An Oracle White Paper
September 2010

Best Practices Guide for Deploying Oracle WebLogic Suite on Sun Systems

ORACLE

About This Guide 2
Audience and Scope 2
Introduction 4
Application Design Considerations 4
Application Overview 4
Application Workflow 5
Oracle Coherence Caching Choices 5
Database Structure 7
Application Deployment Considerations 8
Architecture Overview 8
Application Cluster Tier 9
Database Tier 10
Scaling with the Top Of Rack Switch 10
Tuning Recommendations 11
Oracle Solaris on Sun Systems 11
Oracle WebLogic Server 12
Oracle TopLink Grid 13
Oracle Coherence 13
Oracle Sun Java Virtual Machine 15
Oracle Database 16
Conclusion 17
For More Information 17
Related Resources 17
About the Authors 18
Acknowledgments 18

About This Guide

This document describes best practices that can be applied to develop and deploy Java Platform Enterprise Edition (Java EE) applications using Oracle WebLogic Suite running on Oracle Sun's Systems. Java Enterprise Application architecture is revisited using a sample application that captures the essential design decisions and configurations which impact performance and scalability. Best practice guidelines are provided for the following products:

- Oracle Solaris Operating System
- Oracle WebLogic Server
- Oracle Coherence with TopLink Grid
- Oracle Hotspot JVM 6

Audience and Scope

Developing and deploying a scalable application is challenging; this document presents insights and best practices for Java EE developers and system administrators looking for scalable solutions. Familiarity with developing Java EE applications, managing WebLogic Application server, and Coherence server running in Oracle Solaris environment is assumed. A basic knowledge of Java Persistence API (JPA) is required to understand the caching options that are chosen for the reference application. The following topics are

discussed:

- Using Coherence as the JPA L2 cache.
- Tuning a typical Java EE application that uses TopLink Grid along with EclipseLink JPA and caches entities in a Coherence Grid.
- Grid Read and Grid Entity caching options using TopLink Grid with EclipseLink JPA.
- Available options in providing HTTP Session fail-over while using WebLogic Suite.
- Calculating Coherence Cache size requirements.
- Other Coherence tuning options that help improve performance.
- Tuning options for Java SE, WebLogic Suite and Oracle Database.
- Sun servers selected for Application, Cache and Database tiers.

Introduction

Exponential growth in user base and amount of consumptive data created has transformed almost every organization. The applications that previously served hundreds of internal users for everything from shipment tracking to financial transactions are being transformed to support millions of consumers. These large systems are not only expected to scale but also serve users under the defined Quality of Service, which requires a new way to organize, query and store user data closer to the applications.

Today enterprise architects have embraced several technologies such as data grids that support automatic data partitioning, processing and querying. Oracle WebLogic Suite 11g which includes Oracle WebLogic Server (Application Server), Oracle Coherence (Data Grid) and TopLink (EclipseLink JPA connecting technology) is one such solution. Oracle WebLogic Suite dynamically partitions data, moves data closer to applications to reduce latency and improve performance, and offers a simple means to increase the capacity of shared data resources. Through these capabilities, WebLogic Suite helps maximize web facing application performance and can help achieve near-linear scalability.

Our previous performance white paper "Oracle's Sun Systems for Oracle Coherence:An Optimal In-Memory Data Grid Architecture" demonstrated that the solution with Oracle Coherence offers up to 2.5 times faster queries and transaction per second. Secondly, it was also shown that by adding an Oracle Coherence tier, organizations can reduce CPU utilization in the database tier by up to 40% and support up to 25% more users thereby increasing the total capacity of the system. This white paper compliments the previous one by discussing another implementation using Oracle WebLogic Suite running on Oracle's Sun systems. A sample Java EE application is introduced and adapted to Oracle WebLogic Suite capturing essential design decisions one has to make. Finally, best practices derived from the study are shared. Organizations can take advantage of the pre-tested architecture and best practices offered in this paper to help speed time to deployment for similar or adaptable workloads.

Application Design Considerations

Application Overview

In order to best illustrate the design choices, a sample application – an online auction-style application is presented. This application is an online auction and shopping website in which people and businesses buy and sell a broad variety of goods and services worldwide. **Auction-style listings** allow the seller to offer one or more items for sale for a specified number of days. Bidding on listed items is based on auction-style listings where the winning bidder pays the second-highest bid plus one bid increment amount (i.e., some small predefined amount relative to the bid size). Also, the current winning bid is not sealed, but instead is always displayed.

Once the auction item is created, users can place a bid against the item until the close time at which point the auction item is closed and declared sold. Figure 1 provides a high level view of the application and the data flow .

Application Workflow

As described in the application overview section the auction application provides for two kinds of users.

1. **Seller** - Performs a login after creating an account and then creates a new auction item providing the details such as name, category, photo, minimum bid, etc. All these attributes are captured in the auction *Item* object which is cached on the Oracle Coherence server and then saved in the database. The size of a typical item listed for auction is approximately 5KB. This includes a thumbnail photo of the listed item.
2. **Bidder** - First performs a login then randomly selects or views an auction item from the list of available auction items from the database. If the auction is not closed on the item and there is a bid price available, the bidder places a bid on the auction item. This results in creating a *Bid* record associated with the auction item that gets cached on the Oracle Coherence server.

User	Action	Results in
Seller	Login	Authenticate using Username and Password
	List Item	Create Auction Item Insert Item into Database Put or Insert Item into Cache
Bidder	Login	Authenticate using Username and Password
	View Item	Get Item from Cache
	Bid on Item	Insert Bid into Database Update Item in Database Put Item and Bid into Cache

TABLE 1: WORKFLOW TABLE

A workload was created using the Faban load generator that runs through the Seller's and Bidder's entire workflow mentioned in Table 1. For more information on the open-source Faban load generator, see http://faban.sunsource.net. This workload was used to test the performance of the application and report the best practices and tuning recommendations.

Oracle Coherence Caching Choices

A rapid rise in an application's popularity can impose significant load on the application servers. The application server has to process the incoming requests, authenticate users, fetch the required relational data from the data source, process it and then finally respond back to the requested action. Running through this entire cycle for every similar request is expensive and can reflect on the application response time.

Oracle Coherence provides a way to bring data closer to the application server and retain it in a consumable object form. Further, these cached objects are available to all the WebLogic application server instances in the cluster thereby reducing the redundant object relational mapping calls. Also, the objects can be replicated to support fail over. The following section introduces the caching options

available in Oracle Coherence, primarily the HTTP session caching using Coherence*Web and Entity Caching using Oracle Coherence and TopLink JPA.

HTTP Session Caching

Coherence*Web is an HTTP session management module dedicated to managing session state in clustered environments. Built on top of Oracle Coherence, Coherence*Web enables **session sharing** and management across different web applications, domains and heterogeneous application servers. Also, it brings Oracle Coherence data grid's **data scalability, availability, reliability** and **performance** to in-memory session management and storage.

Coherence*Web supports three kinds of HTTP Session Models to store session data:

1. **Traditional Model**: Stores all session state as a single entity, serializing and deserializing session attributes individually. This model is best suited for applications that only update certain attributes.
2. **Monolithic Model**: Stores all session state as a single entity, serializing and deserializing all attributes as a single operation. Most suited for applications that rewrite or update the entire session every time.
3. **Split Model**: Extends Traditional Model, but separates the larger session attributes into independent physical entities (see Figure 2). This model applies to applications with large attributes such as a user image or a location map.

Among the three types of available session models, the Split Session Model was selected for the auction application deployment as that provided the desired flexibility. One of the session attributes is larger than 1 KB while all the other attributes are less than 256 bytes.

Once the session attributes are defined and a model is chosen, the deployment topology which defines where the session resides during the application run time has to be decided. The following choices are available for the Session Deployment Topologies:

1. **In-Process**: Also known as "local storage enabled", is where session data is stored in-process within the application server.
2. **Out-of-Process**: Also known as "local storage disabled", is where the application servers are configured as cache clients and dedicated JVMs run as cache servers, physically storing and managing the clustered data. Out-of-Process also supports session sharing across data centers using Coherence*Extend which is based on TCP/IP.

The auction application caches HTTP Session data on the Oracle Coherence Server nodes using the Out-of-Process deployment topology. Profiling results from the test have shown that caching sessions in Coherence*Web performs better compared to saving session data in a database. In the Out-of-Process deployment topology, the application servers are configured as Oracle Coherence clients with local storage set to false (i.e, tangosol.coherence.distributed.localstorage=false) and there are dedicated JVMs running cache servers that store and manage clustered data.

A backup count of 1 is maintained for all the HTTP sessions in the Oracle Coherence servers. Also no special tuning was done for Session Locking Modes. Coherence*Web and Coherence*Web SPI are configured with Optimistic Locking by default. The auction application deployment simply used the default locking mode. Although initial test runs had monitoring enabled for Oracle Coherence servers, final runs had the monitoring turned off. Please see the tuning section for additional notes on how to enable monitoring for viewing using JConsole or a web browser.

Entity Caching

TopLink Grid is a combination of EclipseLink Java Persistence API (JPA) and Oracle Coherence data grid that provides retrieval and retention of data objects in one of the configurations mentioned below:

1. **Grid Cache**: Oracle Coherence as Shared (L2) cache replacement.
2. **Grid Read**: All reads by default fetch data from Oracle Coherence cache.
3. **Grid Entity**: In this mode, Oracle Coherence cache becomes the system of record. All the reads and writes happen directly on the Oracle Coherence cache.

The EclipseLink JPA in the auction application uses Oracle Coherence in Grid Cache configuration. At run time, any access to the bid and auction item entities on the Oracle WebLogic Application Server causes a cache look-up and fetch on the Oracle Coherence Server nodes. In Grid Cache configuration, only primary keys are looked up in the database. If the key is present in the server cache, it will be fetched from the cache there by saving the object construction costs. All updates are committed in the database first and then cached in the Oracle Coherence server node. Applications with heavy read operations tend to benefit with the Grid Cache option.

Database Structure

The Auction application uses a total of 24 database tables. Profiling the database while running the workload provided the top sql queries and the most heavily used tables (see Table 2). This motivated the caching of the corresponding entities in Oracle Coherence cache.

Database Tables
Seller
Bidder
Auction Item
Thumbnail Photos

TABLE 2: DATABASE TABLES

For the Auction application, much of the data used resides primarily in the Oracle Coherence cache layer. However, the database is always consistent as JPA commits the auction item updates to the database. There is a performance penalty as the JPA synchronously commits to the database every time. This can be overcome by using a write behind configuration which asynchronously commits the modified data back to the database.

Application Deployment Considerations

Web services and the sudden rise in Internet use are driving organizations to rapidly scale the supporting infrastructure without thinking about the ability of the software to handle a ten, twenty, or hundred fold increase in the number of active users. The following sections offer a description of a modular, balanced, and pre-tested architecture that is optimized for deployments with Oracle WebLogic Suite.

Architecture Overview

Figure 3 depicts the architecture used which consists of the following three basic layers:

- Client Tier - Running heterogeneous clients driving load into the application tier. Note that in a production deployment this tier will be replaced by a combination of firewalls, load balancers, reverse proxy servers, etc.
- Application Server Cluster Tier - Running the actual Java EE application deployed on Oracle WebLogic Suite.
- Database Tier - Hosting the Oracle database for the application data.

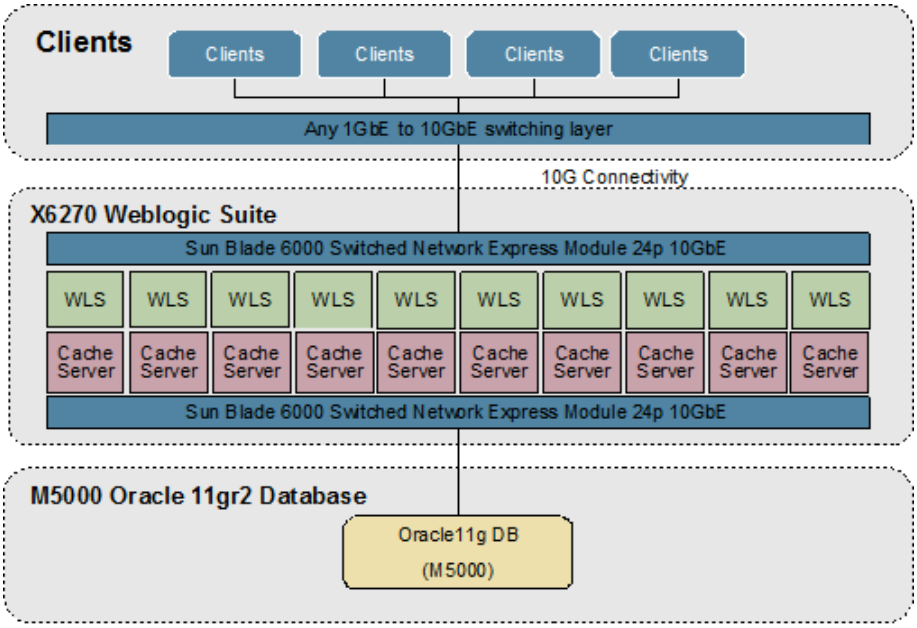

Figure 1: Auction Application Deployment

Each of the individual tiers and components within those tiers are discussed in the sections that follow. Note that all servers run Oracle Solaris 10 Operating System. The software components used and their respective versions numbers are listed in Table 3 below.

Tier	Software	Description	Version

Application Cluster Tier	Oracle WebLogic Server	Application Server	10.3.3
	Coherence*Web SPI	Oracle WebLogic Server HTTP Session Cache	3.5.3
	Oracle TopLink Grid	Integrates with Oracle Coherence and supports EclipseLink JPA	11gR1
	Oracle EclipseLink	Java Persistence API	2.0.2.v20100323-r6872
	Oracle Coherence Grid Edition	Cache Client/Server	3.5.3
	Oracle JVM	HotSpot JVM	6 u21
	Oracle Solaris OS	Operating System	10 Update 8
Database Tier	Oracle Database	Database	11.2.0.1.0
	Solaris OS	Operating System	10 Update 8
Client Tier	Faban	Load Driver / Harness	1.0.1
	Oracle Coherence Grid Edition	Java Client	3.5.3

TABLE 3: SOFTWARE COMPONENTS AND VERSIONS

Application Cluster Tier

A Sun Blade 6000 Modular System with ten Sun Blade X6270 server modules and Oracle's SunBlade 6000 Switched Network Express Module 24p 10GbE provides compute power for the WebLogic Suite Tier. Each node contains two quad-core x86 processors running at 3.2GHz, 48 GB of memory per node, and executes the Java™ Virtual Machines upon which Oracle WebLogic Suite runs. **Memory speed**, **density**, and **CPU processing speed** are the most important factors for WebLogic Suite performance and scalability. Based on a number of internal tests that compared various processing architectures, these systems offer the best balance between memory density, and CPU processor speed, and high performance for WebLogic Suite workloads running Enterprise 2.0 and Web 2.0 applications as the memory controller on these processors talks directly to its associated bank of memory first, before looking to a more distant (and therefore slower) resource.

Interconnect technology plays an important role between each server tier. In particular, utilizing a high-bandwidth, low-latency interconnect between the cluster nodes is critical to reaching top performance. This sample application uses a novel full featured integrated switching module - Oracle's Sun Blade 6000 Switched Network Express Module (NEM) 24p 10GbE Switched provides latencies as low as 300ns. Each Switched NEM provides ten internal 10GbEports for in-chassis switching between the blades and sixteen 10GbE ports for external connectivity using 3 Quad Small Form Factor Pluggable (QSFP) ports and 4 SFP+ 10GbE ports.

Two Sun Blade 6000 24p 10GbE Ethernet switch NEMs are used, one NEM provides 10 GbE connectivity for each server module to the switch that connects the load-generating client systems. A second NEM provides a connection for each server module to the 10 GbE switch attached to the database server. On each server module, the WebLogic Server instances communicate with the TopLink (Eclipse Link) layer, a Java Persistence Architecture (JPA) implementation for storing and accessing objects between application cluster and Oracle Coherence cluster.

Database Tier

The auction application uses Oracle Coherence as a L2 cache for objects required by the application servers. However, the transactions within the Coherence tier did not require immediate commitment and notification of committed writes to the back-end database. Therefore, the interconnect speed to the database was not critical to performance in this configuration. As shown in Figure 3, the Database 11g instance running on the Sun SPARC Enterprise M5000 server is connected to the application tier's Sun Blade 6000 Ethernet Switch NEM's SFP+ port via two 10 GbE add-in PCI Express cards for redundancy. The load on the database is not excessive, and since Oracle Coherence handles logical redundancy, the transactions that need to be stored in the database do not need to be immediately committed.

Scaling with the Top Of Rack Switch

This section describes a larger configuration using Oracle's Sun Network 10GbE 72p Top Of Rack (ToR) switch. ToR with 16 QSFP ports and 8 10GbE ports offers scalability up to 72 servers or 6 SB6000 chassis using the Switched 10GbE NEM. Network micro-benchmark tests running TCP TX/RX performance tests have shown that ToR switch supports linear scalability up to 60 blade servers while maintaining low latencies. An example four chassis solution is provided in Figure 4.

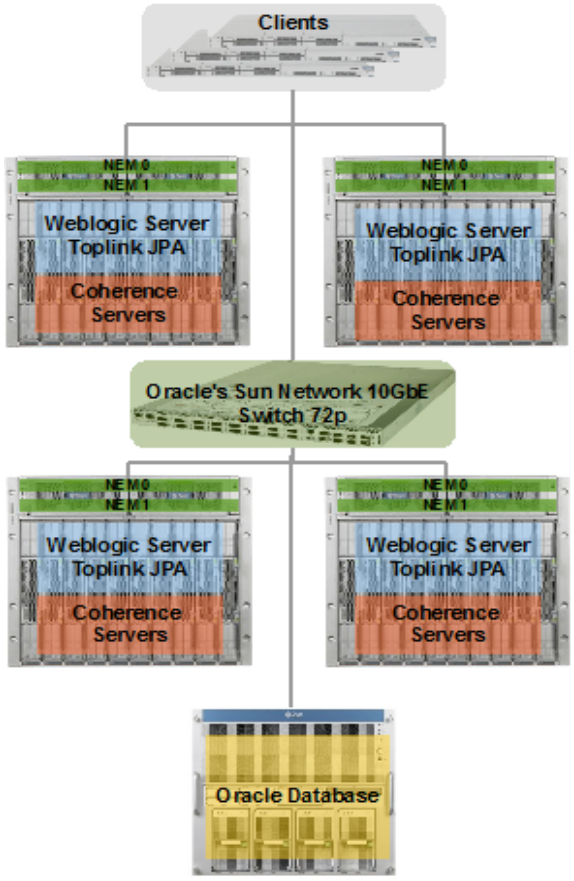

Figure 2: Scaling using Sun Network 10GbE Switch 72p

Tuning Recommendations

Oracle Solaris on Sun Systems

Solaris default network tuning parameters should be sufficient in most cases. For heavy, network-centric workloads, it is recommended to adjust the following parameters. The following settings were used:

```
# Save in /etc/rc2.d/S90ndd


#TCP Tunable Parameters
#
# Following two parameters are associated with the maximum
# number of requests that are associated with per IP address per
```

```
# port.  Increase these values if you do see non-zero values for
# tcpListenDrop in the output for the following command :
# netstat –s | fgrep –i listendrop
# increase the values in steps of 256 starting from the default value
# refer to: http://blogs.sun.com/terrygardner/entry/solaris_tcp_ip_parameters_tcp
ndd -set /dev/tcp tcp_conn_req_max_q 16384
ndd -set /dev/tcp tcp_conn_req_max_q0 16384

# the following two parameters are calculated based on the network
# bandwidth and latency. You could follow this rule to calculate:
# <numbers of bits per second> * <roundtrip latency> * 8 [bits/byte]
ndd -set /dev/tcp tcp_xmit_hiwat 524288
ndd -set /dev/tcp tcp_recv_hiwat 524288

# The following parameter helps in the case of small packets sent
# between the cache server and the application servers.
ndd -set /dev/tcp tcp_naglim_def 1

# Specifies the time in milliseconds that a TCP connection stays
# in TIME-WAIT state
ndd -set /dev/tcp tcp_time_wait_interval 10000
```

Oracle WebLogic Server

Applications deployed on Oracle WebLogic Server can take advantage of several built-in features of the application server. For example, there is a choice in tuning the number of threads manually or it could be left to Oracle WebLogic Server to tune that automatically. Similarly, there are several options to tune the JMS and JDBC resources. Based on the performance tests data, it was decided that going with manually tuned threads was better as the load size is well-known and the required thread count could be optimized. The downside to the manual tuning is that if there is ever a sudden spike in load, the server could get overwhelmed and may not be able to handle the load as the thread-count is fixed.

```
#weblogic server config.xml
    <execute-queue>
       <name>default</name>
       <thread-count>60</thread-count>
    </execute-queue>
    <use81-style-execute-queues>true</use81-style-execute-queues>
```

If you decide to stay with the default auto tuning mode for the threads, it is recommended to use the Work Manager available since WebLogic Server 9.0.

WebLogic JDBC tuning has several parameters to optimize. Depending on the workload, ***PinnedToThread=true*** may be useful. Consider tuning only the required number of connections for the JDBC connection pool. Oracle WebLogic Server can cache prepared and callable statements that can considerably minimize processing costs. This is dictated by the statement cache size which should be tuned.

```
# weblogic server JDBC tuning <domain>/config/jdbc/datasource.xml
  <jdbc-connection-pool-params>
     <initial-capacity>32</initial-capacity>
     <max-capacity>60</max-capacity>
     <capacity-increment>4</capacity-increment>
     <statement-cache-size>128</statement-cache-size>
     <statement-cache-type>LRU</statement-cache-type>
  </jdbc-connection-pool-params>
```

Oracle WebLogic Server was deployed with Oracle Coherence client ,i.e. with *localstorage* disabled. The following options were passed to the WebLogic Server startup script.

```
# weblogic server startup script...coherence startup parameters
# local storage was set to false. However depending on the application enabling this
# feature may benefit in some cases.
-Dtangosol.coherence.distributed.localstorage=false
```

Oracle TopLink Grid

The auction application uses TopLink Grid that supports EclipseLink JPA in order to take advantage of the Oracle Coherence cache. Entities like Auction Items were accessed using the Grid Cache configuration. In this configuration, Oracle Coherence is used as the JPA Shared (L2) cache. This helps in minimizing the load on the database and also minimizes the object construction costs. The other available options are Grid Read and Grid Entity. While Grid Read performs all reads on the Coherence cache, Grid Entity essentially transforms Oracle Coherence to be the system of record and substantially reduces the database load.

Oracle Coherence

Oracle Coherence was run separately as cache server nodes. The following tunings were applied:

```
#Coherence Server nodes tuning


-server -Xmx3200m -Xms3200m -Xmn256m -XX:+UseLargePages
-XX:+AlwaysPreTouch -XX:+PrintCommandLineFlags -XX:SurvivorRatio=9 -XX:MaxPermSize=128m -
```

```
XX:+PrintGCDateStamps -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:ParallelGCThreads=8 -
XX:+CMSScavengeBeforeRemark -XX:CMSInitiatingOccupancyFraction=70 -XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -verbose:gc


# Coherence specific tuning


-Dtangosol.coherence.distributed.localstorage=true -
Dtangosol.coherence.session.localstorage=true
 -Dtangosol.coherence.distributed.threads=10
-Dtangosol.coherence.invocation.threads=5
```

Cache Size Calculations

To calculate the size of memory needed to cache all the data, the following checklist was used :

- Number of objects that need to be cached to maintain 99% cache hit ratio
- Size of the objects and the total size of the cache needed
- Is backup enabled and the number of copies to be maintained
- Size of the session objects and the number of sessions to be cached along with the number of backup copies.
- Additional space on the heap for short-lived objects created during the serialization and deserialization of these objects while sending them over the network to the coherence client nodes.

```
Data Size = Number of Entities * Entity Size
Session Size =  Number of Session Objects * Session Size
Total Cache Size = (Data Size + ( Data Size * Backup Number) +
      Session Size + ( Session Size * Backup Number ) ) * 2
```

```
Database Records: 500,000
Database Record Size: 512 bytes
Avg. Session Size: 512 bytes
Total Sessions: 50,000


Data Size     = 500000 * 512   = 256.0MB
Session Size =   50000 * 512   =   25.6MB
Backup Count for everything = 1


Cache Size = (256 + 256 * 1) + (25.6 + 25.6*1)
           = 563.2MB
```

Accounting for Serialization and Deserialization:

Total Cache Size = Cache Size * 2
= 563.2 * 2
= 1126.4 MB

At least 3 Java VM instances of 512MB heap each should be able to cache the above data set and sessions.

Other Oracle Coherence Tunable Parameters

The following parameter specifies the number of daemon threads used by the distributed cache service. Since the default is set to 0 that causes the tasks to be run on the service thread, this value was adjusted to 10.

```
-Dtangosol.coherence.distributed.threads=10
```

The following parameter specifies the number of daemon threads used by the invocation service. Since the default is set to 0 that causes the tasks to be run on the service thread, this value was adjusted to 5.

```
-Dtangosol.coherence.invocation.threads=5
```

Oracle Sun Java Virtual Machine

Oracle Java SE 1.6.0_18 was used in all the tests. It is recommended to use the CMS garbage collector as Oracle Coherence is very sensitive to GC pauses that are longer than a second. The GC pauses should always be kept to less than half a second.

There is no single tuning parameter for Oracle Java that would simply solve the GC pause times. It is necessary to experiment with some of the options provided below until the GC pauses are in the acceptable range i.e., typically less than a second.

```
# WebLogic Server JVM tuning
-server -Xms2800m -Xmx2800m -Xmn1024m -Xss128k
-XX:+CMSScavengeBeforeRemark -XX:CMSInitiatingOccupancyFraction=70
-XX:+UseParNewGC -Xloggc:./gclog.txt -XX:ParallelGCThreads=8 -XX:LargePageSizeInBytes=4m -
XX:SurvivorRatio=10
-XX:+UseConcMarkSweepGC -XX:PermSize=128m -verbose:gc
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+PrintCommandLineFlags
-XX:+UseLargePages
-XX:+AlwaysPreTouch
```

Start with the following options:

```
-XX:+UseParNewGC, -XX:+UseConcMarkSweepGC
```

and add additional parameters based on the GC behavior. If objects are getting promoted to tenured space without going through the survivor spaces, then increasing the survivor space helps. Sometimes majority of the objects do not survive beyond 3 or 4 full GC events. In that case having survivor spaces large enough to keep objects until the 3rd or 4th GC would help in reducing the number of old GC collection pauses. Further information regarding Hotspot GC Tuning can be found here: Tuning Garbage Collection.

Oracle Database

Tuning the Oracle database improves the performance during the initial warm-up phase of the coherence caches. However, once the data is cached, database load reduced quite significantly. Oracle was tuned to have sufficient buffers in the System Global Area (SGA).

```
# tune the oracle SGA to be sufficient for your data size
*.compatible='11.1.0.6.0'
*.control_files='/db_hdd/ora-ts/cntrlspecdb'
*.db_block_checking='FALSE'
*.db_block_checksum='FALSE'
*.db_block_size=8192
*.db_cache_advice='off'
*.db_cache_size=32G
*.db_file_multiblock_read_count=128
*.db_files=256
*.db_name='auctiondb'
*.db_writer_processes=8
*.disk_asynch_io=TRUE
*.dml_locks=512
*.filesystemio_options='setall'
*.log_buffer=100663296
*.log_checkpoint_interval=0
*.log_checkpoints_to_alert=TRUE
*.open_cursors=1200
*.parallel_max_servers=100
*.processes=1200
*.query_rewrite_enabled='false'
*.sessions=1200
*.shared_pool_size=6G
*.statistics_level='BASIC'
*.timed_statistics=FALSE
*.trace_enabled=FALSE
*.transactions=1200
```

```
*.transactions_per_rollback_segment=1
*.undo_management='AUTO'
*.undo_retention=600
*.undo_tablespace='undo_ts'
```

Conclusion

Java EE applications can be scaled beyond the capacity constraints of the database system by utilizing newer technologies like Oracle Coherence. This document shows how to plan for incorporating Oracle Coherence and Coherence*Web SPI while deploying WebLogic Suite on Sun Servers. The document also explains how to tune various portions of the deployment including Oracle Solaris, Oracle WebLogic Server, Oracle Coherence, Sun Java Virtual Machine, and the Oracle database.For a given existing database infrastructure, the number of concurrent users supported by the existing middleware can be easily scaled up to 3 times using Oracle Coherence for caching http session and database records.

For More Information

For more information on the benefits of Oracle Coherence and Oracle Sun Servers, please see the related resources below.

Related Resources

- Oracle Sun Blade Systems
  *http://www.*Oracle.*com/us/products/servers-storage/servers/blades*
- Oracle Sun Networking Technologies
  *http://www.*Oracle.*com/us/products/servers-storage/networking*
- Sun JVM Tuning Guide
  http://java.sun.com/performance/reference/whitepapers/tuning.html
- Oracle Coherence
  *http://www.*Oracle.*com/goto/coherence*
- Oracle Coherence Technical Information on Oracle Technology Network
  *http://www.*Oracle.*com/technology/products/coherence*
- Defining an In Memory Data Grid (IMDG)
- *http://www.jroller.com/cpurdy/entry/defining_a_data_grid*
- Oracle Coherence Distributed Auction application with push-replication
- *http://coherence.oracle.com/display/INCUBATOR/Auction+Site+Demo+1.0.0*
- Oracle Coherence*Web and Coherence*Web SPI HTTP Session Models

- *http://download.oracle.com/docs/cd/E15357_01/coh.360/e15829/features.htm#CIHHFBAJ*
- Oracle Coherence*Web and Coherence*Web SPI deployment topologies
- *http://download.oracle.com/docs/cd/E15357_01/coh.360/e15829/features.htm#CIHGJHIC*
- Tuning Garbage Collection with the Java Virtual Machine
- *http://www.oracle.com/technetwork/java/gc-tuning-5-138395.html*
- WebLogic Server JDBC Configuration and Tuning
- *http://download.oracle.com/docs/cd/E14571_01/web.1111/e13737/toc.htm*
- Solaris 10 TCP Tunable parameters
- *http://docs.sun.com/app/docs/doc/817-0404/chapter4-31?l=en&a=view*
- WebLogic Server Performance Tuning
- *http://download.oracle.com/docs/cd/E14571_01/web.1111/e13814/toc.htm*
- Coherence command line settings
- *http://coherence.oracle.com/display/COH35UG/Command+Line+Setting+Override+Feature*

About the Authors

Satish Vanga serves as the technical lead for Fusion Middleware, specifically Coherence Grid within the Developers, Performance and Applications group at Sun. Satish has published several SPECjAppServer benchmarks and developed benchmarks for several Fusion Middleware products including Oracle Coherence. His expertise is in massively scalable systems designed with distributed caching engines such as Oracle Coherence, Terracotta, and Oracle TimesTen In-Memory Database.

Nitin Ramannavar is a part of the core Performance Engineering team at Sun where his responsibilities include improving end-to-end system and application performance. Nitin specializes in Web technologies with emphasis on distributed computing, scale-out application architectures using virtualization, Web and application caching, and providing cloud services with guaranteed quality of service.

Acknowledgments

The authors would like to thank :

- Lalit Bhola from Oracle Sun Networking team for his valuable insights into the 10GbE networking using OPUS NEM and OPUS ToR switch.

• Shanti Subramanyam, and Eric Reid from Oracle Developers, Performance and Applications team for their feedback in arranging the technical information in a easy to understand flow.

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

Best Practices Guide for Deploying Oracle Weblogic Suite on Sun Systems

September 2010
Authors: Satish Vanga, Nitin Ramannavar

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.