

CONFIO SOFTWARE'S ORACLE RESOURCE NEWSLETTER

Welcome to this month's newsletter named The Oracle Resource. This newsletter provides technical insight into Oracle database Wait-Event performance issues.

Wait-Event Analysis for Oracle databases is the industry best-practice today for database tuning and performance optimization. Understanding the importance of each Oracle Wait-Event or "Resource" is essential for DBAs in keeping their database performing at its optimal level. Each month, Confio Software brings you this update to provide insight into a specific Oracle Resource, its significance in your Oracle system, and how to identify and address related problems. This month's Oracle Resource is:

read by other session

Definition: When information is requested from the database, Oracle will first read the data from disk into the database buffer cache. If two or more sessions request the same information, the first session will read the data into the buffer cache while other sessions wait. In previous versions this wait was classified under the "buffer busy waits" event. However, in Oracle 10.1 and higher this wait time is now broken out into the "read by other session" wait event.

Excessive waits for this event are typically due to several processes repeatedly reading the same blocks, e.g. many sessions scanning the same index or performing full table scans on the same table. Tuning this issue is a matter of finding and eliminating this contention.

Finding the contention

When a session is waiting on this event, an entry will be seen in the v\$session_wait system view giving more information on the blocks being waited for:



```
SELECT p1 "file#", p2 "block#",  
       p3 "class#"  
FROM   v$session_wait  
WHERE  event = 'read by other  
session';
```

If information collected from the above query repeatedly shows that the same block, (or range of blocks), is experiencing waits, this indicates a "hot" block or object. The following query will give the name and type of the object:

```
SELECT relative_fno, owner,  
       segment_name, segment_type  
FROM   dba_extents  
WHERE  file_id = &file  
AND    &block BETWEEN block_id AND  
       block_id + blocks - 1;
```

Eliminating contention

Depending on the database environment and specific performance situation the following variety of methods can be used to eliminate contention:

1. Tune inefficient queries – inefficient queries read too many blocks into the database buffer cache flushing out blocks that may be useful for other sessions. Tuning queries will lessen the number of blocks read into the buffer cache and reduce the aging out of "good" blocks. Finding the queries to tune is a matter of finding the ones accessing the "hot" objects found above.

2. Redistribute data from the hot blocks – deleting and reinserting the hot rows will often move them to a new data block. This will help decrease contention for the hot block and increase performance. More information about the data residing within the hot blocks can be retrieved with queries similar to the following:

```
SELECT data_object_id
FROM dba_objects
WHERE owner='&owner'
AND object_name='&object';
```

```
SELECT dbms_rowid.rowid_create(
    1,<data_object_id>,
    <relative_fno>,<block>,0)
start_rowid
FROM dual;
--rowid for the first row in the block
```

```
SELECT dbms_rowid.rowid_create(
    1,<data_object_id>,
    <relative_fno>,<block>,500)
end_rowid
FROM dual;
--rowid for the 500th row in the block
```

```
SELECT <column_list>
FROM <owner>.<segment_name>
WHERE rowid BETWEEN <start_rowid>
AND <end_rowid>
```

3. Adjust PCTFREE and PCTUSED – adjusting the PCTFREE value downward for an object will reduce the number of rows physically stored in a block. Adjusting the PCTUSED value for an object keeps that object from getting prematurely put back on the freelist.

Depending on the type of contention, adjusting these values could help distribute data among more blocks and reduce the hot block problem. Be careful to optimize these parameters so blocks do not move in and out of the freelist too frequently.

4. Reduce the Block Size – this is very similar to adjusting the PCTFREE and PCTUSED parameters in that the goal is to reduce the amount of data stored within one block. In Oracle 9i and higher this can be achieved by storing the hot object in a tablespace with a smaller block size. In databases prior to Oracle 9i the entire database must be rebuilt with a smaller block size.

5. Optimize indexes – a low cardinality index has a relatively small number of unique values, e.g. a column containing state data with only 50 values. Similar to inefficient queries, the use of a low cardinality index could cause excessive number of blocks to be read into the buffer cache and cause premature aging out of “good” blocks.

Conclusion

When a session waits on the “read by other session” event, it indicates a wait for another session to read the data from disk into the Oracle buffer cache. If this happens too often the performance of the query or the entire database can suffer. Typically this is caused by contention for “hot” blocks or objects so it is imperative to find out which data is being contended for. Once that is known this document lists several alternative methods for solving the issue.

For a general description of Oracle Wait-Event analysis and the results to be achieved, see www.confio.com/products.

For product details and pricing information, please contact Confio at (303) 938-8282 or Email info@confio.com.

Confio Software helps organizations leverage their investment in IT infrastructure. Through its core software products and Resource Mapping Methodology, Confio allows IT personnel to identify and prioritize performance bottlenecks within enterprise applications and resolve them faster than ever before. For more information, please visit www.confio.com.