Enhancing Application Performance with Oracle Rdb Row Cache

Norm Lastovica, Senior Principal Consultant
Oracle Rdb Engineering

Performance Problems?

If your database application is constrained by system I/O or locking performance, the Oracle Rdb Row Cache feature may be just what you are waiting for. Tremendous throughput and response improvements can be easily achieved. Row Cache gives you additional ability to maximize system performance with no application changes.

When using Row Caches, all processes can share a pool of row occurrences that reside in shared memory. No disk I/O or page locking is needed to share a row in a Row Cache. Only the rows of interest, not the physical database pages, are kept in a Row Cache.

You can create many Row Caches, each with its own set of characteristics. Row caches can be used to efficiently store rows of specific sizes from specified tables. Tables in a cache can be "pinned" or locked in memory. You can specify portions of Row Caches to occupy process private virtual memory, shared memory, or shared physical main memory. Oracle Rdb row caching also allows you to use very large memory (VLM) on OpenVMS Alpha systems.

Why do you want Row Cache?

Row caching is designed to improve performance through reduced I/O and locking by keeping rows of interest in a memory cache. Data can be read or modified directly in memory without having to lock or access database pages and page buffers. The more a cache is accessed, the more useful the cache is and better overall performance results. Toward this end:

- You can cache sorted indexes to reduce page locking and I/O. All the index nodes can be cached in memory.
- Tables can be cached for greater data density in memory. No database page overhead is stored in the Row Cache. As a result, memory is more efficiently used.
- You can take better advantage of physical memory on the system. Row caches can be placed in very large memory (VLM), breaking through 32-bit address space limits.
- To free up more process memory address space, you may be able to reduce the number of global buffers used for a database.

Dramatic performance gains: two real-life examples

Example 1: a credit card company.

By implementing several Row Caches, this business reduced its processing time for a critical end-of-month reporting job from 7 hours to 7 minutes (a sixty-fold time reduction) with no changes to the application or system.

Example 2: a trading application.

Another business, a large financial sector OLTP system, saw a 250% application throughput gain. The DBAs used RMU/SHOW STATISTICS on the running system to determine what the "hot" indexes and tables were. They then used RMU/ANALYZE/INDEX to determine the number of nodes in the target indexes and the table cardinalities. Based on that information, the existing database was modified to cache 4 tables and 28 sorted indexes. In total, approximately one million rows were cached into VLM on an AlphaServer system. With no other changes, back-to-back performance tests with and without Row Cache demonstrated the 250% throughput gain.

How it works: accessing cached data

A Row Cache exists between the application and database page buffers. Database functions using Row Caches can access data in the cache without ever having to access a database page buffer. All row accesses (reading, writing, and erasing data) are performed with Row Caches.

When you request a row from a database, Oracle Rdb first checks to see if the requested row is located in a Row Cache. If the row is found in a Row Cache, the row is retrieved directly from the cache. If the row is not in a cache, Oracle Rdb checks the page buffer pool. If the row is not in the page buffer pool, Oracle Rdb performs a disk I/O operation to retrieve the row. The requested row is then inserted into the Row Cache, if possible.

When a new row is stored in the database, Oracle Rdb will access database pages to find and reserve space for the new row and get a database key (DBKEY) for the row. This may require disk I/O. Once space has been reserved on the database page, Oracle Rdb checks for a Row Cache in which to put the new row. The new row is inserted into a Row Cache, if possible.

If a modification to a row in a cache causes the row to grow (replaces a null value, for example), then a disk I/O operation may be performed to reserve additional space for that row on the database page. If the database page does not have room for the modified row, resulting in record fragmentation, then the row is removed from the cache. If the modification keeps the row the same size or makes it smaller, then the modified row remains in the cache and no disk I/O operation is done.

In order to delete a row that is in a Row Cache, Oracle Rdb sets the length of the row in the cache to zero to erase it. It is not immediately erased from the database page on disk.

Sequential scans

When the execution strategy for a query is a sequential scan, Oracle Rdb scans the physical areas by performing the same I/O operations it would if there were no Row Caches. The major reasons for this are as follows:

- Oracle Rdb does not have a list of all database keys (DBKEYs) in an area; it materializes them by reading all pages and examining all lines on each page. However, data is returned from the Row Cache if it is found there.
- Although Oracle Rdb reads the database pages to find the DBKEYs of rows in the table, it still needs to look in the cache to see if the row is there. A row in the cache may contain more recent data than that which is on disk.
- There is no guarantee that all rows in a sequential scan can fit in a Row Cache. Row caches are often sized to include only some of the of rows so the most commonly used rows can be shared in memory.

Oracle Rdb avoids populating the cache during a strict sequential scan. Otherwise a query performing a sequential scan of a table looking for just a few records would fill the cache with every record and might force existing data in the cache back to disk. This would result in a Row Cache filled with records that you do not need in the cache.

How it works: when snapshots are enabled

During a read-only transaction, Oracle Rdb first checks to see if the row is in a Row Cache. If the row is found and is visible to the transaction, the row is returned from the Row Cache. However, if the row is not visible, Oracle Rdb must find the visible version of this row in the snapshot file.

During a read/write transaction that is performing a record update or delete, Oracle Rdb writes the before-image of the data to the snapshot file. If a read/write transaction updates a data row multiple times, then the snapshot file and database file are updated only when the row is first modified. No disk I/O operations occur for subsequent modifications to the row within the same transaction.

How it works: "lazy" updates and the RCS process

When you use row caching, users can make changes only to rows that are in memory. The after-image journal (AIJ) file is always written with the current row contents, but the database on disk may not be updated for a very long time. A Row Cache server (RCS) process writes modified rows back to disk (either to the database directly or to "backing store" files).

Each database with Row Caches has a Row Cache server process that is responsible for writing rows from the caches to disk. The RCS process performs "checkpoints" and "sweeps." During a Row Cache checkpoint operation, all modified rows (or all rows) from the Row Caches are written to disk storage.

During a Row Cache sweep operation, a set of modified rows are written back to the database from the Row Cache. After the rows are written back to disk, the space they occupied is considered selectable for reuse. A Row Cache sweep operation is initiated when a user process tries to insert rows into a Row Cache and no free space is available.

Database requirements for row caching

To use the Row Cache feature, an Oracle Rdb database must meet the following configuration requirements:

- The number of cluster nodes must be one
- After-image journaling must be enabled
- Fast commit must be enabled
- One or more Row Cache slots must be reserved
- Row caching must be enabled

Row Cache types

There are two types of Row Caches, physical area and logical area.

Physical area

You can create a general Row Cache that is shared by all row types that can reside in one or more storage areas. This is the basic type of Row Cache, called a physical area Row Cache. Because physical area Row Caches are associated with one or more storage areas, several storage areas can map to the same physical area Row Cache. A physical area Row Cache can contain all row types from the associated storage areas. In addition, when a physical area Row Cache is defined, all rows of different sizes in the associated storage areas are candidates for the Row Cache.

Logical area

You can create logical area Row Caches when you create a Row Cache using the same name as an existing table or index. A logical area Row Cache is associated with all partitions of a specific table or index.

Sizing a Row Cache

When you size a Row Cache, you may specify the following:

Slot size

The slot size is the size of the largest row that can be stored in the row cache. Oracle Rdb will not cache a row if it is too large to fit in the cache. Use the ROW LENGTH IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the size of the largest row in the cache.

Oracle Rdb automatically rounds up the row length to the next 4-byte boundary if the value specified is not divisible by four. This is done because longword aligned data structures are optimal for performance.

If you do not specify a slot size when creating a logical cache, Oracle Rdb generates a slot size based on the size of the table row. Note, however, that Oracle Rdb finds the nominal row length of tables and indexes using the area inventory page (AIP). Under certain circumstances this AIP length may not be the actual length of the row. If no entry can be found, Oracle Rdb uses a default length of 256 bytes. Also, if the metadata for a table is modified, then the AIP length is not automatically updated. This can result in incorrect cache sizing. See the Oracle Rdb Guide to Database Performance and Tuning for more details on AIP lengths.

Slot count

The slot count is the number of rows that can be stored in the cache. Use the CACHE SIZE IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the number of rows that can be stored in the cache. If you do not specify the CACHE SIZE clause, Oracle Rdb creates a cache of 1000 rows by default.

The following example shows a Row Cache definition specifying a row length of 200 bytes and 3000 rows in the cache:

```
SQL> ALTER DATABASE FILENAME 'MF_PERSONNEL'
cont> ADD CACHE RCACHE_1
cont> ROW LENGTH IS 200 BYTES
cont> CACHE SIZE IS 3000 ROWS;
```

It is important to select a proper size for the Row Cache. As stated previously, if a row is too large, Oracle Rdb will not cache the row. This can result in poor system performance because Oracle Rdb always checks the cache for the row before retrieving the row from disk. Keep in mind that row sizes within a table can vary greatly. If, for example, the largest row stored in a table is 100 bytes, but the majority of the rows range between 40 and 50 bytes, you may not necessarily want to choose 100 bytes for the slot size. However, you should account for most of the rows, including overhead. If you automatically select the largest row size without comparing it to the sizes of the other rows in the table, you might waste memory.

Summary

When Rdb database applications are constrained by I/O performance or locking constraints, the Oracle Rdb Row Cache feature may be able to help you make large performance gains with no application changes. Read-only caching is supported starting with Oracle Rdb V7.0.1 and read-write caching is supported starting with Oracle Rdb V7.0.1.5.