

Evaluating and Comparing Oracle Database Appliance Performance

Updated for Oracle Database Appliance X8-2-HA

ORACLE WHITE PAPER | JANUARY 2020





DISCLAIMER

The performance results in this paper are intended to give an estimate of the overall performance of the Oracle Database Appliance X8-2-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	1
Audience	2
Objective	2
Oracle Database Appliance Deployment Architecture	3
Oracle Database Appliance Configuration Templates	3
What is <i>Swingbench</i> ?	4
Swingbench Download and Setup	4
Configuring Oracle Database Appliance for Testing	5
Configuring Oracle Database Appliance System for Testing	5
Benchmark Setup	5
Database Setup	6
Schema Setup	6
Workload Setup and Execution	10
Benchmark Results	11
Workload Performance	11
Database and Operating System Statistics	12
Average CPU Busy	12
REDO Writes	13
Transaction Rate	13
Important Considerations for Performing the Benchmark	14
Conclusion	15
Appendix A Swingbench configuration files	16
Appendix B The loadgen.pl file	19
Appendix C The generate_awr.sh script	24
References	25



Introduction and Executive Summary

Oracle Database Appliance X8-2-HA is a highly available Oracle database system. It is an integrated, pre-built, pre-tuned, packaged database solution that contains hardware, software, networking, and storage, all in a small footprint, 8RU configuration. The hardware and software configuration of Oracle Database Appliance X8-2-HA provides redundancy and protects against all single points of failures within the system.

Specifically, Oracle Database Appliance X8-2-HA is a two node cluster based on the latest Intel Xeon processors with direct attached SAS storage that consists of Solid State Disk Drives (SSDs) or a combination of Hard Disk Drives (HDDs) and SSDs, depending on customers' choice. It runs standard, time-tested software components, such as Oracle Linux operating system (OS), Oracle Relational Database Management System (RDBMS), Oracle Real Application Clusters software (RAC), Oracle Clusterware, and Oracle Automatic Storage Management (ASM). The pre-built, pre-tested, and pre-tuned configuration ensures that Oracle Database Appliance can be rapidly deployed, often on the day when the system arrives at a customer site, and does not require typical manual efforts to configure and tune the configuration. Further, Oracle Database Appliance includes Oracle Appliance Manager Software (GUI and CLI) for managing and maintaining the system, including patching, upgrades, and troubleshooting.

When deploying a new platform, users typically want to understand the performance characteristics of the platform. The purpose of this white paper is to demonstrate and document the performance of a simulated workload (using *Swingbench*, a freely available performance testing tool) executing on an Oracle Database Appliance X8-2-HA system. System Architects and Database Administrators can evaluate and compare the performance of such standardized workload executing on Oracle Database Appliance vis-à-vis the same workload executing in their legacy environment. It is *not* a purpose of this white paper to demonstrate the maximum IOPS and MBPS capacity of Oracle Database Appliance although some related information is provided.

Despite its small 8RU size, Oracle Database Appliance X8-2-HA is a very powerful, highly available database system. During the performance testing conducted as part of running the performance benchmark, Oracle Database Appliance demonstrated scalable performance for high volume database workloads. In a configuration with all 32 CPU cores per node activated, Oracle Database Appliance X8-2-HA supported more than 37,500 concurrent *Swingbench* transactions per second while maintaining an average response time of less than 12.1ms, and with plenty of capacity to spare.



Audience

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

Objective

A quick review of the hardware configuration of Oracle Database Appliance will show that the architecture of the system is built for high availability and good out-of-the-box performance. However, due to the presence of many components in any given system and contrasting nature of different workloads, customers often rightfully request baseline comparative performance data for various types of standard workloads. This helps them project their own performance experience and expectations once they migrate their database(s) to the new environment. The objectives of this white paper address this need.

The primary objective of this white paper is to quantify performance of Oracle Database Appliance when running, what may be considered, a typical database workload. This workload performance information is presented in easy to understand terms such as number of users, number of transactions per minute, and transaction execution time. The system performance is presented in terms of resource utilization, and data processing rates.

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 outline a process for executing test workload in non-Oracle Database Appliance (legacy) environments and comparing the results against those captured and presented in this white paper. This objective is facilitated by documenting the process of *Swingbench* setup and test results from multiple *Swingbench* Order Entry workload runs on an Oracle Database Appliance system.

This paper is the result of extensive testing carried out with *Swingbench* workload while varying the user and transaction volumes in different CPU configurations of an Oracle Database Appliance X8-2-HA system running Oracle Appliance Manager software bundle version 18.7.0.0.0. The OLTP workload (provided by the OE schema) was tested. Customers can run these same *Swingbench* workloads on their legacy systems and compare the results with the corresponding results documented in this white paper.

Oracle Database Appliance allows two types of implementations, Bare Metal and Virtualized Platform. The testing during the course of writing of this white paper was conducted on Oracle Database Appliance Bare Metal configuration.

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



Oracle Database Appliance Deployment Architecture

For system details refer to Oracle Database Appliance X8-2-HA Data Sheet available at <https://www.oracle.com/technetwork/database/database-appliance/oda-x8-2-ha-datasheet-5730739.pdf>

Oracle Database Appliance Configuration Templates

Oracle Database Appliance provides and uses several standard database configuration templates for creating databases of different shapes and sizes. Databases created using these templates automatically inherit Oracle database implementation best practices such as most optimal database parameter settings as well as best practices based configuration of storage.

Database shapes are used to configure databases based on the type of specific workload that you wish to run on your Oracle Database Appliance. Multiple database shapes are available for different workload types, viz., OLTP, DSS, and In-Memory. For a given workload type, you can choose an appropriate database shape based on your workload.

Once a database is deployed using a given template, users are not restricted from altering the database parameters subsequently.

Refer to Oracle Database Appliance X8-2-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/sizes on Oracle Database Appliance.

In order to perform a meaningful performance comparison, the configuration of the database running on Oracle Database Appliance and that running in the non-Oracle Database Appliance environment should match closely.

The integrated, high-performance, all SSD storage in Oracle Database Appliance X8-2-HA provides substantial IO capacity for users to deploy heavy OLTP, DSS, Mixed, and In-memory workloads. It may be noted that in a separate set of tests, a fully populated, dual storage shelf configuration, Oracle Database Appliance provided up to 2,161,600 IOPS (IO Operations Per Second) and throughput of up to 21,800 MBPS (Mega Bytes Per Second) whereas a single storage shelf provided up to 1,640,200 IOPS and could support up to 15,300 MBPS. These IOPS / MBPS tests were conducted using 8K random reads and 1M random reads respectively. Some variations were observed depending on the workload mix (READ/WRITE ratios).

It may be noted that in the Oracle Database Appliance X8-2-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.

Based on this data, Table 1 provides a mathematically computed, IO capacity that may be available to different shapes deployed on Oracle Database Appliance X8-2-HA system. Note that available IO capacity is not restricted depending on the shapes you deploy.

Shape →	odb1s	odb1	odb2	odb4	odb8	odb12	odb16	odb24	odb28	odb32
Measure ↓										
Single storage shelf configuration										
IOPS	51,256	51,256	102,513	205,025	410,050	615,075	820,100	1,230,150	1,435,175	1,640,200
MBPS	478	478	956	1,913	3,825	5,738	7,650	11,475	13,388	15,300
Dual storage shelf configuration										
IOPS	67,550	67,550	135,100	270,200	540,400	810,600	1,080,800	1,621,200	1,891,400	2,161,600
MBPS	681	681	1,363	2,725	5,450	8,175	10,900	16,350	19,075	21,800

Table 1 Sample calculated IOPS/MBPS capacity for certain database shapes deployed on Oracle Database Appliance X8-2-HA

This white paper does not cover the HDD storage configuration which is a possible variant of the standard (default) all SSD storage configuration for Oracle Database Appliance X8-2-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 <http://dominicgiles.com/downloads.html>

Swingbench version 2.6.1076 was used to perform the testing documented in this white paper. For more information about Swingbench, please refer to Swingbench documentation available at <http://www.dominicgiles.com/swingbench.html>

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

Swingbench Benchmark version 1 uses PL/SQL stored procedures for building the workload whereas version 2 uses discreet JDBC statements over the network across the Swingbench client and the target server. It may be noted that Order Entry schema creation supports version 1 and version 2 type objects. Version 1 type schema was used during testing. Additionally, Oracle 18c JDBC drivers were used during testing.

The Order Entry benchmark is based on the “OE” (Order Entry) schema. The workload uses a read/write ratio of 60/40 and is designed to run continuously against a small set of tables, producing contention for database resources.

Swingbench Download and Setup

The Swingbench tool can be setup in a Windows or Linux environment. A Linux environment was used during the testing. It is recommended that Swingbench client software be installed on a separate machine that has local network connectivity to the target system where the target database workload will be running. Executing Swingbench client from within the target server itself or from across an external network can affect performance measurements and produce inconsistent results.



To install *Swingbench*, perform the following steps.

1. Download *Swingbench* software

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

At the time of writing of this white paper, the latest *Swingbench* version 2.6 (release 2.6.1076) was used during the course of testing. You should download the latest release of *Swingbench* on to the machine from where you plan to run the workload.

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

Configuring Oracle Database Appliance for Testing

This section describes the setup of the Oracle Database Appliance system for the purpose of conducting this performance benchmark.

Configuring Oracle Database Appliance System for Testing

An Oracle Database Appliance X8-2-HA system with a single SSD storage shelf (fully populated) was used for the purpose of testing workloads. All 64 CPU cores were initially enabled on this system.

Note that irrespective of the number of CPUs activated, the Oracle Database Appliance systems are always automatically configured with the entire available physical memory enabled on both the server nodes and only the CPU component is controlled (enabled/disabled) when customers want to use fewer cores.

From a software stack perspective, the system was deployed with Oracle Database Appliance 18.7.0.0.0 software and the database version used for testing was Oracle Database 18.7.0.0.190716 (July 2019 DB/GI RU).

Oracle Database Appliance systems are configured by choosing the desired configuration options during deployment. These include choice of mirroring (double or triple), provisioning storage for backups (local or external), database version, and initially active CPU cores. The setup for this testing included database storage configured with HIGH redundancy (triple mirroring) and external backups. All CPU cores were activated. While no major changes were made at the system level, database configuration related changes were made as outlined in a subsequent section of this white paper.

Benchmark Setup

This section describes the process of setting up the OE schema for running the Order Entry (OE) OLTP type workload. A similar process can be followed for setting up the SH schema for running the Sales History (SH) workload which is DSS type, or any other workload.

In order to setup *Swingbench* benchmarks, the system, database, schema, and the workload itself must first be setup.

Additionally, it may be noted that when Oracle databases are configured on Oracle Database Appliance using the provided command-line interface (CLI), the default database parameter settings are optimized for most common uses. However, these settings may be adjusted for maximizing performance for specific performance intensive workloads. Such changes that were made during testing are outlined in the Database Setup section below.

For the purpose of this benchmark a *Swingbench* client host was setup on a separate host but on the same subnet as the Oracle Database Appliance. Thus the latency between the *Swingbench* client and DB servers was minimized.

Database Setup

You can create both single-instance and clustered (RAC) databases on Oracle Database Appliance. During the course of testing, a RAC database configuration was used. The database was setup using the *Odb32* shape (32 CPU cores).

During database deployment, the database workload type should be specified using the `--dbclass` argument to 'odacli'. If not specified, then the default workload type is ON-LINE TRANSACTION PROCESSING (OLTP).

For the OLTP workload type, database creation templates automatically configured the SGA and PGA in the ratio of 2:1.

```
# odacli create-database --adminpassword --dbhomeid 5d33d8f1-67d5-4e0e-a61a-334cb1f8b938 --  
dbEdition EE --dbname oltpdb --dbshape odb32 --dbtype RAC --no-cdb --dbstorage ASM
```

Further, the following changes were made to the database configuration and the database was restarted once for the relevant changes to take effect.

1. Create a dedicated SOE tablespace

```
SQL> create bigfile tablespace soe datafile size 30g autoextend on maxsize unlimited  
uniform size 1m segment space management auto;
```

2. Increase tablespace sizes

The UNDO tablespace size should be at least 30GB, while the SYSTEM, SYSAUX, and TEMP tablespace sizes should be at least 10GB.

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

```
alter system reset fast_start_mttr_target scope=spfile sid='*';  
alter system set audit_trail=none scope=spfile sid='*';  
alter system set audit_sys_operations=false scope=spfile sid='*';
```

When setting up your own benchmark environment, DO NOT cut and paste the commands listed above as it may introduce control characters. Instead type the commands or set these parameters using Oracle Enterprise Manager Cloud Control.

Note: Database archiving was enabled during all performance tests.

Schema Setup

This section describes the process of setting up the OE schema for running the Order Entry OLTP workload. A similar process can be followed for setting up the SH schema if running the Sales History DSS workload.

It should be noted that the Order Entry workload generates and modifies 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 if you run multiple workload test cycles to avoid inconsistency of results due to expansion and fragmentation of objects.

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

To begin the schema setup process, login to the Swingbench client machine. This client machine may be running the VNC server that you may be connecting to, from a VNC client running on your desktop/laptop. Once connected to the Swingbench client machine, run *oewizard* utility to install the database schema for executing the benchmark.

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

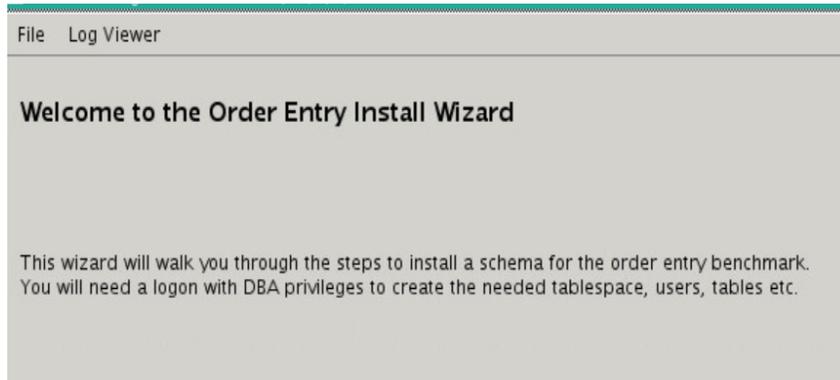


Illustration 1: Swingbench Workload Setup: Starting Order Entry Install Wizard

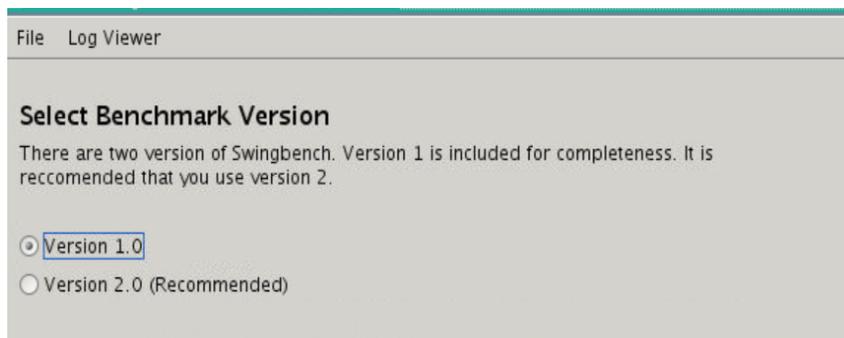


Illustration 2: Swingbench Workload Setup: Order Entry Install Wizard Benchmark Version Selection

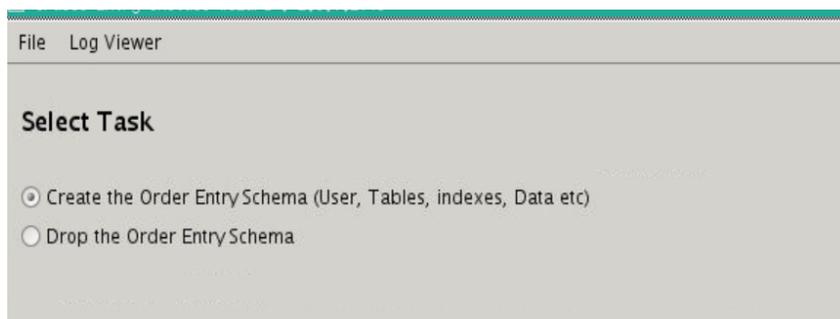


Illustration 3: Swingbench Workload Setup: Order Entry Install Wizard Select Task to Create OE Schema

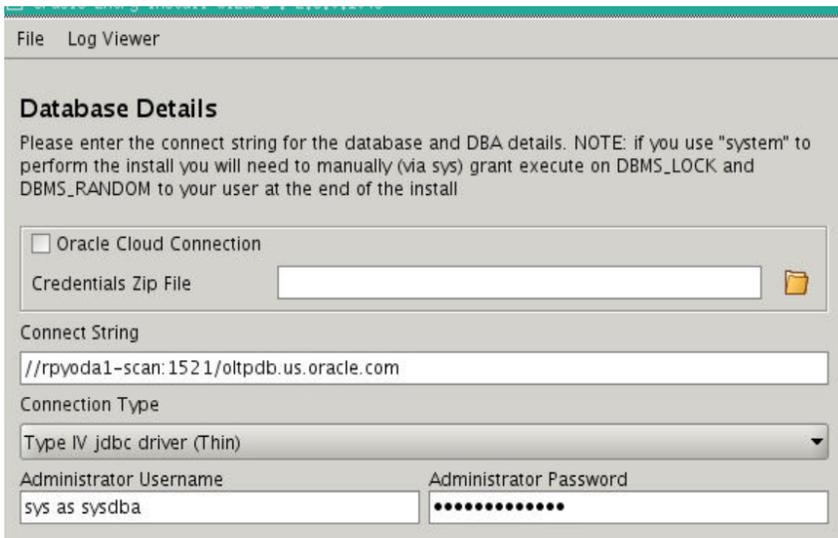


Illustration 4: Swingbench Workload Setup: Provide Database Details in Order Entry Install Wizard

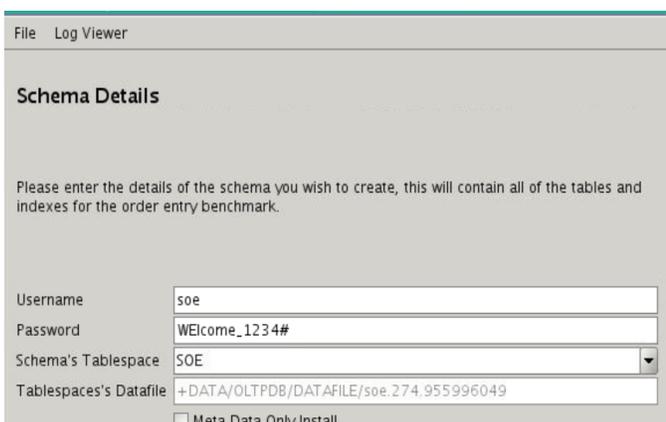
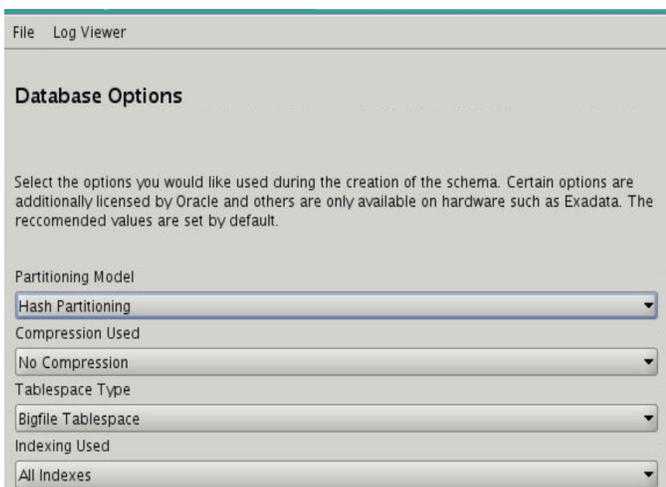


Illustration 5: Swingbench Workload Setup: Provide Schema Details in Order Entry Install Wizard





As noted earlier, the test database for this benchmark was initially created using the odb32 shape. When the database configurations were subsequently changed (scaled down) to fewer active cores, the other database parameters such as SGA size, etc. were not changed down. This decision was made to ensure that the measurements generated are fair for users who may simply want to limit the CPU configuration on the Oracle Database Appliance system and utilize all other available resources without any unnecessary limitations.

Oracle Database Appliance systems allow for the pay-as-you-grow approach to software licensing. However, once provisioned, you have complete access to the hardware. Fewer deployed CPU licenses do not restrict use of memory, storage, and network resources.

As part of this testing, five different CPU configurations were tested by enabling only a given total number of cores (8, 16, 32, 48, and 64) at a time on the Oracle Database Appliance system.

Workload Setup and Execution

The Swingbench Order Entry (OE) workload is an OLTP workload and the Sales History workload is a DSS type workload. The OE workload was used to conduct OLTP workload performance testing during the course of writing of this white paper.

Swingbench allows for configuring a workload to run for a pre-determined period of time.

You can generate the workload by connecting to the Swingbench client machine and launching the loadgen.pl utility.

```
$ perl loadgen.pl -u <#users>
```

The workload simulates multiple concurrent users and allows for specifying “think time” between transactions to make the workload more realistic. During our testing, the workload was simulated with the following attributes.

- » User Count: 150, 320, 680, 950, 1200
- » Active CPU Core Count: 8, 16, 32, 48, 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: 5213, 11016, 22877, 31181, 37538

The OE workload mix used for testing was - Customer Registration (20), Browse Products (80), Order Products (25), Process Orders (25).

Benchmark Results

This section describes the results observed during Swingbench OLTP (OE) workload testing on an Oracle Database Appliance system.

Workload Performance

Table 2 summarizes the performance results captured when executing the Swingbench Order Entry (OLTP) workload on and Oracle Database Appliance system.

Active CPU Core Count: User Count	Workload Element	Total Transactions	Average Response Time (Milliseconds)	Average TPS (Transactions Per Second)
8 Cores: 150	Customer Registration	1251939	2.51	5213
	Process order	1564311	4.68	
	Browse product	5003995	2.78	
	Order Products	1564415	7.48	
16 Cores: 320	Customer Registration	2642736	2.78	11016
	Process order	3304690	5.12	
	Browse product	10578612	3.04	
	Order Products	3303499	7.92	
32 Core: 680	Customer Registration	5491462	3.31	22877
	Process order	6867506	5.90	
	Browse product	21957939	3.61	
	Order Products	6862317	8.79	
48 Core: 950	Customer Registration	7484464	3.77	31181
	Process order	9352951	7.01	
	Browse product	29931083	4.07	
	Order Products	9358421	10.11	
64 Core: 1200	Customer Registration	9012040	4.83	37538
	Process order	11258301	8.79	
	Browse product	36036346	5.22	
	Order Products	11262463	12.10	

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

The above data illustrates the following:

- 1) A 37,538 transactions per second (TPS) workload delivering an average of less than 6.91ms transaction response (average of cumulative transaction response time divided by total number of transactions) on a fully provisioned (64 Cores and 384GB Memory) Oracle Database Appliance X8-2-HA system.

- 2) In fully provisioned state, with a 1,200 user workload, delivering a sustained transaction rate of 37,538 transactions per second (TPS).
- 3) The maximum average transaction response time not exceeding 12.10ms during any of the workload test runs.
- 4) The number of users and number of transactions for given performance levels scaling in linear correlation with system resource allocation (primarily active CPU core count).

Database and Operating System Statistics

This section describes database and operating system statistics collected and observations made during the Swingbench Order Entry (OE) OLTP workload execution on an Oracle Database Appliance X8-2-HA system.

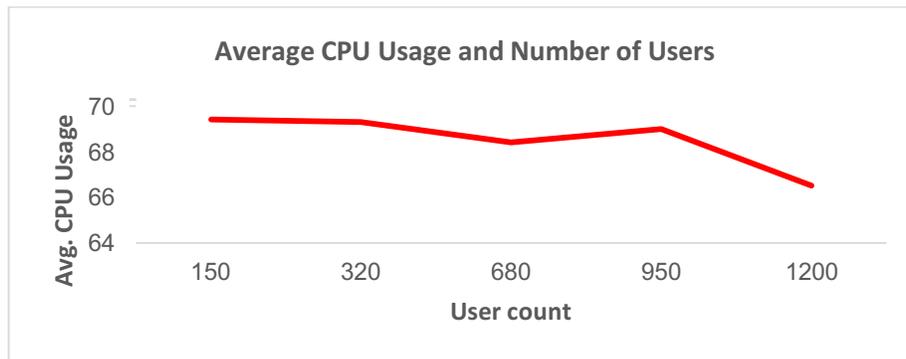
The Oracle Database Appliance X8-2-HA system provides up to 64 CPU cores (32 CPU cores on each host). The active CPU core count on the system is dynamically scalable from 4-cores to 64-cores. During the benchmark, a total of five configurations were tested and the total active CPU core count was gradually scaled between 8 active CPU cores and 64 active CPU cores as illustrated in table 2.

Some of the key observations related to database and operating system statistics collected during the OLTP benchmark are as follows:

- 1) During the workload runs, connections were distributed evenly (although not perfectly) across the two hosts and in no case average CPU usage exceeded 69.4% on either DB host.
- 2) With the configured OLTP workload, transaction rates increased linearly with user volumes as expected.
- 3) Redo read and write volumes increased as expected with the increase in transaction volumes.

Average CPU Busy

Average CPU consumption across the two hosts of the Oracle Database Appliance was measured for each test cycle. The overall CPU busy % was observed to fluctuate within a tight range, between about 66% and 69%. Note that the CPU % Busy measurement is the average of the readings on the two DB hosts on the Oracle Database Appliance X8-2-HA system.

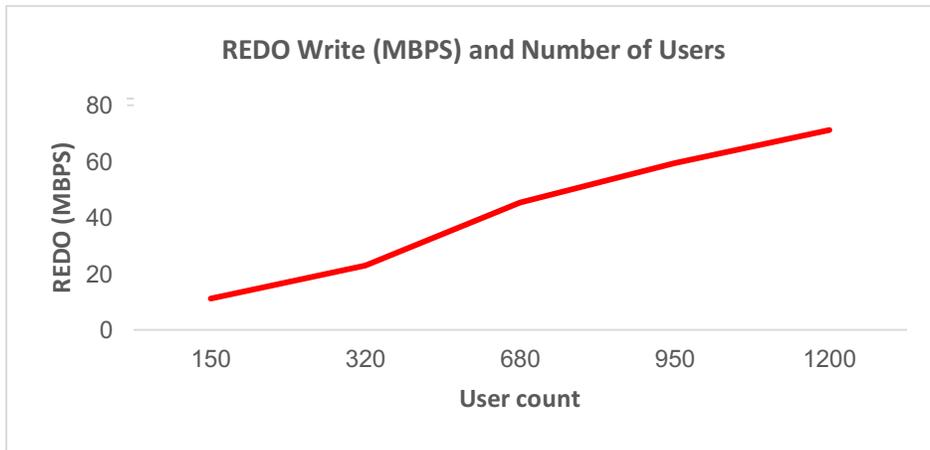


Graph 1: Average CPU Busy

On average, each host of the Oracle Database Appliance maintained a healthy CPU consumption level, with capacity to spare and was not CPU constrained.

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. As expected, the 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 reading for the two DB hosts.

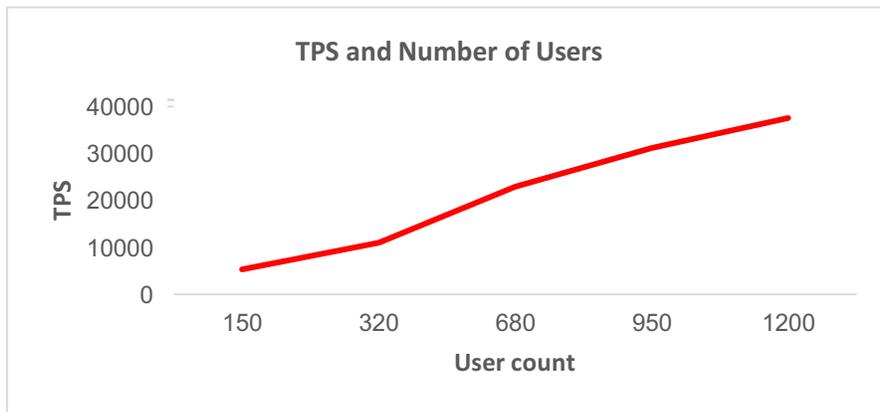


Graph 2: Average REDO Write volume (MB/Second)

Transaction Rate

During the benchmark, the transaction rate (average transactions per second) scaled almost linearly as transaction volume increased from about 5200 transactions per second to about 37000 transactions per second and active CPU core count was increased from 8 cores to 64 cores.

Note that the transaction volume measurement is cumulative from across the two DB hosts.



Graph 3: Average Transaction Volume (Transactions/Second)

Note that in the graph 3, the transaction volume and active CPU core count was changed in tandem while maintaining and measuring transaction performance.

Important Considerations for Performing the Benchmark

Oracle Database Appliance system is built using high performance storage, networking, CPUs and memory components. However, in order to achieve most optimal performance from your Oracle Database Appliance system, certain considerations may be important to note.

This section provides some general guidelines for operating your Oracle Database Appliance environments at its optimal performance levels, during the course of running a benchmark or otherwise.

1. Ensure that databases on Oracle Database Appliance are always created using the Browser User Interface (BUI) or *odacli* command-line interface (previously, the *oakcli* command-line interface) as both of these interfaces use pre-built templates that provide pre-optimized database parameter settings for given DB shapes and sizes.
2. When performing benchmarks in two different environments ensure that an identical workload is run for apples to apples comparison. If you are running different workloads (different SQL, different commit rates, or even if you only have different execution plans, etc.) in the legacy and the Oracle Database Appliance environment, then platform performance comparisons may be pointless.
3. Pay attention to network latency. For example, running the workload client(s) on the same network (but on a separate host) as your Oracle Database Appliance is on, avoids potentially significant latency in the transaction path.

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

4. Size the Oracle Database Appliance environment appropriately and adequately. When conducting benchmarks, it is imperative 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 be different, there may be index differences, or lack of proper optimizer statistics, etc. that may contribute to differences in SQL execution plans and execution timings.
6. When possible, perform comparisons and benchmarks between systems that are running 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 Database Appliance, make sure you do not continue to use obsolete, un-optimized settings for parameters. Do not modify database parameters blindly to match the database parameters from your legacy environment. You may use the “*orachk*” tool to verify your database configuration running on Oracle Database Appliance and the legacy environments.
8. Oracle Database Appliance provides features such as database block checking and verification to protect against data corruption. These features may consume some, albeit small, amount of CPU capacity but are generally desirable to protect the integrity of data. While these features may be temporarily disabled for test purposes, it is recommended that you use these protective features to mitigate data corruption risks.



Conclusion

The performance benchmark conducted during the course of the writing of this white paper indicates that Oracle Database Appliance offers excellent performance for typical database workloads. In a fully sized configuration, an Oracle Database Appliance X8-2-HA system was able to easily support a 37,538 transactions per second (TPS) Swingbench OLTP workload, while maintaining a transaction response time of less than 12.1 milliseconds. Furthermore, the performance scaled almost linearly as workload and CPU resources were increased in tandem as described in detail in this white paper.

The testing conducted during the course of writing of this white paper indicated that some configuration adjustments and tuning of the default configuration can further enhance workload performance.

Appendix A Swingbench configuration files

This section provides a sample listing of the swingbench.xml configuration file that were used for testing of the Swingbench OLTP (OE) workload.

This configuration file can be used for repeating the same tests on any legacy platform. However, the database connection information will need to be updated as appropriate. Please note that other parameters should not be changed for a correct comparison between an existing legacy 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) V1"</Name>
  <Comment>Version 1 of the SOE Benchmark running in the database using PL/SQL</Comment>
  <Connection>
    <UserName>soe</UserName>
    <Password>enc (RQRUi/BMGRK756a5QotrSg==)</Password>
    <ConnectionString>//myoda-scan.example.com:1521/swdb.example.com</ConnectionString>
    <DriverType>Oracle jdbc Driver</DriverType>
    <ConnectionPooling>
      <PooledInitialLimit>100</PooledInitialLimit>
      <PooledMinLimit>250</PooledMinLimit>
      <PooledMaxLimit>250</PooledMaxLimit>
      <PooledInactivityTimeout>0</PooledInactivityTimeout>
      <PooledConnectionWaitTimeout>0</PooledConnectionWaitTimeout>
      <PooledPropertyCheckInterval>0</PooledPropertyCheckInterval>
      <PooledAbandonedConnectionTimeout>0</PooledAbandonedConnectionTimeout>
    </ConnectionPooling>
    <Properties>
      <Property Key="TcpNoDelay">true</Property>
      <Property Key="StatementCaching">50</Property>
    </Properties>
  </Connection>
  <Load>
    <NumberOfUsers>720</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>
  <TransactionList>
    <Transaction>
      <Id>Customer Registration</Id>
      <ShortName>NCR</ShortName>
    </Transaction>
  </TransactionList>
  <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewCustomerProcess</ClassName>
  <Weight>20</Weight>
  <Enabled>true</Enabled>
</SwingBenchConfiguration>
```

```

    </Transaction>
    <Transaction>
      <Id>Browse Products</Id>
      <ShortName>BP</ShortName>
    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.BrowseProducts</ClassName>
      <Weight>80</Weight>
      <Enabled>>true</Enabled>
    </Transaction>
    <Transaction>
      <Id>Order Products</Id>
      <ShortName>OP</ShortName>
    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewOrderProcess</ClassName>
      <Weight>25</Weight>
      <Enabled>>true</Enabled>
    </Transaction>
    <Transaction>
      <Id>Process Orders</Id>
      <ShortName>PO</ShortName>
    <ClassName>com.dom.benchmarking.swingbench.plsqltransactions.ProcessOrders</ClassName>
      <Weight>25</Weight>
      <Enabled>>true</Enabled>
    </Transaction>
  </TransactionList>
  <EnvironmentVariables>
    <Variable Key="SOE_NLSDATA_LOC">data/nls.txt</Variable>
    <Variable Key="SOE_PRODUCTSDATA_LOC">data/productids.txt</Variable>
    <Variable Key="SOE_NAMESDATA_LOC">data/names.txt</Variable>
  </EnvironmentVariables>
</Load>
<Database>
  <SystemUserName>system</SystemUserName>
<SystemPassword>enc (RQRUi/BMGRK756a5QotrSg==)</SystemPassword>
<PerformAWRSnapShots>>true</PerformAWRSnapShots>
  </Database>
  <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>
      </Chart>
    </Charts>
  </Preferences>

```

```

        <Logarithmic>false</Logarithmic>
    </Chart>
</Charts>
<AllowedErrorCodes/>
<RefreshRate>1</RefreshRate>
<OverviewCharts>
    <OverviewChart>
        <Name>Database Time</Name>
<MinimumValue>2.147483647E9</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
        <Displayed>true</Displayed>
    </OverviewChart>
    <OverviewChart>
        <Name>Users</Name>
<MinimumValue>2.147483647E9</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
        <Displayed>true</Displayed>
    </OverviewChart>
    <OverviewChart>
        <Name>Disk</Name>
<MinimumValue>2.147483647E9</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
        <Displayed>false</Displayed>
    </OverviewChart>
    <OverviewChart>
        <Name>CPU</Name>
        <MinimumValue>0.0</MinimumValue>
        <MaximumValue>100.0</MaximumValue>
        <Displayed>true</Displayed>
    </OverviewChart>
    <OverviewChart>
        <Name>Response Time</Name>
<MinimumValue>2.147483647E9</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
        <Displayed>true</Displayed>
    </OverviewChart>
    <OverviewChart>
        <Name>Transactions Per Second</Name>
<MinimumValue>2.147483647E9</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
        <Displayed>true</Displayed>
    </OverviewChart>
    <OverviewChart>
        <Name>Transactions Per Minute</Name>
<MinimumValue>2.147483647E9</MinimumValue>
<MaximumValue>2.147483647E9</MaximumValue>
        <Displayed>true</Displayed>
    </OverviewChart>
</OverviewCharts>
</Preferences>
</SwingBenchConfiguration>

```

Appendix B The loadgen.pl file

Note that you may need to update the sample password 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.example.com";
my $cdb_service="swdb.example.com";
my $port=1521;
my $dbauser="system";
my $dbapwd="WElcome_1234#";
my $config_file_1="SOE_Server_Side_V1_swdb.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 ($l_counter) = @_;
#print "User count"$. $l_counter. "\n";
#print "Out dir"$. $sb_output_dir. "\n";
#print "Run Date "$. $rundate. "\n";
system ("$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 "$_[0]" ) {
print "Direcotry "$. $_[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 $scr_tc=0;
my $scr_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") {
$scr_tc = $scr_tc+$txn_cnt;
}
}
}

```

```

$scr_to_rt = $scr_to_rt+($avg_rt*$txn_cnt);
}
elseif ($app_module eq "Process Orders") {
$spo_tc = $spo_tc+$txn_cnt;
$spo_to_rt = $spo_to_rt+($avg_rt*$txn_cnt);
}
elseif ($app_module eq "Browse Products") {
$bp_tc = $bp_tc+$txn_cnt;
$bp_to_rt = $bp_to_rt+($avg_rt*$txn_cnt);
}
elseif ($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, $scr_tc, ($scr_to_rt/$scr_tc);
}
elseif ($app_module eq "Process Orders") {
printf $OUTFILE "%-22s %16s %0.2f\n", $app_module, $spo_tc, ($spo_to_rt/$spo_tc);
}
elseif ($app_module eq "Browse Products") {
printf $OUTFILE "%-22s %16s %0.2f\n", $app_module, $bp_tc, ($bp_to_rt/$bp_tc);
}
elseif ($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 The generate_awr.sh script

Note that you may need to update the sample password used in the script below.

```
#!/bin/bash
export host=myoda-scan.example.com
l_dbid=449409023
export svc="swdb.example.com"
export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
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}
$ORACLE_HOME/bin/sqlplus -s system/WELCOME_1234#@$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_snapid ));
spool off
exit;
EOC

$ORACLE_HOME/bin/sqlplus -s system/WELCOME_1234#@$host:$port/$svc << 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
```



References

Oracle Database Appliance X8-2-HA Data Sheet

<https://www.oracle.com/technetwork/database/database-appliance/oda-x8-2-ha-datasheet-5730739.pdf>

Oracle Database Appliance X8-2-HA

<https://www.oracle.com/engineered-systems/database-appliance/x8-2-ha>

Oracle Database Appliance Documentation

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

Swingbench

<http://dominicgiles.com/swingbench.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

CONNECT WITH US

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

Integrated Cloud Applications & Platform Services

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. This document is provided *for* information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0615

Evaluating and Comparing Oracle Database Appliance Performance

Date: January 2020

Authors: Paramdeep Saini, Paulo Qiu, Ravi Sharma, Sanjay Singh

Contributing Authors: Francis Na, Erich Kreisler, RACPack, Cloud Innovation and Solution Engineering Team