Oracle Database Locking Mechanism Demystified

Pini Dibask
Quest Software
Israel

Keywords: **Locks, Performance, Database, Scalability, Concurrency, Consistency, Integrity**

Introduction

The purpose of locks in Database systems is to maintain Database concurrency and consistency when multiple transactions access the same shared resources. The reason these are "shared resources" and not just "tables" is because there are various types of shared resources in a Database, it could be rows in a table, entire table, index, stored procedure, etc.
The need for locking mechanism is a result of having multiple user sessions which may access same shared resources and by doing so they can cause "logical corruptions" to the data. In fact, if in our Database systems world, we always had one single session in the Database that is running and updating data or altering data structures, locks were not required.

Concepts of Locking Mechanism in Oracle Database

Different RDBMs (Relational Database Management Systems) have different locking mechanisms, each RDBMS has its own concepts and nuances. This manuscript and the associated presentation will cover the Oracle Database locking mechanism.
In Oracle, a reader (someone who executes a SELECT query), never creates any locks, so other readers or writers will never be blocked because of a reader; having said that, there is only one exception for that and it is when someone executes the [SELECT …. FOR UPDATE] statement.
A writer, never blocks other readers. For example, if a writer updates/deletes/inserts a row and the transaction has not been ended yet, other readers can simultaneously query the data in the table without being affected by the writer operations. Oracle manages to accomplish that by storing a "before images" of Database blocks stored in a tablespace named "Undo Tablespace", so even if a writer modifies data, Oracle still has access to the table's data as if the writer's transaction never occurred. Once the transaction is committed, other readers will be able to see the writer's modifications.
During DML modifications (inserts, updates, deletes), Oracle acquires locks at the row-level. The only time when Oracle blocks a session is when there are concurrent writers. For example, if one session updates a row in a table, then as long as the transaction has not been ended, other sessions cannot update that row, but they can definitely read the table, as previously mentioned.

Generally speaking, Oracle Database has 2 lock modes: share locks and exclusive locks. The major difference between these 2 lock modes is that many share locks can be acquired on a single resource in a given point in time, but exclusive lock cannot be shared on a single resource. For example, assuming the user has updated a single row in the EMPLOYEES table by executing the following statement:

```
UPDATE EMPLOYEES
SET CITY = JERUSALEM
WHERE EMPLOYEE_ID = 123;
```

In this case, Oracle automatically acquires an exclusive row lock on the row for the EMPLOYEE_ID 123 and a share lock on the EMPLOYEES table, meaning that as long as the transaction has not been

ended, other sessions cannot acquire an exclusive row lock for EMPLOYEE_ID 123, but they can definitely acquire exclusive row lock on different rows and share table lock; this also ensures that other sessions cannot acquire exclusive lock on the table, for example by trying to alter the table's structure), because the table is already locked in a share table mode.

DDL Operations

When executing a DDL operation such as CREATE TABLE, DROP INDEX, ALTER TABLE, Oracle will implicitly commit the existing transaction and will auto commit the DDL command, i.e. there is no option to rollback the DDL command. Since DDL commands almost always acquire exclusive lock on the object, any other share or exclusive locks on the table will not allow to execute the DDL as Oracle will raise "ORA-00054: resource busy and acquire with NOWAIT specified" error. For example, even if one row is being updated in a table that holds millions of records, then as long as the transaction for the update statement is still running, Oracle won't allow DROP or ALTER operations against that table.

Note: Starting from Oracle 11gR1 it is possible to specify a time limit for how long DDL statements will wait in a DML lock queue by setting the DDL_LOCK_TIMEOUT parameter. The default value of zero indicates a status of NOWAIT. The maximum value of 1,000,000 seconds will result in the DDL statement waiting forever to acquire a DML lock.

Deadlocks

Deadlock is a locking scenario that occurs when two or more sessions are blocked as they wait because of a lock that is being held by the other session. Oracle Database automatically detects this scenario and performs a statement-level rollback to one of the sessions, because otherwise, they will wait forever as both of them are blocked and waiting to each other's locked resources. The session that its statement is being rolled back, will encounter an "ORA-00060: Deadlock detected while waiting for resource." error message (that will also be recorded in the alert log file). A common misconception is that Oracle handles deadlock events by terminating one of the sessions. While that might be the case in other Databases systems, that's not the case in Oracle Database.

Blocked Inserts

When inserting a row to a table that has either a primary key or unique constraint, the INSERT statement will be blocked if another running transaction already inserted the same value for the column which has the primary key or unique constraint. The reason for that is because Oracle must enforce uniqueness on the column that has the primary key or unique constraint defined. If a session has opened a transaction and tried to insert the value 'X' for a column with unique constraint, and another session tries to insert the same value to the same column, then the second transaction will be blocked until the first transaction will be ended by either a commit or a rollback. If the first transaction will be committed then the second session will encounter the "ORA-00001: unique constraint (string.string) violated" error.

Unindexed Foreign Keys

Let's assume that we have a child table named EMPLOYEE and a parent table named DEPARTMENT. In the child table (EMPLOYEE) we have a foreign key on the DEPARTMENT_ID column. If the DEPARTMENT_ID column in the EMPLOYEE table is not indexed, Oracle will place a full table lock on the EMPLOYEE table and will not allow any DMLs on the EMPLOYEE table if a user deletes an entire table row from the parent table, or if a user updates the value of the parent's table primary key. Unindexed foreign keys can be the cause of lock issues in general and of deadlocks in particular in many cases.

Monitoring Locks in the Database

Oracle's locking mechanism is efficient as DML statements which modify rows acquire locks at the row-level. Even if a user will update 5,000 rows in a table that contains 6,000 rows, oracle will not lock the entire table but rather only the 5,000 rows which have been updated. That's not the only reason why it is efficient; it is also efficient as readers never block writers and writers never block readers. Yet, in some scenarios, if the application is written in a way that is causing lock waits activity, i.e. causing sessions to be wait as they are blocked; this can definitely impact the Database performance as Oracle sessions will spend time on lock wait events which are associated to the "Application" wait category. These locks can be monitored by using various Oracle dictionary views such as the following views (their definition is taken from the Oracle documentation):

V$LOCKED_OBJECT - Lists all locks acquired by every transaction on the system. It shows which sessions are holding DML locks (that is, TM-type enqueues) on what objects and in what mode.

DBA_WAITERS - Shows all the sessions that are waiting for a lock. In an Oracle RAC environment, this only applies if the waiter is on the same instance.

DBA_BLOCKERS - Displays a session if it is not waiting for a locked object but is holding a lock on an object for which another session is waiting. In an Oracle RAC environment, this only applies if the blocker is on the same instance.

The above views can provide valuable information when it comes to investigating locking scenarios; however, DBAs are faced with a serious challenge when they are required to investigate historical lock issues in their Databases. For example, if a user complains on lock scenarios which occurred days/weeks/months ago, the DBA would probably want to proactively investigate it so these issues will not occur again in the future. One way to do that is to query V$ACTIVE_SESSION_HISTORY dictionary view as it contains snapshots of active database sessions including those who waited for locks ; since the information is limited and stored in-memory, users can also access DBA_HIST_ACTIVE_SESS_HISTORY which displays the history of the contents of the in-memory active session history of recent system activity.

Note: Accessing V$ACTIVE_SESSION_HISTORY and DBA_HIST_ACTIVE_SESS_HISTORY dictionary view requires having a license for the Diagnostics pack which is an extra cost option of Oracle Database Enterprise Edition.

Another way to analyse historical lock issues is to use tools like Foglight for Oracle which is a 24x7 enterprise monitoring tool that enables you visualize the instance workload and in particular lock-related activity. It also allows you to view all the lock trees in any give point in time in the history, and filter the information by various dimensions such as: sql statements, users, programs, pluggable databases, services, sessions and more.

In the below screenshot for example, you can see all the lock trees which are associated to all the sessions that run with a client machine named "PROD\ISLARPS20" and with operating system user named "pdibask" in any point in time. In this case, the table in the "Blocking History" tab reveals 2 lock trees and for each lock tree it displays information about the blocking sessions and the blocked sessions as well as the lock duration, blocked SQL statement and more.

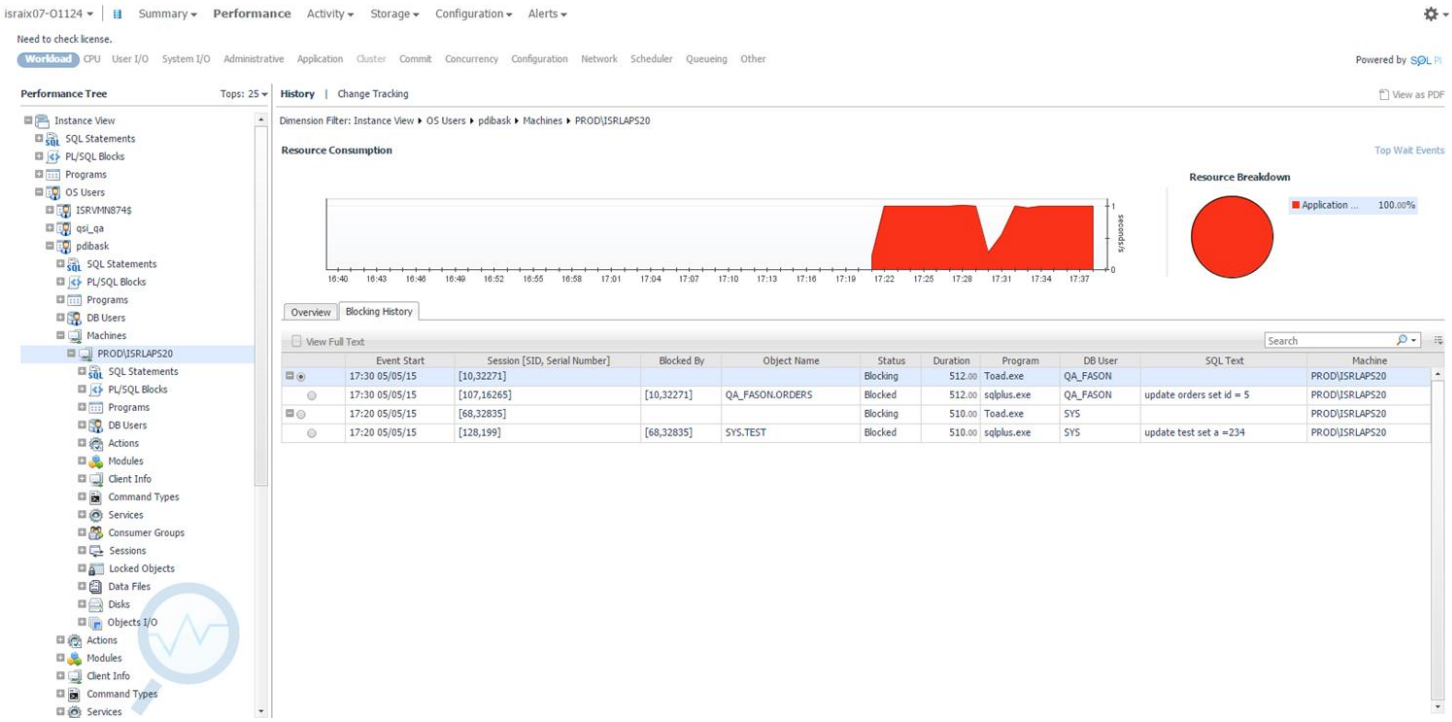

Illustration. 1: Investigating historical lock issues using Foglight for Oracle

Contact address:

Name
Quest Software
Hamifal 13
6022113, Or Yehuda

Phone: +972(0)54-7778395
Email : pini.dibask@software.dell.com
Twitter : @pini_dibask
Blog : http://OracleDBPro.BlogSpot.com
LinkedIn : http://linkedin.com/in/pinidibask
OTN Forums : https://community.oracle.com/people/Pini%20Dibask