ORACLE

Business / Technical Brief

Evaluating and Comparing Oracle Database Appliance Performance

Updated for Oracle Database Appliance X9-2 HA

ORACLE TECHNICAL BRIEF | MARCH 2023

March, 2023, Version 1.12

DISCLAIMER

The performance results in this paper are intended to give an estimate of the overall performance of the Oracle Database Appliance X9-2-HA system. The results are not a benchmark, and cannot be used to characterize the relative performance of Oracle Database Appliance systems from one generation to another, as the workload characteristics and other environmental factors including firmware, OS, and database patches, which includes Spectre/Meltdown fixes, will vary over time. For an accurate comparison to another platform, you should run the same tests, with the same OS (if applicable) and database versions, patch levels, etc. Do not rely on older tests or benchmark runs, as changes made to the underlying platform in the interim may substantially impact results.

Table of contents

Introduction and Executive Summary

Highly available Oracle database systems include Oracle Database Appliance (ODA) X9-2-HA. Hardware, software, networking, and storage are all included in this integrated, pre-built, pre-tuned, packaged database solution, which has an 8-rack unit configuration with a modest footprint. Oracle Database Appliance X9-2 HA's hardware and software combination offers redundancy and protects against all single points of failures in the system.

Oracle Database Appliance X9-2-HA is a two-node cluster based on the most recent Intel Xeon processors with direct attached SAS storage that includes Solid State Disk Drives (SSDs – high performance model) or a combination of Hard Disk Drives and SSDs (high-capacity model), depending on the preferences of customers. Oracle Linux operating system (OS), Oracle Relational Database Management System (RDBMS), Oracle Real Application Clusters software (RAC), Oracle Clusterware, and Oracle Automatic Storage Management are among the common, tried-and-true software components that run on ODA. Oracle Database Appliance can be quickly and easily installed, thanks to the pre-built, pre-tested, and pre-tuned configuration, which eliminates the need for manual configuration and tuning procedures.

There is a customer demand on getting to know the platform's performance characteristics before purchasing a new platform. This technical brief's goal is to illustrate and document the performance of a simulated workload running on an Oracle Database Appliance X9-2-HA system using Swingbench, which is a free performance testing tool. The performance of such a standardized workload running on the Oracle Database Appliance could be assessed and compared to the performance of the same workload running in their legacy environment by system architects and database administrators. Although this document describes the maximum IOPs and MBPS of the ODA X9-2 HA high-performance model, it is not in the scope of this technical brief to describe the steps of testing the maximum IO capabilities of the machine.

Oracle Database Appliance X9-2-HA is an extremely potent, highly available database server despite its compact size. It proved scalable performance for high-volume database workloads throughout the performance testing and benchmark process. Oracle Database Appliance X9-2-HA high-performance model supported more than 35k concurrent Swingbench transactions per second with all 32 CPU cores per node-enabled.

Audience

Database architects, CTOs, CIOs, heads of IT departments, and IT purchase managers who may be interested in comprehending and analyzing Oracle Database Appliance's performance capabilities will find this technical brief helpful. This information could be also useful for Oracle System Administrators, Storage Administrators, and Oracle Database Administrators when performance testing is done in their own setups. They will also be familiar with the best practices that can help get the most performance of different workload types running on an Oracle Database Appliance.

Objective

A quick glance at Oracle Database Appliance's hardware setup reveals that the system's architecture is designed for high availability and solid performance right out of the box. Customers frequently and correctly request baseline comparison performance statistics for different types of standard workloads due to the presence of numerous components in any system and due to the contrasting nature of distinct workloads. When they move their database(s) to a new environment, this helps them project their own performance experience and expectations.

This technical brief's main goal is to quantify Oracle Database Appliance performance under what can be regarded as a normal database workload. Number of users, transactions per minute, and transaction execution time are just a few of the simple measures used to describe how well the workload performed. Data processing rates and resource utilization are used to describe the system performance.

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

The secondary objective of this document is to outline the process to execute the same test workload in non-ODA (legacy) environments or on earlier ODA models and comparing the results against the ones captured and presented in this technical brief. This objective is facilitated by documenting the process of *Swingbench* setup and test results from multiple *Swingbench* Order Entry workload runs on ODA X9-2 HA performance models.

This study was conducted by using Swingbench workloads on an ODA X9-2 HA that was running 19.17 ODA software release version.

User and transaction volumes were varied along with CPU configurations. Tests were performed using Swingbench's SOE schema generated by Swingbench's Order Entry wizard. Swingbench tests can be run locally or remotely. Remote execution requires a client machine, and the test result can be affected by the capacity of the client machine and the network latency between the client and the database server – in our case between the client machine and the ODA. To keep the setup simple, and easily reproducible and to eliminate external factors that can impact performance tests, this document focuses only on local Swingbench tests, when the database and Swingbench are running on the ODA. Customers can run the identical Swingbench workloads on their legacy systems and compare the results with the ones documented in the paper.

The best way to measure the capabilities of ODA X9-2 HA and compare it to the legacy system is to capture the workload on the legacy environment using Real Application Testing (RAT) and replay it on the ODA. RAT also provides various options to speed up the number of replayed transactions, which can help to determine if the environment is indeed future-proof and could support future growth in transaction numbers. RAT requires a license. Please refer to https://www.oracle.com/manageability/enterprise-manager/technologies/real-application-testing.html

Oracle Database Appliance supports databases on bare metal and inside KVM-based DB Systems. All tests in this technical brief were executed on ODA bare metal.

If you want to use a different test workload or you have a different Oracle Database Appliance model, you may still use the approach outlined in this technical brief and run any given workload on both the Oracle Database Appliance environment and your legacy non-Oracle Database Appliance environments and compare the results.

Oracle Database Appliance Deployment Architecture

For system details refer to Oracle Database Appliance X9-2-HA Data Sheet available at https://www.oracle.com/a/ocom/docs/engineered-systems/database-appliance/oda-x9-2-ha-datasheet.pdf

Oracle Database Appliance Configuration Templates

For building databases of various sizes and shapes, Oracle Database Appliance offers and employs number of common database setup templates. Oracle best practices for an Oracle database implementation, such as the most ideal database parameter settings and best practices-based storage setup, are automatically inherited by databases using these templates.

The sort of workload you want to run on your Oracle Database Appliance determines the database shapes that will be used to configure your databases. Different workload types can use different database shapes, including OLTP, DSS, and in-memory. On an ODA, you can easily select the most suitable database shape for your workload.

Once a database is deployed using a given template, users are not restricted from altering the database parameters based on their requirements.

Refer to Oracle Database Appliance X9-2-HA Deployment and User's Guide (Appendix E Database Shapes for Oracle Database Appliance) for a list of database creation templates (for OLTP workloads in this case) available for creating databases of different shapes and sizes on Oracle Database Appliance.

The configuration of the database operating on the Oracle Database Appliance and the database running in the non-Oracle Database Appliance environment should be quite similar to perform a meaningful performance comparison.

Oracle Database Appliance X9-2-HA's high-performance model with a fully populated shelf with SSDs offers significant IO capacity for users to deploy demanding OLTP, DSS, Mixed, and In-memory workloads. It should be noted that in a different set of test cycle, it offered up to 2,789,783 IOPS and throughput of up to 23,187 MBPS with a fully occupied, twin storage shelf configuration, compared to 1,660,754 IOPS and 14,727 MBPS with a single storage shelf. The tests for IOPS and MBPS were performed with 8K and 1M random reads, respectively. Depending on the workload mix (READ/WRITE ratios), some variances were seen.

It may be noted that in the Oracle Database Appliance X9-2-HA system, the Spectre and Meltdown vulnerabilities are mitigated in Silicon (not software), which is more efficient and eliminates the overhead of these mitigations. The software mitigations may not have been included in older performance tests either run by Oracle or third-parties. Any direct comparisons to older benchmarks may understate the performance improvements.

Table 1 presents a mathematically calculated estimate of the possible IO capacity for various shapes implemented on the Oracle Database Appliance X9-2-HA system. Keep in mind that the amount of available IO capacity is not affected by the shapes you use.

SHAPE	ODB1S	ODB1	ODB2	ODB4	ODB8	ODB12	ODB16	ODB24	ODB28	ODB32
Measure	Single storage shelf configuration									
IOPS	51 899	51 899	103 797	207 594	415 189	622 783	830 377	1 245 566	1 453 160	1 660 754
MBPS	460	460	920	1 841	3 682	5 523	7 364	11 045	12 886	14 727
	Double storage shelf configuration									
IOPS	87 181	87 181	174 361	348 723	697 446	1 046 169	1 394 892	2 092 337	2 441 060	2 789 783
MBPS	725	725	1 449	2 898	5 797	8 695	11 594	17 390	20 289	23 187

Table 1 Sample calculated IOPS/MBPS capacity for certain database shapes deployed on Oracle Database Appliance X9-2-HA

This technical brief does not cover HDD storage configuration which is a possible variant of the standard (default) all SSD storage configuration for Oracle Database Appliance X9-2-HA model.

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 https://www.dominicgiles.com/downloads/

Swingbench version 2.7 was used to perform the tests documented in this technical brief. For more information about Swingbench, please refer to Swingbench documentation available at https://www.dominicgiles.com/index.html

Swingbench provides six separate benchmarks, namely, OrderEntry, SalesHistory, TPC-DS Like, JSON, CallingCircle and StressTest. For all benchmarks described in this paper, Swingbench Order Entry (OE) V2 benchmark was used for OLTP workload testing.

In the earlier version of the technical brief, Order Entry version 1 was used for the benchmarks, but that version is not in use anymore. A decision was made to include performance metrics in this paper for X8-2 HA as well, to make previous ODA models comparable to X9-2 HA.

Swingbench Download and Setup

Swingbench can be setup on a client machine or on the same machine where the database runs. Setting it up a client machine might provide better results and higher TPS numbers, but results depend on many factors like latency between the client and the database server, CPU capacity of the client machine.
To keep the setup simple and to document steps that don't depend on any external factors, Swingbench tests were executed on server side in this paper.

To install *Swingbench*, perform the following steps.

1. Download *Swingbench* software
   The *Swingbench* software can be downloaded from the following site: https://www.dominicgiles.com/downloads/

   At the time of writing of this technical brief, the latest *Swingbench* version was 2.7 (release 2.7.1189).

2. Unzip the downloaded file and replace the <download-directory-path>/bin/swingconfig.xml file with the swingconfig.xml filesupplied in Appendix A of this technical brief.

Configuring Oracle Database Appliance for Testing

The configuration of the Oracle Database Appliance system for the purposes of conducting this performance benchmark is described in this section.

Configuring Oracle Database Appliance System for Testing

Oracle Database Appliance X9-2-HA and X8-2 HA systems with a single storage shelf, fully populated with SSDs was used to perform the tests documented in this technical brief. All 64 CPU cores were enabled initially on both ODAs.
X8-2 HA had 384GB physical memory, X9-2 HA had 1TB. Database shape defines 128 GB SGA and 64 GB PGA, so memory size difference between the 2 models would not affect the result of the tests, hence the two models remained comparable.

Note: Regardless the number of active CPUs enabled, Oracle Database Appliance systems can always access all physical memory installed.

From a software stack perspective, the system was deployed with Oracle Database Appliance 19.17.0.0.0 software and the database version used for testing was Oracle Database 19.17.0.0.221018 (Oct 2022 DB/GI RU). Diskgroups were configured with Flex redundancy and databases were created with normal redundancy.

While there was no system-level modification performed, a few database related configuration adjustments were made, as described in a later section of this technical brief.

Benchmark Setup

The procedure for setting up the OE schema to perform the Order Entry (OE) OLTP type workload is described in this section. Similar steps can be used to build up the SH schema which is needed if DSS-type benchmark is required.

Swingbench benchmark preparation requires a deployed ODA, a database, a database schema, and the workload itself. Note that the default database parameter settings for Oracle databases on Oracle Database Appliance, when db was created via the command-line interface (odacli) or the BUI, are optimized and fits for most use-cases. Certain workloads need adjustments to init.ora parameters though. The Database Setup section below goes over these modifications that were made for tests documented in this technical brief.

Database Setup

You can create both single-instance and clustered (RAC) databases on Oracle Database Appliance. A RAC database was used for all tests documented in this paper. Database was created using the *Odb32* shape (32 CPU cores).

During database deployment, the database workload type should be specified using the --dbclass argument in ‘odacli’ command or it can be set in the BUI. If not specified,then the default workload type is ONLINE TRANSACTION PROCESSING (OLTP).

For the OLTP workload type, odb32 database shape defines SGA-PGA ratio 2:1 (SGA: 128GB, PGA:64GB).

```
# odacli create-database --dbhomeid ab80de0b-84c0-40bd-8d3d-329f99289d7f
 --dbname mycdb --dbshape odb32 --dbtype RAC --cdb --dbstorage ASM --associated-
 networks Public-network --pdbname oltpdb
```

Implement the following modifications

1. Create a dedicated SOE tablespace

```
SQL> alter session set container=oltpdb;
SQL> create bigfile tablespace soe datafile size 30g autoextend on maxsize
unlimiteduniform size 1m segment space management auto;
```

2. Increase tablespace sizes inside the PDB

   The UNDO tablespace size should be at least 30GB, while the SYSTEM, SYSAUX tablespace sizes should be atleast 10GB. Temp should be 120GB at least

3. Recreate redo logs using 32GB size for each logs and drop the ones that database initially created

```
SQL> alter database add logfile thread 1 size 32G;
```

```
SQL> alter database add logfile thread 1 size 32G;
SQL> alter database add logfile thread 1 size 32G;
SQL> alter database add logfile thread 1 size 32G;
SQL> alter database add logfile thread 2 size 32G;
SQL> alter database add logfile thread 2 size 32G;
SQL> alter database add logfile thread 2 size 32G;
SQL> alter database add logfile thread 2 size 32G;
SQL> alter system switch logfile;
SQL> alter system switch logfile;
SQL> alter system checkpoint;
SQL> alter system archive log all;
SQL> alter database drop logfile group 1;
SQL> alter database drop logfile group 2;
SQL> alter database drop logfile group 3;
SQL> alter database drop logfile group 4;
```

4. The following database configuration setting changes were made before executing the OLTP benchmark.

```
SQL> alter system set fast_start_mttr_target='900' scope=spfile sid='*';
SQL> alter system set db_recovery_file_dest_size='2T' scope=both;
SQL> alter system set resource_manager_plan = '';
```

DO NOT copy and paste the commands provided above when setting up your own benchmark environment because it may include control characters.

Note: Database archiving was enabled during all performance tests.

Prerequisites

1. Install oracle instant-client using yum install <pkg name>

   Install the following packages:
   # rpm -qa|grep instant
   oracle-instantclient19.17-sqlplus-19.17.0.0.0-1.x86_64
   oracle-instantclient19.17-devel-19.17.0.0.0-1.x86_64
   oracle-instantclient19.17-basic-19.17.0.0.0-1.x86_64

2. Download and install java
   https://www.oracle.com/uk/java/technologies/downloads/
   yum install /root/jdk-19_linux-x64_bin.rpm

3. Set GI's perl as default in the shell:

   export PATH=/u01/app/19.17.0.0/grid/perl/bin:$PATH
   # which perl
   /u01/app/19.17.0.0/grid/perl/bin/perl
   # perl -version
   This is perl 5, version 32, subversion 0 (v5.32.0) built for x86_64-linux-thread-multi

4. Downloaded xml/simple.pm

   Set http/https proxy if needed
   export http_proxy=www-example.domain.com:80
   export https_proxy=www-example.domain.com:80

   export PATH=/u01/app/19.17.0.0/grid/perl/bin:$PATH
   perl -MCPAN -e 'install XML::Simple'
   Press enter each time it asks for username/password. It will ask for it many times.

Schema Setup

The procedure for building up the OE schema to run the Order Entry OLTP workload is described in this section.

It should be highlighted that Order Entry workload generates and alters data within the SOE schema and is intended to cause database contention. If you conduct numerous workload test cycles, it is advised to rebuild the SOE database schema to prevent inconsistent results caused by the expansion and fragmentation of objects. You could also leverage on flashback database feature. Simply create a guaranteed restore point after creating the SOE schema and flashback the database to the restore point after each test cycle.

The following screenshots describe the procedure to configure SOE schema using oewizard GUI.

Log in to the ODA to start the schema setup procedure. Start a vncserver on ODA as root user and connect to the VNC terminal from your laptop or desktop to start use oewizard's GUI.

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

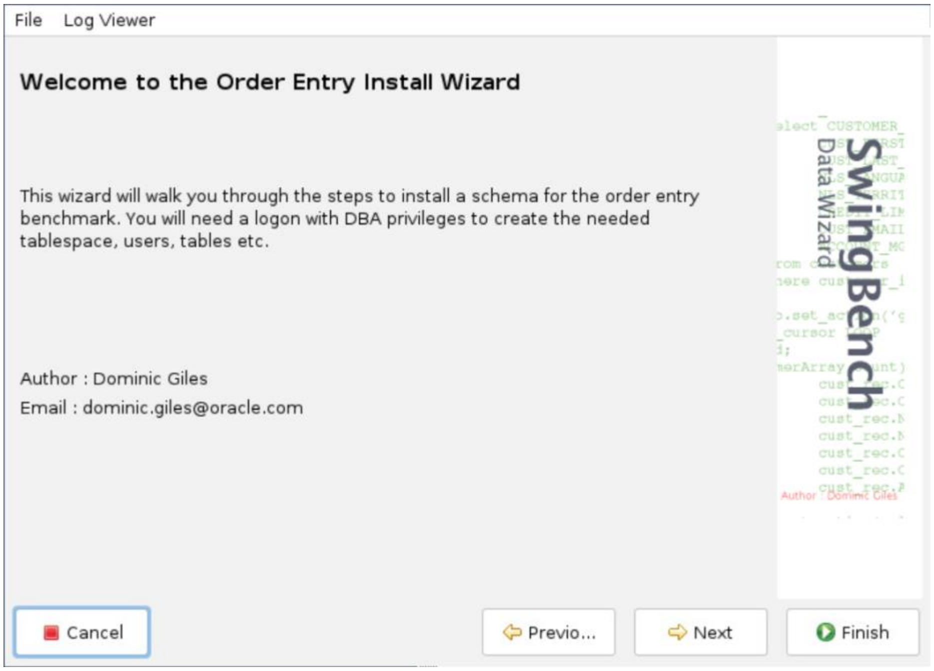


Illustration 1: Swingbench Workload Setup: Starting Order Entry Install Wizard

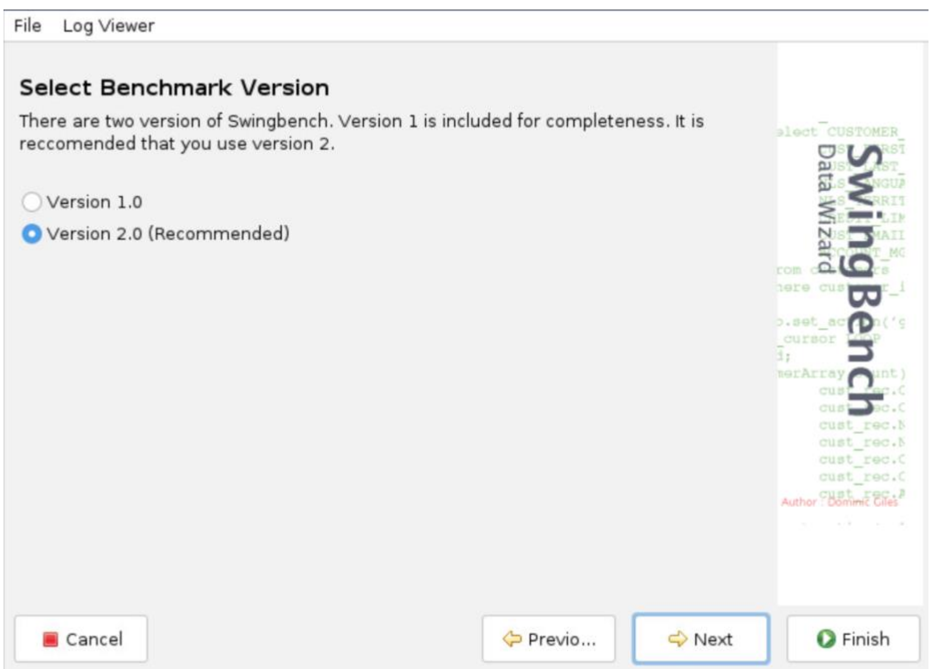


Illustration 2: Swingbench Workload Setup: Order Entry Install Wizard Benchmark Version Selection

Use the PDB’s service name in the connect string

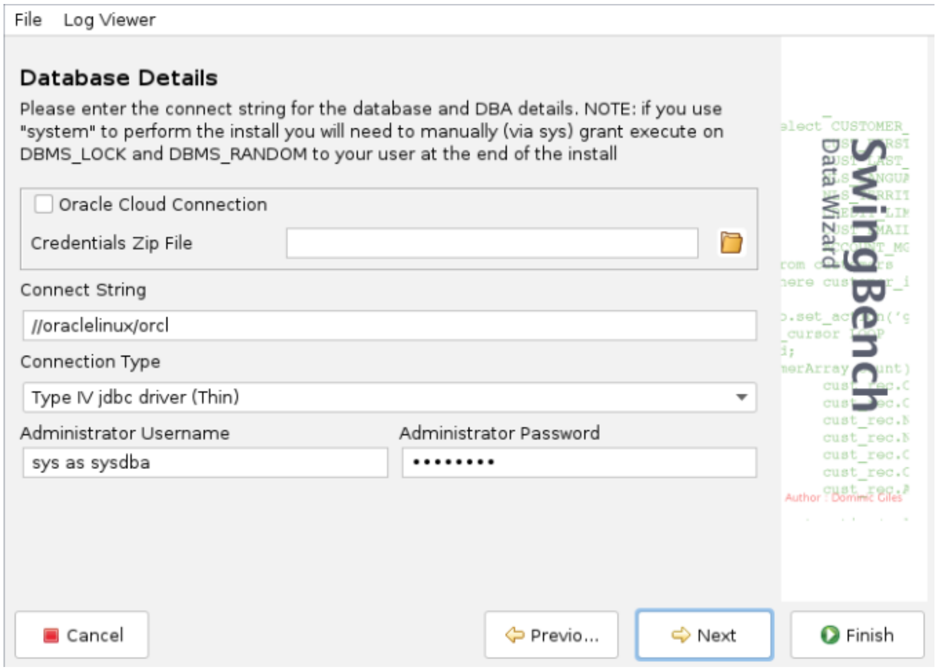


Illustration 3: Swingbench Workload Setup: Order Entry Install Wizard Database Details

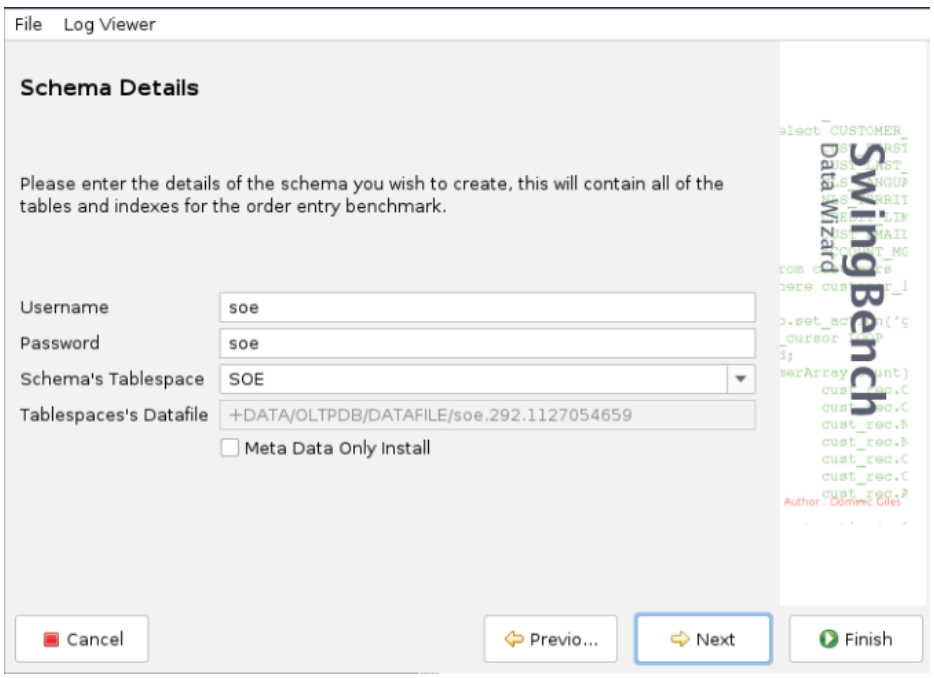


Illustration 4: Swingbench Workload Setup: Provide Schema Details in Order Entry Install Wizard

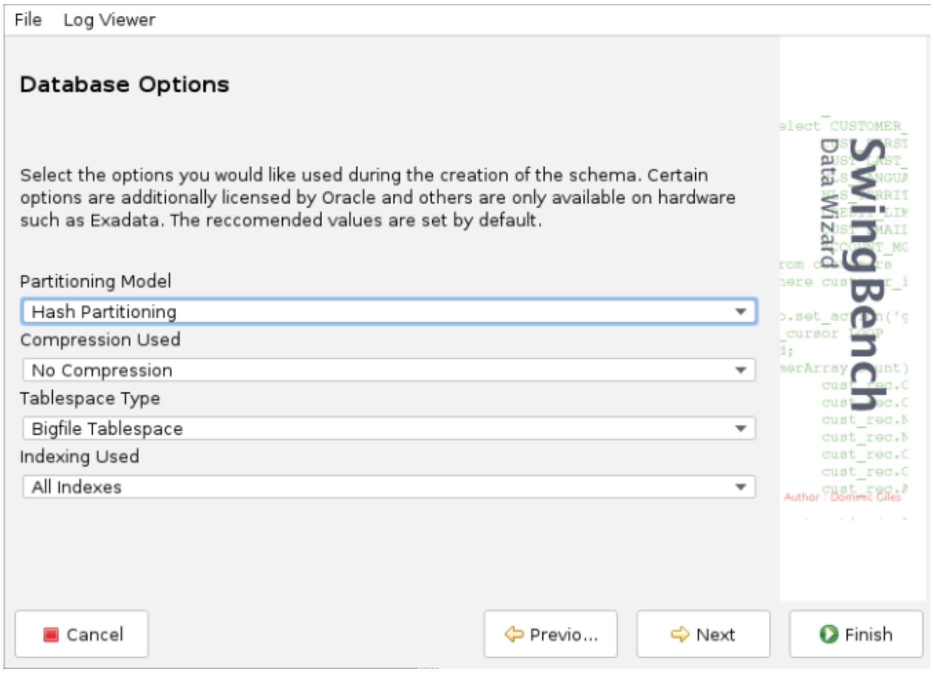


Illustration 5: Swingbench Workload Setup: Select Database Options in Order Entry Install Wizard

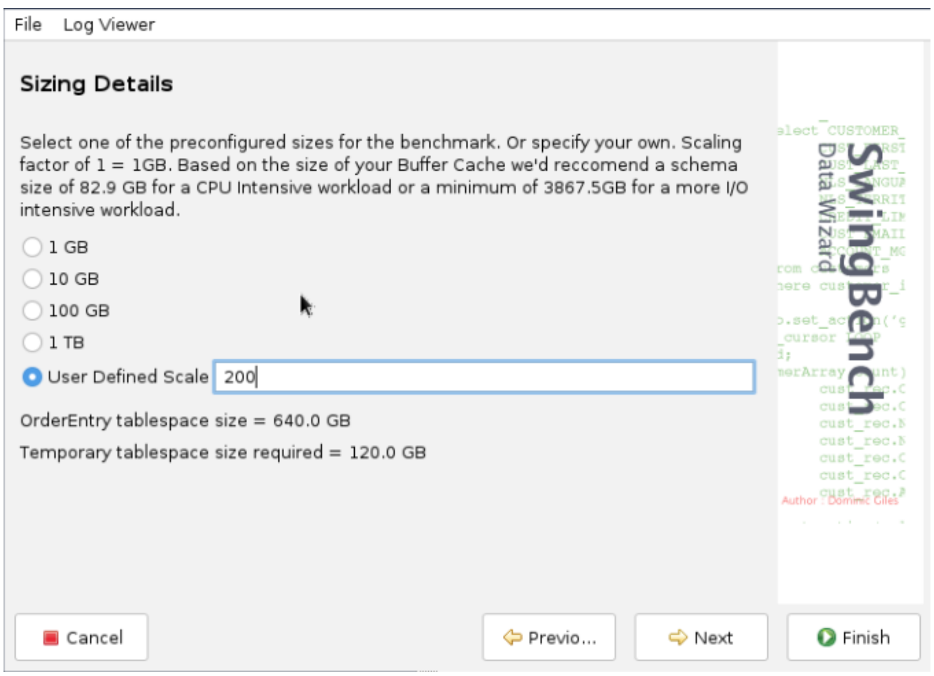


Illustration 6: Swingbench Workload Setup: Select Schema Size for Benchmark (Note: size chosen for final runs was 200GB)

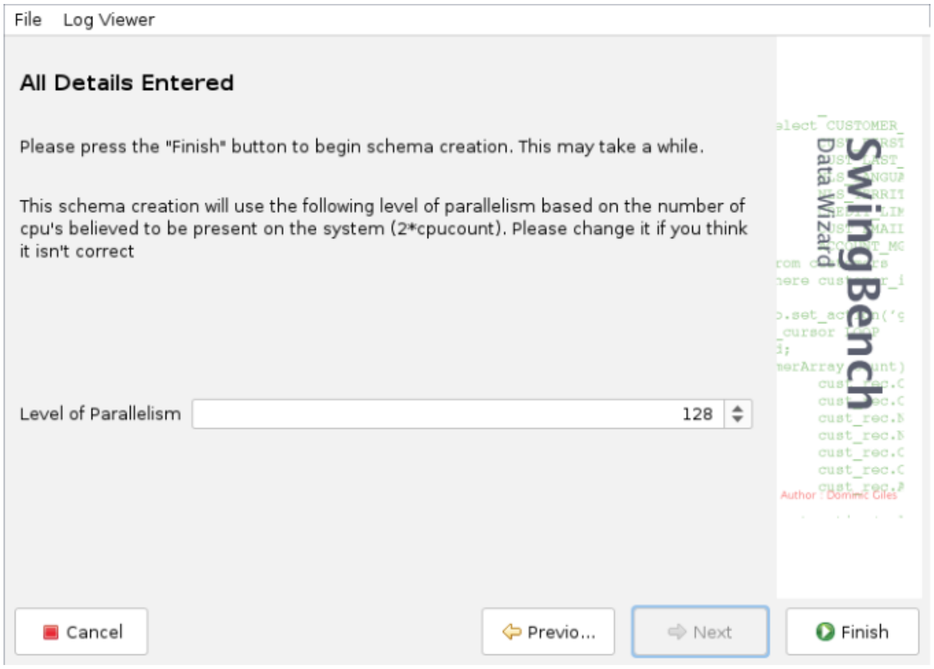


Illustration 7: Swingbench Workload Setup: Select Schema Creation Parallelism for Benchmark in Order Entry Install Wizard

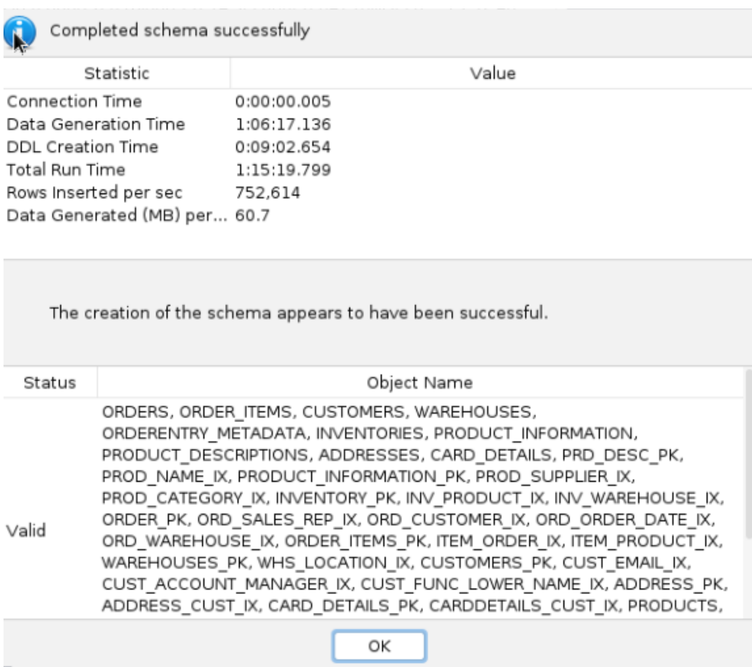


Illustration 8: Swingbench Workload Setup: Schema Created Successfully on X9-2 HA

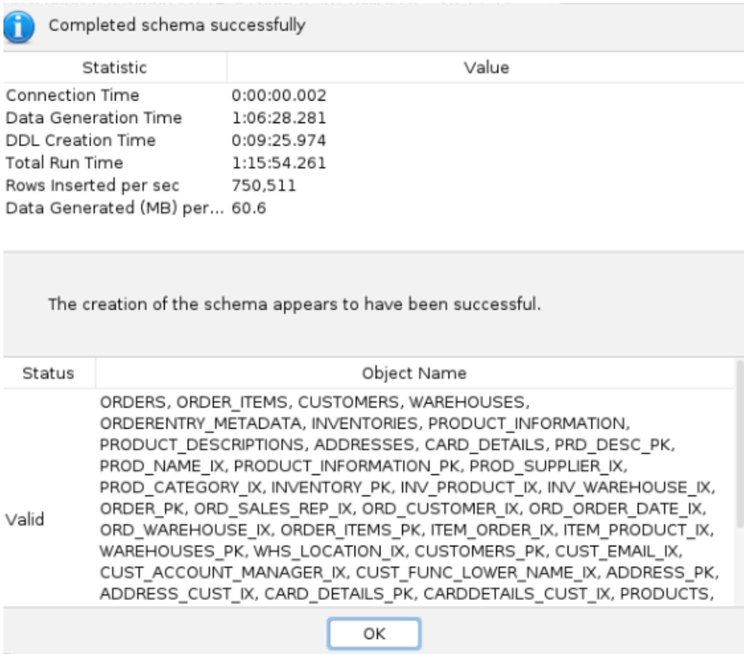


Illustration 8: Swingbench Workload Setup: Schema Created Successfully on X9-2 HA

Once the schema is ready, drop the indexes that benchmark doesn't use:

$ sqlplus / as sysdba

SQL> drop index soe.CUST_ACCOUNT_MANAGER_IX;

SQL> drop index soe.CUST_DOB_IX;

SQL> drop index soe.CUST_EMAIL_IX;

SQL> drop index soe.ITEM_PRODUCT_IX;

SQL> drop index soe.ORD_ORDER_DATE_IX;

SQL> drop index soe.ORD_SALES_REP_IX;

SQL> drop index soe.PROD_NAME_IX;

SQL> drop index soe.PROD_SUPPLIER_IX;

SQL> drop index soe.WHS_LOCATION_IX;

As mentioned earlier, test database for this benchmark was created using database shape odb32. All database init parameters like SGA, PGA were left untouched when database cores got reduced. The choice, to limit the CPU configuration only on the Oracle Database Appliance and fully utilize all other resources, was made in order to ensure that measurements obtained are fair for users.

Oracle Database Appliance systems allow for the pay-as-you-grow approach to software licensing. You have complete access to the hardware in terms of memory, storage, network regardless the number of active cores.

As part of this testing, four different CPU configurations were tested by enabling only a given total number of cores (8, 16, 32, and64) at a time on the Oracle Database Appliance system.

Workload Setup and Execution

Swingbench's Sales History benchmark is a DSS-type workload, whereas Order Entry (OE) is an OLTP type workload. The latter one was used for performance testing in this document.

You can generate the workload by connecting to the ODA and launching loadgen.pl utility

```
$ export SB_HOME=<path of Swingbench>
$ perl loadgen.pl -u <#users>
```

Swingbench provides options to set various parameters for the benchmark including setting the amount of time to run the workload. Configuration of the benchmark will be covered later in this document.

To make the workload more realistic, the workload simulates numerous concurrent users and include "think time" between transactions. The following attributes were used to replicate the workload throughout our testing.

- User Count: 150, 300, 600, 1200
- Active CPU Core Count: 8, 16, 32, 64
- Think Time: 20/30 (sleep time in milliseconds between transactions to emulate real-world workload)
- Workload Run Time: 50 minutes
- Performance Statistics Collection Window: 30 minutes (steady state)
- Transactions Per Second (TPS) Count: 3789, 8023, 16121, 25622

Benchmark Results

In this section, the results of Swingbench OLTP (OE) workload testing on Oracle Database Appliance X9-2 HA and X8-2 HA are discussed.

Workload Performance

Performance metrics gathered from running the Swingbench Order Entry (OLTP) workload on an Oracle Database Appliance system are summarized in the following tables.

Benchmark 1

X9-2 HA – Multitenant RAC database, Swingbench uses SOE schema and both instances in parallel

Active CPU Core Count: User Count	Workload Element	Total Transactions	Average Response Time (Milliseconds)	Average TPS (Transactions Per Second)
4 cores: 150 users		**4286795**		**3789**
	Customer Registration	583044	17.15	
	Process Orders	194873	12.84	
	Browse Products	3508878	7.06	
8 cores: 300 users		**9079143**		**8023**
	Customer Registration	1239086		
	Process Orders	411984		
	Browse Products	7428073		
16 cores: 600 users		**18242389**		**16121**
	Customer Registration	2491429	12.55	
	Process Orders	827381	10.81	
	Browse Products	14923579	5.77	
32 cores: 1200 users		**28986059**		**25622**
	Customer Registration	3952980	24.54	
	Process Orders	1316308	18.63	
	Browse Products	23716771	8.30	

Table 2: Swingbench OE (OLTP) Workload Execution on Oracle Database Appliance X9-2 HA

The above data illustrates the following:

1. A 25,622 transactions per second (TPS) workload delivering an average of 24.54ms transaction response for Customer Registration, 18.63ms for Process Orders and 8.30ms or Browse Products on a fully provisioned (64 CPU cores and 1TB memory) Oracle Database Appliance X9-2-HA system

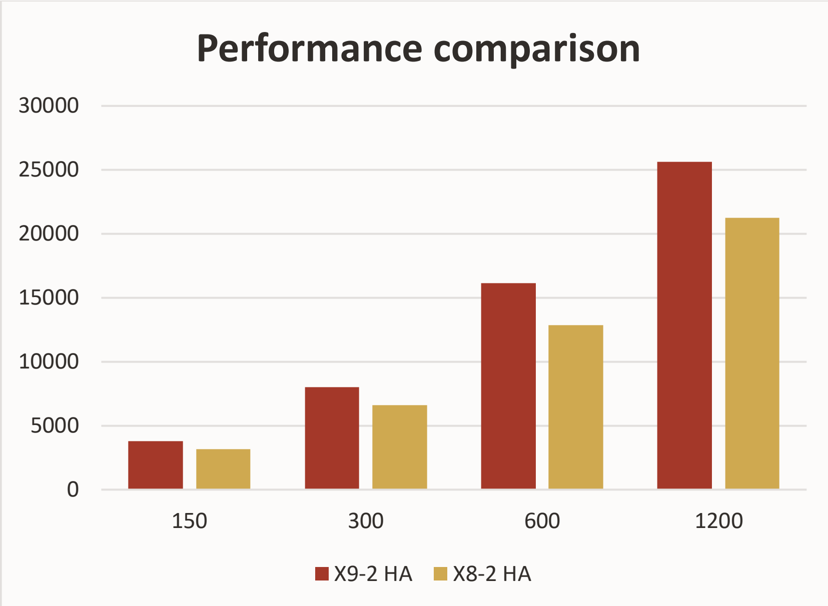
2. In fully provisioned state, during a 1,200 user workload, it delivered a sustained transaction rate of 25,622 transactions per seconds (TPS)
3. The maximum average transaction response time did not exceed 24.54ms during any of the workload test runs
4. Number of users and number of transactions scaled linearly with number of active CPU cores

Benchmark 1 - comparison with X8-2 HA

Multitenant RAC database, Swingbench uses SOE schema and uses both instances in parallel.

Active CPU Core Count: User Count	**Workload Element**	**Total Transactions**	**Average Response Time (Milliseconds)**	**Average TPS (Transactions Per Second)**
4 cores: 150 users		**3577018**		**3163**
	Customer Registration	487804	23.28	
	Process Orders	161718	17.49	
	Browse Products	2927496	10.93	
8 cores: 300 users		**7463868**		**6596**
	Customer Registration	1018016	24.36	
	Process Orders	339748	17.16	
	Browse Products	6106104	8.63	
16 cores: 600 users		**14568072**		**12876**
	Customer Registration	1985875	20.52	
	Process Orders	663089	17.54	
	Browse Products	11919108	9.31	
32 cores: 1200 users		**24061612**		**21266**
	Customer Registration	3281916	35.44	
	Process Orders	1093228	24.56	
	Browse Products	19686468	11.52	

Table 3: Swingbench OE (OLTP) Workload Execution on Oracle Database Appliance X8-2 HA



Graph 1: Swingbench OE (OLTP) Workload based comparison between X9-2 HA and X8-2 HA

Comparison between X9-2 HA and X8-2 HA using RAC db shows ~20% performance difference on average between the two models.

Database and Operating System Statistics

In this section, database and OS statistics related observations are described based on the test executed using Swingbench.

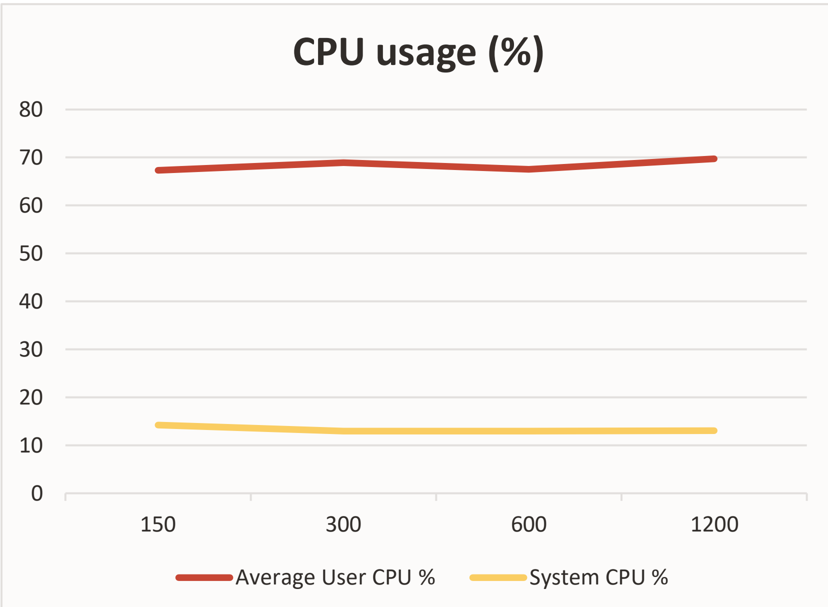
On Oracle Database Appliance X9-2-HA and X8-2 HA machines, 64 CPU cores are available (32 CPU cores on each host). The system's active CPU core count is dynamically expandable from 8 to 64. As shown in table 2, a total of four configurations were examined during the benchmark, and the total number of active CPU cores were steadily raised between 8 and 64.

A few of the major findings about database and operating system statistics gathered during the OLTP benchmark:

1) Connections were evenly (but not exactly) split across the two servers during the workload runs, and the average user CPU usage on each DB host never went above 72%
2) With the configured OLTP workload, transaction rates increased linearly with user volumes as expected
3) Volume of redo read and write operations grew along with the volume of transactions

Average User CPU Busy %

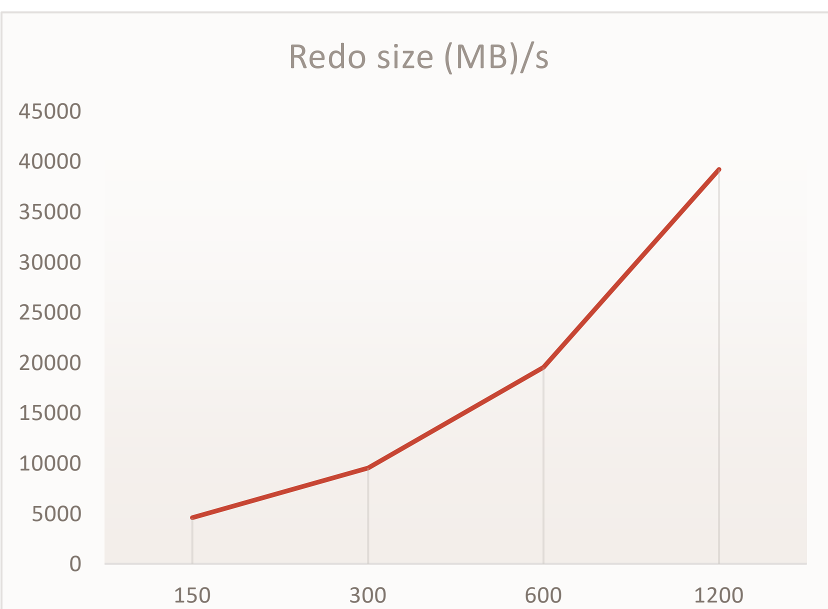
Each test cycle's average User CPU usage across the two hosts of the Oracle Database Appliance was recorded. A narrow range, between around 66% and 72%, was recorded for the overall User CPU busy%, which fluctuated.
On the following graph Average User CPU % is the average of the data from the ODA node where only the database was running. System CPU was fluctuating between 13 and 16 percent during the tests.



Graph 2: Average User CPU Busy

REDO Writes

REDO write rate (MB/Sec) was measured for each test cycle on each node. The graph below illustrates the total REDO volume write-rate across both the nodes of the Oracle Database Appliance system. REDO write-rate increased as number of transactions per second increased in a fairly linear manner.
Note that the REDO write volume is a cumulative metric for the two database nodes.

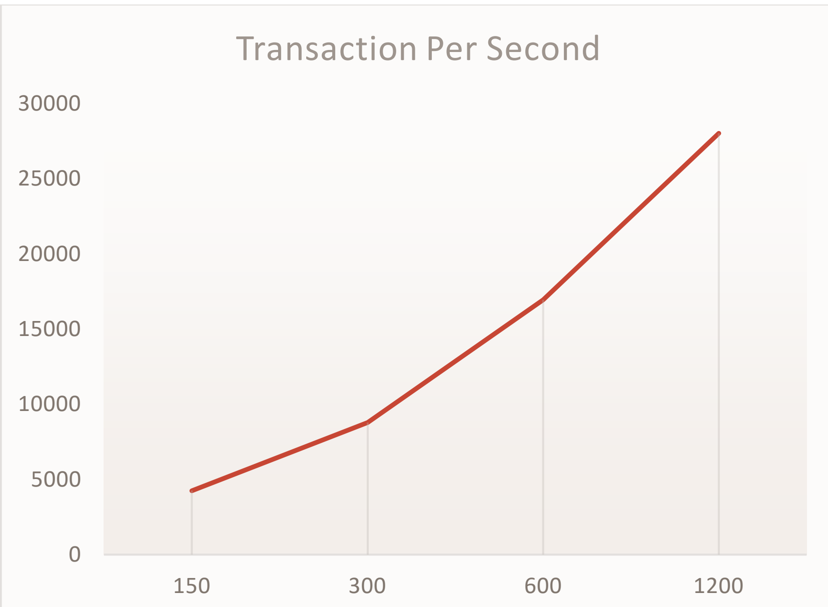


Graph 3: Average REDO Write volume (MB/s)

Transaction Rate

As transaction volume climbed from around 3789 TPS to approximately 25622 TPS and the number of active CPU cores increased from 8 to 64 during the test, the transaction rate (average transactions per second) scaled virtually linearly.

Keep in mind that the estimation of the transaction volume is based on data from both database nodes.



Graph 4: Average Transaction Volume (Transactions Per Second)

It should be noted that in graph 3, the active CPU core count and transaction volume were modified.

Benchmark 2

Comparing X8-2 HA and X9-2 HA when Swingbench is running on both ODA nodes, but each Swingbench benchmark runs against a dedicated PDB which is only available on the local node. Each PDB has its own SOE schema. Refer to Appendix D.

X9-2 HA

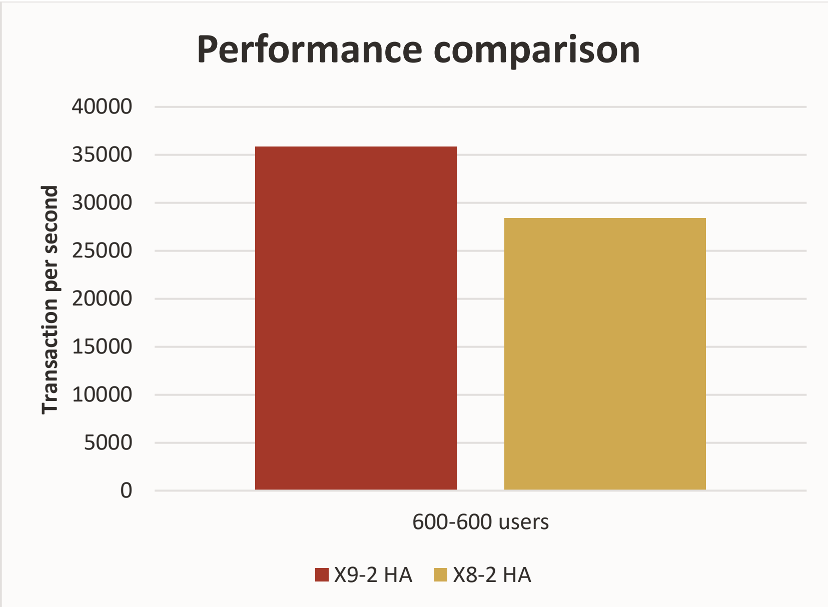
Active CPU Core Count: User Count	Workload Element	Total Transactions	Average Response Time (Milliseconds)	Average TPS (Transactions Per Second)
32 cores: 600 - 600 users		**38381742**		**35830**
	Instance 1	**18863852**		**18577**
	Customer Registration	955490	9.83	
	Process Orders	709504	6.96	
	Browse Products	17198858	2.79	
	Instance 2	**19517890**		**17253**
	Customer Registration	2659799	13.88	
	Process Orders	885764	9.22	
	Browse Products	15972327	3.57	

Table 4: Swingbench OE (OLTP) Workload based comparison – RAC multitenant DB, 2 PDBs, each PDB is available on its dedicated node

X8-2 HA

Active CPU Core Count: User Count	Workload Element	Total Transactions	Average Response Time (Milliseconds)	Average TPS (Transactions Per Second)
32 cores: 600 - 600 users		**32169164**		**28425**
	Instance 1	**15961068**		**14100**
	Customer Registration	2174844	21.16	
	Process Orders	725130	15.09	
	Browse Products	13061094	6.84	
	Instance 2	**16208096**		**14325**
	Customer Registration	2211138	20.25	
	Process Orders	735999	14.05	
	Browse Products	13260959	6.66	

Table 5: Swingbench OE (OLTP) Workload based comparison - RAC multitenant DB, 2 PDBs, each PDB is available on its dedicated node



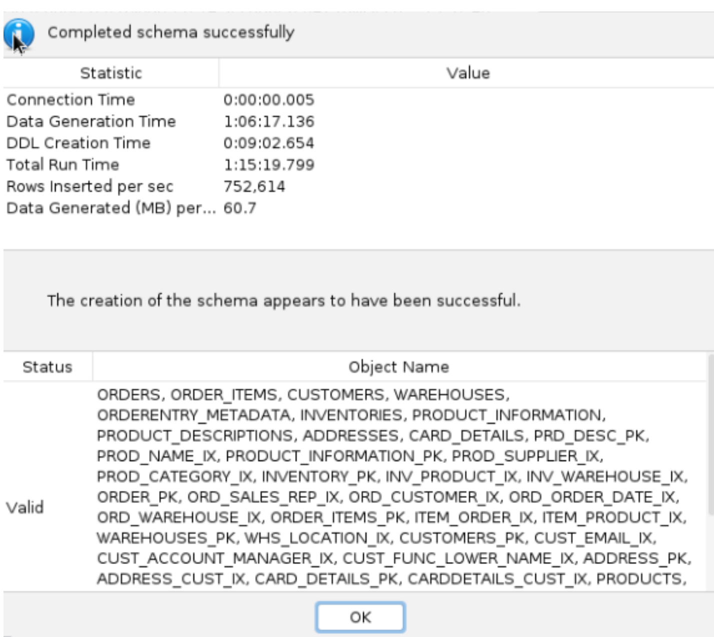
Graph 5: Swingbench OE (OLTP) Workload based comparison - RAC multitenant DB, 2 PDBs, each PDB is available on its dedicated node

Comparison between X9-2 HA and X8-2 HA using RAC db, configured with 2 PDBs, having one SOE schema in each and mapped to 1-1 database instance also shows more than 20% performance difference between the two models.

Benchmark 3

Data generation using oewizard.

<u>X9-2 HA</u>



Completed schema successfully

Statistic	Value
Connection Time	0:00:00.005
Data Generation Time	1:06:17.136
DDL Creation Time	0:09:02.654
Total Run Time	1:15:19.799
Rows Inserted per sec	752,614
Data Generated (MB) per...	60.7

The creation of the schema appears to have been successful.

Status	Object Name
Valid	ORDERS, ORDER_ITEMS, CUSTOMERS, WAREHOUSES, ORDERENTRY_METADATA, INVENTORIES, PRODUCT_INFORMATION, PRODUCT_DESCRIPTIONS, ADDRESSES, CARD_DETAILS, PRD_DESC_PK, PROD_NAME_IX, PRODUCT_INFORMATION_PK, PROD_SUPPLIER_IX, PROD_CATEGORY_IX, INVENTORY_PK, INV_PRODUCT_IX, INV_WAREHOUSE_IX, ORDER_PK, ORD_SALES_REP_IX, ORD_CUSTOMER_IX, ORD_ORDER_DATE_IX, ORD_WAREHOUSE_IX, ORDER_ITEMS_PK, ITEM_ORDER_IX, ITEM_PRODUCT_IX, WAREHOUSES_PK, WHS_LOCATION_IX, CUSTOMERS_PK, CUST_EMAIL_IX, CUST_ACCOUNT_MANAGER_IX, CUST_FUNC_LOWER_NAME_IX, ADDRESS_PK, ADDRESS_CUST_IX, CARD_DETAILS_PK, CARDDETAILS_CUST_IX, PRODUCTS,

OK

Table 6: Swingbench OE wizard screenshot, completion time – X9-2 HA

X8-2 HA

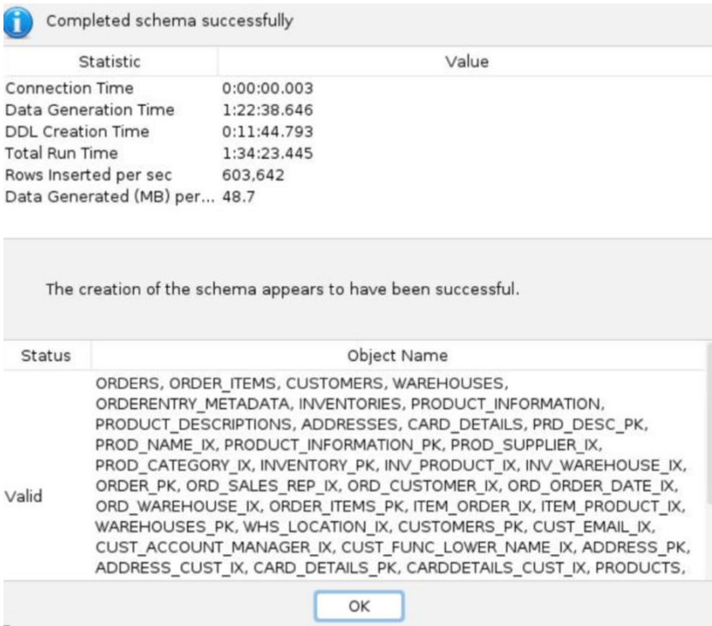


Table 7: Swingbench OE wizard screenshot, completion time – X8-2 HA

X9-2 HA completed the data generation 19 mins earlier than X8-2 HA which means X9-2 HA was 25% faster than X8-2 HA.

Important Considerations for Performing the Benchmark

High performance storage, networking, CPU, and memory components are used in the construction of Oracle Database Appliance systems. However, there are a few things to keep in mind in order to get the best performance out of your Oracle Database Appliance installation.

In order to maintain your Oracle Database Appliance environments at peak performance level, regardless you're doing a benchmark test or not, follow the general instructions in this section.

1. Ensure that databases on Oracle Database Appliance are always created using the Browser User Interface (BUI) or *odacli* command-line interface as both of them use pre-built templates thatprovide pre-optimized database parameter settings for required DB shapes and sizes.

2. When performing benchmarks for comparison in two different environments ensure that identical workload is run for apples-to-apples comparison. If you run different workloads (different SQL, different commit rates, or even if you only have different execution plans, etc.) in the legacy system and in the Oracle Database Appliance environment, then platform performance comparisonsmay be misleading, inaccurate, hence pointless.

3. Keep network latency low. For example, running Swingbench client(s) on the same network (but on a separate host) asyour Oracle Database Appliance is on, might help to prevent significant latency in the transaction path.

4. Size the Oracle Database Appliance environment appropriately and adequately. When conducting benchmarks, it isimperative that the two environments being compared are sized similarly.

5. Check SQL 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. For example, the data volumes in the two environments may bedifferent, there may be index differences, or lack of proper

optimizer statistics, etc. which may contribute to differences in SQLexecution plans and execution timings.

6. Whenever it is possible, perform comparisons and benchmarks between systems that run the same software stack (OS version, GI and RDBMS release, etc.) and have similar resource allocations. Hardware differences are naturally expected.
7. Do not use performance inhibiting database parameters. If migrating databases from legacy environments to Oracle DatabaseAppliance, make sure you do not carry over obsolete, un-optimized settings and parameters. Do not modify database parameters blindly to match the database parameters from your legacy environment. You may use “orachk” tool to verifyyour database configuration running on Oracle Database Appliance and in legacy environments.
8. Oracle Database Appliance provides features such as database block checking and verification to protect against data corruption out of the box. These features may consume some, albeit small, amount of CPU capacity, but they are generally desirable to protectthe integrity of your data. While these features might be temporarily disabled for testing purposes, it is strongly recommended to use these protective features to mitigate data corruption risks.

Conclusion

According to the performance benchmark used to create this technical brief, Oracle Database Appliance provides good performance for typical database workloads. An Oracle Database Appliance X9-2-HA system was easily able to manage a Swingbench OLTP workload of 28,316 transactions per second (TPS) with 32 CPU cores enabled. In addition to that, as workload and CPU resources were increased simultaneously, performance scaled essentially linearly.

Appendix A - Swingbench configuration files

This section described the changes that have been done in Swingbench configuration file for the benchmarks covered in the document.

The configuration file is: SB_HOME/configs/SOE_Server_Side_V2.xml
Changes are highlighted in red.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SwingBenchConfiguration xmlns="http://www.dominicgiles.com/swingbench/config">
    <Name>"Order Entry (PLSQL) V2"</Name>
    <Comment>Version 2 of the SOE Benchmark running in the database using
PL/SQL</Comment>
    <Connection>
        <UserName>soe</UserName>
        <Password>soe</Password>
        <ConnectString>//myoda-scan/oltpdb.domain.com</ConnectString>
        <DriverType>Oracle jdbc Driver</DriverType>
        <Properties>
            <Property Key="StatementCaching">60<s/Property>
            <Property Key="FetchSize">20</Property>
        </Properties>
    </Connection>
    <Load>
        <NumberOfUsers>16</NumberOfUsers>
        <MinDelay>0</MinDelay>
        <MaxDelay>0</MaxDelay>
        <InterMinDelay>20</InterMinDelay>
        <InterMaxDelay>30</InterMaxDelay>
        <QueryTimeout>120</QueryTimeout>
        <MaxTransactions>-1</MaxTransactions>
        <RunTime>0:50</RunTime>
        <LogonGroupCount>1</LogonGroupCount>
        <LogonDelay>0</LogonDelay>
        <LogOutPostTransaction>false</LogOutPostTransaction>
        <WaitTillAllLogon>true</WaitTillAllLogon>
        <StatsCollectionStart>0:15</StatsCollectionStart>
        <StatsCollectionEnd>0:45</StatsCollectionEnd>
        <ConnectionRefresh>0</ConnectionRefresh>

…
Rest of the file remains untouched.
```

Memory size in SB_HOME/launcher/launcher.xml needs to be bumped up

```
<jvmargset id="base.jvm.args">
     <jvmarg line="-Xmx2048m"/>
     <jvmarg line="-Xms512m"/>
     <!--<jvmarg line="-Djava.util.logging.config.file=log.properties"/>-->
</jvmargset>
```

…

Rest of the file remains untouched.

Appendix B - loadgen.pl

Note that you may need to update the sample password, SCAN name in the script below.

```
#!/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="myoda-scan.domain.com";
my $cdb_service="mycdb.domain.com";
my $port=1521;
my $dbauser="system";
my $dbapwd="welcome1";
my $config_file_1="SOE_Server_Side_V2.xml";
### Please modify the above variables as needed #######
my $rundate=strftime("%Y%m%d%H%M", localtime);
my $datevar=strftime("%Y_%m_%d", localtime);
my $timevar=strftime("%H_%M_%S", localtime);
my @app_modules = ("Customer Registration","Process Orders","Browse Products","Order
Products");
my $cdb_snap_id;
my $pdb_snap_id;
my $dbid;
my $cdb_b_snap;
my $cdb_e_snap;
my %opts;
my $tot_uc;
my $cb_sess;
my $counter;
my $uc=100;
my $max_cb_users=100;
```

```
my $min_cb_instances=10;
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/".$datevar."/".$timevar;
print "SB_OUTPUT_DIR : $sb_output_dir"."\n";
my $awr_dir=$sb_output_dir;
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";
print ("$sb_home/bin/charbench -uc $uc -c $sb_home/configs/$config_file_1 -r
$sb_output_dir/results_"."$uc"."_users_"."$rundate"."$l_counter"."_RAC_"."".xml -s");
system ("$sb_home/bin/charbench -uc $uc -c $sb_home/configs/$config_file_1 -r
$sb_output_dir/results_"."$uc"."_users_"."$rundate"."$l_counter"."_RAC_"."".xml -s");
}
sub create_out_dir {
if ( -d "$_[0]" ) {
print "Direcory "."$_[0]"." Exists\n";
}
else{
system("mkdir -p $_[0]");
}
}


sub generate_awr_snap
{
print "Generating Snapshot at DB level...\n";
my $dbh = DBI->connect("dbi:Oracle://$host:$port/$cdb_service","$dbauser","$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,\$cdb_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;
my $outfile = 'result.txt';
@files = <$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";
}
open(my $OUTFILE, ">>$sb_output_dir/$outfile") || die "problem opening $file\n";
print $OUTFILE "Total Number of Application Users : ".$num_users."\n";
print $OUTFILE "Average Transactions Per Second : ".$avg_tps."\n";
print $OUTFILE "------------------------------------------------------\n";
printf $OUTFILE "%-22s %16s %8s\n","Application Module","Txn Count","Avg Res Time";
```

```
print $OUTFILE "------------------------------------------------------\n";
foreach $app_module (@app_modules)
{
if ($app_module eq "Customer Registration") {
printf $OUTFILE "%-22s %16s %0.2f\n",$app_module,$cr_tc,($cr_to_rt/$cr_tc);
}
elsif ($app_module eq "Process Orders") {
printf $OUTFILE "%-22s %16s %0.2f\n",$app_module,$po_tc,($po_to_rt/$po_tc);
}
elsif ($app_module eq "Browse Products") {
printf $OUTFILE "%-22s %16s %0.2f\n",$app_module,$bp_tc,($bp_to_rt/$bp_tc);
}
elsif ($app_module eq "Order Products") {
printf $OUTFILE "%-22s %16s %0.2f\n",$app_module,$op_tc,($op_to_rt/$op_tc);
}
}
close($OUTFILE);
}
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);
$awr_dir=$sb_output_dir;
chk_n_set_env;
set_cb_parameters;
my $rc;
my $sleep_time;
$sleep_time=300/$cb_sess;
print "Sleeping for 30 seconds"."\n";
sleep 30;

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 for $config_file_1 on
inst1"."\n";
$thr->detach();
print "Sleeping for $sleep_time seconds"."\n";
sleep $sleep_time;
}
print "Sleeping for 600 seconds"."\n";
```

```
sleep 600;
generate_awr_snap;
$cdb_b_snap=$cdb_snap_id;
print "Start Snap $cdb_b_snap"."\n";
print "Sleeping for $awr_interval_in_secs seconds"."\n";
sleep $awr_interval_in_secs;
generate_awr_snap;
$cdb_e_snap=$cdb_snap_id;
print "End Snap $cdb_e_snap"."\n";
my $running;
while (1) {
# Checking if any charbench session is running
$running = `ps -ef |grep $rundate| grep -v grep |wc -l`;
if ($running == 0)
{
process_xml_output;
print " Exiting Loop.. \n";
last;
}
sleep 10;
}
#print "DB Id $dbid"."\n";
print "Generating AWR Reports....\n";
system ("$pwd/generate_awr.sh", $cdb_b_snap, $cdb_e_snap, $rundate, $awr_dir);
print " Result ... \n";
system ("cat $sb_output_dir/result.txt");
print " Exiting .. \n";
exit 0;
```

Appendix C - generate_awr.sh script

Note that you may need to update the sample password, dbid of the CDB, SCAN name used in the script below.

```
#!/bin/bash
unset http_proxy
unset https_proxy
export host=myoda-scan.domain.com
l_dbid=2704614255
inst1="mycdb1"
inst2="mycdb2"
export svc="mycdb.domain.com"
export ORACLE_HOME=/u01/app/19.17.0.0/grid
export port=1521
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 ${AWR_DIR}
echo "system/WElcome_12##@$host:$port/$svc1"
$ORACLE_HOME/bin/sqlplus -s system/welcome1@$host:$port/$svc/$inst1 << EOC
set head off
set pages 0
set lines 132
set echo off
set feedback off
spool "awrrpt_1_${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_snapid ));
spool off
exit;
```

```
EOC
$ORACLE_HOME/bin/sqlplus -s system/WElcome_12##@$host:$port/$svc/$inst2 << EOC
set head off
set pages 0
set lines 132
set echo off
set feedback off
spool "awrrpt_2_${l_start_snapid}_${l_end_snapid}_${l_runid}.log"
SELECT
output
FROM
TABLE
(dbms_workload_repository.awr_report_text($l_dbid,2,$l_start_snapid,$l_end_snapid ));
spool off
exit;
EOC
```

Appendix D – Swingbench test using 2 SOE schemas

1. Enable all CPU cores via odacli update-cpucore -c 32
2. Create a 2nd PDB database

```
SQL> CREATE PLUGGABLE DATABASE oltpdb2 ADMIN USER pdbadmin IDENTIFIED BY welcome1;
SQL> ALTER PLUGGABLE DATABASE oltpdb2 OPEN READ WRITE instances=all;
```

3. Increate the size of System, Sysaux tablespaces to 10GB, undo tablespaces to 30GB

```
SQL> alter session set container=oltpdb2;
SQL> select file_id,file_name, bytes/1024/1024 from dba_data_files;
SQL> alter database datafile <file number> resize <size>G;
```

4. Create a new tablespace inside the PDB

```
SQL> create bigfile tablespace soe datafile size 300g autoextend on maxsize
unlimited uniform size 1m segment space management auto;
```

5. Copy Swingbench to the second node of the ODA
6. Create a SOE schema with oewizard in the new PDB
7. Shutdown the 2nd instance of each PDB. Keep oltpdb's first instance up on node1 and oldtpdb2's first instance on node2. Execute the commands at CDB level

```
SQL> alter pluggable database oltpdb close instances=('mycdb2');
SQL> alter pluggable database oltpdb2 close instances=('mycdb1');
```

8. In loadgen.pl change the following parameter on node2

```
from
my $cdb_service="oltpdb.domain.com";
to
my $cdb_service="oltpdb2.domain.com";
```

9. In SB_HOME/configs/SOE_Client_Side.xml change

```
On node1:
       <UserName>soe</UserName>
        <Password>soe</Password>
        <ConnectString>//myoda-scan/oltpdb.domain.com/mycdb1</ConnectString>
On node2:
       <UserName>soe2</UserName>
        <Password>soe2</Password>
        <ConnectString>//myoda-scan/oltpdb2.domain.com/mycdb2</ConnectString>
```

10. Run loadgen.pl on both machines

```
perl loadgen.pl -u 600
```

References

Oracle Database Appliance X9-2-HA Data Sheet

https://www.oracle.com/a/ocom/docs/engineered-systems/database-appliance/oda-x9-2-ha-datasheet.pdf

Oracle Database Appliance X9-2-HA

https://www.oracle.com/engineered-systems/database-appliance/#rc30p9

Oracle Database Appliance Documentation

https://docs.oracle.com/en/engineered-systems/oracle-database-appliance

Swingbench

https://www.dominicgiles.com/index.html

Oracle Corporation, World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000

Fax: +1.650.506.7200

Integrated Cloud Applications & Platform Services

Evaluating and Comparing Oracle Database Appliance Performance Date: March 2023

Author: RAC Pack, Cloud innovation & Solution Engineering Team

"Customer quotes are powerful proof points and may be displayed prominently in this sidebar area. The most effective quotes include measurable business results. You must get all customer quotes approved by the Oracle Customer Reference Program—for every use."

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

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