VERITAS™

Configuring and Tuning Oracle Storage with VERITAS Database Edition™ *for Oracle*

Best Practices for Optimizing Performance and Availability *for Oracle* Databases on Solaris

Table of Contents

Introduction

This paper offers guidelines for configuring and tuning storage-related parameters for Oracle databases on Solaris using VERITAS Database Edition™ *for Oracle*.

VERITAS Database Edition *for Oracle* is an integrated suite of storage-management software products that can enhance the performance, availability, recoverability and manageability of Oracle databases. Database Edition *for Oracle* gives database and system administrators unsurpassed flexibility in administering databases with critical availability requirements or large volumes of data. Administrators can use flexible volume-configuration capabilities to tailor and fine-tune storage for optimal performance or availability or both and to tune storage according to access characteristics. The management interfaces simplify the processes of managing critical or large databases.

This paper offers best practices for using the product to optimize Oracle database performance, availability and manageability. The topics covered include:

- Configuring new databases

- Upgrading existing database to use VERITAS Database Edition *for Oracle*

- Tuning configurations for optimal performance and availability

There is no single "right" way to configure database storage — the specifics depend on performance requirements, access characteristics, availability needs and hardware costs. There are even different theories on the right approach to configuration, ranging from striping and mirroring everything with the same layout to tuning storage for specific data files.

This paper offers guidelines based on generally accepted best practices and VERITAS Software's own performance testing. These are meant to guide your decisions, but cannot replace performance testing using your actual hardware and data. The tuning section provides information to help you optimize performance in your environment.

VERITAS Database Edition *for Oracle* Components

VERITAS Database Edition *for Oracle* is an integrated suite of industry-leading VERITAS storage-management and high-availability technologies, engineered specifically to improve the performance, availability and manageability of Oracle database servers running on Solaris. Although the Database Edition is available for other operating system platforms, this paper is Solaris specific.

There are two versions of this product suite. VERITAS Database Edition *for Oracle* enables single-server Oracle database configurations to meet the performance and availability demands of enterprise database environments. The Database Edition/HA *for Oracle* delivers a high level of availability by integrating VERITAS Cluster Server™ for automatic failure detection and failover support in a clustered environment.

The Database Edition *for Oracle* solution components include:

VERITAS File System™, with Quick I/O, Cached Quick I/O, QuickLog and Storage Checkpoint technologies

VERITAS Volume Manager™

VERITAS Cluster Server with the Database Edition/HA version of the product, including database-specific cluster server agents.

These components are described in more detail below.

VERITAS File System

VERITAS File System is an extent-based, intent-logging file system that offers high performance and quick recovery. VERITAS File System is particularly useful in environments that require high performance and availability for large volumes of data. The online administration capabilities of VERITAS File System support most frequently scheduled maintenance tasks without interrupting access to data.

enhancements, such as Quick I/O, that offer raw-partition performance with the ease of file-system management, making it not only viable but preferable to store Oracle databases in file-system files.

The following features contribute to improved database performance, availability and manageability:

Extent-based allocation: VERITAS File System allocates disk space in large contiguous areas (extents), which reduces I/O operations required for reading or writing large amounts of data.

Fast file-system and database recovery: VERITAS File System's journaling functionality records structural changes and generally provides file-system recovery within seconds following a system crash or reboot.

Online administration: Using VERITAS File System, administrators can resize and defragment file systems while data remains online and available.

Large file support: When used in conjunction with VERITAS Volume Manager, VERITAS File System can support files of up to a terabyte in size.

Quick I/O

The Quick I/O feature delivers raw disk performance to databases running on VERITAS File System files. Quick I/O allows a regular, preallocated VERITAS File System file to be accessed as a raw character device. It improves throughput for Oracle databases built on the VERITAS File System, providing performance at least equivalent to that of raw devices, with the manageability of file-system files.

Quick I/Os improve database performance by:

- Supporting kernel asynchronous I/O (KAIO): Asynchronous I/O enables the system to handle multiple I/O requests simultaneously. Operating systems like Solaris offer kernel support for asynchronous I/O on raw devices, but not files. Quick I/O lets the database server take advantage of kernel-supported asynchronous I/O on file-system files accessed using the Quick I/O interface.

- Supporting direct I/O (without double buffering): Using Quick I/O, data is copied directly between the database cache and disk, without using the system buffer cache. This frees memory that can then be used by the database cache to improve transaction performance.

- Avoiding kernel write locks: When database I/O is performed using the write system call, each system call acquires and releases a write lock inside the kernel. This lock prevents multiple simultaneous write operations on the same file. Because database systems usually implement their own locks for managing concurrent access to files, per file writer locks serialize I/O operations unnecessarily. Quick I/O bypasses file-system locking and lets the database server control data access.

Cached Quick I/O

Cached Quick I/O enhances system performance by leveraging large system memory to buffer frequently accessed data. Cached Quick I/O provides an efficient, selective buffering mechanism for asynchronous I/O.

- For read operations, Cached Quick I/O caches database blocks in the system buffer cache, which can reduce the number of physical I/O operations and therefore improves the read performance.

- For write operations, Cached Quick I/O uses a direct-write, copy-behind technique to preserve its buffer copy of the data. After the direct I/O is scheduled and while it is waiting for the completion of the I/O, the file system updates its buffer to reflect the changed data being written out.

- For online transaction processing, Cached Quick I/O achieves better than raw device performance in database throughput on large platforms with very large physical memories.

- Cached Quick I/O also helps sequential table scans because of the read-ahead algorithm used in the VERITAS File System. For most queries that require sequential table scans, Cached Quick I/O can significantly reduce the query response time.

In an OLTP benchmark test derived from the industry-standard TPC-C benchmark, Cached Quick I/O outperforms the UNIX File System using direct I/O by 483 percent to 605 percent, depending on the size of the Oracle SGA. It also outperforms raw

disk I/O by up to 180 percent, again depending on the size of the Oracle SGA. The benefits are greatest when the file system has a large page cache. These findings affect Cached Quick I/O tuning, described in a later section.
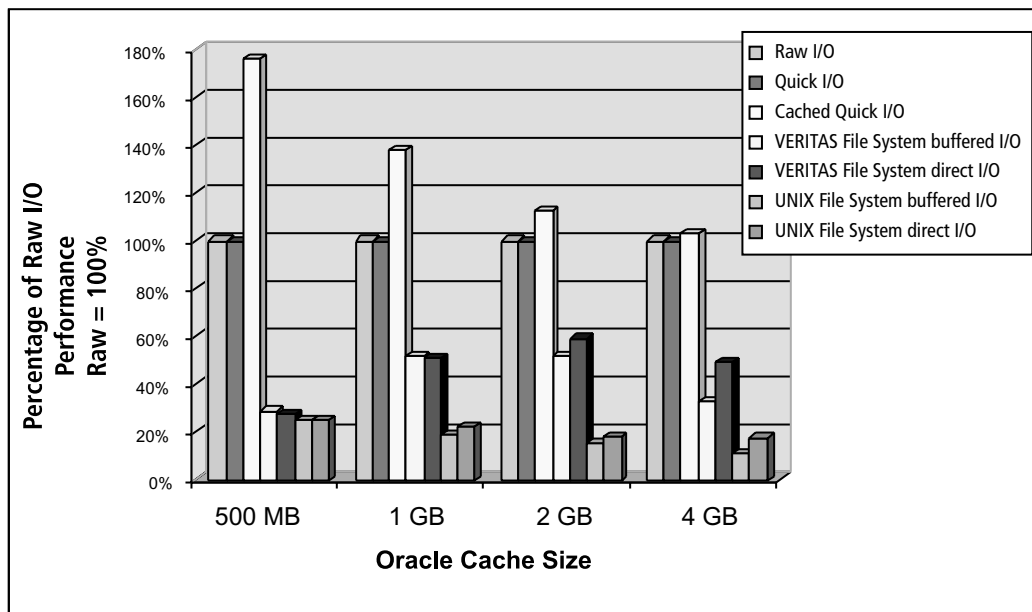
Figure 1: Comparing I/O performance

The test configuration included a Sun Ultra Enterprise 10000 with 10 CPUs and 6 GB of RAM running Solaris 7 (64 bits), VERITAS Database Edition 2.2. for Oracle, Oracle 8.1.5 (64 bits) and an OLTP test derived from a 2000 warehouse TPC-C benchmark.

QuickLog

VERITAS File System maintains an intent log. The QuickLog feature moves the file-system intent log to dedicated storage, physically separate from the file system. It reduces disk head movements between the intent log and the file-system data areas; the disk head is always in position to write the next log record sequentially.

Storage Checkpoint™ and Storage Rollback

VERITAS File System offers a snapshot capability called Storage Checkpoint. VERITAS Storage Checkpoint™ creates an exact image of the VERITAS File System instantly. VERITAS Storage Checkpoint is disk and I/O efficient, providing a consistent, stable, point-in-time view of the VERITAS File System using a copy-on-write technique. Because Storage Checkpoint offers a cost-effective mechanism to provide snapshots of database images, it is possible to keep multiple versions of Storage Checkpoint available.

Storage Checkpoint serves as point-in-time images of files systems or on-disk backup images. They can be used to quickly "roll back" a database state to the exact time at which a particular Storage Checkpoint was taken. Storage Rollback is the restore process for these on-disk backups. Storage Rollback requires "rolling back" data blocks contained in Storage Checkpoint onto the primary file system. This is done by copying before images from appropriate Storage Checkpoint back to the primary file system. Storage Rollback is a fast-restore option available to DBAs for user errors. For more information, see the paper "Guidelines for Using Storage Checkpoint and Storage Rollback with Oracle Databases."

VERITAS Volume Manager

VERITAS Volume Manager builds virtual devices called volumes using physical storage devices. Volumes are accessed by a UNIX file system, a database or other applications in the same way that physical disk partitions would be accessed. Using volumes, VERITAS Volume Manager provides a number of administrative benefits for databases:

- A wide variety of storage layouts: VERITAS Volume Manager volume types include spanned, striped, mirrored, RAID-5 and striping and mirroring combinations (RAID-0+ or RAID-1+0).

- A wide range of supported devices: VERITAS Volume Manager supports a broad range of storage devices, including many hardware RAID devices.

- Online relayout: Administrators can reconfigure volumes without system or database downtime.

- Hot relocation: VERITAS Volume Manager can automatically restore redundancy in mirrored volumes when a single disk in the volume fails.

VERITAS Cluster Server

VERITAS Database Edition/HA *for Oracle* includes VERITAS Cluster Server, a highly scalable and configurable failover solution for clusters of up to 32 nodes. VERITAS Cluster Server provides instant and automated application failover capabilities, including rolling and cascading failover, to keep critical applications up and running in case of problems.

Database Edition/HA for Oracle includes the integrated VERITAS Cluster Server product, as well as application-specific agents for QuickLog and the Oracle database server. These agents monitor system components; when they detect a failure, VERITAS Cluster Server implements the failover policies and procedures that you have defined.

Setting Up a Database with the Database Edition

The following steps provide a roadmap for setting up a database with reasonably good performance. This configuration can be tailored to fit any specific environment.

The steps are as follows:

1. Create a disk group.

2. Select volume layouts.

3. Create file systems for database files.

4. Create database files with preallocated space.

1. Creating a Disk Group

Before creating file systems for a database, set up a disk group for each database. A disk group lets you group disks, volumes, file systems and files that are relevant to a single database into a logical collection for easy administration. Because you can move a disk group and its components as a unit from one machine to another, you can move an entire database if all the configuration objects of the database are in one disk group. This capability is useful in a failover situation.

Follow these guidelines when creating disk groups for use with databases:

- Never put all the disks in the rootdg disk group. The rootdg disk group cannot be moved and has certain limitations.

- Only disks that are online and do not already belong to a disk group can be used to create a new disk group.

- Create one disk group for each database.

- The disk group name must be unique. Name each disk group using the Oracle database instance name specified by the environment variable `$ORACLE_SID` and a dg suffix. The dg suffix helps identify the object as a disk group. Also, each disk name must be unique within the disk group.

- Never create database files using file systems or volumes that are not in the same disk group.

You can use either the graphical Storage Administrator or the vxdg line-mode command to create disk groups and add disks to disk groups.

2. Selecting a Volume Layout

The proper selection of storage layouts provides optimal performance for the database workload. The most important factor in database performance is usually the table placement on the disks. Disk I/O is a determining factor for a database's performance. Having a balanced I/O load usually means optimal performance. Designing a disk layout for the database objects to achieve a balanced I/O is a crucial step in configuring a database.

When deciding where to put tablespaces, it is often difficult to anticipate future usage patterns. VERITAS Volume Manager provides flexibility in the initial storage configuration, as well as easy reconfiguration to support future growth.

Using striped volumes, I/O can be balanced across multiple disk drives. For most databases, ensuring that different database files and tablespaces are distributed across the available disks may be sufficient. Striping also helps sequential table scan performance. When a table is striped across multiple devices, a high transfer bandwidth can be achieved by setting the Oracle parameter DB_FILE_MULTIBLOCK_READ_COUNT to a multiple of the full stripe size divided by DB_BLOCK_SIZE.

Stripe Sizes

When creating a striped volume, you need to decide the number of columns to form a striped volume and the stripe unit size. You also need to decide how to stripe the volume. You may stripe a volume across multiple disk drives on the same controller or across multiple disks on multiple controllers. By striping across multiple controllers or disks, disk I/O can be balanced across multiple I/O channels. The decision is based on the disk and controller bandwidth and the database workload.

Stripe size should be a multiple of the OS (or logical volume) block size. For most OLTP databases, you should use the default stripe unit size of 64K or less for striped volumes and 16K for RAID-5 volumes. Decision Support (DSS) systems may perform better with larger stripe sizes, such as 128 or 256K.

Another school of thought named SAME, for Stripe And Mirror Everything—suggests that you should stripe and mirror everything using 1-Mb block sizes. For more information on the SAME methodology, see http://technet.oracle.com/deploy/availability .

Mirroring and RAID-5

Both mirrored and software-based RAID-5 volumes allow continued access to data in the event of a single disk failure.

VERITAS Volume Manager supports a combination of software-based mirroring and striping (RAID-0+1 or RAID-1+0). This is recommended for most databases, because it offers both performance and protection from disk failure. If disk hardware cost is a significant concern, consider RAID-5 volumes.

Software-based RAID-5 configurations have certain performance implications you must consider. Writes to RAID-5 volumes require parity bit recalculation, which adds significant I/O and CPU overhead. This overhead can cause considerable performance penalties in OLTP workloads. However, if the database has a high read ratio, RAID-5 performance is similar to that of a striped volume.

Volume Configuration Guidelines

- Put the redo logs on a file system created on a striped and mirrored (RAID-0+1) volume separate from the user tablespaces or data files. Stripe multiple devices to create larger volumes if needed. Use mirroring to improve reliability. Do not use VERITAS Volume Manager RAID-5 for redo logs.

- When normal system availability is acceptable, put the tablespaces on file systems created on striped volumes for most OLTP workloads.

- When mirroring, use mirrored plexes on different devices, preferably on different controllers. Do not disable the hot relocation feature.

- Create striped volumes across at least four disks. Try to stripe across disk controllers. For sequential scans, do not stripe across too many disks or controllers. The single thread that processes a sequential scan may not be able to keep up with the disk speed.

- Remember that the mean time between failure decreases with each disk added to a striped device. A RAID-5 device can tolerate only one disk crash.

- For most workloads, use the default 64K stripe unit size for striped volumes and 16K for RAID-5 volumes.

- When system availability is critical, use mirroring for most write-intensive OLTP workloads. Turn on Dirty Region Logging (DRL) to allow fast volume resynchronization in the event of a system crash.

- When system availability is critical, use RAID-5 or mirrored/striping combinations for most read-intensive OLTP workloads to improve database performance and availability. Use RAID-5 logs to allow fast volume resynchronization in the event of a system crash.

- For most decision-support workloads where sequential scans are common, try experimenting with different striping strategies and stripe unit sizes. Put the most frequently accessed tables or tables that are accessed together on separate striped volumes to improve the bandwidth of data transfer.

- If you plan to use QuickLog, use separate, mirrored disk for the QuickLog devices.

3. Creating a VERITAS File System

Once the volumes are created, create the file system using the mkfs command.

Follow these guidelines when creating file systems *for Oracle* databases:

- Specify the maximum block size and log size when creating file systems for databases.

- Never disable the intent logging feature of the file system. Use the QuickLog feature to speed log writing performance.

- For redo logs, create a single file system using a simple (and mirrored, if necessary) volume. Put the other tablespaces and database files on a single file system created on a striped, striped and mirrored, or RAID-5 volume.

- Use the mount points to name the underlying volumes. For example, if a file system name /db01 is to be created on a mirrored volume, name the volume db01 and the mirrors db01-01 and db01-02 to relate to the configuration objects.

- When possible, balance data file placement on file systems in terms of the amount of updates/inserts occurring to the data files.

We recommend that you enable the QuickLog feature, which moves the intent log to a separate physical device, reducing potential disk head movements for writing log records. Using QuickLog in database environments can help minimize the copy-on-write performance overhead of Storage Checkpoint and Storage Rollback and can increase overall I/O performance from 5 to 20 percent.

To enable QuickLog, you will need to do the following:

1. Create a QuickLog device. Use mirrored volumes for reliability. We recommend you use 2-GB logs for optimal performance if you are logging multiple file systems. Use the VMSA interface to create the QuickLog device.

2. Enable logging to the QuickLog device. Either use the VMSA, or use a command-line operation:

```
# qlogenable [ qlog_device] / mount_point
```

4. Creating Database Files Using Quick I/O

Quick I/O allows the database to access regular files on a VERITAS File System as raw character devices. To use Quick I/O, you must do the following:

1. Preallocate files on a VERITAS File System.

    As with raw devices, Oracle does not "know" how to resize a Quick I/O file. As a result, you must preallocate the Quick I/O file.

    Preallocating database files using setext or qiomkfile allocates contiguous space for the files. The file-system space-reservation algorithms attempt to allocate space for the entire file as a single contiguous extent. When this is not possible because of lack of space availability on the file system, the file is created as a series of direct extents. Accessing a file using direct extents is

inherently faster than accessing the same data using indirect extents. Internal tests have shown 5 to 8 percent performance degradation in online transaction processing throughput when using indirect extent access. In addition, this type of preallocation causes no fragmentation of the file system.

2. Use the `::cdev:vxfs:` Quick I/O file naming extension to access files.

   A VERITAS File System file can be accessed using two interfaces: the regular file interface and a device file interface. The device file interface, which accesses a regular file as a raw character device, is achieved by using the naming extension. For example, a file named system.dbf can be accessed as a raw character device when the name system.dbf::cdev:vxfs: is used for database access.

Converting Oracle Database Files on VERITAS File System to Use Quick I/O

The Database Edition *for Oracle* provides scripts to change Oracle database files to use Quick I/O. You can use the `getdbfiles.sh` and `convertdbfiles` scripts, available in the /usr/sbin directory, to extract and convert Oracle database files to use Quick I/O. The `convertdbfiles` script will only convert regular files on the VERITAS File System or links that point to regular files on the VERITAS File System.

Enabling Cached Quick I/O

Caching for Quick I/O files can be enabled online when the database is running. Enabling caching for the Quick I/O files can affect database performance. Read the section Tuning Cached Quick I/O before enabling caching for any Quick I/O files.

Enable caching for Quick I/O files using the following steps:

1. Setting the file system Cached Quick I/O flag, which enables Cached Quick I/O for all files in the file-system

2. Monitoring the file system statistics to determine which files benefit and which files do not benefit from Cached Quick I/O

3. Disabling Cached Quick I/O on the individual Quick I/O files that do not benefit from caching

Quick I/O must be enabled on the file system before enabling caching. This is done automatically at file-system mount time, after Quick I/O is installed on the system.

Setting the file-system Cached Quick I/O flag caching enables caching for all files in the file system. Disable Cached Quick I/O on individual Quick I/O files that do no benefit from caching to avoid consuming memory unnecessarily. (See Tuning Cached Quick I/O for tuning suggestions.)

Upgrading/Converting from Existing Database Configurations

The following sections describe the processes of converting databases from other file system configurations to the VERITAS File System using Quick I/O.

In general, the method of choice, regardless of the source file system, is to perform the conversion when the applications are stopped and the database is closed. The main constraints in this case are the type of alternate storage media used, the amount of storage media available and the speed of the media.

The fastest approach is to use a copy of the disk farm for the conversion; this has the advantage of providing a complete copy of the database as a backup. The other alternative is to copy all data to tape, convert the files and restore everything from tape. This takes longer to perform.

High-end databases with strict downtime restrictions can maintain a running database during conversion if the database uses disk mirroring. The general procedure is as follows.

1. Run the primary database in archive log mode.

2. Make and verify alternate backups.

3. Separate (split) the mirrors while the database is in online backup mode.

4. Convert the secondary side database storage.

5. Stop the database.

6. Save the current redo and control files to alternate media.

7. Copy them to the converted second mirror.

8. Use the converted mirror as the primary and archive logs to "recover" the database to the current point in time.

9. Perform another backup to alternate media.

10. Resilver the mirror, using the converted side as the primary side.

This procedure, developed in part by the Oracle Large Systems Support group working with VERITAS, should be used only for databases with strict downtime requirements and should be thoroughly tested. You should evaluate whether the time to roll forward through the archive logs for the database is actually less than the time required to convert the data files using traditional techniques. One advantage of this approach is that the primary mirror is still intact and could be used to bring the database online quickly should a problem arise with the convert/restore operation.

The following describe general conversion processes for traditional file-system storage and raw device storage.

Upgrading from Traditional File Systems

If you are using the UNIX file system, use the following procedure to upgrade each file system used by the database to a VERITAS File System with Quick I/O. (Do not upgrade the root file system to VERITAS File System.)

1. Shut down the database and create a backup of the UNIX file system.

2. Unmount the UNIX file system.

3. Create a VERITAS file system of the same size as the original UNIX file system, using the mount point where the UNIX file system was originally mounted.

4. Preallocate Quick I/O files using qiomkfile.

5. Restore the backup to the Quick I/O files in the VERITAS File System.

6. Restart the database.

Upgrading from Raw Devices

If the database is currently using raw disks or volumes, use the following procedure to use the Quick I/O feature (for simplicity, the procedure provided assumes the database runs on a single file system):

1. Create a VERITAS file system using a size that is 10 percent larger than the original database or raw device size. You can create more file systems based on your performance and availability requirements.

2. Shut down the database.

3. Preallocate Quick I/O files using qiomkfile.

4. Copy each raw device file to the new file system. For example, use the dd command to copy the file /dev/rdsk/c0t1d0 to /db01/dbfile:

```
$ dd if=/dev/rdsk/c0t1d0 of=/db01/dbfile bs=128k
```

5. If the database uses symbolic links to access the database files, change the symbolic links to point to the Quick I/O files. For example, if the database has a datafile specification /data/file1 that was linked to /dev/rdsk/c0t1d0, change it to point to the new Quick I/O file:

```
$ rm /data/file1
$ ln -s /db01/dbfile /data/file1
```

6. If the database was using absolute paths to access the database files, rename each file within Oracle before bringing the database online. For example:

```
$ svrmgrl
SVRMGR> connect internal;
SVRMGR> startup mount;
SVRMGR> alter database rename file <filename> to <newfilename>;
```

7. Restart the database. For example:

```
SVRMGR> alter database open;
SVRMGR> exit
```

Optimizing Oracle Performance with Database Edition

General Best Practices

Use Striped Volumes for Performance

In general, the best way to achieve optimal OLTP throughput is by balancing the I/O across as many disks as possible. This can be achieved by creating the database on a file system on a striped volume using all available disks. Because redo logs tend to be accessed sequentially, one may see better performance by putting redo logs on a separate file system using a simple or mirrored volume on a dedicated disk.

You should at the least mirror the volume used by the redo logs.

Mirroring/striping combinations offer the highest combination of performance and availability for data files. RAID-5 provides a more cost-effective approach, requiring less hardware, with some trade-off in terms of performance and availability.

Use Quick I/O

The default traditional UNIX file system supports buffered I/O, in which data blocks are buffered in the operating system kernel cache. Because the database maintains its own cache, buffering data in the kernel actually increases CPU overhead for database access. In addition, UNIX semantics require that only one writer have the exclusive right to update a file, forcing other read/write operations on the same file to wait. As a result, a database on file-system files typically performs slower than a database on raw devices.

Quick I/O offers raw-partition performance for databases on file-system files. Quick I/O has I/O characteristics and performance similar to those of raw I/O, such as a higher degree of read/write parallelism with less CPU overhead. It typically sustains higher transaction throughput than buffered I/O in an OLTP environment.

Tune the Database Buffer Cache

When using Quick I/O, tune the database cache as if raw partitions are being used. You can allocate more memory for the database buffer cache, which typically improves OLTP performance.

Use Cached Quick I/O

If additional memory is available on the system, enable Cached Quick I/O for files that can benefit from the caching (typically files under read-intensive workloads).

Tuning VERITAS File System

VERITAS File System provides a rich set of tuning options to optimize file-system performance for different application workloads. Most tuning options have little or no impact on database performance when using Quick I/O. However, you can

gather file-system performance data using Quick I/O and use this information to adjust the system configuration to make the most efficient use of system resources.

To obtain file I/O statistics, use the `qiostat` command, which provides access to information on Quick I/O files on VERITAS File System files. The command reports statistics on the activity levels of files from the time the files are first opened using the Quick I/O interface. The accumulated `qiostat` statistics are reset once the last open reference to the Quick I/O file is closed.

The `qiostat` command displays the following I/O statistics:

- Count of read and write operations
- Count of data blocks (sectors) transferred
- Average time spent on read/write operations.

If you are using Cached Quick I/O, invoke `qiostat` with the `-l` option (for long form) to display caching statistics as well. The command monitors file I/O activity, volume I/O activity and raw disk I/O activity.

1. Use `qiostat -r` to clear existing statistics; then let the database run for awhile with a typical workload period.

2. Then use the `qiostat` command to display active file I/O statistics. You can indicate a time interval with the `-I` option.

Files with an unusually large number of operations or excessive read/write times are "hot" files. Try moving these files or file systems to different disks or changing the layout to balance I/O load.

Tuning Cached Quick I/O

Cached Quick I/O provides significant performance benefits on both 32-bit and 64-bit Oracle. When 32-bit Oracle is used, the largest SGA size that Oracle can allocate is about 4 GB. If the system has a large amount of memory, Cached Quick I/O provides a second-level cache, which may improve performance by caching data blocks that could not be cached in Oracle. In VERITAS testing in the 32-bit Solaris 7 environment, Cached Quick I/O delivered up to a 159 percent performance gain over raw partitions. The benefit for 64-bit Oracle, although diminished, is still significant.

As the Oracle SGA size approaches the total system memory size (8 percent of total memory size or more), the benefits of Cached Quick I/O disappear. To achieve the optimal performance using Cached Quick I/O, you may need to tune the ratio between the Oracle SGA and the OS cache

Deciding which files would benefit from Cached Quick I/O is an iterative process that varies with each application. Turning caching on for all the database files can cause degraded performance because of double buffer copying. You should collect and analyze the caching statistics before you turn the Cached Quick I/O on for any of the database files.

Cached Quick I/O can be enabled or disabled while the database is online and can be easily customized for your environment. For example, you could disable Cached Quick I/O for certain files that store historical performance data during normal working hours when the database supports a large number of concurrent users. Then, you could enable Caching during off-peak hours when the same files are scanned for generating reports. You can also use scripts to automate enabling and disabling Cached Quick I/O on a per file basis, favoring different sets of files. In the hands of the database administrator who understands the applications and the types of workloads on the database, this is a powerful tuning tool that can be used to maximize performance by fully using all the resources available on the system.

Use the following steps to identify candidates for Cached Quick I/O:

1. Turn on caching for all files in the file systems being used with the vxtunefs command.

2. Run qiostat -r to rest the counters.

3. Run the machine under full, normal load for a period of time long enough to get a good usage sample.
   Verify that the system has reached a steady state before measuring throughput. It is common to see a longer ramp-up time when using Cached Quick I/O compared with Quick I/O.

4. Run qiostat -l to report caching statistics. For example:

   ```
   $ qiostat -l /db02/*.dbf
   ```

5. Analyze the output to find candidates by examining the cache hit ratio and studying read and write operations.

Identify those files with a cache hit ratio over a given threshold, such as 20 percent. Of those files, determine if the file experiences more reads or writes. The biggest gains can be realized when Cached Quick I/O is used for files that are read more than written.

For example, the `qiostat -l` command output might look like the following:

```
                          OPERATIONS              FILE BLOCKS          AVG TIME(ms)
                      CACHE    STATISTICS
FILE NAME             READ     WRITE          READ      WRITE       READ     WRITE
                      CREAD    PREAD          HIT       RATIO
/db01/cust.dbf        17128     9634         68509      38536       24.8      0.4
                      17124    15728          8.2
/db01/system.dbf          6        1            21          4       10.0      0.0
                          6        6          0.0
/db01/stk.dbf         62552    38498        250213     153992       21.9      0.4
                      62567    49060         21.6
```

Using this example, it is clear that the file `/db01/system.dbf` does not benefit from caching. In addition, very little I/O is performed on the file during the sampling duration.

However, the file `/db01/stk.dbf` has a cache hit ratio of 21.6 percent, which means the database can benefit from caching the file in the file system. When you compare the number of reads and writes for this file, you see that the number of reads is roughly twice the number of writes. On the basis of these two statistics, stk.dbf is a prime candidate for Cached Quick I/O. You can then enable or disable Cached Quick I/O for each file using the `qioadmin` command.

Tuning VERITAS Volume Manager

The Volume Manager is tuned for most configurations, ranging from small systems to larger servers. On smaller systems with less than about a hundred drives, tuning should not be necessary, and the Volume Manager should be capable of adopting reasonable defaults for all configuration parameters. On very large systems, however, there may be configurations that require additional tuning of these parameters, both for capacity and for performance reasons.

If the database is created on a single file system on a single volume, there is typically no need to monitor volume I/O statistics. If the database is created on multiple file systems on multiple volumes, or if the volume configurations have changed over time, you may want to monitor volume I/O statistics for the databases.

The `vxstat` command provides access to information for activity on volumes and their components under VERITAS Volume Manager control. The vxstat command reports statistics that reflect the activity levels of VERITAS Volume Manager objects since boot time. Statistics for a specific VERITAS Volume Manager object or all objects can be displayed at one time. Use the `-g` option to specify the database disk group to report statistics for objects in the database disk group.

VERITAS Volume Manager records the following I/O statistics for reporting:

- Count of operations

- Number of blocks transferred

- Average operation time.

VERITAS Volume Manager records the preceding three pieces of information for logical I/Os, including reads, writes, atomic copies, verified reads and verified writes for each volume and its components. VERITAS Volume Manager also maintains other statistical data such as read failures, write failures, corrected read failures and corrected write failures. In addition to displaying volume statistics, the vxstat command is capable of displaying more detailed statistics on the volume components.

Use `vxstat -r` to clear all statistics. This can be done for all objects or only specified objects. Resetting just before a particular operation makes it possible to measure the impact of that particular operation afterward.

Use the `vxstat` command to identify volumes with excessive activity, after which you can reorganize or move these volumes.

Tuning Oracle Databases

This section describes some of the Oracle parameters you can tune to improve the Oracle database performance when using Quick I/O.

Optimal Database Block Size

The database block size must be a multiple of the file-system block size. For databases supporting online transaction processing workloads, TPC-C benchmark tests provide evidence that the optimal Oracle block size is 2K for Quick I/O and 8K for buffered or direct I/O. Because the file system block size does not seem to affect database performance, using the default size is recommended. The default file-system block size varies with the overall size of the file system.

Sequential Table Scans

Quick I/O in its default mode performs all I/O as direct I/O. In the case of single-threaded sequential scans, common in decision-support-system workloads, using buffered reads can yield better performance. Because the file system detects these sequential reads and performs read-aheads, the next few blocks that are requested by Oracle are readily available in the system buffer cache and are simply copied to the Oracle system global area (SGA). Because access from memory is inherently faster than access from disk, a significant reduction in response time is possible.

Use one of the following two methods to improve performance when handling large sequential scans with Quick I/O:

1. Use the Oracle parallel query process to break the single large scan into multiple smaller scans. This is done by setting the Oracle parameters PARALLEL_MAX_SERVERS, PARALLEL_MIN_SERVERS, and SORT_AREA_SIZE suitably.

   Note: Consult the Oracle documentation for your system and version of Oracle, and use the settings recommended for these parameters when provided.

2. An alternative method is to enable Cached Quick I/O for the files that would be read by the Oracle sequential scan process. Cached Quick I/O enables buffered reads, and the automatic file system read-ahead helps lower response times by preloading data.

Asynchronous I/O

Asynchronous I/O allows the Oracle DBWR process to schedule multiple I/Os without waiting for the I/O to complete. When the I/O completes, the kernel notifies the DBWR process using an interrupt.

Quick I/O supports kernel asynchronous I/O, reducing CPU use and improving transaction throughput. Enabling the following parameters lets Oracle take advantage of asynchronous I/O and avoids having to configure multiple DBWR processes.

- Set DISK_ASYNCH_IO to TRUE in Oracle8 or Oracle8i databases (TRUE is the default setting).

- Set USE_ASYNC_IO to TRUE in Oracle7 databases (FALSE is the default setting).

DB_BLOCK_BUFFERS

The UNIX buffer cache plays an important role in performance when using UNIX File System in buffered I/O mode. When using Quick I/O, the database cache must be tuned as if raw devices are being used. You can allocate more memory to the database buffer cache because Quick I/O bypasses the file-system cache to improve database performance. Memory pages normally allocated to the file-system cache can be allocated to the database buffer cache (SGA). Allocate more memory for the Oracle buffer cache by increasing the value of the DB_BLOCK_BUFFERS parameter in the Oracle init.ora file for your database.

DB_FILE_MULTIBLOCK_READ_COUNT

The DB_FILE_MULTIBLOCK_READ_COUNT parameter specifies the maximum number of blocks Oracle reads in one I/O operation during a sequential scan. When the file system is created on a striped volume, set this parameter to a value that is a multiple of the full stripe size divided by DB_BLOCK_SIZE. Using a full stripe size allows the read operations to take advantage of the full bandwidth of the striped disks during sequential table scan.

Set the DB_FILE_MULTIBLOCK_READ_COUNT to a value that is a multiple of `(read_pref_io*read_nstream)/DB_BLOCK_SIZE.` The value should not exceed the value of `max_direct_iosz/DB_BLOCK_SIZE.`

Use the `vxtunefs` command to display the value of read_pref_io, read_nstream, and max_direct_iosz. For example:

```
# vxtunefs /db01
```

The `vxtunefs` command displays output similar to the following:

```
Filesystem i/o parameters for /db01
read_pref_io = 65536
read_nstream = 4
read_unit_io = 65536
write_pref_io = 65536
write_nstream = 4
write_unit_io = 65536
pref_strength = 10
buf_breakup_size = 131072
discovered_direct_iosz = 262144
max_direct_iosz = 2097152
default_indir_size = 8192
```

For a description of these parameters, refer to the `vxtunefs(1M)` manual page.

DB_FILE_DIRECT_IO_COUNT (Oracle8 and Oracle8i only)

The DB_FILE_DIRECT_IO_COUNT parameter specifies the number of blocks used for I/O operations during backup, restore or direct-path reads and writes. The I/O buffer size is DB_FILE_DIRECT_IO_COUNT * DB_BLOCK_SIZE.

The range of the I/O buffer size is operating-system dependent and cannot exceed `max_io_size` for your platform. Increasing the DB_FILE_DIRECT_IO_COUNT parameter increases PGA or SGA memory use. Check your current setting for this parameter in the Oracle V$PARAMETER table.

DBWR_IO_SLAVES to 0 (Oracle8 and Oracle8i only)

Quick I/O supports kernel asynchronous I/O, eliminating the need for multiple db writers or db writer slaves. This parameter is set to zero by default.

USE_READV to FALSE (Oracle7 only)

The readv() system call allows multiple I/O requests to be put into a list that can be treated as a single I/O request, which reduces the CPU overhead for buffer copying. In the Oracle Performance Tuning Tips & Techniques document, Oracle recommends setting the use_readv parameter to TRUE to improve sequential scan performance when the UNIX File System is used. However, when raw I/O is used, using readv actually slows down the sequential scan performance. When Quick I/O is used, files are treated by Oracle databases as raw character devices and the use_readv parameter should be set to FALSE. This parameter is set to FALSE by default.

Summary

VERITAS Database Edition *for Oracle* gives administrators the flexibility to optimize Oracle performance and availability and the management tools to do so easily and nonintrusively. It enhances the native Solaris platform and whatever storage hardware you may be using to create a high-performance computing platform that protects availability of critical Oracle data. The product also serves as a foundation for additional Oracle-specific storage-management solutions, including Block Level Incremental Backups with VERITAS NetBackup™ *for Oracle*—Advanced BLI Agent and off-host operations using VERITAS Database Replication Option *for Oracle*.

VERITAS continues to work with Oracle as a long-standing strategic partner to develop new solutions and find new ways to enhance Oracle storage and to integrate VERITAS solutions with Oracle databases. Oracle Corporate itself uses VERITAS Database Edition *for Oracle* on its internal databases, which speaks to the strength of the solution and the commitment of the partners. For more information on VERITAS solutions *for Oracle*, see **http://www.veritas.com/oracle** .

VERITAS

V E R I T A S   W H I T E   P A P E R

VERITAS Software Corporation
Corporate Headquarters
1600 Plymouth Street
Mountain View, CA 94043
650-527-8000 or 800-327-2232

For additional information about
VERITAS, its products, or the location
of an office near you, please call our
corporate headquarters or visit our
Web site at www.veritas.com

VER03-DBEPERFWPR-0001 90-00924-399