ORACLE
DATABASE APPLIANCE

An Oracle White Paper
January 2013

# Evaluating and Comparing Oracle Database Appliance Performance

ORACLE

## Executive Summary

Oracle Database Appliance is a highly available Oracle database system. It is an integrated, pre-built, pre-tuned, packaged database solution that contains hardware, software, networking, and storage, all in a single 4-U rack mountable box. The hardware and software configuration of Oracle Database Appliance provides redundancy and protects against all single points of failures within in the system.

**Figure 1: Oracle Database Appliance Front Panel**

Specifically, Oracle Database Appliance is a two node, Intel X-86 cluster with direct attached SAS and SSD storage. It runs standard, time-tested software components, such as Oracle Linux operating system (OS), Oracle Relational Database Management System (RDBMS), Oracle Real Application Clusters software (RAC), Oracle Clusterware and Oracle Automatic Storage Management (ASM). The pre-built, pre-tested, and pre-tuned configuration ensures that Oracle Database Appliance can be deployed rapidly and does not require typical efforts to tune the configuration. Further, Oracle Database Appliance includes Oracle Appliance Manager, to manage and maintain the system, including patching, upgrades, and troubleshooting.

The purpose of this white paper is to demonstrate and document the performance of OLTP and DSS like workloads executing on Oracle Database Appliance as well as to provide a procedure for system architects and database administrators to compare performance of a standardized *Swingbench* (a freely available performance testing tool) workloads executing on Oracle Database Appliance vis-à-vis a legacy environment.

During the performance testing conducted as part of documenting these results, Oracle Database Appliance demonstrated scalable performance for both OLTP and DSS type of database workloads. With all 24 cores active, Oracle Database Appliance supported 10,000 concurrent *Swingbench* users, and more than 9800 *Swingbench* transactions per second while maintaining an average response time of less than 30ms. Also, with the DSS type workload Oracle Database Appliance easily supported a sustained IO throughput of more than 2400 MB/Second.

## Audience

This white paper will be useful for IT Department heads, Database Management Directors and Managers, Database Architects, CTOs, CIOs, and Purchase Managers who may be interested in understanding or evaluating the performance capabilities of Oracle Database Appliance. Oracle Database Administrators, System Administrators, and Storage Administrators may find the information useful in conducting performance tests in their own environments. They will learn the best practices that can further improve the performance of specific workloads running on Oracle Database Appliance.

## Objectives

A quick review of the hardware configuration of Oracle Database Appliance will show that the architecture of the system is built for high availability and out-of-the-box performance. However, due to the many components in any given system, customer often request baseline comparative performance data for various types of standard workloads. This helps project their own performance experience and expectations once they migrate their databases to a new environment.

The objective of this white paper is to quantify performance of Oracle Database Appliance when running OLTP and DSS like workloads. This performance information is presented to show number of users, number of transactions, and transaction performance.

In addition, this white paper will illustrate a process for executing test workload in non-Oracle Database Appliance (legacy) environments and comparing the results against data captured in this white paper. This is facilitated by documenting the test results for two key *Swingbench* workloads, namely Order Entry (OLTP) and Sales History (DSS) running on Oracle Database Appliance.

This paper is the result of extensive testing carried out with the above standard workloads while varying the volume in different configurations of the Oracle Database Appliance. You can run these same *Swingbench* workloads on your current systems and compare with results obtained and outlined for the Oracle Database Appliance in this white paper.

The following flow diagram illustrates the intended use of the information presented in this white paper when comparing performance of a given workload running in a legacy environment and Oracle Database Appliance environment.

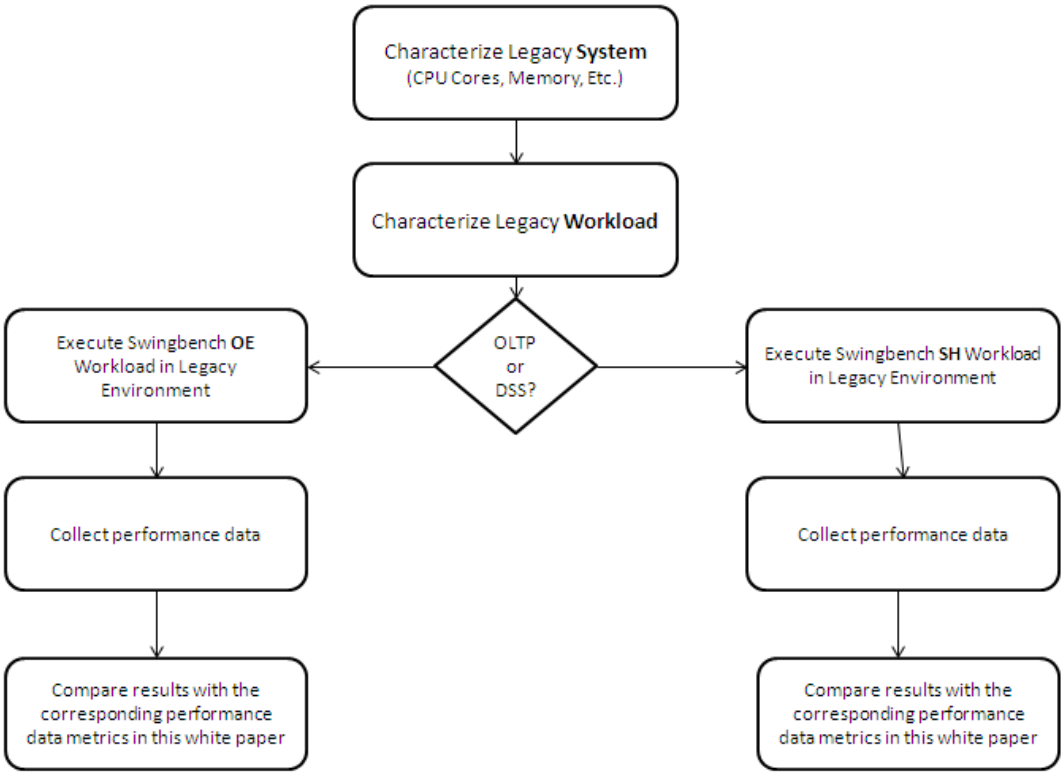



**Figure 2: How to use the information in this white paper?**

## Oracle Database Appliance performance architecture

Oracle Database Appliance consists of two X-86 servers, each with two 6-core Intel X5675 3.07 GHz CPUs and 96GB of memory. The two nodes use direct attached storage that is comprised of twenty dual-ported, 600 GB, 15,000 rpm SAS hard disk drives and four dual-ported, 73 GB SAS solid state disks (SSDs). The dual-ported disks are connected to the two server nodes via disk expanders and redundant HBAs.

Within Oracle Database Appliance, storage redundancy is provided by Oracle Automatic Storage Management (ASM). Data is either double or triple mirrored and distributed to sustain and recover from disk failures. Further, ASM distributes primary and secondary ASM extents across all disks in a manner that prevents any hot spots and eliminates single points of failures.

**Figure 3: Oracle Database Appliance Back Panel**

Server node redundancy is provided by Oracle Real Application Clusters (RAC). Two internal, dedicated, load balanced and redundant, 1-GbE channels provides robust and reliable high speed cluster connectivity within Oracle Database Appliance. In case of a single server node failure, Oracle Database Appliance continues to operate and provide service to the application.

Oracle Database Appliance also provides multiple, 1GbE and 10-GbE network interfaces. The network channels can be used for different purposes. For example, customers may use separate dedicated networks for their applications, disaster recovery, backup and recovery, and management networks.

## Oracle Database Appliance Database configuration templates

Oracle Database Appliance provides and uses five standard database configuration templates for creating databases of different class and size. Databases created using these templates automatically inherit Oracle database implementation best practices such as most optimal parameter settings, configuration and placement of database components such as REDO log files, control files, etc.

| Template | CPU Count | SGA (Gb) | PGA (Gb) | Log Buffer Size (Mb) | Processes | Redo Log Size (Gb) |
|---|---|---|---|---|---|---|
| Very small | 2 | 4 | 2 | 16 | 200 | 1 |
| Small | 4 | 8 | 4 | 16 | 400 | 1 |
| Medium | 8 | 16 | 8 | 32 | 800 | 2 |
| Large | 12 | 24 | 12 | 64 | 1200 | 4 |
| Very large | 24 | 48 | 24 | 64 | 2400 | 4 |

**Table 1: Oracle Database Appliance Sizing Templates**

In order to perform an ideal performance comparison, the database configuration on Oracle Database Appliance and on the non-Oracle Database Appliance environment should match closely. The above matrix can be used to establish the database

configuration in the non-Oracle Database Appliance environment before executing the test workload.

Oracle Database Appliance standard performance ratings are as follows.

| | Very Small | Small | Medium | Large | Very Large |
|---|---|---|---|---|---|
| IO Per Second | 300 | 600 | 1300 | 2000 | 4000 |
| MB Per Second | 250 | 500 | 1000 | 1500 | 3000 |
| Log Generation Rate (MB/Sec) | 6.83 | 6.83 | 13.65 | 27.30 | 27.30 |

**Table 2: Oracle Database Appliance database sizing model based on performance characteristics**

## What is Swingbench?

Swingbench is a simple to use, free, Java based tool to generate database workload and perform stress testing using different benchmarks in Oracle database environments. The tool can be downloaded from http://dominicgiles.com/downloads.html

Swingbench version 2.4.0.845 was used to perform the testing documented in this white paper. For more information about Swingbench, please refer to Swingbench documentation available at http://www.dominicgiles.com/swingbench.html

Swingbench provides four separate benchmarks, namely, Order Entry, Sales History, Calling Circle, and Stress Test. For the tests described in this paper, Swingbench Order Entry benchmark was used for OLTP workload testing and the Sales History benchmark was used for the DSS workload testing.

| Swingbench Benchmark | Workload Type |
|---|---|
| Order Entry | OLTP |
| Sales History | DSS |

**Table 3: Swingbench workload types used for testing**

The Order Entry benchmark is based on the OE schema and is TPC-C like. The workload uses a workload read/write ratio of 60/40 and is designed to run continuously and test the performance of a typical Order Entry workload against a small set of tables, producing contention for database resources.

The Sales History benchmark is based on the SH schema and is TPC-H like. The workload is query (read) centric and is designed to test the performance of queries against large tables.

## Swingbench download and setup

Swingbench can be setup in a Windows or Linux environment. The Linux environment was used during the testing below. It is recommended that Swingbench software be installed on a separate machine that has local network connectivity to the Oracle Database Appliance. Running *Swingbench* client on Oracle Database Appliance itself can affect performance measurements and produce undesirable results.

To install Swingbench, perform the following steps.

1. Download Swingbench software

   The Swingbench software can be downloaded from the following site. http://dominicgiles.com/downloads.html

   Download the Swingbench version 2.4.0.845 or later. Please note that 2.4.0.845 was used during this testing.

2. Unzip the downloaded file. Replace the <path>/bin/swingconfig.xml file by the swingconfig.xml supplied in the Appendix B of this Paper. The Appendix B provides Swingbench configuration files used for both Order Entry and Sales History tests covered in this paper.

3. Specify non-blocking random number generators

   Oracle 11g JDBC driver requires a random number for encrypting connect string. By default this random number is generated from /dev/random. In certain situations, the random number generation may be blocked and remain hung for an extended period due to a system entropy based algorithm used for the random number generation process when using /dev/random. Use of /dev/urandom instead of /dev/random addresses this issue. This change can be made by specifying (in the <swing benchhome>/launcher/launcher.xml file)/dev/urandom in the arguments when launching Swingbench.

| **Old Entry** | **Modified Entry** |
|---|---|
| <jvmargset id="base.jvm.args"><br><jvmarg line="-Xmx1024m" /><br><br></jvmargset> | <jvmargset id="base.jvm.args"><br><jvmarg line="-Xmx1024m –<br>**Djava.security.egd=file:///dev/urandom**"/><br></jvmargset> |

# Part I. Order Entry benchmark (OLTP)

The following steps are required to setup the Order Entry (OE) benchmark.

## Database and schema setup

The tests outlined in this document were performed with Oracle Database Appliance OAK 2.3 image running Oracle Database Release 11.2.0.3.3. When comparing the performance of Oracle Database Appliance against a non-Oracle Database Appliance environment, these tests can be run on other software versions in the non-Oracle Database Appliance environments, if necessary. The database configured on Oracle Database Appliance uses standard templates and standard, optimized parameter settings. The corresponding settings for these parameters may not match their settings on Oracle Database Appliance. In order to perform proper comparison, software versions, and configuration parameters for the database configured in the non-Oracle Database Appliance environment, should be similar to those in Oracle Database Appliance environments as much as possible.

Additionally, the default database parameter settings for Oracle Database Appliance, while optimized for most common uses, may need to be adjusted for further maximizing performance for specific performance intensive workloads. The following steps should be taken for the Oracle Database Appliance (and non-Oracle Database Appliance environment, as applicable) before workload testing begins. Note that the changing of the following database initialization parameters and making them take effect requires restart of the database. This can be accomplished by making all these changes together and then starting the database once.

### Database setup

During the course of performance tests, only the clustered database configuration using Oracle Real Application Clusters (RAC) was used. Thus both server nodes within Oracle Database Appliance were actively running an Oracle database instance and servicing database requests. The following changes should be made together and the database should be restarted once for these changes to take effect.

1. Disable database auditing

   Database auditing is enabled by default on Oracle Database Appliance. It is recommended to turn database auditing off for the duration of the test.

   **SQL> alter system set audit_trail=none scope=spfile;**
   **SQL> alter system set audit_sys_operations=false scope=spfile;**

2. Set higher rollback segment count

The Order Entry workload might show contention for undo segment enqueue. This is typically caused by rapid offlining and onlining of undo segments. Details of this behavior are documented in My Oracle Support (MOS) note 1332738.1. In order to keep a large number of Undo Segments online, set the following parameter.

**SQL> alter system set "_rollback_segment_count"=3000 scope=spfile sid='*';**

3. Set _buffer_busy_wait_timeout

   Bug 13344323 has shown that a high number of inserts may cause buffer busy waits. As a workaround the _buffer_busy_wait_timeout parameter should be set to 20ms (2).

   **SQL> alter system set "_buffer_busy_wait_timeout"=2 scope=spfile sid='*';**

4. Create additional REDO log groups

   Oracle Database Appliance is configured for best practices for a general purpose database workload. For performance intensive stress testing purposes, some changes may further improve the performance of the database. For example, by default, only two redo log groups per instance are created at the time of initial database setup on Oracle Database Appliance. For high performance applications, this default number of redo log groups may be inadequate and can cause performance bottlenecks. This may result in errors such as – "Thread 'n' cannot allocate new log", "Checkpoint not complete" or the free buffer wait event, etc. Therefore, for performance intensive workloads, additional redo log groups may need to be created. For the testing conducted during this project, a total of 4 redo log groups of 1 GB each per instance were created on Oracle Database Appliance.

   This can be done using Oracle Enterprise Manager or using SQL commands.

5. Pre-create SOE tablespace for storing Order Entry schema objects.

   **SQL> create bigfile tablespace soe datafile '+DATA' size 30G autoextend on maxsize unlimited uniform size 1M segment space management auto;**

6. Increase PROCESSES parameter value.

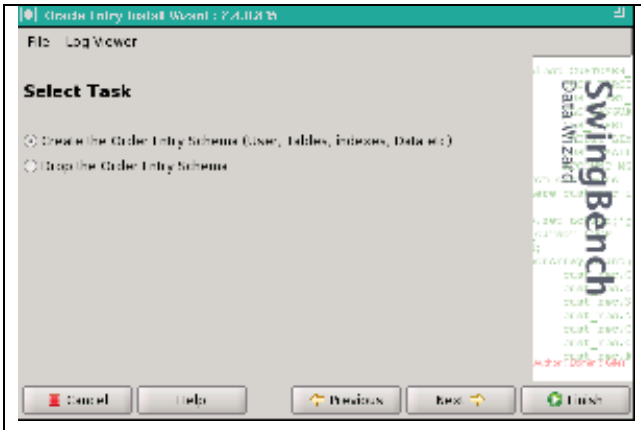
   **SQL>alter system set processes=6000 scope=spfile sid='*';**

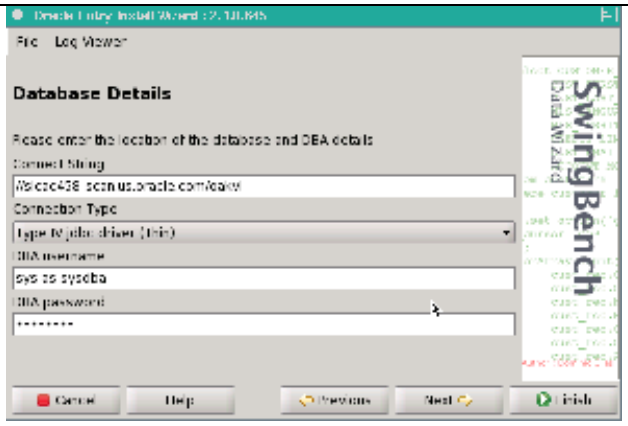
### Schema setup

It should be noted that the Order Entry workload generates data within the OE schema and is designed to introduce contention in the database. For consistency of results, it is recommended to rebuild the OE database schema after a few workload test run cycles to avoid inconsistency of results due to expansion of objects.

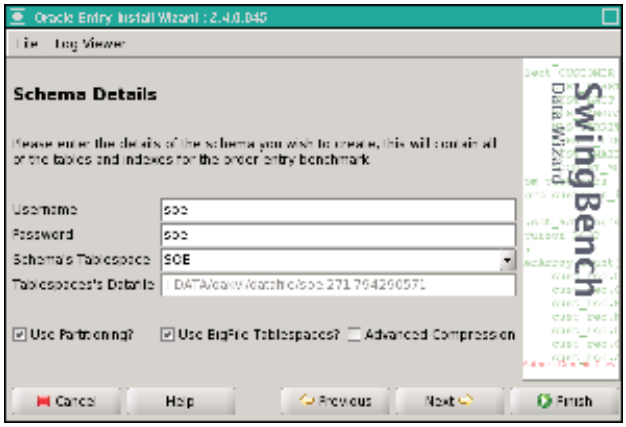
The Order Entry schema can be setup using the Swingbench *oewizard* graphical utility of Swingbench. The following screens illustrate the process of setting up the OE schema and generating data.

$cd /tmp/swingbench/bin
$./oewizard

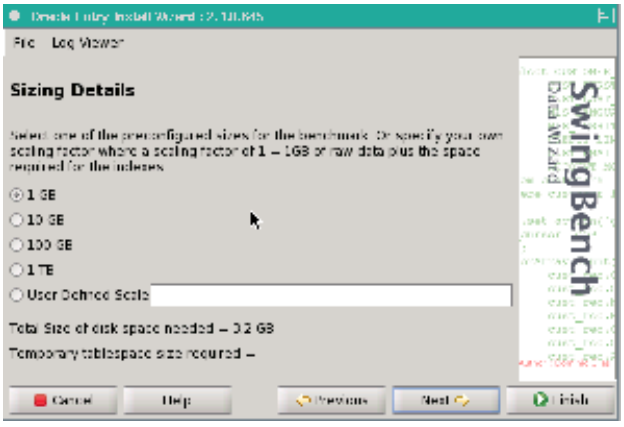


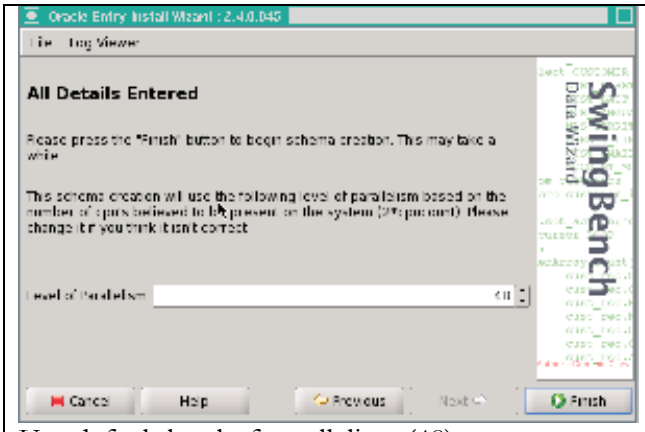

Specify the login and connection details



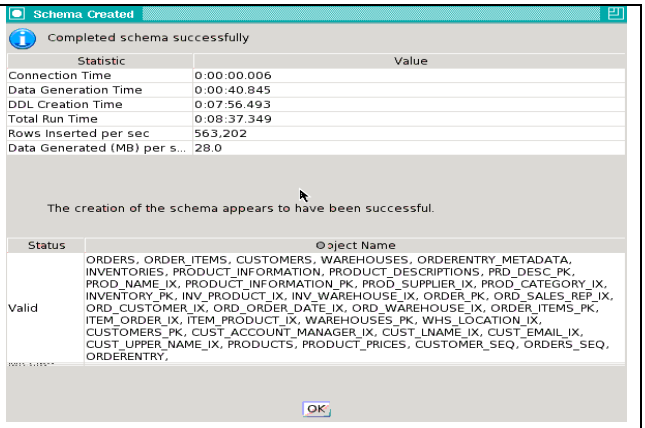

Select Partitioning option, BigFile Tbsp.




Use 1GB data size for these tests.




Use default level of parallelism (48).




Note: It may take up to 10 minutes for data generation process to completed

**Figure 4: Swingbench OE schema setup process**

The workload being tested does not utilize the following two indexes and these can be dropped to improve DML performance.

```
SQL> drop index soe.CUST_LNAME_IX
/
SQL> drop index soe.CUST_EMAIL_IX
/
```

The test database was initially created using the "VERY LARGE" template. When the database configuration were subsequently changed to fewer cores (LARGE, MEDIUM, SMALL, and VERY SMALL), the other database parameters such as SGA size, etc. were not changed. This decision was made to ensure that the measurements generate are fair for users who may simply want to limit the CPU configuration on Oracle Database Appliance and utilize other resources without any limitations.

During the OE workload testing, the database "PROCESSES" parameter was setup at a value of 6000.

Oracle Database Appliance supports pay-as-you-grow licensing model. As part of this testing, five different CPU configurations were tested by enabling a given number of cores (4, 8, 12, 16, and 24 cores) at a time. The 20 active core configuration was not tested.

Note that because of the use of hyper-threading within standard Oracle Database Appliance configuration, CPU performance is further improved.

### Swingbench workload setup

This section describes the setup of the test environment for testing the Order Entry (OLTP) workload.

The OE workload was configured with the following attributes and parameters.

| Attribute | Value |
|---|---|
| User Count | Various (10,000;8000;5500;3500;1750) |
| Think Time | InterMin=850ms; InterMax=1050ms |
| Pool Size | Min Pool Size=100; Max Pool Size=2500 |
| Run Time | 45 minutes |

**Table 4: Swingbench OE workload configuration settings**

The following query and transactional elements comprised the Order Entry workload.

1. Customer Registration
2. Process Orders
3. Browse Products
4. Order Products

While Swingbench supports both PL/SQL and Java transactions, only PL/SQL transactions were used in the testing performed for the purpose of documenting these results.

In order to ensure that the workload execution did not interfere with database performance, the Swingbench tool was run from a separate, external machine available on the same network as Oracle Database Appliance.

For a given user volume, multiple '*charbench'* sessions were invoked to execute the workload in parallel sessions. *Charbench* is the character (command line) interface for invoking *Swingbench*.

A close to realistic, constant think time of between 850ms and 1050ms was used on the client side for all test scenarios.

### OE workload testing performance results

The test results for the Order Entry workload execution highlight the following observations. The test results from different rounds of testing in various configurations are summarized below.

The total duration for each test was 45 minutes (2700 seconds).

| Oracle Database Appliance Configuration, and User Volume | Workload Type | Workload Element | Number of Total Transactions | Average Response Time (ms) | Average Transactions Per Second (TPS) |
|---|---|---|---|---|---|
| | | | | | |
| **24 Cores, 10000 Users** | Order Entry (OE) | Customer Registration | 2209069 | 6.1 | |
| | | Process Orders | 5522167 | 8.7 | **9816** |
| | | Browse Products | 13260326 | 3.9 | |
| | | Order Products | 5522566 | 20.2 | |
| | | | | | |
| **16 Cores, 8000 Users** | Order Entry (OE) | Customer Registration | 1857454 | 7.28 | |
| | | Process Orders | 4644645 | 10 | **8253** |
| | | Browse Products | 11141523 | 4 | |
| | | Order Products | 4646393 | 22.84 | |
| | | | | | |
| **12 Cores, 5500 Users** | Order Entry (OE) | Customer Registration | 1276844 | 5.2 | |
| | | Process Orders | 3190144 | 7 | **5673** |
| | | Browse Products | 7661642 | 3 | |
| | | Order Products | 3192881 | 19.1 | |
| | | | | | |
| **8 Cores, 3500 Users** | Order Entry (OE) | Customer Registration | 807148 | 6 | |
| | | Process Orders | 2022202 | 8 | **3596** |
| | | Browse Products | 4852526 | 3.1 | |
| | | Order Products | 2019776 | 23.1 | |
| | | | | | |
| **4 Cores, 1750 Users** | Order Entry (OE) | Customer Registration | 408946 | 4.7 | |
| | | Process Orders | 1022352 | 6.6 | **1817** |
| | | Browse Products | 2454016 | 3.1 | |
| | | Order Orders | 1021084 | 18.9 | |

**Table 5: Swingbench OE workload benchmark summary**

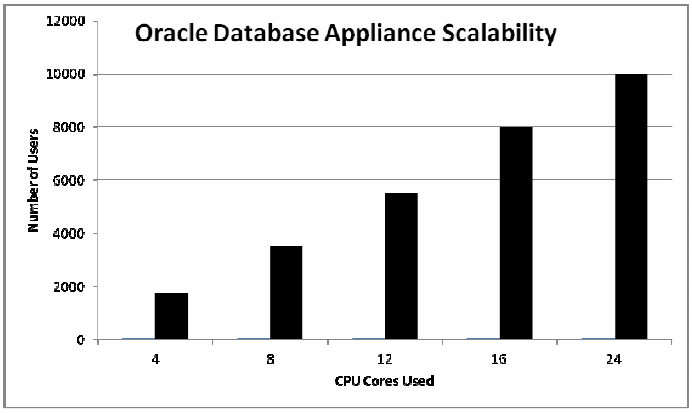
1. A 10000 Swingbench user workload was successfully executed with 24 cores active Oracle Database Appliance while maintaining transaction response times less than 24ms. Higher user volumes may be accommodated on Oracle Database Appliance if higher transaction response time can be tolerated.

2. Average transaction per second rate of more than 9800 was achieved during testing. At a higher transaction rate, the application performance somewhat degraded due to contention.

3. The maximum number of users and number of transactions for the given performance level scaled in a highly correlated and scalable manner as configurations were changed from VERY SMALL to VERY LARGE and user volumes were increased along with transaction volumes.

4. For each test case (run), the maximum CPU utilization was observed on each server node of the Oracle Database Appliance.

The following graph illustrates the scalability of the system as user volume increased and additional CPU cores were activated.




**Graph 1: Oracle Database Appliance Scalability**

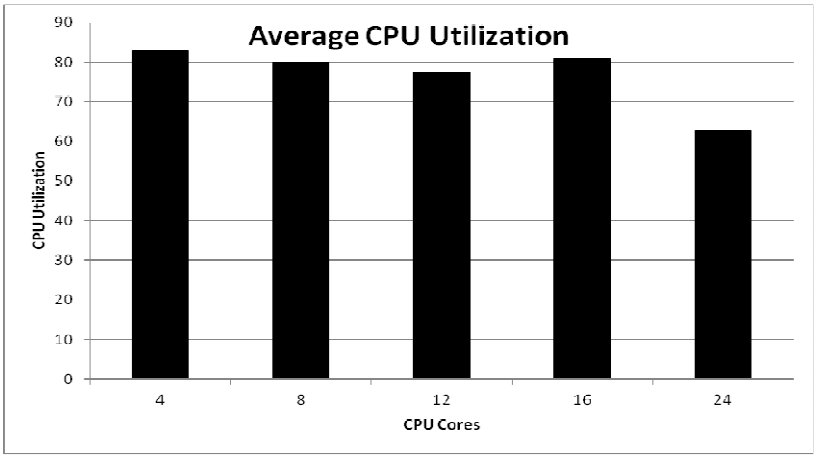
## Database and Operating System statistics

Operating System statistics can be collected using CHM/OS or OSWatcher. During this benchmark, the operating system statistics were gathered and analyzed using the OS Watcher tool. The OS Watcher tool is by default configured to run on each Oracle Database Appliance server node. The collected output of this tool is located under the /opt/oracle/oak/osw/archive directory. Please note that the standalone OS Watcher tool can be obtained from My Oracle Support website and can be installed on any non-ODA, legacy environment. Please refer to My Oracle Support (MOS) note 301137.1 for more information.

| | Average CPU Utilization (Appliance wide) | | |
|---|---|---|---|
| Cores | User | Sys | Idle |
| 24 | 52.65 | 10.15 | 32.75 |
| 16 | 68.10 | 12.60 | 13.90 |
| 12 | 63.85 | 13.55 | 16.85 |
| 8 | 64.40 | 15.65 | 12.55 |
| 4 | 63.85 | 19.10 | 9.65 |

**Table 6: Average CPU Utilization on Oracle Database Appliance**

During the OE workload testing, database connections were load balanced, albeit not perfectly evenly. The run queues fluctuated approximately between 30 and 60. The chart below shows the average CPU utilization on the two Oracle Database Appliance server nodes.




**Graph 2: Relative Average CPU Utilization in Different Configurations**

The Order Entry workload is a typical OLTP workload with significant CPU and REDO log activity. As observed, the IO activity was concentrated on disks belonging to the redo disk group. However, since the REDO disk group comprises of solid state disks, the service times associated with the redo IO activity were low. The following table summarizes the IO performance characteristics of the workload as observed during testing. Please note that these statistics pertain to the 24-cores-active run.

| Performance Characteristic | Measurements | |
|---|---|---|
| | HDD | SSD |
| Service Time (ms) | 4 | 0.2 |
| Disk Busy | 40% | 20% |
| Reads Per Second | 42 | (only archiving related) |
| Writes Per Second | 2170 | 6352 |
| Redo Generation Rate(MB/S) | 25 | |
| Wait Events | log file sync (Avg. wait time: 1ms),<br>gc cr block 2-way(Avg. wait time: 1ms),<br>gc current block 2-way (Avg. wait time: 1ms) | |

**Table 7: Performance observations during OE workload execution**

The above measurements illustrate the efficiency of the IO sub-system of Oracle Database Appliance. The Order Entry workload is designed to introduce contention. The wait events were observed to be well within acceptable range.

The workload profile was observed using AWR reports. The analysis of AWR reports indicated that the overall system load profile was as expected.

| Event | Waits | Time(s) | Avg Wait(ms) | % DB Time | Wait Class |
|---|---|---|---|---|---|
| | | | | | |
| DB CPU | | 50711.43 | | 43.72 | |
| gc current block 2-way | 27,236,965 | 18,688.46 | 0.7 | 16.11 | Cluster |
| gc cr block 2-way | 24,994,013 | 16,382.04 | 0.7 | 14.12 | Cluster |
| log file sync | 11,712,382 | 10,516.64 | 0.9 | 9.07 | Commit |
| db file parallel write | 1,191,650 | 7,710.49 | 6.5 | 6.65 | System I/O |

**Table 8: Top 5 database (cluster wide) wait events profile from OE 24 core run**

The system leveraged cache fusion effectively. During a 24 core run, 30305 blocks per second were exchanged across the cluster interconnect. The average wait of 0.72 ms for gc current block and 0.69ms for gc cr block indicated that the system benefitted from the efficiencies of cache fusion. As most data is served through the large memory available on Oracle Database Appliance, it reduces need for more costly disk IO, thus providing scalability, significantly eliminating the need for disk IO, and improving the overall performance of the IO sub-system.

# Part II. Sales History benchmark (DSS)

The Swingbench Sales History (SH) benchmark is based on the SH database schema and it is designed to test the performance of concurrent queries against large database tables. The Sales History workload comprises of read-only queries. One of the main objectives of this benchmark is to measure the throughput of the underlying IO sub-system.

## Database and schema setup

Oracle Database Appliance is optimized for general purpose database workload. However, for performance intensive application, some changes can enhance the database performance. For the Sales History (SH) workload executed during this testing project, the following configuration steps were taken to setup the test database and SH schema on Oracle Database Appliance, before executing the workload.

1. Setup NOARCHIVELOG mode

The database was put in noarchivelog mode to ensure no unnecessary logging takes place during the execution of the worklaod.

2. Set or modify certain database parameters

The following initialization parameters were configured or changed prior to workload execution.

| PARAMETER | VALUE | DESCRIPTION |
|---|---|---|
| sga_target | 12G | Total SGA Size |
| sga_max_size | 12G | Maximum SGA Size |
| pga_aggregate_target | 24G | Total PGA Size |
| parallel_force_local | TRUE | This parameter forces all the parallel server processes (PQ) to be allocated in the local instance only. |
| parallel_degree_policy | LIMITED | Enables automatic degree of parallelism for explicitly marked tables. Statement queuing and in-memory parallel execution is disabled. |
| parallel_min_time_threshold | 10 | Minimum execution time a statement should have before the statement is considered for parallelism. |

**Table 9: Database parameters used for Sales History benchmark**

Please note that when running the SH benchmark workload in a non-Orcle Database Appliance (legacy) environment, the parameter values similar to the above can be used as appropriate for the version of the database.

3. Set parallelism on database objects

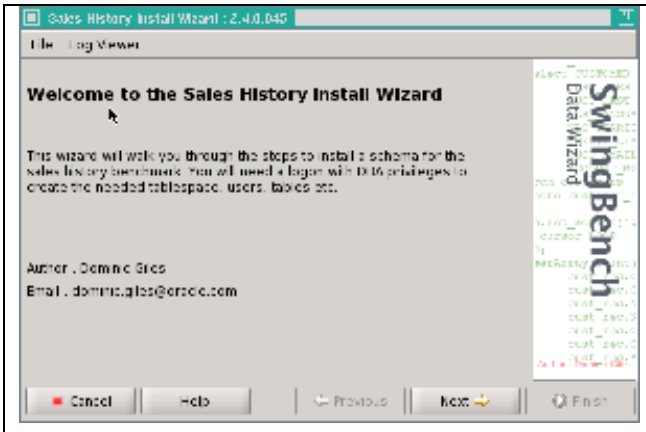
By default, all objects (tables and indexes) in the SH schema are created with parallel degree set to 1. This disables automatic degree of parallelism. This is undesirable for the Sales History workload as use of parallelism can improve SQL performance. The degree of parallelism on all the tables and the indexes was thus set to AUTO for all tables and indexes in the SH schema for enabling automatic degree of parallelism.

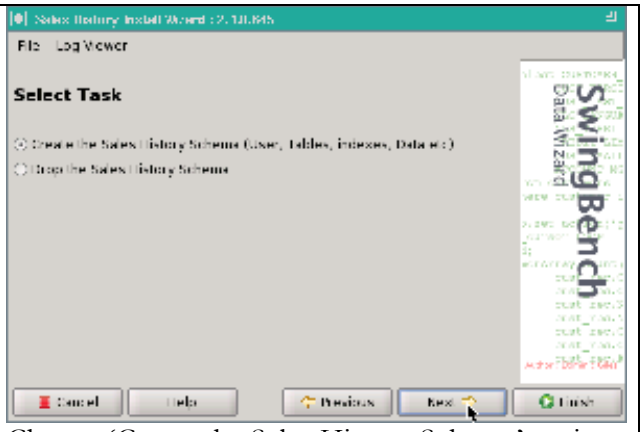
```
begin
      for i in (select table_name from all_tables where OWNER='SH' )
loop
       execute immediate 'alter table '||i.table_name||' parallel';
end loop;


for i in (select index_name from all_indexes where OWNER='SH' )
loop
      execute immediate 'alter index '||i.index_name||'parallel';
end loop;
end;
/
```

### Swingbench workload setup

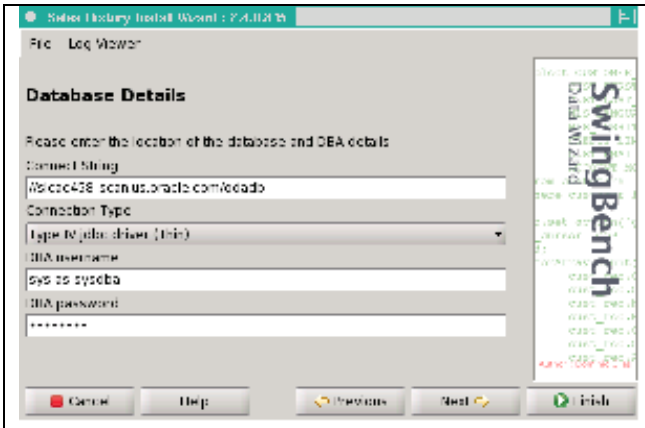
Swingbench provides the ***shwizard*** utility for installing the SH schema. This utility can be found under the bin directory inside Swingbench home. For example,

$ cd /home/oracle/swingbench/bin
$ ./shwizard

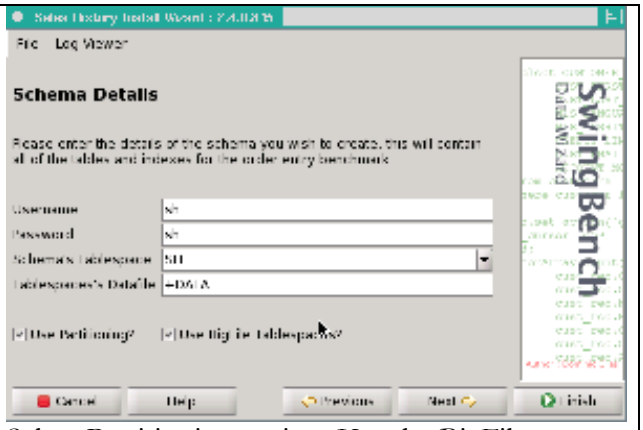


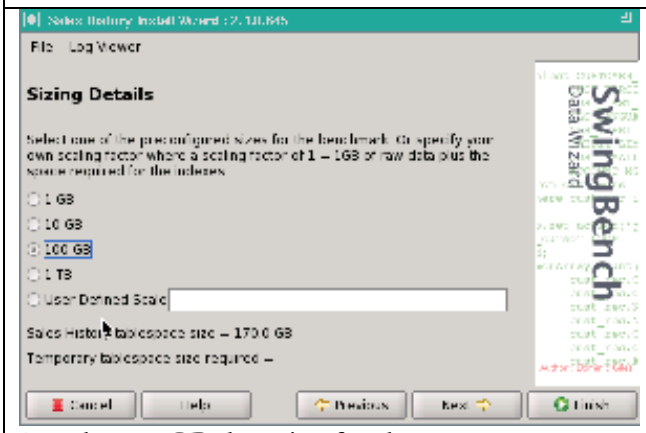

Choose 'Create the Sales History Schema' option



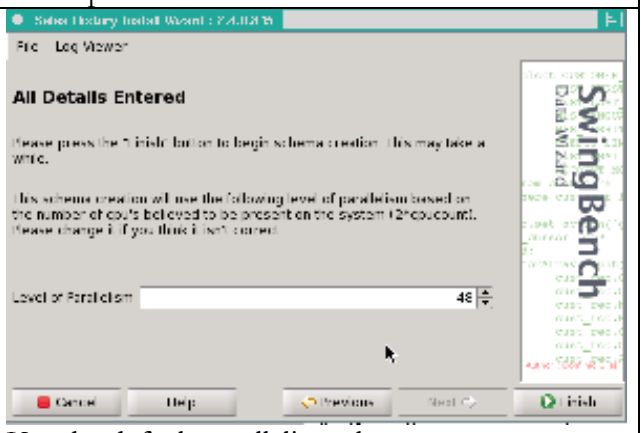

Specify the correct login details



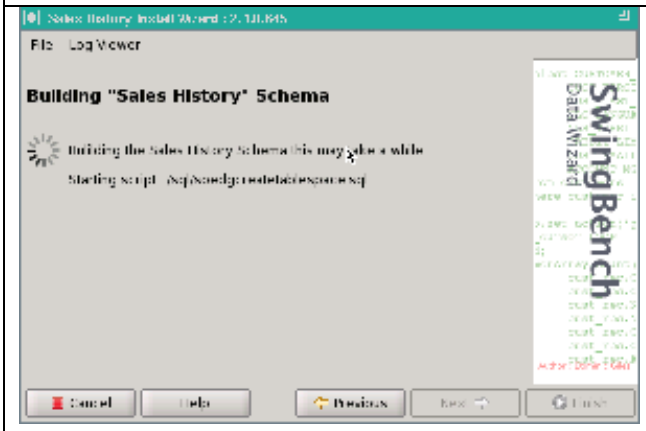

Select Partitioning option. Use the BigFile Tablespace.

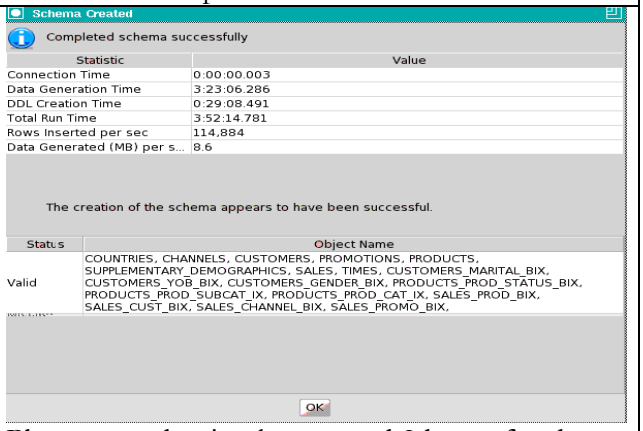



Use the 100GB data size for these tests




Use the default parallelism chosen




Please note that it takes around 2 hours for the data generation to be completed

**Figure 5: Swingbench SH benchmark setup process**

## SH workload testing performance results

### SH workload execution with Automatic Degree of Parallelism enabled

During this benchmark, the SH workload was executed on Oracle Database Appliance for a period of 30 minutes. Ten concurrent users were used to issue heavy workload against the database.

Please refer to Appendix C for the *Swingbench* configuration file used for this setup. This configuration file can also be used for conducting similar *Swingbench* tests in a non-Oracle Database Appliance environment.

| EXECUTIONS / REPORTS | TOTAL EXECUTIONS (IN 30 MINUTES) | AVERAGE RESPONSE(S) |
|---|---|---|
| Sales Rollup by Month and Channel | 29 | 200 |
| Sales Cube by Month and Channel | 38 | 194 |
| Sales Moving Average | 36 | 8 |
| Top Sales by Quarter | 40 | 192 |
| Sales within Quarter by Country | 34 | 195 |

**Table 10: Swingbench SH benchmark transaction performance summary**

With the workload executing with 10 concurrent users and automatic degree of parallelism enabled, the Oracle Database Appliance was able to produce a sustained IO throughput of more than 2400MB/sec. The average time for each Swingbench transaction (report) varied between 172 and 205 seconds.

At the database level, as observed from the AWR reports, during the above run, the average "physical read total bytes" were 2336 MB/sec and the average "physical read total IO requests" were 2368. These results were almost identical to additional test runs when degree of parallelism was set to 8 and when degree of parallelism was not enabled.

During the SH workload execution, the database primarily experienced direct path read waits. This was expected as most of the queries were performing parallel direct scan of the underlying tables.

Some "gc buffer busy acquire" and "gc buffer busy release" waits were also observed, but the time waited on these cluster related waits was rather insignificant.

## Database and Operating System statistics

During the Sales History workload benchmark, operating system statistics were collected using OS Watcher. The collected statistics pertain to the case where automatic degree of parallelism was used.

The disk statistics showed that the average IO service times were around 6ms with the hard disk drives running at 80% utilization and disk reads of about 120 per second per disk.

| Performance Characteristic | Measurements | |
|---|---|---|
| | ***HDD*** | ***SSD*** |
| Service Time (ms) | 6-8 | 0.2 |
| Disk Busy (%) | 80-85 | 0 |
| Throughput (MB/Sec) | 2483 | 0 |

**Table 11: System IO performance characteristics during SH workload execution**

For illustration purposes, the above four performance characteristics, namely, service time, disk busy (%), and throughput (reads and writes per second, and IO size) were taken from a single disk of both HDD and SSD type and extrapolated as the statistics on the other hard disk drives were also similar. Since the SH workload performs significant amount of full table scans, most of these reads are large 1MB reads.

As the Sales History (SH) workload is IO intensive but not CPU intensive, the overall CPU usage during this run was less than 20% on each of the Oracle Database Appliance server nodes.

## Conclusion

Oracle Database Appliance is designed to provide out-of-the-box performance for both OLTP and DSS workloads. When the full Oracle Database Appliance configuration was used, with all cores active to support a very large database configuration, the system was able to support a 10000 user Order Entry (OLTP) workload while providing sustained commit rate of more than 9800 transactions per second. With the same system configuration and the Sales History (DSS) workload, the system provided IO throughput of more than 2480 MB/Second. The testing indicated that some configuration adjustments and tuning were needed to obtain maximum performance from the Oracle Database Appliance.

## Appendix A: Post database deployment script (OE)

The script below was run on the database created on Oracle Database Appliance machine for setting up the Order Entry Test environment.

```
create bigfile tablespace soe datafile '+data' size 30g autoextend on
maxsize unlimited uniform size 1m segment space management auto
/

 begin
 for i in (select file_name from dba_data_files where tablespace_name
in (select tablespace_name from dba_tablespaces
                  where contents = 'UNDO' )
              ) loop
      execute immediate 'alter database datafile
'''||i.file_name||''' resize 10G';
 end loop;
 end;
 /

REM ODA by default is configured only with 2 redo log groups.
REM Adding an additional log group prior to the testing.

declare
l_group# number;
l_bytes number;
begin
      select max(group#) into l_group#  from v$log;
      select max(bytes) into l_bytes  from v$log;
      execute immediate 'alter database add logfile thread 1 group
'||to_number(l_group#+1)||'''+redo'' size '|| l_bytes;
      execute immediate 'alter database add logfile thread 2 group
'||to_number(l_group#+2)||'''+redo'' size '|| l_bytes;
end;
/

REM Please note that a DB restart is required for the changes below

alter system set audit_trail=none scope=spfile
/
alter system set audit_sys_operations=false scope=spfile
/
alter system set "_buffer_busy_wait_timeout"=2 scope=spfile sid='*'
/
alter system set resource_manager_plan='force:system_plan'
scope=spfile sid='*'
/
alter system set "_rollback_segment_count"=3000
/
```

## Appendix B: Swingbench configuration file for OE benchmark

The Swingbench configuration file (swingconfig.xml) that was used for the Order Entry testing is shown below. The highlighted entries were modified for the testing. This parameter file can be used for repeating the same tests on any non-Oracle Database Appliance platform. However, the database connection information needs to be updated as appropriate. The other parameters should not be changed for a correct comparison between an existing non-Oracle Database Appliance system and Oracle Database Appliance.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SwingBenchConfiguration
  xmlns="http://www.dominicgiles.com/swingbench/config">
  <Name>"Order Entry (PLSQL)"</Name>
  <Comment>""</Comment>
- <Connection>
  <UserName>soe</UserName>
  <Password>soe</Password>
  <ConnectString>//host458-scan.us.oracle.com/odadb</ConnectString>
  <DriverType>Oracle jdbc Driver</DriverType>
- <ConnectionPooling>
  <PooledInitialLimit>100</PooledInitialLimit>
  <PooledMinLimit>100</PooledMinLimit>
  <PooledMaxLimit>2500</PooledMaxLimit>
  <PooledInactivityTimeOut>0</PooledInactivityTimeOut>
  <PooledConnectionWaitTimeout>0</PooledConnectionWaitTimeout>
  <PooledPropertyCheckInterval>0</PooledPropertyCheckInterval>
  <PooledAbandonedConnectionTimeout>0</PooledAbandonedConnectionTimeout>
  </ConnectionPooling>
- <Properties>
  <Property Key="TcpNoDelay">true</Property>
  <Property Key="StatementCaching">50</Property>
  </Properties>
  </Connection>
- <Load>
  <NumberOfUsers>2000</NumberOfUsers>
  <MinDelay>0</MinDelay>
  <MaxDelay>0</MaxDelay>
  <InterMinDelay>850</InterMinDelay>
  <InterMaxDelay>1050</InterMaxDelay>
  <QueryTimeout>120</QueryTimeout>
  <MaxTransactions>-1</MaxTransactions>
  <RunTime>0:45</RunTime>
  <LogonGroupCount>1</LogonGroupCount>
  <LogonDelay>2</LogonDelay>
  <LogOutPostTransaction>true</LogOutPostTransaction>
  <WaitTillAllLogon>true</WaitTillAllLogon>
  <StatsCollectionStart>0:0</StatsCollectionStart>
```

```
  <StatsCollectionEnd>0:0</StatsCollectionEnd>
  <ConnectionRefresh>0</ConnectionRefresh>
- <TransactionList>
- <Transaction>
  <Id>Customer Registration</Id>
  <ShortName>NCR</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewCusto
    merProcess</ClassName>
  <Weight>10</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Browse Products</Id>
  <ShortName>BP</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.BrowsePr
    oducts</ClassName>
  <Weight>60</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Order Products</Id>
  <ShortName>OP</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewOrder
    Process</ClassName>
  <Weight>25</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Process Orders</Id>
  <ShortName>PO</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.ProcessO
    rders</ClassName>
  <Weight>25</Weight>
  <Enabled>true</Enabled>
    </Transaction>
    </TransactionList>
- <EnvironmentVariables>
  <Variable Key="SOE_NAMESDATA_LOC">data/names.txt</Variable>
  <Variable Key="SOE_PRODUCTSDATA_LOC">data/productids.txt</Variable>
  <Variable Key="SOE_NLSDATA_LOC">data/nls.txt</Variable>
    </EnvironmentVariables>
    </Load>
- <Database>
  <SystemUserName>system1</SystemUserName>
  <SystemPassword>welcome12</SystemPassword>
  <PerformAWRSnapShots>false</PerformAWRSnapShots>
    </Database>
- <SystemMonitor>
```

```
  <HostName>host458.us.oracle.com</HostName>
  <Username>root</Username>
  <Password>welcome1</Password>
    </SystemMonitor>
- <Preferences>
  <StartMode>manual</StartMode>
  <Output>swingbench</Output>
  <JumpToEvents>true</JumpToEvents>
  <TimingsIncludeSleep>false</TimingsIncludeSleep>
  <TimingsIn>milliseconds</TimingsIn>
  <StatisticsLevel>simple</StatisticsLevel>
  <OutputFile>results.xml</OutputFile>
- <Charts DefaultChart="Overview">
- <Chart>
  <Name>Transaction Response Time</Name>
  <Autoscale>true</Autoscale>
  <MaximumValue>-1.0</MaximumValue>
  <Logarithmic>true</Logarithmic>
    </Chart>
- <Chart>
  <Name>Transactions Per Minute</Name>
  <Autoscale>true</Autoscale>
  <MaximumValue>-1.0</MaximumValue>
  <Logarithmic>false</Logarithmic>
    </Chart>
- <Chart>
  <Name>DML Operations Per Minute</Name>
  <Autoscale>true</Autoscale>
  <MaximumValue>-1.0</MaximumValue>
  <Logarithmic>false</Logarithmic>
    </Chart>
    </Charts>
  <AllowedErrorCodes />
  <RefreshRate>1</RefreshRate>
- <OverviewCharts>
- <OverviewChart>
  <Name>Transactions Per Minute</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>Transactions Per Second</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>Response Time</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>CPU</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
```

```
<Name>Disk</Name>
<MaximumValue>2.147483647E9</MaximumValue>
  </OverviewChart>
  </OverviewCharts>
  </Preferences>
  </SwingBenchConfiguration>
```

## Appendix C: Swingbench configuration file for SH benchmark

The Swingbench configuration file (swingconfig.xml) that was used for the Sales History benchmark is shown below. The highlighted entries were modified for the testing. This parameter file can be used for repeating the same tests on any non-Oracle Database Appliance platform. However, the database connection information needs to be updated as appropriate. The other parameters should not be changed for a correct comparison between an existing non-Oracle Database Appliance system and Oracle Database Appliance.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SwingBenchConfiguration
    xmlns="http://www.dominicgiles.com/swingbench/config">
  <Name>Sales History</Name>
  <Comment>Configuration file for the Sales History Benchmark</Comment>
- <Connection>
  <UserName>sh</UserName>
  <Password>sh</Password>
  <ConnectString>//host458-scan.us.oracle.com/odadb</ConnectString>
  <DriverType>Oracle jdbc Driver</DriverType>
- <Properties>
  <Property Key="TcpNoDelay">true</Property>
  <Property Key="StatementCaching">50</Property>
  <Property Key="FetchSize">20</Property>
    </Properties>
- <InitilizationCommands>
  <Command>alter session set sql_trace=false</Command>
    </InitilizationCommands>
    </Connection>
- <Load>
  <NumberOfUsers>10</NumberOfUsers>
  <MinDelay>0</MinDelay>
  <MaxDelay>0</MaxDelay>
  <InterMinDelay>250</InterMinDelay>
  <InterMaxDelay>750</InterMaxDelay>
  <QueryTimeout>55</QueryTimeout>
  <MaxTransactions>-1</MaxTransactions>
  <RunTime>0:45</RunTime>
  <LogonGroupCount>1</LogonGroupCount>
  <LogonDelay>0</LogonDelay>
  <LogOutPostTransaction>false</LogOutPostTransaction>
  <WaitTillAllLogon>true</WaitTillAllLogon>
  <StatsCollectionStart>0:0</StatsCollectionStart>
  <StatsCollectionEnd>0:0</StatsCollectionEnd>
  <ConnectionRefresh>0</ConnectionRefresh>
- <TransactionList>
- <Transaction>
  <Id>Sales Rollup by Month and Channel</Id>
  <ShortName>SRMC</ShortName>
```

```
    <ClassName>com.dom.benchmarking.swingbench.dsstransactions.SalesRollu
    pByMonth</ClassName>
  <Weight>40</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Sales Cube by Month and Channel</Id>
  <ShortName>SCMC</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.dsstransactions.SalesCubeB
    yMonth</ClassName>
  <Weight>40</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Sales Moving Average</Id>
  <ShortName>SMA</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.dsstransactions.SalesMovin
    gAverage</ClassName>
  <Weight>40</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Top Sales by Quarter</Id>
  <ShortName>TSQ</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.dsstransactions.TopSalesWi
    thinQuarter</ClassName>
  <Weight>40</Weight>
  <Enabled>true</Enabled>
    </Transaction>
- <Transaction>
  <Id>Sales within Quarter by Country</Id>
  <ShortName>SQC</ShortName>

    <ClassName>com.dom.benchmarking.swingbench.dsstransactions.SalesByQua
    rterCountry</ClassName>
  <Weight>40</Weight>
  <Enabled>true</Enabled>
    </Transaction>
    </TransactionList>
    </Load>
- <Database>
  <SystemUserName>system1</SystemUserName>
  <SystemPassword>welcome1</SystemPassword>
  <PerformAWRSnapShots>false</PerformAWRSnapShots>
    </Database>
- <SystemMonitor>
  <HostName>host458.us.oracle.com</HostName>
  <Username>oracle</Username>
```

```
  <Password>welcome1</Password>
    </SystemMonitor>
- <Preferences>
  <StartMode>manual</StartMode>
  <Output>Swingbench</Output>
  <JumpToEvents>true</JumpToEvents>
  <TimingsIncludeSleep>false</TimingsIncludeSleep>
  <TimingsIn>milliseconds</TimingsIn>
  <StatisticsLevel>simple</StatisticsLevel>
  <OutputFile>results.xml</OutputFile>
  <Charts DefaultChart="Overview" />
  <AllowedErrorCodes />
  <RefreshRate>1</RefreshRate>
- <OverviewCharts>
- <OverviewChart>
  <Name>Disk</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>CPU</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>Response Time</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>Transactions Per Second</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
- <OverviewChart>
  <Name>Transactions Per Minute</Name>
  <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
    </OverviewCharts>
    </Preferences>
    </SwingBenchConfiguration>
```

## Appendix D: Script to issue workload and collect database statistics

The script shown below can be used to run multiple charbench sessions in parallel against a database. Copy the scripts into a folder on the system where the Swingbench binaries are installed. Please note that the highlighted entries within the script must be modified, as needed.

Execute the script as follows:

$perl loadgen.pl –u <no. of Swingbench users>

The output of the script will be as shown below..

Total Number of Application Users : 1200
Average Transactions Per Second : 2511.05

```
-----------------------------------------------------------
Application Module                    Txn Count   Avg Res Time
Customer Registration                    125684   3.00
Process Orders                           314146   3.00
Browse Production                        754385   1.00
Order Products                           313674   9.00
Exiting…
-----------------------------------------------------------
```

Ensure to set the following environment variables prior to executing the script.

export LD_LIBRARY_PATH=$ORACLE_HOME/rdbms/lib:$ORACLE_HOME/lib
export SB_HOME=<Directory where swingbench is installed>

Please note that the script uses perl module XML::Simple and DBI. Please ensure that these modules are installed prior to running this script.

The Perl DBI module requires oracle client libraries for connecting to the database. Please ensure that these modules are installed prior to running the script below. Install the Oracle client on the system where Swingbench is installed. The oracle client libraries are not needed by Swingbench as it uses JDBC thin driver for connectivity. The client libraries are only used for collecting AWR snapshots and reports.

Ensure to set the following environment variables prior to executing the script.

export LD_LIBRARY_PATH=$ORACLE_HOME/rdbms/lib:$ORACLE_HOME/lib
export SB_HOME=<Directory where swingbench is installed>

```
---------------------- loadgen.pl -------------------------------

#!/usr/bin/perl

use strict;
use warnings;
use Getopt::Long;
use Data::Dumper;
use POSIX;
use POSIX qw/ceil/;
use POSIX qw/strftime/;
use threads ( 'yield', 'stack_size' => 64*4096, 'exit' =>
'threads_only', 'stringify');
use DBI qw(:sql_types);
use vars qw/ %opt /;
use XML::Simple;
use Data::Dumper;

### Please modify the below variables as needed #######
### If the service does not exist, please create it using SRVCTL

my $host="host458-scan.us.oracle.com";
my $service_name="odadb";
my $port=1521;
my $dbauser="system";
my $dbapwd="welcome1";

### Please modify the above variables as needed #######


my $rundate=strftime("%Y%m%d%H%M", localtime);
my @app_modules = ("Customer Registration","Process Orders","Browse
Products","Order Products");

my $snap_id;
my $b_snap;
my $e_snap;

my %opts;
my $tot_uc;
my $cb_sess;
my $counter;
my $uc=1000;
my $max_cb_users=1000;
my $min_cb_instances=5;
my $output_dir;
my $awr_interval_in_secs=1800;

my $sb_home;

use Cwd();
my $pwd = Cwd::cwd();
```

```
my $sb_output_dir=$pwd."/sb_out";
my $awr_dir=$pwd."/awr";


sub usage { "Usage: $0 [-u <No_of_Users>] \n" }

sub chk_n_set_env
{

if ($ENV{SB_HOME})
{
 $sb_home=$ENV{SB_HOME};
}
else
{
 print "The environment variable SB_HOME is not defined. \n";
 print "Re-run the program after setting SB_HOME to the swingbench
home directory. \n";
 exit 1;
}
}

sub set_cb_parameters
{
      if ( ceil($tot_uc/$max_cb_users) <= $min_cb_instances ) {
       $cb_sess = $min_cb_instances;
       # $uc = int($tot_uc/10);
       $uc = ($tot_uc - ($tot_uc %
$min_cb_instances))/$min_cb_instances;
      }

      if ( ceil($tot_uc/$max_cb_users) > $min_cb_instances ) {
        $cb_sess = ceil($tot_uc/$max_cb_users);
        $uc = $max_cb_users;
      }

      my $rc=$tot_uc;

      print "User count $uc \n";
      print "Total SB Sessions  $cb_sess \n";
}

sub process
{
    my ($l_counter) = @_;
    # print "User count".$l_counter."\n";
    # print "Out dir".$sb_output_dir."\n";
    # print "Run Date ".$rundate."\n";
    system ("$sb_home/bin/charbench -u soe -p soe -uc $uc -r
$sb_output_dir/results_"."$uc"."_users_"."$rundate"."_$l_counter".".x
ml -s");
```

```
}


sub create_out_dir {

if ( -d "$_[0]" ) {
 print "Directory "."$_[0]"."Exists\n";
}
else{
 system("mkdir -p $_[0]");
}
}

sub generate_awr_snap
{
# my $dbh = DBI-
>connect("dbi:Oracle://$host:$port/$service_name",'system','welcome1'
) || die "Database connection not made";
my $dbh = DBI-
>connect("dbi:Oracle://$host:$port/$service_name","$dbauser","$dbapwd
") || die "Database connection not made";

$dbh->{RowCacheSize} = 100;

my $sql = qq{ begin dbms_workload_repository.create_snapshot; end; };

my $sth = $dbh->prepare( $sql );

$sth->execute();

$sql = qq{ select max(snap_id) from dba_hist_snapshot };

$sth = $dbh->prepare( $sql );

$sth->execute();

$sth->bind_columns( undef,\$snap_id );

$sth->fetch();

$sth->finish();

$dbh->disconnect();
}

sub process_xml_output {

my $txn_cnt;
my $avg_rt;
my @files;
my $cr_tc=0;
my $cr_to_rt=0;
```

```
my $po_tc=0;
my $po_to_rt=0;
my $bp_tc=0;
my $bp_to_rt=0;
my $op_tc=0;
my $op_to_rt=0;
my $num_users=0;
my $avg_tps=0;
my $app_module;
my $file;
my $xml;

@files = <$sb_output_dir/\*$rundate*>;

foreach $file (@files) {

$xml = new XML::Simple;
my $ResultList = $xml->XMLin($file);

#print "Processing output file $file \n";
#printf "%-22s %10s %8s\n","Application Module","Txn Count","Avg Res
Time";
#print "-------------------------------------------------------\n";

$num_users = $num_users + $ResultList->{Configuration}-
>{NumberOfUsers};
$avg_tps = $avg_tps + $ResultList->{Overview}-
>{AverageTransactionsPerSecond};

foreach $app_module (@app_modules) {
$txn_cnt=$ResultList->{TransactionResults}->{Result}-
>{"$app_module"}->{TransactionCount};
$avg_rt=$ResultList->{TransactionResults}->{Result}->{"$app_module"}-
>{AverageResponse};

#printf "%-22s %10s %8s\n",$app_module,$txn_cnt,$avg_rt;

if ($app_module eq "Customer Registration") {
        $cr_tc = $cr_tc+$txn_cnt;
        $cr_to_rt = $cr_to_rt+($avg_rt*$txn_cnt);
}
elsif ($app_module eq "Process Orders") {
        $po_tc = $po_tc+$txn_cnt;
        $po_to_rt = $po_to_rt+($avg_rt*$txn_cnt);
}
elsif ($app_module eq "Browse Products") {
        $bp_tc = $bp_tc+$txn_cnt;
        $bp_to_rt = $bp_to_rt+($avg_rt*$txn_cnt);
}
elsif ($app_module eq "Order Products") {
        $op_tc = $op_tc+$txn_cnt;
        $op_to_rt = $op_to_rt+($avg_rt*$txn_cnt);
```

```
}
}

#printf "\n";
}

print "Total Number of Application Users : ".$num_users."\n";
print "Average Transactions Per Second : ".$avg_tps."\n";

print "-------------------------------------------------------\n";
printf "%-22s %16s %8s\n","Application Module","Txn Count","Avg Res
Time";
print "-------------------------------------------------------\n";

foreach $app_module (@app_modules) {
        if ($app_module eq "Customer Registration") {
                printf "%-22s %16s
%0.2f\n",$app_module,$cr_tc,($cr_to_rt/$cr_tc);
        }
        elsif ($app_module eq "Process Orders") {
                printf "%-22s %16s
%0.2f\n",$app_module,$po_tc,($po_to_rt/$po_tc);
        }
        elsif ($app_module eq "Browse Products") {
                printf "%-22s %16s
%0.2f\n",$app_module,$bp_tc,($bp_to_rt/$bp_tc);
        }
        elsif ($app_module eq "Order Products") {
                printf "%-22s %16s
%0.2f\n",$app_module,$op_tc,($op_to_rt/$op_tc);
        }
}
}

GetOptions(\%opts,  'users|u=i' => \$tot_uc, 'runid|r=i' =>
\$rundate,) or die usage;

print "Total # of users is $tot_uc \n";
print "Run ID is $rundate \n";

create_out_dir($sb_output_dir);
create_out_dir($awr_dir);
chk_n_set_env;
set_cb_parameters;

my $rc;

for($counter = 1; $counter <= $cb_sess; $counter++){
  $rc = $tot_uc - ($counter*$uc);
  if ( $rc < 0 ) {
     $uc = ($rc+$uc);
  }
```

```
    my $thr = threads->create('process',$counter);
    print "Charbench "."$counter Starting with usercount $uc"."\n";
    $thr->detach();
    sleep 30;
}

sleep 120;

generate_awr_snap;

$b_snap=$snap_id;

print "Start Snap $b_snap"."\n";

sleep $awr_interval_in_secs;

generate_awr_snap;

$e_snap=$snap_id;

print "End Snap $e_snap"."\n";

system("$pwd/genawr.sh", $b_snap, $e_snap, $rundate, $awr_dir);

my $running;

while (1) {
$running = `ps -ef |grep $rundate| grep -v grep |wc  -l`;
if ($running == 0)
{
 process_xml_output;
 print " Exiting .. \n";
 exit 0;
}
sleep 10;
}

----------------------- loadgen.pl -----------------------------
Script to generate awr reports ..
-------------------------genawr.sh -----------------------------

#!/bin/bash
export ORACLE_SID=odadb1
export ORACLE_HOME=/u01/app/oracle/product/11.2.0.3/dbhome_1
#export AWR_DIR=/home/oracle/ram/scripts/awr

l_dbid=1267760840
l_start_snapid=$1
#l_end_snapid=`expr $1 + 1`
l_end_snapid=$2;
l_runid=$3;
AWR_DIR=$4;
```

```
l_start_snapid=$(sed -e 's/^[[:space:]]*//' <<<"$l_start_snapid");
l_end_snapid=$(sed -e 's/^[[:space:]]*//' <<<"$l_end_snapid");
l_runid=$(sed -e 's/^[[:space:]]*//' <<<"$l_runid");

l_awr_log_file="${AWR_DIR}/awrrpt_1_${l_start_snapid}_${l_end_snapid}
_${l_runid}.log"

echo $l_awr_log_file;

$ORACLE_HOME/bin/sqlplus -s system/welcome1@host458-
scan.us.oracle.com/odadb << EOC
set head off
set pages 0
set lines 132
set echo off
set feedback off
spool $l_awr_log_file
SELECT
output
FROM
TABLE
(dbms_workload_repository.awr_report_text($l_dbid,1,$l_start_snapid,$
l_end_snapid ));
spool off
exit;
EOC

l_awr_log_file="$AWR_DIR/awrrpt_2_${l_start_snapid}_${l_end_snapid}_$
{l_runid}.log"

$ORACLE_HOME/bin/sqlplus -s system/welcome1@host458-
scan.us.oracle.com/odadb << EOC
set head off
set pages 0
set lines 132
set echo off
set feedback off
spool $l_awr_log_file
SELECT
output
FROM
TABLE
(dbms_workload_repository.awr_report_text($l_dbid,2,$l_start_snapid,$
l_end_snapid ));
spool off
exit;
EOC
-------------------------genawr.sh ----------------------------
```

Evaluating and Comparing Oracle Database Appliance Performance
January 2013

Author(s): Ramachandran P., Paulo Qiu, Ravi Sharma, Sanjay Singh

Contributing Authors: Roland Knapp, Michael Zoll, A-Team / RAC Pack

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



**Hardware and Software, Engineered to Work Together**