Siebel on Exadata

Oracle Maximum Availability Architecture White Paper
October 2010

Maximum Availability Architecture

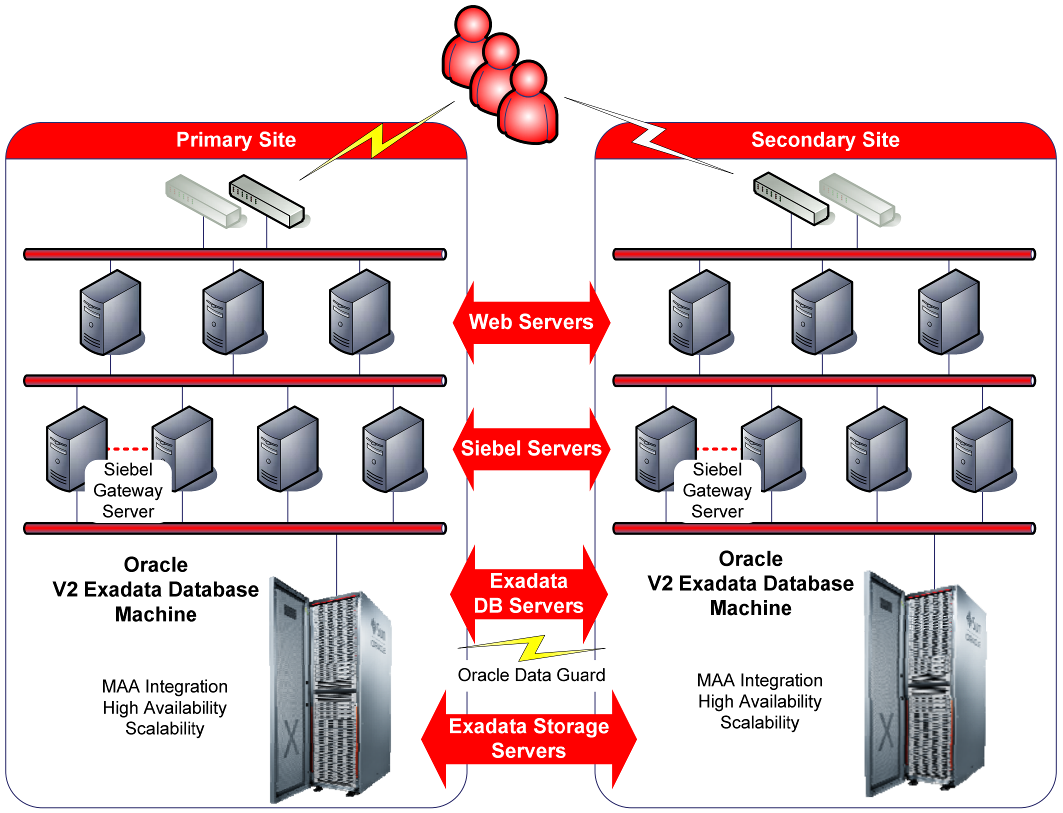
Oracle Best Practices For High Availability

ORACLE

Executive Overview

The Sun Oracle Database Machine is an ideal database platform for data warehouse and OLTP applications, and for database consolidation. It is a scalable grid of industry standard servers – compute and storage – linked by a fast InfiniBand fabric. All Database Machines are highly optimized and come pre-configured and ready to run. This combination of features makes the Sun Oracle Database Machine an easy to deploy solution for hosting a Siebel Database and delivers high levels of performance and availability.

This paper describes the best practices for moving an existing Siebel implementation to a Sun Oracle Database Machine and achieving excellent performance and maximum availability.



To develop and validate our best practices, Siebel 8.0.0.8 Financial Services application was deployed on a Quarter Rack Sun Oracle Database Machine in a Maximum Availability Architecture (MAA) configuration. MAA is Oracle's best practices blueprint for implementing Oracle high-availability technologies, including RAC, Data Guard, and Flashback.

The database behavior and performance was evaluated under several test workloads with the following results:

- 30,000 concurrent users running a combination of call center, partner service, and web service users were able to process 443,334 business transactions per hour with an average end user response time of only 0.12 seconds. Only 23% Database Server CPU and 9,571 storage I/O operations per second was consumed, leaving spare capacity to consolidate other applications onto the same platform.

- 37,037 accounts were imported per minute using Siebel EIM without additional tuning.

Note that better performance is possible with further tuning. The goal of this paper is to document overall best practices, not achieve the maximum possible performance.

When moving to a new platform it is always important to test thoroughly before making the move. Typically testing is done by ad-hoc means. Oracle 11g Real Application Testing provides a much more complete and systematic means of testing an existing production workload on a new platform. In this paper we describe best practices for using Real Application Testing during a migration to a Database Machine. To validate our approach a test workload was captured using Oracle Real Application Testing and successfully replayed with minimal divergence.

As an example of how Siebel data behaves with Exadata Hybrid Columnar Compression, we report the results from one financial customer that achieved a 10:1 compression ratio for a Siebel data table using Exadata Hybrid Columnar Compression.

Introduction

In this paper we offer our recommendations for how to move a Siebel database to Database Machine with best practices for achieving excellent performance and maximum availability.

The paper is divided into the following chapters:

- Project Planning – key preparation and testing activities that will result in a successful migration, with pointers to the later sections and external documentation.
- Testing Siebel with Oracle Real Application Testing – detailed best practices for using Real Application Testing with Siebel.
- Best Practices for Running Siebel on Database Machine – our recommendations for running Siebel with excellent performance and maximum availability.
- Validation of Best Practices in MAA Lab - the internal Oracle testing that was used to develop and validate our recommendations and best practices. The testing was performed with Siebel 8.0.0.8 on Oracle Database 11g Release 2.

Project Planning

A migration to Database Machine will involve careful preparation and testing so that the eventual go-live will be successful. In this chapter we discuss the preparation, testing and migration phases in more detail.

Preparation

The preparation for moving to Database Machine should start as early as possible so as to lay out the detailed project steps and identify any potential obstacles. In the following sections we highlight some areas to consider:

Product Versions and Patch Levels

Identify and validate the product versions and patch levels that will be required on the Database Machine, the current database server, and Siebel.

Siebel Database and Siebel Application Versions

The Database Machine only supports Oracle Database 11g Release 2 databases and so if the current Siebel implementation is running on an earlier version then a database upgrade will be required. It is recommended that the database upgrade be performed before the migration to Exadata.

It may also be necessary to patch or upgrade Siebel to a version that is supported on Database 11g Release 2. It will be best to apply these patches and upgrades as a separate activity before the migration to the Database Machine.

Patch Level for Database Machine

The latest Oracle database version and bundle patch level will be the starting point for defining a patch level for the Database Machine – see Support Note 888828.1 - Database Machine and Exadata Storage Server 11g Release 2 (11.2) Supported Versions for details. In addition, the following patches may be required:

- Patches Recommended for Real Application Testing, see section Database Patch Levels for Real Application Testing for details.
- In some cases, patches for issues identified in the current production environment may not yet be fixed in the latest Database Machine patch level.

Real Application Testing Patches for the Current Production Database

It may be necessary to apply additional patches to the current production system so that a representative production workload can be captured for testing by SQL Performance Analyzer

(SPA) and Database Replay. See section Database Patch Levels for Real Application Testing for details.

Characteristics and Key Performance Metrics of the Current Production Workload

The characteristics and key performance metrics of the current production workload should be clearly understood before embarking on the migration project so that meaningful functional and performance goals can be defined.

It is common for workloads to be divided into different types, for example online, batch, reporting, and they may run at different times during the day. There may be busy times of the day or week that are important to understand, and the workload may be known to be growing over time.

Measurements are best taken at the application level (elapsed times, response times, business throughput) so that they translate easily to the new database platform. Elapsed times for batch work will normally be available from reports or trace files. Response times may be gleaned from web server reports. Business throughput will typically be measured through analytic reports or charts.

Develop a Database Migration Process

Develop a migration process that can be used to copy the database to the Database Machine for testing and also for the final migration before go-live. The migration process will differ depending on the current database platform and system availability requirements. Developing the process should be given top priority so that testing is not delayed.

The migrations performed during testing will provide an opportunity to test, practice and refine the migration process and so keep precise notes so that the process can be repeated exactly during go live. Where possible, steps should be scripted to increase accuracy and reduce errors. It is also useful to measure the time taken during each step so that an accurate estimate of the go-live migration time can be calculated.

Standard database migration techniques can be used to move the Siebel database from the current platform to the Database Machine. The process used will depend on the current database platform and the availability requirements of the system. Please see Best Practices for Migrating to Sun Oracle Database Machine and Exadata Cell for more details. Note, in some cases it may be necessary to gather optimizer statistics after the migration.

If migrating from a non-Oracle database (SQL Server or DB2) then the supported migration method will leverage the Siebel dataexp and dataimp tools.

Initial Test Configuration

The Database Machine must be configured before testing can begin and this may involve changes to the operating system and database configuration. It will be necessary to consider information from several sources to arrive at an appropriate starting point for testing, including:

- Our best practices for running Siebel on Database Machine – see section Best Practices for Running Siebel on Database Machine for details.
- The current production system and database configuration. Ideally this is well documented and will explain why each setting was necessary.
- Generic Siebel tuning best practices.

In some cases there may be conflicts between these different sources and it may be difficult to decide which setting is best, so get started early so there is time to test and resolve any important issues.

As tests are performed and analyzed the configuration will change and will eventually become the final production configuration and so document and manage this process carefully.

Develop a Testing Plan

Define and prioritize the testing that must occur with appropriate success criteria. This will normally include a combination of functional, performance, and load tests.

Oracle SQL Performance Analyzer (SPA) is a single user testing tool for measuring and tuning SQL statement performance. SPA captures the SQL statements used in a production database and runs the exact SQL statements on a test environment. It then compares plans and execution times to identify potential issues. We highly recommend that SPA be used to test and tune SQL performance, and we recommend that this be performed as a first step so that subsequent tests benefit from well performing SQL.

Oracle Database Replay can be used to replay the production workload for load and concurrency testing. We highly recommend that Database Replay be used to run load and concurrency tests on the Database Machine. It is the easiest and most accurate way to reproduce the full concurrent production workload in the test environment.

Testing

The following activities will be performed in the testing phase of the project:

Configure a Remote SPA Testing Environment

It is recommended that SPA be run in a “remote” configuration where it will connect to the production database to capture the SQL Tuning Set (STS) and connect to the test database to perform a SQL trial. This will provide a stable centralized access point to the SPA test functionality and trial results during the project. It is best to keep this database at the latest database version and patch level so as to have the very latest SPA functionality.

Patch Current Production to Enable Database Replay Capture

Make sure any necessary SPA patches are applied on the current production database before Database Replay capture.

Prepare and Configure the Database Machine for Testing

In this paper we will assume that the new Database Machine will also be the test environment for this migration project although this may not always be the case. The test machine should be configured as per the initial test configuration – see Initial Test Configuration.

Perform SPA Testing and Tuning

The general steps for performing the SPA analysis are:

- Use Incremental cursor cache capture method to capture a representative SQL tuning set or sets on the current production system for interesting (peak or critical) workload time periods. The goal here is to capture the important SQL statements to get a clear picture of how they will perform on the new system.
- Copy the current production database to Database Machine for SPA testing using your migration process – see Develop a Database Migration Process. It is important to have the same data volumes and optimizer statistics as production in order to perform a realistic test.
- Test the production SQL on the Database Machine using SPA.
- Analyze results and tune iteratively. SPA offers the user the ability to remedy any bad plans by creating SQL plan baselines or SQL profiles (through the SQL Tuning Advisor).

Please also refer to SQL Performance Analyzer Best Practices for more details.

Perform Database Replay Testing and Tuning

Once the SPA testing is advanced and the SQL performance is well understood it is time to perform database replay testing. The general steps for performing the database replay are:

- Capture a representative workload on the production system.
- Copy the current production database to the Database Machine for replay testing. In general, it will be necessary to restore the database to a specific point-in-time in order to perform a clean replay.
- Replay the workload.
- Analyze the results and tune iteratively.

Please also refer to the section entitled “Database Replay Best Practices” for more details and best practices.

Perform In-house Functional, Performance, Load and Acceptance Tests

There will likely be additional tests to run before the system is ready to go live. It would, for example, be advisable to perform a full stack test to confirm that all the components in the system are configured correctly and work together. As we have already emphasized, it is best to wait until SPA testing and tuning is well advanced before beginning additional testing.

Define the Final Production Configuration and Prepare Database Machine for Production

After all tests have been completed a final production configuration will have been defined and documented and this should now be applied to ready the Database Machine for live operation.

Perform a Go-live Test

If you have a small window for the final migration to production and little scope for error then it is a good idea to test and practice the go-live procedures so that everything will go smoothly on the day.

Migration

Once the testing and tuning has been completed you will be ready to go live. This will require the following steps:

- Shutdown current production. You should have a good estimate of how long the migration will take at this point and so you can prepare for the downtime accordingly.
- Copy the database to the Database Machine. Note, this is a full migration of the latest data, rather than the migration to point-in-time (SCN) that you may have performed during testing.
- Redirect Siebel connections to Database Machine and Start Siebel. This would be the time to implement the middle tier configuration changes if you haven’t already. See section Siebel Tier Configuration for details.
- Perform acceptance testing.
- Go live on the Database Machine!

Testing Siebel with Oracle Real Application Testing

We recommend using the features of Oracle Real Application Testing – SQL Performance Analyzer (SPA) and Database Replay – to test and tune your workload on the database machine prior to go-live. In following sections we discuss the database patch requirements and best practices for running Real Application Testing with Siebel.

Database Patch Levels for Real Application Testing

In some cases it may be necessary to apply patches to the current production system or the Database Machine in order to successfully run Real Application Testing. Please refer to Support Note 560977.1 - Real Application Testing Now Available for Earlier Releases for details of which patches are required or recommended for running Real Application Testing. Note, this may apply to the current production system for capture or for testing on the Database Machine.

SQL Performance Analyzer Best Practices

The SQL Performance Analyzer (SPA) has the ability to collect the existing SQL statements running in the current production and assess their performance on the Database Machine. Detailed analysis reports are available that can highlight poorly performing statements or statements that have different execution plans. SPA can capture the SQL workload simply and with little overhead and can be used to quickly pinpoint problem areas for further analysis and tuning. We recommend that SPA analysis be performed before other tuning so as to provide a well tuned foundation for subsequent testing.

Additional recommendations are described below:

Start Small to Validate the Test Environment

It is best to start with an end-to-end test with just a few SQL statements to confirm that the capture and test environment are functioning correctly before scaling up to larger runs. This will allow you to find any issues quickly and rapidly retest to validate resolution actions.

Allow for Siebel Session-level Database Parameters

The Siebel Object Manager sets session-level database parameters on login to the database and, by default, SPA does not apply them during tests, and so as a result, query plans are likely to be different. This is a useful feature because sometimes you want to change the settings. There are a number of workarounds if you want the session-level parameters to be applied:

- Direct SPA to use the captured environment settings:

```
exec dbms_sqlpa.set_analysis_task_parameter('TASK_123',
```

'APPLY_CAPTURED_COMPILENV', 1);

Note, APPLY_CAPTURED_COMPILENV will cause capture environment settings to be set during a trial and this will override system or session level database parameter settings in the test environment.

- Use a logon trigger that sets the Siebel session variables, for example:

```
CREATE OR REPLACE TRIGGER om_trig
  AFTER LOGON ON DATABASE
  BEGIN
  execute immediate 'alter session set optimizer_mode=FIRST_ROWS_10';
  execute immediate 'alter session set "_hash_join_enabled"=FALSE';
  execute immediate 'alter session set
  "_optimizer_sortmerge_join_enabled"=FALSE';
  execute immediate 'alter session set
  "_optimizer_join_sel_sanity_check"=TRUE';
  END;
  /
```

- Set parameters at the system level in the test environment so that all tests will pick up the settings.

Allow for Bind Peeking

Depending on the version, Siebel may or may not perform bind peeking when executing SQL statements and in some cases this will mean that SPA tests may not perform the same way as captured. This can be remedied with a logon trigger with the "_optim_peek_user_binds" parameter set to correspond with the Siebel behavior. See section Allow for Siebel Session-level Database Parameters for an example of a login trigger.

Database Replay Best Practices

In our Siebel tests it was possible to achieve zero replay divergence by capturing the entire Siebel workload and replaying with SCN ordering.

Note, the Database 11.2.0.2 patchset supports Database Replay + SPA integration enabling STS capture and Database Replay in a single iteration.

We identified the following best practices:

Start Small

It is best to start with an end-to-end test using a small workload, say 5 minutes in duration or with just a few concurrent users, to confirm that the capture and test process is functioning correctly, before scaling up to larger loads. This will allow you to find issues quickly and retest to validate resolution actions.

Restore Database to Capture Point

Due to the nature of Siebel optimistic locking, the Siebel database must be restored to the exact point where the capture was started before replay. See section Background on Siebel Optimistic Locking and Row ID Generation.

Replay in Synchronized Mode

Due to the nature of Siebel optimistic locking, calls must be replayed in the exact SCN for reliable results. This can be done by running the replay in synchronized mode. See section Background on Siebel Optimistic Locking and Row ID Generation .

Capture Entire Siebel Workload

Due to the nature of Siebel optimistic locking and row ID generation, the entire Siebel workload running at any point in time must be captured and replayed. See section Background on Siebel Optimistic Locking and Row ID Generation.

Start Capture before Starting Siebel

To establish a clean starting point for capture, it is best to start capture before starting Siebel. This will allow the session environment setup (alter session statements) for all Siebel database sessions to be captured and replayed completely.

If it is impossible to find a convenient time to restart Siebel to begin capture then some Siebel database connections will already be in-flight when the capture starts and the session environment setup for those connections will not be captured. In this case, it is possible to implement a logon trigger in the replay environment to establish the appropriate alter session statements for all sessions, for example:

```
CREATE OR REPLACE TRIGGER set_siebel_env
AFTER LOGON ON DATABASE
BEGIN
  execute immediate 'alter session set optimizer_mode=FIRST_ROWS_10';
  execute immediate 'alter session set "_hash_join_enabled"=FALSE';
  execute immediate 'alter session set
"_optimizer_sortmerge_join_enabled"=FALSE';
  execute immediate 'alter session set
"_optimizer_join_sel_sanity_check"=TRUE';
  execute immediate 'alter session set
"_optim_peek_user_binds"=FALSE';
END;
/
```

Guard Against Connection Timeout

Client connections are strictly ordered during replay and in some cases a client may timeout waiting to login with error "ORA-01270: TNS:Connect Timeout". To prevent this from happening, lengthy connection timeouts should be configured in the SQLNET.ORA file used by the workload replay clients. For example (time in seconds):

```
TCP.CONNECT_TIMEOUT=500000
SQLNET.OUTBOUND_CONNECT_TIMEOUT=500000
```

Configure Process Threads

During a highly concurrent workload replay, as is common with Siebel, each replay client may run many process threads and if the thread limit is reached the error "ORA-15563: workload replay client cannot spawn new threads" will be reported. To prevent this from happening, increase the thread limit for the user running the replay by increasing the nproc limit in the /etc/security/limits.conf file, for example:

```
oracle     soft    nproc       100000
oracle     hard    nproc       100000
```

Accelerating Replay Workload

It is possible to accelerate the database replay in order to simulate a workload that is larger than was captured. This is done by adjusting think_time and connect_time parameters, for example:

```
think_time=>50, connect_time=>50
```

Background on Siebel Optimistic Locking and Row ID Generation

Siebel optimistic row locking uses the MODIFICATION_NUM column defined on every Siebel table. It is read with every query, and it is first checked and then incremented whenever a row is updated. If the value has changed between query and update then the row will not be updated. Here is an example of an update statement:

```
UPDATE SIEBEL.S_SSA_ID SET
  DB_LAST_UPD = CURRENT_DATE,
  LAST_UPD = :B4,
  LAST_UPD_BY = :B3,
  MODIFICATION_NUM = :B2,
  NEXT_SUFFIX = :B1
WHERE
  ROW_ID = :B6 AND
  MODIFICATION_NUM = :B5

Bind Variables:
:B1: "2W7QCH"
:B2: 538033
:B3: "1-A81OX"
:B4: "03/02/2010 05:11:00"
:B5: 538032
:B6: "0-11"
```

Note how the MODIFICATION_NUM is tested in the WHERE clause and incremented in the SET clause of the same statement. The statement only succeeds if the value has not changed since last queried.

When a row is not updated during replay that was updated during capture it results in a replay divergence - "DMLs with Different Number of Rows Modified" or "New Errors Seen During Replay".

When a workload is replayed, the values bound to each bind variable are as captured - there is no correlation between the values read during replay and the subsequent update operations. As a result:

- If the MODIFICATION_NUM value in a row is not exactly as it was during capture the row will not be updated
- If an update is replayed out of order then the row will not be updated
- If an update is omitted or fails then the row will not be updated for all subsequent updates

In addition, note that the S_SSA_ID table is used by all Siebel processes to generate Siebel row ID's and there is only one row in this table. S_SAA_ID also leverages Siebel optimistic locking and so updates to this table across all processes must be captured and replayed in order to avoid divergence.

Best Practices for Running Siebel on Database Machine

It is necessary to configure Siebel and the Database Machine correctly for excellent performance and availability by following our Maximum Availability Architecture recommendations. Below are our best practice recommendations that we identified and validated through our testing. These recommendations are in addition to the Siebel and Oracle Database documented installation and configuration steps. Although not specifically tested, we would recommend the same best practices for Siebel 8.1.

Database Tier Configuration

Here are the things to consider when configuring the Database Servers:

Patch Level

Please refer to Support Note 888828.1 - Database Machine and Exadata Storage Server 11g Release 2 (11.2) Supported Versions for details of the latest Database Machine supported versions and bundle patches.

Configure Hugepages

Siebel will typically run with many database connections and a large SGA and so configuring hugepages for the Siebel database instances is required. See Support Note 744769.1 - How to Configure HugePages for Oracle Database on 64-bit Linux Platforms for details on how this is done.

Create a Database Service and Configure for Transparent Application Failover (TAF)

A database service provides a simple named access point to the database and more convenient TAF configuration. A service can physically span multiple instances in an Oracle RAC cluster and can be simply moved from one instance to another. By requiring Siebel to connect only through a service we are able to relocate or reconfigure the service without reconfiguring Siebel.

A database service can be created and configured through Enterprise Manager or using the srvctl command line tool. The "SELECT" failover type and "BASIC" failover method are supported by Siebel and should be adequate for most configurations. For example, this is how a service can be created with srvctl:

```
srvctl add service -d SIEBEL -s SIEBEL -r "SIEBEL1,SIEBEL2" -P
basic -e SELECT -j LONG
```

The parameters are defined as follows:

- -d SIEBEL: The database unique name

- -s SIEBEL: The database service name
- - r "SIEBEL1,SIEBEL2": Preferred database instances that will start the service
- -P basic: The TAF failover method
- -e SELECT: The TAF failover type.
- -j LONG: Connection load balancing method (Siebel connections normally last a long time)

Note, the –z (failover retries) and -w (failover delay) parameters have no effect on Siebel behavior because Siebel will always retry 24 times with a delay of 5 seconds.

See the Oracle Real Application Clusters Administration and Deployment Guide for details.

Handle Database Password Expiration

The default behavior of Oracle Database has changed in 11g such that database user passwords will expire after 180 days. Processes should be put in place to refresh passwords regularly or expiration should be extended or disabled. Siebel application availability will be impacted if passwords are allowed to expire. Password expiration for the default user profile can be disabled with the following command:

```
alter profile default limit password_life_time unlimited;
```

Siebel Tier Configuration

Here are the things to consider when configuring the Siebel Servers:

Configure Siebel to Connect to the Database Service

In order for Siebel to take advantage of TAF it must be configured to connect to the database service recommended in section Create a Database Service and Configure for Transparent Application Failover (TAF). The client must also be configured to try all the listeners in the RAC cluster on failover so that a new connection can be established on a surviving node, and not get hung-up on any specific listener. There changes are made in the TNSNAMES.ORA file on each Siebel Server. The way this is configured is different depending on the version of database client software installed on the Siebel Servers. There are two cases described in the following section:

Siebel Server Configuration with Oracle Database Client 11g Release 2

To configure Siebel to connect to the database service the SERVICE_NAME parameter is specified, and this is the same for any version of Oracle Database Client.

So that the Siebel servers can access any node in the Database Machine RAC cluster, the SCAN address is specified. The connection timeout is specified with the CONNECT_TIMEOUT parameter, and the number of retries is governed by the RETRY_COUNT parameter. The SCAN address, and the CONNECT_TIMEOUT and RETRY_COUNT parameters are new in Oracle Database 11g Release 2, see SINGLE CLIENT ACCESS NAME (SCAN) for details. CONNECT_TIMEOUT overrides the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter setting in the SQLNET.ORA file.

Here is an example TNSNAMES.ORA configuration showing the CONNECT_TIMEOUT (3 seconds), RETRY_COUNT (3 retries), scan address, and SERVICE_NAME:

```
SIEBEL = (DESCRIPTION =
   (CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
   (ADDRESS=(PROTOCOL=TCP)(HOST=test-scan)(PORT=1521))
   (CONNECT_DATA= (SERVICE_NAME=SIEBEL))
)
```

The above configuration changes should be made for all Siebel servers.

Siebel Server Configuration with Oracle Database Client 11g Release 1 and Below

To configure Siebel to connect to the database service the SERVICE_NAME parameter is specified, and this is the same for any version of Oracle Database Client.

So that the Siebel servers can access any node in the Database Machine RAC cluster, each SCAN VIP should be listed in the ADDRESS_LIST. See SINGLE CLIENT ACCESS NAME (SCAN) for further details of the SCAN VIP and configuring an earlier database client version with an 11gR2 RAC database.

Here is an example TNSNAMES.ORA configuration with a Quarter Rack Database Machine:

```
SEBL = (DESCRIPTION =
   (ADDRESS_LIST = (LOAD_BALANCE=ON)(FAILOVER=ON)
      (ADDRESS=(PROTOCOL=TCP)(HOST=test_scan_vip1)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=test_scan_vip2)(PORT=1521)))
   (CONNECT_DATA= (SERVER=DEDICATED) (SERVICE_NAME=SIEBEL))
)
```

When connecting or reconnecting to the database, Siebel will keep trying all the addresses in the address list until a connection is established. It is important that Siebel does not get “hung-up” on any one address, as this will delay connection. The SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter dictates the maximum amount of time (in seconds) that we wait for a response from an address before skipping to the next address in the list. A setting of 3 seconds is recommended and is usually acceptable in most cases. This parameter should not be set to less than 3 seconds. Here is an example SQLNET.ORA:

```
SQLNET.OUTBOUND_CONNECT_TIMEOUT=3
```

The above configuration changes should be made for all Siebel servers.

Configuring TCP Keepalive Timeout

Siebel currently does not support Oracle Fast Application Notification (FAN) and so it is necessary to reduce the TCP Keepalive Timeout for the Siebel Servers to release database connections in the event of a database node crash or network outage. This is only for the rare case where the database node crashes or network fails before the TCP connections can be cleaned up, and only for connections where a database request was in-flight at the time of failure or a new request was started before the Virtual Internet Protocol (VIP) Address could be switched to a surviving node. In all other cases the database connection failure is detected immediately and a new connection is established on a surviving node.

Please refer to Support Note 249213.1 - Performance problems with Failover when TCP Network goes down (no IP address) for details on how to configure the TCP Keepalive timeout. Making these configuration changes may have adverse effect on network utilization and so all changes should be tested and monitored carefully.

Backup and Restore

It is possible to achieve very high database backup and recovery rates on a Full Rack Database Machine with High Performance disk drives. In tests, the following results were achieved:

- Up to 7.1 TB/hour for disk-based full image backups
- Incremental disk-based backups - effective rate of 10 to 48 TB/hour
- Database restore - 23 TB/hour
- Redo apply - 2.1 TB/hour (637 MB/sec)

These rates were achieved with a database CPU utilization of less than 10% on the targeted database servers, leaving plenty of bandwidth for concurrent user workloads. Please see Backup and Recovery Performance and Best Practices for Exadata Cell and the Sun Oracle Database Machine for details.

Validation of Best Practices in MAA Lab

Test Workload

The test workload was designed to simulate a large organization, consisting of thousands of concurrent, active users in a call center organization. The following users are simulated:

- Service representatives running Siebel Financial Services Call Center
- Partner service representatives running Siebel Partner Relationship Management
- External application users calling web services

Seed Data

Prior to test execution, the database was seeded with over 100 GB of data, including Accounts, Contacts, Activities, Opportunities and Service Requests .

Business Transactions

Each simulated user executed a complex business transaction comprised of many different business operation, such as "Create a new Contact", and "Add Products to an Opportunity". The response time per operation and the business transaction completion rate were measured.

Lab Configuration

After some initial testing we estimated that given our Siebel mid-tier capacity we should target a quarter rack DB machine and a 30,000 user workload for our testing.

Our configuration was comprised of:

- 9 × Load Generators (Windows Server 2003 R2)
- F5 BIG-IP 6400 Appliance balancing Siebel Web Server load
- 2 × Siebel Web Servers (Oracle HTTP Server, Siebel 8.0.0.8, OEL 4 Update 8, Oracle VM 2.1.2) – 8 × Intel Xeon X5355 – 32 GB RAM
- 8 × Siebel Application Servers (Siebel 8.0.0.8, OEL 4 Update 8, Oracle VM 2.1.2) – 8 × Intel Xeon E5430 – 32 GB RAM
- Sun Oracle Database Machine Quarter Rack (see Sun Oracle Database Machine for detailed specifications)

Database Compression Testing

A financial services customer running Siebel on Exadata produced the following results when compressing the S_EVT_ACT table:

TABLE S_EVT_ACT

COMPRESSION MODE	RATIO
Query (HCC)	9:1
Archive (HCC)	10:1

Database Replay Testing

The Database Replay testing was performed to validate the use of Database Replay for testing and tuning the performance of the Siebel Workload when moving an existing Siebel database to Database Machine. The test details and results are documented in the following section:

Siebel 8.0.0.8 30,000 User Database Replay Test

The full load test workload was captured and then replayed on the Database Machine. After several iterations a successful replay was achieved with the results as follows:

SIEBEL 8.0.0.8 30,000 USER DATABASE REPLAY TEST RESULTS	
User Calls	70,959,030
Total Divergent User Calls	133
Divergence Rate	0.00019%

Performance Testing

The performance testing was performed to better understand how Siebel can leverage Database Machine features so as to develop and validate configuration best practices. The test details and results are documented in the following section:

Siebel 8.0.0.8 EIM Account Import Test

The EIM interface table was primed with account records before the test using the following PL/SQL script:

```
DECLARE
   name CONSTANT VARCHAR2(100) := 'REXLEY'; -- account name prefix
   start_id CONSTANT INTEGER := 1; -- Start ID
   end_id CONSTANT INTEGER := 50000; -- End ID
   start_batch CONSTANT INTEGER := 1; -- Start batch
   end_batch CONSTANT INTEGER := 100; -- End batch
BEGIN
--   delete from siebel.eim_account where if_row_batch_num=batch_num;
--     delete from siebel.eim_account where if_row_stat = 'DELETED';
--     update siebel.eim_account set if_row_stat = 'For Import' where
if_row_stat = 'Imported' or if_row_stat = 'IMPORTED';
--   commit;
   FOR B IN start_batch..end_batch LOOP
      FOR I in start_id..end_id LOOP
         insert into oraperf.eim_account (
            row_id, if_row_batch_num, if_row_stat, name,
party_type_cd, party_uid, cust_stat_cd ) values (
            I, B, 'FOR_IMPORT', name || '-' || B || '-' || I,
'Organization', name || '-' || B || '-' || I, 'Candidate');
      END LOOP;
   END LOOP;

   commit;

END;
/
```

The EIM job was executed for batch number 1 and the elapsed time was recorded. The results were as follows:

EIM ACCOUNT IMPORT PERFORMANCE

ACCOUNT RECORDS IMPORTED	ELAPSED TIME	ACCOUNTS PER MINUTE
50,000	81s	37,037

Load Testing

The load test was performed to discover and validate the best practices for running Siebel on a Database machine. The load test was performed in archive log mode and with flashback database enabled as recommended in our MAA best practices. The test details and results are documented in the following section:

Siebel 8.0.0.8 30,000 User Load Test

Note that these results are intended for general information purposes and are not a substitute for implementation-specific sizing or benchmarks.

In each case, the database was first restored to an initial state and the database servers were started in a 2 node RAC configuration. The Siebel servers and Web servers were started next, and then the workload was started and ramped up to a steady load of concurrent users. The load was maintained and measurements were taken over a 30 minute period.

Here are the results of the test consisting of approximately 30,000 concurrent users:

RESPONSE TIMES AND BUSINESS TRANSACTION THROUGHPUT

WORKLOAD	NUMBER OF USERS	AVERAGE OPERATION RESPONSE TIME (SECONDS)	BUSINESS TRANSACTIONS PER HOUR
Aggregate	30,276	0.12	443,334

DATABASE SERVER PERFORMANCE

	COMPUTE NODE 1	COMPUTE NODE 2	AGGREGATE
CPU Utilization (%)	23%	24%	23%
SGA Size (MB)	36,864	36,864	73,728
Redo Generation (B / s)	3,740,945	4,005,464	7,746,408
Logical Reads (blocks / s)	215,609	233,991	449,600
Physical Reads (blocks / s)	253	273	526
Physical Writes (blocks / s)	710	1,209	1,920
Transactions / s	459	482	941
User Calls / s	7,380	7,609	14,990
Flash Cache Read Hits / s	123	132	255
Flash Cache Hit Rate (%)	50%	49%	49%
Physical Write Requests / s	1,596	1,713	3,310

STORAGE CELL PERFORMANCE

	CELL 1	CELL 2	CELL 3	AGGREGATE
Read Operations / s	216	193	195	604
Write Operations / s	2,725	3,072	3,171	8,968
Total IO Operations / s	2,941	3,264	3,366	9,571
Read MB / s	5	6	6	17
Write MB / s	42	48	48	138
Total MB / s	47	54	54	155

Conclusion

This paper describes the best practices for moving an existing Siebel implementation to a Sun Oracle Database Machine and achieving excellent performance and maximum availability.

To develop and validate our best practices, Siebel 8.0.0.8 Financial Services application was deployed on a Quarter Rack Sun Oracle Database Machine in a Maximum Availability Architecture (MAA) configuration. MAA is Oracle's best practices blueprint for implementing Oracle high-availability technologies, including RAC, Data Guard, and Flashback.

The database behavior and performance was evaluated under several test workloads with the following results:

- 30,000 concurrent users running a combination of call center, partner service, and web service users were able to process 443,334 business transactions per hour with an average end user response time of only 0.12 seconds. Only 23% Database Server CPU and 9,571 storage I/O operations per second was consumed, leaving spare capacity to consolidate other applications onto the same platform.
- 37,037 accounts were imported per minute using Siebel EIM without additional tuning.

Note that better performance is possible with further tuning. The goal of this paper is to document overall best practices, not achieve the maximum possible performance.

When moving to a new platform it is always important to test thoroughly before making the move. Typically testing is done by ad-hoc means. Oracle 11g Real Application Testing provides a much more complete and systematic means of testing an existing production workload on a new platform. In this paper we describe best practices for using Real Application Testing during a migration to a Database Machine. To validate our approach a test workload was captured using Oracle Real Application Testing and successfully replayed with minimal divergence.

As an example of how Siebel data behaves with Exadata Hybrid Columnar Compression, we report the results from one financial customer that achieved a 10:1 compression ratio for a Siebel data table using Exadata Hybrid Columnar Compression.

References

- Sun Oracle Database machine and Exadata Storage Server
  http://www.oracle.com/technology/products/bi/db/exadata/index.html
- Oracle Maximum Availability Architecture web site
  http://www.oracle.com/goto/maa
- Real Application Testing User's Guide (Part #E12254)
  http://www.oracle.com/pls/db112/to_toc?pathname=server.112/e16540/toc.htm
- Best Practices for Migrating to Oracle Exadata Storage Server
  http://www.oracle.com/technetwork/database/features/availability/xmigration-11-133466.pdf

Siebel on Exadata
October 2010
Author: Richard Exley

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

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