Executive Summary ............................................................................................................. 2
Audience ............................................................................................................................ 3
Objectives .......................................................................................................................... 3
Oracle Database Appliance performance architecture ................................................. 4
Oracle Database Appliance configuration templates ...................................................... 5
Oracle Database Appliance X5-2 Tested Configuration .................................................. 6
What is Swingbench? ........................................................................................................... 6
Swingbench download and setup ....................................................................................... 7
Benchmark Setup ............................................................................................................... 8
  Database and schema setup ............................................................................................ 8
  Swingbench workload setup ......................................................................................... 15
  OE workload testing performance results ..................................................................... 15
  Database and Operating System statistics .................................................................. 16
Conclusion .......................................................................................................................... 22
Appendix B: Swingbench configuration file ................................................................. 24
Appendix C: Script to issue workload and collect database statistics 27
Executive Summary

Oracle Database Appliance X5-2 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, 6-U configuration. The hardware and software configuration of Oracle Database Appliance provides redundancy and protects against all single points of failures within the system.

Specifically, Oracle Database Appliance is a two node, Intel X-86 64-bit cluster with direct attached SAS storage that consists of Hard Disk Drives and Solid State Disks. It runs standard, time-tested software components, such as Oracle Linux operating system (OS), Oracle Relational Database Management System (RDBMS), Oracle Real Application Clusters software (RAC), Oracle Clusterware, and Oracle Automatic Storage Management (ASM). The pre-built, pre-tested, and pre-tuned configuration ensures that Oracle Database Appliance can be deployed rapidly, often on the day when it arrives at a customer site, and does not require typical efforts to manually tune the configuration. Further, Oracle Database Appliance includes Oracle Appliance Manager Software for managing and maintaining the system, including patching, upgrades, and troubleshooting.

The purpose of this white paper is to demonstrate and document the performance of simulated workload (using Swingbench, a freely available performance testing tool) executing on Oracle Database Appliance. System architects and database administrators can evaluate and compare the performance of this standardized workload executing on Oracle Database Appliance vis-à-vis their legacy environment. It is not a purpose of this white paper to demonstrate the maximum IOPS and MBPS capacity of Oracle Database Appliance.

Despite its small size, Oracle Database Appliance is a very powerful database server. During the performance testing conducted as part of running this benchmark, Oracle Database Appliance demonstrated scalable performance for database workloads. In a
configuration with all CPU cores activated, Oracle Database Appliance supported more than 30,000 concurrent Swingbench users, and more than 31,200 Swingbench transactions per second while maintaining an average response time of less than 19 milliseconds.

Audience

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

Objectives

A quick review of the hardware configuration of Oracle Database Appliance will show that the architecture of the system is built for high availability and good out-of-the-box performance. However, due to the 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 after they would migrate their databases to a new environment.

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

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

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

This paper is the result of extensive testing carried out with the above standard workload while varying the user and transaction volumes in different configurations of the Oracle Database Appliance. Customers can run these same Swingbench workloads on their legacy
Evaluating and Comparing Oracle Database Appliance X5-2 Performance

systems and compare with corresponding results obtained and outlined for the Oracle Database Appliance 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.

Note: If an Oracle Database Appliance system is available to you at the time of comparison, or if you want to use a different workload, then for a more complete and specific comparison rather than relying on the numbers in this white paper, you can certainly run the same workload yourself in both Oracle Database Appliance environment and your legacy non-Oracle Database Appliance environments to compare results, using the approach outlined above.

Oracle Database Appliance performance architecture

The testing for this white paper was performed on an Oracle Database Appliance X5-2 system. Oracle Database Appliance X5-2 System consists of two X86 servers, each with two 18-core Intel E5-2699 v3 CPUs and 256GB of memory (expandable to up to 768GB per node). The two nodes use direct attached storage that is comprised of sixteen 4 TB, 7,200 rpm SAS hard disk drives, four 400GB Medium Endurance and four 200 GB High Endurance SAS solid state disks (SSDs). Thus, the Oracle Database Appliance X5-2 provides a total of up to 1,536 GB of memory, 72 CPU cores, 64 TB of raw HDD storage, 1600 GB of SSD storage for frequently accessed data and 800GB of SSD storage for database REDO logs. With the addition of the Oracle Database Appliance Storage Expansion Shelf, the available storage can be doubled.

The disks are connected to the two server nodes via redundant I/O Controllers and redundant HBAs. The direct attached storage, while limited in size, provides possible performance benefits as compared to shared SAN storage environments.

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

By default, Oracle Database Appliance storage is configured for performance and is database aware. Database REDO logs and frequently accessed data are hosted on SSDs and data is hosted on the relatively faster outer cylinders of HDDs while the inner cylinders of HDDs host the backup and recovery files (ASM RECO Disk Group). Further, on Oracle Database Appliance and enhanced implementation of ASM (enumerated or fixed ASM partnership) is used to reduce disk rebalance completion time.
On Oracle Database Appliance, server node redundancy is provided by Oracle Real Application Clusters (RAC). Two dedicated and redundant, *Infiniband* channels provide robust and reliable high speed cluster connectivity within Oracle Database Appliance. In case of a single component or node failure, Oracle Database Appliance continues to operate and provide service to the application.

Oracle Database Appliance also provides multiple 10-GbE interfaces for user and application access. These network channels can be used for different purposes. For example, customers may use separate dedicated networks or VLANs for their applications, disaster recovery, backup and recovery, and management networks. The Infiniband interconnect provides 40 Gbps per port. Oracle Database Appliance X5-2 supports exafusion Direct to Wire OLTP protocol from 12.1.0.2.0 onwards which further improves overall performance.

**Oracle Database Appliance configuration templates**

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

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

<table>
<thead>
<tr>
<th>Template</th>
<th>CPU Count</th>
<th>SGA (Gb)</th>
<th>PGA (Gb)</th>
<th>Flash (Gb)</th>
<th>Log Buffer Size (Mb)</th>
<th>Processes</th>
<th>Redo Log Size (Gb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Odb-01s</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>16</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>Odb-01</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>12</td>
<td>16</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>Odb-02</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>24</td>
<td>16</td>
<td>400</td>
<td>1</td>
</tr>
<tr>
<td>Odb-04</td>
<td>4</td>
<td>16</td>
<td>8</td>
<td>48</td>
<td>32</td>
<td>800</td>
<td>1</td>
</tr>
<tr>
<td>Odb-06</td>
<td>6</td>
<td>24</td>
<td>12</td>
<td>72</td>
<td>64</td>
<td>1200</td>
<td>2</td>
</tr>
<tr>
<td>Odb-12</td>
<td>12</td>
<td>48</td>
<td>24</td>
<td>144</td>
<td>64</td>
<td>2400</td>
<td>4</td>
</tr>
<tr>
<td>Odb-16</td>
<td>16</td>
<td>64</td>
<td>32</td>
<td>192</td>
<td>64</td>
<td>3200</td>
<td>4</td>
</tr>
<tr>
<td>Odb-24</td>
<td>24</td>
<td>96</td>
<td>48</td>
<td>192</td>
<td>64</td>
<td>4800</td>
<td>4</td>
</tr>
<tr>
<td>Odb-32</td>
<td>32</td>
<td>128</td>
<td>64</td>
<td>192</td>
<td>64</td>
<td>6400</td>
<td>4</td>
</tr>
<tr>
<td>Odb-36</td>
<td>36</td>
<td>128</td>
<td>64</td>
<td>192</td>
<td>64</td>
<td>7200</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table 1: Oracle Database Appliance Sizing Templates (OLTP) for Oracle Database Appliance X5-2**

In order to perform a meaningful performance comparison, the database configuration on Oracle Database Appliance and on the non-Oracle Database Appliance environment should match closely. While the above matrix relates to the database sizing templates
available on Oracle Database Appliance, it can also be used to manually establish a corresponding database configuration in the non-Oracle Database Appliance environment before executing the test workload.

In terms of data processing measurements, Oracle Database Appliance standard performance ratings are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Odb-01s</th>
<th>Odb-03</th>
<th>Odb-04</th>
<th>Odb-06</th>
<th>Odb-12</th>
<th>Odb-16</th>
<th>Odb-24</th>
<th>Odb-32</th>
<th>Odb-36</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One X5-2 storage shelf</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO Per Second (HDD/Flash)</td>
<td>42/4167</td>
<td>42/4167</td>
<td>83/8333</td>
<td>167/16700</td>
<td>250/25000</td>
<td>500/50000</td>
<td>750/75000</td>
<td>1500/150k</td>
<td>1500/150k</td>
</tr>
<tr>
<td>MB Per Second (HDD/Flash)</td>
<td>83/83</td>
<td>83/83</td>
<td>167/167</td>
<td>333/333</td>
<td>500/500</td>
<td>1000/1000</td>
<td>1500/1500</td>
<td>3000/3000</td>
<td>3000/3000</td>
</tr>
<tr>
<td>Log Generation Rate (MB/Sec)</td>
<td>6.83</td>
<td>6.83</td>
<td>6.83</td>
<td>13.65</td>
<td>27.30</td>
<td>27.30</td>
<td>27.30</td>
<td>27.30</td>
<td>27.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Odb-03</th>
<th>Odb-04</th>
<th>Odb-06</th>
<th>Odb-12</th>
<th>Odb-16</th>
<th>Odb-24</th>
<th>Odb-32</th>
<th>Odb-36</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Two X5-2 storage shelves</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO Per Second (HDD/Flash)</td>
<td>83/8333</td>
<td>83/8333</td>
<td>167/16700</td>
<td>333/33300</td>
<td>500/50000</td>
<td>1000/1000</td>
<td>2000/2000</td>
<td>3000/3000</td>
</tr>
<tr>
<td>MB Per Second (HDD/Flash)</td>
<td>167/167</td>
<td>167/167</td>
<td>333/333</td>
<td>667/667</td>
<td>1000/1000</td>
<td>2000/2000</td>
<td>3000/3000</td>
<td>6000/6000</td>
</tr>
<tr>
<td>Log Generation Rate (MB/Sec)</td>
<td>6.83</td>
<td>6.83</td>
<td>6.83</td>
<td>13.65</td>
<td>27.30</td>
<td>27.30</td>
<td>27.30</td>
<td>27.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Odb-03</th>
<th>Odb-04</th>
<th>Odb-06</th>
<th>Odb-12</th>
<th>Odb-16</th>
<th>Odb-24</th>
<th>Odb-32</th>
<th>Odb-36</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 2: Oracle Database Appliance database sizing model based on performance characteristics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Oracle Database Appliance X5-2 Tested Configuration**

For the purposes of these performance tests, the Oracle Database Appliance X5-2 was configured with a storage expansions shelf. This increased the storage capacity to thirty-two 4 TB, 7,200 rpm SAS hard disk drives, eight 400GB Medium Endurance and eight 200 GB High Endurance SAS solid state disks (SSDs). The additional storage shelf doubles the IO capacity of the system, which doubled the IO performance of the system, and has a direct effect on performance as the workload is IO limited.

The database for the tests was configured such that hot objects such as the undo tablespace were placed in FLASH. This ensured optimal performance for this workload, but may not be applicable for all workloads.

**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](http://dominicgiles.com/downloads.html)
Swingbench version 2.5.0.956 was used to perform the testing documented in this white paper. For more information about Swingbench, please refer to Swingbench documentation available at http://www.dominicgiles.com/swingbench.html

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

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

Swingbench download and setup

The Swingbench tool can be setup in a Windows or Linux environment. A Linux environment was used during the testing conducted during the course of the writing of this white paper. It is recommended that Swingbench software be installed on a separate machine that has local network connectivity to the target system where the system to be evaluated would be running its databases. Executing Swingbench client from the target system itself can affect performance measurements and produce undesirable and inaccurate 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 available is 2.5.0.956 and this version was used during the course of testing. You should download 2.5.0.956 or a later version on to the machine where you plan to run the workload from.

2. Unzip the downloaded file. Replace the <path>/bin/swingconfig.xml file with the swingconfig.xml supplied in the Appendix B of this Paper. Note that Appendix B of this white paper provides two Swingbench configuration files that were used respectively for Order Entry and Sales History tests covered in this paper. You can pick the one that you need.

3. Specify non-blocking random number generators
Oracle 12c JDBC driver requires a random number for encrypting connect string. By default this random number is generated from /dev/random. In certain situations, when using /dev/random the random number generation may be blocked and remain hung for an extended period due to a system entropy based algorithm used for the random number generation process. You can use /dev/urandom instead of /dev/random to address this issue. This change can be made by specifying /dev/urandom (in the <swing benchhome>/launcher/launcher.xml file) in the arguments when launching Swingbench as follows:

**Old Entry**

```xml
<jvmargset id="base.jvm.args">
  <jvmarg line="-Xmx1024m"/>
  <jvmarg line="-Xms512m"/>
  <!--<jvmarg line="-Djava.util.logging.config.file=log.properties"/>-->
</jvmargset>
```

**New Entry**

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

**Benchmark Setup**

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

**Database and schema setup**

The tests outlined in this document were performed with Oracle Appliance Manager (OAK) 12.1.2.2.0 image running Oracle Database release 12.1.0.2.2. When comparing the performance of Oracle Database Appliance against a non-Oracle Database Appliance environment, these tests can be run on other software versions in the non-Oracle Database Appliance environments, if necessary. The database configured on Oracle Database Appliance uses standard templates and standard, optimized parameter settings. Note that the corresponding settings for these parameters may not match their settings on legacy systems. In order to perform proper comparison, configuration parameters for the
Evaluating and Comparing Oracle Database Appliance X5-2 Performance

database configured in the legacy environment, should be similar to those in Oracle Database Appliance environments as much as possible.

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

Database setup
While Oracle Database Appliance allows you to create Oracle Real Application Clusters (RAC), RAC One (active-passive), and Single Instance Enterprise Edition databases, during the course of performance tests, only Oracle RAC configuration was used. Thus both server nodes within Oracle Database Appliance were actively running an Oracle database instance and servicing database workload requests.

The database was setup using the OLTP template for the Odb-24 configuration. The OLTP templates automatically configure the SGA and PGA in the ratio of 2:1.

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

1. **Reset CPU_COUNT**

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

   SQL> alter system reset CPU_COUNT scope=spfile;

2. **Disable database auditing**

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

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

3. **Set higher rollback segment count and tune UNDO**

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

```
SQL> alter system set "_rollback_segment_count"=12000 scope=spfile sid='*';
SQL> alter system set "_undo_autotune"=false scope=spfile sid='*';
SQL> alter system set undo_retention=180 scope=spfile sid='*';
```

4. Set the “_buffer_busy_wait_timeout” parameter to 2

Database performance related bug 13344323 should be addressed. This bug is related to the possibility of a missing post from DBWR that can cause buffer busy waits. As a workaround the _buffer_busy_wait_timeout parameter should be set to 20ms (2).

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

5. Create additional REDO log groups

By default, only two redo log groups per instance are created at the time of initial database setup on Oracle Database Appliance. For high performance applications, this default number of redo log groups may be inadequate and can cause performance bottlenecks. This may result in errors such as – “Thread ‘n’ cannot allocate new log”, “Checkpoint not complete” or the free buffer wait event, etc. Therefore, for performance intensive workloads, additional redo log groups should be created. For the testing conducted during this project, a total of 5 redo log groups of 8 GB each per instance were created on Oracle Database Appliance.

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

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

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

7. Increase PROCESSES parameter value.

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

8. Disable incremental checkpoint

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

```
SQL> alter system reset fast_start_mttr_target scope=spfile sid='*';
```
9. Increase the LMS processes

The default value of the gcs_server_processes is adequate in most of the cases. However, as the number of users increases, the gcs_server_processes can become a bottleneck. For a 30000 user workload, the gcs_server_processes parameter was increased to 12.

```
SQL> alter system set gcs_server_processes=12 scope=spfile sid='*';
```

10. Adjust PGA settings

PGA Memory allocation was reduced to 12GB.

```
SQL>alter system pga_aggregate_target=12G scope=spfile sid='*';
```

11. Disable the Resource Manager Plan

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

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

Schema setup

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

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

```
$ cd /tmp/swingbench/bin
$./oewizard
```
Evaluating and Comparing Oracle Database Appliance X5-2 Performance

Welcome to the Order Entry Install Wizard

This wizard will walk you through the steps to install a schema for the order entry benchmark. You will need a login with DBA privileges to create the needed tablespace, users, tables etc.

Author: Dominic Giles
Email: dominic.giles@oracle.com

Select Benchmark Version

There are two version of Swingbench. Version 1 is included for completeness it is recommended that you use version 2.

- Version 1.0
- Version 2.0 (Recommended)

Select Task

- Create the Order Entry Schema (User, Tables, Indexes, Data etc)
- Drop the Order Entry Schema

Database Details

Please enter the connect string for the database and DBA details. (Note: use "system" as the user at the end of the install)
Evaluating and Comparing Oracle Database Appliance X5-2 Performance

Schema Details

Please enter the details of the schema you wish to create; this will contain all of the tables and indexes for the order entry benchmark.

- Username: sue
- Password: sue
- Schema's Tablespace: SDE
- Tablespace's Datafile: /u01/oracle/app/oracle/sde/datafile/01_sde_sue_b47397fkd.../n

Database Options

Select the options you would like used during the creation of the schema. Certain options are additionally licensed by Oracle and others are only available on hardware such as Exadata. The recommended values are set default.

- Replication Mode:
  - Hash Partitioning
  - Compression Used
  - No Compression

- Tablespace Type:
  - Normal Tablespace

- Indexing Method:
  - All Indexes

Sizing Details

Select one of the preconfigured sizes for the benchmark. Or specify a scaling factor of 1 = 1GB. Based on the size of your Buffer Cache we recommend a schema size of 2.3 GB for a CPU intensive workload or minimum of 109.4 GB for a more I/O intensive workload.

- 1 GB
- 10 GB
- 100 GB
- 1 TB
- User Defined Scale: 109.4

Order/Entry tablespace size = 1.2 GB

All Details Entered

Please press the "Finish" button to begin schema creation. This may take while.

This schema creation will use the following level of parallelism based on the number of cpus believed to be present on the system (CPU*64). Feel free to change it if you think it isn't correct.

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

```
SQL> drop index "SOE"."CUST_LNAME_IX";
SQL> drop index "SOE"."CUST_EMAIL_IX";
SQL> drop index "SOE"."ITEM_PRODUCT_IX";
SQL> drop index "SOE"."CUST_ACCOUNT_MANAGER_IX";
SQL> drop index "SOE"."ORD_CUSTOMER_IX";
SQL> drop index "SOE"."ORD_ORDER_DATE_IX";
SQL> drop index "SOE"."CUST_UPPER_NAME_IX";
```

As noted earlier, the test database was initially created using the Odb-24 template. When the database configurations were subsequently changed to fewer cores, the other database parameters such as SGA size, etc. were not notched down. This decision was made to ensure that the measurements generate are fair for users who may simply want to limit the CPU configuration on Oracle Database Appliance and utilize other resources without any limitations.

*Note: On Oracle Database Appliance you can benefit from the pay-as-you-grow approach to software licensing. Fewer CPU licenses however do not restrict use of memory, storage, and network resources.*

As part of this testing, six different CPU configurations were tested by enabling a given number of cores (8, 16, 32, 48, and 72) at a time. Note that because of the use of hyper-threading within standard Oracle Database Appliance configuration, CPU performance is further improved.
Swingbench workload setup

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

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Count</td>
<td>Various (4250; 8500; 16000, 24000, 30000)</td>
</tr>
<tr>
<td>Think Time</td>
<td>InterMin=850ms; InterMax=1050ms</td>
</tr>
<tr>
<td>Run Time</td>
<td>50 minutes</td>
</tr>
</tbody>
</table>

Table 3: Swingbench OE workload configuration settings

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

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

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

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

For a given user volume, multiple ‘charbench’ sessions were invoked to execute the workload in parallel sessions. Charbench is the character (command line) interface for invoking Swingbench. A close to realistic, constant think time of between 850ms and 1050ms was used on the client side for all test scenarios.

OE workload testing performance results

The test results for the Order Entry workload execution highlight the following observations. The test results from different rounds of testing in various configurations are summarized below. The total duration for each test was 45 minutes (2700 seconds).
<table>
<thead>
<tr>
<th>Cores</th>
<th>User Volume</th>
<th>Customer Registration</th>
<th>Process Orders</th>
<th>Browse Products</th>
<th>Order Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 cores</td>
<td>4250 users</td>
<td>667401</td>
<td>1669759</td>
<td>4006828</td>
<td>1667717</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 cores</td>
<td>8500 users</td>
<td>1336616</td>
<td>3342208</td>
<td>8013888</td>
<td>3340846</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 cores</td>
<td>16000 users</td>
<td>2515473</td>
<td>6284675</td>
<td>15078217</td>
<td>6287797</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48 cores</td>
<td>24000 users</td>
<td>3761449</td>
<td>9406514</td>
<td>22582022</td>
<td>9409531</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72 Cores</td>
<td>30000 Users</td>
<td>4683200</td>
<td>11706524</td>
<td>28097260</td>
<td>11706832</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Swingbench workload benchmark summary

1. A 30,000 Swingbench user workload was successfully executed on a fully sized Oracle Database Appliance while maintaining transaction response times under 19ms. Higher user volumes could be accommodated if higher transaction response time is allowed.

2. Average transaction per second rate of more than 31,200 was achieved during testing.

3. The maximum number of users and number of transactions for the given performance level scaled in a highly correlated and scalable manner as configurations were changed from 8 core configuration to a 72 core configuration and user volumes were increased along with transaction volumes.

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

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

<table>
<thead>
<tr>
<th>Cores</th>
<th>Average CPU Utilization (Appliance wide)</th>
<th>Average Busy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>usr</td>
<td>sys</td>
</tr>
<tr>
<td>8</td>
<td>56.75</td>
<td>12.65</td>
</tr>
<tr>
<td>16</td>
<td>54.55</td>
<td>10.55</td>
</tr>
<tr>
<td>32</td>
<td>54.65</td>
<td>8.75</td>
</tr>
<tr>
<td>48</td>
<td>55.1</td>
<td>8.35</td>
</tr>
<tr>
<td>72</td>
<td>47.05</td>
<td>7.3</td>
</tr>
</tbody>
</table>

Table 5: Average CPU Utilization under different workloads

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

![Total CPU Usage Each Run](image)

Figure 3: Relative CPU Utilization in Different Configurations

*Note: Core measures in the above graph relates to the five different core configurations that were tested as outlined in Table 5.*

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

<table>
<thead>
<tr>
<th>Statistics</th>
<th>8 Cores</th>
<th>16 Cores</th>
<th>32 Cores</th>
<th>48 Cores</th>
<th>72 Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Generation MB Per Sec</td>
<td>10.2</td>
<td>20.75</td>
<td>36.59</td>
<td>52.16</td>
<td>64.53</td>
</tr>
<tr>
<td>Reads Per Sec</td>
<td>46.82</td>
<td>22.3</td>
<td>32.88</td>
<td>2.51</td>
<td>16.9</td>
</tr>
<tr>
<td>Writes Per Sec</td>
<td>229.03</td>
<td>401.13</td>
<td>684.57</td>
<td>849.07</td>
<td>984.73</td>
</tr>
<tr>
<td>Interconnect Traffic MBPS</td>
<td>113.95</td>
<td>225.50</td>
<td>403.05</td>
<td>624.78</td>
<td>766.62</td>
</tr>
</tbody>
</table>

**Table 6: Performance observations during OE workload execution**

Although the Order Entry workload is designed to introduce contention, yet the top wait events were observed to be well within acceptable range. Even with high interconnect traffic, the system was able to scale very well with added load. In addition, the work load scaled almost linearly as CPU capacity was added to the configuration.

<table>
<thead>
<tr>
<th>Cores</th>
<th>TPS</th>
<th>Users</th>
<th>Redo MBPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>4451</td>
<td>4250</td>
<td>6.74</td>
</tr>
<tr>
<td>16</td>
<td>8907</td>
<td>8500</td>
<td>15.77</td>
</tr>
<tr>
<td>32</td>
<td>16759</td>
<td>16000</td>
<td>27.86</td>
</tr>
<tr>
<td>48</td>
<td>25088</td>
<td>24000</td>
<td>35.12</td>
</tr>
<tr>
<td>72</td>
<td>31218</td>
<td>30000</td>
<td>44.10</td>
</tr>
</tbody>
</table>

**Table 7: CPU core count vs transaction per second and user count correlation**

Graph below illustrated the linearity in a pictorial format.
The correlation with user population is also linear as evident from the data above.
Troubleshooting performance mismatch between traditional and Oracle Database Appliance deployments

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

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

   Oracle Database Appliance specific database creation templates provide pre-optimized and proportional database parameter settings for databases created on Oracle Database Appliance. It is recommended that you use Oracle Appliance Manager command-line interface (oakcli) to create databases on Oracle Database Appliance.

2. **Verify that workload run is identical**

   Needless to say, apples to apples comparison should be performed when comparing performance across two environments. If you are running different workloads (SQL, commit rates, etc.) in the legacy versus the Oracle Database Appliance environment, then same performance levels cannot be expected.

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

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

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

4. **Verify system resource allocation**

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

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

6. **Verify database resource allocation**

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

7. **Verify database parameters**

Do not use performance inhibiting database parameters on Oracle Database Appliance. If you migrated your database from your legacy environment to Oracle Database Appliance, make sure you are not using obsolete, un-optimized settings for parameters. Use the “orachk” tool to verify your database configuration running on Oracle Database Appliance.

8. **Account for additional benefits Oracle Database Appliance provides**

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

Oracle Database Appliance supports and provides excellent performance for typical database workloads. When the full Oracle Database Appliance configuration, including an expansion shelf, was used, with all cores active to support a very large database configuration, the system was able to support a 30000 user Swingbench Order Entry workload while providing sustained commit rate of more than 31200 transactions per second. The testing conducted during the course of writing of this white paper indicated that some configuration adjustments and tuning were needed for different workloads to obtain maximum performance from the Oracle Database Appliance.
Appendix A: Post database deployment script

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

```sql
create bigfile tablespace soe datafile '+data' size 30g autoextend on maxsize unlimited uniform size 1m segment space management auto
/
begin
  for i in (select file_name from dba_data_files where tablespace_name
in (select tablespace_name from dba_tablespaces
  where contents = 'UNDO' )
  ) loop
    execute immediate 'alter database datafile '''||i.file_name||''' resize 18G';
  end loop;
end;
/
REM ODA by default is configured only with 2 redo log groups.
REM Adding an additional log group prior to the testing.

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

Note: For higher workloads, consider creating the redo log of size 8GB.
Appendix B: Swingbench configuration file

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

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SwingBenchConfiguration
xmlns="http://www.dominicgiles.com/swingbench/config">
  <Name>"Order Entry (PLSQL) V2"</Name>
  <Connection>
    <UserName>soe</UserName>
    <Password>soe</Password>
    <ConnectString>\hostname-scan/odadb</ConnectString>
    <DriverType>Oracle jdbc Driver</DriverType>
    <ConnectionPooling>
      <PooledInitialLimit>1000</PooledInitialLimit>
      <PooledMinLimit>1000</PooledMinLimit>
      <PooledMaxLimit>2500</PooledMaxLimit>
      <PooledInactivityTimeout>0</PooledInactivityTimeout>
      <PooledConnectionWaitTimeout>0</PooledConnectionWaitTimeout>
      <PooledPropertyCheckInterval>0</PooledPropertyCheckInterval>
      <PooledAbandonedConnectionTimeout>0</PooledAbandonedConnectionTimeout>
    </ConnectionPooling>
    <Properties>
      <Property Key="FetchSize">20</Property>
      <Property Key="StatementCaching">50</Property>
    </Properties>
  </Connection>
  <Load>
    <NumberOfUsers>1500</NumberOfUsers>
    <MinDelay>0</MinDelay>
    <MaxDelay>0</MaxDelay>
    <InterMinDelay>850</InterMinDelay>
    <InterMaxDelay>1050</InterMaxDelay>
    <QueryTimeout>120</QueryTimeout>
    <MaxTransactions>-1</MaxTransactions>
    <RunTime>0:50</RunTime>
    <LogonGroupCount>1</LogonGroupCount>
    <LogonDelay>1</LogonDelay>
    <LogOutPostTransaction>false</LogOutPostTransaction>
    <WaitTillAllLogon>true</WaitTillAllLogon>
    <StatsCollectionStart>0:15</StatsCollectionStart>
    <StatsCollectionEnd>0:45</StatsCollectionEnd>
    <TransactionList>
      <Transaction>
        <Id>Customer Registration</Id>
        <ShortName>NCR</ShortName>
      </Transaction>
    </TransactionList>
  </Load>
</SwingBenchConfiguration>
<ClassName>com.dom.benchmarking.swingbench.plsqltransactions.NewCustomerProcess</ClassName>
   <Weight>10</Weight>
   <Enabled>true</Enabled>
</Transaction>
<Transaction>
   <Id>Browse Products</Id>
   <ShortName>BP</ShortName>
</Transaction>

<ClassName>com.dom.benchmarking.swingbench.plsqltransactions.BrowseProducts</ClassName>
   <Weight>60</Weight>
   <Enabled>true</Enabled>
</Transaction>
<Transaction>
   <Id>Order Products</Id>
   <ShortName>OP</ShortName>
</Transaction>
</TransactionList>
<EnvironmentVariables>
   <Variable Key="SOE_NLSDATA_LOC">data/nls.txt</Variable>
   <Variable Key="SOE_NAMESDATA_LOC">data/names.txt</Variable>
</EnvironmentVariables>
</Load>
<Preferences>
   <StartMode>manual</StartMode>
   <Output>swingbench</Output>
   <JumpToEvents>true</JumpToEvents>
   <TimingsIncludeSleep>false</TimingsIncludeSleep>
   <TimingsIn>milliseconds</TimingsIn>
   <StatisticsLevel>simple</StatisticsLevel>
   <OutputFile>results.xml</OutputFile>
   <Charts DefaultChart="Overview">
      <Chart>
         <Name>DML Operations Per Minute</Name>
         <Autoscale>true</Autoscale>
         <MaximumValue>1.0</MaximumValue>
         <Logarithmic>false</Logarithmic>
      </Chart>
      <Chart>
         <Name>Transaction Response Time</Name>
         <Autoscale>true</Autoscale>
         <MaximumValue>-1.0</MaximumValue>
      </Chart>
   </Charts>
</Preferences>
<Logarithmic>true</Logarithmic>
</Chart>
<Chart>
    <Name>Transactions Per Minute</Name>
    <Autoscale>true</Autoscale>
    <MaximumValue>-1.0</MaximumValue>
    <Logarithmic>false</Logarithmic>
</Chart>
</Charts>
<AllowedErrorCodes>
    <ErrorCode Type="ORA">4063</ErrorCode>
</AllowedErrorCodes>
<RefreshRate>1</RefreshRate>
<OverviewCharts>
    <OverviewChart>
        <Name>Disk</Name>
        <MinimumValue>0.0</MinimumValue>
        <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
    <OverviewChart>
        <Name>CPU</Name>
        <MinimumValue>0.0</MinimumValue>
        <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
    <OverviewChart>
        <Name>Response Time</Name>
        <MinimumValue>0.0</MinimumValue>
        <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
    <OverviewChart>
        <Name>Transactions Per Second</Name>
        <MinimumValue>0.0</MinimumValue>
        <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
    <OverviewChart>
        <Name>Transactions Per Minute</Name>
        <MinimumValue>0.0</MinimumValue>
        <MaximumValue>2.147483647E9</MaximumValue>
    </OverviewChart>
</OverviewCharts>
Appendix C: Script to issue workload and collect database statistics

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

Execute the script as follows:

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

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

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

<table>
<thead>
<tr>
<th>Application Module</th>
<th>Txn Count</th>
<th>Avg Res Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Registration</td>
<td>125684</td>
<td>3.00</td>
</tr>
<tr>
<td>Process Orders</td>
<td>314146</td>
<td>3.00</td>
</tr>
<tr>
<td>Browse Production</td>
<td>754385</td>
<td>1.00</td>
</tr>
<tr>
<td>Order Products</td>
<td>313674</td>
<td>9.00</td>
</tr>
</tbody>
</table>

Exiting...

#!/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="hostname-scan.example.com";
my $service_name="odadb";
my $port=1521;
my $dbauser="system";
my $dbapwd="welcome1";
### Please modify the above variables as needed #######

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

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

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

my $sb_home;
use Cwd();
my $pwd = Cwd::cwd();
my $sb_output_dir=$pwd."/sb_out";
my $awr_dir=$pwd."/awr";

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

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

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

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

my $rc=$tot_uc;
print "User count $uc
n";
print "Total SB Sessions $cb_sess
n";
}

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

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

sub generate_awr_snap
{
    # my $dbh = DBI->connect("dbi:Oracle://$host:$port/$service_name","system","welcomel"
    ) || die "Database connection not made";
    my $dbh = DBI->connect("dbi:Oracle://$host:$port/$service_name","$dbauser","$dbapwd"
    ) || die "Database connection not made";
    $dbh->{RowCacheSize} = 100;
    my $sql = qq{ begin dbms_workload_repository.create_snapshot; end; };
my $sth = $dbh->prepare( $sql );

$sth->execute();
$sql = q{ select max(snap_id) from dba_hist_snapshot };
$sth = $dbh->prepare( $sql );
$sth->execute();
$sth->bind_columns( undef, \$snap_id );
$sth->fetch();
$sth->finish();
$dbh->disconnect();

sub process_xml_output {

my $txn_cnt;
my $avg_rt;
my @files;
my $cr_tc=0;
my $cr_to_rt=0;
my $po_tc=0;
my $po_to_rt=0;
my $bp_tc=0;
my $bp_to_rt=0;
my $op_tc=0;
my $op_to_rt=0;
my $num_users=0;
my $avg_tps=0;
my $app_module;
my $file;
my $xml;

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

foreach $file (@files) {

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

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

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

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

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

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

#printf "\n";

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

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

foreach $app_module (@app_modules) {
    if ($app_module eq "Customer Registration") {
        printf "%-22s %16s %0.2f\n", $app_module, $cr_tc, ($cr_to_rt/$cr_tc);
    }
    elsif ($app_module eq "Process Orders") {
        printf "%-22s %16s %0.2f\n", $app_module, $po_tc, ($po_to_rt/$po_tc);
    }
    elsif ($app_module eq "Browse Products") {
        printf "%-22s %16s %0.2f\n", $app_module, $bp_tc, ($bp_to_rt/$bp_tc);
    }
    elsif ($app_module eq "Order Products") {
        printf "%-22s %16s %0.2f\n", $app_module, $op_tc, ($op_to_rt/$op_tc);
    }
}
GetOptions(\%opts, 'users|u=i' => \$tot_uc, 'runid|r=i' => \$rundate,) or die usage;

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

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

my \$rc;

for($counter = 1; $counter <= $cb_sess; $counter++){
  \$rc = \$tot_uc - ($counter*$uc);
  if ( \$rc <= 0 ) { \n    \$uc = ($rc+$uc);
  }
  my \$thr = threads->create('process',\$counter);
  print "Charbench \".$\counter Starting with usercount \$uc\"."\n";
  $thr->detach();
  sleep 30;
}

sleep 400;
generate_awr_snap;

\$b_snap=$snap_id;
print "Start Snap \$b_snap"."\n";
sleep $awr_interval_in_secs;
generate_awr_snap;

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

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

my \$running;

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

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

Script to generate awr reports ..

-------------------------- genawr.sh -------------------------------

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

l_dbid=1267760840
l_start_snapid=$1
l_end_snapid=`expr $1 + 1`
1_end_snapid=$2;
1_runid=$3;
AWR_DIR=$4;

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

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

echo $l_awr_log_file;

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

1 awr_log_file="$AWR_DIR/awrrpt_2_{$1_start_snapid}_{$1_end_snapid}_{$1_runid}.log"

$ORACLE_HOME/bin/sqlplus -s system/welcome1@<hostname>-scan.example.com/odadb << EOC
set head off
set pages 0
set lines 132
set echo off
set feedback off
spool $1_awr_log_file
SELECT
output
FROM
TABLE
(dbms_workload_repository.awr_report_text($1_dbid,2,$1_start_snapid,$1_end_snapid));
spool off
exit;
EOC

-------------------------genawr.sh -------------------------