Oracle Cache Buffers Chain Latch

**By**[**Janis Griffin**](http://logicalread.solarwinds.com/author/janis-griffin/)

[**http://logicalread.solarwinds.com/oracle-cache-buffers-chain-latch-jg01/#.VO3GfDSUfME**](http://logicalread.solarwinds.com/oracle-cache-buffers-chain-latch-jg01/#.VO3GfDSUfME)

A user process acquires the CACHE BUFFERS CHAINS latch to scan the System Global Area (SGA) for database cache buffers. Blocks in the buffer cache are placed on linked lists (cache buffer chains). Blocks are put on the hash chain according to their DBA (data block address) and CLASS of the block. Each hash chain is protected by a single child latch. The latch allows a process to scan a hash chain without having the linked list change while it scans.

**When this latch can become a problem**

High or rapidly increasing wait counts on the CACHE BUFFERS CHAINS latch is an indication that the latch is not able to service the different sessions fast enough.

Contention for these latches can be caused by very heavy access to a single block or heavy concurrent execution of poorly written SQL statements which touch too many blocks (i.e. same blocks). This latch used to be big problem; however, since Oracle9i, operations on this latch have been optimized significantly.

**Analysis**

Use [SolarWinds Database Performance Analyzer](http://www.solarwinds.com/database-performance-analyzer-oracle.aspx) or another tool to identify anybuffer chain contention problem. Then, determine the object that the block belongs to.  First, determine which latch waits are potential problems by examining the number of sleeps for this latch. The higher the sleep count, the more likely that the latch wait is a problem.

Use the following command to get the “sleep count” value, which you will then use in the nex command:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | select count(\*)    "cCHILD",         sum(GETS)   "sGETS",         sum(MISSES) "sMISSES",         sum(SLEEPS) "sSLEEPS"    from v$latch\_children    where name = 'cache buffers chains'    order by 4, 1, 2, 3; |

The “sleep count” value (obtained from the sSLEEPS column of the query output) is used as the lower limit for the search for relevant block accesses. Any value of the sSLEEPS column can be used to run the next query, below. Be sure to choose a relevant value from the top “n” resulting values.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | select /\*+ ordered \*/       e.owner ||'.'|| e.segment\_name segment\_name,       e.extent\_id extent#,       x.dbablk - e.block\_id + 1 block#,       x.tch,       l.child#    from       sys.v$latch\_children l,       sys.x$bh x,       sys.dba\_extents e    where       l.name    = 'cache buffers chains' and       l.sleeps > &sleep\_count and       x.hladdr = l.addr and       e.file\_id = x.file# and       x.dbablk between e.block\_id and e.block\_id + e.blocks - 1; |

Here’s an example of the output of this SQL :



|  |  |
| --- | --- |
| 1  2  3 | SEGMENT\_NAME       EXTENT#       BLOCK#    TCH     CHILD#  SCOTT.EMP          1             449       2       7,668  SCOTT.EMP\_PK       5             474       17      7,668 |

Depending on the TCH column (the number of times the block is hit by a SQL statement), you can identify a hot block. The higher the value of the TCH column, the more frequent the block is accessed by SQL statements. Be sure to run the query above multiple times to understand the trend, and avoid being misled by any one-time activity.

**Solving the cache buffers chain latch**

Where there are hot blocks, you can relieve contention by doing one of the following:

* Splitting the buffer pool into multiple pools
* Altering PCTFREE/PCTUSED to allow fewer rows per block, thus reducing contention on a certain block
* Reducing the frequency the application accesses the object in question.
* In older versions of Oracle, there are often far fewer “cache buffers chains” latches (especially with large buffer caches) and so there can be many buffers covered by a single hash latch. An upgrade might alleviate the problem.
* Tuning queries so that they won’t touch as many blocks. This will alleviate the problem with this latch if the query is heavily executed.
* Avoid doing too many concurrent DML and Queries against the same row/block. Too many concurrent DML and Queries against the same block can result in multiple versions of the block created in the same cache buffer chain. Longer chains means more time spent by the session traversing through the chain while holding on to the latch.