



Improving the Performance of Oracle9i Database Instance Recovery

By Arun Kumar R., Ph.D.

Oracle9i is one of the most stable of databases currently available. With a slew of novel features built into the 9i version, Oracle has improved its performance from previous versions and competing products. Even if you have not deployed new features and options such as Oracle Real Application Clusters (RAC), it would be advisable to review some of the recommendations by Oracle and tune your databases for faster instance recovery. In this article, we briefly cover the tips to prepare the Oracle9i database for faster instance recovery. Most of these tips are applicable to Oracle8i as well. This article includes timing results from a set of real tests performed in my lab environment.

What is Meant by “Instance Recovery”?

When an Oracle database crashes due to hardware or application failure, it has to go through a series of steps before it comes back online. During normal operation when an instance is shutdown cleanly using the SHUTDOWN IMMEDIATE statement the database changes held in memory are written to disk as part of the checkpoint performed during shutdown. When an Oracle instance crashes before it is shutdown properly, these changes are not yet written to disk and therefore are lost. When the instance is brought back online, the recovery operation is started automatically. In an RAC environment, when one or more instance(s) of the Cluster crashes, one of the surviving instances will start the recovery operation automatically.

We can define this process of “instance recovery” in Oracle as the automatic application of redo log records to Oracle data blocks after a crash or system failure. Instance recovery is also referred to as “instance and crash recovery” in Oracle documentation. Instance and crash recovery occur in two steps: cache recovery (rolling forward) then transaction recovery (rolling back). Since we are more interested in keeping as much data intact as possible in a database failure as rolling back, we will discuss cache recovery methods in detail here.

Cache Recovery and Transaction Recovery

In an RAC environment, the cache fusion services and structures enable the instances to locate global resources and database blocks that are currently in use by one or more instances. The information maintained by the Global Resource Directory allows real application clusters to minimize the time taken to start up a database after an instance failure. When a database undergoes cache recovery, Oracle applies all committed and uncommitted changes in the redo log files to its effected data blocks. The estimated time and effort needed for cache recovery processing is proportional to the rate of changes made to the database (update transactions per second) and the time elapsed between two checkpoints.

In transaction recovery, the changes that were not committed to the database at the time of the crash must be undone or rolled back. When an Oracle database undergoes transaction recovery, Oracle will apply the rollback segments to undo the uncommitted changes. The estimated time and work needed for such a cache recovery processing is proportional to the number and size of uncommitted transactions when the crash occurred. This is true for RAC as well as single instance databases.

How Does Checkpointing Affect Cache Recovery Processes and Performance?

You may be aware that Oracle issues a checkpoint when the database writer (DBWR) process writes all modified buffers in the SGA to the database data files. Checkpoints occur at every redo log switch and at intervals specified by the DBA. When a database failure occurs, only the redo records containing changes at system change numbers (SCN) higher than the checkpoint will be applied for recovery. The duration of the cache recovery processing is then determined by two factors: the number of data blocks that have SCNs greater than the SCN of the checkpoint, and the number of redo log blocks that need to be read to find those changes.

Let us now look at how the frequency of checkpointing affects the database performance. In Oracle9i, frequent checkpointing will write dirty buffers to the datafiles more often. So the application of the redo records in the redo log between the current checkpoint position and the end of the log involves the processing of relatively fewer data blocks. This reduces the cache recovery time in case of an instance failure.

On one hand, in an update intensive system such as a banking application, frequent checkpointing can reduce runtime performance due to more frequent writes. Checkpointing will force DBWn processes to perform writes frequently and slow the system down. On the other hand, when Oracle is forced to checkpoint often, it will reduce the duration of instance recovery by keeping the number of redo log records to be applied during recovery to a minimum. In a nutshell, if daily operational efficiency is more important than minimizing recovery time, the DBA should increase the interval of checkpoints. This should improve operational efficiency but will also increase instance recovery time.

To reduce the checkpoint frequency and optimize runtime performance of an Oracle9i database, the DBA has to do the following:

- Change the size of the online redo log files according to the amount of redo generated by the application. The logfiles should be large enough to avoid increased checkpoint activity. A rough guide is to allow log switches at most once every 20 minutes.
- Set the value of the LOG_CHECKPOINT_INTERVAL initialization parameter (in multiples of physical block size) to zero.
- Set the value of the LOG_CHECKPOINT_TIMEOUT initialization parameter to zero. This will eliminate any time-based checkpoints.
- Set the value of FAST_START_MTTR_TARGET (and FAST_START_IO_TARGET in 8i) to zero to disable fast-start checkpointing.

Impact of Cache Recovery on Performance

We can deploy the following methods to tune cache recovery in 9i and control the recovery duration (according to the nature of database application).

1. Tune the Redo Log Switching Frequency

We can influence the number of redo logs by setting LOG_CHECKPOINT_TIMEOUT and LOG_CHECKPOINT_INTERVAL parameters appropriately. Further, we can set the LOG_CHECKPOINT_TIMEOUT initialization parameter to an integer value 'n' to require the latest checkpoint position to follow the most recent redo block by no more than 'n' seconds. This will force 'n' seconds' worth of logging activity to occur between the most recent checkpoint position and the end of the redo log. As a result, the checkpoint position will keep pace with the most recent redo block in the database.

If the LOG_CHECKPOINT_TIMEOUT is set to be 100, then no buffers remain dirty in the cache for more than 100 seconds. The default value for LOG_CHECKPOINT_TIMEOUT is 1800 or 30 minutes. We can set the LOG_CHECKPOINT_INTERVAL parameter to an integer value 'n' to require that the checkpoint position never follow the most recent redo block by more than 'n' blocks. This will force 'n' redo blocks to exist between the most recent checkpoint position and the last block written to the redo log. As a result, the amount of redo blocks that can exist between the checkpoint and the end of the log is limited. Oracle9i (as well as 8i) limits the maximum value of LOG_CHECKPOINT_INTERVAL to 90% of the smallest log to ensure that the checkpoint advances into the current log before that log fills and attempts a log switch. LOG_CHECKPOINT_INTERVAL is specified in redo blocks (which in turn is the same size as operating system blocks). The DBA can query the LOG_FILE_SIZE_REDO_BLK column in V\$INSTANCE_RECOVERY and see the number of redo blocks corresponding to 90% of the size of the smallest redo log file. For example if the smallest logfile is 20 Megs (20971520) on a UNIX machine with 8,192 bytes block size, the LOG_FILE_SIZE_REDO_BLK will be limited to 2,304 (20971520 x 0.9 / 8192).

2. Using Fast-Start Checkpointing

One of the new and improved features in Oracle9i Enterprise Edition is the fast-start fault recovery functionality. Fast-Start Recovery functionality reduces the time required for cache recovery. It makes the recovery process predictable by limiting the number of dirty buffers and the number of redo records generated between the most recent redo record and the last checkpoint.

Fast-start recovery is based on the fast-start checkpointing architecture. Instead of bulk write operation from conventional log switching, fast-start checkpointing occurs incrementally. Each DBWn process periodically writes buffers to disk to advance the checkpoint position. The oldest modified blocks are written first to disk to ensure that every write lets the checkpoint advance. Fast-start checkpointing eliminates the need for bulk writes and resulting I/O spikes associated with it.

DBA can specify a target (bounded) time to complete the cache recovery phase of the instance recovery with the FAST_START_MTTR_TARGET initialization parameter. The FAST_START_MTTR_TARGET initialization parameter specifies the expected "mean time to recover" (MTTR) in seconds. MTTR is the expected amount of time for Oracle to perform crash/instance recovery for a single instance. Oracle automatically varies the incremental checkpoint writes to meet this target time.

The maximum value for FAST_START_MTTR_TARGET is 3,600 or one hour. If the value is set higher than 3,600, then Oracle9i will reduce it to 3,600. If the value of FAST_START_MTTR_TARGET is set too low, then the effective mean time to recover (MTTR) target will be the best MTTR target the system can achieve. If the value of FAST_START_MTTR_TARGET is set to a low value to reduce recovery time, then the effective MTTR target will be the estimated MTTR in the worst-case scenario (when the whole buffer cache is dirty). The DBA can query the TARGET_MTTR column in V\$INSTANCE_RECOVERY to find the effective MTTR value.

3. Use of Parallel Recovery

Parallel recovery is another method used to tune the cache recovery phase. The idea behind Parallel Recovery is very simple to follow. Parallel recovery uses a division of labor approach to allocate different processes to different data blocks during the cache recovery phase of recovery. For example, during recovery process, the redo log is read and the blocks that require redo application are parsed out. These blocks are evenly distributed evenly to all recovery processes and read into the buffer cache.

The RECOVERY_PARALLELISM (init parameter) is used to specify the number of concurrent recovery processes for instance or crash recovery. In order to use parallel processing, the value of RECOVERY_PARALLELISM should be greater than 1, but less than the value of the PARALLEL_MAX_SERVERS (init parameter). In a database instance, the recovery is usually dependent on reads made to data blocks. Hence, parallelism at the block level will only help recovery performance by speeding up the total I/Os involved. The only caveat is on systems with efficient asynchronous I/O. In these systems, parallel recovery will not significantly improve the recovery performance.

A performance test was conducted to determine the recovery time using one million records in a single pass query in a two node Oracle9i RAC Sun cluster environment with different parameter values. In a RAC lab environment, we tested an Oracle9i database with values of (300 and 0), (300 and 2), and (20 and 2) for FAST_START_MTTR_TARGET and RECOVERY_PARALLELISM. The following results were obtained, which clearly demonstrated the benefits of fine tuning a database for faster instance recovery.

continued on page 48

FAST_START_MTTR_TARGET	RECOVERY_PARALLELISM	Time to recover
300	0	9.01
300	2	3.55
20	2	2.77

Table 1

Summary

Improving the performance of database instance recovery in Oracle9i is not a tedious task if the DBA knows of the problem areas and how to address them. Readers can find more information on the initialization parameters and recovery mechanisms from the documentation listed at Oracle Technology Network website (<http://otn.oracle.com>).



About the Author

Arun Kumar, R., Ph.D., works in the Enterprise Architecture and Data Management Group of a major cellular carrier in Dallas, Texas. He has been working in the field of information technology over the last nine years primarily in (Oracle) databases, Data Warehousing, Supply Chain Management, and E-Commerce.