ORACLE Technology NETWORK

효율적인 오라클 데이타베이스 성능 관리 세미나 I

ORACLE

세미나 일정

시간	내용
10:00-11:30	Memory part I & II
11:30-13:00	점심 식사
13:00-14:00	Memory part III
14:10-15:00	I/O Part I
15:10-16:00	I/O Part II

ORACLE

효율적인
오라클 데이타베이스
성능 관리 세미나 I
- Memory Part -

백선덕
DB기술팀
한국오라클

ORACLE

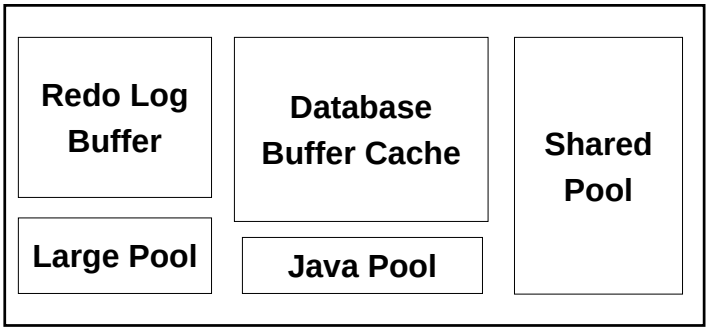
Contents

- Shared pool
- Buffer cache
- Other SGA structures

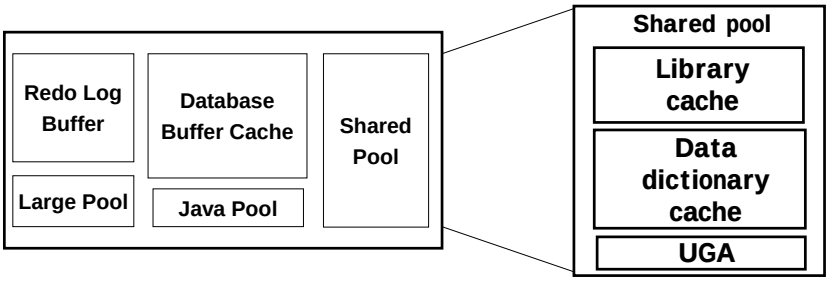
ORACLE

- Shared pool
- Buffer cache
- Other SGA structures

System Global Area (SGA)



Shared Pool



- **SHARED_POOL_SIZE**

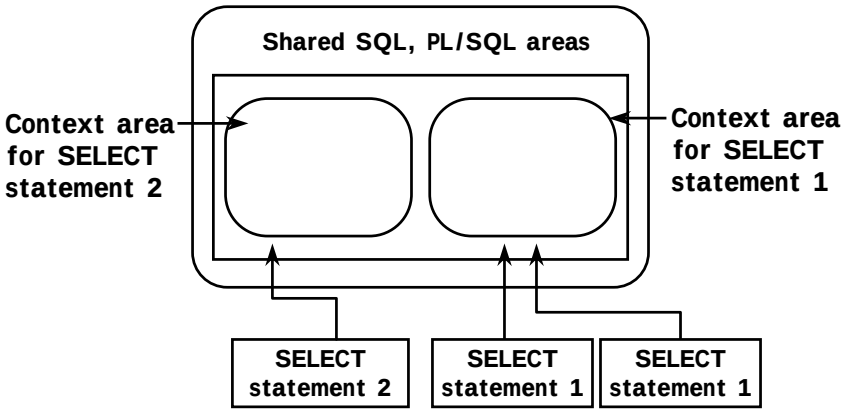
Shared Pool

- Library Cache
- Dictionary Cache
- User Global Area (UGA)
- Large Pool

Library Cache

- Store SQL statements, PL/SQL blocks
  - ➢ to be shared
- LRU (least recently used) algorithm
- Prevent statements reparsing

Library Cache



Library Cache Tuning

Reduce misses
> keep parsing to a minimum:

- Share statements
- Prevent statements from being aged out
- Avoid invalidations

Library Cache Tuning

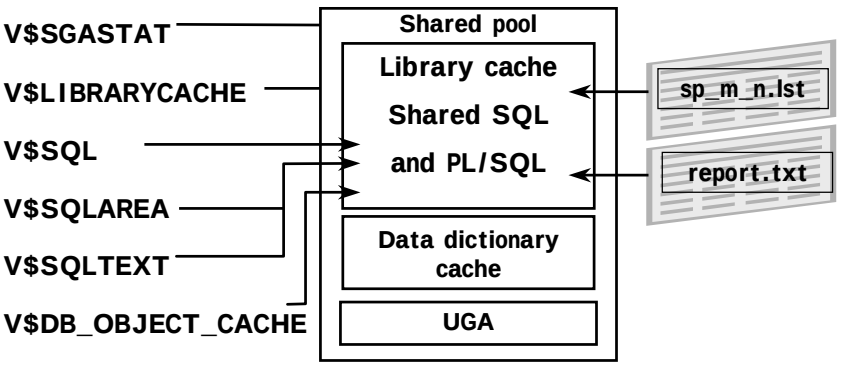
Avoid fragmentation:

- Reserve space for large mem. reqs.
- Pin frequently required large objects
- Eliminate large anonymous PL/SQL blocks
- Use large pool for Oracle Shared Server cons.

Terminology

- **Gets**: (Parse)
  - # of lookups for objects of the namespace
- **Pins**: (Execution)
  - # of reads or executions of the objects of the namespace
- **Reloads**: (Reparse)
  - # of library cache misses on the execution step, causing implicit reparsing of the statement and block

Diagnostic Tools for Tuning the Library Cache



Parameters:
SHARED_POOL_SIZE, OPEN_CURSORS
SESSION_CACHED_CURSORS, CURSOR_SPACE_FOR_TIME

Cursors

A cursor is a handle (a name or pointer) for the memory associated with a specific statement.

Cursors Shared?

- **V$LIBRARYCACHE.GETHITRATIO:**

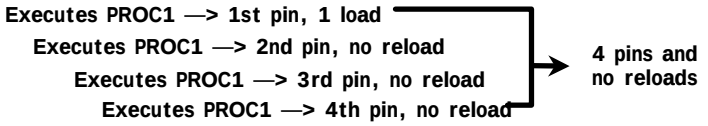
```
SQL> select gethitratio
  2  from v$librarycache
  3  where namespace = 'SQL AREA';
```

- Statements running:

```
SQL> select sql_text, users_executing,
 2          executions, loads
 3   from v$sqlarea;
```

```
SQL> select * from v$sqltext
 2   where sql_text like
 3   'select * from hr.employees where %';
```

Guidelines: Library Cache Reloads



- Reloads < 0.01 * the pins:

```
SQL> select sum(pins) "Executions",
  2      sum(reloads) "Cache Misses",
  3      sum(reloads)/sum(pins)
  4  from v$librarycache;
```

- If reloads-to-pins ratio > 1%, increase (SHARED_POOL_SIZE).

Library Cache Guidelines: STATSPACK Report

```
Library Cache Activity for DB: ORA92  Instance: ora92  Snaps: 1 -2
->"Pct Misses"  should be very low

                        Get    Pct            Pin    Pct             Invali-
Namespace           Requests   Miss       Requests   Miss    Reloads  dations
--------------- ------------ ------ -------------- ------ ---------- --------
BODY                      19   26.3             20   35.0          0        0
CLUSTER                   51    2.0             47    4.3          0        0
SQL AREA                 544    9.4          2,649    4.8          0        0
TABLE/PROCEDURE          695    3.9            790   14.1          1        0
```

Invalidations

- # of times objects (of the namespace) marked invalid, causing reloads:

```
SQL> select count(*) from hr.employees;

SQL> select namespace,pins,reloads,invalidations
  2  from v$librarycache;

Namespace            Pins Reloads  Invalidations
--------------- --------- -------  -------------
SQL AREA            47772      14              0
TABLE/PROCEDURE      1551       0              0
BODY                   12       0              0
...

SQL> ANALYZE TABLE hr.employees COMPUTE STATISTICS;
```


Invalidations

```
SQL> select count(*) from hr.employees;

SQL> select namespace,pins,reloads,invalidations
  2  from v$librarycache;

Namespace            Pins Reloads  Invalidations
--------------- --------- -------  -------------
SQL AREA            48045      15             11
TABLE/PROCEDURE      1596       0              0
BODY                   12       0              0
...
```

Sizing Library Cache

- Global space for stored objs. (packages, views, etc.)
- Amount of memory used by the usual SQL stmts.
- Space for large memory reqs.
  - to avoid misses and fragmentation
- Frequently used objects
- Large anonymous PL/SQL blocks into small anonymous blocks calling packaged functions

Cached Execution Plans

- Oracle server preserves the actual execution plan of a cached SQL statement in memory.
- When the SQL statement ages out, the corresponding cached execution plan is removed.
- Benefit : better diagnosis of query performance.

V$SQL_PLAN for Cached Execution Plans

- **V$SQL_PLAN**
  - Actual execution plan info. for cached cursors.
- **PLAN_TABLE** cols.(w/o LEVEL col.) + extra cols.

V$SQL for Cached Execution Plans

- **PLAN_HASH_VALUE** column
  - hash value built from the corresponding execution plan.
  - to compare cursor plans
  - the same way the **HASH_VALUE** column for comparing cursor SQL texts.

Example: V$SQL_PLAN

```
SQL> SELECT p.id, p.parent_id,
  2         lpad(' ',p.depth*4)||p.operation "OPERATION",
  3         p.options, p.object_owner "OWNER",
  4         p.object_name "OBJECT"
  5  FROM v$sql s, v$sql_plan p
  6  WHERE s.hash_value = p.hash_value
  7  AND s.sql_text like 'select count(*) from hr.employees%'
  8  ORDER BY p.id;


 ID PARENT_ID  OPERATION        OPTIONS   OWNER OBJECT
--- ---------  ---------------- --------- ----- ------------
  0            SELECT STATEMENT
  1         0      SORT         AGGREGATE
  2         1           INDEX    FULL SCAN HR    EMP_EMAIL_UK
```


Global Space Allocation

- Stored objects such as packages and views:

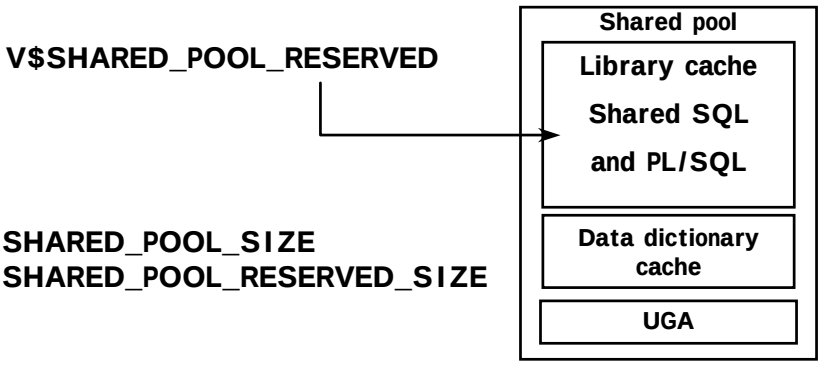
```
SQL> SELECT sum(sharable_mem)
 2   FROM v$db_object_cache
 3   WHERE type in ('PACKAGE','PACKAGE BODY','VIEW');
SUM(SHARABLE_MEM)
-----------------
           746429
```

- SQL statements:

```
SQL> select sum(sharable_mem)
 2   from V$SQLAREA where executions > 5;
SUM(SHARABLE_MEM)
-----------------
          1718776
```

Large Memory Requirements

- Satisfy requests for large contiguous memory
- Reserve contiguous memory within the shared pool



Tuning the Shared Pool Reserved Space

- Diagnostic tools for tuning:
  - **V$SHARED_POOL_RESERVED**
  - Supplied package and procedure:
    - **DBMS_SHARED_POOL.**

      ABORTED_REQUEST_THRESHOLD
- Guidelines
  - Set the parameter
    - **SHARED_POOL_RESERVED_SIZE**

Keeping Large Objects

- Find those PL/SQL objects that are not kept in the library cache:

```
SQL> SELECT owner, name, type FROM v$db_object_cache
  2  WHERE sharable_mem > 10000
  3  AND (type='PACKAGE' or type='PACKAGE BODY' or
  4       type='FUNCTION' or type='PROCEDURE')
  5  AND  KEPT='NO';

OWNER NAME         TYPE
----- --------     ------------
SYS   STANDARD     PACAKGE
SYS   STANDARD     PACAKGE BODY
SYS   DBMS_UTILITY PACAKGE BODY
...
```


Keeping Large Objects

- Pin large packages in the library cache:

```
SQL> EXECUTE dbms_shared_pool.keep('package_name');
```

```
SQL> EXECUTE dbms_shared_pool.keep('SYS.STANDARD');
```

```
SQL> SELECT owner, name, type FROM v$db_object_cache
  2  WHERE sharable_mem > 10000
  3  AND (type='PACKAGE' or type='PACKAGE BODY' or
  4       type='FUNCTION' or type='PROCEDURE')
  5  AND  KEPT='NO';

OWNER NAME         TYPE
----- --------     ------------
SYS   DBMS_UTILITY PACAKGE BODY
...
```

Anonymous PL/SQL Blocks

Find the anonymous PL/SQL blocks and convert them into small anonymous PL/SQL blocks that call packaged functions:

```
SQL> select sql_text from v$sqlarea
  2  where command_type = 47
  3  and length(sql_text) > 400;
```

```
command_type desription
------------ -------------
           2 insert
           3 select
           6 update
          47 pl/sql execute
         ... ...
```


Other Parameters Affecting the Library Cache

- **OPEN_CURSORS**
- **CURSOR_SPACE_FOR_TIME**
- **SESSION_CACHED_CURSORS**
- **CURSOR_SHARING**

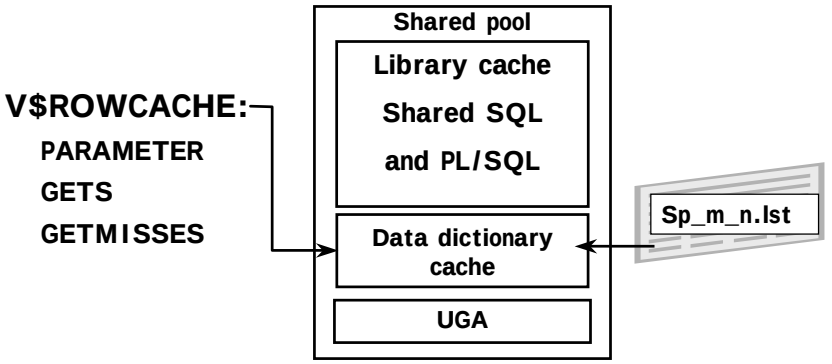
Shared Pool

- Library Cache
- Dictionary Cache
- User Global Area (UGA)
- Large Pool

Content, Terminology, and Tuning

- Content: Definitions of dictionary objects
- Terminology:
  - **GETS**: # of requests on objects
  - **GETMISSES**: # of requests resulting in cache misses
- Goal: Avoid dictionary cache misses

Diagnostic Tools for Data Dictionary Cache



Measuring the Dictionary Cache Statistics

Dictionary Cache Stats section of STATSPACK:

- Percent misses should be very low:
  - < 2% for most data dictionary objects
  - < 15% for the entire data dictionary cache
- Cache Usage : # of cache entries being used.
- Pct SGA : the ratio of usage to allocated size for that cache.

Tuning the Data Dictionary Cache

SUM(GETMISSES)*100 / SUM(GETS) < 15%

```
SQL> select parameter, gets, getmisses
 2   from v$rowcache;
PARAMETER                      GETS GETMISSES
------------------------- --------- ---------
dc_objects                   143434       171
dc_synonyms                  140432       127
...
```


Guidelines: Dictionary Cache Misses

STATSPACK report output:

```
NAME            GET_REQS GET_MISS

--------------- -------- --------

dc_objects         143434       171
dc_synonyms        140432       127
...
```

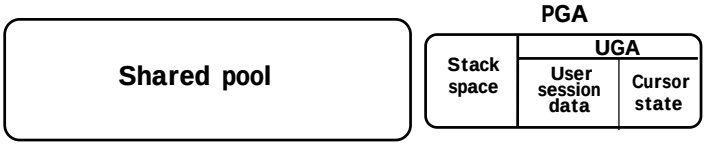
If too many cache misses, increase **SHARED_POOL_SIZE**.

Shared Pool

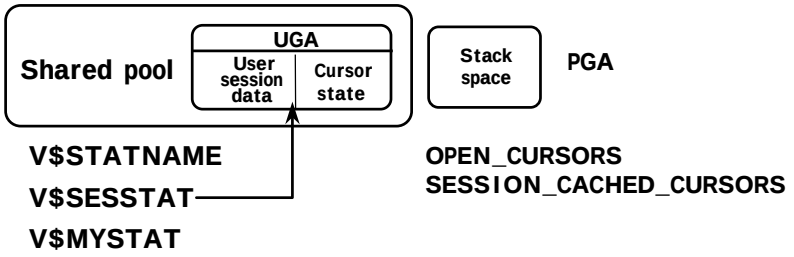
- Library Cache
- Dictionary Cache
- User Global Area (UGA)
- Large Pool

UGA and Oracle Shared Server

- Dedicated server connection:



- Oracle Shared Server connection: no large pool



Sizing the User Global Area

- UGA space used by your connection:

```
SQL> select SUM(value) ||'bytes' "Total session memory"
  2  from V$MYSTAT, V$STATNAME
  3  where name = 'session uga memory'
  4  and v$mystat.statistic# = v$statname.statistic#;
```

- UGA space used by all Oracle Shared Server users:

```
SQL> select SUM(value) ||'bytes' "Total session memory"
  2  from V$SESSTAT, V$STATNAME
  3  where name = 'session uga memory'
  4  and v$sesstat.statistic# = v$statname.statistic#;
```

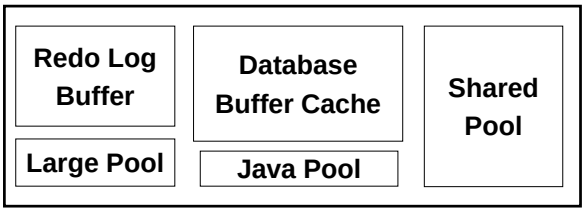
- Maximum UGA space used by all users:

```
SQL> select SUM(value) ||'bytes' "Total max memory"
  2  from V$SESSTAT, V$STATNAME
  3  where name = 'session uga memory max'
  4  and v$sesstat.statistic# = v$statname.statistic#;
```


Shared Pool

- Library Cache
- Dictionary Cache
- User Global Area (UGA)
- Large Pool

Large Pool



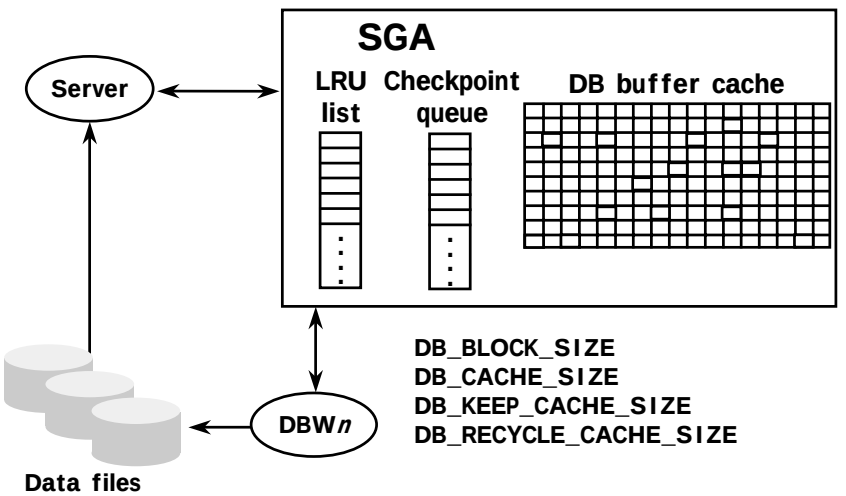
- A separate memory area in the SGA, used for memory with:
  - **DBWR_IO_SLAVES**
  - Backup and restore operations
  - Session memory
  - Parallel query messaging
- Useful in these situations to avoid performance overhead caused by shrinking the shared SQL cache
- Parameter : **LARGE_POOL_SIZE**

- Shared Pool
- Buffer Cache
- Other SGA Structures

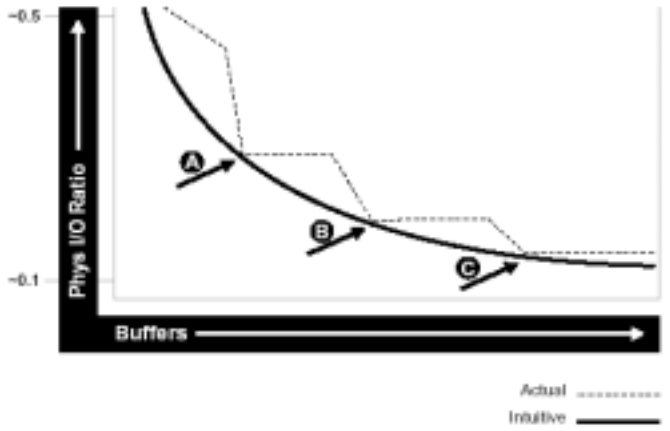
Buffer Cache

- Overview of buffer cache
- Dynamic SGA allocation
- **DB_CACHE_ADVICE**
- Multiple buffer pools
- Free list contention

Overview



Buffer Cache



Buffer Cache Parameters (Oracle9*i*)

- Independent subcaches
- Multiple block sizes.
- **DB_BLOCK_SIZE** parameter : primary block size (for the **SYSTEM** tbs)
- Sizes of the caches for buffers:
  - **DB_CACHE_SIZE**
  - **DB_KEEP_CACHE_SIZE**
  - **DB_RECYCLE_CACHE_SIZE**

Buffer Cache

- Overview of buffer cache
- Dynamic SGA allocation
- **DB_CACHE_ADVICE**
- Multiple buffer pools
- Free list contention

Dynamic SGA Feature in Oracle9*i*

- Allowing the server to change its SGA configuration w/o shutting down the instance.
- With a dynamic SGA, the Oracle server can modify its physical address space use to respond to the operating system's use of physical memory.
- A dynamic SGA provides an SGA that will grow and shrink in response to a DBA command.

Granule (Oracle9i)

- SGA memory is tracked in granules.
- Unit of contiguous virtual memory allocation.
- **V$BUFFER_POOL**
- size of a granule :
  - 4 MB if estimated_size(SGA) < 128 MB
  - 16 MB otherwise

Allocating Granules at Startup

- At instance startup, the Oracle server allocates granule entries, one for each granule to support **SGA_MAX_SIZE** bytes of address space.
- As startup continues, each component acquires as many granules as it requires.
- Minimum SGA configuration is 3 granules:
  - 1 granule for fixed SGA (includes redo buffers)
  - 1 granule for the buffer cache
  - 1 granule for the shared pool

Adding Granules to Components

- A DBA can dynamically increase memory allocation to a component by issuing an **ALTER SYSTEM** command.
- Increase of the memory use of a component succeeds only if there are enough free granules to satisfy the request.
- Memory granules are not freed automatically from another component in order to satisfy the increase.
- Decrease of a component is possible, but is successful only if the corresponding number of granules remain unused by the component.

Dynamic Buffer Cache Size Parameters

- Parameters that specify the size of buffer cache components are dynamic, and can be changed while the instance is running by means of the **ALTER SYSTEM**:

```
SQL> ALTER SYSTEM SET DB_CACHE_SIZE = 1100M;
```

- Each parameter is sized independently.
- New cache sizes are set to the next granule boundary.
- limitations:
  - Integer multiple of the granule size.
  - The total SGA size cannot exceed **MAX_SGA_SIZE**.
  - **DB_CACHE_SIZE** can never be set to zero.

Example: Increasing the Size of an SGA Component

- Initial parameter values:
  - **SGA_MAX_SIZE = 128M**
  - **DB_CACHE_SIZE = 88M**
  - **SHARED_POOL_SIZE = 32M**

```
SQL> ALTER SYSTEM SET SHARED_POOL_SIZE = 64M;
```

  - Error message indicating insufficient memory

```
SQL> ALTER SYSTEM SET DB_CACHE_SIZE = 56M;
SQL> ALTER SYSTEM SET SHARED_POOL_SIZE = 64M;
```

  - Error message indicating insufficient memory
  - Check **V$BUFFER_POOL** to see shrinking completed

```
SQL> ALTER SYSTEM SET SHARED_POOL_SIZE = 64M;
```

  - The statement is now processed.

Deprecated Buffer Cache Parameters

- 3 parameters:
  - backward compatibility
  - **DB_BLOCK_BUFFERS**
  - **BUFFER_POOL_KEEP**
  - **BUFFER_POOL_RECYCLE**
- Not combined with the dynamic size params.
  - ORA-00381: cannot use both new and old parameters for buffer cache size specification

Buffer Cache

- Overview of buffer cache
- Dynamic SGA allocation
- **DB_CACHE_ADVICE**
- Multiple buffer pools
- Free list contention

Dynamic Buffer Cache Advisory Parameter

- Enable and disable statistics gathering
  - for predicting behavior with different cache sizes
- Help DBAs size the buffer cache
- **DB_CACHE_ADVICE**:

```
SQL> ALTER SYSTEM SET DB_CACHE_ADVICE = OFF|ON|READY;
```

V$DB_CACHE_ADVICE

- Predicting the estimated number of physical reads for different cache sizes.
- The rows also compute a physical read factor, which is the ratio of

  (# of estimated reads) /

  (# of reads actually performed during the measurement interval by the real buffer cache).

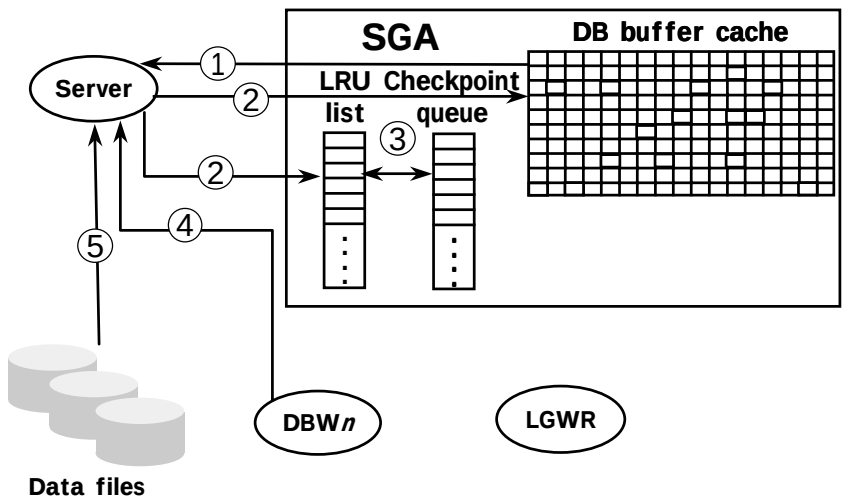
Example: V$DB_CACHE_ADVICE

```
SQL> SELECT size_for_estimate,
  2         buffers_for_estimate,
  3         estd_physical_read_factor,
  4         estd_physical_reads
  5  FROM V$DB_CACHE_ADVICE
  6  WHERE name = 'DEFAULT‘
  7  AND block_size = (
  8        SELECT value FROM V$PARAMETER
  9        WHERE name = 'db_block_size')
  10       AND advice_status = 'ON';
```

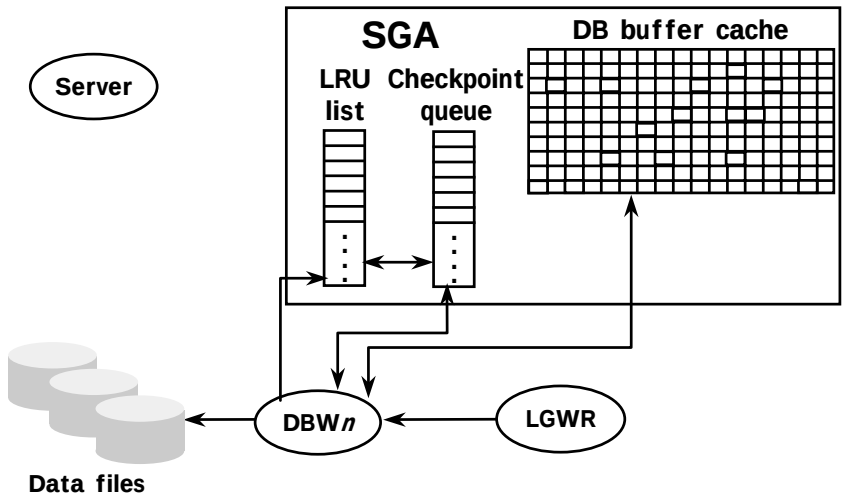
Example: V$DB_CACHE_ADVICE

Cache Size (MB)	Buffers	Estd Phys Read Factor	Estd Phys Reads	
30	3,802	18.70	182,317,943	10% of Current Size
60	7,604	12.83	131,949,536	
91	11,406	7.38	75,965,861	
121	15,208	4.97	51,111,658	
152	19,010	3.64	37,460,786	
182	22,812	2.50	25,669,196	
212	26,614	1.74	17,850,847	
243	30,416	1.33	13,720,149	
273	34,218	1.13	11,583,180	
304	38,020	1.00	10,282,475	Current Size
334	41,822	.93	9,515,878	
364	45,624	.87	8,909,026	
395	49,426	.83	8,495,039	
424	53,228	.79	8,116,496	
456	57,030	.76	7,824,764	
486	60,832	.74	7,563,180	
517	64,634	.71	7,311,729	
547	68,436	.69	7,104,280	
577	72,238	.67	6,895,122	
608	76,040	.66	6,739,731	200% of Current Size

Server Process & Buffer Cache



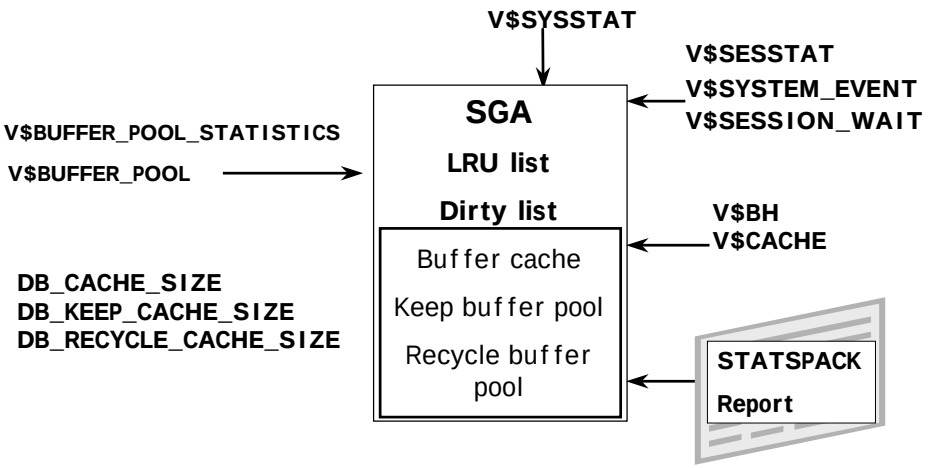
DBWn Process & Buffer Cache



Tuning Goals and Techniques

- Goals:
  - Servers find data in memory
  - 90% hit ratio for OLTP
- Diagnostic measures
  - Cache hit ratio
  - **V$DB_CACHE_ADVICE**
- Tuning techniques:
  - Reduce # of blocks required by SQL statements
  - Increase buffer cache size
  - Use multiple buffer pools
  - Cache tables
  - Bypass the buffer cache for sorting and parallel reads

Diagnostic Tools



Measuring the Cache Hit Ratio

From **V$SYSSTAT:**

```
SQL> SELECT 1 - (phy.value – lob.value – dir.value)
   / ses.value  "CACHE HIT RATIO"
  2  FROM   v$sysstat ses, v$sysstat lob,
  3         v$sysstat dir, v$sysstat phy
  3  WHERE  ses.name = 'session logical reads'
  4  AND    dir.name = ‘physical reads direct'
  5  AND    lob.name = 'physical reads direct (lob)'
  6  AND    phy.name = 'physical reads';
```

From the **STATSPACK** report:

Statistic	Total	Per Second	Per Trans
physical reads	7,666	3.0	294.9
physical reads direct	7,229	2.9	278.0
Physical reads direct(lob)	0	0	0
session logical reads	59,234	23.5	2,278.2

Guidelines for Using the Cache Hit Ratio

Hit ratio is affected by data access methods:

- Full table scans
- Data or application design
- Large table with random access
- Uneven distribution of cache hits

Guidelines to Increase the Cache Size

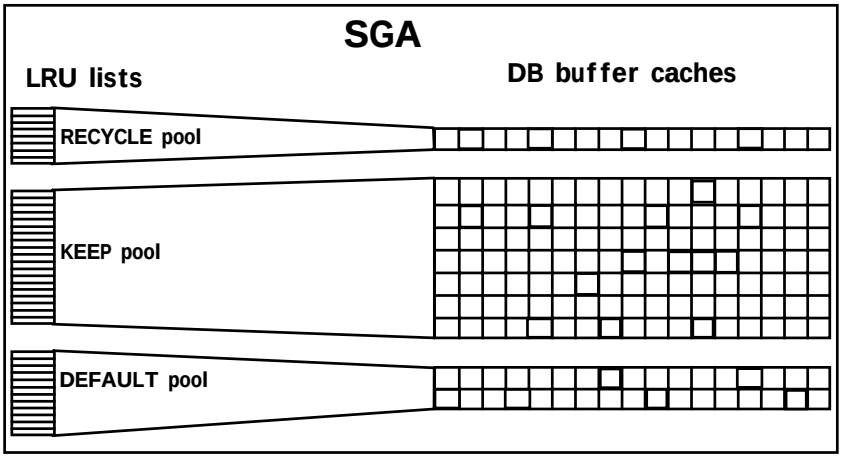
Increase the cache size when:

- Cache hit ratio is less than 0.9 on OLTP system.
- There is no undue page faulting.
- If the previous increase was effective.

Buffer Cache

- Overview of buffer cache
- Dynamic SGA allocation
- **DB_CACHE_ADVICE**
- Multiple buffer pools
- Free list contention

Using Multiple Buffer Pools



Defining Multiple Buffer Pools

- In Oracle9*i*,
  - Individual pools have their own size defined by **DB_CACHE_SIZE, DB_KEEP_CACHE_SIZE, DB_RECYCLE_CACHE_SIZE.**
  - These parameters are dynamic.
  - Latches are automatically allocated by Oracle RDBMS.
- In Oracle8*i*,
  - Pool blocks : **DB_BLOCK_BUFFERS.**
  - Latches are taken from **DB_BLOCK_LRU_LATCHES.**
  - At least 50 blocks per latch.

Enabling Multiple Buffer Pools

```
CREATE INDEX cust_idx …
  STORAGE (BUFFER_POOL KEEP);

ALTER TABLE customer
  STORAGE (BUFFER_POOL RECYCLE);

ALTER INDEX cust_name_idx
  STORAGE (BUFFER_POOL KEEP);
```

KEEP Buffer Pool Guidelines

- Goal: Keeping blocks in memory
- Size: Holds all or nearly all blocks
- Sizing tool: **`ANALYZE ... ESTIMATE STATISTICS`**

```
SQL> ANALYZE TABLE hr.countries ESTIMATE STATISTICS;

SQL> SELECT  table_name, blocks
  2    FROM  dba_tables
  3    WHERE owner = 'HR'
  4    AND table_name = 'COUNTRIES';

TABLE_NAME     BLOCKS
---------- ----------
COUNTRIES       14
```


RECYCLE Buffer Pool Guidelines

- Goal: Eliminating blocks from memory when transactions are completed
- Size: Holds only active blocks (1/4 ?)
- Tool: **`V$CACHE (catclust.sql)`**

```
SQL> SELECT  owner#, name, count(*) blocks
  2  FROM  v$cache
  3  GROUP BY owner#, name;

OWNER# NAME           BLOCKS
------ ---------- ----------
     5 CUSTOMER          147
     5 REGIONS            20
...
```

- Sum(blocks targeted for RECYCLE) / 4

RECYCLE Buffer Pool Guidelines

Tool: **V$SESS_IO : I/O stat by session**

```
SQL> SELECT s.username,
  2         io.block_gets,
  3         io.consistent_gets,
  4         io.physical_reads
  5  FROM   v$sess_io io, v$session s
  6  WHERE  io.sid  = s.sid ;

USERNAME BLOCK_GETS CONSISTENT_GETS PHYSICAL_READS
-------- ---------- -------------- --------------
HR             2187          23271           1344
```


Calculating the Hit Ratio for Multiple Pools

```
SQL> SELECT name,
     1 - (physical_reads / (db_block_gets + consistent_gets))
     "HIT_RATIO"
 2   FROM   sys.v$buffer_pool_statistics
 3   WHERE  db_block_gets + consistent_gets > 0;

NAME                 HIT_RATIO
-----------------   ----------
KEEP                .983520845
RECYCLE             .503866235
DEFAULT             .790350047
```

Identifying Candidate Pool Segments

- **KEEP** Pool
  - Blocks are accessed repeatedly.
  - Segment size is less than 10% of the DEFAULT buffer pool size.
- **RECYCLE** Pool
  - Blocks are not reused outside of transaction.
  - Segment size is more than twice the DEFAULT buffer pool size.

Views with Buffer Pools

```
SQL> SELECT  id, name, block_size, buffers
  2  FROM  v$buffer_pool;

ID NAME    BLOCK_SIZE BUFFER
-- ------- ---------- ------
 1 KEEP          4096  14000
 2 RECYCLE       4096   2000
 3 DEFAULT       4096   4000
```

Caching Tables

- LRU end
- Enable caching during full table scans by:
  - Creating the table with the **CACHE** clause
  - Altering the table with the **CACHE** clause
  - **CACHE** hint in a query
- Guideline: Do not overcrowd the cache.
- Use a KEEP pool

Other Cache Performance Indicators

- **From V$SYSSTAT**
  - **free buffer inspected : # of buffers skipped on LRU list**
  - **high or increasing**

```
SQL> SELECT name, value
  2  FROM   v$sysstat
  3  WHERE  name = 'free buffer inspected';

NAME                           VALUE
-------------------------- --------
free buffer inspected             183
```

Other Cache Performance Indicators

- From V$SYSTEM_EVENT
  - free buffer waits :
  - buffer busy waits :

```
SQL> SELECT event, total_waits
  2  FROM   v$system_event
  3  WHERE  event in
  4         ('free buffer waits', 'buffer busy waits');

EVENT                  TOTAL_WAITS
---------------------- -----------
free buffer waits              337
buffer busy waits             3466
```

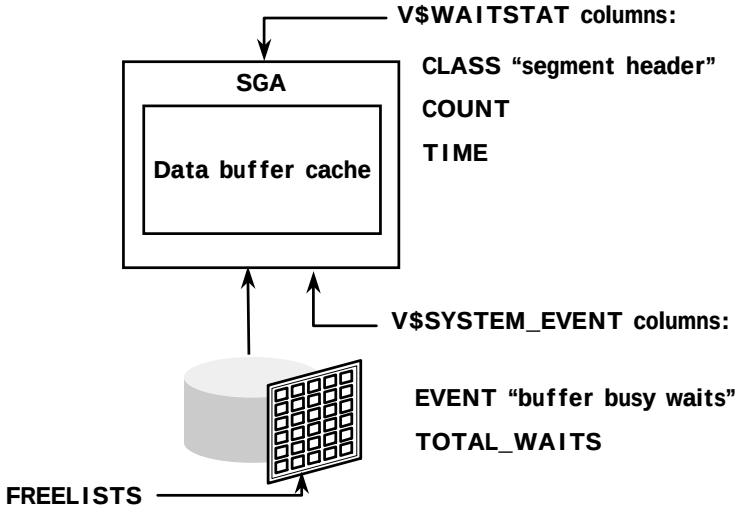

Buffer Cache

- Overview of buffer cache
- Dynamic SGA allocation
- **DB_CACHE_ADVICE**
- Multiple buffer pools
- Free list contention

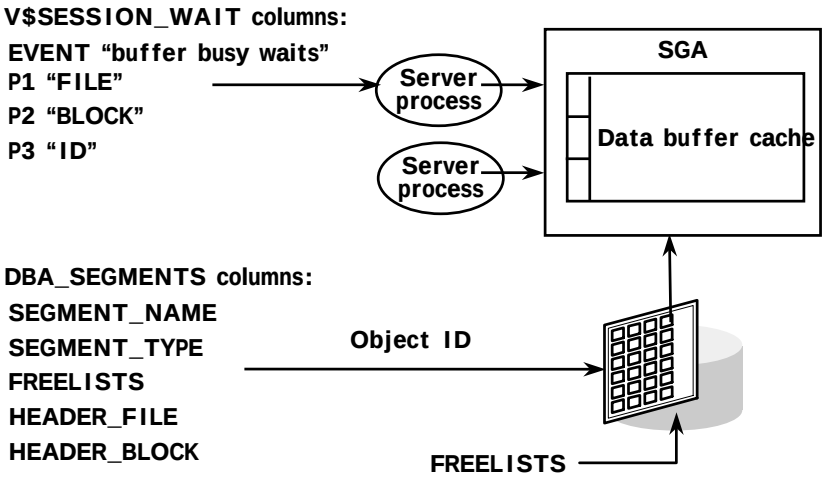
Free Lists

- A free list for an object maintains a list of blocks that are available for inserts.
- The number of free lists for an object can be set dynamically.
- Single-CPU systems do not benefit greatly from multiple free lists.
- The tuning goal is to ensure that an object has sufficient free lists to minimize contention.
- Using Automatic Free Space Management eliminates the need for free lists, thus reducing contention on the database.

Diagnosing Free List Contention



Diagnosing Free List Contention



Resolving Free List Contention

- Query **V$SESSION_WAIT**.
- Identify the object and get free lists for the segment from **DBA_SEGMENTS**.

```
SQL> SELECT s.segment_name, s.segment_type, s.freelists,
  2         w.wait_time, w.seconds_in_wait, w.state
  3  FROM   dba_segments s, v$session_wait w
  4  WHERE  w.event = 'buffer busy wait'
  5  AND    w.p1 = s.header_file
  6  AND    w.p2 = s.header_block;
```

- Either:
  - ALTER TABLE ... FREELISTS, or
  - Move into an auto-managed tablespace.

Auto-management of Free Space

- Create an auto-managed tablespace:

```
SQL> CREATE TABLESPACE BIT_SEG_TS
  2  DATAFILE '$HOME/ORADATA/u04/bit_seg01.dbf' SIZE 1M
  3  EXTENT MANAGEMENT LOCAL
  4  SEGMENT SPACE MANAGEMENT AUTO;
```

- Create a table that uses auto-management of free space:

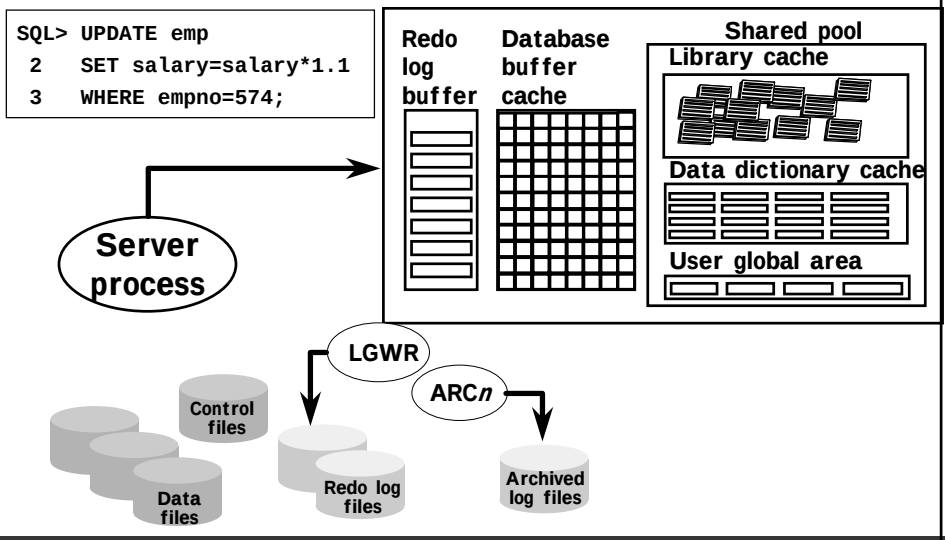
```
SQL> CREATE TABLE BIT_SEG_TABLE
  2  (IDNUM NUMBER)
  3  TABLESPACE BIT_SEG_TS;
```


- Shared Pool
- Buffer Cache
- Other SGA Structures

Other SGA Structures

- Redo log buffer
- Java pool
- Java session memory used by a session
- Configure the instance to use I/O slaves
- Configure and use multiple DBW processors

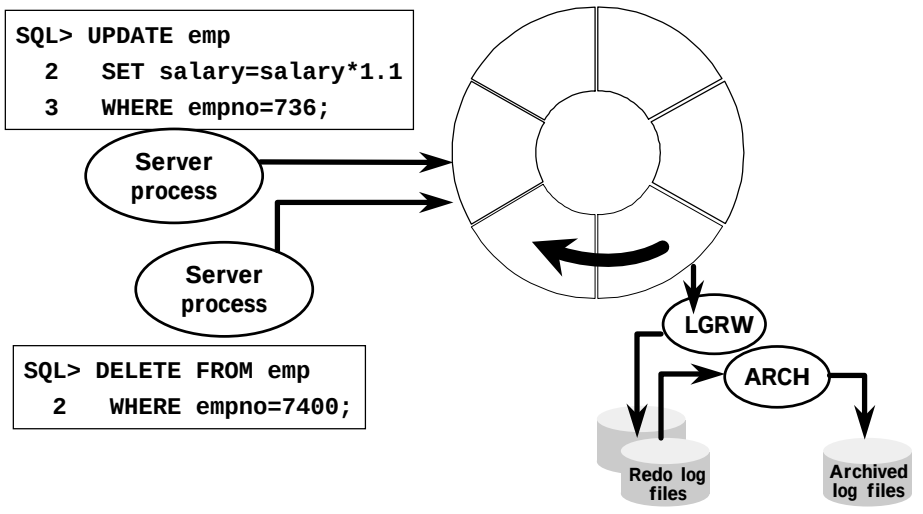
Redo Log Buffer



Sizing the Redo Log Buffer

- **LOG_BUFFER** parameter
- Default value: MAX (512K, 128K * COUNT(CPU))

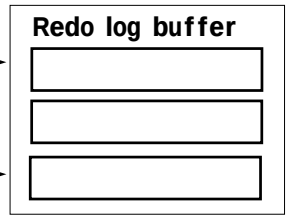
Diagnosing Redo Log Buffer Inefficiency



Analyze Redo Log Buffer Efficiency

V$SESSION_WAIT
Log Buffer Space event

V$SYSSTAT
Redo Buffer Allocation Retries statistic



Redo Log Buffer Tuning Guidelines

- There should be no Log Buffer Space waits.

```
SQL> SELECT sid, event, seconds_in_wait, state
  2  FROM   v$session_wait
  3  WHERE  event = 'log buffer space';

  SID EVENT            SECONDS_IN_WAIT STATE
----- --------------- -------------- -------
   12 log buffer space             112 WAITING
```

Redo Log Buffer Tuning Guidelines

- Redo Buffer Allocation Retries value should be near 0, and should be less than 1% of redo entries.

```
SQL> SELECT r.value "Retries", e.value "Entries",
  2         r.value/e.value * 100 "Percentage"
  3  FROM v$sysstat r, v$sysstat e
  4  WHERE r.name = 'redo buffer allocation retries'
  5  AND e.name = 'redo entries';

   Retries    Entries Percentage
---------- ---------- ----------
         0        431          0
```


Reducing Redo Operations

- Direct Path loading w/o archiving
- Direct Path loading w/ archiving can use **`Nologging`** mode.
- Direct Load Insert can use **`Nologging`** mode.
- Some SQL statements can use **`Nologging`** mode.
  - **`CREATE TABLE ... AS SELECT`**
  - **`CREATE INDEX`**
  - **`ALTER INDEX ... REBUILD`**

Monitoring Java Pool Memory

```
SQL> SELECT * FROM v$sgastat
  2  WHERE pool = 'java pool';
POOL        NAME                            BYTES
----------- -------------------------- ----------
java pool   free memory                  30261248
java pool   memory in use                19742720
```

Limiting Java session memory

- **JAVA_SOFT_SESSIONSPACE_LIMIT**
  - **warning**
- **JAVA_MAX_SESSIONSPACE_SIZE**
  - **ORA-29554: unhandled Java out of memory ...**
  - **session killed**

Sizing the SGA For Java

- **SHARED_POOL_SIZE:**
  - 8 KB per loaded class
  - 50 MB for loading large **JAR** files
- **JAVA_POOL_SIZE**
  - 20 MB default
  - 50 MB for medium-sized Java application

Multiple I/O Slaves

- Provide nonblocking asynchronous I/O requests
- Deployed by the **DBW0** process
- Typically not recommended if asynchronous I/O is available
- Follow the naming convention **ora_i*nnn*_SID**
- Turn asynchronous I/O on or off with:
  - **DISK_ASYNCH_IO : TRUE/FALSE**
  - **TAPE_ASYNCH_IO : TRUE/FALSE**

Multiple DBWR Processes

- Multiple **DBW*n*** processes can be deployed with **DB_WRITER_PROCESSES (DBW0 to DBW9).**
- Useful for SMP systems with large numbers of CPUs.
- Multiple processes cannot concurrently be used with multiple I/O slaves.

Tuning DBW*n* I/O

- Tune the DB Writer processes by looking at the value of the **FREE BUFFER WAITS** event

```
SQL> SELECT total_waits
  2  FROM v$system_event
  3  WHERE event = 'free buffer waits';
```

  - **Consider increasing DBWn if high**