Database configuration

Enterprise applications

NetApp
May 08, 2024

Table of Contents

Database configuration

Oracle database block sizes

ONTAP internally uses a variable block size, which means Oracle databases can be configured with any block size desired. However, filesystem block sizes can affect performance and in some cases a larger redo block size can improve performance.

Datafile block sizes

Some OSs offer a choice of file system block sizes. For file systems supporting Oracle datafiles, the block size should be 8KB when compression is used. When compression is not required, a block size of either 8KB or 4KB can be used.

If a datafile is placed on a file system with a 512-byte block, misaligned files are possible. The LUN and the file system might be properly aligned based on NetApp recommendations, but the file I/O would be misaligned. Such a misalignment would cause severe performance problems.

File systems supporting redo logs must use a block size that is a multiple of the redo block size. This generally requires that both the redo log file system and the redo log itself use a block size of 512 bytes.

Redo block sizes

At very high redo rates, it is possible that 4KB block sizes would perform better because high redo rates allow I/O to be performed in fewer and more efficient operations. If redo rates are greater than 50MBps, consider testing a 4KB block size.

A few customer problems have been identified with databases using redo logs with a 512-byte block size on a file system with a 4KB block size and many very small transactions. The overhead involved in applying multiple 512-byte changes to a single 4KB file system block led to performance problems that were resolved by changing the file system to use a block size of 512 bytes.

NetApp recommends that you do not change the redo block size unless advised by a relevant customer support or professional services organization or the change is based on official product documentation.

Oracle database parameters: db_file_multiblock_read_count

The `db_file_multiblock_read_count` parameter controls the maximum number of Oracle database blocks that Oracle reads as a single operation during sequential I/O.

This parameter does not, however, affect the number of blocks that Oracle reads during any and all read operations, nor does it affect random I/O. Only the block size of sequential I/O is affected.

Oracle recommends that the user leave this parameter unset. Doing so allows the database software to automatically set the optimum value. This generally means that this parameter is set to a value that yields an I/O size of 1MB. For example, a 1MB read of 8KB blocks would require 128 blocks to be read, and the default value for this parameter would therefore be 128.

Most database performance problems observed by NetApp at customer sites involve an incorrect setting for this parameter. There were valid reasons to change this value with Oracle versions 8 and 9. As a result, the parameter might be unknowingly present in `init.ora` files because the database was upgraded in place to Oracle 10 and later. A legacy setting of 8 or 16, compared to a default value of 128, significantly damages sequential I/O performance.

NetApp recommends setting the `db_file_multiblock_read_count` parameter should not be present in the `init.ora` file. NetApp has never encountered a situation in which changing this parameter improved performance, but there are many cases in which it caused clear damage to sequential I/O throughput.

Oracle database parameters: filesystemio_options

The Oracle initialization parameter `filesystemio_options` controls the use of asynchronous and direct I/O.

Contrary to common belief, asynchronous and direct I/O are not mutually exclusive. NetApp has observed that this parameter is frequently misconfigured in customer environments, and this misconfiguration is directly responsible for many performance problems.

Asynchronous I/O means that Oracle I/O operations can be parallelized. Before the availability of asynchronous I/O on various OSs, users configured numerous dbwriter processes and changed the server process configuration. With asynchronous I/O, the OS itself performs I/O on behalf of the database software in a highly efficient and parallel manner. This process does not place data at risk, and critical operations, such as Oracle redo logging, are still performed synchronously.

Direct I/O bypasses the OS buffer cache. I/O on a UNIX system ordinarily flows through the OS buffer cache. This is useful for applications that do not maintain an internal cache, but Oracle has its own buffer cache within the SGA. In almost all cases, it is better to enable direct I/O and allocate server RAM to the SGA rather than to rely on the OS buffer cache. The Oracle SGA uses the memory more efficiently. In addition, when I/O flows through the OS buffer, it is subject to additional processing, which increases latencies. The increased latencies are especially noticeable with heavy write I/O when low latency is a critical requirement.

The options for `filesystemio_options` are:

- **async.** Oracle submits I/O requests to the OS for processing. This process allows Oracle to perform other work rather than waiting for I/O completion and thus increases I/O parallelization.
- **directio.** Oracle performs I/O directly against physical files rather than routing I/O through the host OS cache.
- **none.** Oracle uses synchronous and buffered I/O. In this configuration, the choice between shared and dedicated server processes and the number of dbwriters are more important.
- **setall.** Oracle uses both asynchronous and direct I/O. In almost all cases, the use of `setall` is optimal.

The `filesystemio_options` parameter has no effect in DNFS and ASM environments. The use of DNFS or ASM automatically results in the use of both asynchronous and direct I/O.

Some customers have encountered asynchronous I/O problems in the past, especially with previous Red Hat Enterprise Linux 4 (RHEL4) releases. Some out-of-date advice on the internet still suggests avoiding asynchronous IO because of out-of-date information. Asynchronous I/O is stable on all current OSs. There is no reason to disable it, absent a known bug with the OS.

If a database has been using buffered I/O, a switch to direct I/O might also warrant a change in the SGA size. Disabling buffered I/O eliminates the performance benefit that the host OS cache provides for the database. Adding RAM back to the SGA repairs this problem. The net result should be an improvement in I/O performance.

Although it is almost always better to use RAM for the Oracle SGA than for OS buffer caching, it might be impossible to determine the best value. For example, it might be preferable to use buffered I/O with very small SGA sizes on a database server with many intermittently active Oracle instances. This arrangement allows the flexible use of the remaining free RAM on the OS by all running database instances. This is a highly unusual situation, but it has been observed at some customer sites.

NetApp recommends setting `filesystemio_options` to `setall`, but be aware that under some circumstances the loss of the host buffer cache might require an increase in the Oracle SGA.

Oracle Real Application Clusters (RAC) timeouts

Oracle RAC is a clusterware product with several types of internal heartbeat processes that monitor the health of the cluster.

The information in the misscount section includes critical information for Oracle RAC environments using networked storage, and in many cases the default Oracle RAC settings will need to be changed to ensure the RAC cluster survives network path changes and storage failover/switchover operations.

disktimeout

The primary storage-related RAC parameter is `disktimeout`. This parameter controls the threshold within which voting file I/O must complete. If the `disktimeout` parameter is exceeded, then the RAC node is evicted from the cluster. The default for this parameter is 200. This value should be sufficient for standard storage takeover and giveback procedures.

NetApp strongly recommends testing RAC configurations thoroughly before placing them into production because many factors affect a takeover or giveback. In addition to the time required for storage failover to complete, additional time is also required for Link Aggregation Control Protocol (LACP) changes to propagate. Also, SAN multipathing software must detect an I/O timeout and retry on an alternate path. If a database is extremely active, a large amount of I/O must be queued and retried before voting disk I/O is processed.

If an actual storage takeover or giveback cannot be performed, the effect can be simulated with cable pull tests on the database server.

NetApp recommends the following:

- Leaving the `disktimeout` parameter at the default value of 200.
- Always test a RAC configuration thoroughly.

misscount

The `misscount` parameter normally affects only the network heartbeat between RAC nodes. The default is 30 seconds. If the grid binaries are on a storage array or the OS boot drive is not local, this parameter might become important. This includes hosts with boot drives located on an FC SAN, NFS-booted OSs, and boot

drives located on virtualization datastores such as a VMDK file.

If access to a boot drive is interrupted by a storage takeover or giveback, it is possible that the grid binary location or the entire OS temporarily hangs. The time required for ONTAP to complete the storage operation and for the OS to change paths and resume I/O might exceed the `misscount` threshold. As a result, a node immediately evicts after connectivity to the boot LUN or grid binaries is restored. In most cases, the eviction and subsequent reboot occur with no logging messages to indicate the reason for the reboot. Not all configurations are affected, so test any SAN-booting, NFS-booting, or datastore-based host in a RAC environment so that RAC remains stable if communication to the boot drive is interrupted.

In the case of nonlocal boot drives or a nonlocal file system hosting `grid` binaries, the `misscount` will need to be changed to match `disktimeout`. If this parameter is changed, conduct further testing to also identify any effects on RAC behavior, such as node failover time.

NetApp recommends the following:

- Leave the `misscount` parameter at the default value of 30 unless one of the following conditions applies:
  - `grid` binaries are located on a network-attached drive, including NFS, iSCSI, FC, and datastore-based drives.
  - The OS is SAN booted.
- In such cases, evaluate the effect of network interruptions that affect access to OS or `GRID_HOME` file systems. In some cases, such interruptions cause the Oracle RAC daemons to stall, which can lead to a `misscount`-based timeout and eviction. The timeout defaults to 27 seconds, which is the value of `misscount` minus `reboottime`. In such cases, increase `misscount` to 200 to match `disktimeout`.

Copyright information

Trademark information