maximum
```

- **Db_block_size=8K.** The default value is 2K. An 8K value for Siebel is optimal.
- **distributed_transactions=0.** Setting this value to zero disables the Oracle background process called `reco`. Siebel does not use distributed transactions. Therefore, it is possible to get back CPU and bus bandwidth by having one less Oracle background process. The default value is 99.
- **replication_dependency_tracking=FALSE.** Siebel does not use replication. Therefore, it is safe to turn it off by setting this parameter to `FALSE`.
- **transaction_auditing=FALSE.** Setting `transaction_auditing` to `FALSE` reduces the amount of redo written per commit. Siebel CRM workloads (OLTP) consist of many small transactions with frequent commits. Setting this parameter to `FALSE` results in reduced CPU and bus consumption.

Following is the listing of the *init.ora* file used with all the parameters set for 8,000 Siebel users.

```
# Oracle 9.2.0.6 init.ora for solaris 10, running upto 15000 Siebel 7.7.1 user benchmark.
db_block_size=8192
db_cache_size=6048576000
db_domain=""
db_name=oramst
background_dump_dest=/export/pspp/oracle/admin/oramst/bdump
core_dump_dest=/export/pspp/oracle/admin/oramst/cdump
timed_statistics=FALSE
user_dump_dest=/export/pspp/oracle/admin/oramst/udump
control_files=("/disk1/oramst77/control01.ctl", "/disk1/oramst77/control02.ctl",
"/disk1/oramst77/control03.ctl")
instance_name=oramst
job_queue_processes=0
```

```
aq_tm_processes=0
compatible=9.2.0.0.0
hash_join_enabled=TRUE
query_rewrite_enabled=FALSE
star_transformation_enabled=FALSE
java_pool_size=0
large_pool_size=8388608
shared_pool_size= 838860800
processes=2500
pga_aggregate_target=25165824
log_checkpoint_timeout=10000000000000000
nls_sort=BINARY
sort_area_size                  = 1048576
sort_area_retained_size         = 1048576
nls_date_format                 = "MM-DD-YYYY:HH24:MI:SS"
transaction_auditing            = false
replication_dependency_tracking = false
session_cached_cursors          = 8000
open_cursors=4048
cursor_space_for_time           = TRUE
db_file_multiblock_read_count   = 8 # stripe size is 64K and not 1M
db_block_checksum=FALSE
log_buffer                    = 10485760
filesystemio_options=setall
pre_page_sga=TRUE
fast_start_mttr_target=0
db_writer_processes=6
transaction_auditing          = FALSE
#timed_statistics             = TRUE # turned off
max_rollback_segments         = 1201
job_queue_processes           = 0
java_pool_size                = 0
#db_block_lru_latches         = 48 # obsolete
session_cached_cursors        = 8000
#FAST_START_IO_TARGET         = 0 # obsolete
#DB_BLOCK_MAX_DIRTY_TARGET    = 0 # obsolete
optimizer_mode                = choose
cursor_sharing                = exact
hash_area_size                = 1048576
```

```
optimizer_max_permutations    = 100
partition_view_enabled        = false # default
#query_rewrite_enabled        = true
query_rewrite_integrity       = trusted
optimizer_index_cost_adj      = 1
parallel_max_servers          = 32
ROLLBACK_SEGMENTS = (rbs, Rbs0, Rbs100..,..,.. ..,Rbs1499) # auto undo not used
```

Oracle Solaris Kernel Parameters on Oracle Database Server

TABLE 15. ORACLE SOLARIS KERNEL PARAMETERS ON ORACLE DATABASE SERVER.

PARAMETER	SCOPE	DEFAULT VALUE	TUNED VALUE
shmsys:shminfo_shmmax	/etc/system		0xffffffffffffffff
shmsys:shminfo_shmmin	/etc/system		100
shmsys:shminfo_shmseg	/etc/system		200
semsys:seminfo_semmns	/etc/system		16384
semsys:seminfo_semmsl	/etc/system		4096
semsys:seminfo_semmni	/etc/system		4096
semsys:seminfo_semmap	/etc/system		4096
semsys:seminfo_semmnu	/etc/system		4096
semsys:seminfo_semopm	/etc/system		4096
semsys:seminfo_semume	/etc/system		2048
semsys:seminfo_semvmx	/etc/system		32767
semsys:seminfo_semaem	/etc/system		16384
msgsys:msginfo_msgmni	/etc/system		4096
msgsys:msginfo_msgtql	/etc/system		4096
msgsys:msginfo_msgmax	/etc/system		16384
msgsys:msginfo_msgmnb	/etc/system		16384
rlim_fd_max	/etc/system	1024	16384
rlim_fd_cur	/etc/system	64	16384

SQL Query Tuning

During the course of the test, the most resource-intensive and long-running queries were tracked. In general, the best way to tune a query is to change the SQL statement while keeping the result set the same. Another approach is to add or drop indexes so the execution plan changes. The latter method is the only option in most benchmark tests. Four additional indexes were added to the Siebel schema, helping performance. The Siebel 7.5 version supported only Rule Based Optimization (RBO) with Oracle Database. Cost Based Optimization (CBO) support is available in Siebel 7.7 and later versions. Full statistics were collected for all the tables and indexes in the Siebel schema, and then CBO was enabled. These steps significantly improved the performance of queries. The example below shows one of the resource consuming queries.

```
Buffer Gets    Executions  Gets per Exec  % Total  Hash Value
-------------- ------------ -------------- ------- ------------
220,402,077       35,696        6,174.4    33.2   2792074251
```

It was responsible for 33 percent of the total buffer gets from all queries during the benchmark tests.

```
SELECT
        T4.LAST_UPD_BY,
        T4.ROW_ID,
        T4.CONFLICT_ID,
        T4.CREATED_BY,
        T4.CREATED,
        T4.LAST_UPD,
        T4.MODIFICATION_NUM,
        T1.PRI_LST_SUBTYPE_CD,
        T4.SHIP_METH_CD,
        T1.PRI_LST_NAME,
        T4.SUBTYPE_CD,
        T4.FRGHT_CD,
        T4.NAME,
        T4.BU_ID,
        T3.ROW_ID,
        T2.NAME,
        T1.ROW_ID,
        T4.CURCY_CD,
        T1.BU_ID,
        T4.DESC_TEXT,
        T4.PAYMENT_TERM_ID
FROM
```

```
        ORAPERF.S_PRI_LST_BU T1,
        ORAPERF.S_PAYMENT_TERM T2,
        ORAPERF.S_PARTY T3,
        ORAPERF.S_PRI_LST T4
WHERE
        T4.PAYMENT_TERM_ID = T2.ROW_ID (+)
AND
        T1.BU_ID = :V1
AND
        T4.ROW_ID = T1.PRI_LST_ID
AND
        T1.BU_ID = T3.ROW_ID
AND
        ((T1.PRI_LST_SUBTYPE_CD != 'COST LIST'
AND
        T1.PRI_LST_SUBTYPE_CD != 'RATE LIST')
AND
        (T4.EFF_START_DT <= TO_DATE(:V2,'MM/DD/YYYY HH24:MI:SS')
AND
        (T4.EFF_END_DT IS NULL
OR
        T4.EFF_END_DT >= TO_DATE(:V3,'MM/DD/YYYY HH24:MI:SS'))
AND
        T1.PRI_LST_NAME LIKE :V4
AND
        T4.CURCY_CD = :V5))
ORDER BY T1.BU_ID, T1.PRI_LST_NAME;
```

The output below shows the executing plan and statistics before the new index was added.

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=RULE
   1    0   NESTED LOOPS (OUTER)
   2    1     NESTED LOOPS
   3    2       NESTED LOOPS
   4    3         TABLE ACCESS (BY INDEX ROWID) OF 'S_PRI_LST_BU'
   5    4           INDEX (RANGE SCAN) OF 'S_PRI_LST_BU_M1' (NON-UNIQUE)
   6    3         TABLE ACCESS (BY INDEX ROWID) OF 'S_PRI_LST'
```

```
   7    6           INDEX (UNIQUE SCAN) OF 'S_PRI_LST_P1' (UNIQUE)
   8    2       INDEX (UNIQUE SCAN) OF 'S_PARTY_P1' (UNIQUE)
   9    1     TABLE ACCESS (BY INDEX ROWID) OF 'S_PAYMENT_TERM'
  10    9       INDEX (UNIQUE SCAN) OF 'S_PAYMENT_TERM_P1' (UNIQUE)



Statistics
----------------------------------------------------------
        364  recursive calls
          1  db block gets
      41755  consistent gets
          0  physical reads
          0  redo size
     754550  bytes sent via SQL*Net to client
      27817  bytes received via SQL*Net from client
        341  SQL*Net roundtrips to/from client
          4  sorts (memory)
          0  sorts (disk)
       5093  rows processed
```

After the new index was created:

```
create index S_PRI_LST_X2 on S_PRI_LST
(CURCY_CD, EFF_END_DT, EFF_START_DT)
STORAGE(INITIAL 512K NEXT 512K
MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0 FREELISTS 7 FREELIST
GROUPS 7 BUFFER_POOL DEFAULT) TABLESPACE INDX NOLOGGING PARALLEL 4 ;
```

The difference in statistics show the new index resulted in 1.5 times fewer consistent gets.

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=RULE
   1    0   SORT (ORDER BY)
   2    1     NESTED LOOPS
   3    2       NESTED LOOPS
   4    3         NESTED LOOPS (OUTER)
   5    4           TABLE ACCESS (BY INDEX ROWID) OF 'S_PRI_LST'
   6    5             INDEX (RANGE SCAN) OF 'S_PRI_LST_X2' (NON-UNIQUE
```

```
   7    4           TABLE ACCESS (BY INDEX ROWID) OF 'S_PAYMENT_TERM'
   8    7             INDEX (UNIQUE SCAN) OF 'S_PAYMENT_TERM_P1' (UNIQUE)
   9    3         TABLE ACCESS (BY INDEX ROWID) OF 'S_PRI_LST_BU'
  10    9           INDEX (RANGE SCAN) OF 'S_PRI_LST_BU_U1' (UNIQUE)
  11    2       INDEX (UNIQUE SCAN) OF 'S_PARTY_P1' (UNIQUE)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
      27698  consistent gets
          0  physical reads
          0  redo size
     754550  bytes sent via SQL*Net to client
      27817  bytes received via SQL*Net from client
        341  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
5093     rows processed
```

Similarly, the three new indexes added were as follows. The last two indexes are for assignment manager tests. No inserts occurred on the base tables during these tests.

```
create index S_CTLG_CAT_PROD_F1 on ORAPERF.S_CTLG_CAT_PROD (CTLG_CAT_ID ASC) PCTFREE 10
INITRANS 2
MAXTRANS 255 STORAGE (INITIAL 5120K NEXT 5120K MINEXTENTS 2 MAXEXTENTS UNLIMITED PCTINCREASE
0 FREELISTS 47 FREELIST GROUPS 47 BUFFER_POOL DEFAULT) TABLESPACE INDX_5120K NOLOGGING;


create index S_PROG_DEFN_X1 on ORAPERF.S_PROG_DEFN
 (NAME, REPOSITORY_ID)
STORAGE(INITIAL 512K NEXT 512K
MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0
BUFFER_POOL DEFAULT) TABLESPACE INDX_512K NOLOGGING;


create index S_ESCL_OBJECT_X1 on ORAPERF.S_ESCL_OBJECT
(NAME, REPOSITORY_ID, INACTIVE_FLG)
STORAGE(INITIAL 512K NEXT 512K
MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0
BUFFER_POOL DEFAULT) TABLESPACE INDX_512K NOLOGGING;
```

Rollback Segment Tuning

An incorrect number and size of rollback segments can cause poor performance. The right number and size of rollback segments depends on the application workload. Most OLTP workloads require several small rollback segments. The number of rollback segments should be equal to or greater than the number of concurrent active transactions in the database during peak load. In the Siebel tests, this was about 80 during a 8,000 user test. The size of each rollback segment should be approximately equal to the size in bytes of a user transaction. For the Siebel workload 400 rollback segments of 20 MB with an extent size of 1 MB was found suitable. An important fact to keep in mind with rollback segment sizing is that if the size is larger than is required by the application, valuable space in the database cache can be wasted. The new `UNDO Segments` Oracle feature, which can be used instead of rollback segments, was not tested during this testing effort with the Siebel application.

Database Connectivity Using Host Names Adapter

Observations reveal that high-end Siebel test runs using Oracle as the back end perform better when client to Oracle server connectivity is done via the hostnames adapter feature. The hostnames adapter is an Oracle feature where the *tnsnames.ora* file is no longer used to resolve the connect-string to the database server. Instead, resolution is done at the Oracle Solaris OS level using the */etc/hosts* file. The procedure below can be use to set this up.

- Set `GLOBAL_DBNAME` to a value other than `Oracle_SID` in the *Listener.ora* file on the database server as shown below. Bounce the Oracle listener after making the change.

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = dbserver)(PORT = 1521))
      )
    )
    (DESCRIPTION =
      (PROTOCOL_STACK =
        (PRESENTATION = GIOP)
        (SESSION = RAW)
      )
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbserver)(PORT = 2481))
    )
  )
SID_LIST_LISTENER =
```

```
(SID_LIST =
  (SID_DESC =
    (SID_NAME = PLSExtProc)
    (ORACLE_HOME = /export/pspp/oracle)
    (PROGRAM = extproc)
  )
  (SID_DESC =
    (GLOBAL_DBNAME = oramst.dbserver)
    (ORACLE_HOME = /export/pspp/oracle)
    (SID_NAME = oramst)
```

- On the Siebel applications server, go into the Oracle client installation and delete or rename the *tnsnames.ora* and *sqlnet.ora* files. These files are no longer needed as Oracle now connects to the database by resolving the name from the */etc/hosts* file.
- As `root`, edit the */etc/hosts* file on the client machine (the Siebel application server) and add an entry like the following. The name `oramst.dbserver` should be the same as the `GLOBAL_DBNAME` in the *listener.ora* file. This becomes the connect string to connect to the database from any client.

```
<ip.address of database server>   oramst.dbserver
```

High I/O with Oracle Shadow Processes Connected to Siebel

The disk on which Oracle binaries were installed was close to 100 percent busy during the peak load of 1,000 concurrent users. This problem was diagnosed to be the well-known *oraus.msb* problem. For Oracle clients that make use of the Oracle Call Interface (OCI), the OCI driver makes thousands of calls to translate messages from the oraus.msb file. This problem is documented by Oracle under bug ID 2142623.

There is a workaround for this problem — caching the *oraus.msb* file in memory, and translating the file access and system calls to user calls and memory operations. The caching solution is dynamic, and code changes are not needed. With Siebel this workaround helped in reducing the calls: the 100 percent busy I/O problem went away and a 4 percent reduction in CPU was observed. The response times for transactions received an 11 percent boost.

This problem is fixed in Oracle 9.2.0.4. Details on how to implement the workaround by the `LD_PRELOAD` interpose library can be found at *http://developers.sun.com/solaris/articles/oci_cache.html*

Siebel Database Connection Pooling

The database connection pooling feature built into the Siebel server software provides improved performance. A users:database connection ratio of 20:1 has been proven to provide good results with Siebel7.7/Oracle9206. This connection ratio reduced CPU utilization by approximately 3 percent at

Siebel server, as fewer connections are made from the Siebel server to the database during a 2,000 user Call Center test. Siebel memory per user is 33 percent lower, and Oracle memory/user is 79% percent lower as 20 Siebel users share the same Oracle connection.

Siebel anonymous users do not use connection pooling. If the anonuser count is set too high (greater than the recommended 10 to 20 percent) tasks can be wasted, as `maxtasks` is inclusive of real users. Also, the `anon` sessions do not use connection pooling, resulting in many one-to-one connections that can lead to increased memory and CPU usage both on the database server and application server.

To enable connection pooling, perform the following steps:

- Set the following Siebel parameters at the server level via the Siebel thin client GUI or `svrmgr`.

```
MaxSharedDbConns        integer full    <number of connections to be used>
MinSharedDbConns        integer full    <number of connections to be used>
MaxTrxDbConns integer   full            <number of connections to be used>
```

- Bounce the Siebel Server. For example, if configured to run 1,000 users, then the value for number of connections to be used is 1000/20=50. Set the above three parameters to the same value (50). This directs Siebel to share a single database connection for 20 Siebel users or tasks.

```
srvrmgr:SIEBSRVR> change param MaxTasks=1100, MaxMTServers=20, MinMTServers=20,
MinSharedDbConns=50, MaxSharedDbConns=50, MinTrxDbConns=50 for comp esalesobjmgr_enu
```

To check if connection pooling is enabled, login to `dbserver` and execute the `ps -eaf | grep NO | wc -l` command during the steady state. This should return around 50 for this example. If it returns 1,000 then connection pooling is not enabled.

Performance Tweaks with No Gains

This section discusses the non-tunables, the ones that provided no benefits and mostly have no impact on the performance tests. These are tunables that may help other applications in a different scenario, or are default settings already in effect. Below is a list of some non-tunables encountered.

- Changing the `mainwin` address `MW_GMA_VADDR=0xc0000000` to other values did not seem to make a significant impact or difference. This value is set in the *siebenv* file.
- The Oracle Solaris kernel stksize parameter has a default value of 16 KB, or 0x4000 on the sun4u architecture machines booted in 64-bit mode (the default). Increasing the value to 24 KB (0x6000) via the settings below in the */etc/system* file did not result in any performance gains during the tests.

Note—These observations are specific to the workload, architecture, and software versions used during testing project. The outcome of certain tunables may vary when implemented with a different workload on a different architecture or configuration.

```
set rpcmod:svc_default_stksize=0x6000
set lwp_default_stksize=0x6000
```

- Enabling the Siebel Server Recycle Factor did not provide any performance gains. The default is disabled.
- Siebel Server `SISSPERSISSCONN` is the parameter that changes the multiplexing ratio between Siebel server and Web server. The default value is 20. Varying the `SISSPERSISSCONN` parameter did not result in any performance gains for the specific modules tested in this project with the PSPP standard workload.
- When the Oracle iPlanet Web Server `maxprocs` parameter was changed from the default setting of one, the software starts up more than one Web server process (`ns-httpd`). No gain was measured with a value greater than one. It is better to use a new Web server instance.
- Enabling Database Connection pooling with Siebel Server components for the server component batch workload caused performance to degrade and server processes could not start. Some of the server component modules connect to the database using the Open Database Connectivity (ODBC) standards, which does not support connection pooling.
- Several Oracle Database Server parameters are worth noting.
  - For an 8,000 Siebel user benchmark, a `sharedpoolsize` value of 400 MB was more than sufficient. Using too large a value wastes valuable database cache memory.
  - Using a larger SGA size than is required does not improve performance, while using too small a value can degrade performance.
  - Using a larger RBS value than is required by the application can waste space in the database cache. A better strategy is to make the application commit more often.

Tips and Scripts for Diagnosing Oracle's Siebel on Oracle Systems

This section reveals some of the tips found to be helpful for diagnosing performance and scalability issues while running Siebel on the Oracle platform.

Monitoring Siebel Open Session Statistics

The following URLs can be used to monitor the amount of time a Siebel end user transaction is taking within a Siebel Enterprise. This data is updated near real time. These statistics pages provide a lot of diagnostic information. Watch out for any rows that appear in bold, as they represent requests that have been waiting for over 10 seconds.

TABLE 16. DIAGNOSTIC URLS.

SERVICE	URL
Call Center	http://webserver:port/callcenter_enu/_stats.swe?verbose=high
eSales	http://webserver:port/esales_enu/_stats.swe?verbose=high
eService	http://webserver:port/eservice_enu/_stats.swe?verbose=high
eChannel	http://webserver:port/prmportal_enu/_stats.swe?verbose=high

Listing the Parameter Settings for a Siebel Server

Use the following server command to list all parameter settings for a Siebel Server. Parameters of interest include `MaxMTServers`, `MinMTSServers`, `MaxTasks`, `MinSharedDbConns`, `MaxSharedDbConns`, and `MinTrxDbConns`.

```
$> srvrmgr /g gateway /e enterprise /u sadmin /p sadmin -c "list tasks for comp component show
SV_NAME, CC_ALIAS, TK_PID, TK_DISP_RUNSTATE" | grep Running | sort -k 3 | uniq -c | sort -k 2,2
-k 1,1
```

The output below is a result of a run. Run States are `Running`, `Online`, `Shutting Down`, `Shutdown`, and `Unavailable`. Run Tasks should be evenly distributed across servers and close to `Max Tasks`.

```
 47    siebelapp2_1    eChannelObjMgr_enu       19913   Running
132    siebelapp2_1    eChannelObjMgr_enu       19923   Running
133    siebelapp2_1    eChannelObjMgr_enu       19918   Running
158    siebelapp2_1    eChannelObjMgr_enu       19933   Running
159    siebelapp2_1    eChannelObjMgr_enu       19928   Running
118    siebelapp2_1    eSalesObjMgr_enu         19943   Running
132    siebelapp2_1    eSalesObjMgr_enu         19948   Running
156    siebelapp2_1    eSalesObjMgr_enu         19963   Running
160    siebelapp2_1    eSalesObjMgr_enu         19953   Running
160    siebelapp2_1    eSalesObjMgr_enu         19958   Running
169    siebelapp2_1    eServiceObjMgr_enu       19873   Running
175    siebelapp2_1    eServiceObjMgr_enu       19868   Running
178    siebelapp2_1    eServiceObjMgr_enu       19883   Running
179    siebelapp2_1    eServiceObjMgr_enu       19878   Running
179    siebelapp2_1    eServiceObjMgr_enu       19888   Running
 45    siebelapp2_1    FINSObjMgr_enu           19696   Running
 45    siebelapp2_1    FINSObjMgr_enu           19702   Running
 51    siebelapp2_1    FINSObjMgr_enu           19697   Running
104    siebelapp2_1    FINSObjMgr_enu           19727   Running
```

Determining the Number of Active Servers for a Component

Use the following server command to determine the number of active MTS Servers for a component. The number of active MTS Servers should be close to the number of Max MTS Servers.

```
srvrmgr> list comp component for server servername show SV_NAME, CC_ALIAS, CP_ACTV_MTS_PROCS,
CP_MAX_MTS_PROCS
```

Finding the Tasks for a Component

Use the following server command to find the tasks for a component. Ordering by task ID places the most recently started task at the bottom. If the most recently started tasks are still running, such as those started in the past few seconds or minutes, then it is a good sign. Otherwise, further investigation is required.

```
srvrmgr> list task for comp component server servername order by TK_TASKID
```

Setting Detailed Trace Levels on the Siebel Server Processes

Log in to **svrmgr** and execute the following commands.

```
Change evtloglvl taskcounters=4 for comp sccobjmgr
change evtloglvl taskcounters=4 for comp eserviceobjmgr
change evtloglvl taskevents=3 for comp sccobjmgr
change evtloglvl taskevents=3 for comp eserviceobjmgr
change evtloglvl mtwaring=2 for comp sccobjmgr
change evtloglvl mtwaring=2 for comp eserviceobjmgr
change evtloglvl set mtInfraTrace = True
```

Finding the Number of GUEST Logins for a Component

Use the following server command to find out the number of **GUEST** logins for a component.

```
$ srvrmgr /g gateway /e enterprise /s server /u sadmin /p sadmin /c “list task for comp
component” | grep Running | grep GUEST | wc -l
```

Calculating the Memory Usage for an OM

- The *pmem_sum.sh* script can be used to calculate the memory usage for an OM.

```
#!/bin/sh
if [ $# -eq 0 ]; then
    echo "Usage: pmem_sum.sh <pattern>"
fi
WHOAMI=`/usr/ucb/whoami`
PIDS=`/usr/bin/ps -ef | grep $WHOAMI" " | grep $1 | grep -v "grep $1" | grep -v
```

```
pmem_sum | \
    awk '{ print $2 }'`
for pid in $PIDS
do
echo 'pmem process :' $pid
pmem $pid > `uname -n`.$WHOAMI.pmem.$pid
done


pmem $PIDS | grep total | awk 'BEGIN { FS = " " } {print  $1,$2,$3,$4,$5,$6} {tot+=$4}
{shared+=$5} {private+=$6} END {print "Total memory used:", tot/1024 "M by "NR" procs.
Total Private mem: "private/1024" M Total Shared mem: " shared/1024 "M  Actual used
memory:" ((private/1024)+(shared/1024/NR)) "M"}'
```

- To use the pmem_sum.sh script, type `pmem_sum.sh` siebmtshmw.

Finding the Log File Associated with a Specific OM

- Check the Server log file for the creation of the multithreaded server process.

```
ServerLog       Startup 1       2003-03-19 19:00:46     Siebel Application Server is ready
and awaiting requests
...
ServerLog       ProcessCreate   1       2003-03-19 19:00:46  Created multithreaded server
process (OS pid = 24796) for Call Center Object Manager (ENU) with task id 22535
...
```

- The log file associated with the above OM is FINSObjMgr_enu_24796.log.

```
1021 2003-03-19 19:48:04 2003-03-19 22:23:20 -0800 0000000d 001 001f 0001 09 FINSObjMgr_enu
24796 24992 111
/export/pspp/siebsrvr/enterprises/siebel2/siebelapp1/log/FINSObjMgr_enu_24796.log 7.5.2.210
[16060] ENUENU
...
```

Producing a Stack Trace for the Current Thread of an OM

- Determine the current thread number the OM is running. This example assumes the process ID is 24987. The thread number for this example is 93.

```
% > cat FINSObjMgr_enu_24987.log
1021 2003-03-19 19:51:30 2003-03-19 22:19:41 -0800 0000000a 001 001f 0001 09 FINSObjMgr_enu
24987 24982 93
/export/pspp/siebsrvr/enterprises/siebel2/siebelapp1/log/FINSObjMgr_enu_24987.log 7.5.2.210
[16060] ENUENU
```

- Use the `pstack` command to produce the stack trace.

```
$ > pstack 24987 | sed -n '/lwp# 93/,/lwp# 94/p'
-----------------  lwp# 93 / thread# 93  --------------------
7df44b7c lwp_mutex_lock (c00000c0)
 7df40dc4 mutex_lock_kernel (4ea73a00, 0, 7df581b8, 7df56000, 0, c00000c0) + c8
 7df41a64 mutex_lock_internal (4ea73a00, 7df581ac, 0, 7df56000, 1, 0) + 44c
 7e3c430c CloseHandle (11edc, 7e4933a8, c01f08c8, 7ea003e4, c1528, 4ea73a98) + a8
 7ea96958 __1cKCWinThread2T6M_v_ (7257920, 2, c1538, 1d, 0, 0) + 14
 7ea97768 __SLIP.DELETER__B (7257920, 1, 7ebc37c0, 7ea00294, dd8f8, 4a17f81c) + 4
 7ea965f0 __1cMAfxEndThread6FIi_v_ (7257ab8, 7257920, 0, 1, 1, 0) + 58
 7edd2c6c __1cVOSDSolarisThreadStart6Fpv_0_ (7aba9d0, 1, c01f08c8, 51ecd, 1, 0) + 50
 7fb411bc __1cUWslThreadProcWrapper6Fpv_I_ (7aba9e8, 7e4933a8, c01f08c8, c01f08c8, 0,
ffffffff) + 48
 7ea9633c __1cP_AfxThreadEntry6Fpv_I_ (51ecc, ffffffff, 1, 7ea9787c, 4000, 4a17fe10) + 114
 7e3ca658 __1cIMwThread6Fpv_v_ (1, 7e4a6d00, 7e496400, c0034640, c01f0458, c01f08c8) + 2ac
 7df44970 _lwp_start (0, 0, 0, 0, 0, 0)
-----------------  lwp# 94 / thread# 94  --------------------
```

Showing System-wide Lock Contention Issues Using lockstat

Use the `lockstat` command to find out a lot of things regarding lock contentions. For example, `lockstat` can be used to locate the most contended lock in the system. The following example output shows the system locks in contention during one of the 4,600 user runs with large latch values and double ramp up time.

```
# lockstat sleep 5
Adaptive mutex spin: 17641 events in 4.998 seconds (3529 events/sec)

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
 3403  19%  19% 1.00       51 0x30017e123e0          hmestart+0x1c8
 3381  19%  38% 1.00      130 service_queue          background+0x130
 3315  19%  57% 1.00      136 service_queue          background+0xdc
 2142  12%  69% 1.00       86 service_queue          qenable_locked+0x38
  853   5%  74% 1.00       41 0x30017e123e0          hmeintr+0x2dc
…
    1   0% 100% 1.00        5 0x300267b75f0          lwp_unpark+0x60
    1   0% 100% 1.00       18 0x3001d9a79c8          background+0xb0
-------------------------------------------------------------------------------
```

```
Adaptive mutex block: 100 events in 4.998 seconds (20 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
   25  25%  25% 1.00    40179 0x30017e123e0          hmeintr+0x2dc
    8   8%  33% 1.00   765800 0x30017e123e0          hmestart+0x1c8
    6   6%  39% 1.00   102226 service_queue          background+0xdc
    5   5%  44% 1.00    93376 service_queue          background+0x130
…
    1   1% 100% 1.00    74480 0x300009ab000          callout_execute+0x98
-------------------------------------------------------------------------------

Spin lock spin: 18814 events in 4.998 seconds (3764 events/sec)

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
 2895  15%  15% 1.00     2416 sleepq_head+0x8d8      cv_signal+0x38
  557   3%  18% 1.00     1184 cpu[10]+0x78           disp_getbest+0xc
  486   3%  21% 1.00     1093 cpu[2]+0x78            disp_getbest+0xc
…
    1   0% 100% 1.00     1001 turnstile_table+0xf68  turnstile_lookup+0x50
    1   0% 100% 1.00     1436 turnstile_table+0xbf8  turnstile_lookup+0x50
    1   0% 100% 1.00     1618 turnstile_table+0xc18  turnstile_lookup+0x50
-------------------------------------------------------------------------------
Thread lock spin: 33 events in 4.998 seconds (7 events/sec)

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
    2   6%   6% 1.00      832 sleepq_head+0x8d8      setrun+0x4
2   6%  12% 1.00      112 cpu[3]+0xb8            ts_tick+0xc
    2   6%  18% 1.00      421 cpu[8]+0x78            ts_tick+0xc
...
1   3%  97% 1.00        1 cpu[14]+0x78           turnstile_block+0x20c
1   3% 100% 1.00      919 sleepq_head+0x328      ts_tick+0xc
-------------------------------------------------------------------------------

R/W writer blocked by writer: 73 events in 4.998 seconds (15 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
```

```
-------------------------------------------------------------------------------
    8  11%  11% 1.00   100830 0x300274e5600          segvn_setprot+0x34
    5   7%  18% 1.00    87520 0x30029577508          segvn_setprot+0x34
    4   5%  23% 1.00    96020 0x3002744a388          segvn_setprot+0x34
…
    1   1%  99% 1.00   152960 0x3001e296650          segvn_setprot+0x34
    1   1% 100% 1.00   246960 0x300295764e0          segvn_setprot+0x34
-------------------------------------------------------------------------------

R/W writer blocked by readers: 40 events in 4.998 seconds (8 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
    4  10%  10% 1.00    54860 0x300274e5600          segvn_setprot+0x34
3   8%  18% 1.00    55733 0x3002744a388          segvn_setprot+0x34
3   8%  25% 1.00   102240 0x3001c729668          segvn_setprot+0x34
...
    1   2%  98% 1.00    48720 0x3002759b500          segvn_setprot+0x34
    1   2% 100% 1.00    46480 0x300295764e0          segvn_setprot+0x34
-------------------------------------------------------------------------------

R/W reader blocked by writer: 52 events in 4.998 seconds (10 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
    5  10%  10% 1.00   131488 0x300274e5600          segvn_fault+0x38
    3   6%  15% 1.00   111840 0x3001a62b940          segvn_fault+0x38
    3   6%  21% 1.00   139253 0x3002792f2a0          segvn_fault+0x38
...
    1   2%  98% 1.00    98400 0x3001e296650          segvn_fault+0x38
    1   2% 100% 1.00   100640 0x300295764e0          segvn_fault+0x38
-------------------------------------------------------------------------------

Lockstat record failure: 5 events in 4.998 seconds (1 events/sec)

Count indv cuml rcnt          Lock                   Caller
-------------------------------------------------------------------------------
    5 100% 100% 0.00          lockstat_lock          lockstat_record
-------------------------------------------------------------------------------
```

Showing the Lock Statistic of an OM Using plockstat

The syntax of the `plockstat` command is as follows:

```
plockstat [-o outfile] -p pid
```

The program grabs a process and shows the lock statistics upon exit or interrupt. The following output shows the lock statistics of an OM during one of the 4,600 user runs with large latch values and double ramp up time.

```
$> plockstat -p 4027
^C


----------- mutex lock statistics -----------
   lock   try_lock  sleep  avg sleep   avg hold  location:
  count count  fail count  time usec  time usec   name
   2149     0     0     1       5218        142  siebmtshmw: __environ_lock
   2666     0     0     0          0          3  [heap]: 0x9ebd0
    948     0     0     0          0          1  [heap]: 0x9f490
    312     0     0     2        351         88  [heap]: 0x9f4c8
    447     0     0     0          0          2  [heap]: 0x9f868
    237     0     0     0          0        101  [heap]: 0x9f8a0
   2464     0     0     1       4469          2  [heap]: 0xa00f0
      1     0     0     0          0         11  [heap]: 0x17474bc0
...
    219     0     0     0          0          2  libsscassmc: m_cacheLock+0x8
     41    41     0     0          0          2  0x79a2a828
 152295     0     0    15      11407          1  libthread: tdb_hash_lock
   2631     0     0    10     297603        468  libc: _time_lock
 1807525    0     0 16762      59752         14  libc: __malloc_lock
     ----------- condvar statistics -----------
 cvwait  avg sleep tmwait timout  avg sleep signal brcast  location:
  count  time usec  count  count  time usec  count  count   name
      0          0     41     40    4575290      0      0  [heap]: 0x2feec30
      8   16413463      0      0          0      8      0  [heap]: 0x305fce8
     20    7506539      0      0          0     20      0  [heap]: 0x4fafbe8
     16    6845818      0      0          0     16      0  [heap]: 0x510a8d8
...
     12    8960055      0      0          0     12      0  [heap]: 0x110f6138
     13   10375600      0      0          0     13      0  [heap]: 0x1113e040
```

```
      ----------- readers/writer lock statistics -----------
 rdlock   try_lock  sleep  avg sleep wrlock   try_lock  sleep  avg sleep   avg hold
location:
  count count  fail count  time usec  count count  fail count  time usec  time usec   name
    382     0     0     0          0      0     0     0     0          0          0  [heap]:
0x485c2c0
 102100     0     0     0          0      0     0     0     0          0          0
libsscfdm: g_CTsharedLock
```

Trussing an OM

- Modify `siebmtshw`.

```
#!/bin/ksh
. $MWHOME/setmwruntime
MWUSER_DIRECTORY=${MWHOME}/system
LD_LIBRARY_PATH=/usr/lib/lwp:${LD_LIBRARY_PATH}
#exec siebmtshmw $@
truss -l -o /tmp/$$.siebmtshmw.trc siebmtshmw $@
```

- After the servers starts up, this wrapper creates the truss output in a file named *pid.siebmtshmw.trc* in the */tmp* directory.

Tracing the SQL Statements for a Particular Siebel Transaction

If response times are high, or the database is believed to be a bottleneck, check how long SQL queries are taking to execute by running a SQL trace on a LoadRunner script. The SQL trace is run through Siebel and tracks all database activity and how long things take to execute. If execution times are too high, there is a problem with the database configuration which most likely is contributing to high response times. The following steps can be used to run a SQL Trace on a particular script.

1. Configure the Siebel environment to the default settings. For example, the component should have only one OM.
2. Open the LoadRunner script in question in the Virtual User Generator.
3. Place a breakpoint at the end of Action 1 and before Action 2 to stop the user at the breakpoint.
4. Run a user.
5. Once the user is stopped at the breakpoint, enable SQL tracing via `srvrmgr`.

   ```
   change evtloglvl ObjMgrSqlLog=4 for comp component
   ```

6. Press Play to resume the user.
7. Wait until the user is finished.

8. Under the *$SIEBEL_SERVER_HOME/enterprises/<enterprise>/<server> directory*, a log exists for the running component. The log contains detailed information on database activity, including how long SQL queries took to execute. Search the logs for high execution times — times greater than 0.01 seconds.
9. When finished, disable SQL tracing via `srvrmgr`.

```
change evtloglvl ObjMgrSqlLog=0 for comp <component>
```

Changing the Database Connect String

- Edit `$ODBCINI` with the `vi` editor and edit the `ServerName` field.

```
srvrmgr /g siebgateway /e siebel /u sadmin /p sadmin /s <server name>
```

- At the `srvrmgr` prompt, verify and change the value of the `DSConnectString` parameter.
  - List parameters for subsystems named serverdatasrc.
  - Change the `DSConnectString=new_value` parameter for the subsystems named `serverdatasrc`.

Enabling and Disabling Siebel Components

To disable a component:

- Bring up the `srvrmgr` console.

```
srvrmgr /g siebgateway /e siebel /u sadmin /p sadmin /s <server name>
```

- Disable the component.

```
disable comp <component name>
```

- List the components and verify status.

```
list components
```

Disabling a component may disable the component definition. The component definition may need to be enabled. To enable a component:

- Bring up the `srvrmgr` console at the enterprise level. Do not use the `/s` switch.

```
srvrmgr /g siebgateway /e siebel /u sadmin /p sadmin
```

- Enable the component definition. Note this action enables the component definition at all active servers. Be sure to disable the component at servers where the component is not needed.

```
enable compdef <component name>
```

- Bring up the `srvrmgr` console at server level.

```
srvrmgr /g siebgateway /e siebel /u sadmin /p sadmin /s <server name>
```

- Enable the component definition at the server level.

```
enable compdef <component name>
```

- Bounce the gateway and all active servers.

Note—Sometimes the component may not be enabled even after following the above steps. In this case, the component group may need to be enabled at the enterprise level before enabling the actual component:

```
enable compgrp <component group name>
```

Acknowledgments

This paper was originally written by Khader Mohiuddin, who worked on technology adoption projects across Oracle hardware and software technologies at Sun Microsystems. Khader has also worked in Oracle Consulting Services to help improve Oracle software performance in complex customer implementations. He also held senior engineering roles in the Oracle Server Technologies. Previously, Khader was an Oracle DBA and developer at AT&T Bell Labs, New Jersey for three years.

In addition, we would like to thank the following individuals and groups for their contributions:

- Oracle's Siebel staff, including Santosh Hasani, Mark Farrier, Francisco Casas, Farinaz Farsai, Vikram Kumar, Harsha Gadagkar, and others.
- Scott Anderson, George Drapeau, and Patric Chang for their roles as management and sponsors.
- Engineers in the Performance and Availability Engineering, Systems Group, and Data Management Group for their subject matter expertise in server and storage tuning and configuration.
- Oracle’s Enterprise Technology Center (ETC) staff for hosting the equipment and providing support.
- Engineers Mitesh Pruthi and Giri Mandalika, who executed the benchmark runs described in this technical white paper.

References

To learn more about Oracle’s Siebel products, refer to the documents and Web sites listed in Table 17 below, or email ssc_oracle@sun.com.

TABLE 17. RELATED WEB SITES AND DOCUMENTS.

DESCRIPTION	URL
IPGE Ethernet Device Driver Tunable Parameters (Sun Fire T2000 Server Only)	http://blogs.sun.com/roller/page/ipgeTuningGuide?entry=ipge_ethernet_device_driver_tunable
Sun Java System Webserver Performance Tuning, Sizing and Scaling Guide	http://docs.sun.com/source/816-5690-10/perf6.htm
Oracle Solaris Tunable Parameters Reference Manual	http://docs.sun.com/app/docs/doc/816-0607?q=kernel+tuning
Siebel Resource Library and Benchmark Reports	http://oracle.com/applications/crm/siebel/resources/siebel-resource-library.html
Sun Fire E2900 - E25K Servers Benchmarks	http://sun.com/servers/midrange/sunfire_e2900/benchmarks.html
Performance and Scalability Benchmark: Siebel CRM Release 7.7 Industry Applications on Sun Microsystems UltraSPARC Servers and Oracle 9i Database (64-Bit)	http:://oracle.com/apps_benchmark/doc/sun-siebel-8000-benchmark-white-paper.pdf
Performance and Scalability Benchmark: Siebel CRM Release 7.7 Industry Applications on Sun Microsystems UltraSPARC Servers and Oracle 9i Database (64-Bit)	http://oracle.com/apps_benchmark/doc/sun-siebel-12500_benchmark-white-paper.pdf
Sun Solution Centers	http://sun.com/solutioncenters
Sun Solution Center for Oracle	http://partner.sun.com/ssc-oracle
Siebel SupportWeb	https://ebusiness.siebel.com/supportweb

Appendix A: Transaction Response Times

Table 18 lists the different transactions executed by the OLTP workload of 8,000 Siebel users. The collection noted here is an average of all 8,000 users in steady state for one hour, executing multiple iterations with 30 second average think times between each transaction.

TABLE 18. TRANSACTIONS EXECUTED.

Scenario Name	E:\LR_7.8_15K_scenario\Scenario_8000_panther_db.lrs
Results in Session	f:\PSPP_Results-Phase_II\8000_panther-db3\8000_panther-db3.lrr
Duration	2 hours, 38 minutes and 32 seconds

Statistics Summary

TABLE 19. STATISTICS SUMMARY.

Maximum Running Vusers	8,000
Total Throughput (Bytes)	9,437,761,383
Average Throughput (Bytes/Second)	2,621,600
Total Hits	3,090,208
Average Hits/Second	858.391

Transaction Summary

TABLE 20. TRANSACTION SUMMARY.

Total Passed	2,226,269
Total Failed	10,932
Total Stopped	0

TABLE 21. TRANSACTION DETAILS

TRANSACTION NAME	MINIMUM	AVERAGE	MAXIMUM	STANDARD DEVIATION
Action_Transaction	0	55.896	233.25	69.462
Action1_Transaction	262.75	407.485	555.063	35.545
CC2_Iter1_101_ClickBinocularButton	0.031	0.171	15.734	0.355
CC2_Iter1_102_SelectContact	0	0	0	0
CC2_Iter1_103_EnterLastNameAnd Queary	0	0.12	10.438	0.3
CC2_Iter1_104_CloseSearchCenter	0	0.042	7.453	0.192
CC2_Iter1_105_GotoSR	0.047	0.299	12.859	0.482
CC2_Iter1_106_ClickNewSR	0.016	0.067	12.578	0.281
CC2_Iter1_107_ClickShowMoreButtonOnSRDetail	0.063	0.191	11.438	0.373
CC2_Iter1_108_ClickLastNamePickApplet	0.047	0.135	11.016	0.337
CC2_iter1_109_ClickQueryLastName	0	0.041	9	0.242
CC2_Iter1_110_EntereApps100AndQuery	0.031	0.099	10.953	0.297
CC2_Iter1_111_ClickOKtoSelectName	0.016	0.048	8.078	0.238
CC2_Iter1_112_ClickAccountPickApplet	0.047	0.14	11.484	0.355
CC2_Iter1_113_ClickQueryAccount	0	0.038	8.172	0.236
CC2_Iter1_114_Query_eApps_Account1	0.031	0.09	10.031	0.287
CC2_Iter1_115_ClickOKtoPickAccount	0.016	0.059	10.141	0.276
CC2_Iter1_116_ClickVerifyButton	0.063	0.182	15.047	0.415
CC2_Iter1_117_ClickOKToSelectEntitlement	0.047	0.136	11.156	0.333
CC2_Iter1_118_ClickPolicyPickApplet	0.016	0.083	9.516	0.297
CC2_Iter1_119_ClickQueryPolicy	0	0.034	4.766	0.068
CC2_Iter1_120_EnterInsgroup-204849111AndQuery	0.016	0.047	9.609	0.238
CC2_Iter1_121_ClickOKtoSelectPolicy	0	0.041	8.781	0.243
CC2_Iter1_122_ClickProductPickApplet	0.016	0.091	9.922	0.318

CC2_Iter1_123_EnterAbackAndQuery	0.063	0.202	11.938	0.368
CC2_Iter1_124_ClickOktoSelectProduct	0.031	0.106	9.75	0.273
CC2_Iter1_125_SaveSR	0.047	0.167	12.172	0.361
CC2_Iter1_126_ClickShowLessSR	0.047	0.164	9.75	0.331
CC2_Iter1_127_GotoSRActivityPlan	0.063	0.166	10.5	0.344
CC2_Iter1_127A_DrilldownOnSR	0.078	0.193	5.453	0.229
CC2_Iter1_128_NewActSRPlan	0.016	0.058	9.422	0.272
CC2_Iter1_129_SelectPlanAndSaveSR	0.344	0.852	17.016	0.878
CC2_Iter1_Pick_Type	0.047	0.151	11.25	0.35
Click_Service	0.109	0.321	17.172	0.658
Click_Site_Map	0.031	0.098	9.344	0.255
New_Oppty	0.172	0.455	16.625	0.743
New_SR	0.094	0.236	14.813	0.543
ResetStates	0	0.018	7.75	0.211
Save_Oppty	0.125	0.344	20.094	0.635
Save_SR	0.172	0.468	17.953	0.77

HTTP Response Summary

TABLE 22. HTTP RESPONSE SUMMARY.

HTTP RESPONSES	TOTAL	PER SECOND
HTTP_200	3,090,208	858,391

Appendix B: Database Object Growth During the Test

TABLE 23. DATABASE GROWTH SUMMARY.

SIEBEL OBJECT NAME	TYPE	GROWTH (BYTES)
S_DOCK_TXN_LOG	TABLE	1,177,673,728
S_EVT_ACT	TABLE	190,341,120
S_DOCK_TXN_LOG_P1	INDEX	96,116,736
S_DOCK_TXN_LOG_F1	INDEX	52,600,832
S_ACT_EMP	TABLE	46,202,880
S_SRV_REQ	TABLE	34,037,760
S_AUDIT_ITEM	TABLE	29,818,880
S_OPTY_POSTN	TABLE	28,180,480
S_ACT_EMP_M1	INDEX	25,600,000
S_EVT_ACT_M1	INDEX	23,527,424
S_EVT_ACT_M5	INDEX	22,519,808
S_ACT_EMP_U1	INDEX	21,626,880
S_ACT_CONTACT	TABLE	21,135,360
S_EVT_ACT_U1	INDEX	18,391,040
S_ACT_EMP_M3	INDEX	16,850,944
S_EVT_ACT_M9	INDEX	16,670,720
S_EVT_ACT_M7	INDEX	16,547,840
S_AUDIT_ITEM_M2	INDEX	16,277,504
S_ACT_EMP_P1	INDEX	15,187,968
S_AUDIT_ITEM_M1	INDEX	14,131,200
S_EVT_ACT_F9	INDEX	13,852,672
S_ACT_CONTACT_U1	INDEX	13,361,152

S_ORDER_ITEM	TABLE	13,066,240
S_REVN	TABLE	12,943,360
S_CONTACT	TABLE	12,779,520
S_SRV_REQ_M7	INDEX	12,492,800
S_SRV_REQ_M2	INDEX	11,960,320
S_ACT_EMP_F1	INDEX	11,804,672
S_SRV_REQ_U2	INDEX	10,731,520
S_SRV_REQ_U1	INDEX	10,444,800
S_DOC_QUOTE	TABLE	10,321,920
S_QUOTE_ITEM	TABLE	9,666,560
S_SRV_REQ_M9	INDEX	8,970,240
S_ACT_CONTACT_F2	INDEX	8,716,288
S_OPTY	TABLE	8,396,800
S_ACT_CONTACT_P1	INDEX	8,183,808
S_AUDIT_ITEM_F2	INDEX	7,987,200
S_ORDER	TABLE	7,987,200
S_SRV_REQ_F13	INDEX	7,905,280
S_SRV_REQ_P1	INDEX	7,872,512
S_SRV_REQ_M10	INDEX	7,823,360
S_RESITEM	TABLE	7,798,784
S_AUDIT_ITEM_P1	INDEX	7,634,944
S_REVN_U1	INDEX	7,454,720
S_SRV_REQ_M8	INDEX	7,331,840
S_SRV_REQ_M3	INDEX	7,208,960
S_SRV_REQ_F6	INDEX	7,135,232
S_SRV_REQ_F1	INDEX	7,086,080

S_REVN_M1	INDEX	7,004,160
S_OPTY_U1	INDEX	6,676,480
S_REVN_U2	INDEX	6,676,480
S_EVT_ACT_F11	INDEX	6,602,752
S_OPTY_TERR	TABLE	6,569,984
S_SRV_REQ_M6	INDEX	6,471,680
S_SRV_REQ_F2	INDEX	5,611,520
S_DOC_ORDER	TABLE	4,972,544
S_SR_RESITEM	TABLE	4,972,544
S_ORG_EXT	TABLE	4,833,280
S_ACCNT_POSTN	TABLE	4,341,760
S_OPTY_CON	TABLE	4,136,960
S_DOC_QUOTE_BU	TABLE	4,096,000
S_PARTY	TABLE	4,096,000
S_SRV_REQ_M5	INDEX	4,055,040
S_POSTN_CON	TABLE	3,932,160
S_SRV_REQ_F7	INDEX	3,768,320
S_SRV_REQ_M4	INDEX	3,563,520
S_OPTY_BU	TABLE	3,194,880
S_REVN_M3	INDEX	3,162,112
S_CONTACT_M13	INDEX	3,153,920
S_OPTY_U2	INDEX	3,072,000
S_REVN_U3	INDEX	3,031,040
S_OPTY_BU_M9	INDEX	2,990,080
S_OPTY_BU_P1	INDEX	2,949,120
S_PARTY_M2	INDEX	2,949,120

S_ORDER_BU_M2	INDEX	2,867,200
S_OPTY_BU_U1	INDEX	2,744,320
S_ORG_EXT_F1	INDEX	2,629,632
S_CONTACT_M11	INDEX	2,621,440
S_CONTACT_M21	INDEX	2,621,440
S_CONTACT_M14	INDEX	2,621,440
S_PARTY_M3	INDEX	2,621,440
S_CONTACT_F6	INDEX	2,580,480
S_OPTY_V2	INDEX	2,580,480
S_RESITEM_M4	INDEX	2,547,712
S_REVN_F6	INDEX	2,539,520
S_ORDER_M5	INDEX	2,498,560
S_PARTY_M4	INDEX	2,498,560
S_REVN_M2	INDEX	2,498,560
S_POSTN_CON_M1	INDEX	2,498,560
S_CONTACT_M12	INDEX	2,416,640
S_OPTY_BU_M1	INDEX	2,416,640
S_CONTACT_M22	INDEX	2,416,640
S_OPTY_BU_M2	INDEX	2,334,720
S_OPTY_BU_M5	INDEX	2,334,720
S_OPTY_BU_M6	INDEX	2,334,720
S_OPTY_BU_M8	INDEX	2,334,720
S_ORDER_POSTN	TABLE	2,334,720
S_OPTY_BU_M7	INDEX	2,334,720
S_CONTACT_M9	INDEX	2,293,760
S_CONTACT_X	TABLE	2,293,760

S_EVT_ACT_M8	INDEX	2,252,800
S_OPTY_BU_M4	INDEX	2,211,840
S_RESITEM_U2	INDEX	2,138,112
S_RESITEM_M5	INDEX	2,097,152
S_RESITEM_M6	INDEX	2,097,152
S_ORDER_BU	TABLE	2,088,960
S_REVN_F3	INDEX	2,088,960
S_DOC_QUOTE_U1	INDEX	2,048,000
S_RESITEM_U1	INDEX	2,023,424
S_OPTY_BU_M3	INDEX	1,966,080
S_REVN_P1	INDEX	1,966,080
S_REVN_F4	INDEX	1,966,080
S_DOC_QUOTE_U2	INDEX	1,925,120
S_SR_RESITEM_U1	INDEX	1,892,352
S_POSTN_CON_M2	INDEX	1,843,200

Oracle is committed to developing practices and products that help protect the environment

Optimizing Oracle's Siebel Application on
Oracle Servers with CoolThreads Technology
April 2010
Author: Khader Mohiuddin

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
oracle.com

SOFTWARE. HARDWARE. COMPLETE.