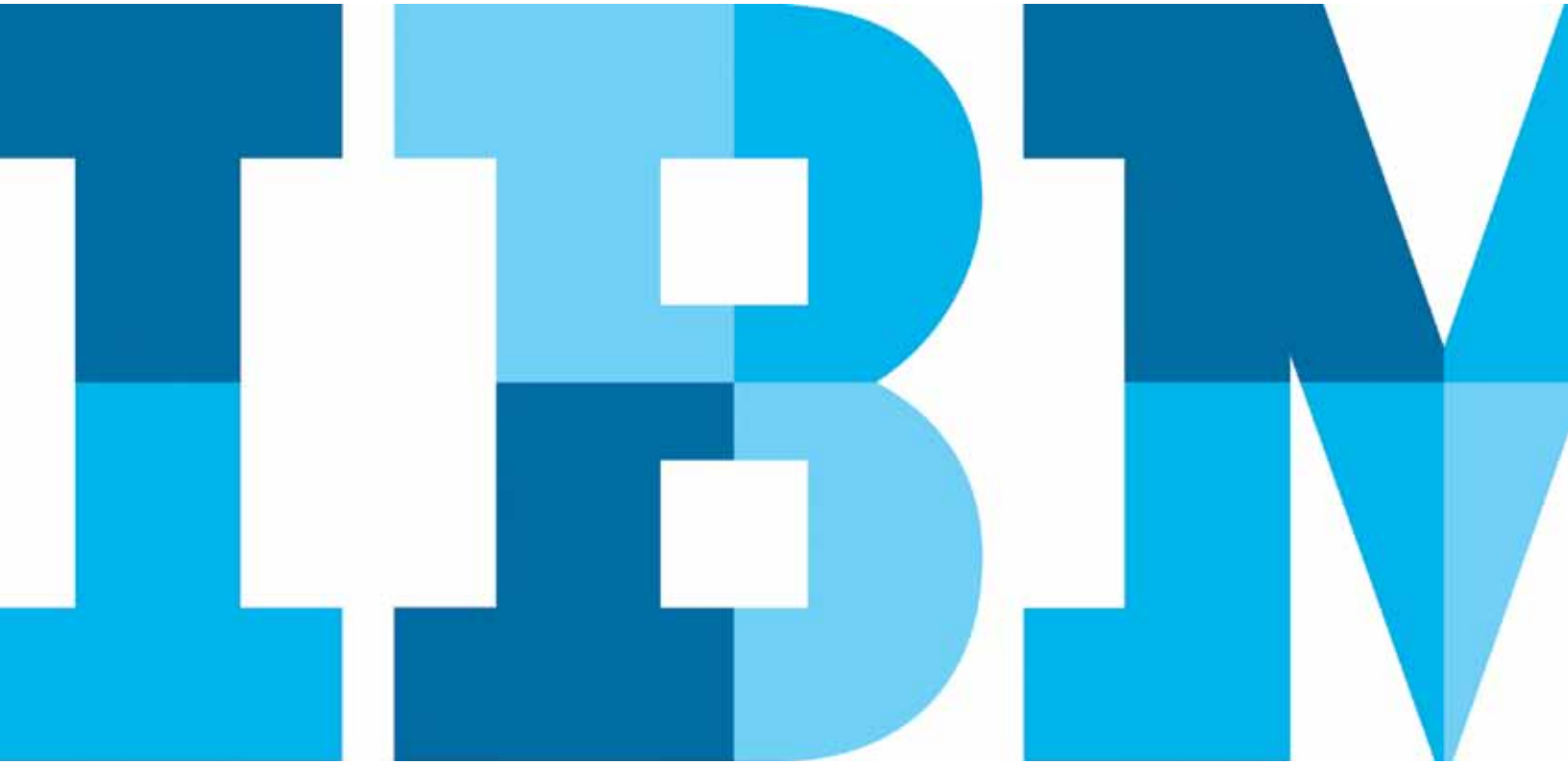
**IBM Systems and Technology Group**
Technical White Paper

April 2013

# Quick Oracle Statspack and AWR report analysis



IBM

## Introduction

I just sat back from my screen and realized in the last quarter I had analyzed 131 Automated Workload Repository (AWR) or Statspack results. In my job as Oracle Flash Storage Consulting Manager I analyze clients AWR or Statspack reports to see if flash storage (specifically IBM® FlashSystem™) will help them. I also manage a Statspack and AWR analysis website. Most of the time the issues on the website are format related (the site only takes text versions of the reports, not HTML or HTML that has been converted into text). The site also cannot handle reports that have any corruption issues such as the first byte of each line being garbled by whatever transfer protocol is used (typified by underscores being replaced by "~" when you look at the report). Of course I have my doubts that everyone actually tries hard to get their report to analyze, preferring for some reason to have me look at it personally! This paper tries to encapsulate in just a few pages the key things I examine with looking at an AWR or Statspack report.

## Analysis methodology

Now, 20 reports in one and half days are quite a few I realize but I have come up with methods to rapidly get to the heart of AWR or Statspack issues. Once I have the issues I depend on three things to put together recommendations for the users:

1. Morethan 20 years of Oracle experience
2. The documentation (http://tahiti.oracle.com)
3. Google and various online forums and boards

I am afraid there is no way to quickly give you item 1 of the list, but items 2 and 3 are available to everyone. You would be surprised to see the number of issues that repeat over and over again, but then again, maybe you would not! Anyway, here is my quick method of AWR or Statspack analysis.

1. Review header information looking at Oracle® release data, the duration of the report period, number of sessions, is this real application clusters (RAC) or not, various memory area sizing and block size. Oracle release level can highlight specific release related bugs and issues. Generally long duration reports will tend to average out issues and you may need to request a report specific

to a problem time period. Number of sessions is used with later analysis to determine overall system load. RAC determines if you need to review interconnect issues. Also, only one node's report from a 2 - 4 node system may be useless unless the actual problem is occurring on that node. Memory sizes such as DB cache size and shared pool size can quickly pinpoint issues. For example, if the shared pool is larger than the DB cache size then there are probably bind variable or versioning issues in the database.

2. Review the load profile section next. I pay most attention to physical reads, physical writes, hard parse to parse ratio and executes to transaction ratio. The sum of physical reads and writes will tell you the amount of block IOPS, then multiply times block size and you have a quick throughput estimate. The ratio of hard parses to parses tells you how often SQL is being fully parsed. Full parsing of SQL statements has a negative effect on performance. High hard parse ratios (>2 - 3 percent) indicate probable bind variable issues or maybe versioning problems. Rows per sort can also be reviewed here to see if large sorts are occurring.
3. Review the Instance Efficiencies section next. The hit rates for the cache and shared pool, Execute to Parse, Parse CPU to Parse Elapsed, In Memory Sort and Soft Parse percents are what I look at here. The first two being low indicate possible cache or shared pool issues, the next two, bind variable and versioning issues. In Memory Sort being low (in the high 90s or lower) indicates `PGA_AGGREGATE_TARGET` or `SORT_AREA_SIZE` issues and Soft Parsing being low ties in with the first two indicating bind variable and versioning issues. It only takes a few percent of non-memory sorts to drive the temporary tablespace to the top of the IO list.
4. Next look at the Shared Pool Statistics. If your usage is low (<85 percent) then your shared pool is over sized. If your reuse is low (<60 - 70 percent) you may have bind variable or versioning issues. Ideally all the percentages in this area of the report should be as high (close to 100) as possible.
5. The next section, the Top Five Wait Events, is arguably the most important part of the report. Using the wait events highlighted here you can quickly find where issues are occurring, allowing you to zoom into other areas of the report. Here are a few hot buttons:
    a. Db file sequential reads: Usually indicates memory starvation, look at the db cache analysis and for buffer busy waits along with cache latch issues.
    b. Db file scattered reads: Usually indicates excessive full table scans, look at the AWR segment statistics for tables that are fully scanned. If this is a Statspack and you are at or above Oracle9i, look at the `v$sql_plan` table to see where the full table scans or full index scans are happening, here is an example script to look at the full table scans.

```
rem fts report
rem based on V$SQL_PLAN table
col operation format a13
col object_name format a32
col object_owner format a10
col options format a15
col executions format 999,999,999
column dt new_value today noprint
select to_char(sysdate,'ddmonyyyyhh24miss') dt from dual;
set pages 55 lines 132 trims on
ttitle 'Full Table/Index Scans'
spool fts&&today
select
a.hash_value,a.object_owner,a.object_name, rtrim(a.operation) operation,
a.options, sum(b.executions) executions, c.bytes, (sum(b.executions)*c.bytes)/(1024*1024)
fts_meg
from
v$sql_Plan a, v$sqlarea b, dba_segments c
where
(a.object_owner=c.owner
and a.object_name=c.segment_name)
and a.address=b.address
and a.operation IN ('TABLE ACCESS','INDEX')
and nvl(a.options,'NULL') in ('FULL','NULL','FULL SCAN')
and a.object_owner not in ('SYS','SYSTEM','PERFSTAT','SYSMAN','WKSYS','DBSNMP')
and b.executions>1
group by a.hash_value,a.object_owner, a.object_name, operation, options, c.bytes
order by a.object_owner,a.object_name,operation, options, executions desc
/
spool off
set pages 20
ttitle off
```

c. Log file related waits: Look at excessive log switches, excessive commits or slow IO subsystems.
d. PX DEQ Send Blkd: This can indicate that the parallel execution message size is too small and can also indicate that you have a parallel feed going into a single process DML statement such as a `INSERT`, `UPDATE` or `DELETE`.
e. Direct path read/write to temp: Shows excessive `sorting/hashing/global temp table/bitmap` activity going to your temporary tablespace. Review `PGA_AGGREGATE_TARGET` settings. Even if it looks like it is big enough, if you are getting multiple small sorts to disk it could mean your user load is over-utilizing it.

6. I do not usually give more than a passing glance to the time model statistics. I do look at the Wait Class breakdown, paying attention to what wait classes are at the top and using them to help further define issues. Usually it is `USER IO`.
7. The detailed wait lists are only of interest if the top five indicate that further waits may be a problem, otherwise I just skip over them.
8. Next I look at the Operating System Statistics. I look at busy versus idle versus IO wait to see if IO wait is an issue. However, on many OS platforms the IO wait may not be populated or is not populated properly so this cannot be the only place you watch this. Also included in this section is a list showing the various OS statistics (load, busy, idle, IO wait, etc) for each AWR period included in the report. Use this list to pinpoint times when load or IO wait is the highest and generate additional AWR reports of those periods.
9. The Service Statistics and Service Wait Class Stats are only of interest if you do not know who is using resources. You can sum the IO waits for each service and the amount of time waited to get a good idea of total IO wait if the Operating Statistics area is not accurate as far as IO wait time.
10. The various SQL areas allow you to get the top SQL statements. If the same statement is appearing in two or more of the report sections in the top 5 - 10 then it is probably a statement of interest. Far more interesting is the complete SQL listing area. In the complete SQL listing you can verify that bind variables either are or are not being used properly. Another SQL report is the SQL Versions report. If you see large numbers of statements with high version counts (>100 versions) then this is probably at least part of the source of a large shared pool. Look at using FORCE instead of SIMILAR for `CURSOR_SHARING` or set the undocumented parameter "`_SQLEXEC_PROGRESSION_COST`" to its highest value. I also examine large SQL statements for the proper number of join paths (N-1 where N is the number of tables) and for total number of tables in the joins (<7 is optimal since the `_OPTIMIZER_MAX_PERMUTATIONS` is set to 2000 which falls between N!6 and N!7). If there are more than six tables, make sure the most important one is first in the `FROM` clause.

11. The Instance Activity Stats are next, pages of them. Only a few of the Instance Activity Stats need to be reviewed in a quick analysis. Now for a detailed analysis for the root cause of particular problems, you may want to go back and look at specific numbers, but during a quick analysis you are interested in "low hanging fruit" only. Here is a list of what I look at:
    a. SQL*Net Round Trips from Client: Too many per transaction indicate that array passing for results is not being used efficiently.
    b. SQL*Net Round Trips from DBLink: See above. Also look at pushing the work to the remote host and only getting back results using hints.
    c. Index Fast Full Scans (full): Use this with Table Scan statistics to see where the majority of db file scattered reads are occurring.
    d. Index fetch by key: Use to see efficiency of indexing.
    e. Parse count (hard) and Parse count (total): Use to verify bind variable and versioning issues.
    f. Physical read total `io` requests + Physical write total `io` requests: Use to determine physical IOPS by calculating a ratio of block IOPS to physical IOPS. You can see if the system is weighted towards full table scan or single block look up. Use session count and physical IOPS to help determine if the disk farm can handle the load based on IOPS and concurrency needs.
    g. Sorts (disk): If present and excessive may indicate insufficient `PGA_AGGREGATE_TARGET or SORT_AREA_SIZE` settings.
    h. SQL Area evicted: May indicate excessive object editing.
    i. SQL area purged: May indicate too small a shared pool or bind/versioning issues.
    j. Summed dirty queue length or write requests: If excessive indicates you need more `DB_WRITER_PROCESSES`. This is platform dependent.
    k. Table fetch by rowid and Table scan rows gotten: In an online transaction performance (OLTP) system rowid fetches should be greater than scan rows, in a data warehouse (DWH) or OLAP scan rows should be greater than rowid fetches. If scan rows is way too high then look at lack of indexing issues.
    l. Table fetch continued rows: May indicate chained rows, however, is also affected by LOBs.
    m. Transaction rollbacks: Used to see if rollbacks are excessive. This ties in with excessive UNDO tablespace IO.
    n. User IO wait time: If the IOwait is not populated on your system, this can provide needed data.
    o. Workarea executions one pass or multi-pass: Indicates `PGA_AGGREGATE_TARGET` is not set correctly.

12. Instance Activity Stats (Absolute Values): These are interesting to look at but not really used in a quick look. May help with `PGA_AGGREGATE_TARGET` sizing.

13. Instance Activity Stats (Thread Activity): This is useful to see redo log activity. The old rule of thumb is four switches per hour which is about right. I have not seen a need (except in Dataguard maybe) to have this more frequent. Less frequent, as long as it is not ridiculous, is ok. If logs are switching too often, increase the physical redo log size.
14. In Oracle11g, version 11.2, in the AWR, they have added an IO Stats section that breaks down all the IO in the database into various areas. This is a good place to see where your IOs are generated and to get IOPS numbers for read and write activity.
15. Tablespace and File IO statistics: These are useful to see what your hot tablespaces are. For example, having the SYSTEM tablespace as the number one source of IO could indicate you have improper temporary tablespace assignments as these used to default to SYSTEM. Having the TEMP or UNDO tablespaces in the top position has already been discussed. Usually in an OLTP system one of your index tablespaces should be at the top. In a DWH or OLAP a data tablespace should be at the top. Also look at the latency values. For disk based systems 5.0 ms is considered good performance. If you are getting >10 ms then you are having queuing issues and need to review your IO subsystem. If your database release is <10 then you can get IOPS by summing the reads and writes per second from this section.
16. Immediately after the tablespace statistics is a small area (usually one line-worth unless you are using `KEEP`, `RECYCLE` or multiple block size pools), called Buffer Pool Statistics. Look at buffer busy waits. If these are high, you are either IO bound or have insufficient buffers in one or more of your buffer pools.
17. Instance recovery stats I usually do not look at as they are not performance related. Next I look at the Buffer Pool Advisory section. This section can be confusing, because if your buffer pool is too small, it may indicate it is ok. If you are seeing high buffer busy waits and lots of cache buffer latch issues, but the advisor says everything is fine, do not believe it! Most of the time I just look at the very end of the section to see what the effect of doubling the cache size would be. If it is significant (>20 percent reduction in physical IO) I suggest increasing the pool. However, even if it is not significant but I see lots of buffer busy waits and cache buffer latch issues, I will suggest gradually increasing the pool in 20 percent increments to see if buffer busy waits decrease.
18. In the PGA Aggr sections I look to see if a significant hit rate is being seen, as this would indicate that the setting for the `PGA_AGGREGATE_TARGET` is too small. I also look at the histogram section to see where the majority of single or multi-pass sorting is happening. If I am getting single or multi-pass sorts I take the higher of the histogram range settings and multiply it by 40 and then times one-tenth the number of reported sessions. This should be a reasonable estimate of the needed `PGA_AGGREGATE_TARGET` setting. If the setting is already higher than this number, then your users are doing more sorts than average and the setting should

be increased in 20 percent intervals until sorting to disk is under control. Generally speaking the advisory section in the PGA area is useless. Note: If `DISPATCHERS` or `SHARED_SERVERS` are set, then some parallel processes will not use `PGA_AGGREGATE_TARGET` but will instead default to the old `SORT_AREA_SIZE` parameter, if you see loads of sorts under 512 megabytes, this could be the cause, either turn of `DISPATCHERS` or `SHARED_SERVERS` or set `SHARED_POOL_SIZE`.

19. I also find the shared pool advisory generally useless as it always seems to indicate the current setting is just great. Look at the SQL purges and the use/reuse ratios to see if the shared pool is being properly used. If you consistently run at 90 percent or greater usage then you may need to increase the shared pool size.
20. The SGA Target and Streams pool advisories are also generally useless. Unless you are really hitting your steams pool hard, the default size is good. If you start seeing disk spills, increase the size. I have also never seen the Java Pool advisory section give any useful information.
21. The Buffer Wait Statistics help give you an indication of where the buffer waits you noted at the beginning of the Buffer Statistics section are coming from. In the majority of cases these are going to be data block waits, which indicate either your IO subsystem is too slow or you have insufficient buffers, or both!
22. In the Enqueue Waits area, look at the top few enqueues. Usually they will be TX type enqueues, indicating issues with the application such as row locking or non-indexed foreign keys. In RAC you may see SQ enqueues if sequence caching is not set properly.
23. Most times I do not see anything in the UNDO statistics that warrants attention. However, in systems where you see values in the `uS/uR/uU/ eS/eR/eU` column other than zeroes, you may want to dig deeper into how they are using commits and rollbacks and look at the sizing of the undo tablespaces and setting of the "`_transactions_per_rollback_segment`" parameter to tune usage. Also parameters dealing with undo retention such as `UNDO_RETENTION` should be verified.
24. I use the latch sections to confirm if there are buffer or shared pool issues. On a quick run through, you are not going to do detailed latch analysis unless you get some sort of odd wait event that shows up in the top five and requires more detailed analysis. Latch free and cache buffer latches will probably be the dominant latches. Pay attention to latches that have excessive sleeps or anything in the sleep1, 2 or 3 columns.

25. The Segment Statistics is a very important part of the report. Review segments that have excessive reads, excessive full scans and any excessive buffer busy waits. In a RAC environment the ones that have excessive CR or current waits might benefit from a smaller block size tablespace. Excessive reads could indicate insufficient indexes.
26. The dictionary cache area I find rather useless since you cannot do anything about it other than increase the shared pool size. The Library Cache Activity area can be used to see SQL Area evictions and purges to determine if the pool is large enough.
27. The process areas I find useless most of the time. The SGA breakdown difference statistics can be helpful to determine if your settings for `SGA_MAX_SIZE` and `SGA_TARGET` are sufficient or if you are on 11g, if the settings for `MEMORY_MAX_SIZE` and `MEMORY_TARGET` are sufficient. Also look at the large pool free, shared pool free, streams pool free and java pool free settings to see if you need to increase these parameters. Look for lots of deferred actions. If your report shows the Resize actions and you see them, then those parameters probably are incorrectly sized. Look at the `v$sga_resize_ops` table to see what is happening as far as resize events in the SGA. Here is an SQL report to get an idea of the activity from `v$sga_resize ops`:

```
set lines 200
set pages 55
column inst_id format 999 heading 'Inst'
column component format a24 heading 'Component'
column oper_type heading 'Oper Type' format a12
column status format a9
column s_time format a12
column e_time format a12
column parameter format a21
column now new_value dt noprint
select to_char(sysdate,'yyyymmddhh24mi') now from dual;
ttitle 'Resize OPS'
spool resize_ops&&dt
select inst_id,component,oper_type,parameter,initial_size, target_size, final_size,status,
to_char(start_time,'yyyymmddhh24mi') s_time,to_char(end_time,'yyyymmddhh24mi') e_time
from gv$sga_resize_ops
order by 1,2,9
/
spool off
set lines 80 pages 20
clear columns
ttitle off
```

28. The streams, resource and remainder of the statistics sections usually are not useful unless you are looking for a particular issue with those areas.
29. The final area will be the init.ora parameters report. I usually look at the various pool and buffer sizes (if they are set). One thing I have been seeing more and more is the improper setting of the `SGA_MAX_SIZE` and `SGA_TARGET` or `MEMORY_MAX_SIZE` and `MEMORY_TARGET` settings on Oracle10g and Oracle11g. There are two settings for a reason; they are not supposed to be set to the same value. The `TARGET` parameters should be calculated based on a set of base parameters: `DB_CACHE_SIZE, SHARED_POOL_SIZE, JAVA_POOL_SIZE, STREAMS_POOL_SIZE`, `PGA_AGGREGATE_SIZE`. Set these to a setting that gives you minimal acceptable performance. Once either the `SGA_TARGET` (Oracle10g) or `MEMORY_TARGET` (in Oracle11g) calculates, set the `MAX_SIZE` parameter to 25 percent greater than the setting of the `TARGET` parameter. This gives adjustment room. If the two parameters are set equal to each other, then Oracle has to steal from one parameter when one of the others needs memory. Monitor the settings with the `v$sql_resize_ops` view. I then usually verify that the other parameters that are set in the `init.ora` are reasonable for their database and also look up any undocumented settings. (If I can! After all, they are undocumented!)

## Summary

I realize a 29 item list is rather large, but if you look at the items in the list, many of them have only a couple of things (or none) that you have to look at. I also did not dive deeply into the RAC extra sections that you will see if you have a RAC database. Generally in RAC you need to look at your current and CR block transfer times, if they are less than your disk latency then you are fine. If the interconnect latencies are greater than the disk latencies then look at the TCP buffer settings and perhaps use jumbo frames. As you get more experience, you will learn what items are in their “normal” range and what items set off an alarm bell and warrant further research or investigation. I hope this paper helps you in your analysis of AWR and Statspack reports.

# Notes