RULES OF RAC DIAGNOSTICS

By Riyaj Shamsudeen

This paper explains optimization techniques in a RAC database. While most practices employed in a single database is applicable to a RAC database also, there are a few key differences that require special attention. I will also review a few differences in the 12c version of the database.

Golden rules of RAC diagnostics

Here are the top 5 rules of performance diagnostics in a RAC database, in my opinion, anyway.

1. Beware of top event tunnel vision. Search for the symptoms of root causes mentioned in the following section.
2. Eliminate infrastructure as an issue.
3. Identify problem-inducing instance(s).
4. Review the Send side metrics, and not just the Receive side metrics.
5. Use histograms to identify any abnormalities, and not just averages. This is self-explaining rule.

Things that can go bad!

To understand various performance issues encountered, let's look at the root causes of performance issues (somewhat) unique to a RAC database.

1. CPU / Memory starvation in a node[1]. Suppose if there is memory starvation in a node, then global cache receive time in other nodes will increase sharply, as LMS processes can suffer from memory starvation issues. Further, CPU/memory starvation in the receiving side also can affect the receiving side global cache performance metrics.

2. As we discussed earlier, LMS process may need wait for LGWR to complete a log flush before sending the block. Slowness in LGWR performance will increase the global cache receive time in the receiving nodes.

3. Packet loss due to a network issue can increase the global cache receive time in the receiving nodes.

[1] Effect of CPU starvation to an LMS process is minimal. LMS processes usually operate in RT priority.

4. SQL statements requesting huge number of blocks due to inefficient execution plan or code, causing global cache workload increase.

5. Localized inserts in to a few blocks.

6. Localized, excessive access to a few blocks.

Top Event Tunnel vision

Usually, in a single instance database, you would focus on top wait events and ignore other wait events while analyzing the performance issues. However, in a RAC database, you should not ignore a few wait events even if the direct impact of those wait events are minimal. Side effect impacts of those wait events can result in higher receive time in the other receiving nodes. Let's review few examples:

Example #1:

The following AWR snippet shows top 5 wait events. Traditionally, you would focus on the top wait events. However, the root cause in this case is a DRM event, which is the fifth event in the list. The direct effect of DRM freeze is 3.6% only. However, the side effect of excessive DRM activity is gc buffer busy event. Secondary side effect of gc buffer busy wait is Locking contention.

Top 5 Timed Events

Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time	Wait Class
gc buffer busy	949,868	278,050	293	53.2	Cluster
enq: TX - index contention	191,747	91,470	477	17.5	Concurrenc
buffer busy waits	108,313	67,074	619	12.8	Concurrenc
CPU time		22,217		4.3	
gcs drm freeze in enter server	38,686	19,029	492	3.6	Other

Example #2:

Review the following AWR snippet. You would tend to focus on locking contention, as that is the top consumer of time. However, locking contention in this case was a side effect of gc buffer busy wait events. Root cause was that second node suffered from memory starvation due to a shared memory segment not cleaned out properly after an instance crash. That memory starvation caused LMS process to suffer from excessive swapping and paging, leading to gc buffer busy wait events in other nodes.

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
enq: TX - index contention	6,050	320	53	20.6	Concurrenc
gc buffer busy release	36,983	276	7	17.9	Cluster
gc buffer busy acquire	12,479	227	18	14.7	Cluster
buffer busy waits	16,614	159	10	10.3	Concurrenc
DB CPU		126		8.1	

Example #3:

In the following AWR output, gc buffer busy is the higher waited event. However, notice the log file sync wait event, with an average of 74ms. That is the root cause event and I/O performance to the online redo log files caused LMS to stuck waiting for gc log flush. That caused all other side effects.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
gc buffer busy	222,666	48,168	216	32.5	Cluster
gc cr block busy	68,514	25,191	368	17.0	Cluster
CPU time		14,781		10.0	
gc current block busy	17,463	9,735	557	6.6	Cluster
log file sync	92,021	6,845	74	4.6	Commit

Eliminate infrastructure as an issue

While analyzing a performance issue in a RAC database, it is beneficial to understand if the network, CPU or memory infrastructure is an issue or not. Wouldn't be nice to have one event that summarizes the infrastructure performance? Fortunately, "gc cr block 2-way" wait event comes handy.

Wait events - gc cr block 2-way, gc cr block 3-way, gc current block 2-way, and gc current block 3-way - indicate that those block transfers did not suffer from any additional RAC processing delays. After receiving a request for the block, LMS process sends the block with out any additional global cache processing delays if the block is in a compatible state. If there were any additional processing tasks such as applying undo records, wait for log flush, etc, then the time is recorded to different events, such as - gc cr block busy - etc.

So, by reviewing the performance statistics of those four baseline wait events, you can identify if there are any issues in basic RAC infrastructure. Generally, you should see average wait time of <2ms for these events (In Exadata platform you should expect <0.5ms). Higher wait time for these wait events usually indicate that there is systemic issue.

Script in Appendix #1 can be used to identify the current average wait time for these events. AWR report also can be used to identify the average time for these events. In the following example, average wait time is below 1 ms indicating no performance issues.

```
   INST_ID EVENT                                AVERAGE_WAIT
---------- ---------------------------------- ------------
         1 gc cr block 2-way                           .04
         2 gc cr block 2-way                           .21
```

```
|----|------------------------------|-------|-----|----|
|Inst| Event                        |  Wt tm|Wait | %  |
|    |                              |     ms|count|    |
|----|------------------------------|-------|----|-----|
|   1|             gc cr block 2-way|      1| 810|   90|
|   1|             gc cr block 2-way|      2|  94|   10|
|   1|             gc cr block 2-way|      4|   0|    0|
|   1|             gc cr block 2-way|      8|   0|    0|
|   1|             gc cr block 2-way|     16|   0|    0|
|   1|             gc cr block 2-way|     32|   0|    0|
|   1|             gc cr block 2-way|     64|   0|    0|
|   1|             gc cr block 2-way|    128|   0|    0|
|   2|             gc cr block 2-way|      1| 764|   91|
|   2|             gc cr block 2-way|      2|  75|    9|
|   2|             gc cr block 2-way|      4|   0|    0|
|   2|             gc cr block 2-way|      8|   0|    0|
|   2|             gc cr block 2-way|     16|   0|    0|
|   2|             gc cr block 2-way|     32|   0|    0|
-------------------------------------------------------
```

There are few key points to understand:

1. These baseline wait events measure the performance in the receiving side. Still, these numbers doesn't indicate a root cause as, in most cases, the problem is in the sending side.
2. In 3+ node clusters, if one node has better RX(Receive) metrics, and all other nodes have worse RX metrics, then, there is a higher probability that problem is in that node with better RX metrics. Remember that node suffering from performance issue will affect the RX performance in other nodes. However, that performance-suffering node itself might enjoy better global cache RX time, as other nodes are working fine.
3. Reviewing of the Send metrics also important.

Identify problem-inducing Instance

The view gv$instance_cache_transfer can be used to identify the problem-inducing instance. This view externalizes the receive side performance, at instance and block class level. Script gc_identify_slow_inst.sql queries the gv$instance_cache_transfer view and can be used to identify the problem-inducing instance.

Refer to the following partial output of the script. From the output, it is clearly visible that instance 3 is misbehaving. All CR blocks received from that node have much higher CR receive time. You can also see that receive time of CR blocks received by instance 3 is within normal range.

(Note: Output heavily edited for brevity).

```
----------------------|-----------|-----------|-----------|-----------|…
```

Inst->Inst class	CR blk Tx	CR blk av	CR 2hop	CR 2hop av
1->2 data block	897	1.29	312	1.01
1->3 data block	945	1.53	323	0.98
...				
3->1 data block	1120	6.23	378	7.23
3->2 data block	765	7.23	289	8.23
...				
4->1 data block	465	1.01	112	1.31
4->2 data block	378	0.98	109	1.08
...				

Essentially, you can identify the problem instance with this approach. After identifying the probable problem node, you would review the send metrics in that problem node.

Review Send Side Metrics

Review of the Send-side metrics provides clues about the root cause of the slowness. Performance problems in the sending side will lead to performance issues in the receiving side.

CR block send time can be broken in to build time, send time, and flush time. By reviewing these metrics, you can identify the layer causing performance issue. In the following AWR snippet, Avg global cache cr block flush time is higher, implying the log file writes are not efficient enough.

Similarly, Current block send time is affected by pin time, send time, and flush time. In the following AWR snippet, you can see that flush time is causing performance issues in the receiving side.

```
Global Cache and Enqueue Services - Workload Characteristics
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

                    Avg global enqueue get time (ms):      0.1

          Avg global cache cr block receive time (ms):      0.3
      Avg global cache current block receive time (ms):      0.5

             Avg global cache cr block build time (ms):      0.0
              Avg global cache cr block send time (ms):      0.0
       Global cache log flushes for cr blocks served %:     12.8
             Avg global cache cr block flush time (ms):     40.5

        Avg global cache current block pin time (ms):      0.0
       Avg global cache current block send time (ms):      0.0
Global cache log flushes for current blocks served %:      0.2
     Avg global cache current block flush time (ms):     56.7
```

Appendix #3 can be used to break down the send side time during problem timeframe.

Summary

Version 12c introduces many new features in various areas such as Query optimization, Database performance tuning, High Availability, etc. However, there are not many performance features that are specific to RAC itself. I have provided a solid method and tools to debug performance issues in a RAC cluster with this paper.

Appendix #1

```
set serveroutput on size 100000
REM ------------------------------------------------------------------------------------
--------------
REM Author: Riyaj Shamsudeen @OraInternals, LLC
REM         www.orainternals.com
REM
REM Functionality: This script is to print GC processing timing for the past N seconds
or so
REM **************
REM
REM Source  : gv$sysstat
REM
REM Note : 1. Keep window 160 columns for better visibility.
REM
REM Exectution type: Execute from sqlplus or any other tool.  Modify sleep as needed.
Default is 60 seconds
REM
REM Parameters:
REM No implied or explicit warranty
REM
REM Please send me an email to rshamsud@orainternals.com for any question..
REM  NOTE   1. Querying gv$ tables when there is a GC performance issue is not exactly
nice. So, don't run this too often.
REM         2. Until 11g, gv statistics did not include PQ traffic.
REM         3. Of course, this does not tell any thing about root cause :-)
REM @copyright : OraInternals, LLC. www.orainternals.com
REM Version    Change
REM ---------- --------------------
REM ------------------------------------------------------------------------------------
--------------
set serveroutput on size 100000
undef event_name
undef sleep
PROMPT
PROMPT
PROMPT  gc_instance_cache.sql v1.10 by Riyaj Shamsudeen @orainternals.com
PROMPT
PROMPT  ...Prints various timing related information for the past N seconds
PROMPT  ...Default collection period is 60 seconds.... Please wait for at least 60
seconds...
PROMPT
PROMPT Column name key:
PROMPT   Inst -> Inst class :  source and target instance and class of the block
transfer
PROMPT   CR blk TX  :  CR blocks transmitted
PROMPT   CR blk tm  :  CR blocks time taken
PROMPT   CR blk av  :  Average time taken for CR block
PROMPT   CR bsy     :  Count of blocks suffered from "busy" events
PROMPT   CR bsy tm  :  Amount of time taken due to "busy" waits
PROMPT   CR bsy %   :  Percentage of CR busy time to CR time
PROMPT   CR congest :  Count of blocks suffered from "congestion" events
PROMPT   CR cngsttm :  Amount of time taken due to "congestion" waits
PROMPT   CR cng %   :  Percentage of CR congestion time to CR time
undef event_name
```

```
undef sleep
set lines 170 pages 100
set verify off

declare
        type t_number_table   is table of number       index by varchar2(32);
        type t_varchar2_table  is table of varchar2(60)       index by varchar2(32);
        type t_key_table  is table of varchar2(60)       index by binary_integer;

         key_tab                                  t_key_table;
        b_inst_id                                 t_number_table;
        b_event                                  t_varchar2_table;
        b_wait_time_milli                         t_number_table;
        b_wait_count                              t_number_table;
        b_tot                                     t_number_table;

        e_inst_id                                 t_number_table;
        e_event                                  t_varchar2_table;
        e_wait_time_milli                         t_number_table;
        e_wait_count                              t_number_table;
        e_tot                                     t_number_table;


        v_ver number;
        l_sleep number:=60;
        l_cr_blks_served number :=0;
        l_cur_blks_served number :=0;

        i number:=1;
        ind varchar2(32);
begin

        for c1 in (
                  select inst_id ||'-'|| event || '-' || wait_time_milli indx,
                         inst_id, event, wait_time_milli, wait_count, tot from (
                            select inst_id, event,wait_time_milli, wait_count,
                            sum (wait_count) over(partition by inst_id, event order by
inst_id rows between unbounded preceding and unbounded following ) tot
                            from (
                               select * from gv$event_histogram where event like
'%&&event_name%'
                               order by inst_id, event#, WAIT_TIME_MILLI
                             )
                    )
                   order by inst_id, event, WAIT_TIME_MILLI
                 )
         loop
                key_table(i):= c1.indx;
                b_inst_id (c1.indx) := c1.inst_id;
                b_event (c1.indx) := c1.event;
                b_wait_time_milli (c1.indx) := c1.wait_time_milli;
                b_wait_count (c1.indx) := c1.wait_count;
                b_tot (c1.indx) := c1.tot;
                i := i+1;
        end loop;
```

```
        select upper(nvl('&sleep',60)) into l_sleep from dual;
       dbms_lock.sleep(l_sleep);

        i:=1;

       for c2 in (
                 select inst_id ||'-'|| event || '-' || wait_time_milli indx,
                        inst_id, event, wait_time_milli, wait_count, tot from (
                    select inst_id, event,wait_time_milli, wait_count,
                    sum (wait_count) over(partition by inst_id, event order by
inst_id rows between unbounded preceding and unbounded following ) tot
                    from (
                       select * from gv$event_histogram where event like
'%&&event_name%'
                       order by inst_id, event#, WAIT_TIME_MILLI
                     )
                  )
                order by inst_id, event, WAIT_TIME_MILLI
               )
        loop
               e_inst_id (c2.indx) := c2.inst_id;
               e_event (c2.indx) := c2.event;
               e_wait_time_milli (c2.indx) := c2.wait_time_milli;
               e_wait_count (c2.indx) := c2.wait_count;
               e_tot (c2.indx) := c2.tot;
               i := i+1;
       end loop;


       dbms_output.put_line ( '-----------|------------------------------|------------
|-----------|-----------|');
       dbms_output.put_line ( 'Inst ID    | Event                        |Wait time ms|
Wait count| Percent   |');
       dbms_output.put_line ( '-----------|------------------------------|------------
|-----------|-----------|');

       for indx in key_table.first .. key_table.last
         loop
          ind :=  key_table (indx);
          dbms_output.put_line ( lpad( e_inst_id(ind),11)  || '|' ||
                                 lpad(to_char(e_event (ind)),30)  || '|'||
                                 lpad(to_char(e_wait_time_milli (ind) ),12)  || '|'||
                                  lpad(e_wait_count (ind) - b_wait_count(ind),11) || '|'
||
                                 lpad(to_char(case when e_tot(ind) - b_tot(ind)=0 then 0
                                           else trunc (100*(e_wait_count (ind) -
b_wait_count(ind))/(e_tot(ind) - b_tot(ind)),2)
                                           end
                                       ),11) || '|'  )
                                ;
        end loop;
        dbms_output.put_line ( '------------------------------------------------------
------------------------');
        dbms_output.put_line ( ' ');
end;
/
```

Appendix #2

```
set serveroutput on size 100000
REM ------------------------------------------------------------------------------------------
--------------
REM Author: Riyaj Shamsudeen @OraInternals, LLC
REM         www.orainternals.com
REM
REM Functionality: This script is to print GC processing timing for the past N seconds
or so
REM ***************
REM
REM Source  : gv$sysstat
REM
REM Note : 1. Keep window 160 columns for better visibility.
REM
REM Exectution type: Execute from sqlplus or any other tool.  Modify sleep as needed.
Default is 60 seconds
REM
REM Parameters:
REM No implied or explicit warranty
REM
REM Please send me an email to rshamsud@orainternals.com for any question..
REM  NOTE   1. Querying gv$ tables when there is a GC performance issue is not exactly
nice. So, don't run this too often.
REM         2. Until 11g, gv statistics did not include PQ traffic.
REM         3. Of course, this does not tell any thing about root cause :-)
REM @copyright : OraInternals, LLC. www.orainternals.com
REM Version    Change
REM ---------- --------------------
REM ------------------------------------------------------------------------------------------
--------------
PROMPT
PROMPT
PROMPT  gc_instance_cache.sql v1.10 by Riyaj Shamsudeen @orainternals.com
PROMPT
PROMPT  ...Prints various timing related information for the past N seconds
PROMPT  ...Default collection period is 60 seconds.... Please wait for at least 60
seconds...
PROMPT
PROMPT Column name key:
PROMPT   Inst -> Inst class :  source and target instance and class of the block
transfer
PROMPT   CR blk TX  :  CR blocks transmitted
PROMPT   CR blk tm  :  CR blocks time taken
PROMPT   CR blk av  :  Average time taken for CR block
PROMPT   CR bsy     :  Count of blocks suffered from "busy" events
PROMPT   CR bsy tm  :  Amount of time taken due to "busy" waits
PROMPT   CR bsy %   :  Percentage of CR busy time to CR time
PROMPT   CR congest :  Count of blocks suffered from "congestion" events
PROMPT   CR cngsttm :  Amount of time taken due to "congestion" waits
PROMPT   CR cng %   :  Percentage of CR congestion time to CR time
undef sleep
set lines 170 pages 100
set verify off
```

```
declare
        type t_number_table   is table of number       index by varchar2(32);
        type t_varchar2_table   is table of varchar2(32)       index by varchar2(32);
        type t_key_table   is table of varchar2(32)       index by binary_integer ;

         key_table t_key_table;

        b_inst_id  t_number_table;
        b_instance t_number_table;
        b_class t_varchar2_table;
        b_lost t_number_table;
        b_lost_time t_number_table;
        b_CR_BLOCK        t_number_table;
        b_CR_BLOCK_TIME   t_number_table;
        b_CR_2HOP         t_number_table;
        b_CR_2HOP_TIME    t_number_table;
        b_CR_3HOP         t_number_table;
        b_CR_3HOP_TIME    t_number_table;
        b_CR_BUSY         t_number_table;
        b_CR_BUSY_TIME    t_number_table;
        b_CR_CONGESTED    t_number_table;
        b_CR_CONGESTED_TIME   t_number_table;
        b_CURRENT_BLOCK   t_number_table;
        b_CURRENT_BLOCK_TIME    t_number_table;
        b_CURRENT_2HOP    t_number_table;
        b_CURRENT_2HOP_TIME  t_number_table;
        b_CURRENT_3HOP    t_number_table;
        b_CURRENT_3HOP_TIME    t_number_table;
        b_CURRENT_BUSY    t_number_table;
        b_CURRENT_BUSY_TIME   t_number_table;
        b_CURRENT_CONGESTED   t_number_table;
        b_CURRENT_CONGESTED_TIME             t_number_table;

        e_inst_id  t_number_table;
        e_instance t_number_table;
        e_class t_varchar2_table;
        e_lost t_number_table;
        e_lost_time t_number_table;
        e_CR_BLOCK        t_number_table;
        e_CR_BLOCK_TIME   t_number_table;
        e_CR_2HOP         t_number_table;
        e_CR_2HOP_TIME    t_number_table;
        e_CR_3HOP         t_number_table;
        e_CR_3HOP_TIME    t_number_table;
        e_CR_BUSY         t_number_table;
        e_CR_BUSY_TIME    t_number_table;
        e_CR_CONGESTED    t_number_table;
        e_CR_CONGESTED_TIME                               t_number_table;
        e_CURRENT_BLOCK   t_number_table;
        e_CURRENT_BLOCK_TIME              t_number_table;
        e_CURRENT_2HOP    t_number_table;
        e_CURRENT_2HOP_TIME     t_number_table;
        e_CURRENT_3HOP    t_number_table;
        e_CURRENT_3HOP_TIME   t_number_table;
        e_CURRENT_BUSY    t_number_table;
        e_CURRENT_BUSY_TIME    t_number_table;
```

```
        e_CURRENT_CONGESTED        t_number_table;
        e_CURRENT_CONGESTED_TIME                                t_number_table;

        v_ver number;
        l_sleep number:=60;
        l_cr_blks_served number :=0;
        l_cur_blks_served number :=0;

        i number:=1;
        ind varchar2(32);
begin
        for c1 in ( select instance ||','|| inst_id || ',' || class indx, ic.* from
gv$instance_cache_transfer ic where cr_block >0)
         loop
                key_table(i):= c1.indx;
                b_inst_id (c1.indx) := c1.inst_id;
                b_instance (c1.indx) := c1.instance;
                b_class (c1.indx) := c1.class;
                b_lost (c1.indx) := c1.lost;
                b_lost_time (c1.indx) := c1.lost_time;
                b_CR_BLOCK  (c1.indx) := c1.cr_block;
                b_CR_BLOCK_TIME (c1.indx) := c1.cr_block_time;
                b_CR_2HOP   (c1.indx) := c1.cr_2hop;
                b_CR_2HOP_TIME (c1.indx) := c1.cr_2hop_time;
                b_CR_3HOP  (c1.indx) := c1.cr_3hop;
                b_CR_3HOP_TIME (c1.indx) := c1.cr_3hop_time;
                b_CR_BUSY  (c1.indx) := c1.cr_busy;
                b_CR_BUSY_TIME  (c1.indx) := c1.cr_busy_time;
                b_CR_CONGESTED  (c1.indx) := c1.cr_congested;
                b_CR_CONGESTED_TIME (c1.indx) := c1.cr_congested_time;
                b_CURRENT_BLOCK    (c1.indx) := c1.current_block;
                b_CURRENT_BLOCK_TIME (c1.indx) := c1.current_block_time;
                b_CURRENT_2HOP     (c1.indx) := c1.current_2hop;
                b_CURRENT_2HOP_TIME (c1.indx) := c1.current_2hop_time;
                b_CURRENT_3HOP  (c1.indx) := c1.current_3hop;
                b_CURRENT_3HOP_TIME  (c1.indx) := c1.current_3hop_time;
                b_CURRENT_BUSY     (c1.indx) := c1.current_busy;
                b_CURRENT_BUSY_TIME (c1.indx) := c1.current_busy_time;
                b_CURRENT_CONGESTED (c1.indx) := c1.current_congested;
                b_CURRENT_CONGESTED_TIME (c1.indx) := c1.current_congested_time;
                i := i+1;
        end loop;

          select upper(nvl('&sleep',60)) into l_sleep from dual;
         dbms_lock.sleep(l_sleep);

        for c1 in ( select instance ||','|| inst_id || ',' || class indx, ic.* from
gv$instance_cache_transfer ic where cr_block >0)
         loop
                e_inst_id (c1.indx) := c1.inst_id;
                e_instance (c1.indx) := c1.instance;
                e_class (c1.indx) := c1.class;
                e_lost (c1.indx) := c1.lost;
                e_lost_time (c1.indx) := c1.lost_time;
                e_CR_BLOCK  (c1.indx) := c1.cr_block;
                e_CR_BLOCK_TIME (c1.indx) := c1.cr_block_time;
```

```
                e_CR_2HOP   (c1.indx) := c1.cr_2hop;
                e_CR_2HOP_TIME (c1.indx) := c1.cr_2hop_time;
                e_CR_3HOP  (c1.indx) := c1.cr_3hop;
                e_CR_3HOP_TIME (c1.indx) := c1.cr_3hop_time;
                e_CR_BUSY  (c1.indx) := c1.cr_busy;
                e_CR_BUSY_TIME  (c1.indx) := c1.cr_busy_time;
                e_CR_CONGESTED  (c1.indx) := c1.cr_congested;
                e_CR_CONGESTED_TIME (c1.indx) := c1.cr_congested_time;
                e_CURRENT_BLOCK    (c1.indx) := c1.current_block;
                e_CURRENT_BLOCK_TIME (c1.indx) := c1.current_block_time;
                e_CURRENT_2HOP     (c1.indx) := c1.current_2hop;
                e_CURRENT_2HOP_TIME (c1.indx) := c1.current_2hop_time;
                e_CURRENT_3HOP  (c1.indx) := c1.current_3hop;
                e_CURRENT_3HOP_TIME  (c1.indx) := c1.current_3hop_time;
                e_CURRENT_BUSY     (c1.indx) := c1.current_busy;
                e_CURRENT_BUSY_TIME (c1.indx) := c1.current_busy_time;
                e_CURRENT_CONGESTED (c1.indx) := c1.current_congested;
                e_CURRENT_CONGESTED_TIME (c1.indx) := c1.current_congested_time;
        end loop;
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|-----------|-----------|-----------|');
        dbms_output.put_line ( 'Inst->Inst class      | CR blk Tx |CR blk tm  | CR blk
av | CR 2hop   | CR2hop tm | CR 2hop   | CR 3hop   |CR 3hop  tm|CR 3hop av |');
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|-----------|-----------|-----------|');
        for i in key_table.first .. key_table.last
          loop
           ind :=  key_table (i);
           if ( e_cr_block (ind) - b_cr_block(ind) >0) then
           dbms_output.put_line ( rpad( e_instance(ind) ||'->'|| e_inst_id(ind)||'
'||e_class (ind), 22 )  || '|' ||
                                  lpad(to_char(e_cr_block (ind) - b_cr_block(ind)),11)  ||
'|'||
                                  lpad(to_char(e_cr_block_time (ind) -
b_cr_block_time(ind)),11)  || '|'||
                                  lpad(to_char(case when e_cr_block(ind) -
b_cr_block(ind)=0 then 0
                                                 else trunc ((e_cr_block_time (ind) -
b_cr_block_time(ind))/(e_cr_block(ind) - b_cr_block(ind))/1000,2)
                                                 end
                                               ),11)   ||'|'||
                                  lpad(to_char(e_cr_2hop (ind) - b_cr_2hop(ind)),11)  ||
'|'||
                                  lpad(to_char(e_cr_2hop_time (ind) -
b_cr_2hop_time(ind)),11)  || '|'||
                                  lpad(to_char(case when e_cr_2hop(ind) - b_cr_2hop(ind)=0
then 0
                                                 else trunc ((e_cr_2hop_time (ind) -
b_cr_2hop_time(ind))/(e_cr_2hop(ind) - b_cr_2hop(ind))/1000,2)
                                                 end
                                               ),11)   ||'|'||
                                  lpad(to_char(e_current_3hop (ind) -
b_current_3hop(ind)),11)  || '|'||
                                  lpad(to_char(e_current_3hop_time (ind) -
b_current_3hop_time(ind)),11)  || '|'||
```

```
                                lpad(to_char(case when e_current_3hop(ind) -
b_current_3hop(ind)=0 then 0
                                             else trunc ((e_current_3hop_time (ind) -
b_current_3hop_time(ind))/(e_current_3hop(ind) - b_current_3hop(ind))/1000,2)
                                             end
                                            ),11)   || '|'
                                 );
         end if;
         end loop;
         dbms_output.put_line ( '-----------------------------------------------------
-------------------------------------------------------------------------');
         dbms_output.put_line ( ' ');
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|-----------|-----------|-----------|');
        dbms_output.put_line ( 'Inst->Inst class      | CUR blk Tx|CUR blk tm | CUR blk
av| CUR 2hop  | CUR2hop tm| CUR 2hop  | CUR 3hop  |CUR 3hop tm|CUR 3hop av|');
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|-----------|-----------|-----------|');
        for i in key_table.first .. key_table.last
          loop
          ind :=  key_table (i);
          if ( e_current_block (ind) - b_current_block(ind) >0) then
          dbms_output.put_line ( rpad( e_instance(ind) ||'->'|| e_inst_id(ind)||'
'||e_class (ind), 22 )  || '|' ||
                                lpad(to_char(e_current_block (ind) -
b_current_block(ind)),11)  || '|'||
                                lpad(to_char(e_current_block_time (ind) -
b_current_block_time(ind)),11)  || '|'||
                                lpad(to_char(case when e_current_block(ind) -
b_current_block(ind)=0 then 0
                                             else trunc ((e_current_block_time (ind) -
b_current_block_time(ind))/(e_current_block(ind) - b_current_block(ind))/1000,2)
                                             end
                                            ),11)   ||'|'||
                                lpad(to_char(e_current_2hop (ind) -
b_current_2hop(ind)),11)  || '|'||
                                lpad(to_char(e_current_2hop_time (ind) -
b_current_2hop_time(ind)),11)  || '|'||
                                lpad(to_char(case when e_current_2hop(ind) -
b_current_2hop(ind)=0 then 0
                                             else trunc ((e_current_2hop_time (ind) -
b_current_2hop_time(ind))/(e_current_2hop(ind) - b_current_2hop(ind))/1000,2)
                                             end
                                            ),11)   ||'|'||
                                lpad(to_char(e_current_3hop (ind) -
b_current_3hop(ind)),11)  || '|'||
                                lpad(to_char(e_current_3hop_time (ind) -
b_current_3hop_time(ind)),11)  || '|'||
                                lpad(to_char(case when e_current_3hop(ind) -
b_current_3hop(ind)=0 then 0
                                             else trunc ((e_current_3hop_time (ind) -
b_current_3hop_time(ind))/(e_current_3hop(ind) - b_current_3hop(ind))/1000,2)
                                             end
                                            ),11)  || '|'
                                 );
         end if;
```

```
         end loop;
         dbms_output.put_line ( '-------------------------------------------------------
-------------------------------------------------------------------------------');
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|');
        dbms_output.put_line ( 'Inst->Inst class      |  CRbsy    |  CRbsy tm |  CRbsy %
|  CRcongest| CRcngst tm| CRcng %   |');
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|');
        for i in key_table.first .. key_table.last
          loop
          ind :=  key_table (i);
          if ( e_cr_block (ind) - b_cr_block(ind) >0) then
          dbms_output.put_line ( rpad( e_instance(ind) ||'->'|| e_inst_id(ind)||'
'||e_class (ind), 22 )  || '|' ||
                                 lpad(to_char(e_cr_busy (ind) - b_cr_busy(ind)),11)  ||
'|' ||
                                 lpad(to_char(e_cr_busy_time (ind) -
b_cr_busy_time(ind)),11)  ||'|'||
                                 lpad(to_char(case when e_cr_block_time (ind) -
b_cr_block_time(ind)=0 then 0
                                                  else trunc (100*(e_cr_busy_time (ind) -
b_cr_busy_time(ind))/(e_cr_block_time (ind) - b_cr_block_time(ind)),2)
                                                  end
                                               ),11)   ||'|'||
                                 lpad(to_char(e_cr_congested (ind) -
b_cr_congested(ind)),11)  || '|' ||
                                 lpad(to_char(e_cr_congested_time (ind) -
b_cr_congested_time(ind)),11)  ||'|'||
                                 lpad(to_char(case when e_cr_block_time (ind) -
b_cr_block_time(ind)=0 then 0
                                                  else trunc (100*(e_cr_congested_time (ind)
- b_cr_congested_time(ind))/(e_cr_block_time (ind) - b_cr_block_time(ind)),2)
                                                  end
                                               ),11) || '|'
                                  );
         end if;
         end loop;
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|');
        dbms_output.put_line ( '-------------------------------------------------------
-------------------------------------|');
        dbms_output.put_line ( 'Inst->Inst class      | CURbsy    | CURbsy tm | CURbsy %
| CURcongest|CURcngst tm|CURcng %   |');
        dbms_output.put_line ( '----------------------|-----------|-----------|---------
--|-----------|-----------|-----------|');
        for i in key_table.first .. key_table.last
          loop
          ind :=  key_table (i);
          if ( e_current_block (ind) - b_current_block(ind) >0) then
          dbms_output.put_line ( rpad( e_instance(ind) ||'->'|| e_inst_id(ind)||'
'||e_class (ind), 22 )  || '|' ||
                                 lpad(to_char(e_current_busy (ind) -
b_current_busy(ind)),11)  || '|' ||
                                 lpad(to_char(e_current_busy_time (ind) -
b_current_busy_time(ind)),11)  ||'|'||
```

```
                              lpad(to_char(case when e_current_block_time (ind) -
b_current_block_time(ind)=0 then 0
                                           else trunc (100*(e_current_busy_time (ind)
- b_current_busy_time(ind))/(e_current_block_time (ind) - b_current_block_time(ind)),2)
                                           end
                                         ),11)   ||'|'||
                              lpad(to_char(e_current_congested (ind) -
b_current_congested(ind)),11)  || '|' ||
                              lpad(to_char(e_current_congested_time (ind) -
b_current_congested_time(ind)),11)  ||'|'||
                              lpad(to_char(case when e_current_block_time (ind) -
b_current_block_time(ind)=0 then 0
                                           else trunc (100*(e_current_congested_time
(ind) - b_current_congested_time(ind))/(e_current_block_time (ind) -
b_current_block_time(ind)),2)
                                           end
                                         ),11) || '|'
                               );
         end if;
         end loop;
        dbms_output.put_line ( '----------------------------------------------------------
-------------------------------------|');
end;
/
set verify on
```

Appendix #3:

```
set serveroutput on size 100000
REM ------------------------------------------------------------------------------------------
--------------
REM Author: Riyaj Shamsudeen @OraInternals, LLC
REM         www.orainternals.com
REM
REM Functionality: This script is to print GC processing timing for the past N seconds
or so
REM ***************
REM
REM Source  : gv$sysstat
REM
REM Note : 1. Keep window 160 columns for better visibility.
REM
REM Exectution type: Execute from sqlplus or any other tool.  Modify sleep as needed.
Default is 60 seconds
REM
REM Parameters:
REM No implied or explicit warranty
REM
REM Please send me an email to rshamsud@orainternals.com for any question..
REM  NOTE   1. Querying gv$ tables when there is a GC performance issue is not exactly
nice. So, don't run this too often.
REM         2. Until 11g, gv statistics did not include PQ traffic.
REM         3. Of course, this does not tell any thing about root cause :-)
REM @copyright : OraInternals, LLC. www.orainternals.com
REM Version    Change
REM ---------- --------------------
REM ------------------------------------------------------------------------------------------
--------------
set serveroutput on size 100000
undef name_name
undef sleep
PROMPT
PROMPT
PROMPT  name_histogram_delta.sql v1.10 by Riyaj Shamsudeen @orainternals.com
PROMPT
PROMPT  ...Prints various timing related information for the past N seconds
PROMPT  ...Default collection period is 60 seconds.... Please wait for at least 60
seconds...
PROMPT
undef name_name
undef sleep

PROMPT Average instance-wide
PROMPT ---------------------
select inst_id, name, average_wait from gv$system_name
where name like '%&&name_name%'
;
set lines 170 pages 100
set verify off

declare
       type t_number_table   is table of number       index by varchar2(32);
```

```
        type t_varchar2_table  is table of varchar2(60)       index by varchar2(32);
        type t_key_table  is table of varchar2(60)       index by binary_integer;

         key_table                   t_key_table;

        b_inst_id                   t_number_table;
        b_name                     t_varchar2_table;
        b_value           t_number_table;
        b_wait_count                t_number_table;
        b_tot                       t_number_table;

        e_inst_id                   t_number_table;
        e_name                     t_varchar2_table;
        e_value           t_number_table;
        e_wait_count                t_number_table;
        e_tot                       t_number_table;


        v_ver number;
        l_sleep number:=60;
        l_cr_blks_served number :=0;
        l_cur_blks_served number :=0;

        i number:=1;
        ind varchar2(32);
begin

        for c1 in (
          select inst_id||'_'||sys.name indx,inst_id, sys.name, sys.value from
gv$sysstat sys
          where sys.name in (
            'gc cr block build time',
            'gc cr block send time',
            'gc cr block flush time',
            'gc current block pin time',
            'gc current block send time',
            'gc current block flush time'
          )
        )
        loop
               key_table(i):= c1.indx;
               b_inst_id (c1.indx) := c1.inst_id;
               b_name (c1.indx) := c1.name;
               b_value (c1.indx) := c1.value;
               i := i+1;
        end loop;

        select upper(nvl('&sleep',60)) into l_sleep from dual;
        dbms_lock.sleep(l_sleep);

        i:=1;

        for c2 in (
          select inst_id||'_'||sys.name indx,inst_id, sys.name, sys.value from
gv$sysstat sys
          where sys.name in (
```

```
            'gc cr block build time',
            'gc cr block send time',
            'gc cr block flush time',
            'gc current block pin time',
            'gc current block send time',
            'gc current block flush time'
          )
         )
         loop
                e_inst_id (c2.indx) := c2.inst_id;
                e_name (c2.indx) := c2.name;
                e_value (c2.indx) := c2.value;
                i := i+1;
        end loop;


        dbms_output.put_line ( '|----|------------------------------|-----------|');
        dbms_output.put_line ( '|Inst| Name                         |      Time |');
        dbms_output.put_line ( '|----|------------------------------|-----------|');

        for indx in key_table.first .. key_table.last
          loop
           ind :=  key_table (indx);
           dbms_output.put_line (  '|' ||
                                   lpad( e_inst_id(ind),4)  || '|' ||
                                  lpad(to_char(e_name (ind)),30)  || '|'||
                                  lpad(to_char(e_value (ind) ),11)  || '|'
                                    )
                                   ;
         end loop;
         dbms_output.put_line ( '-------------------------------------------------------
------------------------');
        dbms_output.put_line ( ' ');
end;
/
set verify on
```