Configuring Oracle Database for OLTP on SPARC T5 Series Servers with the Oracle ZFS Storage Appliance as Primary Storage

An overview of configuration and best practice recommendations based on the tested reference architecture of Oracle Database with Real Application Clusters, SPARC T5-2 Server and Oracle Solaris, and Oracle ZFS Storage ZS3-4.
Introduction

Databases are the backbone of today’s business, whether providing transaction integrity for key business systems, such as payment engines, or providing the core analytical data for decision-making. These broad and varied uses require similarly varied resources and investment to meet different needs like potentially high input-output operations per second (IOPS), extreme throughput or extensive capacity requirements.

Unlike traditional solutions, Oracle's hardware and software solutions are engineered to work together. Combining this engineering with the extreme performance and market-leading efficiency inherent in the flexible architecture of the Oracle ZFS Storage Appliance produces the ideal platform for database consolidation.

The combined architecture of the Oracle ZFS Storage Appliance and Oracle Database with Oracle Real Application Clusters (Oracle RAC) makes it an ideally suited platform to provide the flexibility required for the ever-changing availability, capacity and performance requirements of online transaction processing, or OLTP. Using an Oracle Solaris-driven Oracle ZFS Storage Appliance can reduce the cost and complexity of managing data and storage devices for hosting your Oracle Database.

What is OLTP and why is it so demanding?

OLTP is a class of software programs capable of supporting transaction-oriented applications on the Internet.

Typically, OLTP systems are used for order entry, financial transactions, customer relationship management (CRM) and retail sales. Such systems have a large number of users who conduct short transactions. The database queries require subsecond response times and return relatively few records. These low latency/fast response OLTP workloads are very demanding on the architecture hardware.

This document provides planning information and step-by-step instructions for configuring and tuning both Oracle's SPARC T5 servers and an Oracle ZFS Storage Appliance in an OLTP-driven Oracle Database with Oracle RAC environment.
Architectural Overview of Tested Solution for Oracle Database with OLTP Application

The focus of this architecture is to provide a high throughput, low latency system that satisfies the transactional demands of an OLTP application-driven environment while offering solid cost efficiency relative to enterprise-level configurations.

This document discusses how the reference architecture – comprised of an Oracle ZFS Storage ZS3-4 cluster, SPARC T5-2 Server running the Oracle Solaris operating system, Oracle Database using Oracle Real Application Clusters, and InfiniBand network interconnect links – meets these OLTP environment requirements.

The Oracle ZFS Storage ZS3-4 for this architecture uses a two-node cluster configuration. Both nodes are identical in hardware configuration and are configured in an active-active mode of operation, which means both nodes actively provide data services to clients. If a catastrophic failure occurs on one node, the other node takes over the data services and network connections of the failed node.
An Oracle ZFS Storage ZS3-4 cluster with eight Oracle Storage Drive Enclosure DE2-24P disk shelves is used as the primary storage source of the Oracle Database with Oracle RAC. Four InfiniBand HCAs, with one port from each host channel adapter (HCA) set as active, is configured in each head to provide both optimal redundancy and superior network bandwidth. This easily allows for an aggregate of over 3 GB/sec on the IP network that is multipathed over the two InfiniBand interfaces, and provides network headroom if additional database nodes are added to the configuration.
Overview of System Components

The following tables describe the hardware configuration, operating systems, and software releases utilized by the reference architecture demonstrated in this white paper. Table 1 details the hardware used in the reference architecture.

<table>
<thead>
<tr>
<th>TABLE 1. HARDWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUIPMENT</td>
</tr>
<tr>
<td>Storage</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Network</td>
</tr>
<tr>
<td>Server</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 2 shows the software components used in the reference architecture.

<table>
<thead>
<tr>
<th>TABLE 2. SOFTWARE COMPONENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATING SYSTEM /SOFTWARE</td>
</tr>
<tr>
<td>Oracle Solaris 11 Version: 0.5.11</td>
</tr>
<tr>
<td>Branch: 0.175.1.18.0.4.2</td>
</tr>
<tr>
<td>Oracle ZFS Storage Appliance Software</td>
</tr>
</tbody>
</table>

Table 3 shows the database software used.

<table>
<thead>
<tr>
<th>TABLE 3. SOFTWARE USED IN REFERENCE ARCHITECTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFTWARE</td>
</tr>
<tr>
<td>Oracle Database 11gR2 with Oracle Real Application Clusters</td>
</tr>
<tr>
<td>Swingbench (load generator)</td>
</tr>
</tbody>
</table>
Best Practices for Configuring an Oracle Database System for OLTP

This section provides best practices and recommendations for configuring an Oracle Database with Real Application Clusters for OLTP on SPARC T5 servers running Oracle Solaris and using InfiniBand to connect to an Oracle ZFS Storage ZS3-4 cluster.

Configuring the Oracle ZFS Storage Appliance

The following configurations for the Oracle ZFS Storage Appliance are recommended to optimize performance for the OLTP environment with Oracle Database and the Oracle Real Application Clusters option.

Drive Tray, SSD and Disk Pools Requirements

AN OLTP database environment infrastructure produces high levels of random I/O patterns and requires the highest level of response time. Utilizing high performance, low latency storage is a top priority.

To meet these demands, use a **mirrored** data profile. This configuration offers the best I/O response time and produces reliable storage by dividing access and redundancy between two sets of disks.

In combination with write SSD’s log devices and the Oracle ZFS Storage Appliance’s hybrid storage pool architecture, this profile can produce a large amount of input/output operations per second (IOPS) to meet the high demands that are critical for OLTP workloads.

The recommended disk storage configuration for this reference architecture includes:

- A mirrored disk pool of 84 x 300/600 GB or 900 GB (10000k RPM performance disks) with at least sixteen 73 GB SSD devices for LogZilla working with a striped log profile.

- At least two 1.46 TB L2 cache (L2ARC) – Striped cache.

- A single storage pool on each controller. Each storage pool should be configured with half of the available hard disk drives (HDD) and write cache (SSD) in each Oracle Storage Enclosure DE2 disk shelf. This allows for maximum performance and redundancy. Read cache disks from each node should be assigned to each corresponding pool as well. This design will provide you with more storage resources as well as better I/O load balancing, performance, and throughput.
Network Settings

This section describes the necessary steps to configure the InfiniBand network HCA links in the Oracle ZFS Storage ZS3-4 to achieve maximum throughput and provide redundancy.

Because of InfiniBand's high bandwidth and low latency, InfiniBand HCAs are used exclusively. Four dual port QDR M2 InfiniBand HCA cards are used in each head of the Oracle ZFS Storage ZS3-4 cluster.

1. Configure each InfiniBand port to reside in the same partition.

Create a new network datalink for each IB (InfiniBand) port on both nodes. Assign the same partition key to each of the network datalinks. The figure 2 example uses the partition key ffff.

![Network Datalink](image)

Figure 2. Configuring an InfiniBand network datalink in the Oracle ZFS Storage Appliance BUI

2. Create the IB network interfaces.

Create a network interface for each of the IB datalinks created. Assign an IP address that resides within the IB fabric subnet. This is the IP address that the Oracle Database with RAC servers will use to access the storage using the direct NFS (dNFS) protocol. Figure 3 shows an example of IB port 0, configured on a 192.168.36.0/22 IP subnet.
Configuring Oracle Database for OLTP on SPARC T5 Series Servers with the Oracle ZFS Storage Appliance as Primary Storage

3. Configure each IB port to reside in the same IB partition.

Create a network interface for each of the IB datalinks. One port should be configured and enabled for each dual port HCA card. The other port is reserved for failover purposes should the other cluster head fail. Figure 4 illustrates how one port for each HCA is assigned an IP address and is active.

Figure 3. Network Interface configuration window in the Oracle ZFS Storage Appliance BUI

Figure 4. Network interface created on one port of each IB HCA
**Note:** IP Multipathing group (IPMP) is not enabled for this architecture as load balancing and path redundancy will be handled by Oracle dNFS and the oranfstab. Further details on creating the oranfstab are contained in the following section "Configuring the InfiniBand Ports into a Multipathing Group."

**File Share Configuration**

Multiple NFS filesystem shares are created across both Oracle ZFS Storage ZS3-4 controllers to ensure optimum performance. The following section details the file share creation and configuration process.

Each filesystem is created with varying ZFS record sizes and synchronous write bias based on the database file type, block size and I/O throughput requirements of the performance testing effort. Each filesystem should be created and mounted on the Oracle Solaris 11 nodes on the Oracle ZFS Storage ZS3-4 prior to the Oracle Database installation.

The filesystem share layouts are outlined in Table 4. The explanation, details and reasoning for creating multiple varying filesystems is described in detail in the "Oracle Database Layout" section later in the document.

**Database data file share creation and configuration**

Figure 5 details example settings for creating an NFS share for Oracle RAC data files on the Oracle ZFS Storage ZS3-4 through the browser user interface (BUI).

Figure 5. Database share configuration

Consider the following configuration recommendations for the data file shares in the Properties window:

- **Data compression:** Select the LZJB algorithm of data compression if compression is required.
  
  Before writing data to the storage pool, shares can optionally compress data utilizing different algorithms of compression. For further discussion on enabling compression, see the section "Performance Observations"
• **Cache device usage:** The ‘All data and metadata’ option is recommended. With this option, all files, LUNs, and any metadata will be cached. By default, all data sets make use of any read-optimized flash on the system. For some workloads that are write intensive but with little read operations, it may be advisable to ensure balanced usage of cache by disabling the caching. However, with an OLTP “mixed” workload, set the cache device usage to “All data and metadata”.

• **Synchronous write bias:** The Oracle ZFS Storage Appliance allows the user to determine how the storage array should handle synchronous writes at the share level. Storage administrators can optimize their configuration by utilizing expensive write-optimized flash for latency-sensitive workloads (Oracle REDO, for example) and bypassing the write-optimized flash for all other workloads. To provide fast response time, select the Latency option for all Redo and controlfile shares. Reference table 4 for Synchronous Write Bias settings for each share in the configuration.

• **Database record size:** Configure this setting according to the following table:

<table>
<thead>
<tr>
<th>FILE SYSTEM</th>
<th>ZFS RECORD SIZE</th>
<th>SYNCHRONOUS WRITE BIAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool1 (Controller1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/export/orcl/DATA1</td>
<td>8 KB</td>
<td>Throughput</td>
</tr>
<tr>
<td>/export/orcl/control</td>
<td>8 KB</td>
<td>Latency</td>
</tr>
<tr>
<td>/export/orcl/RECO1</td>
<td>1 MB</td>
<td>Latency</td>
</tr>
<tr>
<td>/export/orcl/orcl_orc</td>
<td>128k</td>
<td>Latency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pool2 (Controller2)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/export/orcl/DATA2</td>
<td>8 KB</td>
<td>Throughput</td>
</tr>
<tr>
<td>/export/orcl/control</td>
<td>8 KB</td>
<td>Latency</td>
</tr>
<tr>
<td>/export/orcl/RECO2</td>
<td>1 MB</td>
<td>Latency</td>
</tr>
</tbody>
</table>
Configuring the SPARC T5-2 Server and Oracle Solaris for Oracle Database with RAC

Using Oracle Database 11gR2 and the SPARC T5 series servers in conjunction with the Oracle Solaris 11 operating system achieved world record performance with the order entry database workload TPC-C benchmark. Naturally, the SPARC T5 series servers and Oracle Solaris 11 make an ideal fit when paired with the Oracle ZFS Storage ZS3-4 to achieve the fast response time required of an OLTP database infrastructure.

Configuring the Oracle Solaris 11 Operating System

The following sections detail many key performance tuning enhancements to take advantage of the Oracle Solaris 11 operating system's superior resource and memory management.

1. Configure memory allocation.

In order to define the default project for the Oracle system and the database administrator, add the following lines to file `/etc/user_attr`.

```
# vi /etc/user_attr
oracle::::project=user.oracle
```

Set the project resources for the oracle user.

```
# projmod -s -K "project.max-shm-ids=(priv,4096,deny)" user.oracle
# projmod -s -K "process.max-file-descriptor=(priv,131072,deny)" user.oracle
# projmod -s -K "process.max-sem-nsems=(privileged,1024,deny)" user.oracle
# projmod -s -K "process.max-sem-ops=(privileged,512,deny)" user.oracle
# projmod -s -K "project.max-msg-ids=(privileged,4096,deny)" user.oracle
# projmod -s -K "project.max-sem-ids=(privileged,65535,deny)" user.oracle
# projmod -s -K "project.max-tasks=(priv,131072,deny)" user.oracle
```

The `projmod` command should have successfully added the oracle user resources.

```
# vi /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.oracle:100::oracle::process.max-file-descriptor=(priv,131072,deny);process.max-sem-nsems=(priv,256,deny);project.max-sem-ids=(priv,100,deny);project.max-shm-ids=(priv,2048,deny);project.max-shm-memory=(privileged,500363689984,deny)
```
The kernel parameters used in the configuration are listed in Table 5. Use the Oracle Solaris 11 Resource Control Facility to set them. The values set should be no less than the recommended values listed in the table.

<table>
<thead>
<tr>
<th>RESOURCE CONTROL NAME</th>
<th>RECOMMENDED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-file-descriptor</td>
<td>131072</td>
</tr>
<tr>
<td>max-sem-nsems</td>
<td>1024</td>
</tr>
<tr>
<td>max-sem-ops</td>
<td>512</td>
</tr>
<tr>
<td>max-msg-ids</td>
<td>4096</td>
</tr>
<tr>
<td>max-sem-ids</td>
<td>65535</td>
</tr>
<tr>
<td>max-shm-ids</td>
<td>4096</td>
</tr>
<tr>
<td>max-tasks</td>
<td>131072</td>
</tr>
<tr>
<td>max-shm-memory</td>
<td>Greater than or equal to the database's system global area (SGA) size</td>
</tr>
<tr>
<td>zfs:zfs_arc_max</td>
<td>0x1000000000</td>
</tr>
</tbody>
</table>

2. Configure the NFS mounts.

Any server that accesses the Oracle ZFS Storage Appliance is considered a client. To configure the database server client NFS mounts, a target directory structure for access to the Oracle ZFS Storage Appliance is created and specific NFS mount options are assigned.

To reduce configuration complexity, Automatic Storage Management (ASM) diskgroups and disk pools were not configured in this reference architecture environment. Instead, NFS client shares are hard mounted and Direct NFS (dNFS) is configured to enhance performance. Enabling dNFS is described in a configuration step under the section "Configuring the InfiniBand ports into IP Multipathing Groups" that follows.

When using NFS mounts for data files, ensure there is sufficient disk space for the Oracle Grid Infrastructure on the Oracle ZFS Storage ZS3-4, the Oracle Database data files and any additional expansion of data files subsequently added to the database. The Oracle ZFS Storage ZS3-4 is ideal for a filesystem target because it has the ability to scale I/O throughput, processor performance, and storage capacity as storage needs change.

In order to use Oracle dNFS, the NFS filesystems must first be mounted and available over regular NFS mounts using Oracle’s recommended NFS mount options.
The following sections' instructions ensure the NFS filesystems exported from the Oracle ZFS Storage ZFS3-4 are properly mounted to ensure high performance and efficiency.

3. Create mount point directories.

Mount points must be created on all Oracle RAC nodes.

```bash
# mkdir -p /zfssa/orcl/DATA
# mkdir -p /zfssa/orcl/DATA2
# mkdir -p /zfssa/orcl/RECO
# mkdir -p /zfssa/orcl/RECO2
# mkdir -p /zfssa/orcl/control01
# mkdir -p /zfssa/orcl/control02
# chown oracle:oinstall /zfssa/orcl
```

4. Add entries to `vfstab`.

Entries must be added to each of the Oracle RAC nodes.

```bash
# vi /etc/vfstab
192.168.36.231:/export/orcl/DATA1 - /zfssa/orcl/DATA nfs - yes
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,proto=tcp,forcedirectio,actimeo=0,vers=3,suid
192.168.36.232:/export/orcl/RECO - /zfssa/orcl/RECO nfs - yes
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,proto=tcp,forcedirectio,actimeo=0,vers=3,suid
192.168.36.232:/export/orcl/RECO - /zfssa/orcl/RECO2 nfs - yes
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,proto=tcp,forcedirectio,actimeo=0,vers=3,suid
192.168.36.231:/export/orcl/control1 - /zfssa/orcl/control01 nfs - yes
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,proto=tcp,forcedirectio,actimeo=0,vers=3,suid
192.168.36.232:/export/orcl/control2 - /zfssa/orcl/control02 nfs - yes
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,proto=tcp,forcedirectio,actimeo=0,vers=3,suid
192.168.36.232:/export/orcl/DATA2 - /zfssa/orcl/DATA2 nfs - yes
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,proto=tcp,forcedirectio,actimeo=0,vers=3,suid
```
5. Create a mount point for the Cluster Ready Services (CRS) voting disk or Oracle Cluster Registry (OCR).

The Oracle Database requires the CRS voting disk and the OCR to be set up on shared storage. The Oracle ZFS Storage ZS3-4 is an ideal repository for this application.

```
192.168.36.231:/export/orcl/orcl_ocr - /u01/oradata nfs - yes
rw,bg,hard,rsize=32768,wsize=32768,vers=3,noac,suid,forcedirectio,nointr,proto=tcp
```

Configuring the InfiniBand Ports into IP Multipathing Groups

IP multipathing (IPMP) is a mechanism that allows for an increase in reliability and availability of network connections by supplying multiple paths for traffic over the IP network. Configuration of IPMP is done by configuring two or more Interface objects into an IPMP group on the Oracle Solaris host. An IPMP group consists of a number of Interface objects and one or more logical data addresses.

The following section details how to configure the Interface objects in an IPMP group for throughput and failures. As described, if a failure is detected for an active Interface object, a standby Interface object is made active and IPMP automatically migrates all data IP addresses used on the failed Interface object to one of the remaining available Interface objects within the IPMP group.

IPMP is handled by the Oracle Solaris 11 operating system, so configuring IPMP groups on the Oracle ZFS Storage ZS3-4 is unnecessary. This allows for a simpler Oracle ZFS Storage ZS3-4 network configuration.

1. Identify the InfiniBand network devices.

First, determine the InfiniBand network ports on the host in order to determine which ports should to be added to the IPMP groups that will be made.

```
root@aie-t5-2a:~# dladm show-ib
LINK HCAGUID PORTGUID PORT STATE PKEYS
net10 10E00001293EB4 10E00001293EB6 2 up FFFF
net11 10E00001293EB0 10E00001293EB1 1 up FFFF
net12 10E00001293EB0 10E00001293EB2 2 up FFFF
net9 10E00001293EB4 10E00001293EB5 1 up FFFF
net17 10E00001293D10 10E00001293D11 1 up FFFF
net18 10E00001293D10 10E00001293D12 2 up FFFF
net13 10E00001293E4C 10E00001293E4D 1 up FFFF
net14 10E00001293E4C 10E00001293E4E 2 up FFFF

root@aie-t5-2a:~# dladm show-phys -L
LINK DEVICE LOC
net0 ixgbe0 /SYS/MB
net1 ixgbe1 /SYS/MB
net2 ixgbe4 /SYS/MB
```
2. Determine which IB devices to use for configuring IPMP groups.

For both load balancing and redundancy, an IPMP pair should be spread across separate HCAs, as shown in the following.

| Slot3/Port2 | net18 - ibp9 - 192.168.36.181 |
| Slot4/Port1 | net13 - ibp10 - 192.168.36.182 |
| Slot4/Port2 | net14 - ibp11 - 192.168.36.183 |
| Slot5/Port1 | net9 - ibp0 - 192.168.36.184 |
| Slot5/Port2 | net10 - ibp1 - 192.168.36.185 |
| Slot6/Port1 | net11 - ibp4 - 192.168.36.186 |
| Slot6/Port2 | net12 - ibp5 - 192.168.36.187 |
| Slot3/Port1 | net17 - ibp8 - 192.168.36.180 |

3. Create an IPMP partition.

Create an IPMP partition with the same identification key as the key assigned to the Oracle ZFS Storage-ZS3-4 InfiniBand partition. This partition is used to identify which ports on the InfiniBand network will be able to communicate with each other. In this example partition FFFF is used. Create ffff partitions for each ib port. A new datalink will be created. This example uses the net18 and net13 links; you must complete the datalinks on the remaining ports as well.

```
# dladm create-part -l net18 -P 0xFFFF pFFFF.net18
# dladm show-part
```

<table>
<thead>
<tr>
<th>LINK</th>
<th>PKEY</th>
<th>OVER</th>
<th>STATE</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFFFF.net18</td>
<td>FFFF</td>
<td>net18</td>
<td>unknown</td>
<td>----</td>
</tr>
</tbody>
</table>

4. Create IP addresses for new datalinks.

Assign an IP to each of the InfiniBand datalinks.

```
# ipadm create-ip pFFFF.net18
```
5. Create an IPMP group.

# ipadm create-ipmp ipmp0

6. Add IP addresses to IPMP groups.

# ipadm add-ipmp -i pFFFF.net18 -i pFFFF.net13 ipmp0

7. Set one port to standby.

# ipadm set-ifprop -p standby=on -m ip pFFFF.net13

8. Create an IPMP ip address.

# ipadm create-addr -T static -a 192.168.36.100/22 ipmp0/v4

9. Verify IPMP groups are successfully created and that the IP address was created.

Verify the IPMP group was successfully created and online. Verify the correct IP address is assigned to the IPMP groups and that the correct datalinks are assigned to the IPMP group.
10. Test the connectivity of the IPMP group.

```plaintext
# ping 192.168.36.100
192.168.36.100 is alive
```

11. Create the remaining IB datalinks, associated IP addresses, and the IPMP groups.

```plaintext
# dladm create-part -l net14 -P 0xFFFF pFFFF.net14
# dladm create-part -l net9  -P 0xFFFF pFFFF.net9
# dladm create-part -l net10 -P 0xFFFF pFFFF.net10
# dladm create-part -l net11 -P 0xFFFF pFFFF.net11
# dladm create-part -l net12 -P 0xFFFF pFFFF.net12
# dladm create-part -l net17 -P 0xFFFF pFFFF.net17
# dladm show-part

<table>
<thead>
<tr>
<th>LINK</th>
<th>PKEY</th>
<th>OVER</th>
<th>STATE</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFFFF.net14</td>
<td>0xFFF</td>
<td>net14</td>
<td>up</td>
<td>----</td>
</tr>
<tr>
<td>pFFFF.net9</td>
<td>0xFFF</td>
<td>net9</td>
<td>unknown</td>
<td>----</td>
</tr>
<tr>
<td>pFFFF.net10</td>
<td>0xFFF</td>
<td>net10</td>
<td>unknown</td>
<td>----</td>
</tr>
<tr>
<td>pFFFF.net11</td>
<td>0xFFF</td>
<td>net11</td>
<td>unknown</td>
<td>----</td>
</tr>
<tr>
<td>pFFFF.net12</td>
<td>0xFFF</td>
<td>net12</td>
<td>unknown</td>
<td>----</td>
</tr>
<tr>
<td>pFFFF.net17</td>
<td>0xFFF</td>
<td>net17</td>
<td>unknown</td>
<td>----</td>
</tr>
</tbody>
</table>

# ipadm create-ip pFFFF.net14
# ipadm create-ip pFFFF.net9
# ipadm create-ip pFFFF.net10
# ipadm create-ip pFFFF.net11
# ipadm create-ip pFFFF.net12
# ipadm create-ip pFFFF.net17

# ipadm create-addr -T static -a 192.168.36.183  pFFFF.net14/ipv4
# ipadm create-addr -T static -a 192.168.36.184  pFFFF.net9/ipv4
# ipadm create-addr -T static -a 192.168.36.185  pFFFF.net10/ipv4
# ipadm create-addr -T static -a 192.168.36.186  pFFFF.net11/ipv4
# ipadm create-addr -T static -a 192.168.36.187  pFFFF.net12/ipv4

# ipadm create-ipmp ipmp1
# ipadm create-ipmp ipmp2
# ipadm create-ipmp ipmp3

# ipadm add-ipmp -i pFFFF.net14 -i pFFFF.net9 ipmp1
# ipadm add-ipmp -i pFFFF.net10 -i pFFFF.net11 ipmp2
# ipadm add-ipmp -i pFFFF.net12 -i pFFFF.net17 ipmp3

# ipadm set-ifprop -p standby=on -m ip pFFFF.net9
# ipadm set-ifprop -p standby=on -m ip pFFFF.net11
# ipadm set-ifprop -p standby=on -m ip pFFFF.net17

# ipadm create-addr -T static -a 192.168.36.101/22 ipmp1/v4
# ipadm create-addr -T static -a 192.168.36.102/22 ipmp2/v4
# ipadm create-addr -T static -a 192.168.36.103/22 ipmp3/v4
```
12. Verify connectivity of the IPMP groups.
Verify the connectivity of the IPMP groups by pinging the IP addresses from the Oracle ZFS Storage ZFS3-4 shell.

```
# ping 192.168.36.100
192.168.36.100 is alive
# ping 192.168.36.101
192.168.36.101 is alive
# ping 192.168.36.102
192.168.36.102 is alive
# ping 192.168.36.103
192.168.36.103 is alive
```

13. Enable dNFS on the database base nodes.
Oracle Direct NFS offers higher performance for I/O operations compared to kernel NFS due to important optimizations in the relational database management system (RDBMS) instance, the NFS client, and the remote procedure call (RPC) layer. Oracle Direct NFS also allows for striping I/O over multiple IP addresses and network interfaces to eliminate network interface bottlenecks. Enable dNFS on both database nodes with the following command and restart the database instances afterword.

```
$ make -f $ORACLE_HOME/rdbms/lib/ins_rdbms.mk dnfs_on
```

14. Create `oranfstab` on both nodes.
The `oranfstab` file configures load balancing of dNFS connections over multiple interfaces on the Oracle ZFS Storage Appliance (path). Load balancing over multiple interfaces may alleviate two possible system bottlenecks: network interface bandwidth and TCP/IP buffering.

It is recommended to match the number of path IP addresses defined in the `oranfstab` file with the number of active interfaces on the Oracle ZFS Storage Appliance controller. Also, it is recommended to match the number of local IP addresses defined in `oranfstab` with the number of active interfaces on the database server.

In this reference architecture, four active interfaces are utilized on both the database nodes and the ZFS controller. In the following example, four IP addresses are configured.

```
server: aie-zs3-4a-h1
local: 192.168.36.100 path: 192.168.36.231
local: 192.168.36.101 path: 192.168.36.233
local: 192.168.36.102 path: 192.168.36.235
local: 192.168.36.103 path: 192.168.36.237
export: /export/orcl/DATA1 mount: /zfssa/orcl/DATA
export: /export/orcl/control1 mount: /zfssa/orcl/control1
export: /export/orcl/RECO mount: /zfssa/orcl/RECO

server: aie-zs3-4a-h2
local: 192.168.36.100 path: 192.168.36.232
local: 192.168.36.101 path: 192.168.36.234
local: 192.168.36.102 path: 192.168.36.236
local: 192.168.36.103 path: 192.168.36.238
export: /export/orcl/RECO mount: /zfssa/orcl/RECO2
export: /export/orcl/DATA2 mount: /zfssa/orcl/DATA2
export: /export/orcl/control2 mount: /zfssa/orcl/control102
```

15. Verify Oracle Direct NFS.

Verify that the Oracle ZFS Storage ZS3-4 NFS shares are properly configured to utilize dNFS.

```
SQL> select * from v$dnfs_servers;
```
Oracle Database Layout

The following section offers recommendations for laying out the Oracle Database for OLTP using Oracle RAC with SPARC T5 series servers and Oracle ZFS Storage ZS3-4 attached as primary storage.

This architecture implemented the Oracle Optimal Flexible Architecture (OFA) standard. This architecture helps to standardize database deployments for performance while addressing maintenance complexity. This layout allows the database administrator to provide a software tree that eases patching and upgrades, to ensure Oracle installations are the same across all environments, and to keep separate Oracle files and installation types.

Paths and Directory Structures

The following table shows recommended path locations.

<table>
<thead>
<tr>
<th>DIRECTORY</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Base</td>
<td>/u01/app/oracle</td>
</tr>
<tr>
<td>Oracle Inventory</td>
<td>/u01/oradata</td>
</tr>
<tr>
<td>Oracle Home</td>
<td>/u01/app/oracle/product/11.2.0/dbhome_1</td>
</tr>
<tr>
<td>Oracle Database Files</td>
<td>/zfssa/orcl/DATA/ORCL</td>
</tr>
<tr>
<td></td>
<td>/zfssa/orcl/DATA2/ORCL</td>
</tr>
</tbody>
</table>
Oracle Recovery Files

- /zfssa/orcl/RECO/mp1
- /zfssa/orcl/RECO2/mp2

Oracle control files

- /zfssa/orcl/control01
- /zfssa/orcl/control02

The following table shows a recommended Oracle Database directory structure.

<table>
<thead>
<tr>
<th>PROJECT</th>
<th>PATH</th>
<th>MOUNT POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool1 (Controller1)</td>
<td>/export/orcl/DATA1</td>
<td>/zfssa/orcl/DATA/ORCL</td>
</tr>
<tr>
<td></td>
<td>/export/orcl/control1</td>
<td>/zfssa/orcl/control01</td>
</tr>
<tr>
<td></td>
<td>/export/orcl/RECO1</td>
<td>/zfssa/orcl/RECO1</td>
</tr>
<tr>
<td></td>
<td>/export/orcl/orcl_ocr</td>
<td>/u01/oradata</td>
</tr>
<tr>
<td>Pool2 (Controller2)</td>
<td>/export/orcl/DATA2</td>
<td>/zfssa/orcl/DATA2/ORCL</td>
</tr>
<tr>
<td></td>
<td>/export/orcl/control2</td>
<td>/zfssa/orcl/control02</td>
</tr>
<tr>
<td></td>
<td>/export/orcl/RECO2</td>
<td>/zfssa/orcl/RECO2</td>
</tr>
</tbody>
</table>

Recommended Management of Files and Logs

The following lists recommended best practices and configurations for the logs and files associated with the Oracle Database.

**Redo logs:**

- Isolate redo logs from all other data, and set them up to reside on a separate share.
- Configure one share from each controller – in this particular example configuration, `/zfssa/orcl/RECO` and `/zfssa/orcl/RECO2` – on the SPARC T5 database servers with Oracle RAC.
- Establish multiple redo shares on multiple Oracle ZFS Storage ZS3-4 controllers to add resilience in the event that one controller fails over. This will potentially lead to less redo logs being involved in data recovery or rollbacks, saving recovery and redo rebuild time.
- Ensure the redo share has no compression and 1 M recordsize with logbias=latency.
- For database layout, ensure that redo logs never share the same mount point with .dbf files. Combining data files and redo log files in the same share will result in performance degradation.
Datafiles:

- Datafiles (.dbf files) should reside in separate shares from each controller and should not be shared with ANY other data types.
- For both DATA shares residing on both controllers, use 8K recordsize with logbias=throughput, with no compression.
- You can enable LZJB compression for this share if storage space is a consideration or limitation, but there will be a performance impact. Reference figure 7 in the "Performance Observations" section that follows for further details.

Controlfiles:

- The database should contain two multiplexed controlfiles, with each controlfile residing on a separate share from a separate controller. This provides for database redundancy and provides better performance than placing both controlfiles in the same share from the same controller.

Archive logs

- Archive logs can be located, if necessary, within the same share as the redo logs for simplicity. The logs can be placed within their own share, with LZJB compression, if the configuration is space sensitive. If the archive logs are placed in a separate share, use recordsize=1M, logbias=latency.

Oracle Database tunables

The following table details the database resource tunables used to generate the results of this testing effort.

<table>
<thead>
<tr>
<th>RESOURCE CONTROL NAME</th>
<th>TESTED VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_cache_size</td>
<td>9428795392</td>
</tr>
<tr>
<td>java_pool_size</td>
<td>33554432</td>
</tr>
<tr>
<td>large_pool_size</td>
<td>268435456</td>
</tr>
<tr>
<td>pga_aggregate_target</td>
<td>118279372800</td>
</tr>
<tr>
<td>sga_target</td>
<td>12046041088</td>
</tr>
<tr>
<td>shared_io_pool_size</td>
<td>0</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>db_block_size</td>
<td>8192</td>
</tr>
<tr>
<td>db_files</td>
<td>512</td>
</tr>
<tr>
<td>pga_aggregate_target</td>
<td>118248964096</td>
</tr>
<tr>
<td>processes</td>
<td>1000</td>
</tr>
<tr>
<td>sga_target</td>
<td>12025069568</td>
</tr>
</tbody>
</table>
Performance Observations

This section details the performance metrics gathered during this testing effort, which recorded comparative transaction throughput from an Oracle Database running on an Oracle Exadata half rack and this paper's reference architecture using both a no compression configuration and an LZJB compression configuration.

Transaction Throughput Results

The following bar charts show the database transaction throughput results.

Figure 6. Database transaction throughput results, Transactions Per Minute

Comparing results of the Swingbench OLTP workload of a 7 TB database on an Oracle Exadata half rack system and the SPARC T5-2 and Oracle ZFS Storage reference architecture for this paper, the transactional output of the SPARC T5-2 Oracle Database with RAC with an Oracle ZFS Storage ZS3-4 as primary storage exceeded the throughput of a standard Exadata X2-2 half-rack deployment.
As indicated in the chart and following table, the SPARC T5-2/Oracle ZFS Storage ZS3-4 solution's transactional throughput per second is comparable to that of a standard Exadata half rack deployment.

**TABLE 9 TRANSACTION DATA RESULTS**

<table>
<thead>
<tr>
<th></th>
<th>Max TPM</th>
<th>Avg TPM</th>
<th>Max TPS</th>
<th>Avg TPM</th>
<th>RS TM Max</th>
<th>RS TM Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exadata ½ Rack</td>
<td>200299</td>
<td>174497</td>
<td>3828</td>
<td>2959</td>
<td>5316</td>
<td>71</td>
</tr>
<tr>
<td>SPARC T5-2 with Oracle RAC, Oracle ZFS Storage ZS3-4 Data 8k, Redo 1M (No compression)</td>
<td>201484</td>
<td>186785</td>
<td>3677</td>
<td>3140</td>
<td>36646</td>
<td>28</td>
</tr>
<tr>
<td>SPARC T5-2 with Oracle RAC, Oracle ZFS Storage ZS3-4 Data 8k Redo 1M (LZJB compression)</td>
<td>144274</td>
<td>136022</td>
<td>3301</td>
<td>2286</td>
<td><strong>3482</strong></td>
<td>67</td>
</tr>
</tbody>
</table>

The database created for the testing effort contained 4 TB of transactional data, with the overhead of the entire database, including system, redo and undo tables, reaching approximately 7.1 TB. With LZJB compression enabled on the datafiles, the overall database size was reduced to 4.55 TB.

Though overall throughput was about 30 percent less, the transactional response time was significantly lower and the data storage footprint was reduced by over 35 percent when LZJB compression was enabled on the datafile shares.
The following graphic shows the data throughput based on ZFS record size.

![Maximum OLTP Transactions per Minute](image)

Figure 8. Database transaction throughput results, Transactions Per Minute

For per minute I/O transaction rates, redo logs configured with 128 k ZFS record size had a better sustainable throughput than redo logs configured with a 1 MB ZFS record size. Setting logbias to latency for redo logs should be a consideration if configuring the share with a 128 k ZFS record size.

Figure 9 details the CPU, disk and adaptive replacement cache (ARC) utilization during the performance run with the highest I/O transaction throughput. As indicated by the red outlined boxes, Oracle ZFS Storage ZS3-4 resources were not a limiting factor on performance.
Configuring Oracle Database for OLTP on SPARC T5 Series Servers with the Oracle ZFS Storage Appliance as Primary Storage

Figure 9. Database transaction throughput results, resource utilization

The high level ARC access data misses (roughly 33 percent on each head) proves that the database did not reside entirely in cache and that transactional read requests were sent to the physical disk to retrieve the data.

Disk utilization of less than 45 per cent, for both pools, indicates disk access or disk response was not slowing transactional requests.

CPU utilization on the ZS3-4 for this load was clearly not a limiting factor, with CPU utilization under 15 percent, even with LZJB enabled.

Figure 10 shows the consistent IOPS throughput of the Oracle ZFS Storage ZS3-4. This further illustrates the viability of the Oracle ZFS Storage ZS3-4 as a primary source for Oracle Database datafiles.
Figure 10. Database transaction throughput results, Transactions Per Second
Data Warehouse Workload and Performance Impact with the Oracle ZFS Storage ZS3-4

The following section takes a brief look at the impact of a data warehouse workload. Using the same Oracle Database environment, Figure 11 illustrates the resource utilization a Data Warehouse workload has on the Oracle ZFS Storage ZS3-4.

Figure 11. An analytics example of a Data Warehouse workload, Transactions Per Second
The previous displays show these analytics:

**CPU: percent utilization broken down by CPU mode** – Nearly 60 percent of the CPU per head is being utilized for the Data Warehouse queries. There are still CPU resources available for additional load if needed.

**Disk: I/O bytes per second broken down by type of operation** – Less than 10 percent of the I/O is coming directly from the backend disks. This indicates most of the queries are cache hits.

**Cache: ARC size broken down by component** – The large cache allows staging of query reads, and assists Data Warehouse read throughput.

**Disk: Disks broken down by percent utilization** – Again confirming very little backend disk activity.

**Disk: I/O operations per second broken down by size** – Due to the 8 k database record size and the ZFS record size.

**Cache: ARC accesses per second broken down by hit/miss** – A majority of the requests are read from the cache, which allows for superior performance compared to physical disk.

**Cache: L2ARC accesses per second broken down by hit/miss** – Of the requests to the L2ARC, approximately 50 percent are cached.

**Network: interface bytes per second broken down by interface** – With a majority of the requests coming from the cache, the InfiniBand network throughput reaches over 1.5 GB per second on each Oracle ZFS Storage ZS3-4 controller, for an aggregate of over 3 GB/sec.

### Memory Management and Throughput Results

The following section discusses various Oracle Database memory management approaches, and looks at ways to efficiently tune the database for both OLTP and Data Warehouse workloads.

Automatic Shared Memory Management (ASMM) dynamically adjusts the sizes of the SGA components. The database administrator specifies the amount of memory for the database instance using the `sga_target` parameter.

Automatic Memory Management (AMM) enables the Oracle Database to manage both the SGA and the program global area (PGA) components of the instance memory using the `memory_target` parameter. Once the `memory_target` parameter is set, no additional memory management configuration is needed. Thus, due to the ease of management, Oracle recommends using AMM.

The following shows the database configured with ASMM and 12 G of memory allocated for the database instance.

```
SQL> show parameter memory
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
</table>

---
AMM is enabled by issuing the following commands.

```
alter system set memory_max=12G scope=spfile;
alter system set sga_target=0 scope=spfile;
alter system set sga_max_size=0 scope=spfile;
alter system set pga_aggregate_target=0 scope=spfile;
```

The following screen output shows the changes to the database after the database is restarted.

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
hi_shared_memory_address             integer     0
memory_max_target                    big integer 12G
memory_target                        big integer 12G
shared_memory_address                integer     0
SQL> exit
```

Figure 12 demonstrates substantial benefits to using ASMM when allocating memory for an OLTP type workload. Specifically, assigning memory to the SGA and the PGA using ASMM allows for much higher transactional IOPS than if AMM is enabled and Oracle Database is allowed to dynamically assign SGA and PGA variables. For an OLTP workload, ASMM memory management is recommended.
Table 10 shows bandwidth in a Data Warehouse workload using ASM compared to AMM. Recall that bandwidth will correlate to IOPS.

<table>
<thead>
<tr>
<th>MEMORY MANAGEMENT</th>
<th>DATA WAREHOUSE THROUGHPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMM</td>
<td>3.09 GB/sec</td>
</tr>
<tr>
<td>AMM with 12 G memory_target</td>
<td>3.10 GB/sec</td>
</tr>
<tr>
<td>AMM with 4 G memory_target</td>
<td>3.04 GB/sec</td>
</tr>
</tbody>
</table>

Notice that there is no advantage to configuring the database memory management with ASMM for a Data Warehouse workload; therefore, the recommendation for these workloads would be to implement AMM for its ease of use.
Migrating an Existing Database to the Oracle ZFS Storage ZS3-4

The following illustrates the steps used to migrate an existing database to an Oracle ZFS Storage ZS3-4. Consider this procedure as a reference, as each database configuration will be unique.

Log on as the oracle user on the existing environment where the original database resides. As the oracle user, with the ORACLE_SID instance set to the original database to be migrated, retrieve the database ID (DBID) of the database that is to be migrated. This DBID will be referenced in the RMAN duplicate script used to migrate the database to the Oracle ZFS Storage ZS3-4.

```
SQL> select name, dbid from v$database;
NAME            DBID
--------- ----------
DOOM 1218299132
```

Set the original Oracle Database Remote Application Clusters to operate as a single node. This simplifies the RMAN duplication process and will be switched back to clustered mode after the duplication.

```
SQL> alter system set cluster_database=FALSE  scope=spfile sid='*';
System altered.
```

Stop and restart the database for the `alter system set` command to take effect.

```
$srvctl stop database –d ao; srvctl start database –d ao
```

Start `rman`.

```
$rman target /
```

Run RMAN in mount only. Execute a backup of the current database. Run a full back and include the controlfile. Alter the database to open and run a backup and include the archive logs. Validate and crosscheck the backup. Specify a location for the backup to occur. In this example, the backup location resides on an Oracle ZFS Storage ZS3-2 cluster used as an RMAN backup repository.

```
RMAN> startup mount
RMAN> backup format '/zfssa/stage/backup3/%U' database include current controlfile;
```
The next steps take place on the SPARC T5-2/Oracle ZFS Storage ZS3-4 architecture.

Set up the Oracle ZFS Storage ZS3-4 filesystems, configure Oracle Solaris 11 memory modules, and install and configure the Oracle Database Remote Application Clusters on the SPARC T5-2 servers. This will be the target location for the duplicate database. These configuration procedures, described earlier in this document, should be followed prior to completing the next steps.

Create additional directories on the mounted Oracle ZFS Storage ZS3-4 filesystems to reflect the layout for the migration of the new database.

The following command line executions should be done as the oracle user.

```bash
$ mkdir -p /zfssa/doom/RECO/mp1/doom
$ mkdir -p /zfssa/doom/RECO2/mp2/doom
$ mkdir -p /zfssa/doom/DATA/doom/datafile
$ mkdir -p /zfssa/doom/DATA2/doom/datafile
$ mkdir -p /zfssa/doom/RECO/doom
$ mkdir -p /u01/app/oracle/admin/doom/adump

Edit the tnsnames.ora file on all Oracle RAC nodes. Add an entry for a target/duplicate database.

```bash
$vi $ORACLE_HOME/network/admin/tnsnames.ora

ORCL =
   (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = oooscan)(PORT = 1521))
      (CONNECT_DATA =
         (SERVER = DEDICATED)
         (SERVICE_NAME = orcl)
      )
   )
```
Create the `init.ora` file for the target/duplicate database.

```bash
$cd $ORACLE_HOME/dbs
$vi initdoom1.ora
```

Add only the name of the target database into the `init.ora` file. The other database parameters will be assigned during the duplication process.

```ora
db_name=doom
```

Format the time/date environment variable `NLS_DATE_FORMAT` to the following. This format will be used to call the backup timestamp for the RMAN backup used to duplication the database.

```bash
$ NLS_DATE_FORMAT=DD_MON_YYYY_HH24:MI:SS;export NLS_DATE_FORMAT
```

Export the `ORACLE_SID` environment variable to reflect the target database SID id.

```bash
$export ORACLE_SID=doom1
```

Start the target database in nomount mode.

```bash
$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Aug 1 11:19:42 2014
Copyright (c) 1982, 2013, Oracle. All rights reserved.
Connected to an idle instance.
SQL> startup nomount
```

Start RMAN in auxiliary mode.

```bash
$rman auxiliary /
```

Run the duplication script to migrate the database to the SPARC T5-2/Oracle ZFS Storage ZS3-4 environment.
The following is an example of a duplication script to be used as reference.

Eight channels are allocated to distribute resources across multiple threads.

The set newname for datafile is used for each datafile. The datafiles are distributed evenly across the two DATA mount points. This way, the datafiles will reside equally on both Oracle ZFS Storage ZS3-4 controllers and will spread evenly across all the disk trays in the storage configuration.

A new location is created for the multiplexed controlfiles and the redo logs. The path should point to the newly created and mounted Oracle ZFS Storage ZS3-4 shares.

```
run {
  allocate auxiliary channel ch1 type disk;
  allocate auxiliary channel ch2 type disk;
  allocate auxiliary channel ch3 type disk;
  allocate auxiliary channel ch4 type disk;
  allocate auxiliary channel ch5 type disk;
  allocate auxiliary channel ch6 type disk;
  allocate auxiliary channel ch7 type disk;
  allocate auxiliary channel ch8 type disk;
  set newname for datafile 6 to '/zfssa/doom/DATA/doom/datafile/%U';
  set newname for datafile 7 to '/zfssa/doom/DATA2/doom/datafile/%U';
  set newname for datafile 8 to '/zfssa/doom/DATA/doom/datafile/%U';
  set newname for datafile 9 to '/zfssa/doom/DATA2/doom/datafile/%U';
  ...
  set newname for datafile 399 to '/zfssa/doom/DATA2/doom/datafile/%U';
  set newname for datafile 406 to '/zfssa/doom/DATA/doom/datafile/%U';
  duplicate database ao dbid 1218299132 to doom
  spfile
  set db_create_file_dest='/zfssa/doom/DATA/doom/datafile'
  set control_files='/zfssa/doom/control01/control01.ctl','/zfssa/doom/control02/control02.ctl'
  set instance_number='3'
  set db_create_online_log_dest_1 = '/zfssa/doom/RECO/mp1/doom/'
  set db_create_online_log_dest_2 = '/zfssa/doom/RECO2/mp2/doom/'
  set db_recovery_file_dest='/zfssa/doom/RECO/doom'
  backup location '/zfssa/stage/backup3';
}
```

Set the newly duplicated database to cluster mode. This allows the database to run multiple database instances on separate nodes.

```
$ sqlplus / as sysdba
SQL> alter system set cluster_database=TRUE scope=spfile sid='**';
SQL> shutdown immediate
```
Create a temporary init.ora file. This file must be manually edited to reflect parameters of the newly duplicated database.

```
$sqlplus / as sysdba
SQL> startup
SQL> create pfile='/zfssa/doom/DATA/doom/tempdoom.ora' from spfile;
SQL> shutdown immediate
```

Edit the newly created pfile.

```
$vi /zfssa/doom/DATA/doom/tempdoom.ora
```

Remove all entries of the original database. In this example all references to database AO are replaced with doom within the .ora pfile.

Ensure there are entries for each database RAC node. In this example, entries are added for both instance doom1 and doom2.

Remove the added instance_number line. In this example, there are currently two nodes, and the 
*.instance_number=3 line was added to avoid instance number conflict during the duplication process.

```
doom2._db_cache_size=9160359936
doom1._db_cache_size=8992587776
doom2._java_pool_size=234881024
doom1._java_pool_size=234881024
doom2._large_pool_size=436207616
doom1._large_pool_size=436207616
doom2._pga_aggregate_target=118279372800
doom1._pga_aggregate_target=118279372800
doom2._sga_target=12046041088
doom1._sga_target=12046041088
doom2._shared_io_pool_size=0
doom1._shared_io_pool_size=0
doom2._stream_pool_size=67108864
doom1._stream_pool_size=67108864
*.audit_file_dest='/u01/app/oracle/admin/doom/adump'
*.audit_trail='db'
*.cluster_database=TRUE
*.instance_number=3
*.compatible='11.2.0.4.0'
*.control_files='/zfssa/doom/control01/control01.ctl','/zfssa/doom/control02/control02.ctl'
*.db_block_size=8192
```
As the oracle user, start up the local node database using the newly created pfile. Create an spfile from the pfile and shut down the database.

```
$ sqlplus / as sysdba
SQL> startup pfile=/zfssa/doom/DATA/doom/tempdoom.ora
SQL> create spfile='/zfssa/doom/DATA/doom/spfiledoom.ora' from pfile='/zfssa/doom/DATA/doom/tempdoom.ora';
SQL> shutdown immediate
```

```
$vi /u01/app/oracle/product/11.2.0/dbhome_1/dbs/initdoom1.ora

SPFILE='/zfssa/doom/DATA/doom/spfiledoom.ora'

Add an init.ora file on remaining Oracle RAC nodes.

```
$vi /u01/app/oracle/product/11.2.0/dbhome_1/dbs/initdoom2.ora

Enter the location of the newly created spfile located on the shared filesystem.

```

```
SPFILE='/zfssa/doom/DATA/doom/spfileloki.ora'

Use the srvctl command to add the newly duplicated database. Adding the database using srvctl allows the oracle user to manage the Oracle Database RAC instances and services.
$ srvctl add database -d doom -o /u01/app/oracle/product/11.2.0/dbhome_1

Add the spfile location with `srvctl`.

$ srvctl modify database -d doom -p '/zfssa/doom/DATA/doom/spfiledoom.ora' -s open

Add each instance of the Oracle RAC.

$ srvctl add instance -d doom -i doom1 -n aie-t5-2a
$ srvctl add instance -d doom -i doom2 -n aie-t5-2b

Start each instance of the Oracle RAC.

$ srvctl start instance -d doom -i doom1
$ srvctl start instance -d doom -i doom2

Verify the Oracle RAC-based database starts without error.

$ srvctl status database -d doom
Instance loki1 is running on node aie-t5-2a
Instance loki2 is running on node aie-t5-2b
Conclusion

This document has provided planning information and step-by-step instructions for configuring and tuning both the SPARC T5-2 servers and the Oracle ZFS Storage Appliance in an OLTP-driven Oracle Database with Oracle RAC environment. Additionally, this document demonstrated that the architecture of the Oracle ZFS Storage Appliance is ideally suited for a database environment that requires flexibility, large capacities and high IOPS throughput requirements.

Thus, the result of this testing effort clearly demonstrates that the Oracle ZFS Storage Appliance, when paired with a SPARC T5 series server and the Oracle RAC feature of Oracle Database, makes for a viable and cost effective solution for a transactional type database deployment.
Appendix A: Swingbench Workload

The Swingbench Order Entry Workload, or SOE, is a classic order entry benchmark similar to the TPC-C benchmark. This type of workload typically has a slight read bias, and for this testing effort a 60/40 read/write ratio was utilized. Swingbench uses the Oracle Database “oe” scheme as its basis for an order entry workload. This heavy contention workload focuses on a small number of tables and is designed to stress interconnects and memory. As discussed earlier in the document, it is important to note the role of the Oracle ZFS Storage ZS3-4 ARC cache. Due to the small number of tables utilized and the large size of the ARC cache, a majority of the OLTP read requests probably become cached in memory.

To determine an ideal users set for the Swingbench workload, users were added in blocks of 100 until the additional users significantly impacted the maximum response time in milliseconds. The runs were completed using one 8 K record size share for data, and one 128 K share for redo logs.

<table>
<thead>
<tr>
<th>Table 9</th>
<th>Transaction Data Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max TPM</td>
</tr>
<tr>
<td>100 Users</td>
<td>51009</td>
</tr>
<tr>
<td>200 Users</td>
<td>104857</td>
</tr>
<tr>
<td>300 Users</td>
<td>142856</td>
</tr>
<tr>
<td>400 Users</td>
<td>156814</td>
</tr>
<tr>
<td>500 Users</td>
<td>144492</td>
</tr>
<tr>
<td>600 Users</td>
<td>141691</td>
</tr>
</tbody>
</table>

This table demonstrates the reasoning for choosing 300 users as a data input for the Swingbench OLTP workload. When the users exceeded 300, the maximum transactional response time spiked significantly, indicating the user transactions queue began to fill and users requests were becoming difficult to process.

Appendix B: Automatic Workload Repository Report

For a more in-depth look at the Oracle Database load profile used for gathering OLTP metrics, portions of the Automatic Workload Repository (AWR) report are included.

Load Profile: This section shows important rates expressed in units of per second and per transactions per second in a snapshot of the database workload that occurred during the snapshot interval.
**Load Profile**

<table>
<thead>
<tr>
<th></th>
<th>Per Second</th>
<th>Per Transaction</th>
<th>Per Exec</th>
<th>Per Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS Time(s):</td>
<td>45.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
</tr>
<tr>
<td>DS CPU(s):</td>
<td>10.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>Redo size (bytes):</td>
<td>6,737,667.7</td>
<td>4,403.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical read (blocks):</td>
<td>104,525.3</td>
<td>107.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block changes:</td>
<td>34,748.5</td>
<td>22.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical read (blocks):</td>
<td>10,531.0</td>
<td>6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical write (blocks):</td>
<td>4,643.1</td>
<td>3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read I/O requests:</td>
<td>10,531.0</td>
<td>6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write I/O requests:</td>
<td>4,121.2</td>
<td>2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redo I/O (MB):</td>
<td>823.3</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write I/O (MB):</td>
<td>37.9</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Cache blocks received:</td>
<td>4,369.5</td>
<td>2.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Cache blocks served:</td>
<td>4,437.1</td>
<td>2.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User calls:</td>
<td>4,841.6</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parses (SQL):</td>
<td>1,549.8</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard parses (SQL):</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL Work Area (MB):</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logins:</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executes (SQL):</td>
<td>16,022.4</td>
<td>10.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rollbacks:</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transactions:</td>
<td>1,530.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 13. AWR report database load profile

**I/O Section:** This section shows the all important I/O activity for the instance and shows I/O activity by tablespace, data file, and includes buffer pool statistics, tablespace I/O Stats, file IO Stats, and Buffer Pool Statistics

**IO Profile**

<table>
<thead>
<tr>
<th></th>
<th>Read+Write Per Second</th>
<th>Read per Second</th>
<th>Write Per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Requests:</td>
<td>17,391.8</td>
<td>10,514.6</td>
<td>6,877.1</td>
</tr>
<tr>
<td>Database Requests:</td>
<td>14,852.2</td>
<td>10,531.0</td>
<td>4,121.2</td>
</tr>
<tr>
<td>Optimized Requests:</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Redo Requests:</td>
<td>2,753.6</td>
<td>0.0</td>
<td>2,746.6</td>
</tr>
<tr>
<td>Total (MB):</td>
<td>148.9</td>
<td>88.2</td>
<td>59.6</td>
</tr>
<tr>
<td>Database (MB):</td>
<td>120.2</td>
<td>82.3</td>
<td>37.9</td>
</tr>
<tr>
<td>Optimized Total (MB):</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Redo (MB):</td>
<td>28.8</td>
<td>0.6</td>
<td>14.0</td>
</tr>
<tr>
<td>Database (blocks):</td>
<td>15,379.0</td>
<td>10,531.0</td>
<td>4,846.1</td>
</tr>
<tr>
<td>Via Buffer Cache (blocks):</td>
<td>15,376.9</td>
<td>10,525.4</td>
<td>4,847.6</td>
</tr>
<tr>
<td>Direct (blocks):</td>
<td>2.1</td>
<td>1.6</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Figure 14. AWR report showing database I/O profiles

**Wait Events:** This AWR report section provides more detailed wait event information for foreground user processes, which includes top wait events and wait types that occurred during the snapshot interval.
### Top 10 Foreground Events by Total Wait Time

<table>
<thead>
<tr>
<th>Event</th>
<th>Waits</th>
<th>Total Wait Time (sec)</th>
<th>Wait Avg (ms)</th>
<th>% DB time</th>
<th>Wait Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>db file sequential read</td>
<td>38,488,189</td>
<td>1045.93</td>
<td>3</td>
<td>66.7</td>
<td>User I/O</td>
</tr>
<tr>
<td>DB CPU</td>
<td>38,920</td>
<td>2086.1</td>
<td>10070</td>
<td>1.8</td>
<td>Other</td>
</tr>
<tr>
<td>ioq file sync</td>
<td>5,615,030</td>
<td>19.05</td>
<td>3</td>
<td>11.0</td>
<td>Commit</td>
</tr>
<tr>
<td>enq: TX - contention</td>
<td>309</td>
<td>3111.7</td>
<td>2</td>
<td>1.7</td>
<td>Other</td>
</tr>
<tr>
<td>gci grant 2-way</td>
<td>11,960,025</td>
<td>3004.6</td>
<td>0</td>
<td>1.7</td>
<td>Cluster</td>
</tr>
<tr>
<td>enq: TX - index contention</td>
<td>309</td>
<td>2086.1</td>
<td>0</td>
<td>1.7</td>
<td>Other</td>
</tr>
<tr>
<td>gci current block 2-way</td>
<td>8,478,371</td>
<td>2659.9</td>
<td>0</td>
<td>1.6</td>
<td>Cluster</td>
</tr>
<tr>
<td>gci current block 2-way</td>
<td>7,247,383</td>
<td>2559.3</td>
<td>0</td>
<td>1.5</td>
<td>Cluster</td>
</tr>
<tr>
<td>gci current grant 2-way</td>
<td>7,468,451</td>
<td>1906.7</td>
<td>0</td>
<td>1.1</td>
<td>Cluster</td>
</tr>
<tr>
<td>gci current grant busy</td>
<td>2,155,283</td>
<td>784.4</td>
<td>0</td>
<td>0.5</td>
<td>Cluster</td>
</tr>
</tbody>
</table>

### Wait Classes by Total Wait Time

<table>
<thead>
<tr>
<th>Wait Class</th>
<th>Waits</th>
<th>Total Wait Time (sec)</th>
<th>Avg Wait (ms)</th>
<th>% DB time</th>
<th>Avg Active Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>User I/O</td>
<td>38,593,856</td>
<td>104,741</td>
<td>3</td>
<td>60.8</td>
<td>28.6</td>
</tr>
<tr>
<td>System I/O</td>
<td>6,368,502</td>
<td>52,752</td>
<td>8</td>
<td>30.6</td>
<td>14.4</td>
</tr>
<tr>
<td>DB CPU</td>
<td>38,920</td>
<td>2086.1</td>
<td>10070</td>
<td>1.8</td>
<td>10.5</td>
</tr>
<tr>
<td>Commit</td>
<td>5,615,030</td>
<td>18,975</td>
<td>3</td>
<td>11.0</td>
<td>5.2</td>
</tr>
<tr>
<td>Cluster</td>
<td>37,848,032</td>
<td>11,491</td>
<td>0</td>
<td>6.7</td>
<td>3.1</td>
</tr>
<tr>
<td>Other</td>
<td>1,268,383</td>
<td>3,746</td>
<td>3</td>
<td>2.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Concurrency</td>
<td>2,321,957</td>
<td>3,141</td>
<td>1</td>
<td>1.8</td>
<td>0.9</td>
</tr>
<tr>
<td>Network</td>
<td>11,348,831</td>
<td>23</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Config</td>
<td>3,589</td>
<td>22</td>
<td>6</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Application</td>
<td>997</td>
<td>1</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Administrative</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 15. AWR report wait times
Appendix C: References

Oracle ZFS Storage Appliance Documentation

See the following resources for additional information relating to the products covered in this document:

- Oracle ZFS Storage Appliance Documentation Library, including Installation, Analytics, Customer Service, and Administration guides:  

- The Oracle ZFS Storage Appliance Administration Guide is also available through the Oracle ZFS Storage Appliance help context.  
The Help function in Oracle ZFS Storage Appliance can be accessed through the browser user interface.

- Oracle Support Center  
  http://www.oracle.com/support

- Patches and updates downloads from My Oracle Support (MOS) (search under Oracle ZFS Storage Software Patches)

- Oracle ZFS Storage Appliance Plug-ins  

- Oracle Storage Product Information  

- Oracle ZFS Storage Appliance Technical White Papers and Solution Briefs, including "Best Practices for Oracle ZFS Storage Appliance and VMware vSphere5.x"  