Cache Fusion Demystified

By Arup Nanda

Introduction

Let me start by asking a question. Is it true that if the size of the database is only 100MB, then all I need is a 100MB buffer cache and the sessions never have to go to the disk? As a corollary, shouldn't this statement be true as well: anything more than 100MB in buffer cache is wastage of memory? A vast majority of the folks I asked this question to answer yes. While that is not quite surprising considering how pitiful the documentation is on the mechanism of buffer cache management and the resultant lack of understanding, it is quite disturbing since your ability to tune performance of a RAC database depends heavily on that knowledge. The answer to the question is no; 100MB is not enough. In a RAC database you need an even bigger buffer cache. That should be even more surprising if you don't understand the answer to the first question. RAC should take advantage of the Cache Fusion mechanism to bypass the disk access in most cases going instead to the buffer cache of the other instances; so the buffer size requirement should be less; not more, right?

Speaking of Cache Fusion, have you ever wondered how it knows where to get the block from – from the local cache or the remote cache? If it comes from a remote cache, then that of which instance? Also, do you wonder what the terms - Global Cache Service (GCS), Global Resource Directory (GRD) and Global Enqueue Service (GES) – are for, apart from containing the word "Global" in each of them? You might have heard about a term called "Buffer Lock" in the context of RAC. If you want to know how that is different from a row lock and everything about all the others, this article is for you. After finishing this, you will understand:

(1) How buffers are allocated and deallocated in the cache, especially in RAC
(2) What is the difference between CR and Current modes
(3) What are buffer locks and how it is different from row locks
(4) What are different types of Buffer Locks – Exclusive Current and Shared?
(5) How CF decides to get the buffers from different instances

Why understanding all these is important? In the very least, this will help you design the proper buffer size. But more important, it will help you in doing performance tuning or troubleshooting in a RAC environment.

Buffer States

In a RAC environment, there are at least two buffer caches; there may be more. When a block is requested by a session, the buffer cache is searched in the local instance. If it is not found, there are two options for the server process – to get from disk or get from the cache of one of the other instances. Even if the buffer is found in the local cache, there are three options – send the buffer to the user, examine other caches for the presence of this buffer in a more compatible state and get it from the disk anyway. To decide which option to take, Oracle relies on the many factors, one of which is the buffer state.

For clearer explanation, I need to make sure you understand the difference between a block and a buffer. A block is the least addressable unit of a database and it's on the disk. There is only one copy of a block in the database. When a user requests a row in the block, the server process loads the entire block from the disk to the buffer cache and returns it to the user. In the buffer cache, the buffer can be in various states, depending on how it was retrieved. When a user requests a buffer, it can be retrieved in one of the two modes:

- Consistent Read (CR) mode – when the buffer is requested to be simply be selected; not updated
- Current mode – when then buffer is requested for the intention of modifying it, even if actually not updated, e.g. in case of SELECT FOR UPDATE.

When another session requests the same block; but as of a different time (or, more accurately, different SCN), Oracle must provide a read consistent image of the buffer; not just one found in the cache. To accomplish this, the server process must create a new copy of the buffer. There can be several CR buffer copies of a single block; but only one current mode. This is completely logical; the word "current" indicates the buffer is the most recent; there can't be more than one current copy.

Remember, in a RAC database there is more than one buffer cache. Each cache may contain its own current copy of the buffer. In that case, each current buffer is said to be in Shared Current mode. If a buffer is modified, then the copy in the instance is the more recent that the others. That buffer is said to be in Exclusive Current mode. Only one buffer in the entire cluster can be Exclusive Current. If there is one, then all other Shared Current buffers can no longer be called current; so they turn to CR copies.

Block – Row Relationship

Let's take a closer look at the buffer states and modes. Suppose there is a two node RAC database, in which there is a table with only four rows all of which fit into just one block. In the beginning of the transaction, suppose the SCN number was 10, which is the SCN number of the block on the disk. Consider a session connected instance 1 updates only a single row of the table – Row1. The server process will retrieve the block from the database in a current mode (since the intention is to modify the block) and modify the row1, as shown in Fig 1. The SCN number of the buffer that time is 20. Instance 2

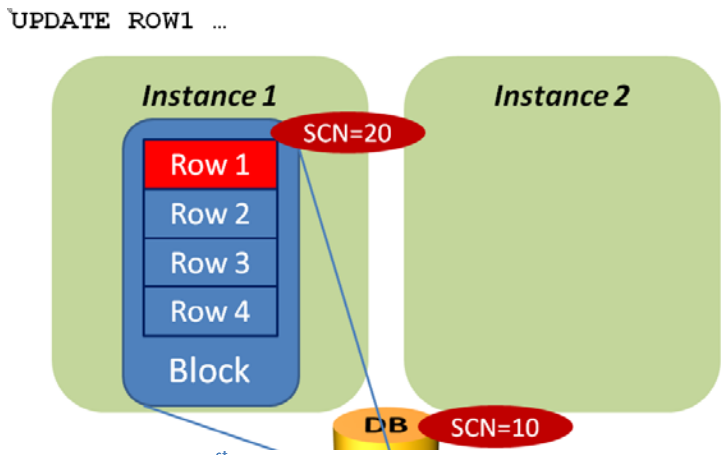

Figure 1 State after 1st Update

still does not have the buffer; because no one there has asked for any row in the block.

Now a session connected to instance 2 updates a different row – Row2. It can do that since the session 1 has placed a lock on Row1; not Row2. To update, the server process finds that a copy of the block is available in the buffer cache of instance 1; so it requests the buffer in current mode (since the intention is to modify). The buffer is transferred via cache fusion. To modify the buffer, the instance must have the buffer to itself with no other instance potentially modifying it. To accomplish that objective, the instance must request to put a lock on the buffer – an **Exclusive Current** lock. Until it gets the lock, the instance is not allowed to modify the buffer. After that lock, the buffer in the cache of the other instance becomes CR, since it is no longer current, as shown in Fig 2.

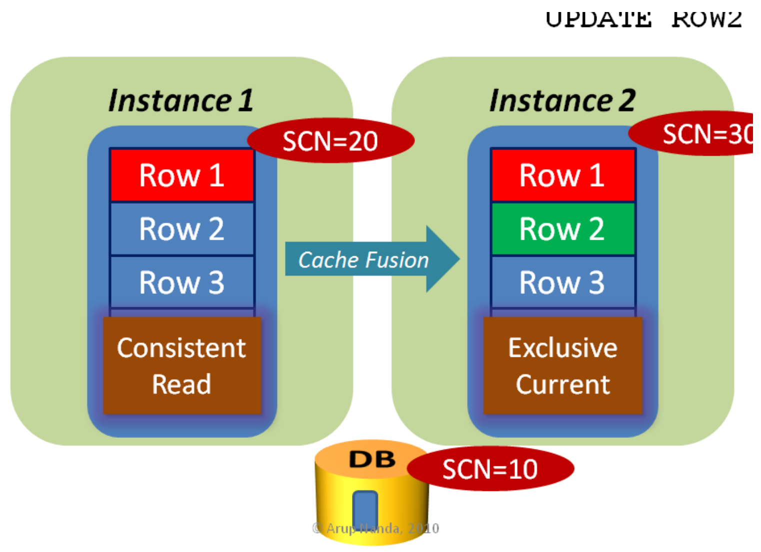

Figure 2 State after 2nd Update

When a checkpoint occurs and the dirty buffer is written to the disk, the SCN numbers on the block on the disk and the buffer in the cache become the same – making the buffer in the cache non-exclusive. A process reading the buffer from the cache or the disk would get the same result. Therefore the Exclusive Current copy of the buffer turns to a CR copy after it is written to the disk.

Had the instance 2 asked for the buffer to simply select from it, it would have got the buffer in CR mode. The instance 1, in that case, would have created a CR copy to be sent to the other instance. This process is known as **CR Processing**.

Consider a different session connected to instance 1 updates yet another row – Row3. In this case, instance 1 requests the buffer from instance 2. Once placed in the cache, the buffer must be made Exclusive Current, since the intention is to modify it. The buffer in instance 2, which was in Exclusive Current, now becomes CR, as shown in Fig 3.

The exclusivity of the buffer in a specific instance to modification is enforced by placing a special lock on the buffer, called a Buffer Lock, also known as Parallel Cache Management (PCM) lock. This is different from the row lock. Refer to Fig 3. Row1 is locked by session1, Row3 is locked by session3 and Row2 is locked by session2. There are two copies of the block, or two buffers on instance 1, one with XCUR lock (for Exclusive Current mode) and the other with a NULL lock (for the CR copy). On instance 2, there is only one copy of the buffer with a NULL lock. These are buffer locks, different from the row locks. Had the sessions committed after every modification, there would not have been any row level locks; but the

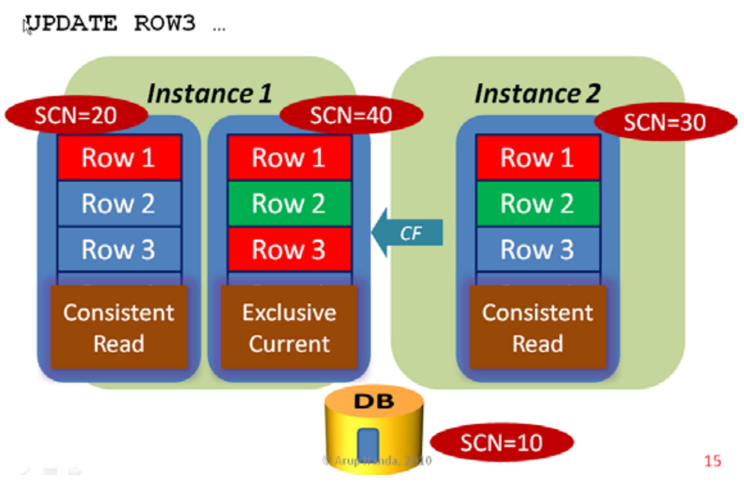

Figure 3 State after 3rd Update

buffer locks would still be there. Similarly, after the checkpoint, the buffer locks would have changed from XCUR to NULL but the row locks would still be there. As you can clearly see, there is no relationship between row and buffer locks.

Past Image

There is one more possible mode of the buffer. To understand it, consider the event of instance recovery. In Fig 2, if the instance 2 crashes, what would be the impact on the block on the disk? None. The buffer was in CR mode, i.e. it was not modified. Actually it was modified but the most recent copy is not in the buffer cache of that instance; it's in instance 1. So the instance recovery process will not have to consider the buffer at all. On the other hand, if instance 1 crashes, the instance recovery process will not need to even consider anything in any buffer cache; rather it will merely get the relevant redo entries from the online redo log and apply them to the blocks in the database. This is due to the fact that the CR copy of the buffer is identical to the block on the disk and does not need to be flushed to the disk.

This is true, but with a very important caveat. Consider Fig 3 once again. What if the buffer was transferred from instance 2 to instance 1 before it was flushed to the disk? In that case the CR buffer in instance 2 will not be the same as in the disk. Its SCN is 30 while that of the block on the disk is 10. If instance 1 crashes, the block on the disk must undergo instance recovery. However, a more recent copy of the block is available right there – on instance 2 – with SCN 20. Applying archived logs from SCN 10 to 30 is definitely worse compared to SCNs 10 through 20; so it would make a lot of sense to flush the buffer in instance 2 to disk and recover it. But, unfortunately, it will not happen since the state of the buffer is CR, which is assumed to be in sync with the disk.

To avoid this quandary, Oracle does not mark this buffer in instance 2 as CR; but rather as **Past Image** (PI), which is a special state. It means that the block has been modified (and therefore is not CR) but modified again in another instance (therefore not Exclusive Current). If recovery is required, this buffer can be used rather than applying all archived logs to the block on the disk. The buffer remains in the PI state until it is flushed to the disk by a checkpoint or manually. After the flush the buffer copy is identical to the block on the disk and therefore is not considered during recovery – so it is converted to a regular CR copy.

One important point to note: the term Past Image is not documented in Oracle Manuals. It's just widely understood and acceptable.

Cache Fusion Processes

The part of the RAC infrastructure that transfers buffer from one instance to the other is called **Global Cache Service** (GCS).

GCS makes sure buffers are served up as appropriate; but it does not know who has what type of buffer lock exists on the buffer. That information is provided by another component - **Global Enqueue Service** (GES), which used to be called Dynamic lock Manager (DLM) in the previous versions of the database. GES holds the information on the locks on the buffers. These locks are exposed through the view – V$LOCK_ELEMENT (or X$LE). If a buffer is locked, the lock element name is shown in LOCK_ELEMENT column of the view V$BH. You can examine the state of all the buffers of an object and the locks on them by issuing:

```
select file#, block#,
      decode(class#,1,'data block',2,'sort block',3,'save undo block', 4,
'segment header',5,'save undo header',6,'free list',7,'extent map',
8,'1st level bmb',9,'2nd level bmb',10,'3rd level bmb', 11,'bitmap block',
12,'bitmap index block',13,'file header block',14,'unused',
15,'system undo header',16,'system undo block', 17,'undo header',
18,'undo block') class_type, status, LOCK_ELEMENT_ADDR
from v$bh
where objd = <DataObjectId of the Object>
order by 1,2,3
```

Lock Queuing

To summarize, when an instance wants to change the state of the buffer from CR to Exclusive Current, or vice versa, it must get a lock on that buffer. This is called a Buffer Lock and it is different from the row lock, which is on a specific row. When a process wants to acquire the buffer in an Exclusive Current state, it must get the XCUR lock on the buffer. If the buffer was already in CR mode, this additional restrictive locking is known as **Lock Upgrade**. Similarly when an instance wants to upgrade its own buffer to a higher lock, which means the buffer on the other instance must be converted to a CR copy, the lock on the remote instance is converted down from XCUR – a process known as **Lock Downgrade**. The lock is never said to go away; it simply becomes a NULL lock. As long as the buffer is there, there is a lock on the buffer. In case of a CR mode, the lock mode is NULL.

Consider the implication of the above statement. In a normal course of the database operation there will be a lot of requests to upgrade and downgrade locks on a specific buffer. To make sure the locking process is smooth, there has to be a queue. Each block in a RAC instance has two queues:

- Grant Queue – the queue where the requesters are queued for the locks to be granted in a certain mode
- Convert Queue – the queue where the granted requests are queued to be notified to the requesters

When a process requests a lock to be downgraded or upgraded on a buffer, it must put a request in the Grant Queue, shown in Fig 4. When the request is filled, the process is moved to the convert queue to let it know that its request has been satisfied. These queues are memory structures. To replicate them on all instances will be time consuming and will affect the locking process. So, Oracle does not replicate the queues to all instances. The queues for a particular block are kept in only one instance. This instance is known as the **Master Instance** of the block. A block has only one master instance and each instance may hold queues of many blocks. When any instance requests a lock on a specific buffer, it contacts the LMS process of the master instance via its own LMS process. When the request is fulfilled, it also gets notified through the LMS process. The master instance of a buffer is not fixed; it may change. If an instance requests a buffer lock several times but is not the master, Oracle may make it the master for that block – a process known as **Dynamic Resource Mastering**. It can be manually remastered as well.

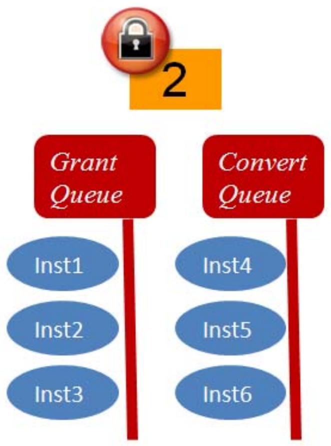

Figure 4 Buffer Lock Queues

When an instance wants to get a lock, it has to check with the master. You may be wondering how the instance knows where the master is located. Think of this as a phone book. When you want to use someone's office for a private meeting, you call the person to find out if anyone is already using it. To call, you need to know the phone number – which is in the phone book. The phone book is replicated and sent to everyone. The same analogy applies here – the phone book is akin to a directory of all the resources and is very aptly called **Global Resource Directory** (GRD). The GRD is replicated across all instances and holds the information on which is the master instance of a specific block; not who holds the lock. Whoever wants to get the lock first queries the GRD to get the master instance and then sends a message to the master for the lock request.

You can also find out the master information of a specific block. To find out the master of all the blocks of a specific object, first find out the Data Object ID of the object and then issue the following query:

```
select  b.dbablk, r.kjblmaster master_node
from x$le l, x$kjbl r, x$bh b
where b.obj = <DataObjectId>
and b.le_addr = l.le_addr
and l.le_kjbl = r.kjbllockp
```

When a block is remastered, the master instance information is updated and replicated across all the instances. While the replication is going on, GRD should not be queried. Therefore when the master instance changes, the entire GRD is frozen until the updated information appears in all instances.

Putting it all Together

Buffers are gotten in 2 modes: CURRENT – when it is intended to be modified and CR – when it is selected only for reading; not modifying. Every time a node wants the buffer from a remote node, the buffer is copied to a new buffer and then sent, a process known as CR processing. There can be only one current state of the buffer in an instance in Shared Mode when the buffers on all instances are identical. If one node modifies the buffer, it must acquire the buffer in Exclusive Current mode. RAC enforces the different modes by locking the buffers in an appropriate mode, e.g. Exclusive Current means locked by XCUR lock. When an instance requests a block to modify, it must check for its master in the Global Resource Directory. Then it sends a message to the master to acquire the lock. The request goes to the queue and is fulfilled in the order it was received.

Now that you understand the mechanics of the cache fusion process you should be able to find out where the potential issues could come up. If the message processing system is slow, which typically means LMS is not getting enough CPU cycles, the requestor would obviously wait, negatively affecting the overall performance.

You should have also realized that that for every block on the disk, there could be more than one copy in the buffer cache – at most one Shared Current, one Past Image and many CR copies. So, there will be many buffers of a specific block in the buffer cache. By defining only 100MB of buffer cache per instance for a 100MB database will not be enough. Since the CR copies might proliferate flooding the buffer cache and denying other blocks, there is a limit of 6 CR copies per block. When the seventh CR copy is necessary, Oracle ages out the oldest CR copy to make room for the new one.

Let's examine the original questions and their answers. Table 1 shows that in a summary format. I hope you liked the article and found it useful in making your understanding of RAC a little better.

Question	Answer
A block may have multiple copies on different instances, How does Oracle make sure that the correct copy of a block is updated?	By placing a Exclusive Current buffer lock on the block
When another instance wants to update the same buffer, what happens to the original buffer? Is it written to the disk?	No; the lock on the original buffer is downgraded to NULL, the buffer is converted to CR, sent over to the other node where an Exclusive Current Lock is placed on that buffer making it an Exclusive Current.
Where is the buffer's lock information stored?	In only one place in memory, the buffer's master instance. The requests for locks are placed in a grant queue and when they are filled, the request is moved to a convert queue.
How does an instance know a block's master	By querying a data structure called Global

instance?	Resource Directory.

About the Author

Arup Nanda (arup@proligence.com) has been an Oracle DBA for last 16 years working on many aspects of database management from modeling to disaster recovery and has been working on RAC for last 11 years. He has co-authored 4 books, written about 300 articles and delivered about 100 sessions on Oracle technology. He was the winner of DBA of the Year Award in 2003, is an Oracle ACE Director and a member of the Oak Table Network. He also maintains a blog: arup.blogspot.com.