Internals of Oracle Database Locks

Riyaj Shamsudeen

In this article, Riyaj Shamsudeen explains various data structures associated with locks and provides a cursory overview of some locking-related initialization parameters.

LOCKS (also known as "enqueues") are used to control the access to resources. Example of resources include rows, tables, redo threads, media recovery threads, and so forth.

Internal operation of locks

Before we dive in to the details of the internal locking behavior, it's important to understand the terminology that's used in this article.

What's a resource identifier?

When a session needs access to a particular resource, it requests a lock on that resource in a specific mode. The mode requested depends upon the intended operation. Resources are identified with the format {<two char resource type >, id1, id2}.

This two-character type identifies the type of the resource for which a lock has been requested. For example, TX identifies a transaction lock and is used for row-level locking. Id1 and id2 identify the specific resource within that resource type.

Another example is MR locks. MR is the resource type for the Media Recovery, and DBW0 holds locks on every datafile on this resource. Id1 is the file#.

A combination of the lock type, id1, and id2 is unique, and each resource will have a unique resource identifier.

Arrays and hash chains

A few arrays and a hash chain are used to track the enqueues and locks. The resources array is an array of resource structures. A resource structure is used to track a single resource, at any given point in time. Similarly, a locks array is used to track an array of lock structures.

A hash array is used to improve the performance of the enqueue lookup operation. There are buckets in this hash array, and each hash bucket is pointing to a hash chain. A hash chain is a linked list of resource structures, and the head of the linked list is linked with the hash bucket. Since these structures need to be protected during the read / write operations to this array, latches are used. Holding a latch, starting from the bucket, the hash chain is searched for the specific resource.

These arrays are fixed arrays and are statically allocated during the startup and don't grow dynamically. The length of the resources array is determined by the enqueue_resources parameter. The length of the locks array is determined by the _enqueue_locks parameter. The number of hash buckets is determined by the _enqueue_hash parameter. The number of latches protecting these hash buckets is determined by the _enqueue_hash_chain_latches parameter.

x$ tables

These fixed arrays are externalized as x$ tables. The resources array is externalized as x$ksqrs fixed table, and the locks array is externalized as x$ksqeq fixed table.

Tracking the resources

When a session requests a lock on a resource, a unique resource identifier is generated of the format (type, id1, id2). This resource identifier is hashed in to the enqueues hash table to determine the status of the resource. Holding one of the enqueue_hash_chain latches, this hash chain is searched for the specific resource (see Figure 1).

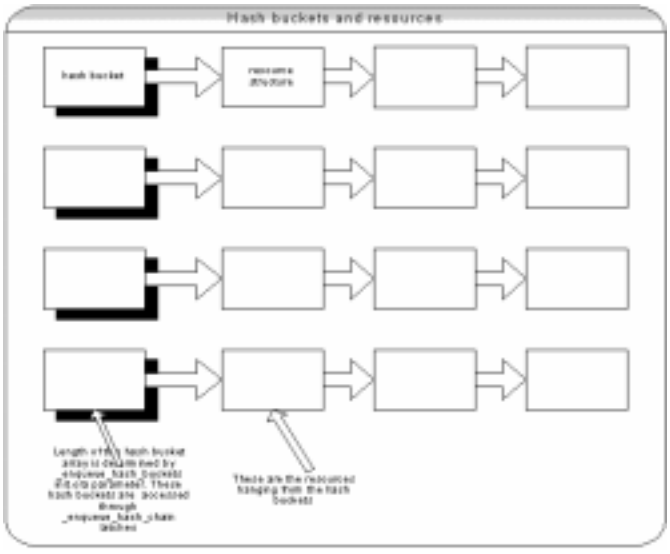

Figure 1. Hash buckets and resources.

Now depending upon whether that particular resource has been locked or not, the operation differs.

Resource available

If the resource isn't in the hash chain, then no process is holding a lock on the resource. So the requesting session can take a lock on this resource. One of the free array elements from the resources array is selected, and it's linked with a bucket of the hash array.

Now the resource has been set up, and a lock on this resource needs to be set up as well. An element from the locks array is linked with the resource structure in the owner's queue, and proper lock properties are set up. These lock structures are implemented as queue (as a two-way linked list). These lock structures are linked with the resource structure connected by the linked list (see Figure 2).

Furthermore, all of these structures are embedded in specialized state objects for PMON's recovery.

Resource unavailable

If the resource is unavailable, then the resource is already allocated. If the resource is in the hash chain, then there's no need to allocate a new resource structure. But a new lock structure needs to be allocated. The requested mode is checked with the mode held. If the resource is locked in an incompatible state, then a new lock structure is selected from the lock array and attached to the waiters queue of the resource structure.

Processes waiting for the resources sleep on the semaphores, and semaphores (or the post wait driver on a few platforms) are used as sleep / wakeup mechanisms. After enqueueing in to the queue, the requesting process will sleep on the semaphore using the sync_op call.

```
sync_op(SYNC_WAIT, SYNCF_BINARY, 300)          = 1
```

Once the process holding the resource is ready to release the resource, it looks at the queue attached to the resource structure. If there's a process in the queue, it sends a semaphore signal to the waiting process using sync_op call.

```
sync_op(0x0005, SYNCF_BINARY, 134491620)       = 1
```

The waiting process will handle the signal and will wake up. This waiting process will modify the status of the resource structure, unlink its own lock structure, and proceed.

Other features

There are a few other features available to improve the performance of lock operations.

Vector post

_use_vector_post is implemented in a few platforms. This is implemented using the SYNC_POSTVEC option of the sync_op call. It's used to signal multiple processes sleeping on semaphores to be signaled in a single sync_op call. Certain background processes can take advantage of this feature and can pass the list of PIDs or process handles to be signaled. With this implementation, multiple processes can be awakened using a single sync_op call. Since numerous calls to the sync_op call are avoided, and the resulting context switch overhead is also avoided, there's a slight performance improvement.

Post wait drivers

Some platforms support the post_wait_driver in place of semaphores. Semaphores are implemented using the OS system calls, and each operation on the semaphore involves system calls. Hence, context switches take place for each of these calls.

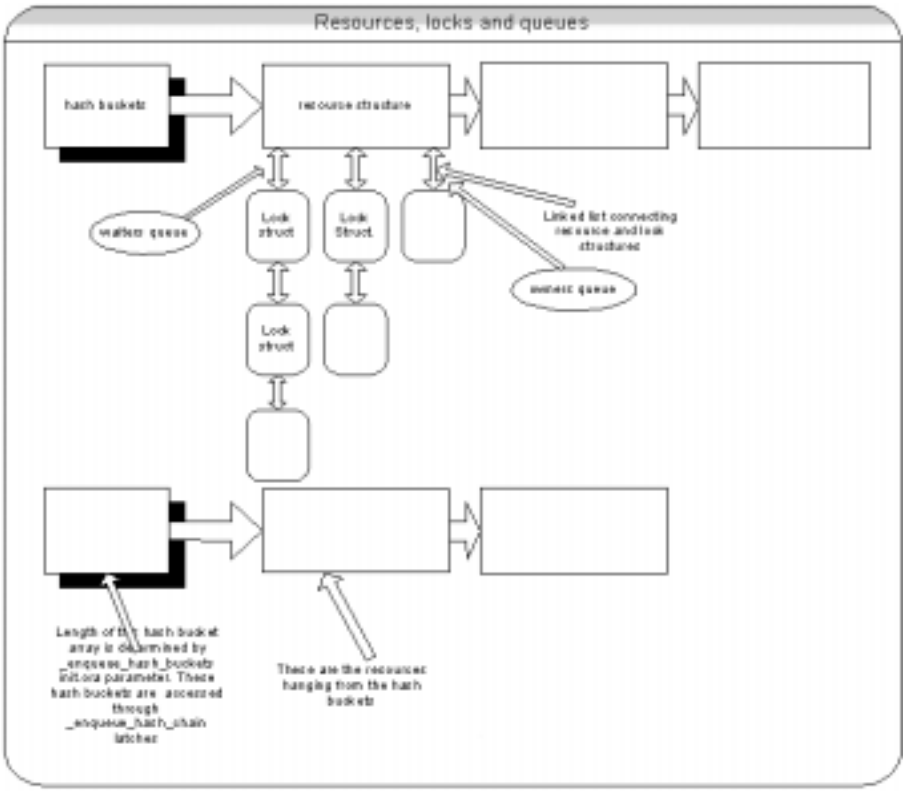

Figure 2. Resources, locks, and queues.

Continues on page 19

Oracle Database Locks...

Continued from page 15

Post_wait_drivers are implemented in user space, without the need for a kernel context switch, to minimize the overhead associated with process signaling and control.

Resource limit

To determine whether there are enough slots in the resources array, you need to query the v$resource_limit. The enqueue_resources parameter is usually over-allocated by the DBA, and each resource structure takes around 72 bytes of memory. The _enqueue_locks parameter is derived and best left undisturbed.

Enqueue dumps

You can dump the enqueues with the following command:

```
Alter session set events 'immediate trace name enqueues
level 3'
```

Here are details for various levels for the dump:

Level 1	enqueues hash table
Level 2	hash table + resources
Level 3	hash table + resources + locks + queues

Riyaj, this line is too long—how should it break?

Enqueue frequency and distribution

To find the frequency and distribution of enqueue waits, you could use the x$ksqst table. This table lists various enqueue types and the number of times a process has waited for this resource. Since this doesn't list the impact of wait on these enqueues, use this information with caution.

All right, how do I use this information?

Now we'll take a look at some real-world applications of the information I've presented here.

Tuning

Knowledge of internal operation is a key to performance tuning. If your application has locking contention issues, then you might be able to use this information to pinpoint the problem area and tune it. For example, to find the breakup of enqueues, you could use x$ksqst table. If you have waits on the enqueues hash chain latches, then you could increase the enqueues_hash_chain latches to decrease contention and improve performance.

At times, LGWR has to post many processes using many semaphore calls. Vector posting can be enabled if that's the case to improve the performance.

ORA-52, ORA-53 errors

If you receive either of these errors, it indicates that you ran out of slots in the resources or locks array, respectively. Check the v$resource_limit table and increase the size of the resources and locks array using the enqueue_resources and _enqueue_locks init.ora parameters, if necessary.

Disclaimer

Oracle resource locking and management is a complex subject. Due to space constraints, some details have been omitted. Additionally, I can only speak to the relevance of the information in this article as it relates to the Sun, Sequent, and HP platforms. However, other UNIX platforms can probably be expected to provide similar services and functionality as outlined here. The Windows NT platform might introduce other problems or subtleties not explored in this article. ▲

Riyaj Shamsudeen is a certified Oracle DBA, currently working for i2 technologies. He's worked with numerous versions of Oracle and many flavors of UNIX. Performance tuning is his favorite part of the job. riyaj_shamsudeen@i2.com.

Downloads May 2001 Source Code

- HOTKA05.ZIP—Source code to accompany Dan Hotka's article, "Geeky Internal Stuff: Rollback Segment Internals."
- FEUER05.ZIP—Source code to accompany Steven Feuerstein's article, "Bulking up with Oracle8i PL/SQL—Part 2."

Oracle Professional Subscription Information:1-800-788-1900 or http://www.oracleprofessionalnewsletter.com

Subscription rates:
United States: One year (12 issues): $199
Two years (24 issues): $358
Canada:* One year: $214; two years: $373
Other:* One year: $219; two years: $378

Single issue rate: $25 ($27.50 in Canada; $30 outside North America)*

Australian newsletter orders:
Ashpoint Pty., Ltd., 9 Arthur Street,
Dover Heights, N.S.W. 2030, Australia.
Phone: +61 2-9371-7399. Fax: +61 2-9371-0180.
E-mail: sales@ashpoint.com.au
Internet: http://www.ashpoint.com.au

* Funds must be in U.S. currency.

Contributing Editors Steven Feuerstein, Dan Hotka; **Publisher** Robert Williford; **Vice President/General Manager** Connie Austin; **Executive Editor of IT** Farion Grove; **Executive Editor** Heidi Frost; **Editorial Assistant** Micki Bailey

Direct all editorial, advertising, or subscription-related questions to Pinnacle Publishing, Inc.:
1-800-788-1900 or 770-992-9401
Fax: 770-993-4323
Pinnacle Publishing, Inc.
PO Box 769389
Roswell, GA 30076-8220
E-mail: **oracledev@pinpub.com**
Pinnacle Web Site:
http://www.pinnaclepublishing.com
Oracle technical support:
Call Oracle Corporation at 414-506-1500

Oracle Professional (ISSN 1525-1756) is published monthly (12 times per year) by Pinnacle Publishing, Inc., 1000 Holcomb Woods Pkwy, Bldg 200, Suite 280, Roswell, GA 30076.

POSTMASTER: Send address changes to *Oracle Professional*, PO Box 769389, Roswell, GA 30076-8220.

DOWNLOAD

The Source Code portion of the *Oracle Professional* Web site is available to paid subscribers only. Log in for access to all current and archive content and source code. For access to this issue only, go to www.oracleprofessionalnewsletter.com, click on "Source Code," select the file(s) you want from this issue, and enter the User name and Password at right when prompted.

User name `iodine`
Password `jealous`