ORACLE
SOLARIS

An Oracle White Paper
May 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

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

- Planning Oracle Solaris ZFS for Oracle Database Configuration
- Configuring Oracle Solaris ZFS File Systems 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

A large number of enterprise customers today 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. Using the Oracle Solaris ZFS file system can reduce the cost and complexity of managing data and storage devices for hosting your 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 Oracle Solaris ZFS for Oracle Database Configuration

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

- **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 141444-09 (for SPARC) or patch 141445-09 (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.

  Some of the recommendations in this document refer to the latest OpenSolaris release and the required release is noted in those cases. The Sun Storage S7000 series runs a derivative of OpenSolaris so some of the features described below are available on this appliance.

- **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 when it is available as a NFS-shared file system.

- **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 flush write cache request issued by Oracle Solaris ZFS.

  - 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 as many LUNs as there are physical spindles in the storage array. Using too few large LUNs can cause Oracle Solaris ZFS to queue up too few read I/O operations to actually drive the storage to optimal performance. Conversely, using too 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 spare 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.

- **Storage pool considerations** - If you have traditional hardware RAID protection and you are comfortable with that level of protection, then use it. Otherwise, for the highest level of protection, we recommend using Oracle Solaris ZFS redundancy, such as a mirrored storage pool. With a mirrored storage pool, Oracle Solaris ZFS can automatically detect and repair data storage corruption. We do not recommend RAIDZ redundancy when performance and IOPS is the main criteria 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.
- **Match Oracle Solaris ZFS record size to Oracle database block size** - The general rule is to set `recordsize = db_block_size` for the file system that contains the Oracle data files.
- 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 data file, 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 data files 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.
- **Improve database writing and caching performance** - On systems running the current OpenSolaris release (including S7000), you can use the Oracle Solaris ZFS

`logbias`[1] property to improve performance when writing or caching database files. On systems running the Solaris 10 release since 10/09, you can use the `primarycache` property to control what is cached in main memory (the primary ARC).

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**
Data files	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 MB/s throughput capacity. 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**
Data files	128 KB	`throughput`	`all` (data and metadata)
Index data	db_block_size (8 KB, 16 KB, or 32 KB)	`throughput`	`all` (data and metadata)

- **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 data files, 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

[1] In the S7000 appliance user interface, the `logbias` property is labeled "Synchronous write bias"

possible to avoid the double write issue. Consult the Oracle Solaris ZFS Tuning wiki for details. `http://www.solarisinternals.com/wiki/index.php/Oracle Solaris ZFS_Evil_Tuning_Guide`

- **Use secondary cache (L2ARC)** - You can add LUNs as secondary cache (L2ARC) devices in a storage pool starting in the Oracle 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 determine which file systems will use the secondary cache and what the cache contents should be.

  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.

Configuring Oracle Solaris ZFS File Systems for an Oracle Database

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

How to Configure Oracle Solaris ZFS File Systems for an Oracle Database

1. **Verify your Solaris release information**.

Confirm that you are running the latest Oracle Solaris 10 release. Apply Oracle Solaris 10 patch 141444-09 (for SPARC) or patch 141445-09 (for x86/x64) or later patch versions for any release prior to Oracle 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 data file pool

The slices for the main data file 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 data files for tables, index, undo and temp data.**

```
# zpool create dbpool c5t600A0B800029BA0800000BBF4B208991d0
```

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
            c1t226000C0FFA001ABD0   ONLINE       0     0     0
            c2t216000C0FF80024Fd0   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
       c5t600A0B800029BAD200000C6B4B25A77Ed0s1   ONLINE       0     0     0
     logs
       c5t600A0B800029BAD200000C6B4B25A77Ed0s0   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. The archivelog file system can also be a dataset in the `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 data files. 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 data files and index data files with an 8 KB recordsize. Use the default value for `primarycache`.

```
# zfs create -o recordsize=8k -o mountpoint=/my_db_path/data dbpool/data
# zfs get primarycache,recordsize dbpool/data
NAME            PROPERTY      VALUE         SOURCE
dbpool/data     primarycache  all           default
dbpool/data     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 or on a S7000 system, also consider setting the `logbias` property to `throughput` when the data and index file systems are created.

b) **Create file systems for temporary and undo table spaces, 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 or on the S7000 appliance, also consider setting the `logbias` property to `throughput` when the temporary and undo table space 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 or on a S7000 system, use the default `logbias` value of `latency` when the redo log file system is created.

d) Create a file system for archivelog 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
```

7. **If running an Oracle database is the main activity on the system, tune the `zfs_immediate_write_sz` parameter**.

In this case, include the following parameter entry in the `/etc/system` file and reboot the system (Oracle Solaris 10 10/08 to 10/09 systems only, use the `logbias` property on systems that run the OpenSolaris release).

For SPARC systems:

```
set zfs:zfs_immediate_write_sz=8000
```

For x86 systems:

```
set zfs:zfs_immediate_write_sz=4000
```

For more information, see Planning Oracle Solaris ZFS for Oracle Database Configuration.

8. (Solaris 10 10/09 systems only): **If the storage pool is more than 80% full, consider tuning write activity.**

Adjusting the `metaslab_df_free_pct` parameter to 4 in the `/etc/system` file might help the write activity on full pools.

```
set zfs:metaslab_df_free_pct=4
```

For more information, see Planning Oracle Solaris ZFS for Oracle Database Configuration.

9. (Systems with HDS or EMC storage arrays): **Consider tuning storage array I/O queues**.

Oracle Solaris ZFS aggregates read and write I/O and manages the priority of I/O before sending it to the driver level, which handles the device. The `zfs_vdev_max_pending`

parameter defines the maximum number of I/Os that Oracle Solaris ZFS will send to any storage pool device.

In a legacy storage environment, the `ssd_max_throttle` and `sd_max_throttle` parameters define the maximum number of concurrent I/Os that the driver can send to the storage. By setting the `zfs_vdev_max_pending` default value equal to the value of the [s]`sd_max_throttle` parameter, we prevent Oracle Solaris ZFS from queuing I/O to yet another unnecessary SD layer.

If you have `ssd:ssd_max_throttle` or `sd:sd_max_throttle` in the `/etc/system` file in your existing environment, then set `zfs:zfs_vdev_max_pending` at the same value.

For example, if the storage array administrator asked for the following setting:

```
set ssd:ssd_max_throttle=20
```

Then, also set this parameter as follows:

```
set ssd:ssd_max_throttle=20
set zfs:zfs_vdev_max_pending=20
```

Setting this parameter allows Oracle Solaris ZFS to control each LUN queue. This means that the total number of pending I/O in the storage can grow as follows:

```
number of LUNs * Oracle Solaris ZFS_VDEV_MAX_PENDING
```

For more information, see Planning Oracle Solaris ZFS for Oracle Database Configuration.

10. **Allocate sufficient memory and swap resources**.

You can reduce Oracle Solaris ZFS memory consumption by tuning the `zfs_arc_max` parameter to a low value, but we still recommend provisioning 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 hold index files is the one that has the largest benefit from file system caching because it is the last one to invalidate in case of lack of memory.

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

The following example sets this parameter to 2 GB:

```
set zfs:zfs_arc_max=2147483648
```

or

```
set zfs:zfs_arc_max=0x80000000
```

To prevent applications from failing due to lack of memory, you must configure some amount of swap space. The amount of swap equivalent to all of system memory is always enough for this purpose. This swap space is not expected to be used, but is needed as a reservation area.

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

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 table spaces** - 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 data files 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 table spaces in an Oracle database, but rather use Oracle Solaris ZFS's file system quotas and reservations. Then, you can use the data files auto extent feature to have high usage of the volume allocated to the table spaces and keep storage pool free space to less than 80% capacity.
- **Oracle Large Object (LOB) Considerations** - The Oracle LOB, CLOB, NCLOB or BLOB data types are quite rarely used. Some applications have high LOB activities. For these applications, we can use the Oracle SQL syntax to set apart the LOB columns in specific tablespaces. The LOB are frequently large, many MB, and 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 requires also some 32 KB cache in the SGA (System Global Area), as we do in 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 Oracle Solaris ZFS database optimally maintained.

1. **Monitor disk space and memory resources**.

   Keep 20% free space in your Oracle Solaris ZFS storage pools.

   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. **Use Oracle Solaris ZFS quotas and reservations to keep free space in storage pools**.

   Oracle Solaris ZFS writing strategies change when the storage volume used goes over 80% of the storage pool capacity. This change can impact the performance of rewriting data files as Oracle's main activity. Keep more than 20% of free space is suggested for an OLTP database. Consider setting quotas on the main pool's file systems to guarantee that 20% free space is available at all time.

   For a data warehouse database, keep 20% free space in the storage pool as a general rule. Periodically copying data files reorganizes the file location on disk and gives better full scan response time. For a large data warehouse database, we can have a specific rule for read-only table spaces. When the data loading phase is ended, the table space is set to read only. We can then copy the data files of the table space in a storage pool dedicated to read-only table spaces, and for this type of usage, we can use more than 80% of the storage pool's capacity.

   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.**

   Copy operations, like 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 data files. Copying the data files, 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.

   If this scenario becomes a problem, consider creating a replicated environment and back up the replicated database. You can use any replication tool.

   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.

   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 tend to read data stored with some physical locality relationship, completing its operations with comparatively fewer total physical IOPS than a full scan.

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

Configuring Oracle Solaris ZFS for an Oracle Database
May 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