Oracle Rdb
Performance Management Guide

Solving the Five Most Common Problems with Rdb Application Performance and Availability

White Paper

ALI Database Consultants
803-648-5931
www.aliconsultants.com
February 2, 1999

Table of Contents

I. Introduction

"Technology failures have immediate top-line (revenue) impact."

Paul Dravis

NationsBanc
Montgomery
Securities

Oracle Rdb has long been an environment of choice for mission-critical applications that require full-time operation. It clearly outperforms many of today's client/server databases. In fact, Oracle Rdb has won *Intelligent Enterprise's* Reader's Choice Award. However, Oracle Rdb users still experience periodic performance problems that impact business productivity. These problems also impact the productivity of the IT department by requiring lengthy and complex tuning sessions. This document will provide a number of ways to help your IT organization better manage the performance of your Oracle Rdb environment and minimize the impact that these problems have on the supporting business units.

Common Oracle Rdb Performance Issues

Ensuring Oracle Rdb performance can be a difficult challenge, even to the finest of DBAs. Fragmentation, improper indexing, locking, and I/O balancing problems often cause user response time and availability problems. It not only can be cumbersome to identify the problems, but can also require a significant amount of time to resolve (e.g., generating database tuning scripts).

Solutions to the most common Oracle Rdb problems are not easy to implement; however, there are certain actions that can be taken to ensure these problems are minimized and performance is optimized with minimal time. These actions enable your Oracle Rdb applications to meet the demanding requirements of today's business users.

Criteria for Choosing an Effective Solution

The right solution to solve many of these problems with your Oracle Rdb environment requires a specific set of criteria. First, the solution must be easy to implement. A solution that requires hours of time to learn can often introduce many unexpected costs and eventually become shelf-ware. Second, a solution must be able to continuously monitor your Oracle Rdb environment and identify problems for you before your users experience them. This will enable your 1T team to focus on other areas of importance and rely on technology to proactively locate problems. Third, a solution should provide the ability to produce a tuning process that is quick, predictable, and reliable. Otherwise, your IT team may spend too much time generating tuning scripts, or may produce scripts that require significant changes. Finally, a solution should address the most common problems first to ensure your team can resolve them quickly and effectively.

These items, as well as more detailed requirements, will be discussed throughout the following analysis of each problem. There are multiple ways that can enable your IT team to better manage the Oracle Rdb environment and automate many of the tedious functions in database management.

II. Oracle Rdb Fragmentation

Oracle Rdb fragmentation can originate from misallocation of storage parameters for tables and indexes, unplanned growth, lack of partitioning, and poor distribution of the database across available disks.

The Problem

Oracle Rdb internal fragmentation of storage area files and records is one of the leading causes of poor application performance, and the impact can be detrimental to all users. This problem increases the demand on system resources and can force all users to implement an inefficient method of accessing data from the database. Subsequently, this problem typically results in response time issues for many users.

Oracle Rdb fragmentation increases the time to locate data and can originate from many different sources, including misallocation of storage parameters for tables and indexes, unplanned growth, lack of partitioning, and poor distribution of the database across available disks.

Storing multiple tables/indexes with varying record sizes in the same storage area contributes to fragmentation problems. When multiple items with varying sizes are stored together, data is stored unevenly across the storage area pages. Additionally, as new data is added, dispersion of data for the same table across pages in different locations of the storage area will occur. These situations force the addition of inefficient I/O to retrieve the rows of the tables, along with the potential for I/O contention, as different items with different uses are being accessed simultaneously in the same area. Although additional storage areas can help reduce these problems, they cannot prevent fragmentation from occurring if they are inadequately tuned. A properly tuned database will minimize the search time it takes to locate data within Oracle Rdb's storage areas.

Besides internal record fragmentation due to inadequate tuning of storage parameters, file extends within the overall allocation of storage areas can also affect fragmentation. Oracle Rdb will dynamically extend storage area files as needed when no room is available in the existing storage area. This leads to file fragmentation, which also contributes to performance problems. Storage areas should be sized for current records plus adequate growth to minimize the possibility of file fragmentation. Improper sizing of the storage areas can cause availability problems if a disk fills up because of extended files and causes a database shutdown until additional space is made available.

The Solution

Multiple Storage Areas

Storage area fragmentation can be greatly reduced by creating multiple storage areas. Creating storage areas for larger tables/indexes or related tables provides a more manageable environment. This allows storage parameters for tables and indexes contained in the areas to be tuned specifically for the items stored. The page size and allocation can be tuned only at the storage area level. With correct page sizing in correspondence with the database buffer size, retrievals and inserts to the storage area will be more efficient as Oracle Rdb does not waste time splitting records across pages and searching for space to store the records. Additionally, optimal I/O performance can be achieved with this highly partitioned approach to storage areas because more active storage areas can be separated across multiple disks/controllers to reduce I/O contention.

Oracle Rdb stores internal data about the database structure in ORACLE RDB$SYSTEM. All requests to the database require some access to this internal information. It is recommended that all non-system data be moved out of ORACLE RDB$SYSTEM to other storage areas to reduce the contention for this system area which is used by almost all transactions.

Storage Area Sizing

A truly proactive solution will identify the growth levels of the database and size the storage areas appropriately.

The use of storage areas can reduce the logical fragmentation of data. Storage areas can still be extended, however, causing storage area file fragmentation. Estimating/anticipating storage area growth beforehand and allocating sufficient pages can address this storage area file fragmentation.

Another factor in properly sizing storage areas is the page size. Oracle Rdb pages contain overhead to track data on each page. Page size can be incremented in blocks of 512 bytes. The page size of the stored table along with the Oracle Rdb overhead determines how many occurrences (records) can fit on a page. The number of occurrences per page divided into the total number of table records indicates the number of pages that are required for the given allocation. Thus, appropriate storage area sizing can further reduce Oracle Rdb fragmentation and increase performance.

The use of multiple storage areas alone can dramatically increase I/O performance. However, further tuning options are also available. A few of these are the use of PLACEMENT VIA INDEX, taking advantage of HASHED and SORTED indexes, and CLUSTERING and SHADOWING Oracle Rdb components within storage areas.

Choosing the Right Solution

An effective solution to the fragmentation problem will enable you to achieve tuning that will not only defragment the database, but also partition the database via storage areas, distribute storage areas across available disk drives, and size the storage areas to handle growth within the database. This will enable your team to proactively tune the database and minimize the amount of time spent on future fragmentation problems.

A truly proactive solution will identify the growth levels of the database and size the storage areas appropriately. Otherwise, fragmentation problems will occur, frequently requiring periodic reorganization of storage areas. A proactive solution will also be flexible enough to resolve fragmentation problems in the entire database, or simply a few storage areas. This will enable you to tune your entire database, or simply the most important areas, depending on the maintenance window you have available.

Finally, an effective solution should perform error checking to ensure all dependencies, steps, and procedures are followed.

III. Poorly Indexed Tables

Performance problems will arise when many indexes are stored in a single storage area, or the indexes used for table access are stored inefficiently.

The Problem

Choosing the right indexes for Oracle Rdb record retrieval can be quite a challenge, since there are so many different types. Each index type, whether SORTED, HASHED (ordered/scattered), or ranked, has a particular application for the specific data access needs of your Oracle Rdb application. The wrong indexes can cause serious performance problems that impact user responses times. Additionally, not using an index can cause performance problems as well due to extensive long table scans and locking issues.

In most cases, an index improves data access as long as it is appropriate to typical data access patterns and stored/sized in the correct storage areas. However, performance problems will arise when many indexes are stored in a single storage area, or the indexes used for table access are stored inefficiently. Placing the indexes in the right locations requires significant analysis of the database structure.

The Solution

The first step in optimizing the use of indexes is to determine which type should be used for each table. The most common types of indexes and their uses are explained below.

SORTED

SORTED keys provide an efficient means for range retrievals. For example, selecting records from September 15, 2000, through October 21, 2000, could be facilitated by using a SORTED index. The SORTED index allows for a greater-than or equal-to search to quickly locate the first value, then proceeds sequentially through the records based on their indexed values. This indexing strategy uses a B-tree structure to enable range retrieval. Because of this structure, however, one characteristic of SORTED keys is the necessity to lock index nodes during index updates. This locking reduces concurrency as records seemingly unrelated to the selected record are also locked, making them unavailable for access by other users.

HASHED

HASHED keys provide an efficient means for exact match retrieval. For example, retrievals often based on invoice numbers and employee identifications are based on a match with an exact number, identification, or other similar value. HASHED keys use the index value itself to determine a relative storage location. A HASHED key value may be located with a single I/O. Additionally, only the page containing the value needs to be locked. Thus HASHED keys can be used for certain exact match retrievals and avoid the concurrency reduction of SORTED indexes.

Placement Via Option

In addition to the ability to select an index strategy to match retrieval needs, Oracle Rdb allows further optimization with the PLACEMENT VIA option. Each table may be placed via an index.

Oracle Rdb attempts to place records for such tables in order according to the index sequencing, thus decreasing the I/O necessary to retrieve desired records. This occurs because the I/O associated with finding the index and the record will most likely retrieve the related data records at the same time.

PLACEMENT VIA can be used with either a SORTED or HASHED index. Used with a SORTED index, range retrievals can be dramatically improved. Multiple records can be buffered with a single I/O. If the desired records are physically placed together, then subsequent record retrievals can occur within memory rather than requiring additional disk I/Os.

The use of PLACEMENT VIA with a HASHED key functions similarly but is often employed for different reasons. Using a HASHED key indicates that the user is looking for exact matches. These situations often occur with interactive inquiries that are characteristic of transaction-based systems. By using such PLACEMENT VIA, a user may achieve retrieval with a single I/O without locking other records.

Implementation of PLACEMENT VIA is not for every situation and should only be considered with knowledge of how the application typically uses the data.

Choosing the Right Solution

You should receive indexing strategy recommendations based on actual Oracle Rdb activity and appropriate rules for the various types of index use.

An effective solution will enable you to make indexing changes during the tuning process. This feature enables you to ensure tuning results are predictable and are tailored to your specific Oracle Rdb environment.

Additionally, you should receive indexing strategy recommendations based on actual Oracle Rdb activity and appropriate rules for the various types of index use. This lets you make better decisions on implementation and optimization, depending on the use of the tables, by monitoring the database over time to determine patterns of user activity. This type of solution would monitor activity over time to determine how the users are accessing the database, and which indexes are used most, etc.

Finally, a solution that enables you to implement the PLACEMENT VIA option will ensure you maximize application performance with the limited I/O resources available.

IV. User Locking

Locked users may leave a runaway process, and may begin to quickly consume massive amounts of system resources.

The Problem

For many Oracle Rdb applications that require instant access to data from Oracle Rdb, concurrency is crucial. When locking issues occur, not only must users wait, but they can also incur additional strains on system resources. These can cause I/O bottlenecks and high CPU consumption, impacting all other users accessing the server.

Additionally, locked users may simply log off, then log back on in order to continue. This may leave a runaway process, and may begin to quickly consume massive amounts of system resources. The IT organization must locate this problem, identify the user, and resolve the lock situation immediately or all users will be halted by sluggish system performance. Regardless, the downtime the business unit may experience can become extremely costly.

The Solution

An optimally tuned database that is retrieving/updating rows efficiently can greatly reduce the potential of locking problems. Additionally, advanced techniques like Clustering and Shadowing are available to extend a low locking, low I/O approach to typical transaction processing to achieve very high performance.

Clustering and Shadowing

Clustering and Shadowing are techniques that can consistently provide 1/0 performance improvements for retrieving the parent record (one) and all its children (many).

Clustering and Shadowing may apply to retrievals of one-to-many relationships. Examples are quite common: invoice and invoice-items, order and order-items, and employees and degrees entries. Typically, these types of retrievals require instant access to answer queries from clients, users, or others. Quick access requirements indicate that concurrency is crucial--locks from other users that might prevent the needed access could prove costly or even disastrous. Additionally, these systems may have numerous users whose combined retrievals may incur significant 1/O. HASHING the identifier for the one-to-many relationship can reduce or even eliminate lock contention. Placing the records via the indexes of each table will group the records so that they can be retrieved together, reducing disk 1/O compared with a similar approach with SORTED indexes. The use of HASHING to implement Clustering and Shadowing enables the concurrency required for multi-user appends and updates in a timely fashion. It is often the only means of satisfying the requirements of certain time-critical, transaction-based systems.

Clustering and Shadowing are techniques that can consistently provide I/O performance improvements for retrieving the parent record (one) and all its children (many). To successfully implement this strategy, many of the features described thus far are employed. The HASHED key for the parent and child tables are placed into the storage area of the 'parent' (one) table. Both indexes must be comprised of exactly the same columns in the same order. Otherwise, they would not HASH together. The parent table is PLACED VIA the HASH index. Thus, with one I/O the Parents' HASH key, its table record, and the child's HASH key values are all located. The second I/O can then retrieve all of the child's table records as they are stored in a second storage area, PLACED VIA the HASH value and thus "shadowing" the parent.

Successful implementation requires a proper page size for each area and sufficient allocation to contain the desired number of records. Remember that the Oracle Rdb HASH algorithm is static. The HASHED placement is based on the initial space allocated to the storage area. Once the storage area is full the HASH algorithm will cause collisions and performance will degrade. Thus, it is beneficial to anticipate and plan for file growth in advance.

Choosing the Right Solution

An effective solution to help minimize locking will have several characteristics: the ability to identify the occurrence of locking problems; the ability to identify hashing opportunities; the ability to calculate appropriate pages sizes and calculations; and the ability to implement clustering and shadowing techniques during the tuning process.

When choosing a tuning solution, ensure that the product enables you to proactively calculate space within the pages and to take advantage of the clustering and shadowing functions. These combined capabilities allow you to not only locate and resolve user-locking problems, but also to proactively minimize them in the future.

V. Inadequate Disk Space for Extends

The Problem

Too often, a storage area will attempt to extend itself to handle user updates, and the disk the storage area is located on simply will not allow it due to inadequate space. This will prevent the user from making updates to the storage area and may cause downtime while the situation is resolved. The problem originates from not adequately sizing the storage areas to handle the growth, as well as from poorly distributing the database components across the available disks and storage space.

The Solution

I/O Distribution and Storage Area Sizing

When sizing the storage areas, it is a safe practice to know which are busy and which are relatively static, and provide enough room to allow 3-6 months growth.

Appropriately distributing the storage areas across the available disk drives and sizing them appropriately will minimize space problems. When performing I/O distribution it is important to remember a few rules of thumb:

- Do not place all your indexes on a single disk drive when not using large RAID units with much built-in caching
- Separate the table and index if you are not using the PLACEMENT VIA option
- Keep RDA and SNP files on separate disks
- Keep your busiest storage areas on separate disks

Storage Area Sizing

Appropriately sizing storage areas to handle growth will minimize the occurrence of extends and the possibility of running out of disk space. When the storage area is initially created on the disk drive, allocate enough space, depending on the table usage, to handle current records and expected growth until the next tuning window. This will minimize the fire-fighting necessary to resolve a "space" problem, and will minimize the cost to the business unit that is affected.

When sizing the storage areas, it is a safe practice to know which are busy and which are relatively static, and provide enough room to allow 3-6 months growth. This will enable you to determine if adequate space is available before the occurrence, so you can purchase the necessary hardware proactively, rather than in the middle of a crisis. Additionally, it will help minimize the maintenance required to manage the storage areas.

Choosing the Right Solution

A product should allow you to generate a tuning script to move a particular problem storage area to another disk.

An effective solution to this problem will assist you in distributing your I/O across your available disk drives, and enables you to allocate space within the storage areas to handle the growth of your particular Oracle Rdb environment.

In assisting you with I/0 distribution, a product should allow you to generate a tuning script to move a particular problem storage area to another disk. Additionally, you can receive extra value from a product that will monitor your database beforehand, and will provide recommendations for distributing the database appropriately across your disk drives, following the rules of thumb and observed activity levels, and helping you save time during your maintenance periods.

Finally, a product should help you size the storage areas to handle their growth for extended periods. This includes resizing them correctly for current records plus some amount of growth to handle extends and fragmentation. This will eliminate unexpected fire-fighting to fix problems during production hours.

VI. Disk I/0 Queue Length

Typically, if the queue length averages two or more transactions on several database drives, you may need to rebalance the 1/0 to improve performance.

The Problem

Queue lengths (number of user transactions waiting in a queue to be processed) occurring on thc disk drives are often causes of performance problems, since user requests are building up faster than the device can service them. This can often lead to locking situations and can sometimes completely halt user activity. If this occurs on several database drives continuously for long periods, users will experience response time and performance degradation.

The Solution

In order to resolve this problem, an entire redistribution of I/0 may be necessary. Typically, if the queue length averages two or more transactions on several database drives, you may need to rebalance the l/O to improve performance. The following I/0 distribution rules should be adhered to:

· Do not place all your indexes on a single disk drive when not using large RAID units with much built-in caching

· Separate the table and index if you are not using the PLACEMENT VIA option

· Keep RDA and SNP files on separate disks

· Keep your busiest storage areas on separate disks

Choosing the Right Solution

An effective solution for this problem will enable you to follow the rules of thumb in I/O distribution (see previous section), size pages appropriately, and distribute tables across the available disk drives. I/O distribution is one of the most simplistic, yet often overlooked areas to improve performance. With the right tool, it can be very simple and require a minimal amount of time to implement.

VII. ALI Controller for Oracle Rdb

"Need database design statistics? Anyone tuning Oracle Rdb databases will like this...A very easy-to-use tool for tuning an Oracle Rdb database."

Dr. Lillian Hobbs
Oracle Corporation

Tuning efforts can be reduced drastically, and performance improvements typically range from 20 to 50 percent.

ALI's Rdb ControllerTM provides a solution for many Oracle Rdb environments by tuning the entire database and monitoring the system to identify problem areas before they impact the user community. ALI Controller not only properly tunes the storage areas, indexes, table distribution, and 1/O distribution, it performs these actions in record time. Tuning efforts can be reduced drastically, and performance improvements typically range from 20 to 50 percent.

Additionally, to ensure your Oracle Rdb environment is constantly performing as expected, Rdb Controller provides you with the ability to continuously monitor Oracle Rdb, and identify problem areas before they impact your user community.

Minimizing Fragmentation

Rdb Controller performs database tuning that covers the items above. Using the DBTune feature, users can automatically implement a solution to resolve fragmentation problems. DBTune partitions the database via storage areas, intelligently distributes storage areas across multiple disk drives, and appropriately sizes storage areas to handle growth within the database. This enables you to proactively tune the database, so your database will perform optimally for months.

Rdb Controller first analyzes your database activity to determine the busiest storage areas and disk drives, and identifies the growth level for the storage areas. It also analyzes where to distribute tables and indexes, so that user access times are minimized. Then, Rdb Controller generates all the necessary tuning scripts to tune the entire database, or a limited number of specific storage items. These scripts can take days to generate manually, where ALI Controller can complete them in minutes and ensure all dependencies and steps are in place with error checking for the tuning process.

Rdb Controller provides a complete tuning process implemented with a DCL driver that runs the generated SQL scripts to perform the tuning. These scripts can be executed to tune your entire database, or simply the most important areas, depending on the maintenance window you have available. ALI's customers typically experience a 20 to 50 percent improvement in application performance.

Choose the Right Indexes

Rdb Controller reviews indexes and database activity to identify conversion considerations. It provides recommendations to change the index type to either SORTED or HASHED based on likely access approaches. The decision should be made based on the actual retrieval and concurrency needs of the database users. Rdb Controller implements all necessary work to modify SORTED and HASHED indexes and structure them for optimal performance. It generates a SQL procedure that will change the database and tune it accordingly.

Minimize Locking Occurrences

Rdb Controller can assist with these advanced tuning requirements in several ways: by locating hashing opportunities and providing record counts, record sizes, and index information to calculate appropriate page size and allocation.

Rdb Controller can be used to implement and properly tune for clustering and shadowing.

I/0 Distribution

Rdb Controller automatically balances I/O for individual storage areas, or the entire database, to ensure your users have the quickest access to data. It identifies which tables are the most active, sizes their storage areas accordingly, distributes them across separate disk drives, and follows the rules of thumb identified in Section V.

The following I/O distribution rules are adhered to:

· Do not place all your indexes on a single disk drive when not using large RAID units with much built-in caching

· Separate the table and index if you are not using the PLACEMENT VIA option

· Keep RDA and SNP files on separate disks

· Keep your busiest storage areas on separate disks

VIII. About ALI Database Consultants

ALI Database Consultants is the leader in comprehensive database management solutions that promote the value of IT by improving the performance and reliability of business-critical applications. Today's growing, database computing environments have become increasingly complex, often producing application performance and reliability problems. ALI Database Consultants recognizes that these applications are the lifeblood of an organization and has the experience, expertise, and tools in database management to deliver the right mix of capabilities and services to support IT.

ALI Database Consultants has provided database management solutions since 1995. Worldwide, customers include St. Laurent, Raytheon Aircraft, Ericsson, Intel, Koch Industries, National Revenue Corporation. SUNOCO Chemicals, Swedish Match, Westinghouse, and DuPont. The company's flagship product, *Rdb Controller,* is a complete database management solution. ALI Database Consultants (and its predecessor Empirical Software) was ranked as one of the nation's TOP 100 companies by DM Review in 2000, one of *Software Magazine's* Software 500 in 1997, and Hot 100 Private Companies by *Upside Magazine in 1996*. The company's strategic partnerships with a broad network of value-added resellers provide ready access to ALI Database Consultant's comprehensive solutions for database management.

Contact ALI Database Consultants on the Web at www. aliconsultants.com or by calling 1-803-648-5931 or Fax 1-803-641-0345.