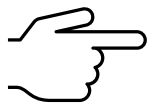
Oracle DB Monitoring and Performance Tuning SQL Scripts

Data Guard, ASM, DATAPUMP ...

Asfaw Gedamu Haileselasie



Caution: Please use the commands with care, try them on test environments first.

Find CPU and memory usage of oracle DB server

Below script is useful in getting CPU, memory and core, socket information of a database server from SQL prompt.

SCRIPT:

```
set pagesize 299
set lines 299
select STAT_NAME,to_char(VALUE) as VALUE  ,COMMENTS from v$osstat where
stat_name  IN ('NUM_CPUS','NUM_CPU_CORES','NUM_CPU_SOCKETS')
union
select STAT_NAME,VALUE/1024/1024/1024 || ' GB'  ,COMMENTS from v$osstat where
stat_name  IN ('PHYSICAL_MEMORY_BYTES')
```

OUTPUT:

```
STAT_NAME                      VALUE
COMMENTS---- ----------------------------------------------------------
NUM_CPUS                       256
Number of active CPUs
NUM_CPU_CORES                  32
Number of CPU cores
NUM_CPU_SOCKETS                4
Number of physical CPU sockets
PHYSICAL_MEMORY_BYTES          255.5 GB
Physical memory size in bytes
```

Find sessions that are consuming lot of CPU

Use below query to find the sessions using a lot of CPU.

```
col program form a30 heading "Program"
col CPUMins form 99990 heading "CPU in Mins"
```

```
select rownum as rank, a.*
from (
SELECT v.sid, program, v.value / (100 * 60) CPUMins
FROM v$statname s , v$sesstat v, v$session sess
WHERE s.name = 'CPU used by this session'
and sess.sid = v.sid
and v.statistic#=s.statistic#
and v.value>0
ORDER BY v.value DESC) a
where rownum < 11;
```

Find CPU usage and wait event information in oracle database

Below script will give information about the CPU usage and wait events class information of every minute for last 2 hours. As this query uses gv$active_session_history, so make sure you have TUNING license pack of oracle, before using this.

```
set lines 288
col sample_time for a14
col CONFIGURATION head "CONFIG" for 99.99
col ADMINISTRATIVE head "ADMIN" for 99.99
col OTHER for 99.99

SELECT TO_CHAR(SAMPLE_TIME, 'HH24:MI ') AS SAMPLE_TIME,
       ROUND(OTHER / 60, 3) AS OTHER,
       ROUND(CLUST / 60, 3) AS CLUST,
       ROUND(QUEUEING / 60, 3) AS QUEUEING,
       ROUND(NETWORK / 60, 3) AS NETWORK,
       ROUND(ADMINISTRATIVE / 60, 3) AS ADMINISTRATIVE,
       ROUND(CONFIGURATION / 60, 3) AS CONFIGURATION,
       ROUND(COMMIT / 60, 3) AS COMMIT,
       ROUND(APPLICATION / 60, 3) AS APPLICATION,
       ROUND(CONCURRENCY / 60, 3) AS CONCURRENCY,
       ROUND(SIO / 60, 3) AS SYSTEM_IO,
       ROUND(UIO / 60, 3) AS USER_IO,
       ROUND(SCHEDULER / 60, 3) AS SCHEDULER,
       ROUND(CPU / 60, 3) AS CPU,
       ROUND(BCPU / 60, 3) AS BACKGROUND_CPU
  FROM (SELECT TRUNC(SAMPLE_TIME, 'MI') AS SAMPLE_TIME,
               DECODE(SESSION_STATE,
                      'ON CPU',
                      DECODE(SESSION_TYPE, 'BACKGROUND', 'BCPU', 'ON CPU'),
                      WAIT_CLASS) AS WAIT_CLASS
          FROM V$ACTIVE_SESSION_HISTORY
         WHERE SAMPLE_TIME > SYSDATE - INTERVAL '2'
         HOUR
           AND SAMPLE_TIME <= TRUNC(SYSDATE, 'MI')) ASH PIVOT(COUNT(*)
  FOR WAIT_CLASS IN('ON CPU' AS CPU,'BCPU' AS BCPU,
'Scheduler' AS SCHEDULER,
'User I/O' AS UIO,
'System I/O' AS SIO,
```

```
'Concurrency' AS CONCURRENCY,
'Application' AS  APPLICATION,
'Commit' AS  COMMIT,
'Configuration' AS CONFIGURATION,
'Administrative' AS   ADMINISTRATIVE,
'Network' AS  NETWORK,
'Queueing' AS   QUEUEING,
'Cluster' AS   CLUST,
'Other' AS  OTHER))
```

OUTPUT WILL LOOK AS BELOW:

```
SAMPLE_TIME OTHER CLUST QUEUEING NETWORK ADMIN CONFIG COMMIT APPLICATION CONCURRENCY SYSTEM_IO USER_IO SCHEDULER CPU BACK
GROUND_CPU
-------------- ------ ---------- ---------- ---------- ------ ------ ---------- ----------- ----------- ----------
---------- ---------- ---------- --------------
10:13 1.60 .033 0 0 .00 .00 1.2 0 .017 .617 .017 0 2.567 .317
10:14 2.08 .033 0 0 .00 .00 1.9 .067 0 .783 .217 0 3.433 .417
10:15 2.83 0 0 0 .00 .00 1.6 .017 .033 .9 .5 0 4.05 .767
10:16 3.15 .05 0 .033 .00 .00 2.033 0 .033 .9 .617 0 4.7 .75
10:17 3.18 0 0 .017 .00 .00 1.883 0 0 .717 .467 0 4.65 .683
10:18 2.78 0 0 0 .00 .00 1.417 0 0 .65 .067 0 3.867 .65
10:19 3.30 0 0 0 .00 .00 1.233 0 .083 .533 .033 0 4.533 .9
10:20 4.18 .033 0 .033 .00 .00 1.3 0 .017 .933 .117 0 5.5 .817
10:21 2.85 .033 0 0 .00 .00 1.15 0 0 .783 .05 0 4.1 .75
10:22 3.32 .017 0 .017 .00 .00 4.133 0 .017 .9 .033 0 5.317 1.033
```

Monitor parallel queries in oracle db

Use below query to monitor currently running queries with parallel threads.

```
col username for a9
col sid for a8
set lines 299
select
      s.inst_id,
      decode(px.qcinst_id,NULL,s.username,
            ' - '||lower(substr(s.program,length(s.program)-4,4) ) )
"Username",
      decode(px.qcinst_id,NULL, 'QC', '(Slave)') "QC/Slave" ,
      to_char( px.server_set) "Slave Set",
      to_char(s.sid) "SID",
      decode(px.qcinst_id, NULL ,to_char(s.sid) ,px.qcsid) "QC SID",
      px.req_degree "Requested DOP",
     px.degree "Actual DOP", p.spid
   from
     gv$px_session px,
     gv$session s, gv$process p
   where
     px.sid=s.sid (+) and
     px.serial#=s.serial# and
     px.inst_id = s.inst_id
```

```
     and p.inst_id = s.inst_id
     and p.addr=s.paddr
  order by 5 , 1 desc
/
```

Find user commits per minute in oracle DB

Below script is useful in getting user commit statistics information in the oracle database. user commits is the number of commits happening the database. It will be helpful in tracking the number of transactions in the database. STAT_PER_MIN -Number of commits per minutes, during that snap time

```
col STAT_NAME for a20
col VALUE_DIFF for 9999,999,999
col STAT_PER_MIN for 9999,999,999
set lines 200 pages 1500 long 99999999
col BEGIN_INTERVAL_TIME for a30
col END_INTERVAL_TIME for a30
set pagesize 40
set pause on


select hsys.SNAP_ID,
       hsnap.BEGIN_INTERVAL_TIME,
       hsnap.END_INTERVAL_TIME,
           hsys.STAT_NAME,
           hsys.VALUE,
           hsys.VALUE - LAG(hsys.VALUE,1,0) OVER (ORDER BY hsys.SNAP_ID) AS
"VALUE_DIFF",
           round((hsys.VALUE - LAG(hsys.VALUE,1,0) OVER (ORDER BY
hsys.SNAP_ID)) /
           round(abs(extract(hour from (hsnap.END_INTERVAL_TIME -
hsnap.BEGIN_INTERVAL_TIME))*60 +
           extract(minute from (hsnap.END_INTERVAL_TIME -
hsnap.BEGIN_INTERVAL_TIME)) +
           extract(second from (hsnap.END_INTERVAL_TIME -
hsnap.BEGIN_INTERVAL_TIME))/60),1)) "STAT_PER_MIN"
from dba_hist_sysstat hsys, dba_hist_snapshot hsnap
 where hsys.snap_id = hsnap.snap_id
 and hsnap.instance_number in (select instance_number from v$instance)
 and hsnap.instance_number = hsys.instance_number
 and hsys.STAT_NAME='user commits'
 order by 1;
```

OUTPUT:

```
SNAP_ID BEGIN_INTERVAL_TIME END_INTERVAL_TIME STAT_NAME VALUE VALUE_DIFF STAT_PER_MIN
--------- ---------------------------- ---------------------------- ------------------- -----
-------------
6626 11-NOV-17 05.00.13.272 PM 11-NOV-17 06.00.29.527 PM user commits 350001525 1,147,017 19,022
6627 11-NOV-17 06.00.29.527 PM 11-NOV-17 07.00.14.759 PM user commits 351130223 1,128,698 18,875
6628 11-NOV-17 07.00.14.759 PM 11-NOV-17 08.00.02.845 PM user commits 351987886 857,663 14,342
6629 11-NOV-17 08.00.02.845 PM 11-NOV-17 09.00.22.109 PM user commits 352829839 841,953 13,963
6630 11-NOV-17 09.00.22.109 PM 11-NOV-17 10.00.07.076 PM user commits 353478483 648,644 10,865
6631 11-NOV-17 10.00.07.076 PM 11-NOV-17 11.00.24.303 PM user commits 353939928 461,445 7,652
6632 11-NOV-17 11.00.24.303 PM 12-NOV-17 12.00.11.904 AM user commits 354335275 395,347 6,611
6633 12-NOV-17 12.00.11.904 AM 12-NOV-17 01.00.29.406 AM user commits 354604745 269,470 4,469
6634 12-NOV-17 01.00.29.406 AM 12-NOV-17 02.00.17.332 AM user commits 354955934 351,189 5,873
6635 12-NOV-17 02.00.17.332 AM 12-NOV-17 03.00.03.228 AM user commits 356918293 1,962,359 32,815
6636 12-NOV-17 03.00.03.228 AM 12-NOV-17 04.00.20.577 AM user commits 357821672 903,379 14,981
6637 12-NOV-17 04.00.20.577 AM 12-NOV-17 05.00.09.204 AM user commits 358154880 333,208 5,572
6638 12-NOV-17 05.00.09.204 AM 12-NOV-17 06.00.25.507 AM user commits 358296694 141,814 2,352
6639 12-NOV-17 06.00.25.507 AM 12-NOV-17 07.00.09.734 AM user commits 358692156 395,462 6,624
6640 12-NOV-17 07.00.09.734 AM 12-NOV-17 08.00.01.047 AM user commits 359373748 681,592 11,379
6641 12-NOV-17 08.00.01.047 AM 12-NOV-17 09.00.17.981 AM user commits 360418586 1,044,838 17,327
6642 12-NOV-17 09.00.17.981 AM 12-NOV-17 10.00.04.542 AM user commits 362476024 2,057,438 34,405
6643 12-NOV-17 10.00.04.542 AM 12-NOV-17 11.00.22.732 AM user commits 364469092 1,993,068 33,053
6644 12-NOV-17 11.00.22.732 AM 12-NOV-17 12.00.09.693 PM user commits 365611444 1,142,352 19,103
6645 12-NOV-17 12.00.09.693 PM 12-NOV-17 01.00.27.672 PM user commits 366866479 1,255,035 20,813
6646 12-NOV-17 01.00.27.672 PM 12-NOV-17 02.00.14.537 PM user commits 368466462 1,599,983 26,756
```

Find active transactions in oracle database

Below script can be used to find the active transactions in the oracle database.

```
col name format a10
col username format a8
col osuser format a8
col start_time format a17
col status format a12
tti 'Active transactions'

select s.sid,username,t.start_time, r.name, t.used_ublk "USED BLKS",
decode(t.space, 'YES', 'SPACE TX',
decode(t.recursive, 'YES', 'RECURSIVE TX',
decode(t.noundo, 'YES', 'NO UNDO TX', t.status)
)) status
from sys.v_$transaction t, sys.v_$rollname r, sys.v_$session s
where t.xidusn = r.usn
and t.ses_addr = s.saddr
/
```

Find distributed pending transactions in oracle DB

Below script will display information about the distributed pending transactions in oracle.

```
COL local_tran_id FORMAT a13
COL in_out FORMAT a6
COL database FORMAT a25
COL dbuser_owner FORMAT a15
COL interface FORMAT a3
SELECT local_tran_id, in_out, database, dbuser_owner, interface
FROM dba_2pc_neighbors
/
```

Active/ Standby Data Guard useful SQL scripts

1. Basic information of database (primary or standby)

```
SQL> SELECT DATABASE_ROLE, DB_UNIQUE_NAME INSTANCE, OPEN_MODE,
PROTECTION_MODE, PROTECTION_LEVEL, SWITCHOVER_STATUS FROM V$DATABASE;
DATABASE_ROLE INSTANCE OPEN_MODE PROTECTION_MODE PROTECTION_LEVEL
SWITCHOVER_STATUS
-------- --------------- ----------
PHYSICAL STANDBY stdby READ ONLY MAXIMUM PERFORMANCE MAXIMUM PERFORMANCE NOT
ALLOWED
```

2. Check for messages/errors

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;
MESSAGE
----------------------------------------
ARC0: Archival started
ARC1: Archival started
ARC2: Archival started
ARC2: Becoming the 'no FAL' ARCH
ARC1: Becoming the heartbeat ARCH
ARC1: Becoming the active heartbeat ARCH
ARC3: Archival started
```

3. To display current status information for specific physical standby database background processes.

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, BLOCK#, BLOCKS FROM
V$MANAGED_STANDBY ;
PROCESS STATUS THREAD# SEQUENCE# BLOCK# BLOCKS
----- ------ ----- ----- ----- -----
```

```
ARCH CONNECTED 0 0 0 0
ARCH CONNECTED 0 0 0 0
ARCH CLOSING 1 54 45056 755
ARCH CLOSING 1 57 1 373
RFS IDLE 0 0 0 0
RFS IDLE 0 0 0 0
RFS IDLE 0 0 0 0
RFS IDLE 1 58 30239 1

8 rows selected.
```

4. Show received archived logs on physical standby

Run this query on physical standby

```
SQL> select registrar, creator, thread#, sequence#, first_change#,
next_change# from v$archived_log;
REGISTR CREATOR THREAD# SEQUENCE# FIRST_CHANGE# NEXT_CHANGE#
---- ---- ----- ----- ------- ------
RFS ARCH 1 29 1630326 1631783
RFS ARCH 1 30 1631783 1632626
RFS LGWR 1 31 1632626 1669359
RFS ARCH 1 33 1676050 1676124
RFS ARCH 1 32 1669359 1676050
RFS ARCH 1 35 1681145 1681617
RFS ARCH 1 34 1676124 1681145
RFS ARCH 1 37 1688494 1688503
RFS ARCH 1 36 1681617 1688494
RFS ARCH 1 38 1688503 1689533
RFS LGWR 1 39 1689533 1697243
```

5. To check the log status

```
SQL> select 'Last Log applied : ' Logs, to_char(next_time,'DD-MON-
YY:HH24:MI:SS') Time
from v$archived_log
where sequence# = (select max(sequence#) from v$archived_log where
applied='YES')
union
select 'Last Log received : ' Logs, to_char(next_time,'DD-MON-YY:HH24:MI:SS')
Time
from v$archived_log
where sequence# = (select max(sequence#) from v$archived_log);
LOGS TIME
---------- -------------
Last Log applied : 24-MAR-14:10:11:10
Last Log received : 27-MAR-14:12:40:17
```

6. To display various information about the redo data. This includes redo data generated by the primary database that is not yet available on the standby database and how much redo has not yet been applied to the standby database.

```
set lines 132
col value format a20
```

```
SQL> select name, value from V$DATAGUARD_STATS;
NAME VALUE
--------------- ----------
transport lag +00 00:00:00
apply lag
apply finish time
estimated startup time 23
```

7. to monitor efficient recovery operations as well as to estimate the time required to complete the current operation in progress:

```
SQL> select to_char(start_time, 'DD-MON-RR HH24:MI:SS') start_time,
item, round(sofar/1024,2) "MB/Sec"
from v$recovery_progress
where (item='Active Apply Rate' or item='Average Apply Rate');
START_TIME ITEM MB/SEC
------ ---------------- ----
27-MAR-14 15:49:44 Active Apply Rate 8.5
27-MAR-14 15:49:44 Average Apply Rate 6.30
```

8. To find last applied log

```
SQL> select to_char(max(FIRST_TIME),'hh24:mi:ss dd/mm/yyyy') FROM
V$ARCHIVED_LOG where applied='YES';
TO_CHAR(MAX(FIRST_T
----------
10:11:08 24/03/2014
```

9. To see if standby redo logs have been created. The standby redo logs should be the same size as the online redo logs. There should be ((# of online logs per thread + 1) * # of threads) standby redo logs. A value of 0 for the thread# means the log has never been allocated.

```
SQL> SELECT thread#, group#, sequence#, bytes, archived, status FROM
v$standby_log order by thread#, group#;
THREAD# GROUP# SEQUENCE# BYTES ARC STATUS
----- ----- ----- ----- -- ----------
1 8 0 104857600 NO UNASSIGNED
1 9 58 104857600 YES ACTIVE
1 10 0 104857600 NO UNASSIGNED
1 11 0 104857600 YES UNASSIGNED
```

10. To produce a list of defined archive destinations. It shows if they are enabled, what process is servicing that destination, if the destination is local or remote, and if remote what the current mount ID is. For a physical standby we should have at least one remote destination that points the primary set.

```
column destination format a35 wrap
column process format a7
column ID format 99
column mid format 99
```

```
SQL> SELECT thread#, dest_id, destination, gvad.status, target, schedule,
process, mountid mid FROM gv$archive_dest gvad, gv$instance gvi WHERE
gvad.inst_id = gvi.inst_id AND destination is NOT NULL ORDER BY thread#,
dest_id;
THREAD# DEST_ID DESTINATION STATUS TARGET SCHEDULE PROCESS MID
---- ---- ----------------------------------------
1 1 USE_DB_RECOVERY_FILE_DEST VALID LOCAL ACTIVE ARCH 0
1 2 brij VALID REMOTE PENDING LGWR 0
1 32 USE_DB_RECOVERY_FILE_DEST VALID LOCAL ACTIVE RFS 0
```

11. Verify the last sequence# received and the last sequence# applied to standby database.

```
SQL> SELECT al.thrd "Thread", almax "Last Seq Received", lhmax "Last Seq
Applied" FROM (select thread# thrd, MAX(sequence#) almax FROM v$archived_log
WHERE resetlogs_change#=(SELECT resetlogs_change# FROM v$database) GROUP BY
thread#) al, (SELECT thread# thrd, MAX(sequence#) lhmax FROM v$log_history
WHERE resetlogs_change#=(SELECT resetlogs_change# FROM v$database) GROUP BY
thread#) lh WHERE al.thrd = lh.thrd;
Thread Last Seq Received Last Seq Applied
------ --------- --------
1 57 53
```

Oracle EXPDP/IMPDP (DATAPUMP) Monitoring Scripts

Usually we monitor the EXPDP/IMPDP jobs by monitoring the log files generated by expdp/impdp process. Also we monitor alert log too just in case some error pops up. This helps most of the time. If you have a long running expdp/impdp sessions as you are exporting/importing huge GBs then it helps to have a more detailed monitoring of the expdp/impdp jobs. Some of the useful queries which can be used to monitor the Data Pump Jobs are mentioned below.

To start with some of the important tables/views that you should refer to monitor Data Pump Jobs are:

```
DBA_DATAPUMP_JOBS
DBA_DATAPUMP_SESSIONS
DBA_RESUMABLE
V$SESSION_LONGOPS
V$SESSION
V$DATAPUMP_JOB
```

1. Script to find status of work done

```
select x.job_name,ddj.state,ddj.job_mode,ddj.degree
, x.owner_name,z.sql_text, p.message
, p.totalwork, p.sofar
, round((p.sofar/p.totalwork)*100,2) done
, p.time_remaining
```

```
from dba_datapump_jobs ddj
left join dba_datapump_sessions x on (x.job_name = ddj.job_name)
left join v$session y on (y.saddr = x.saddr)
left join v$sql z on (y.sql_id = z.sql_id)
left join v$session_longops p ON (p.sql_id = y.sql_id)
WHERE y.module='Data Pump Worker'
AND p.time_remaining > 0;
```

2. Another simple script using only longops view

```
select
 round(sofar/totalwork*100,2) percent_completed,
 v$session_longops.*
from
 v$session_longops
where
 sofar <> totalwork
order by
 target, sid;
```

3. Procedure to find the status of job in terms of percentage & number of rows

```
SET SERVEROUTPUT ON
DECLARE
  ind NUMBER;
  h1 NUMBER;
  percent_done NUMBER;
  job_state VARCHAR2(30);
  js ku$_JobStatus;
  ws ku$_WorkerStatusList;
  sts ku$_Status;
BEGIN
h1 := DBMS_DATAPUMP.attach('&JOB_NAME', '&JOB_OWNER');
dbms_datapump.get_status(h1,
           dbms_datapump.ku$_status_job_error +
           dbms_datapump.ku$_status_job_status +
           dbms_datapump.ku$_status_wip, 0, job_state, sts);
js := sts.job_status;
ws := js.worker_status_list;
      dbms_output.put_line('** Job percent done = ' ||
                           to_char(js.percent_done));
      dbms_output.put_line('restarts - '||js.restart_count);
ind := ws.first;
  while ind is not null loop
    dbms_output.put_line('rows completed - '||ws(ind).completed_rows);
    ind := ws.next(ind);
  end loop;
DBMS_DATAPUMP.detach(h1);
end;
/
```

This package will need JOB_NAME and JOB_OWNER as input parameter. You can fetch this information from your export/import log or you can use the previous SQL script to get this information.

Remember that if you are doing expdp/impdp by SYSDBA then execute this package using the same SYSDBA privilege.

Oracle Scripts for Improving Database Performance

1. This script invokes the tracker utility to capture the UTLESTAT information into permanent tables and then drop the temporary tables. Remember This SQL script will load the tracking tables.

```
insert into track_stats
( oracle_sid, collection_started)
select '&1',min(stats_gather_times)
from sys.stats$dates;

update track_stats
set collection_ended =
                        (select max(stats_gather_times)
                        from sys.stats$dates),
        run_date = to_date(substr(collection_started,1,12),'DD-MON-YY HH24'),
                consistent_gets =
                        (select change
                        from sys.stats$stats
                        where name = 'consistent gets'),
                block_gets =
                        (select change
                        from sys.stats$stats
                        where name = 'db block gets'),
                physical_reads =
                        (select change
                        from sys.stats$stats
                        where name = 'physical reads'),
                buffer_busy_waits =
                        (select change
                        from sys.stats$stats
                        where name = 'buffer busy waits'),
                buffer_free_needed =
                        (select change
                        from sys.stats$stats
                        where name = 'free buffer requested'),
                free_buffer_waits =
                        (select change
                        from sys.stats$stats
                        where name = 'free buffer waits'),
                free_buffer_scans =
                        (select change
                        from sys.stats$stats
                        where name = 'free buffer scans'),
                enqueue_timeouts =
                        (select change
```

```
                        from sys.stats$stats
                        where name = 'enqueue timeouts'),
                redo_space_wait =
                        (select change
                        from sys.stats$stats
                        where name = 'redo log space wait time'),
                write_wait_time =
                        (select change
                        from sys.stats$stats
                        where name = 'write wait time'),
                write_complete_waits =
                        (select change
                        from sys.stats$stats
                        where name = 'write complete waits'),
                rollback_header_gets =
                        (select sum(trans_tbl_gets)
                        from sys.stats$roll),
                rollback_header_waits =
                        (select sum(trans_tbl_waits)
                        from sys.stats$roll)
where collection_ended is null;

insert into latches
(ls_latch_name, ls_latch_gets, ls_latch_misses,
        ls_latch_sleeps, ls_latch_immed_gets,
ls_latch_immed_misses)
select name, gets, misses, sleeps, immed_gets, immed_miss
from sys.stats$latches;

update latches set
        ls_collection_started =
                        (select min(stats_gather_times)
                        from sys.stats$dates)
where ls_oracle_sid is null;

update latches set
        run_date = to_date(substr(ls_collection_started,
                1,12),'DD-MON-YY HH24')
where ls_oracle_sid is null;

update latches
set ls_oracle_sid =
                        (select '&1'
                        from sys.dual),
        ls_collection_ended =
                        (select max(stats_gather_times)
                        from sys.stats$dates)
where ls_oracle_sid is null;
```

2. This routine interrogates all tablespaces and dumps the information into a statistical table.

```
insert into tablespace_stat values (
select dfs.tablespace_name,
round(sum(dfs.bytes)/1048576,2),
round(max(dfs.bytes)/1048576,2)
```

```
from  sys.dba_free_space dfs
group by dfs.tablespace_name
order by dfs.tablespace_name);
```

3. Attach this script to a cron process to gather table-extent information at a specified time interval.

```
insert into tab_stat values(
select ds.tablespace_name,
dt.owner,
dt.table_name,
ds.bytes/1024,
ds.extents,
dt.max_extents,
dt.initial_extent/1024,
dt.next_extent/1024,
dt.pct_increase,
dt.pct_free,
dt.pct_used
from sys.dba_segments ds,
sys.dba_tables dt
where ds.tablespace_name = dt.tablespace_name
and ds.owner = dt.owner
and ds.segment_name = dt.table_name
order by 1,2,3);
```

4. This table-extents-report script joins the extents table against itself to show growth in extents.

```
break on c0 skip 2 on c1 skip 1
ttitle " Table Report| > 50 Extents or new extents";
spool /tmp/rpt10
select
distinct
b.sid c0,
substr(b.owner,1,6) c1,
substr(b.tablespace_name,1,10) c2,
substr(b.table_name,1,20) c3,
(b.blocks_alloc*2048)/1024 c4,
c.next_extent/1024 c5,
a.extents c6,
b.extents c7
from tab_stat a,
tab_stat b,
dba_tables c
where
rtrim(c.table_name) = rtrim(b.table_name)
```

```
and
a.sid = b.sid
and
  rtrim(a.tablespace_name) <> 'SYSTEM'
and
a.tablespace_name = b.tablespace_name
and
a.table_name = b.table_name
and
to_char(a.run_date) = to_char(b.run_date-7)
-- compare to one week prior
and
(
a.extents < b.extents
-- where extents has increased
or
b.extents > 50
)
order by b.sid;
```

5. Use this script to get a fast overview of the state of a troubled system.

```
spool /tmp/snap;

prompt*************************************************
prompt Hit Ratio Section
prompt*************************************************
prompt
prompt ========================
prompt BUFFER HIT RATIO
prompt ========================
prompt (should be > 70, else increase db_block_buffers in init.ora)

--select trunc((1-(sum(decode(name,'physical reads',value,0))/
--(sum(decode(name,'db block gets',value,0))+
--(sum(decode(name,'consistent gets',value,0)))))
-- )* 100) "Buffer Hit Ratio"
--from v$sysstat;

column "logical_reads" format 99,999,999,999
column "phys_reads" format 999,999,999
column "phy_writes" format 999,999,999
select a.value + b.value "logical_reads",
c.value  "phys_reads",
d.value  "phy_writes",
round(100 * ((a.value+b.value)-c.value) /
(a.value+b.value))
```

```
"BUFFER HIT RATIO"
from v$sysstat a, v$sysstat b, v$sysstat c, v$sysstat d
where
a.statistic# = 37
and
b.statistic# = 38
and
c.statistic# = 39
and
d.statistic# = 40;

prompt
prompt
prompt ========================
prompt DATA DICT HIT RATIO
prompt ========================
prompt (should be higher than 90 else increase shared_pool_size in init.ora)
prompt

column "Data Dict. Gets"    format 999,999,999
column "Data Dict. cache misses" format 999,999,999
select sum(gets) "Data Dict. Gets",
sum(getmisses) "Data Dict. cache misses",
trunc((1-(sum(getmisses)/sum(gets)))*100) "DATA DICT CACHE HIT
RATIO"
from v$rowcache;
prompt
prompt ========================
prompt LIBRARY CACHE MISS RATIO
prompt ========================
prompt (If > .1, i.e., more than 1% of the pins
prompt  resulted in reloads, then increase the shared_pool_size in init.ora)
column "LIBRARY CACHE MISS RATIO" format 99.9999
column "executions" format 999,999,999
column "Cache misses while executing" format 999,999,999
select sum(pins) "executions", sum(reloads) "Cache misses while executing",
(((sum(reloads)/sum(pins)))) "LIBRARY CACHE MISS RATIO"
from v$librarycache;

prompt
prompt ========================
prompt Library Cache Section
prompt ========================
prompt hit ratio should be > 70, and pin ratio > 70 ...
prompt

column "reloads" format 999,999,999
```

```
select namespace, trunc(gethitratio * 100) "Hit ratio",
trunc(pinhitratio * 100) "pin hit ratio", reloads "reloads"
from v$librarycache;
prompt
prompt
prompt ========================
prompt REDO LOG BUFFER
prompt ========================
prompt
set heading off
column value format 999,999,999
select substr(name,1,30),
value
from v$sysstat where name = 'redo log space requests';

set heading on
prompt
prompt
prompt

column bytes format 999,999,999
select name, bytes from v$sgastat where name = 'free memory';

prompt
prompt*************************************************
prompt SQL Summary Section
prompt*************************************************
prompt
column "Tot SQL run since startup" format 999,999,999
column "SQL executing now"  format 999,999,999
select sum(executions) "Tot SQL run since startup",
sum(users_executing) "SQL executing now"
from v$sqlarea;

prompt
prompt
prompt*************************************************
prompt Lock Section
prompt*************************************************
prompt
prompt ========================
prompt SYSTEM-WIDE LOCKS - all requests for locks or latches
prompt ========================
prompt
select substr(username,1,12) "User",
substr(lock_type,1,18) "Lock Type",
substr(mode_held,1,18) "Mode Held"
```

```
from sys.dba_lock a, v$session b
where lock_type not in ('Media Recovery','Redo Thread')
and a.session_id = b.sid;
prompt
prompt ==========================
prompt DDL LOCKS - These are usually triggers or other DDL
prompt ==========================
prompt
select substr(username,1,12) "User",
substr(owner,1,8) "Owner",
substr(name,1,15) "Name",
substr(a.type,1,20) "Type",
substr(mode_held,1,11) "Mode held"
from sys.dba_ddl_locks a, v$session b
where a.session_id = b.sid;

prompt
prompt ==========================
prompt DML LOCKS - These are table and row locks...
prompt ==========================
prompt
select substr(username,1,12) "User",
substr(owner,1,8) "Owner",
substr(name,1,20) "Name",
substr(mode_held,1,21) "Mode held"
from sys.dba_dml_locks a, v$session b
where a.session_id = b.sid;

prompt
prompt
prompt*****************************************************
prompt Latch Section
prompt*****************************************************
prompt if miss_ratio or immediate_miss_ratio > 1 then latch
prompt contention exists, decrease LOG_SMALL_ENTRY_MAX_SIZE in init.ora
prompt
column "miss_ratio" format .99
column "immediate_miss_ratio" format .99
select substr(l.name,1,30) name,
(misses/(gets+.001))*100 "miss_ratio",
(immediate_misses/(immediate_gets+.001))*100
"immediate_miss_ratio"
from v$latch l, v$latchname ln
where l.latch# = ln.latch#
and (
(misses/(gets+.001))*100 > .2
or
```

```
(immediate_misses/(immediate_gets+.001))*100 > .2
)
order by l.name;

prompt
prompt
prompt******************************************************
prompt Rollback Segment Section
prompt******************************************************
prompt if any count below is > 1% of the total number of requests for data
prompt then more rollback segments are needed
 --column count format 999,999,999
select class, count
from v$waitstat
where class in ('free list','system undo header','system undo block',
'undo header','undo block')
group by class,count;

column "Tot # of Requests for Data" format 999,999,999
select sum(value) "Tot # of Requests for Data" from v$sys stat where
name in ('db block gets', 'consistent gets');
prompt
prompt ========================
prompt ROLLBACK SEGMENT CONTENTION
prompt ========================
prompt
prompt   If any ratio is > .01 then more rollback segments are needed

column "Ratio" format 99.99999
select name, waits, gets, waits/gets "Ratio"
from v$rollstat a, v$rollname b
where a.usn = b.usn;

column "total_waits" format 999,999,999
column "total_timeouts" format 999,999,999
prompt
prompt
set feedback on;
prompt******************************************************
prompt Session Event Section
prompt******************************************************
prompt if average-wait > 0 then contention exists
prompt
select substr(event,1,30) event,
total_waits, total_timeouts, average_wait
 from v$session_event
where average_wait > 0 ;
```

```
--or total_timeouts > 0;

prompt
prompt
prompt**************************************************
prompt Queue Section
prompt**************************************************
prompt average wait for queues should be near zero ...
prompt
column "totalq" format 999,999,999
column "# queued" format 999,999,999
select paddr, type "Queue type", queued "# queued", wait, totalq,
decode(totalq,0,0,wait/totalq) "AVG WAIT" from v$queue;

set feedback on;
prompt
prompt
--prompt**************************************************
--prompt Multi-threaded Server Section
--prompt**************************************************
--prompt
--prompt If the following number is > 1
--prompt then increase MTS_MAX_SERVERS parm in init.ora
--prompt
-- select decode( totalq, 0, 'No Requests',
--    wait/totalq || ' hundredths of seconds')
--    "Avg wait per request queue"
-- from v$queue
-- where type = 'COMMON';

--prompt
--prompt If the following number increases, consider adding dispatcher processes

--prompt
-- select decode( sum(totalq), 0, 'No Responses',
-- sum(wait)/sum(totalq) || ' hundredths of seconds')
-- "Avg wait per response queue"
-- from v$queue q, v$dispatcher d
-- where q.type = 'DISPATCHER'
-- and q.paddr = d.paddr;

--set feedback off;
--prompt
--prompt
--prompt ========================
--prompt DISPATCHER USAGE
--prompt ========================
```

```
--prompt (If Time Busy > 50, then change
MTS_MAX_DISPATCHERS in init.ora)
--column "Time Busy" format 999,999.999
--column busy  format 999,999,999
--column idle  format 999,999,999
--select name, status, idle, busy,
-- (busy/(busy+idle))*100 "Time Busy"
--from v$dispatcher;

--prompt
--prompt
--select count(*) "Shared Server Processes"
-- from v$shared_server
-- where status = 'QUIT';

--prompt
--prompt
--prompt high-water mark for the multi-threaded server
--prompt

--select * from v$mts;

--prompt
--prompt**************************************************
--prompt file i/o should be evenly distributed across drives.
--prompt

--select
--substr(a.file#,1,2) "#",
--substr(a.name,1,30) "Name",
--a.status,
--a.bytes,
--b.phyrds,
--b.phywrts
--from v$datafile a, v$filestat b
--where a.file# = b.file#;

--select substr(name,1,55) system_statistic, value
-- from v$sysstat
-- order by name;

spool off;
[px009u]: 99 99 52
```