Business / Technical Brief

Evaluating and Comparing Oracle Database Appliance Performance

Updated for Oracle Database Appliance X10-HA

ORACLE TECHNICAL BRIEF | Feb 2024

Feb, 2024, Version 1.40

DISCLAIMER

The performance results in this paper are intended to give an estimate of the overall performance of the Oracle Database Appliance X10-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 Speculative Return Stack Overflow (SRSO) 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

Oracle Database Appliance X10 is Oracle's popular entry-level Engineered System. Oracle Database Appliance (ODA) has been adopted worldwide and in all industries by many customers, especially small and medium-sized organizations and business units and those with distributed locations. It allows them to run Oracle Database and applications on economical, easy-to-deploy/manage systems.

Oracle Database Appliance X10 High-Availability (HA) system includes hardware, software, networking, and storage in an integrated, pre-built, pre-tuned, packaged database solution with a modest 8-rack unit (RU) footprint. Oracle Database Appliance X10-HA's hardware and software combination offers redundancy and protects against all single points of failure in the system.

Oracle Database Appliance X10-HA is a two-node cluster based on the most recent AMD EPYC™ 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 8 operating system (OS), Oracle Relational Database Management System (RDBMS), Oracle Real Application Clusters software (RAC), Oracle Grid Infrastructure (GI), and Oracle Automatic Storage Management (ASM) 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 X10-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 X10 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 X10-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 X10-HA high-performance model supported more than 87k concurrent Swingbench transactions per second with all 64 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 environments or on earlier ODA models and compare 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 X10-HA performance models.

This study was conducted by using Swingbench workloads on an ODA X10 HA that was running 19.21.0.0 ODA software release version which includes the Speculative Return Stack Overflow mitigation.

User and transaction volumes 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 X10 HA and compare it to the non-X10 ODA or a DIY system is to capture the workload on the non-X10 ODA or a DIY system 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*](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 X10-HA Data Sheet available at https://www.oracle.com/engineered-systems/database-appliance/

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.

Even if 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 X10-HA Deployment and User's Guide (Appendix D 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 X10-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 3,251,529 IOPS and throughput of up to 29,602 MBPS with a fully occupied, twin storage shelf configuration, compared to 1,649,000 IOPS and 14,797 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.

Table 1 presents the measured raw IOPS and bandwidth using both nodes, based on the number of disks in disk shelf on the Oracle Database Appliance X10-HA system.

# OF DRIVES	SIX	TWELVE	EIGHTTEEN	ONE SHELF (24 DRIVES)	TWO SHELVES (48 DRIVES)
IOPS	1 151 517	1 506 132	1 564 838	1 649 000	3 251 529
MBPS	10 920	13 857	14 583	14 797	29 602

Table 1: IOPS and MBPS capacity for different number of disks on Oracle Database Appliance X10-HA

Since this technical brief focuses on performance, only high-performance configuration (SSDs) is covered, though high-capacity model (SSDs+HDDs) is also available for Oracle Database Appliance X10-HA.

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 older versions of the technical brief, Order Entry version 1 was used for the benchmarks, but that version is not in use anymore. X9-2 and X10 performance technical briefs use version 2.

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 2.7.1313).
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 X10-HA system with a single storage shelf, fully populated with SSDs was used to perform the tests documented in this technical brief. All 128 CPU cores were enabled initially on the ODA.
X10 HA had 1.5TB. Database shape defines 256 GB SGA and 128 GB PGA.

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 19.21.0.0 software and the database version used for testing was Oracle Database 19.21.0.0.231017 (Oct 2023 DB/GI RU). Disk Groups 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 database 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 *Odb64* shape (64 CPU cores a node).

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, odb64 database shape defines SGA-PGA ratio 2:1 (SGA: 256GB, PGA:128GB).

```
# odacli create-database -v 19.21 --dbname mycdb --dbshape odb64 --dbtype RAC --cdb --dbstorage ASM --associated-networks
   Public-network --pdbname oltpdb
```

Implement the following modifications

1. Create a dedicated SOE tablespace in the PDB

   ```
   sqlplus / as sysdba

   SQL> alter session set container=oltpdb;
   SQL> create bigfile tablespace soe datafile size 2T autoextend on maxsize unlimited uniform size 20m segment space
   management auto;
   ```

2. Increase tablespace sizes of the CDB

```
sqlplus / as sysdba

SQL> set linesize 600
SQL> set pagesize 100
SQL> column FILE# Format 999
SQL> column NAME  Format a100
SQL> column SIZEINMB Format 9999999
SQL> select file#, name, round(bytes/1024/1024) sizeinMB from v$datafile;

FILE# NAME                                             SIZEINMB
----- ---------------------------------------------------------------------------------------------
    1 +DATA/MYCDB/DATAFILE/system.269.1143318723                            1040
    3 +DATA/MYCDB/DATAFILE/sysaux.270.1143318747                            1270
    4 +DATA/MYCDB/DATAFILE/undotbs1.271.1143318755                            90
    5 +DATA/MYCDB/00B11CA841E0A29BE063CF0AF40A2C07/DATAFILE/system.273.1143318831        420
    6 +DATA/MYCDB/00B11CA841E0A29BE063CF0AF40A2C07/DATAFILE/sysaux.274.1143318831        390
    7 +DATA/MYCDB/00B11CA841E0A29BE063CF0AF40A2C07/DATAFILE/undotbs1.275.1143318831       60
    8 +DATA/MYCDB/DATAFILE/undotbs2.277.1143319047                            25
    9 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/system.281.1143319365      10240
   10 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/sysaux.280.1143319365      10240
   11 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/undotbs1.279.1143319365      30720
   12 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/undo_2.283.1143319371       30720
   13 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/users.284.1143319371        5
   14 +DATA/MYCDB/DATAFILE/users.285.1143319691                             5

13 rows selected.

SQL> alter database datafile 1 resize 10G;
SQL> alter database datafile 3 resize 10G;
SQL> alter database datafile 4 resize 30G;
SQL> alter database datafile 8 resize 30G;
```

3. Increase tablespace sizes inside the PDB

sqlplus / as sysdba

```
SQL> alter session set container=oltpdb;
SQL> set linesize 600
SQL> set pagesize 100
SQL> column FILE# Format 999
SQL> column NAME  Format a100
SQL> column SIZEINMB Format 9999999
```

```
select file#, name, round(bytes/1024/1024) sizeinMB from v$datafile;
```

```
FILE# NAME                                         SIZEINMB
----- ------------------------------------------------------------------------------------------
    9 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/system.281.1143319365          420
   10 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/sysaux.280.1143319365          470
   11 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/undotbs1.279.1143319365         60
   12 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/undo_2.283.1143319371           60
   13 +DATA/MYCDB/017D5470FFAD60EDE06336715F0AE74E/DATAFILE/users.284.1143319371            5
```

```
SQL> alter database datafile 9 resize 10G;
SQL> alter database datafile 10 resize 10G;
SQL> alter database datafile 11 resize 30G;
SQL> alter database datafile 12 resize 30G;
```

```
SQL> alter tablespace undotbs1 add datafile size 30g autoextend off;
SQL> alter tablespace undo_2 add datafile size 30g autoextend off;
SQL> alter tablespace undotbs1 add datafile size 30g autoextend off;
SQL> alter tablespace undo_2 add datafile size 30g autoextend off;
SQL> alter tablespace undotbs1 add datafile size 30g autoextend off;
SQL> alter tablespace undo_2 add datafile size 30g autoextend off;
```

4. Add tempfiles to the PDB

sqlplus / as sysdba

```
SQL> alter session set container=oltpdb;
SQL> alter tablespace temp add tempfile '+DATA' size 16g autoextend on next 1g maxsize 32767m;
SQL> alter tablespace temp add tempfile '+DATA' size 16g autoextend on next 1g maxsize 32767m;
SQL> alter tablespace temp add tempfile '+DATA' size 16g autoextend on next 1g maxsize 32767m;
SQL> alter tablespace temp add tempfile '+DATA' size 16g autoextend on next 1g maxsize 32767m;
```

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

sqlplus / as sysdba

```
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 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> select thread#, group# from v$log;

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;
```

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

```
sqlplus / as sysdba

alter system set fast_start_mttr_target='900' scope=spfile sid='*';
alter system set db_recovery_file_dest_size='16T' scope=both;
alter system set resource_manager_plan = '';
alter system set "_lm_res_hash_bucket"=65536 scope=spfile;
alter system set target_pdbs=5 scope=spfile;
alter system set max_pdbs=5 scope=both;
```

7. Restart the database

```
srvctl stop database -d mycdb
srvctl start database -d mycdb
```

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-instantclient-sqlplus-21.12.0.0.0-1.el8.x86_64
   oracle-instantclient-basic-21.12.0.0.0-1.el8.x86_64
   oracle-instantclient-release-el8-1.0-2.el8.x86_64
   oracle-instantclient-devel-21.12.0.0.0-1.el8.x86_64

2. Download and install java
   https://www.oracle.com/uk/java/technologies/downloads/
   # yum install /root/jdk-21_linux-x64_bin.rpm

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

   export PATH=/u01/app/19.21.0.0/grid/perl/bin:$PATH
   # which perl
   /u01/app/19.21.0.0/grid/perl/bin/perl
   # perl -version
   This is perl 5, version 36, subversion 0 (v5.36.0) 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/19.21.0.0/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 flashback your database SOE database schemas 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 duplicating the PDB containing it. 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

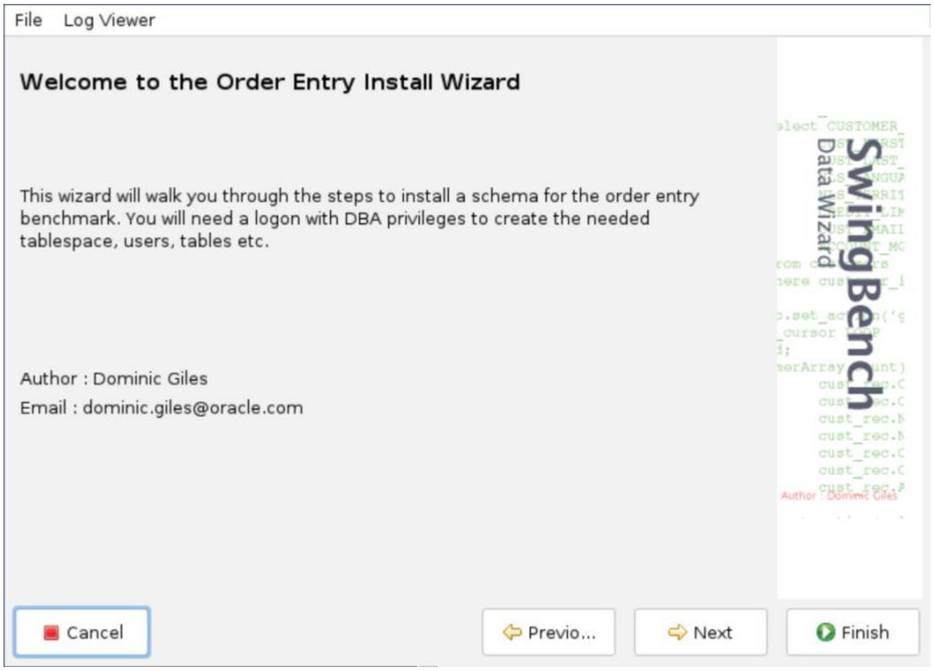


Illustration 1: Swingbench Workload Setup: Starting Order Entry Install Wizard

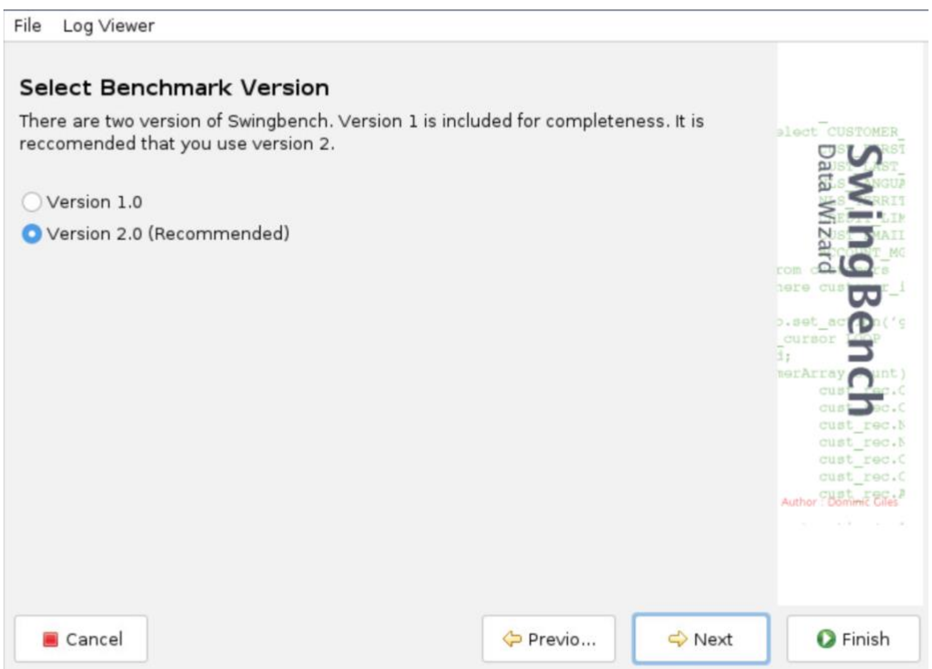


Illustration 2: Swingbench Workload Setup: Order Entry Install Wizard Benchmark Version Selection

Use the PDB’s service name in the connect string

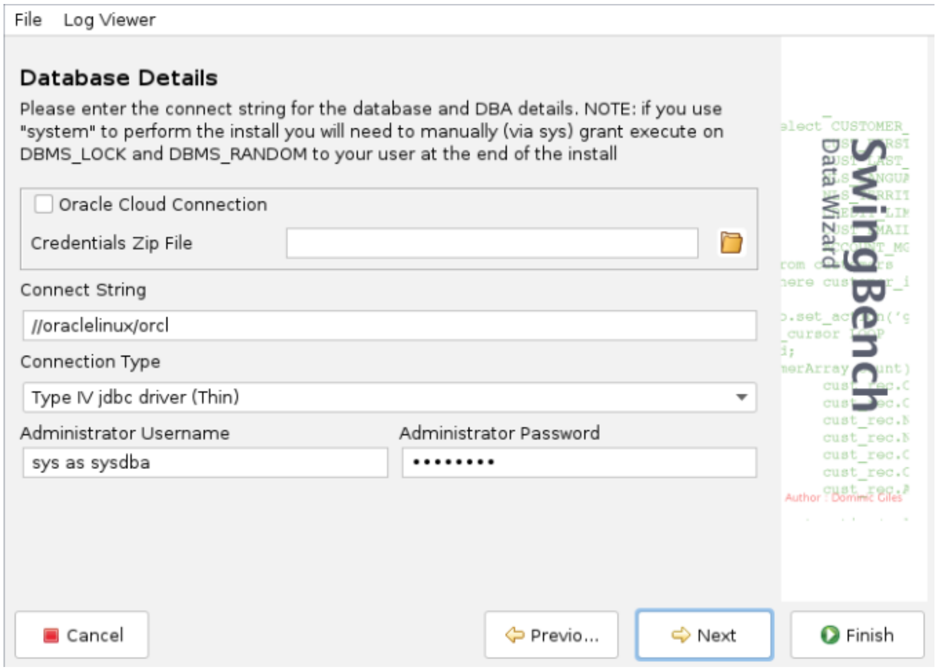


Illustration 3: Swingbench Workload Setup: Order Entry Install Wizard Database Details

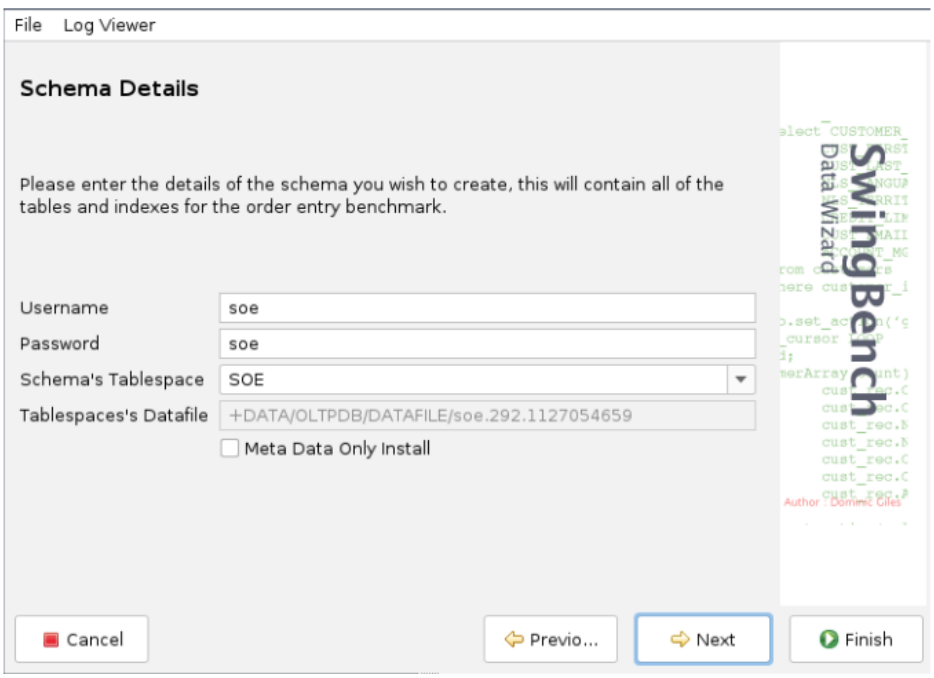


Illustration 4: Swingbench Workload Setup: Provide Schema Details in Order Entry Install Wizard

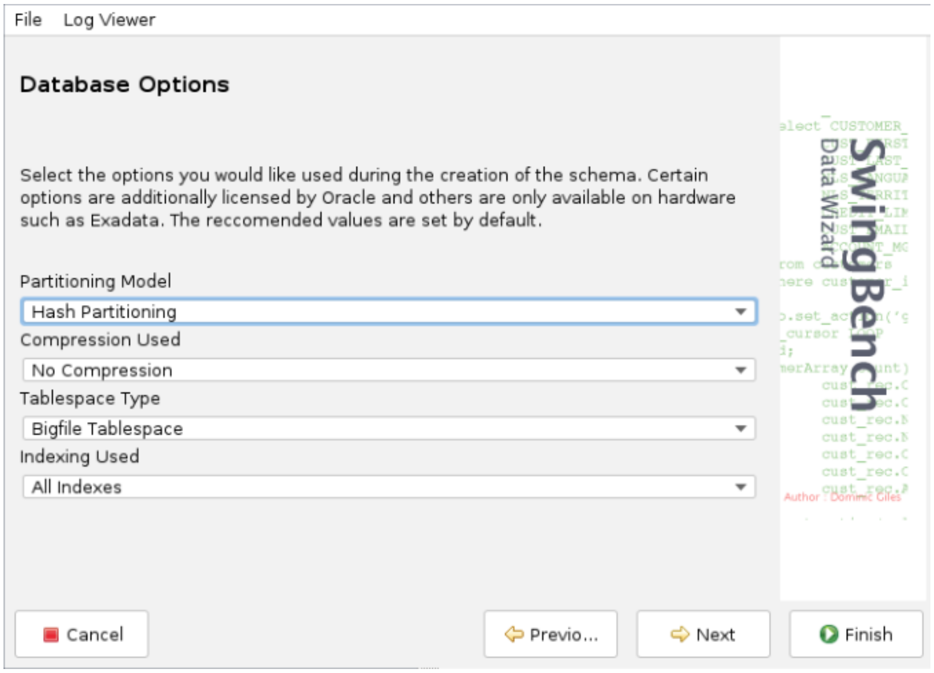


Illustration 5: Swingbench Workload Setup: Select Database Options in Order Entry Install Wizard

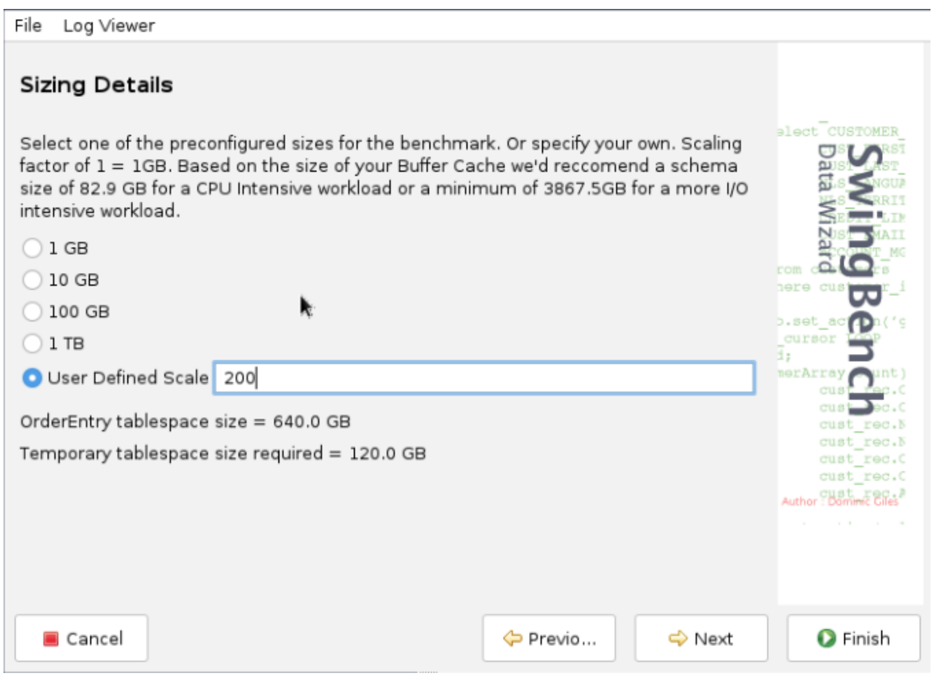


Illustration 6: Swingbench Workload Setup: Select Schema Size for Benchmark (Note: size chosen for final runs was 200GB)

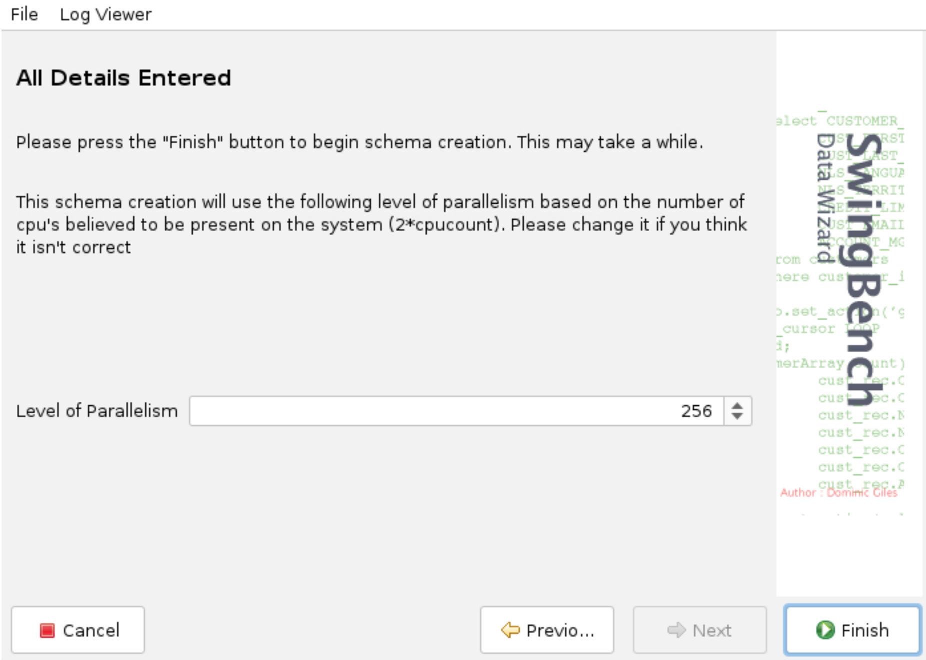


Illustration 7: Swingbench Workload Setup: Select Schema Creation Parallelism for Benchmark in Order Entry Install Wizard

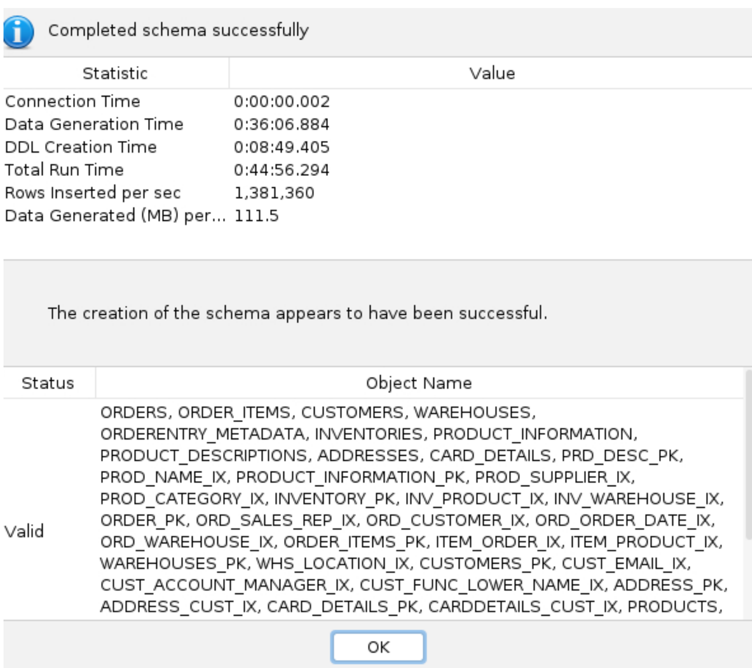


Illustration 8: Swingbench Workload Setup: Schema Created Successfully on X10 HA

Once the schema is ready, drop the indexes that benchmark doesn't use and recreate carddetails_cust_ix index.

sqlplus / as sysdba

sql> alter session set container=oltpdb;

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;
sql> drop index soe.carddetails_cust_ix;
sql> create index "soe"."carddetails_cust_ix" on "soe"."card_details" ("customer_id") global partition by hash ("customer_id") partitions 200
   pctfree 50 initrans 100 maxtrans 255 compute statistics nologging
   storage(initial 1048576 next 1048576 minextents 1 maxextents 2147483645
   pctincrease 0 freelists 1 freelist groups 1
   buffer_pool default flash_cache default cell_flash_cache default)
   tablespace "soe"
   parallel 256;
```

Finally duplicate the PDB including the SOE as the test uses 2 PDBs. Refer to Appendix D.

As mentioned earlier, test database for this benchmark was created using database shape odb64. 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, seven different CPU configurations were tested by enabling only a given total number of cores (8, 16, 32, 48,64, 96 and 128) 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.

Run Swingbench on both nodes using the same number of user connections to simulate workload. Swingbench on the first node should use oltpdb pluggable database and open connections to the local database instance only. Swingbench on the second node should use oltp2db pluggable database and open connections to its local database instance only.

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, 900, 1200, 1500, 1800
- Active CPU Core Count: 8, 16, 32, 48, 64, 96, 128
- 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: 6183, 12672, 25061, 36518, 50737, 69843, 86547

Benchmark Results

In this section, the results of Swingbench OLTP (OE) workload testing on Oracle Database Appliance X10 HA is 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

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 E.

Active CPU Core Count: User Count	Workload Element	Total Transactions	Average Response Time (Milliseconds)	Average TPS (Transactions Per Second)
4 cores: 150 users		**6233694**		**6183**
	Customer Registration	1336531	8,99	
	Process Orders	446351	6,93	
	Browse Products	4450812	4,21	
8 cores: 300 users		**12772050**		**12672**
	Customer Registration	2735867	10,23	
	Process Orders	910831	8,6	
	Browse Products	9125352	4,56	
16 cores: 600 users		**25263538**		**25061**
	Customer Registration	5413650	10,71	
	Process Orders	1805940	9,25	
	Browse Products	18043948	4,23	
24 cores: 900 users		**36809104**		**36518**
	Customer Registration	7888862	11,85	
	Process Orders	2630684	10,45	
	Browse Products	26289558	4,18	
32 cores: 1200 users		**51147189**		**50737**
	Customer Registration	10959840	10,59	
	Process Orders	3655992	9,59	
	Browse Products	36531357	3,66	
48 cores: 1500 users		**70401462**		**69843**
	Customer Registration	15091832	8,13	
	Process Orders	5029954	7,27	
	Browse Products	50279676	2,22	
64 cores: 1800 users		**87239133**		**86547**
	Customer Registration	18699260	7,69	
	Process Orders	6229923	6,84	
	Browse Products	62309950	1,86	

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

The above data illustrates the following:

1) An 86,574 transactions per second (TPS) workload delivering an average of 7,69ms transaction response for Customer Registration, 6.84ms for Process Orders and 1.85ms or Browse Products on a fully provisioned (128 CPU cores and 1.5TB memory) Oracle Database Appliance X10-HA system.
2) In fully provisioned state, during a 1,800 user workload, it delivered a sustained transaction rate of 86,574 transactions per seconds (TPS).
3) The maximum average transaction response time did not exceed 12,27ms during any of the workload test runs.
4) Number of CPU cores and number of transactions scaled almost linearly.

Database and Operating System Statistics

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

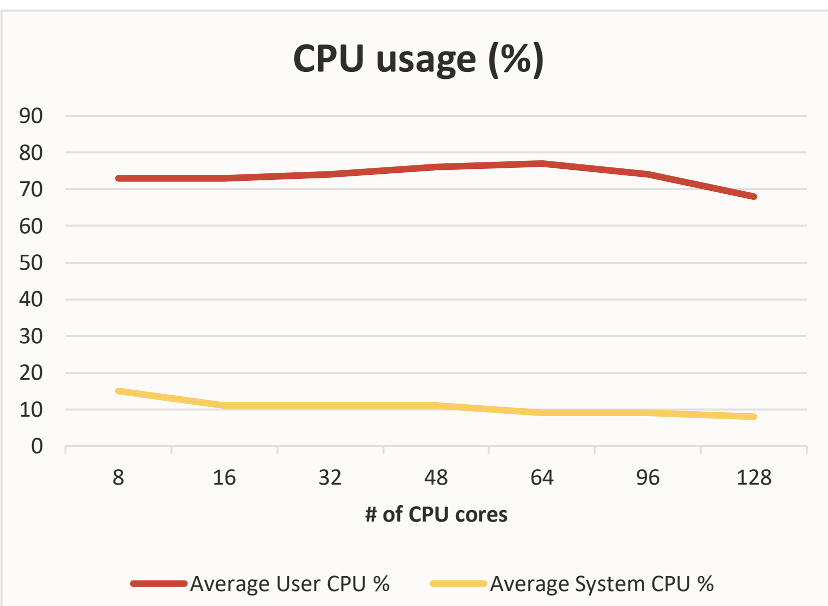
On Oracle Database Appliance X10-HA machine, 128 CPU cores are available (64 CPU cores on each host). The system's active CPU core count is dynamically expandable from 8 to 128. As shown in table 2, a total of seven configurations were examined during the benchmark, and the total number of active CPU cores were steadily raised between 8 and 128.

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

1) Average user CPU usage on each DB host never went above 78%
2) With the configured OLTP workload, transaction rates increased 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 67% and 80%, 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 8 and 24 percent during the tests.

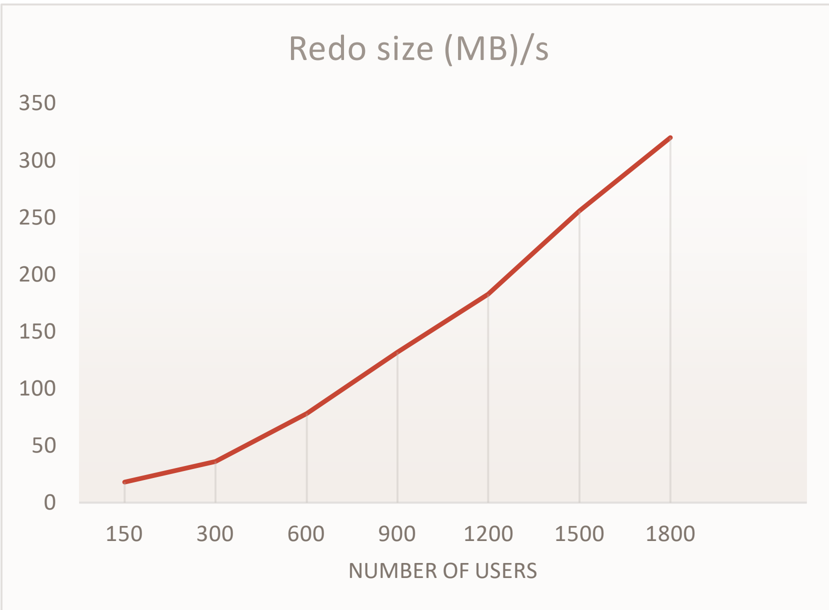


Graph 1: Average User CPU Busy

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.

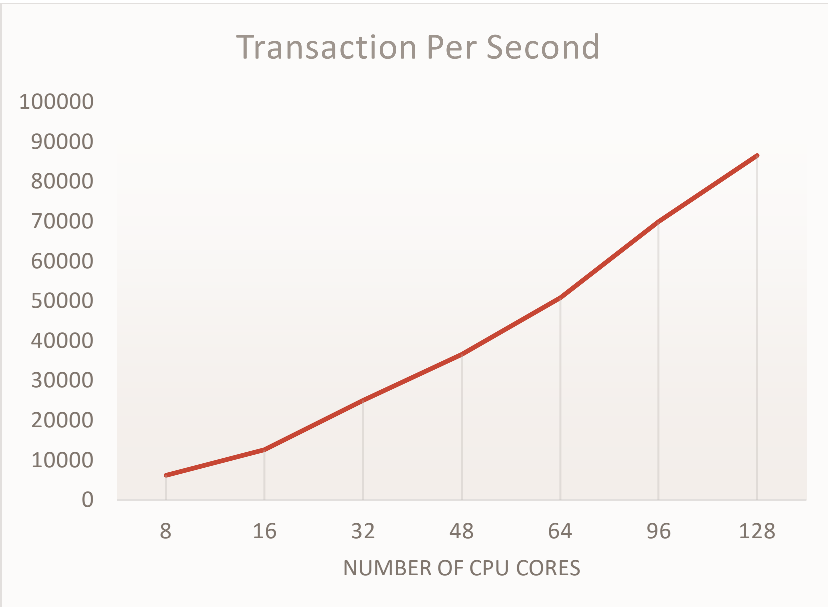


Graph 2: Average REDO Write volume (MB/s)

Transaction Rate

As transaction volume climbed from around 6,120 TPS to approximately 88,763 TPS and the number of active CPU cores increased from 8 to 128 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 3: Average Transaction Volume (Transactions Per Second)

Transaction Per Second rate using 64 cores was 42% higher than on X9-2HA using the same benchmark method.
X10HA: 50737 TPS, X9-2HA:35656 TPS

Benchmark 2

Data generation using oewizard.

<u>X10 HA</u>

Completed schema successfully

Statistic	Value
Connection Time	0:00:00.005
Data Generation Time	0:37:36.272
DDL Creation Time	0:09:07.411
Total Run Time	0:46:43.694
Rows Inserted per sec	1,326,634
Data Generated (MB) per...	107.0

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 3: Swingbench OE wizard screenshot, completion time – X10 HA

<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 4: Swingbench OE wizard screenshot, completion time – X9-2 HA

X10 HA completed the data generation 28 mins earlier than X9-2 HA which means X10 HA was 62% faster than X9-2 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 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 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 Database Appliance, 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 verify your 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 X10-HA system was easily able to manage a Swingbench OLTP workload of 86,547 transactions per second (TPS) with 128 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/mycdb1</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>0</InterMinDelay>
    <InterMaxDelay>30</InterMaxDelay>
    <QueryTimeout>120</QueryTimeout>
    <MaxTransactions>-1</MaxTransactions>
    <RunTime>0:50</RunTime>
    <LogonGroupCount>1</LogonGroupCount>
    <LogonDelay>20</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=200;
```

```
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");
system ("lscpu|grep On-line");
system ("cat /sys/devices/system/cpu/vulnerabilities/spec_rstack_overflow");
print " Exiting .. \n";
exit 0;
```

Appendix C - generate_awr.sh

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/19.21.0.0/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
$ORACLE_HOME/bin/sqlplus -s system/welcome1@$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 – Duplicate oltpdb Pluggable Database

Use dbca to create the 2nd PDB using oltpdb PDB as a source.

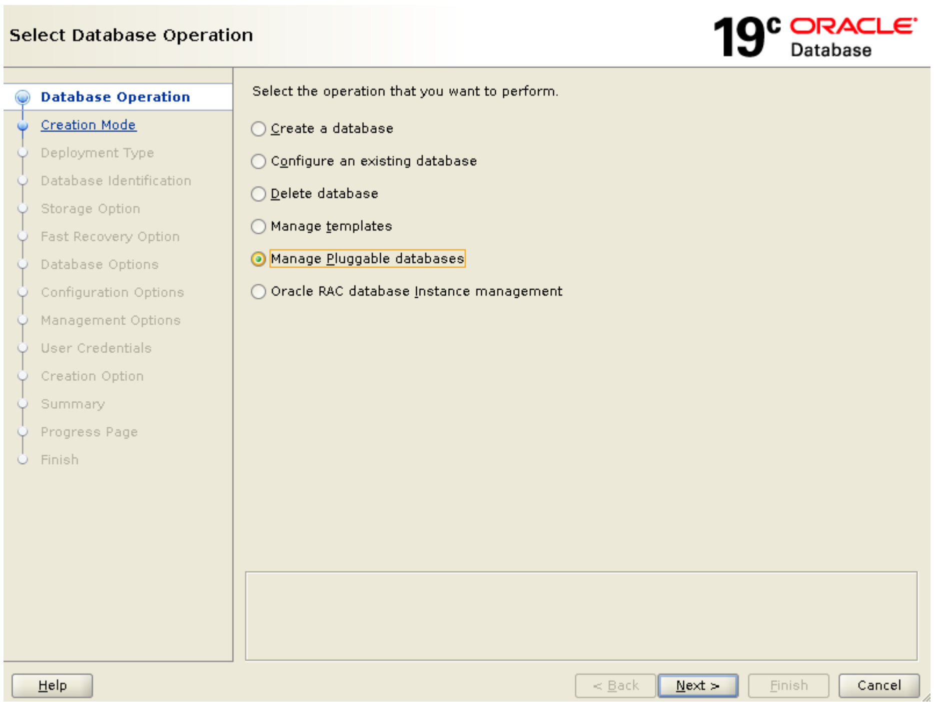


Illustration 9: Manage Pluggable databases

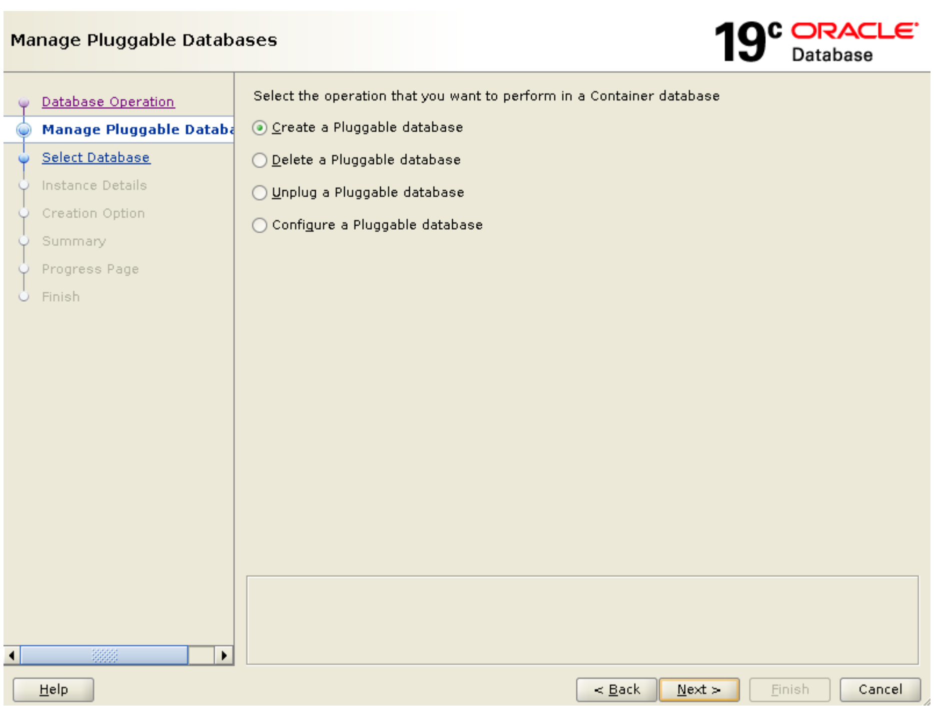


Illustration 10: Create a Pluggable database

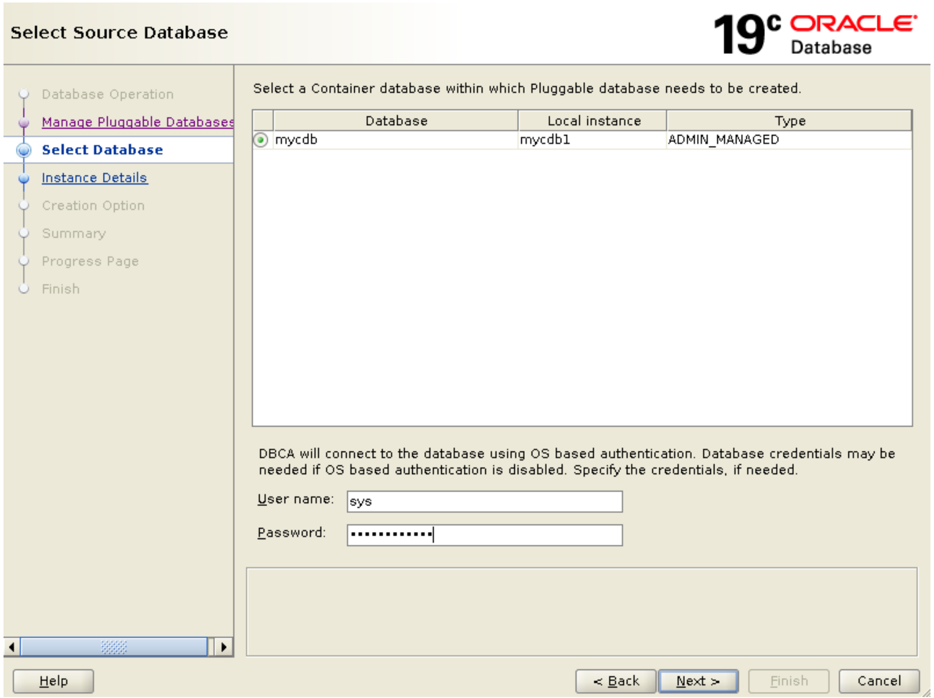


Illustration 11: Select Source Database

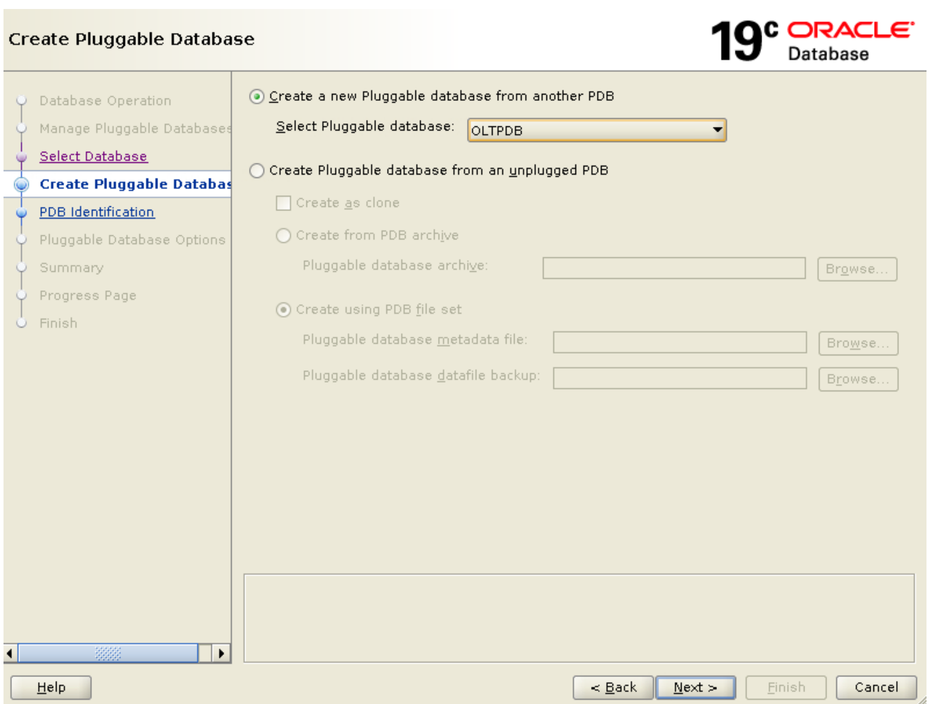


Illustration 12: Create a new Pluggable database from another PDB

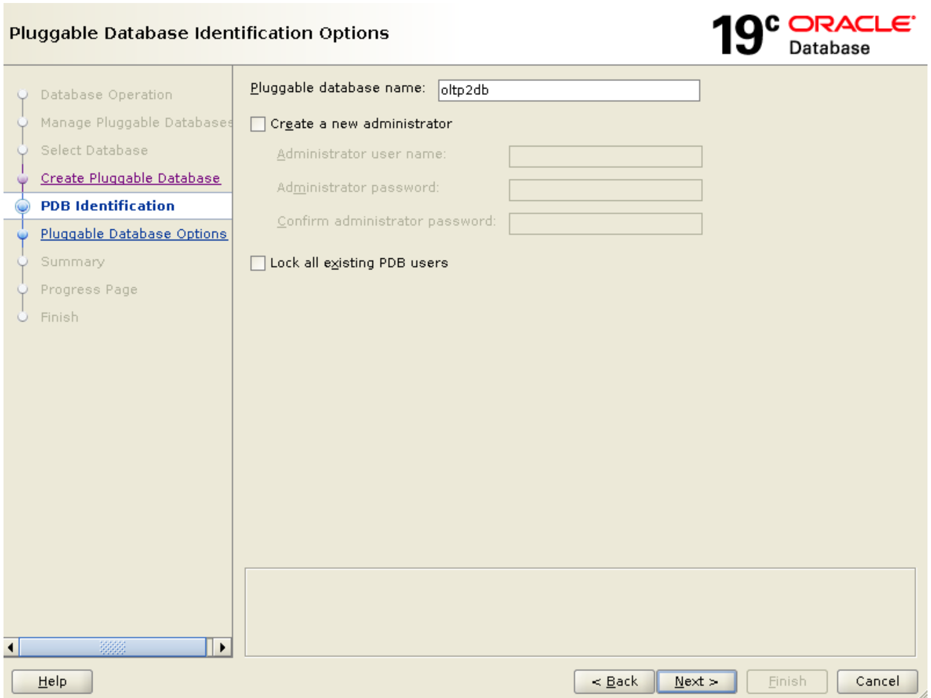


Illustration 13: Set new Pluggable database name

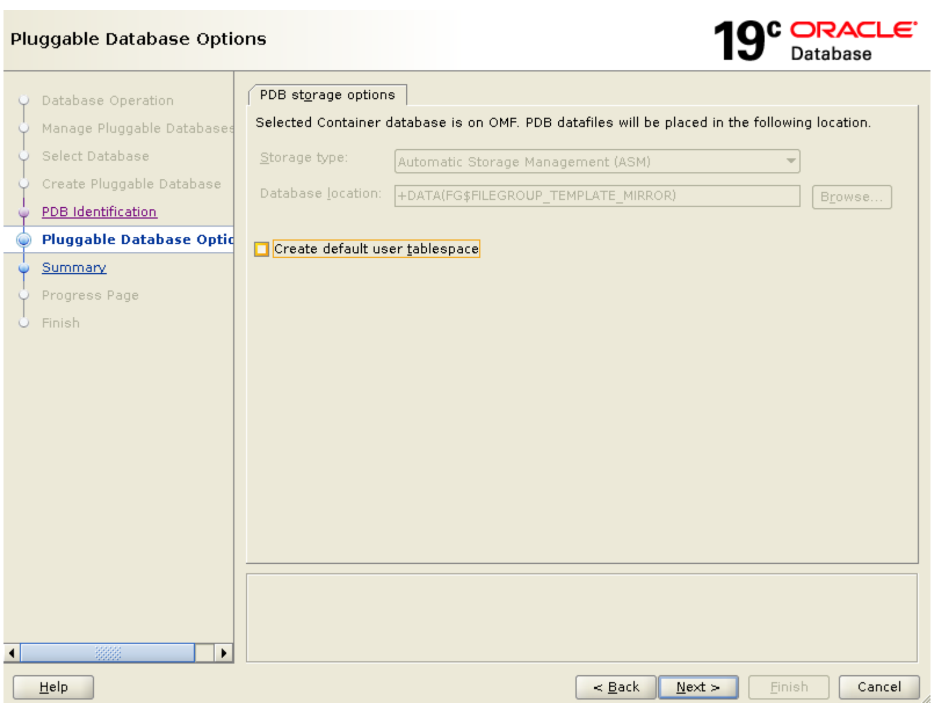


Illustration 14: Pluggable Database Options screen

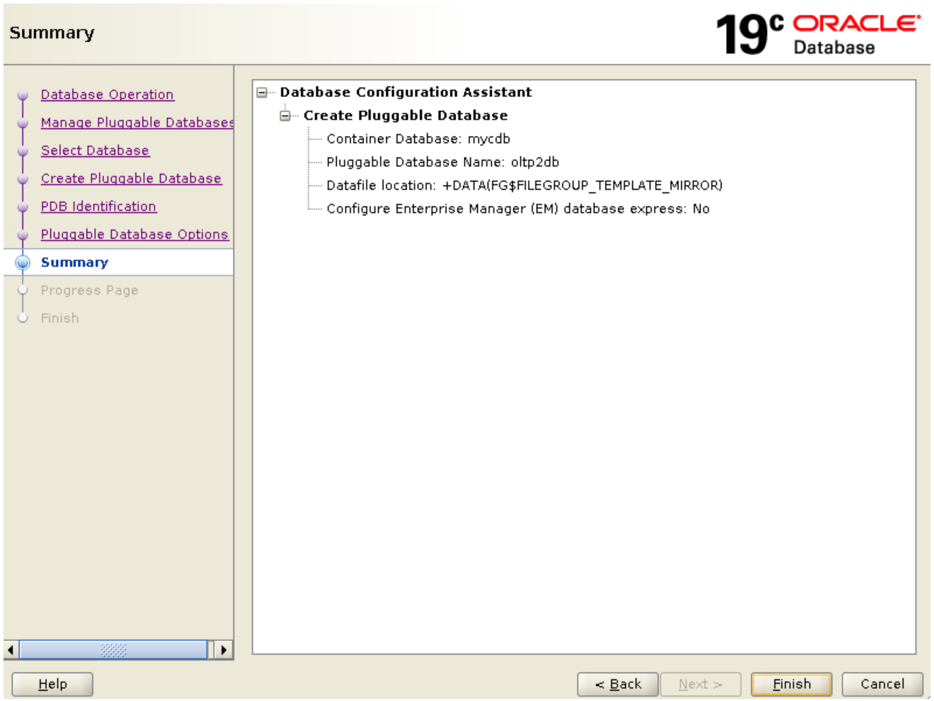


Illustration 15: Summary screen

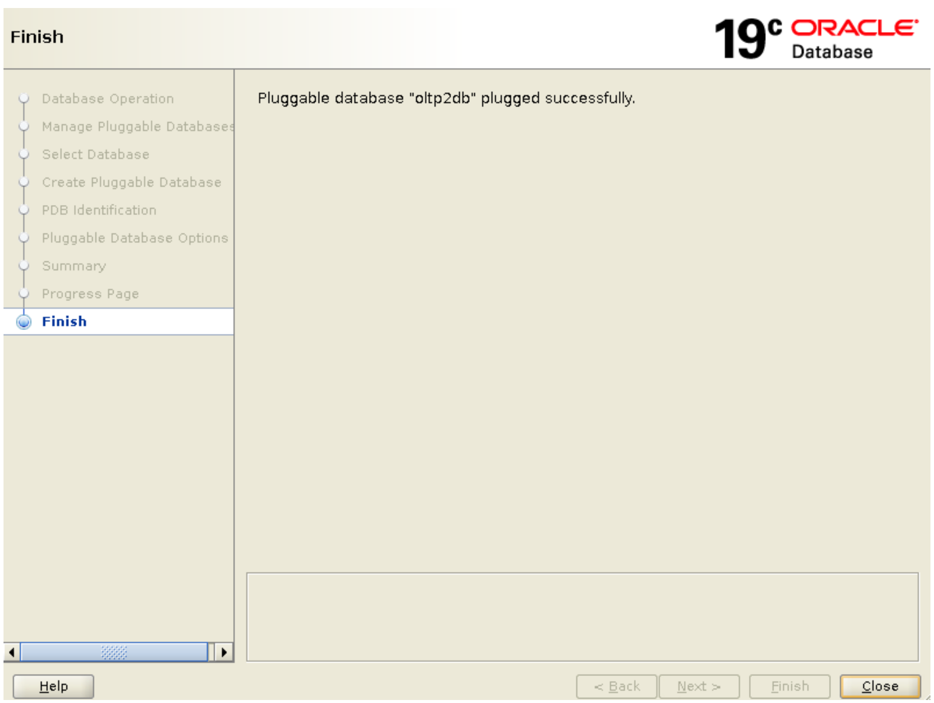


Illustration 16: Finish screen

Appendix E – Swingbench test using 2 SOE schemas

1. Enable all CPU cores via odacli update-cpucore -c 64

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

   from
   my $cdb_service="oltpdb.domain.com";
   to
   my $cdb_service="oltp2db.domain.com";

3. 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/oltp2db.domain.com/mycdb2</ConnectString>
   ```

4. Run loadgen.pl on both machines

   perl loadgen.pl -u 900

References

Oracle Database Appliance X10-HA Data Sheet

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

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: February 2024

Author: RACPack, Cloud Innovation & Solution Engineering Team

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