



An Oracle White Paper
July 2010

Configuring Oracle[®] Solaris ZFS for an Oracle[®] Database

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Configuring Oracle Solaris ZFS for an Oracle Database	1
Overview	1
Planning an Oracle Solaris ZFS for Oracle Database Configuration	2
Configuring a Sun Storage 7000 System for an Oracle Database	6
Configuring an Oracle Solaris ZFS System for an Oracle Database	11
Configuring Your Oracle Database on Oracle Solaris ZFS File Systems	14
Maintaining Oracle Solaris ZFS File Systems for an Oracle Database	15

Configuring Oracle Solaris ZFS for an Oracle Database

This document provides planning information and step-by-step instructions for configuring and maintaining Oracle Solaris ZFS file systems for an Oracle database.

- Planning an Oracle Solaris ZFS for Oracle Database Configuration
- Configuring a Sun Storage 7000 System for an Oracle Database
- Configuring an Oracle Solaris ZFS System for an Oracle Database
- Configuring Your Oracle Database on Oracle Solaris ZFS File Systems
- Maintaining Oracle Solaris ZFS File Systems for an Oracle Database

Overview

Today, a large number of enterprise customers run an Oracle relational database on top of storage arrays with different sophistication levels and many customers would like to take advantage of the Oracle Solaris ZFS file system. The paper focuses on the characteristics of the Sun ZFS Storage 7000 as a flexible storage solution providing great functionality and value. This paper also describes high performance storage array usage, such as Oracle's Sun Storage 6000 Array, to help customers who are transitioning to ZFS on an existing storage infrastructure.

Using the Oracle Solaris ZFS file system can reduce the cost and complexity of managing data and storage devices for hosting an Oracle database. Since storage product capabilities often overlap Oracle Solaris ZFS features, a wide range of options are offered for deployment. This document provides guidance on how to economically deploy an Oracle database on Oracle Solaris ZFS to maximize data integrity and performance. We start with a brief planning section and follow with step-by-step instructions for configuring and maintaining Oracle Solaris ZFS file systems for an Oracle database.

Planning an Oracle Solaris ZFS for Oracle Database Configuration

The following sections provide specific planning information that is needed before configuring Oracle Solaris ZFS file systems for an Oracle database.

1. **Oracle Solaris release information** - Use the latest Oracle Solaris 10 release, such as Oracle Solaris 10 10/09 release and apply the latest kernel patch, if possible. Apply Oracle Solaris 10 patch 142900-09 (for SPARC) or patch 142901-09 (for x86/x64) or later patch versions for any Oracle Solaris release prior to Oracle Solaris 10 10/09 release. The minimum Oracle Solaris release required is the Oracle Solaris 10 10/08 release.

The Sun Storage 7000 system provides access to all of the features described in this paper.

2. **Oracle release information** - Oracle Solaris ZFS is recommended for any Oracle database version in single instance mode, and Oracle Solaris ZFS can be used with an Oracle RAC database only through a NFS-shared file system, typically with a Sun ZFS Storage 7000.
3. **Storage array considerations** – Consider the following points when configuring your storage arrays:
 - Confirm with your array vendor that the disk array is *not* flushing its cache after a flush write cache request is issued by Oracle Solaris ZFS. Oracle's Sun Storage 6000 and 7000 series systems behave properly with respect to those cache flush requests.
 - Use whole disks, not disk slices, as storage pool devices so that Oracle Solaris ZFS will then format the disks and activate the local small disk caches, which get flushed at appropriate times. Slices can be used for the separate redo logs pool with a separate log device as described later in this document.
 - For best performance, create one LUN for each physical disk in the array. Using only one large LUN can cause Oracle Solaris ZFS to queue up too few read I/O operations to actually drive the storage to optimal performance. Conversely, using many small LUNs could have the effect of swamping the storage with a large number of pending read I/O operations.
 - A storage array that uses dynamic provisioning software to implement virtual space-allocation is not recommended for Oracle Solaris ZFS. When Oracle Solaris ZFS writes the modified data to free space, it rapidly writes to the entire LUN volume, even with only a small percentage of used space. The Oracle Solaris ZFS write process allocates all the virtual space from the storage array's point of view, which negates the benefit of dynamic provisioning.
 - If you have traditional hardware RAID protection and you are comfortable with that level of protection, then it is possible to use that mode. Otherwise, for the highest level of protection, we recommend using Oracle Solaris ZFS redundancy, such as a mirrored storage pool.

4. **Storage pool considerations** - With a mirrored storage pool, Oracle Solaris ZFS can automatically detect and repair data-corruption. We do not recommend RAIDZ redundancy when I/O operations per second (IOPS) performance is the main-criterion for selecting a storage configuration for your database. For more information about creating redundant Oracle Solaris ZFS storage pools, see the Oracle Solaris ZFS Administration Guide.
5. **Match Oracle Solaris ZFS record size to Oracle database block size** – Oracle Solaris ZFS has a configurable `recordsize` property to set a suggested block size for file systems that you can apply to different database components. The general rule is to set `recordsize = db_block_size` for the file system that contains the Oracle datafiles.
- When the `db_block_size` is less than the page size of the server, 8 KB on SPARC systems and 4 KB on x64 systems, set the record size to the page size. On SPARC systems, typical record sizes for large databases are as follows:

File System	Record Size
Table data	8 KB
Redo logs	128 KB (default)
Index files	8 KB
Undo data	128 KB (default) or sometimes 8 KB
Temp data	128 KB (default)
Archive data	128 KB (default) compression on

Note: Keep in mind that modifying the Oracle Solaris ZFS file system `recordsize` parameter affects only the files created after the change. To change the size that is used by an existing datafile, you must first change the `recordsize` property of the file system used to store the file, and then copy the file into the file system.

- Many databases have all their datafiles in a single Oracle Solaris ZFS file system with the record size set to `db_block_size`. This setting provides reasonable performance and is the minimum you should do to improve database performance.
 - For better performance, and for very large databases, you can use different `db_block_size` values for different database components. In these cases, you should use one Oracle Solaris ZFS file system per `db_block_size` and match file system record size to `db_block_size`. We recommend that you host all Oracle tablespaces dedicated file systems within a single storage pool. The redolog files can be stored on a separate pool as described below.
6. **Improve database writing and caching performance** - On systems running the current OpenSolaris release or on SS7000 systems, you can use the “Synchronous write bias” setting, equivalent to the Oracle Solaris ZFS `logbias` property, to improve performance when writing or caching database files.

On systems running the Solaris 10 10/09 release, you can use the `primarycache` property to control what is cached in main memory (the primary ARC).

- **SS7000 systems** – The following table identifies the file system type and matching record size, synchronous write bias, and caching property settings in the SS7000 GUI interface.

File System	Database record size	Synchronous write bias	Cache device usage
Table data	8 KB	Throughput	All data and metadata
Redo logs	128 KB	Latency	Do not use cache devices
Index files	8 KB	Throughput	All data and metadata
Undo data	128 KB	Throughput	Do not use cache devices
Temp data	128 KB	Throughput	Do not use cache devices

- **Solaris systems** - The following table identifies the file system type and the matching record size value, `logbias` value, and `primarycache` values for a generic OLTP database with a `db_block_size` set to 8 KB.

File System	Record Size	Logbias Value	Primarycache Value
Table data	8 KB	throughput	all (data and metadata)
Redo logs	128 KB (default)	Latency (default)	all (data and metadata)
Index files	8 KB	throughput	all (data and metadata)
Undo data	128 KB or 8 KB	throughput	metadata
Temp data	128 KB (default)	throughput	all (data and metadata)
Archive data	128 KB, compression on	throughput	metadata

The redo log is listed above with the `logbias` property set to `latency`, the default value, which should be used in most environments. The exception to using the default `logbias` value is when using a storage subsystem that is saturating its-throughput capacity (MB/s). In this case, setting `logbias=throughput` for redo logs might prevent the doubling of writes to the redo log pool, thus reducing the MB/s impact on the storage subsystem.

Consider the following `logbias` and `primarycache` property values for a data warehouse database.

File System	Record Size	logbias Value	primarycache Value
Table data	128 KB	throughput	all (data and metadata)
Index data	<code>db_block_size</code> (8 KB, 16 KB, or 32 KB)	throughput	all (data and metadata)

7. **Tune synchronous write activity** - Oracle Solaris ZFS manages synchronous writes internally with the Oracle Solaris ZFS intent log (ZIL). In general, writes are stored in the ZIL at low latency and later stored in the pool, leading to a doubling of the flow of data between the host and the storage. For redo log activity, this is beneficial for response latency and considered a necessary tradeoff. For Oracle datafiles, setting the `logbias` property, when available, avoids the doubling of writes. For the Solaris 10 10/08, 05/09 or 10/09 releases, if the extra load is a source of concern, then it is possible to avoid the double write issue. Consult the Oracle Solaris ZFS Tuning wiki for details.
http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide
8. **Use secondary cache (L2ARC)** - You can add LUNs as secondary cache (L2ARC) devices in a storage pool starting in the Solaris 10 10/09 release. These devices store a cached copy of disk content for quick access and to augment the total random read throughput of the storage. To this end, the use of low latency and efficient \$/IOPS devices, such as an SSD device, is recommended.

Use the `secondarycache` property to specify which file systems will be allowed into the secondary cache.

Using a secondary cache is expected to be beneficial for read-latency sensitive workloads, for both index and database blocks. As usual, a cache layer is most effective when cache hit rates are high, implying that the total size of cache devices should be sized according to the application's warm working set. Moreover, in an environment where performance peaks when physical disks are saturated with random reads, the caching devices offer a venue for additional read IOPS.

Currently, caching devices do not retain data between system reboots. So, allow for many hours of operation to gauge their effectiveness in your environment. Enabling the secondary cache for redo logs is not recommended at this time.

9. Allocate sufficient memory and swap resources.

One advantage of using Sun ZFS Storage 7000 is that a very large storage cache is managed at the storage level and does not compete for memory with the Oracle database engine and application. However, if you are running ZFS on the same Solaris server-that runs the Oracle database,-then consider the system's total memory consumption. Like most file systems, ZFS will use most of a system's memory to cache data and will shrink it's allocation based on memory pressure.-Identify the memory requirements for the Oracle database and applications (all processes and associated memory allocation), and prevent ZFS from growing beyond those requirements.

You can reduce Oracle Solaris ZFS memory consumption by setting the `zfs_arc_max` parameter. Note that-we still recommend provisioning at least enough memory to cache metadata for the actively used portion of the database, which is estimated at 1.5% with an 8 KB Oracle Solaris ZFS record size and proportionately less or more with larger or smaller records. The file system that benefits most from file system caching is the file system that holds the index files because it is the last one to be invalidated by the Oracle database when memory pressure increases. Configuring more than the minimum amount is often a good way to reduce the I/O requirements on the disk storage.

The `zfs_arc_max` parameter is in bytes and accepts decimal or hexadecimal values.

The two following examples set this parameter at the same value of 2 GB:

```
set zfs:zfs_arc_max=2147483648
set zfs:zfs_arc_max=0x8000000
```

In addition, to prevent applications from failing due to lack of memory, you must configure swap space when the system running the Oracle database also manages a ZFS pool. An amount of swap equivalent to 100% of the system's memory is always enough for this purpose. This swap space is not expected to be used, but is needed as a reservation area. Because this space is only reserved, the performance characteristics of swap storage should not be a critical element of a deployment.

For information about increasing swap space, see the Oracle Solaris ZFS Administration Guide.

10. Consider additional performance tuning with the usual caveats.

You might be tempted to increase performance through specific per-workload tuning. Tuning a system outside of its normal state is risky, and generally, is not recommended. Nevertheless, if you wish to operate under such conditions, the ZFS Evil Tuning Guide describes some tunables that have had some success in increasing performance under Oracle database workload. These tunable parameters are as follows:-

- `zfs_vdev_max_pending`
- `ssd_max_throttle`
- `metaslab_df_free_pct`
- `zfs_immediate_write_sz`

For more information, see

http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide.

Configuring a Sun Storage 7000 System for an Oracle Database

After you have reviewed the preceding planning section, use the steps below to configure a Sun Storage 7000 system for an Oracle database.

When using the Sun Storage 7000 system, the recommended hardware configuration can be directly implemented at the storage level. After the storage appliance is configured, the ZFS file systems can be created, shared over NFS, and can be accessed with Oracle's dNFS client software.

This configuration works for any Oracle database including a clustered RAC database. The Solaris NFSv3 client is also supported with the `forcedirectio` mount option, which has

been shown in the past to be effective with an Oracle database. Because an Oracle database does not use any-NFSv4 features, we recommend using NFSv3.

Running a database over NFS is extremely flexible and easy to manage. Moreover, the SS7000 system with its Analytics tool offers unmatched observability into the workloads. The Sun ZFS Storage 7000 system also provides great value with all data services required for running an enterprise from a single administration point.

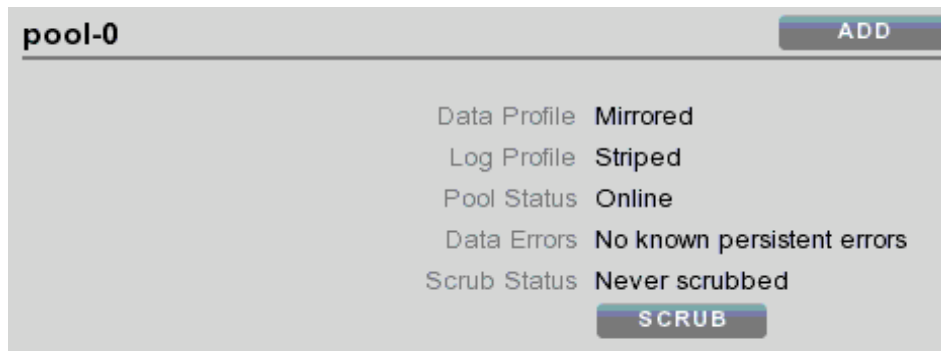
The steps below can be done by using the SS7000 system's by using a graphical user interface in a browser or by using scriptable command lines.

How to Configure a Sun Storage 70000 System for an Oracle Database

1. Create a mirrored storage pool for data files for tables, index, undo and temp data.

In many cases, the databases will be better handled by a mirrored storage pool. Refer to this Sun Storage 7000 url <http://yourserver:215/#configuration/storage> for more information about your storage profile.

From the Configuration:Storage screen, follow the steps to create a mirrored storage pool.



2. Create a project for each database and define the default values for the project.

From the Shares/Projects screen, follow the steps to create a project.

As the datafiles for the tables and the index are often in many directories or file systems, use default settings for the table and index datafiles, which are provided as ZFS properties under the Shares/Projects - Properties section. See the above planning section for a description of each of these settings under **Improve database writing and caching performance**. In the following example, the project SID is created with the following inherited properties:

- **Database record size** - This property is equivalent to the ZFS file system `recordsize` property. For example, set the database record size (`recordsize`) and match the Oracle `db_block_size` value, which is typically set to 8KB.
- **Synchronous write bias** - This property is equivalent to the ZFS file system `logbias` property. Use the default value "Throughput."
- **Cache device usage** - This property is equivalent to the ZFS file system `secondarycache` property. Use the default value "all data an metadata."

Use the default values for the other Shares/Properties settings. Data redundancy is provided by the mirrored storage pool configuration so no additional block replication is needed.

The ZFS property names can be specified when using the scriptable command lines.

3. Select the settings (or file system properties) for the project from the Shares/Project screen.

Inherited Properties

Mountpoint

Read only

Update access time on read

Non-blocking mandatory locking

Data deduplication

Data compression

Checksum

Cache device usage

Synchronous write bias

Database record size

Additional replication

Virus scan

Prevent destruction

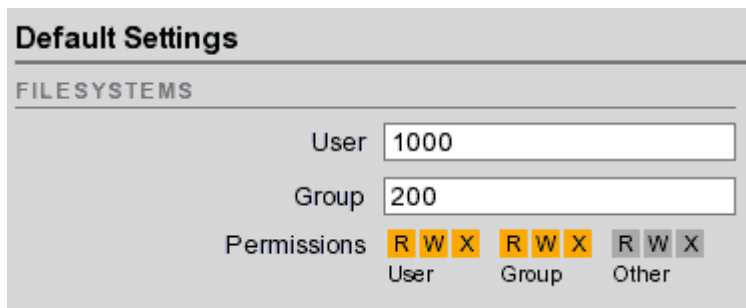
Select the default values except select the `db_block_size` (here at 8k) for the Database record size setting.

The default project configuration expected is (relative to url `/#shares/projects=SID+general`).

4. Configure the numeric user ID and group ID (UID and GID) of the Oracle account to be used with the mounted file systems.

Use the UNIX `id` command under the Oracle database owner account on the server where the NFS file system will be mounted to identify the UID and GID as example:

```
$ id
uid=1000(oracle) gid=200(dba)
```



Default Settings

FILESYSTEMS

User

Group

Permissions **R W X** **R W X** **R W X**

 User Group Other

5. Create shares in the project for each file system and adjust the properties from the Shares/General screen, as follows:

- For redo log file system, make the following property changes (as shown in the following graphic /#shares/shares=SID/redo+general.)
 - **Synchronous write bias** - This property is equivalent to the ZFS file system `logbias` property. Change this setting to "Latency."
 - **Database record size** - This property is equivalent to the ZFS file system `recordsize` property. Use the default value of 128 KB.
 - **Cache device usage** - This property is equivalent to the ZFS file system `secondarycache` property. Change this setting to "Do not use cache devices."

Properties Inherit from project

Mountpoint /export/SID/redo

Read only

Update access time on read

Non-blocking mandatory locking

Data deduplication

Data compression Off

Checksum Fletcher2 (Legacy)

Cache device usage Do not use cache devices

Synchronous write bias Latency

Database record size 128k

Additional replication Normal (Single Copy)

Virus scan

Prevent destruction

6. For the archive share, select the default values, except to set the Cache device usage to “Do not use cache devices” and Data compression to LZJB as follows.

Properties Inherit from project

Mountpoint /export/SID/arch

Read only

Update access time on read

Non-blocking mandatory locking

Data deduplication

Data compression LZJB (Fastest)

Checksum Fletcher2 (Legacy)

Cache device usage Do not use cache devices

Synchronous write bias Throughput

Database record size 128k

Additional replication Normal (Single Copy)

Virus scan

Prevent destruction

7. For the archive share, adjust the quota setting to limit disk space.

See graphic/#shares/shares=SID/arch+general :

Space Usage

DATA

Quota 100 G ▼

Include snapshots

Reservation 0 G ▼

Include snapshots

8. Mount the database file systems.

The file systems defined are to be mounted for the database usage. The list of the required mount options are in the Oracle database documentation and includes the `forcedirectio` option as set in the following typical example:

```
# mount -F nfs -o rw,bg,hard,nointr,rsize=32768,wsiz=32768,proto=tcp,noac,\
forcedirectio,vers=3 s7410-05a:/export/SID/index /u3/SID/index
```

Configuring an Oracle Solaris ZFS System for an Oracle Database

After you have reviewed the preceding planning section, use the steps below to configure Oracle Solaris ZFS file systems for an Oracle database on a Solaris server.

How to Configure an Oracle Solaris ZFS System for an Oracle Database

1. Verify your Solaris release information.

Confirm that you are running the latest Solaris 10 release. Apply Solaris 10 patch 142900-09 (for SPARC) or patch 142901-09 (for x86/x64) or later patch versions for any release prior to Solaris 10 10/09 release.

```
# cat /etc/release
Solaris 10 10/09 s10s_u8wos_08a SPARC
Copyright 2009 Sun Microsystems, Inc. All Rights Reserved.
Use is subject to license terms.
Assembled 16 September 2009
```

2. If needed, create LUNs for your Oracle Solaris ZFS storage pools.

Use your storage array tools to create LUNs that will be presented to the Oracle Solaris ZFS storage pool.

Or, consider using whole disks for your mirrored Oracle Solaris ZFS storage pools. For more information, see the Oracle Solaris ZFS Administration Guide.

If database activity is not high enough to justify the administrative task of creating many large and small LUNs for each database, you can configure the database on a single LUN provided

by a large shared storage array. When a single LUN is divided into 3 slices using `format(1m)`, you can have the two recommended storage pools as follows:

- Two slices for the redo log pool, one for the files and one for the separate log device
- One slice of the remaining LUN capacity for the main datafiles pool

The slices for the main datafiles and for the redo log files should be sized according to the requirements of the database administrator. The slice used as a separate log device is sized according to the maximum expected throughput to the redo log files. Only approximately 15 seconds of data is ever kept on the separate log device as a protection against catastrophic events. For a database generating at most 10 MB/sec of redo log activity, then a 150 MB slice is sufficient. Oracle Solaris ZFS enforces a minimum of 64 MB for a separate log device.

When creating slices with `format(1)`, be aware that alignment of the slices can have important consequences on performance of the storage array. Create slices that align with the internal block size used in the array. For simplicity, aligning slices on 1 MB boundary should work in most circumstances.

3. Create a storage pool for datafiles for tables, index, undo and temp data.

```
# zpool create dbpool c5t600A0B800029BA080000BBF4B208991d0
```

Consider creating a mirrored storage pool to provide a higher level of data redundancy. For example:

```
# zpool create dbpool mirror c1t226000C0FFA001ABd0 c2t216000C0FF80024Fd0
mirror c1t226000C0FFA001ABd2 c2t216000C0FF80024Fd2
```

```
# zpool status dbpool
pool: dbpool
state: ONLINE
scrub: none requested
config:
```

NAME	STATE	READ	WRITE	CKSUM
dbpool	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c1t226000C0FFA001ABd0	ONLINE	0	0	0
c2t216000C0FF80024Fd0	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c1t226000C0FFA001ABd2	ONLINE	0	0	0
c2t216000C0FF80024Fd2	ONLINE	0	0	0

```
errors: No known data errors
```

4. Create a storage pool for redo logs with a separate log device.

In this example, the disk volume is partitioned into two slices, a small slice, `s0`, in the 64 to 150 MB range, for the separate log device. The `s1` slice contains the remaining disk space for the redo log.

```
# zpool create redopool c5t600A0B800029BAD200000C6B4B25A77Ed0s1
log c5t600A0B800029BAD200000C6B4B25A77Ed0s0
```

```
# zpool status redopool
pool: redopool
state: ONLINE
scrub: none requested
```

```

config:

NAME                               STATE    READ WRITE CKSUM
redopool                            ONLINE      0    0    0
  c5t600A0B800029BAD20000C6B4B25A77Ed0s1 ONLINE      0    0    0
logs
  c5t600A0B800029BAD20000C6B4B25A77Ed0s0 ONLINE      0    0    0

errors: No known data errors

```

For databases with high redo log activity, such as a typical OLTP database with many commits, use a separate log device LUN.

5. Create a storage pool for the archivelog.

A system's internal disk can handle this type of load. In the slice method, a dedicated slice can be used for the archivelog pool. The archivelog file system can also be a dataset in the main dbpool.

```
# zpool create archivepool c3t0d0
```

6. Create the Oracle Solaris ZFS file systems and set the specific file system properties by using the following guidelines:

Create a separate file systems for redo, archive, undo, and temp database components using the default record size of 128 KB. The general rule is to set the file system `recordsize = db_block_size` for the file systems that contains Oracle datafiles. For table data and index components, create a file system with an 8 KB record size. Also consider providing metadata caching hints for your database file systems by using the `primarycache` property. For more information about Oracle Solaris Oracle Solaris ZFS file system properties, see the Oracle Solaris ZFS Administration Guide.

a) Create file systems for the table datafiles and index datafiles with an 8 KB recordsize.

Use the default value for `primarycache`.

```

# zfs create -o recordsize=8k -o mountpoint=/my_db_path/table dbpool/table
# zfs get primarycache,recordsize dbpool/table
NAME                PROPERTY    VALUE    SOURCE
dbpool/table        primarycache all      default
dbpool/table        recordsize  8K      local

# zfs create -o recordsize=8k -o mountpoint=/my_db_path/index dbpool/index
# zfs get primarycache,recordsize dbpool/index
NAME                PROPERTY    VALUE    SOURCE
dbpool/index        primarycache all      default
dbpool/index        recordsize  8K      local

```

On systems running the OpenSolaris release, also consider setting the `logbias` property to `throughput` when the table and index file systems are created.

b) Create file systems for temporary and undo tablespaces, using the default `recordsize` and `primarycache` values.

```

# zfs create -o mountpoint=/my_db_path/temp dbpool/temp
# zfs create -o mountpoint=/my_db_path/undo dbpool/undo

```

On systems running the OpenSolaris release, also consider setting the `logbias` property to `throughput` when the temporary and undo tablespaces file systems are created.

- c) Create a file system for redo logs in the redo pool. Use default file system values for **recordsize** and **primarycache**.

```
# zfs create -o mountpoint=/my_db_path/redo redopool/redo
```

On systems running the OpenSolaris release, use the default `logbias` value of `latency` when the redo log file system is created.

- d) Create a file system for archive log files in the archive pool, enabling compression, use the default value for **recordsize** and set **primarycache** to **metadata**.

```
# zfs create -o compression=on -o primarycache=metadata -o
mountpoint=/my_db_admin_path/archive archivepool/archive
# zfs get primarycache,recordsize,compressratio,compression,available,
used,quota archivepool/archive
```

NAME	PROPERTY	VALUE	SOURCE
archivepool/archive	primarycache	metadata	local
archivepool/archive	recordsize	128K	default
archivepool/archive	compressratio	1.32x	-
archivepool/archive	compression	on	local
archivepool/archive	available	40.0G	-
archivepool/archive	used	10.0G	-
archivepool/archive	quota	50G	local

Configuring Your Oracle Database on Oracle Solaris ZFS File Systems

After you have created your Oracle Solaris ZFS storage pools and Oracle Solaris ZFS file systems, create your Oracle database components.

Review the following Oracle database configuration recommendations:

- **Do not pre-allocate Oracle tablespaces** - As Oracle Solaris ZFS writes the new blocks on free space and marks free the previous block versions, the subsequent block versions have no reason to be contiguous. The standard policy of allocating large empty datafiles at the start of the database to provide continuous blocks is not beneficial with Oracle Solaris ZFS. Thus, we do not recommend pre-allocating large tablespaces in an Oracle database, but rather use Oracle Solaris ZFS's file system quotas and reservations. Then, you can use the datafiles auto extent feature to have high usage of the volume allocated to the tablespaces and keep enough storage pool free space (allocation at less than 80% capacity).
- **Oracle Large Object (LOB) Considerations** - The Oracle LOB, CLOB, NCLOB or BLOB data types are rarely used. Some applications have high LOB activities. For these applications, use the Oracle SQL syntax to set apart the LOB columns in specific tablespaces. The LOBs are frequently split over many Oracle data blocks. Because this type of data leads to large read and write I/Os, such as a full table scan, consider creating a separate file system and specify a larger record size for the LOB tablespace. You can use a Oracle Solaris ZFS record size of 128 KB if the average size of the LOB

is larger than 256 KB. No reason exists to have Oracle Solaris ZFS records aligned with Oracle LOB data.

You can define a specific tablespace for the LOB and use a 32 KB block size tablespace and it also requires some 32 KB cache in the SGA (System Global Area), for a data warehouse environment, by using syntax similar to the following:

```
# zfs create -o mountpoint=/my_db_path/lob dbpool/lob
```

In `init.ora` as follows:

```
DB_32K_CACHE_SIZE=100M
```

In SQL as follows:

```
CREATE TABLESPACE lob_example
  BLOCKSIZE 32K
  DATAFILE '/my_db_path/lob/lob01.dbf' SIZE 5G
  AUTOEXTEND ON NEXT 50M MAXSIZE 20G;

CREATE TABLE print_media_new
  ( product_id      NUMBER(6)
  , ad_id           NUMBER(6)
  , ad_sourcetext   CLOB
  , ad_finaltext    NCLOB
  , ad_photo        BLOB
  )
  LOB (ad_sourcetext, ad_finaltext, ad_photo ) STORE AS
  (TABLESPACE lob_example
  STORAGE (INITIAL 1M NEXT 1M)
  NOCACHE LOGGING);
```

The LOB storage clause can also be modified in a `ALTER TABLE` command.

Maintaining Oracle Solaris ZFS File Systems for an Oracle Database

Review the following steps to keep your Oracle database optimally maintained on Oracle Solaris ZFS file systems.

1. Monitor memory resources.

The following command gives the current memory size in bytes that is used as Oracle Solaris ZFS cache:

```
# kstat zfs::arcstats:size
```

Monitor Oracle Solaris ZFS cache sizes with the above command and readjust the `zfs_arc_max` parameter when needed. If the `vmstat` command shows always large free memory, you can also increase the value of `zfs_arc_max`.

2. Monitor disk space.

When a database is very active with non-negligible writes, you should reserve space to help ZFS find contiguous space to store new data. This reservation allows ZFS to reduce the number of write IOPS necessary to manage changing data. For a database that has a portion of very dynamic data, reserving space that amounts to 20% of the dynamic data should be beneficial. As a way to easily set and forget this rule of thumb, create an empty dataset with a reservation equal to the estimated 20% reserved space.

```
# zfs create pool/reserve
# set reservation=50G pool/reserve
```

Use Oracle Solaris ZFS quotas to avoid surprises. Because reserving space is suggested for an OLTP database, consider setting quotas on the main pool's file systems to guarantee that different databases do not overrun the shared resource.

For a data warehouse database, which is loaded once and then used in a read-only mode, then the 20% rule would apply only to the portion of data being loaded at any one time. The read-only portion of the database can grow to fill most of the available disk space.

Quotas and reservations are set on the file system. For example, set a 50 GB quota on the `archivepool/archive` file system.

```
# zfs set quota=50G archivepool/archive
# zfs get quota archivepool/archive
```

NAME	PROPERTY	VALUE	SOURCE
archivepool/archive	quota	50G	local

For more information about using Oracle Solaris ZFS quotas, see the Oracle Solaris ZFS Administration Guide.

3. Replicate Oracle database files.

The I/O load associated with copy operations as well as backups, are similar to a full scan: long logical sequential reads and are also subject to IOPS inflation as compared to traditional file system storage. The elapsed time for a database backup can be irregular due to the copy-on-write evolution of the on-disk format for datafiles. Copying the datafiles, although cumbersome, might help to relocate the blocks into a more continuous physical layout, particularly if the pool is left with a minimum of free disk blocks as suggested above.

If this scenario becomes a problem, consider creating a replicated environment and back up the replicated database.

You might use the Oracle Solaris ZFS `send` and `receive` features to keep replica database file systems as snapshot streams. Sending snapshot streams from a replica relieves the primary system from servicing those I/Os. Note that unlike full file scans, frequent Oracle Solaris ZFS `send` operations read data recently stored with some

physical locality relationship, completing its operations with comparatively fewer total physical IOPS than a full scan because it handles only the modified data.

A storage based replication strategy can also be used to preserve the primary disks' I/O capacity by running backups from a secondary storage subsystem. With storage based replication of an Oracle Solaris ZFS storage pool, you must configure all LUNs belonging to a storage pool to be in the same *coherency group* in the storage array and allow the storage pool to be imported on the backup server.

For more information about sending and receiving snapshots, see the Oracle Solaris ZFS Administration Guide.

Conclusion

Oracle Solaris ZFS is a robust file system that can be easily configured to provide optimal performance for an Oracle database.

This paper describes step-by-step instructions for Oracle database administrators and system administrators to configure ZFS pools and file systems for their Oracle database components on a Sun Storage 7000 system or on a traditional disk storage array, such as the Sun Storage 6000 Array.



Configuring Oracle Solaris ZFS for an Oracle Database

July 2010

Authors: Roch Bourbonnais, Alain Chéreau

Contributing Authors: Nicolas Houix and Cindy Swearingen

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com



| Oracle is committed to developing practices and products that help protect the environment

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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

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

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 0110