Tuning a Database Reorganization for Maximum Speed

By Heather Compher and Gil Asherie (Quest Software)

Introduction

Today, several methods are available for performing database reorganization and restructuring. Among these methods, reorganization inside the database, using native SQL and PL/SQL scripts, has proven extremely fast and reliable. Of course, the speed of reorganization will vary by hardware platform and database configuration.

Quest Software's reorganization solutions, Space Manager and LiveReorg, provide a number of options and parameters for achieving optimal reorganization speed in any given environment. This paper reviews these settings and discusses how to take advantage of them.

The areas covered include:

- The UNRECOVERABLE Option
- The Parallel Query Option (PQO)
- ALTER SESSION Parameters
- Other Considerations

In conclusion, we offer a benchmark of a large-scale reorganization of an Oracle Applications database performed with Space Manager.

The UNRECOVERABLE Option

The UNRECOVERABLE option allows Oracle to build tables and indexes without writing to redo logs or rollback segments. According to leading Oracle experts, using UNRECOVERABLE can increase the performance of table and index builds by 30 to 50 percent.[1]

It is recommended that after reorganizing an object with the UNRECOVERABLE option, a hot backup be taken of the affected tablespace, to maintain compliance with standard backup and recovery policies.

The Parallel Query Option (PQO)

The Parallel Query Option (PQO) is included at no charge with the Oracle Server. It is not related to or dependent upon Oracle Parallel Server (OPS). PQO can be used to build tables and indexes in parallel.

The Parallel Query Option (PQO) is strongly recommended to increase the speed of reorganizations, provided adequate resources are available. If the system's CPUs and disk controllers are already heavily loaded, parallel execution will usually have an adverse effect. Due to the complexity and resources involved, it is advisable to avoid using PARALLEL on small objects. Consider using the UNRECOVERABLE option with PQO, as serialized writes to the redo logs will decrease the impact of parallelization, especially if the degree of parallelism is greater than four.

Use care in selecting the degree of parallelism. For tables, if you choose a degree of parallelism of N, Oracle will try to use N + 1 processes. The extra process is a controller process known as the query coordinator. For indexes, there will be two processes for each degree of parallelism, one for scanning the table and one for sorting the rows. Thus, if you choose a degree of index

---

[1] Aronoff, Eyal; Loney, Kevin; and Sonawalla, Noorali. *Oracle8 Advanced Tuning and Administration.* Oracle Press, 1998. p. 482.

parallelism of N, Oracle will try to use 2N + 1 processes to build the index. Each one of the N sorting processes may use up to SORT_AREA_SIZE of memory.

As expected, the optimal degree of parallelism will be highly dependent on system configuration. Consider the following:

- Generally, the degree of parallelism should not be set to more than twice the number of CPUs. Doing so may cause high wait times, as the parallel query slaves compete for CPU time.[2] If very powerful CPUs are used, it may be advantageous to increase the degree of parallelism beyond this level in some cases.[3] In the case of the highly CPU intensive index creations, don't exceed the number of CPUs for the degree of parallelization.

- The degree of parallelism should not be far greater than the degree of striping. Otherwise, the disks may become saturated.[4]

- When parallelizing index builds, consider the amount of memory available for the sorting processes.

To help you choose the most effective degree of parallelism, use sar –u and sar –d or the TOP utility to monitor processor and disk I/O utilization.

Tuning the Query Server Pool

The common set of parallel query servers available in an instance is known as the Query Server Pool. To manage the Query Server Pool, tune the following init.ora parameters:

- PARALLEL_MIN_SERVERS – This parameter sets the number of query server processes that are started when the instance starts. This eliminates the performance penalties of frequent query server process startups and shutdowns.

- PARALLEL_MAX_SERVERS – This parameter sets the maximum size of the query server pool. The recommended value is < 2 * max_degree * number_of_concurrent_users >.

The number of parallel processes available will be bound between PARALLEL_MIN_SERVERS and PARALLEL_MAX_SERVERS.

To determine if there is any contention for parallel query slaves, you can query V$PQ_SYSSTAT as follows:

```
select Statistic, Value from V$PQ_SYSSTAT
order by Statistic;
```

If the value for the statistic "Servers Busy" is high, increase PARALLEL_MAX_SERVERS.

Instance Monitor can be used to monitor the parallel query servers.

Extent Sizes and PQO

The number of parallel processes used will also effect extent allocation. When creating tables or indexes in parallel, each of the parallel query processes will allocate one of more extents, as determined by the MINEXTENTS setting for the object. Each of these MINEXTENTS will be of size INITIAL. For example, when creating a table with PARALLEL degree 4, MINEXTENTS 2

[2] Alomari. Ahmed. *Oracle and Unix Performance Tuning.* Prentice Hall, 1997. p. 212.
[3] Oracle Corporation, Doc Id 2071112.6, "How to Tune the Performance of the Parallel Query Option," p. 3.
[4] Alomari. Op. cit. p. 482.

and INITIAL 20MB, each of the four parallel processes will allocate an extent of 40MB, for a total 160MB. Although the extents are coalesced at the end of the operation, over-allocation is a concern.

To take advantage of PQO while minimizing over-allocation, you may want to use smaller extent sizes. The presence of multiple of extents should not pose a problem. As explained in Oracle Paper #711, "the performance for DML operations is largely independent of the number of extents in the segment."[5] In fact, Paper #711 outlines a strategy for using multiple equally-sized extents to eliminate freespace fragmentation at the tablespace level.

It is also possible to create "pockets'" of free space if you specify more query servers than there are datafiles in the tablespace. Oracle can only coalesce the free space in the last extent of a table or index in each datafile. The pockets caused in extents other than the last extent will only be available for subsequent inserts into the affected object.

Lastly, remember to check for adequate contiguous freespace for both tables and indexes. One of the most common issues that sites face using PQO for the first time is running out of space because they under-estimated the requirement.

ALTER SESSION Parameters

A variety of session parameters can be tuned to improve the performance of a reorganization. These settings include a variety of sorting parameters and the DB_MULTIBLOCK_READ_COUNT parameter. For Oracle 8, Space Manager executes an ALTER SESSION command to dynamically set these parameters for the reorganizing session, without affecting other users. The settings in the INIT.ORA file are not affected.

Sorting Parameters

Oracle provides several parameters that can be set to speed up sort operations. Use of these parameters can significantly improve the performance of index reorganization. The parameters to consider are:

- SORT_AREA_SIZE – This parameter controls the maximum amount, in bytes, of memory for each sort. If the sort area is too small, an excessive amount of I/O will result. If sort area is too large, the OS paging rate will be too high.[6] It is especially important to consider this when using a high degree of parallelism to build indexes, since that total amount of sort area used will be [((degree of parallelism *2) + 1) * SORT_AREA_SIZE]. 1-2 M is a good rule of thumb when using PQO. If not using parallelism, consider setting SORT_AREA_SIZE to the size of the largest index being reorganized.

- SORT_AREA_RETAINED_SIZE – This parameter controls the maximum amount, in bytes, of user global area memory retained after a sort run completes. Set to the same as SORT_AREA_SIZE if using parallel query. Otherwise, depending on available memory, set to the same or one half of SORT_AREA_SIZE. Note that excess memory is not returned to the operating system.

- SORT_DIRECT_WRITES – Setting SORT_DIRECT_WRITES = true allows Oracle to bypass the buffer cache for the writing of sort runs to the temporary tablespace. This can improve the performance improvement can be a factor of three or more. Be sure to also set SORT_WRITE_BUFFERS = 8 and SORT_WRITE_BUFFER_SIZE=65536.[7]

---

[5] Himatsingka, Bhaskar and Loaiza, Juan. Paper #711, "How to Stop Defragmenting and Start Living: The Definitive Word on Fragmentation." Oracle Corporation.
[6] Oracle Corporation, Doc ID 2071112.6, "How to Tune the Performance of the Parallel Query Option," p. 1.
[7] Aronoff et al. Op. cit.

SORT_DIRECT_WRITES, SORT_WRITE_BUFFERS and SORT_WRITE_BUFFER_SIZE are obsoleted in 8.1.3. The same considerations for SORT_AREA_SIZE apply to SORT_DIRECT_WRITES when using the parallel query option. Under Oracle 8I, sorts always use direct writes and automatically configure the number and size of the direct write buffers.[8]

DB_FILE_MULTIBLOCK_READ_COUNT

DB_FILE_MULTIBLOCK_READ_COUNT controls the number of data blocks read for each read request during a full table scan. If you are using LVM or striping, this parameter should be set so that <DB_BLOCK_SIZE * DB_FILE_MULTIBLOCK_READ_COUNT> is a multiple of the LVM stripe size. If you are not using LVM or striping, <DB_BLOCK_SIZE * DB_FILE_MULTIBLOCK_READ_COUNT> should equal the maximum operating system read buffer. On many UNIX systems this is 64 KB. In any case, DB_FILE_MULTIBLOCK_READ_COUNT cannot be larger than <DB_BLOCK_BUFFERS / 4>.

The maximum read buffer is generally higher on raw file systems. It varies from 64K (on AIX) to 128K (on Solaris) to 1MB (HP-UX). On a UNIX file system, it is usually only possible to read one buffer per I/O, usually 8K. On 32-bit Windows, the buffer is 256K.

This parameter will significantly increase the performance of a reorganization if properly tuned. For example, suppose the OS read buffer is 64 KB, the database blocksize is 4 KB and DB_FILE_MULTIBLOCK_READ_COUNT is set to 8. During a full table scan, each I/O operation will read only 32 KB. If DB_FILE_MULTIBLOCK_READ_COUNT is now set to 16, performance will almost double because twice as much data can in each I/O operation.

Other Considerations

When executing especially large reorganizations, a few additional init.ora parameters should considered. These parameters cannot be altered dynamically for the session. As explained at the end of this section, it is also important to ensure that there is a tablespace available for sorting whose contents is TEMPORARY instead of PERMANENT.

Asynchronous I/O and Multiple DBWRs

It is recommended to use either asynchronous I/O or multiple database writers to minimize the blocking of the DBWR I/O. Oracle recommends the use of asynchronous I/O as “more effective and efficient” than multiple DBWRs because “asynchronous I/O will perform parallel I/O across the disks, where multiple DBWRs only attempts to emulate this process.”[9] One exception is Sun files systems where we have seen better performance when async I/O is turned off and multiple DBWRs are used instead. I/O Slaves are new with Oracle 8 and replace multiple DBWRs. They are specialized processes whose only function is to perform I/O, and can operate whether or not asynchronous I/O is available.

If asynchronous I/O is used, use only one DBWR process. On many platforms, if Oracle finds asynchronous I/O enabled, it will only start one DBWR. Asynchronous I/O is generally possible only on raw character devices, with the exception of Sun and AIX, where it is available for both file systems and raw devices.

When asynchronous I/O is either not available or not desired, use multiple DBWR processes or (in Oracle8) a single DBWR process with multiple I/O slaves depending upon the platform. If using multiple DBWRs, try using 1-2 DBWR per disk or one per CPU. In Oracle7 set the DB_WRITERS parameters. In Oracle8, DB_WRITER_PROCESSES replaces the parameter

[8] Oracle Corporation, Doc ID 102896.1, “Obsolete and New Sort Parameters.”
[9] Oracle Corporation, Doc ID 1059403.6, “Cannot Start Multiple Database Writers, No Errors Reported.”

DB_WRITERS and specifies the initial number of database writer processes for an instance. If you use DBWR_IO_SLAVES, only one database writer process will be used, regardless of the setting for DB_WRITER_PROCESSES.

The Oracle8 Tuning Guide for Sun recommends "DBWR_IO_SLAVES should only be set to greater than 0 if ASYNC I/O (that is, DISK_ASYNCH_IO, or TAPE_ASYNCH_IO) has been disabled, otherwise DBWR will become a bottleneck. In this case the optimal value on Solaris 2.x for DBWR_IO_SLAVES should be 4. In the case of LGWR_IO_SLAVES, it is not recommended to deploy more than 9 slaves."[10]

The Importance of a TEMPORARY Tablespace

The tablespace of type TEMPORARY is critical for supporting index sorts that cannot be handled in memory. The index builds will be very slow if adequate TEMPORARY tablespace is not available. To reduce temporary segment lock allocation contention and save the costs involved in temporary segment allocation (equivalent to approximately ten insert/update/delete statements), make sure that you have designated a tablespace with TEMPORARY contents.[11] Check the 'contents' field in DBA_TABLESPACES to view the status of your tablespaces.

In Oracle Applications, the temporary tablespace contains permanent tables and has the status PERMANENT. In order to change the status to TEMPORARY, you must move the permanent tables to another tablespace. Another option is to create another tablespace of type TEMPORARY to be used for the reorganization.

In SAP environment due to experiences in the past, SAP recommends to generally set the status of the temporary table space to PERMANENT. You can check with SAP for the latest recommendation. Generally in SAP environment tablespaces are created with content permanent and the parameters for initial/next are set to the same value and pctincrease to 0. The tablespace's pctincrease set to 0 (zero) causes smon not coalescing the free space and although the temporary table is dropped, new users don't have to waste time breaking up a large extent into a smaller one they need. They will find an exact sized extent in the tablespace.

Example of a Large Reorganization Performed by Space Manager

This example shows how Space Manager was used to fully reorganize and restructure a large Oracle Applications database. Through successful tuning, a very high reorganization speed was achieved.

Environment

The environment is which the reorganization was performed was as follows:

- Application: Oracle Applications
- Server: HP-model running HP-UX 10.20, 12 CPUs, 3.4 GB Memory
- Disk Subsystem: EMC model 3930 (4 channels, 5 GB cache) with raw devices and asynchronous I/O
- Database: Oracle 7.3.4
- Data volume (tables + indexes) to be reorganized: 208 Gig

Objective

The objective of the full database reorganization was to relocate all objects from an existing EMC disk array to a newer EMC array. In addition to providing much-needed data defragmentation, this would allow the site to take advantage of optimally sized datafiles and more powerful disk

[10] Oracle Corporation. *Oracle8 Administrator's Reference Release 8.0.5 for Sun SPARC Solaris 2.x*. Chapter 3.
[11] Aronoff et al. Op. cit. p. 486.

technology. This complete Oracle Applications implementation included about 3000 tables and over 6000 indexes, across 62 different tablespaces.

Methods and Parameters

Space Manager was used to create the new optimal tablespaces and datafiles. Space Manager then generated the SQL and PL/SQL scripts necessary to reorganize each data tablespace and its associated index tablespace.

The tuning options used were:

- Parallel query with a degree of 4 was used for large tables and indexes.
- UNRECOVERABLE was used for all objects.
- The temp tablespace was converted to TEMPORARY contents.
- SORT_AREA_SIZE and SORT_AREA_RETAINED_SIZE were increased from their production values to 60MB and 30MB respectively. SORT_DIRECT_WRITES was enabled.
- DB_FILE_SIMULTANEOUS_WRITES was increased from the production value to 64. The number of database block buffers was also increased for the reorganization.

Results

As a result of these tuning efforts, the total time for the reorganization was improved from an initial test run of 69 hours to a final production run of 16 hours. This equated to a reorganization rate of 13 GB/hour.

Conclusions

As illustrated in the benchmark above, reorganizations inside the database using SQL and PL/SQL scripts can be extremely fast. Taking advantage of a few simple Oracle parameters will help you achieve the optimal reorganization speed for your hardware environment.