ORACLE
DATABASE APPLIANCE

Evaluating and Comparing Oracle Database Appliance X6-2-HA Performance

ORACLE WHITE PAPER | JANUARY 2017

ORACLE

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Executive Summary

The Oracle Database Appliance X6-2-HA is an Oracle Engineered System that saves time and money by simplifying deployment, maintenance, and support of high availability database solutions. Optimized for the world's most popular database— Oracle Database—it integrates software, compute, storage, and network resources to deliver high availability database services for a wide range of custom and packaged online transaction processing (OLTP), in-memory database, and data warehousing applications. All hardware and software components are engineered and supported by Oracle, offering customers a reliable and secure system with built-in automation and best practices. In addition to accelerating the time to value when deploying high availability database solutions, the Oracle Database Appliance X6-2-HA offers flexible Oracle Database licensing options and reduces operational expenses associated with maintenance and support.

The Oracle Database Appliance X6-2-HA is designed for simplicity and reduces that element of risk and uncertainty to help customers deliver higher availability for their databases. The number of processor cores, amount of main memory, and SSDs storage capacity in the fully integrated appliance is balanced to provide optimal database performance for a wide range of enterprise applications. Oracle Database sizing templates ensure that the system resources are properly allocated for database workloads running on the appliance.

The Oracle Database Appliance X6-2-HA is architected with SSDs storage to increase database performance and system reliability. Database workloads can realize a significant improvement in input/output operations per second (IOPS) and bandwidth while achieving extremely low latency and CPU overhead with SSDs storage over similar systems configured with conventional SAS drives.

Oracle database appliance runs standard, time-tested software components, such as Oracle Linux operating system (OS), Oracle Relational Database Management System (RDBMS), Oracle Clusterware and Oracle Automatic Storage Management (ASM). The pre-built, pre-tested, and pre-tuned configuration ensures that Oracle Database Appliance X6-2-HA can be deployed rapidly and does not require typical efforts to tune the configuration. Further, Oracle Database Appliance X6-2-HA 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 workload 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 X6-2-HA vis-à-vis a legacy environment.

With all 40 cores active, Oracle Database Appliance supported 19000 concurrent Swingbench users, and more than 19500 Swingbench transactions per second while maintaining an average response time of less than 10ms.

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 X6-2-HA. 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

The objective of this white paper is to quantify performance of Oracle Database Appliance X-2 HA when running OLTP like workloads. This performance information is presented to show number of users, number of transactions, and transaction performance.

This workload performance information is presented in easy to use terms such as number of users, number of transactions per minute, and transaction performance. The system performance is presented in terms of resource utilization, data processing rates.

Note: The specific workload tested during this benchmark is the Swingbench Order Entry (OE) workload which is TPC-C like.

A secondary objective of this white paper is to illustrate a process for executing test workload in non-Oracle Database Appliance (legacy) environments and comparing the results against those captured in this white paper. This is facilitated by documenting the process of Swingbench setup and the test results for the Swingbench Order Entry workload, 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.

Oracle Database Appliance X6-2 Model Family

The Oracle Database Appliance X6-2 model family is the fifth generation of Oracle Database Appliances. Each appliance consists of hardware and software that save customers time and money by simplifying deployment, maintenance, and support of Oracle Database environments.

The Oracle Database Appliance X6-2S, X6-2M,X6-2L and X6-2-HA expand the reach of the database appliance to support various deployment scenarios and database editions. For more details, Please refer Oracle Database Appliance X6-2 Model Family white paper published on OTN. Oracle database Appliance X6-2 family contains X6-2S/X6-2M/X6-2L and X6-2-HA.

Oracle Database Appliance X6-2-HA Architecture

The Oracle Database Appliance X6-2-HA hardware is a 6U rack-mountable system containing two Oracle Linux servers and one storage shelf. Each server features two 10-core Intel® Xeon® processors E5-2630 v4, 256 GB of memory, and 10-Gigabit Ethernet (10GbE) external networking connectivity.

The two servers are connected together via a redundant InfiniBand or optional 10GbE interconnect for cluster communication and share direct-attached high performance solid-state SAS storage. The storage shelf in the base system is half populated with ten solid-state drives (SSDs) for data storage, totaling 12 TB of raw storage capacity.

The storage shelf in the base system also includes four 200 GB high endurance SSDs for database redo logs to improve performance and reliability. For more details, please refer Oracle Database Appliance X6-2-HA Data Sheet published on OTN.

Oracle Database Appliance X6-2-HA Configuration Templates

Oracle Database Appliance provides and uses several standard database configuration templates for creating databases of different classes and sizes. Databases created using these templates automatically inherit Oracle database implementation best practices such as most optimal setting for database parameters, best practices based configuration and placement of database components such as database files, recovery files, REDO log files, control files, etc.

Table-1 illustrates the different database creation templates available for creating databases of different sizes on Oracle Database Appliance. More detailed sizing matrix for Oracle Database Appliance is available in Oracle product documentation.

Shape	CPU Core	SGA (GB)	PGA (GB)	Processes	Redo Log File Size (GB)	Log Buffer (MB)
odb1s	1	2	1	200	1	16
odb1	1	4	2	200	1	16
odb2	2	8	4	400	1	16
odb4	4	16	8	800	1	32
odb6	6	24	12	1200	2	64
odb8	8	32	16	1600	2	64
odb10	10	40	20	2000	2	64
odb12	12	48	24	2400	4	64

odb16	16	64	32	3200	4	64
odb20	20	80	40	4000	4	64

Table 1 Oracle Database Appliance X-2-HA sizing templates

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 dominicgiles.com

Swingbench version 2.5.971 was used to perform the testing documented in this white paper. For more information about Swingbench, please refer to Swingbench documentation.

Swingbench provides four separate benchmarks, namely, Order Entry, Sales History, Calling Circle, and Stress Test. Order Entry schema creation supports version 1 and version 2 type objects. Version 1 type schema was used during testing. Additionally, Oracle 12c JDBC drivers were used during testing. For the tests described in this paper, Swingbench Order Entry benchmark was used for OLTP workload 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.

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 dominicgiles.com site. Download the Swingbench version 2.5.971 or later. Please note that 2.5.971 was used during this testing.

2. Setup Swingbench

   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 Order Entry test covered in this paper.

3. Specify non-blocking random number generators

   Oracle 12c JDBC driver requires a random number for encrypting connect string. By default this random number is generated from /dev/random. In certain situations, when using /dev/random, 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. You can use /dev/urandom instead of /dev/random to address this issue. This change can be made by specifying /dev/urandom (in the <swing benchhome>/launcher/launcher.xmlfile)in the arguments when launching Swingbench as follows:

Old Entry	New Entry
<jvmargset id="base.jvm.args"> <jvmarg line="-Xmx1024m"/> <jvmarg line="-Xms512m"/> <!--<jvmarg line="- Djava.util.logging.config.file=log.properties"/>--> </jvmargset>	<jvmargset id="base.jvm.args"> <jvmarg line="-Xmx1024m"/> <jvmarg line="-Xms512m"/> <jvmarg line="- Djava.security.egd=file:/dev/./urandom"/> <!--<jvmarg line="- Djava.util.logging.config.file=log.properties"/>--> </jvmargset>

Table 2 Non blocking random number entry

Benchmark Setup

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

Database and Schema Setup

The tests outlined in this document were performed with Oracle Appliance Manager (OAK) 12.1.2.9 image running Oracle Database release 12.1.0.2. When comparing the performance of Oracle Database Appliance X6-2-HA 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 X6-2-HA uses standard templates and standard, optimized parameter settings.

Note that the corresponding settings for these parameters may not match their settings on legacy systems. In order to perform proper comparison, configuration parameters for the database configured in the legacy environment, should be similar to those in Oracle Database Appliance X6-2-HA environments as much as possible.

Additionally, the default database parameter settings for Oracle Database Appliance X6-2-HA, 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 the database.

Database Setup

While Oracle Database Appliance X6-2-HA allows you to create single instance enterprise edition and RAC enterprise edition database, during the course of performance tests, only Oracle RAC enterprise edition database was used. The database was setup using the OLTP template for the Odb-20 configuration. The OLTP templates automatically configure the SGA and PGA in the ratio of 2:1.

The following changes should be made together and the database should be restarted once for these changes to take effect.

1. Reset CPU_COUNT

   The tests were conducted by allocating all the CPU cores assigned to the system.

   ```
   SQL> alter system reset cpu_count scope=spfile sid='*';
   ```

2. Disable Database Auditing

Database auditing is enabled by default on Oracle Database Appliance. Database auditing can cause unnecessary performance impact and can skew test results. It is recommended to turn database auditing off for the duration of the test.

```
SQL> alter system set audit_trail=none scope=spfile sid='*';

SQL> alter system set audit_sys_operations=false scope=spfile sid='*';
```

3. Set higher rollback segment count and tune UNDO

   The Order Entry work load 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 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"=12000 scope=spfile
sid='*';

SQL> alter system set "_undo_autotune"=false scope=spfile sid='*';

SQL> alter system set undo_retention=180 scope=spfile sid='*';
```

4. Set the "_buffer_busy_wait_timeout" parameter to 2

   Database performance related bug 13344323 should be addressed. This bug is related to the possibility of a missing post from DBWR that can cause buffer busy waits. As a workaround the _buffer_busy_wait_timeout parameter should be set to 20ms.

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

5. Create additional REDO log groups

   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 should be created. For the testing conducted during this project, a total of 4 redo log groups per instance of 8GB each were created on Oracle Database Appliance.

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

6. Resize undo tablespace datafiles to 20G.
7. Resize temp tablespace to 5G.
8. Resize sysaux tablespace to 10G.
9. Increate the ASH size for database

```
SQL>alter system set "_ash_size"=1000M scope=spfile sid='*';
```

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

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

11. Increase PROCESSES parameter value.

```
SQL>alter system set processes=16000 scope=spfile sid='*';
```

12. Disable incremental checkpoint

    The order entry workload was tested with disabling the incremental checkpoint feature by resetting the fast_start_mttr_target parameter.

```
SQL>alter system reset fast_start_mttr_target scope=spfile sid='*';
```

13. Disable the Resource Manager Plan

    In order to avoid resource manager accidently kicking in and enforcing cpu cap, the resource manager was completely turned off for the tests.

    ```
    SQL> alter system set "_resource_manager_always_off" = false scope=spfile
    sid='*';
    ```

14. High Waits on Log File Sync

    The Order Entry work load might show high waits on 'Log File Sync'. This happens because of high commit frequency by OLTP Order Entry workload. We have set the priority of LGWR processes to high to address high commit frequency. For details, please refer MOS Note# 1373500.1.

    ```
    SQL>alter system set "_high_priority_processes"='VKTM|LMS*|LG*'
    scope=spfile sid='*';
    ```

    During OLTP Order Entry workload, we have also disabled multiple log writer workers and tested with single log writer process. For details, please refer to MOS Note# 2174075.1.

    ```
    SQL> alter system set "_use_single_log_writer"=true 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 steps illustrate the process of setting up the OE schema and generating data.

1. Start the Order Entry Install wizard using oewizard utility.

   ```
   $./oewizard
   ```

   You should see Welcome Wizard. Click Next.

2. Choose version 1 on Select Benchmark Version screen. Click Next.
3. Select Create the Order Entry Schema (User,Tables,Indexes,Data etc.) on Select Task screen. Click Next.
4. Provide the connection details on Database Detail screen.
   - Connect String: //<scan-name>:<Port>/<DB Service Name>
   - DBA username: sys as sysdba
   - DBA Password: <Sys Password>

   Click Next
5. Provide username and password details on Schema Detail screen.
   - Username : soe
   - Password: soe
   - Schema Tablespace SOE

   Click Next.
6. Select following on Database Options screen.
   - Partitioning Model : Hash Partitioning
   - Compression Used : No Compression

- Tablespace type : Bigfile Tablespace
- Indexing Used : All Indexes

Click Next

7. Select 1GB on Sizing Details screen and click Next.
8. Select default level of parallelism and click Finish.
9. You need to wait for 10-15 minutes and you should see Completed Schema Sucessfully Screen with all the Valid Objects.

For details of graphical screen to setup Order Entry Schema using swingbench utility, please refer Schema Setup section under Benchmark Setup in Evaluating and Comparing Oracle Database Appliance Performance white paper.

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

```
SQL> drop index "SOE"."CUST_LNAME_IX”;

SQL> drop index "SOE"."CUST_EMAIL_IX”;

SQL> drop index "SOE"."ITEM_PRODUCT_IX";

SQL> drop index "SOE"."CUST_ACCOUNT_MANAGER_IX";

SQL> drop index "SOE"."ORD_CUSTOMER_IX";

SQL> drop index "SOE"."ORD_ORDER_DATE_IX";

SQL> drop index “SOE”.”CUST_UPPER_NAME_IX”;
```

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:

Attribute	Value
User Count	Various (19000,15550,12250,10250,8250,4250)
Think Time	InterMin=850ms; InterMax=1050ms
Run Time	50 Minutes

Table 3 Swingbench OE workload configuration settings

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

- Customer Registration
- Process Order
- Browse Product
- 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 on Oracle Database Appliance X6-2-HA are summarized below. The total duration for each test was 50 minutes (3000 seconds).

ODA X6-2-HA				
Oracle Database Appliance X6-2-HA Configuration, and User Volume	Workload Element	Total Transactions	Average Response Time	Average Transactions Per Second (TPS)
8 Cores : 4250 users	Customer Registration	667696	2.63	4456
	Process Order	1671853	4.17	
	Browse Products	4008979	2.65	
	Order Products	1672960	6.89	
16 Cores : 8250 users	Customer Registration	1297303	2.77	8650
	Process Order	3245997	4.1	
	Browse Products	7784397	2.8	
	Order Products	3243811	6.5	
20 Cores :10250 users	Customer Registration	1612188	2.8	10750
	Process Order	4031490	4.2	
	Browse Products	9671769	2.8	
	Browse Products	4030480	6.5	
24 Cores :12250 users	Customer Registration	1922982	3.41	12400
	Process Order	4814150	5.6	
	Browse Products	11546098	3.43	
	Order Products	4811022	8.4	
32 Cores : 15550 users	Customer Registration	2432061	3.4	16250
	Process Order	6090601	5.8	
	Browse Products	14608074	3.6	
	Order Products	6085573	8.8	
40 Cores : 19000 users	Customer Registration	2979223	3.83	19900
	Process Order	7455847	6.6	
	Browse Products	17897358	4.03	
	Order Products	7452026	9.4	

Table 4 Swingbench Workload benchmark summary

- A 19000 Swingbench user workload was successfully executed on a fully sized Oracle Database Appliance while maintaining transaction response times under 10ms. Higher user volumes could be accommodated if higher transaction response time is allowed. Average transaction per second rate of 19500 was achieved during testing.
- 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 4 core configuration to a 20 core configuration and user volumes were increased along with transaction volumes.
- For each test case (run), the maximum CPU utilization was observed on the Oracle Database Appliance.

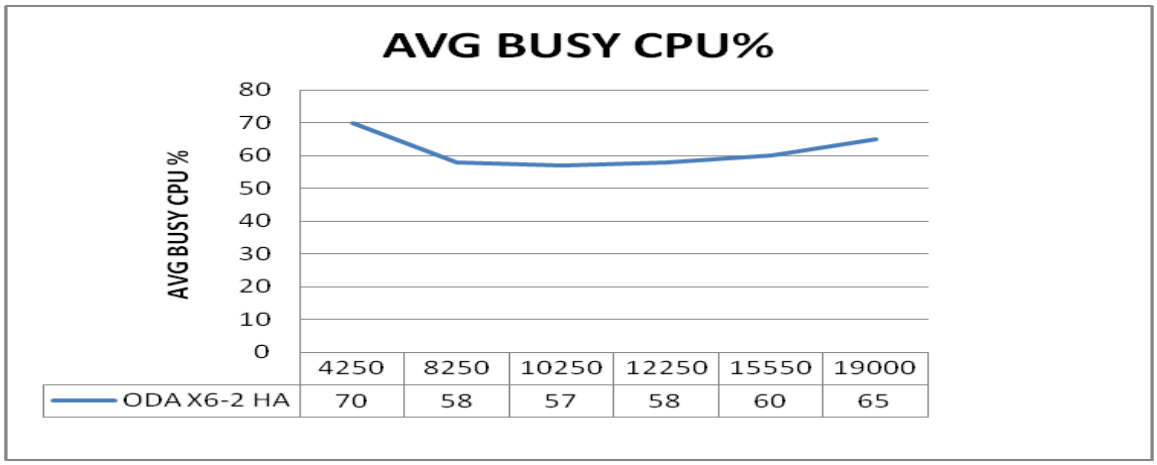
Database and Operating System Statistics

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 Oracle Database Appliance X6-2-HA. The collected output of this tool is located under the /opt/oracle/oak/oswbb/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 MOS note 301137.1for more information.

ODA X6-2-HA				
Cores	Average CPU Utilization %			Average Busy%
	Usr	Sys	Idle	
8	55.9	14	30	70
16	49	9	42	58
20	49	8	43	57
24	48	10	42	58
32	50	10	40	60
40	55	10	35	65

Table 5 Average CPU Utilization under different workload

The chart below shows the average CPU utilization on the Oracle Database Appliance X6-2-HA.

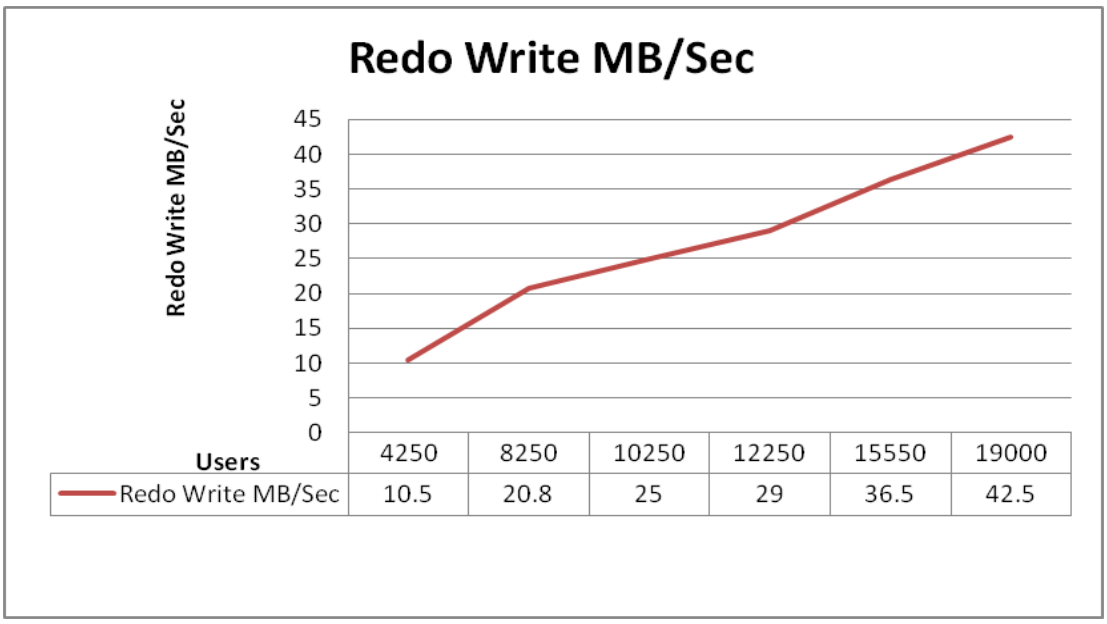


The Swingbench Order Entry workload is a typical OLTP workload with significant CPU and REDO log activity. As observed during testing, the IO activity was concentrated on disks belonging to the REDO disk group. However, since the REDO disk group comprises of SSDs disks, the service times associated with the redo IO activity were very low. The following table summarizes the IO performance characteristics of the workload as observed during testing. This statistics captured from AWR from IO profile section.

	Statistics	8 cores	16 cores	20 cores	24 cores	32 cores	40 cores
ODA X6-2-HA	Redo Write MB/Sec	10.5	20.8	25	29	36.5	42.5
	Reads Request /Sec	23	33	37	39	51	57
	Writes Request /Sec	7519	13369	15212	15826	16000	16500

Table 6 IO Performance observation during OE workload execution

The chart below shows the redo write MB/sec is increasing as the numbers of users are increasing on the Oracle Database Appliance X6-2-HA.

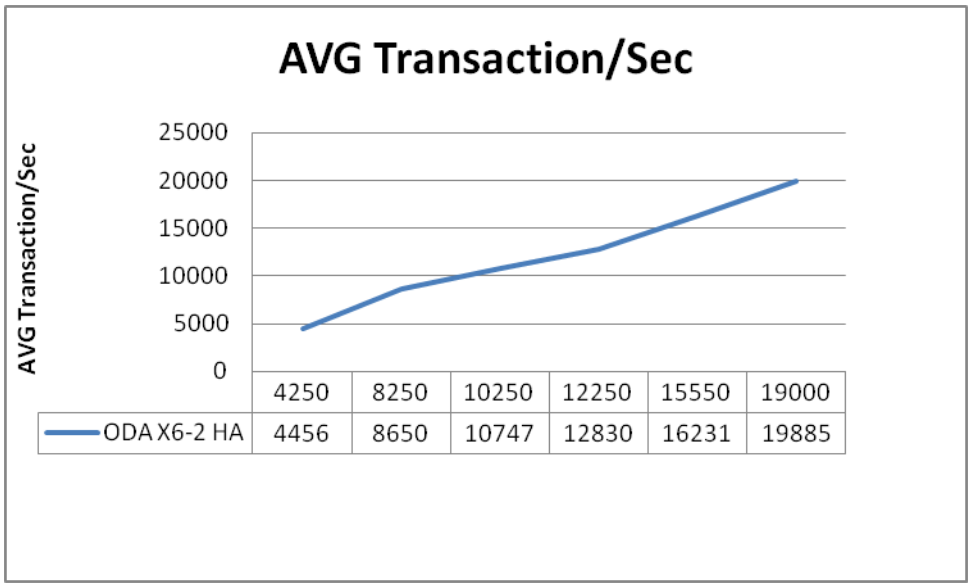


The above table and chart illustrates the efficiency of the IO sub-system of Oracle Database Appliance X6-2-HA. The Order Entry workload is designed to introduce contention. The wait events were observed to be well within acceptable range. In addition, the work load scaled almost linearly as CPU capacity was added to the configuration.

	Statistics	8 cores	16 cores	20 cores	24 cores	32 cores	40 cores
	Users	4250	8250	10250	12250	15550	19000
ODA X6-2-HA	Average TPS	4456	8650	10750	12400	16250	19900

Table 7 CPU core count/Users vs Transaction per second and User count correlation

Graph below illustrated the linearity in a pictorial format.



The correlation with user population is also linear as evident from the data above.

Troubleshooting performance mismatch between traditional and Oracle Database Appliance deployments

Though the Oracle Database Appliance X6-2-HA has Oracle best practices built-in and is optimized for Oracle databases. The number of processor cores, amount of main memory, and SSDs storage capacity in the fully integrated appliance is balanced to provide optimal database performance for a wide range of enterprise applications.

It is possible that when you deploy your application database on Oracle Database Appliance the performance may not match with that in your legacy environment. When performance is relatively better in the Oracle Database Appliance environment, then it may not be a concern at all. In many situations, performance improvements may even be expected as you move away from older slower performing, low capacity systems to Oracle Database Appliance. However, if the performance of your database workload is as per your expectations, you may consider the following:

1. Ensure that database on Oracle Database Appliance is created properly

   Oracle Database Appliance X6-2-HAprovides database templates which are pre-optimized and proportional database parameter settings. It is recommended that you use Oracle Appliance Manager command-line interface (odacli) to create databases on Oracle Database Appliance.

2. Verify that workload is identical

   If you are running different workloads (SQL, commit rates, etc.) in the legacy versus the Oracle Database Appliance environment, then same performance levels cannot be expected.

3. Ensure that the network topology during your legacy and Oracle Database Appliance tests is comparable

   Network latency can be a significant drag on performance in your legacy or Oracle Database Appliance environment and should be accounted for. For example, if you run your workload clients on a different network than that of the database hosts, there could be significant latency in the transaction patch as compared to when the workload client and the database server are on the same network.

Note: The tests during the writing of this white paper were performed with both the workload clients and the database servers running on the same subnet although they were on separate hosts.

4. Verify System Resource Allocation

   Your old system may simply have more resources than you have provisioned for your database on Oracle Database Appliance. This could have a significant impact on observed performance behavior in the two environments.

5. Verify SQL execution plans

   Check execution plan of relevant SQL statements in your legacy and Oracle Database Appliance environments. If execution plans differ, try to identify the cause and address it or justify it. For example, the data volumes in the two environments may be different. There may be index differences, or lack of proper optimized statistics, etc.

6. Verify database resource allocation

   Verify database parameters, CPU count, and so forth in the two environments. In order to perform a meaningful comparison, you should allocate at least the same level of resources on the Oracle Database Appliance. Note, that parameter settings should be appropriate for your RDBMS version and may differ from your legacy system if an older version of RDBMS is used.

7. Verify Database Parameters

   Do not use performance inhibiting database parameters on Oracle Database Appliance. If you migrated your database from you're your legacy environment to Oracle Database Appliance, make sure you are not using obsolete, un-optimized settings for parameters.

8. Account for additional benefits Oracle Database Appliance provides

   Oracle Database Appliance provides features such as database block checking and verification to protect against data corruption. These features may consume some, albeit small, capacity but are generally desirable. For theoretical performance comparison purposes, these parameters may be temporarily disabled if you are not using these in your legacy environment. It is however strongly recommended that you use these protective features to reduce data corruption risks.

Considerations When Comparing High-Capacity to High-Performance Configurations

In most situations, the Oracle Database Appliance X6-2 high performance configuration outperforms the Oracle Database Appliance X5-2 high capacity configuration. For workloads where the hot working set will not fit in the limited X5-2 flash storage, testing shows the X6-2 is able to support over five times as many transactions per second with 20-40x faster response time. However, the X5-2 performance paper suggests the X5-2 can out perform the X6-2. When comparing the results of this paper with those published for the X5-2, there are some important considerations to keep in mind.

- The X6-2 provides much higher IO capacity with the base configuration, and is able to drive the measured workloads with a single storage shelf. The X5-2 performance numbers required an expansion shelf to provide the necessary IO capacity. Thus, the X6-2 configuration (as tested for these papers) is substantially less expensive.
- The results published in the X5-2 performance paper make use of the limited flash in the system to boost the results. In some cases it is possible to get very good results by storing the workload's hot working set

in the flash diskgroup. Some of this tuning was done with the X5-2 performance report to achieve the measured performance. Depending on your workload and working set size, this may or may not be feasible. With the X6-2, all data is always in flash—no tuning is required.

- On the X5-2, the ACFS accelerator volume may need to consume some of the limited space in the FLASH diskgroup. This accelerator volume stores in flash metadata for HDD-backed file systems to speed performance. It can either be staged in the FLASH diskgroup, or the REDO diskgroup. If there are many databases running on the system, you will need to store it to the FLASH diskgroup to allow space for redo logs. There is no ACFS accelerator volume in the X6-2, as there are no HDDs in the system.

You can get very good performance from the Oracle Database Appliance X5-2, should your hot working set fit in flash. If this is the case, the X5-2 will yield the best price performance. If the working set will not fit in flash, and your workload becomes IO limited, then you will be able to drive much higher workloads on the Oracle Database Appliance X6-2. Oracle recommends carefully considering the IO requirements of your workload, and evaluating how much of the hot working set will fit in flash when evaluating which Oracle Database Appliance is best for you.

Conclusion

Oracle Database Appliance X6- HA supports and provides excellent performance for typical database workloads. When the full Oracle Database Appliance X6-2-HA configuration was used, with all cores active to support a very large database configuration, the system was able to support a 19000 user Swingbench Order Entry workload while providing sustained commit rate of more than 19500 per second. The testing conducted during the course of writing of this white paper indicated that some configuration adjustments and tuning were needed for different workloads to obtain maximum performance from the Oracle Database Appliance X6-2-HA.

Appendix A: Post database deployment script

The script below was run on the database created on ODA 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

/
```

Appendix B: Swingbench Configuration File

The Swingbench configuration file (swingconfig.xml) that was used for the workload 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-ODA 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-ODA system and Oracle Database Appliance.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<SwingBenchConfiguration

xmlns="http://www.dominicgiles.com/swingbench/config">

<Name>"Order Entry (PLSQL) V2"</Name>

<Comment>""</Comment>

<Connection>

<UserName>soe</UserName>

<Password>soe</Password>

<ConnectString>//oda-scan.us.oracle.com:1521/odadb.us.oracle.com</ConnectString>

<DriverType>Oracle jdbc Driver</DriverType>

<ConnectionPooling>

<PooledInitialLimit>1000</PooledInitialLimit>

<PooledMinLimit>1000</PooledMinLimit>

<PooledMaxLimit>2500</PooledMaxLimit>

<PooledInactivityTimeOut>0</PooledInactivityTimeOut>

<PooledConnectionWaitTimeout>0</PooledConnectionWaitTimeout>

<PooledPropertyCheckInterval>0</PooledPropertyCheckInterval>

<PooledAbandonedConnectionTimeout>0</PooledAbandonedConnectionTimeout>

</ConnectionPooling>

<Properties>
```

```
<Property Key="FetchSize">20</Property>
<Property Key="StatementCaching">50</Property>
</Properties>
</Connection>
<Load>
<NumberOfUsers>100</NumberOfUsers>
<MinDelay>0</MinDelay>
<MaxDelay>0</MaxDelay>
<InterMinDelay>850</InterMinDelay>
<InterMaxDelay>1050</InterMaxDelay>
<QueryTimeout>120</QueryTimeout>
<MaxTransactions>-1</MaxTransactions>
<RunTime>0:50</RunTime>
<LogonGroupCount>1</LogonGroupCount>
<LogonDelay>1</LogonDelay>
<LogOutPostTransaction>false</LogOutPostTransaction>
<WaitTillAllLogon>true</WaitTillAllLogon>
<StatsCollectionStart>0:15</StatsCollectionStart>
<StatsCollectionEnd>0:45</StatsCollectionEnd>
<ConnectionRefresh>0</ConnectionRefresh>
<TransactionList>
<Transaction>
<Id>Customer Registration</Id>
<ShortName>NCR</ShortName>
<ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewCustomerProcess<
/ClassName>
<Weight>10</Weight>
<Enabled>true</Enabled>
</Transaction>
<Transaction>
<Id>Browse Products</Id>
<ShortName>BP</ShortName>
```

```
<ClassName>com.dom.benchmarking.swingbench.plsqltransactions.BrowseProducts</Cla
ssName>
<Weight>60</Weight>
<Enabled>true</Enabled>
</Transaction>
<Transaction>
<Id>Order Products</Id>
<ShortName>OP</ShortName>
<ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewOrderProcess</Cl
assName>
<Weight>25</Weight>
<Enabled>true</Enabled>
</Transaction>
<Transaction>
<Id>Process Orders</Id>
<ShortName>PO</ShortName>
<ClassName>com.dom.benchmarking.swingbench.plsqltransactions.ProcessOrders</Clas
sName>
<Weight>25</Weight>
<Enabled>true</Enabled>
</Transaction>
</TransactionList>
<EnvironmentVariables>
<Variable Key="SOE_NLSDATA_LOC">data/nls.txt</Variable>
<Variable Key="SOE_NAMESDATA_LOC">data/names.txt</Variable>
</EnvironmentVariables>
</Load>
<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>DML Operations Per Minute</Name>
<Autoscale>true</Autoscale>
<MaximumValue>-1.0</MaximumValue>
<Logarithmic>false</Logarithmic>
</Chart>
<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>
</Charts>
<AllowedErrorCodes>
<ErrorCode Type="ORA">4063</ErrorCode>
</AllowedErrorCodes>
<RefreshRate>1</RefreshRate>
<OverviewCharts>
<OverviewChart>
<Name>Disk</Name>
<MinimumValue>0.0</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
```

```
</OverviewChart>
<OverviewChart>
<Name>CPU</Name>
<MinimumValue>0.0</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
</OverviewChart>
<OverviewChart>
<Name>Response Time</Name>
<MinimumValue>0.0</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
</OverviewChart>
<OverviewChart>
<Name>Transactions Per Second</Name>
<MinimumValue>0.0</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
</OverviewChart>
<OverviewChart>
<Name>Transactions Per Minute</Name>
<MinimumValue>0.0</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
</OverviewChart>
</OverviewCharts>
</Preferences>
</SwingBenchConfiguration>
```

Appendix C: 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 : 2000

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…

---

export LD_LIBRARY_PATH=$ORACLE_HOME/rdbms/lib:$ORACLE_HOME/lib

export SB_HOME=<Directory where swingbench is installed>

Please note that the loadgen.pl 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 #######


my $host="oda-large.us.oracle.com";

my $service_name="oda1001.us.oracle.com";

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 swingbenchhome direcotry.
\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"."."xml -s");
}

sub create_out_dir {
if ( -d "$_[0]" ) {
print "Direcory "."$_[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 ResTime";
#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;
my $sleep_time;
$sleep_time=300/$cb_sess;
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 $sleep_time;
```

```
}
# sleeping for 10 minutes so that load get adjusted.
sleep 600;

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 host=oda-large.us.oracle.com
l_dbid=1382218992
export svc=oda1001.us.oracle.com
export ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1
export port=1521
```

```
export AWR_DIR=/home/oracle/ram/scripts/awr

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"

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


echo $l_awr_log_file;

cd ${AWR_DIR}

$ORACLE_HOME/bin/sqlplus -s system/welcome1@$host:$port/$svc << EOC

set head off

set pages 0

set lines 132

set echo off

set feedback off

spool "awrrpt_1_${l_start_snapid}_${l_end_snapid}_${l_runid}.log"

SELECT

output

FROM

TABLE

(dbms_workload_repository.awr_report_text($l_dbid,1,$l_start_snapid,$l_end_snapi
d ));

spool off

exit;

EOC
```

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

Evaluating and Comparing Oracle Database Appliance X6-2-HA Performance
January, 2017
Author: Sanjay Singh, Paramdeep Saini , Paulo Qiu
Contributing Authors: Ramsu, Ravi Sharma, Ram P, RACPack team

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