

Cache Buffer Chains Demystified

Arup Nanda

Longtime Oracle DBA

Agenda

```
SQL> select state, event from v$session where sid = 123;
```

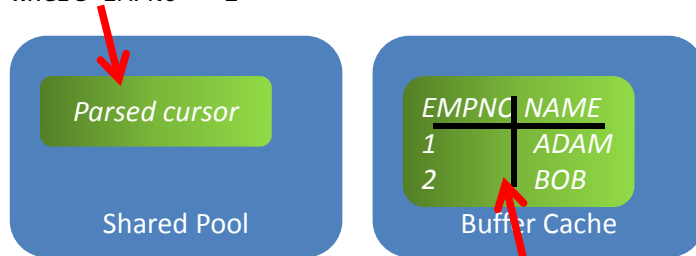
STATE	EVENT
WAITING	latch: cache buffers chains

- How buffer cache works
- How buffers are populated
- Buffer states and versioning
- How buffers are flushed
- Role of Cache Buffer Chain latch
- Reducing CBC Latches
- Other kinds of latches

Database and Instance

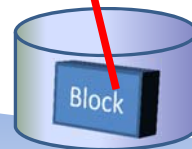
```
update EMP  
set NAME = 'ROB'  
where EMPNO = 1
```

True or False: Buffer = Block?



Host

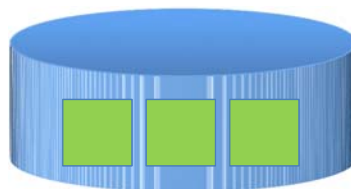
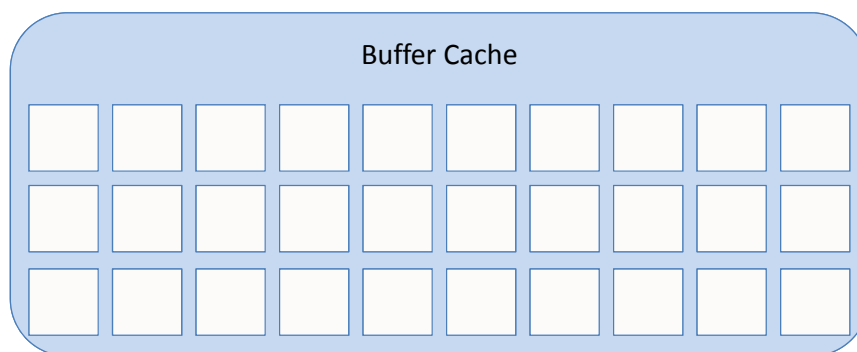
Storage



Arup Nanda

3

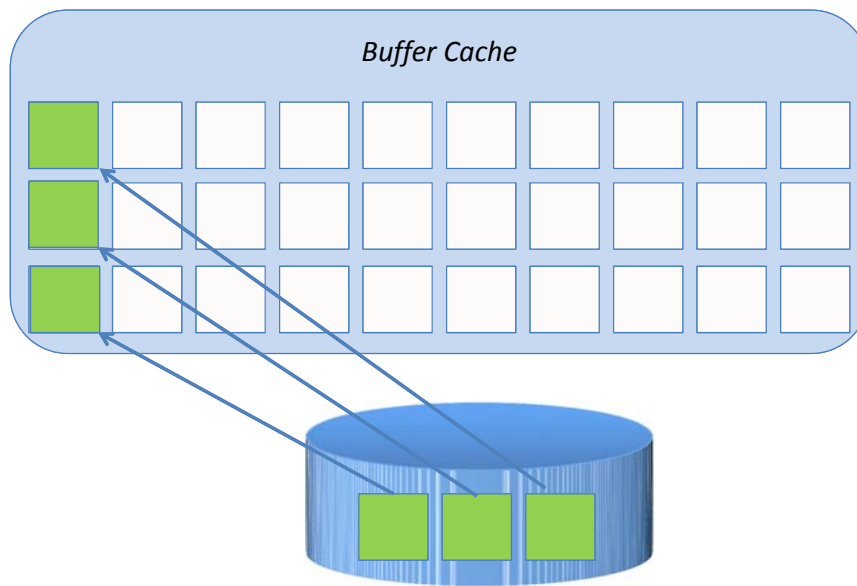
Buffer and Block



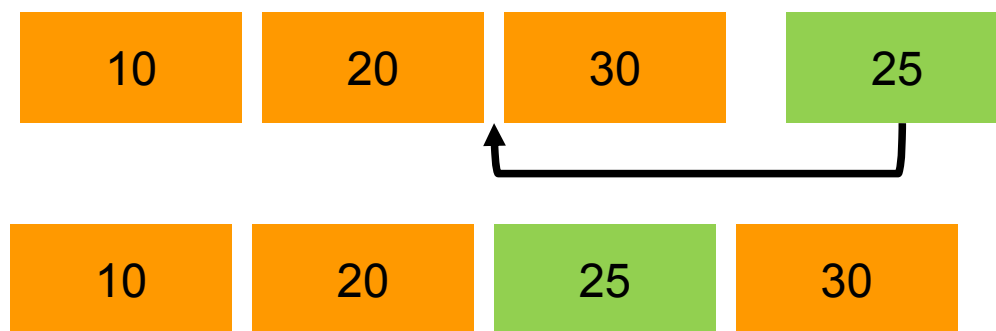
Arup Nanda

4

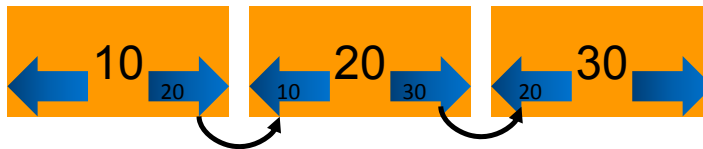
Buffer and Block



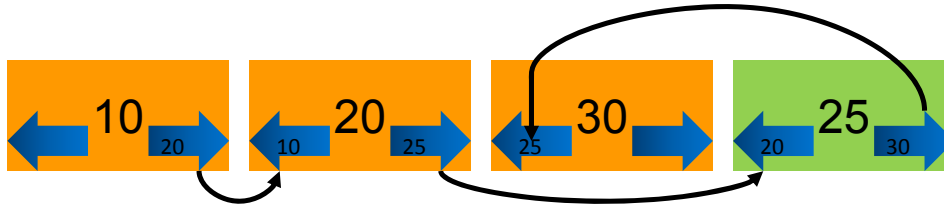
Buffer Insertion



Buffer Management



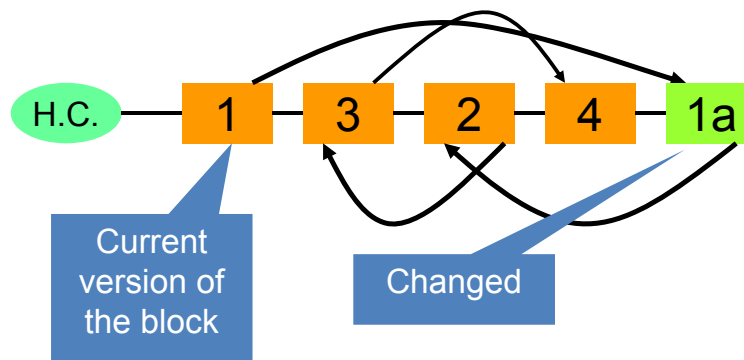
Each buffer has a prior and a next pointer that shows the next buffer in this list. This is known as a linked list.



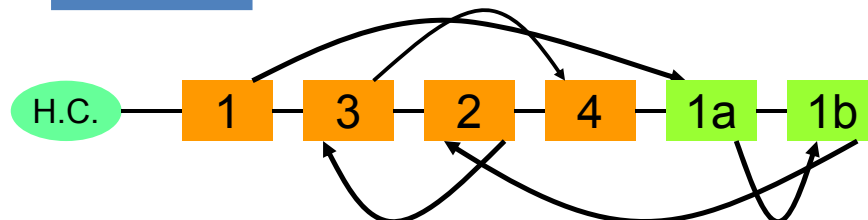
When a new buffer comes in, only the pointers are updated

Multi-versioning

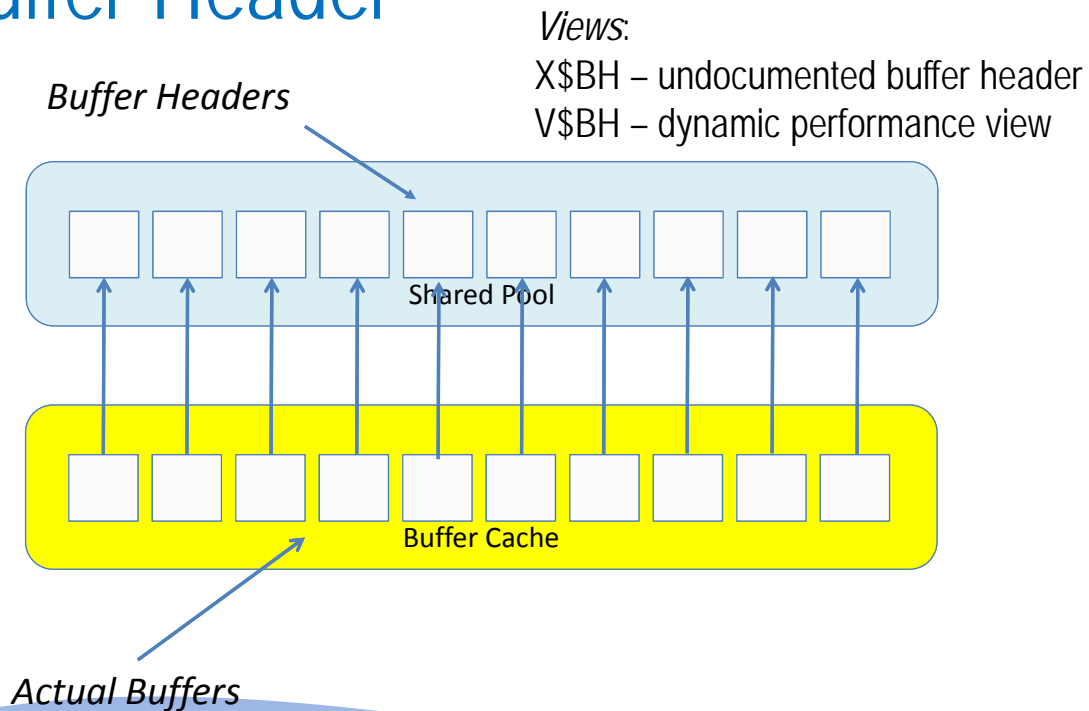
Buffer 1
was
updated



Buffer 1
was
updated
once more



Buffer Header



Arup Nanda

9

Buffer Handles in Shared Pool

- Find out the space used by buffer handles in shared pool.

```
select * from v$sgastat
where pool = 'shared pool'
and name = 'buffer handles';
```

POOL	NAME	BYTES
shared pool	buffer handles	1440008

buffhan.sql

Arup Nanda

Partitioning Tips and Tricks

10

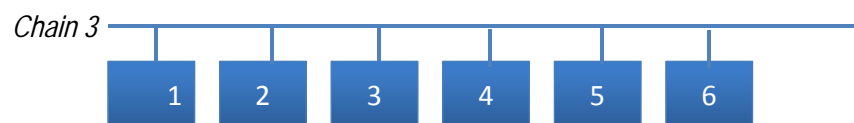
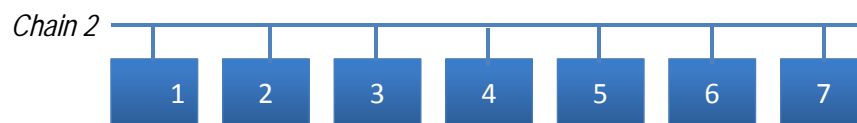
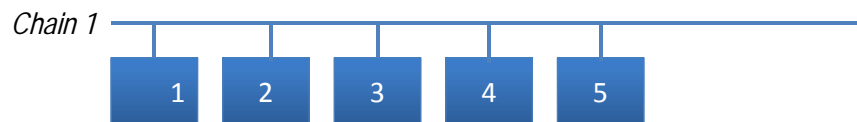
Buffer Chains

Hash
buckets



Number of hash buckets is determined by
`_db_block_hash_buckets`

Walking the Chain



Data Block Address

- Get the relative file# and block#

```
select col1,  
       dbms_rowid.rowid_relative_fno(rowid) rfile#,  
       dbms_rowid.rowid_block_number(rowid) block#  
from cbctest;
```

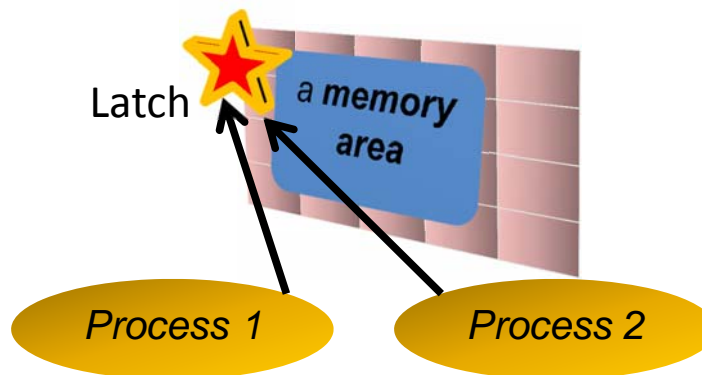
fblk.sql

- Get the DBA

```
select dbms_utility.make_data_block_address  
(file#,block#) from dual;
```

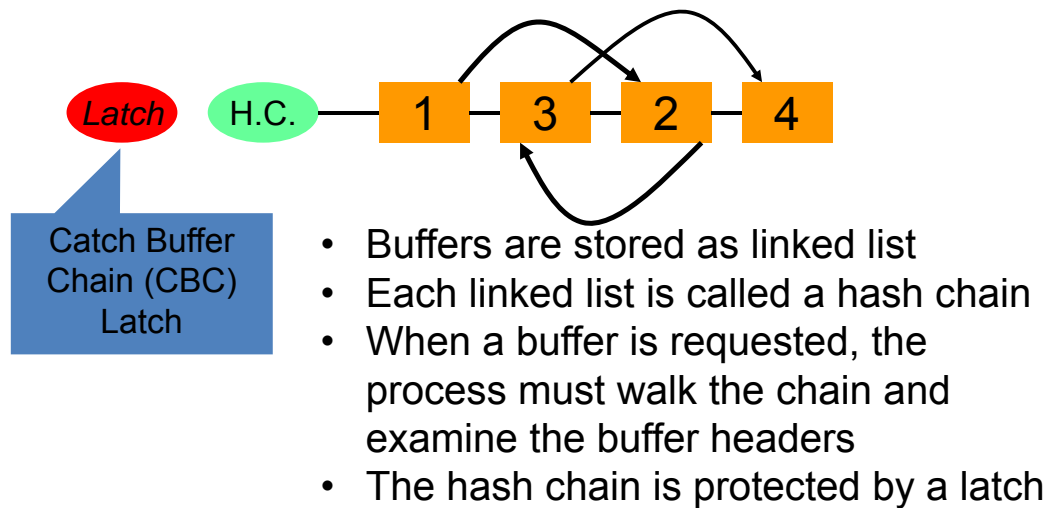
get_dba.sql

Latches

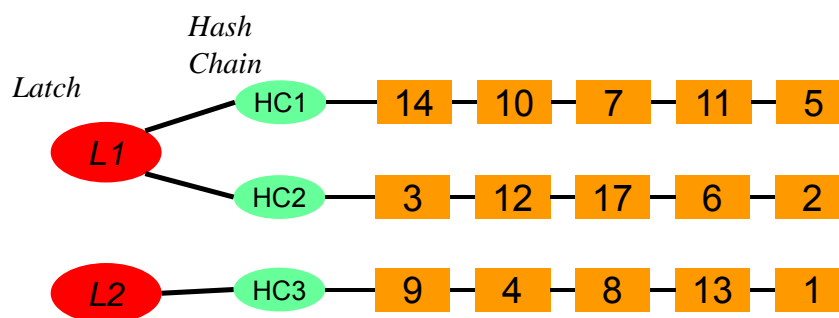


- Process 1 and 2 will try to get the “latch”, a area in memory that does not have any required data.
- Whoever gets the latch now gets to access the memory area exclusively
- When done, the process releases the latch

Buffer Cache



Latches and Hash Chains



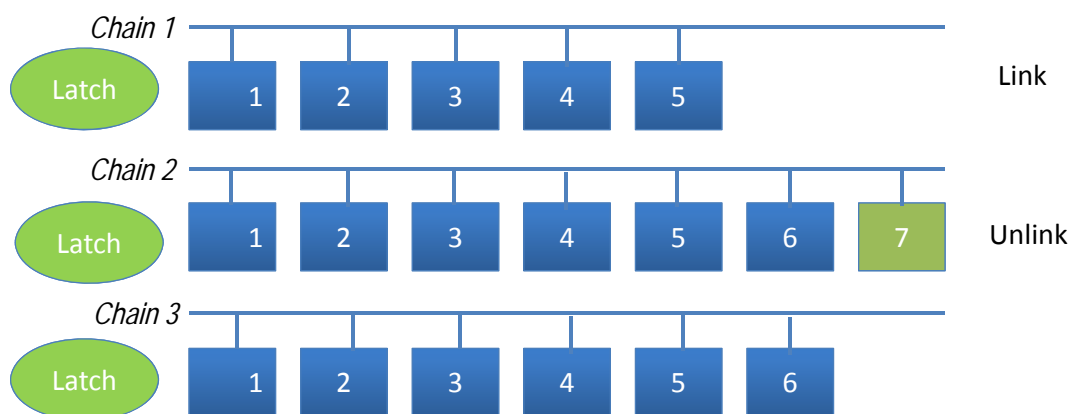
No. of hash buckets = init.ora parameter `_db_block_hash_buckets`
No. of latches = `_db_block_hash_latches`

hladdr.sql

More about Latches

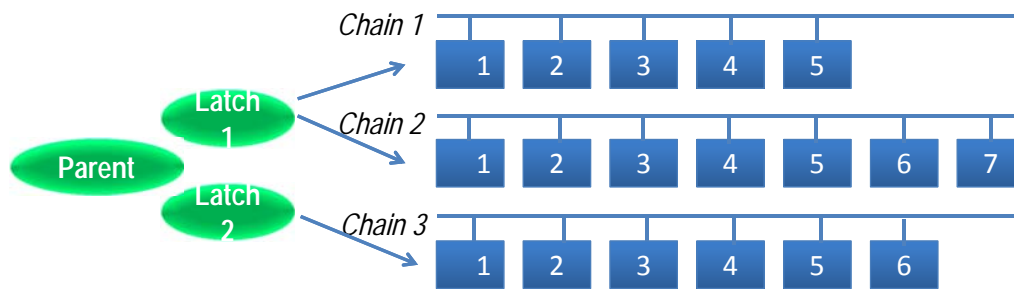
- Suppose process 1 gets the latch, accesses the memory
- How will process 2 know when the latch is available?
 - No central latch repository
 - No communication to the process
- Process 2 will constantly loop to check if the latch is free
- This is called **spinning** – a CPU intensive process
- After n times, it will stop spinning and will go to sleep
 - $n = \texttt{_spin_count}$ in init.ora, defaults to 2000
- After that it will wake up after 1 ms, check, go to sleep
- Check again in 1ms, sleep, then check in 2 ms, sleep ...

Linking/Unlinking



Cache Buffer Chain Latch
`_db_block_hash_latches`

Parent and Child CBC Latch



- To find the latch protecting the hash chain
select hladdr
from x\$bh where dbarfil = 6 and dbablk =
220;

CBC Latches

- Find the Latch
select latch# from v\$latch
where name = 'cache buffers chains';
203
- This is the parent latch
- Child Latches
SQL> select count(1) from v\$latch_children
where latch# = 203;
16384
- Checking the hash buckets and CBC latches:
_db_block_hash_buckets 524288
_db_block_hash_latches 16384

Diagnosis of CBC Latch Waits

- Check the P1, P2, P3 values

```
select p1, p1raw, p1text
from v$session where sid = 366;
```

- Result

P1	P1RAW	P1TEXT
5553027696	000000014AFC7A70	address

- Get the latch information:

```
SQL> select gets, misses, sleeps, name
       2  from v$latch where addr = '000000014AFC7A70';
```

GETS	MISSES	SLEEPS	NAME
49081	14	10	cache buffers chains

- This confirms that this is a CBC Latch

Identify the Object

- Get the File# and Block#

```
select dbarfil, dbablk, tch, obj
from x$bh
where hladdr = '000000014AFC7A70';
DBARFIL DBABLK TCH      OBJ
-----
        6      220 34523 93587
```

- Dump the block

```
alter system dump
datafile 6 block min 220 block max 220;
```

- Get the object

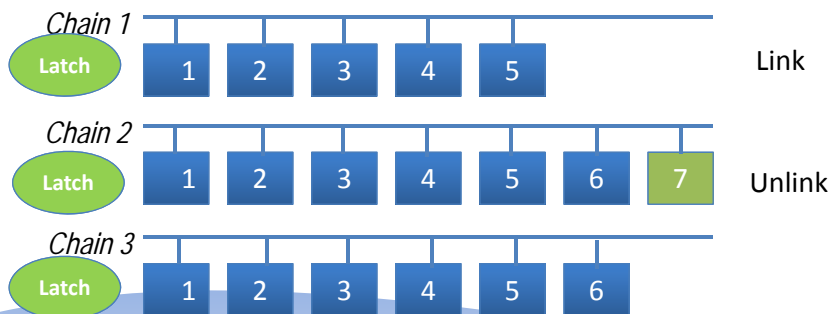
```
SQL> select object_name
       2  from dba_objects
       3  where object_id = 93587;
```

OBJECT_NAME
CBCTEST

Start dump data blocks tsn: 4 file#:6 minblk 220 maxblk 220
Block dump from cache:
Dump of buffer cache at level 4 for pdb=0 tsn=4 rdba=25166044
BH (0x7ff72f6b918) file#: 6 rdba: 0x018000dc (6/220) class: 1
ba: 0x7ff7212a000
set: 12 pool: 3 hsz: 8192 bsi: 0 sflg: 0 pwc: 39,28
dbwrid: 0 obj: 93587 objn: 93587 tsn: [0/4] afn: 6 hint: f

Reducing CBC Latches

- Reduce Logical IO
- Avoid Nested Loops
 - NLs visit the same blocks several times



App Visiting Same Data

- Example

```
for i in 1..100000 loop
  select ...
  into l_var
  from tablea
  where ...;
  exit when sql%notfound;
end loop;
```
- Workaround
 - Collections
 - Bulk Collect into Collections

Identify the Hot Objects

- Find from ASH

```
select p1raw,event,count(*)
from v$active_session_history
where sample_time < sysdate - 1/24
and event = 'latch: cache buffers chain'
group by event, p1raw
order by 3 desc;
```
- Find the object

```
select o.name, bh.dbarfil, bh.dbablk, bh.tch
from x$bh bh, sys.obj$ o
where tch > 0
and hladdr='<p1raw value>'
and o.obj#=bh.obj
order by tch;
```

Reducing Hotness

- Reduce Logical IO
- Reduce Nested Loops
- Reduce Contention
 - Unpack blocks
 - Partition
 - If index, hash partition global index

Summary

- Buffer cache starts with empty buffers to be filled with blocks
- Buffers are not moved around; the headers are updated
- Buffer headers are linked to a hash chain and unlinked
- To link/unlink a buffer to/from a chain, a latch is needed
- This latch is called cache buffer chain latch
- X\$BH and V\$BH shows the buffer headers
- X\$BH.HLADDR shows the latch address
- V\$LATCH shows the latch, LATCH# is the identifier
- V\$LATCH_CHILDREN shows the child latches
- A smaller number of latches protect all the buffers
- When the latch is not available, the session waits
- The more popular the buffer the more the CBC Latch wait



Thank You!

Blog: arup.blogspot.com

Tweeter: [@arupnanda](https://twitter.com/arupnanda)

Facebook.com/ArupKNanda