Statspack

S.K. Srivastava
DBA
Indorama Synthetics tbk, Indonesia

Abstract
Statspack is a performance diagnosis tool. It collects performance related information into its internal table and then later on this stored data can be analyzed with help of Report provided by the Statspack. The tool provides a systematic and effective approach of database tuning, which can save a lot of time of DBA.

Statspack

In order to tune Database, there could be following approaches

1- Complete Proactive
2- Proactive
3- Reactive

In Complete Proactive approach Application is designed in such a way such that no further tuning is needed. It is difficult to achieve this type of design, and practically most of the Application needs performance tuning.

Under Proactive approach database needs to be monitored at specified intervals even though users are not facing any performance issues, such that rectification of problem can be done at early stage itself.

Reactive approach is to resolve the issues when problems have already created serious impacts.

Statspack is ideally suited for Proactive approach and can be used for reactive approach also.

Statspack is a diagnosis tool, which indicates the area where tuning is needed.

Oracle keeps it's most of the performance related data in v$ views and these data gets updated continuously. In order to analyze the data it is required to capture one set of data and then on this static data analysis can be done. Statspack is basically a set of sql scripts, which can collect performance related data from these v$views at desired time intervals. The statspack procedure collects these data and stores it into statspack's internal table.

Statspack is available for release 8.1.6 and onwards. Statspack scripts are available for some earlier versions of Database also but for production purpose statspack is supported for release 8.1.6 and onwards.

Installation Procedure

There are few prerequisites for installation

1- Ensure that catalog.sql (sql script which is normally run after database creation) is already executed
2- Check for dbms_shared_pool package, if it is not there then dbmspool.sql needs to be run
3- It is preferred to create a separate tablespace for statspack, but it can be installed in existing tablespace also.

Statspack should be installed through sql*plus, there is detailed installation procedure in following file

ORACLE_HOME Dir → rdbms → admin → statspack.doc (name could be different for different releases)

Script name is changed from version to version that's why all the steps should be followed as per the document of statspack.

Alter installation it is very important to check output files for error (.lis files). These files are generated in the directory of statspack i.e. ORACLE_HOME → rdbms → admin

There are sql scripts, which can drop the statspack, so incase statspack is not needed then it can be dropped. During database upgradation there are few steps for upgradation of Statspack so there is no need of dropping and recreating Statspack objects during upgradation.

Collecting the performance data

Once statspack is installed properly, it will create one user named as statspack.
Statspack.snap is the procedure which needs to be run for collecting data from oracle's v$ views to statspack's internal tables.

Running the procedure is called taking a **snap.** Statspack gives unique id to each snap and stores corresponding data into internal tables with reference of snap id.

Snap execution can be automated for data collection with the help of dbms_job package

Customize the data collection
In order to control the amount of information during snap execution Oracle has provided some options.

Level and **threshold** are the two options, which can be customized as per database requirement.

Level – There could be four possible levels

Level:0 - Statspack collects general performance statistics such as wait statistics, system events etc.

Level:5 - collects Level 0 data and information on high-resource-usage sql. This level is the default level of statspack.

Level:10 – collects all the statistics of Level 0 , Level 5 and as well as child-latch information.

Level:6 – This is a new level added in 9i , it includes Level5 information , Execution path and plan for high resource sql statement.

Threshold – As high-resource sqls are collected during snap execution (not in Level 0). There are filters on the basis of which these sql are selected, value of these filters can be customized as per databas e size.

i_executions_th → Number of execution
i_disk_reads_th → Number of Disk reads
i_parse_calls_th → Number of Parse calls
i_buffer_gets_th → Number of buffer gets

Other Parameters – In addition to the above threshold there are other parameters.

i_snap_level → default snap level
i_ucomment → comment to be stored with statspack
i_session_id → sid , used for capturing details of a particular sid
i_modify_parameter → true/false , save the change in default table

Statspack stores the default values of the above parameter in stats$statspack_parameter table, which can be changed with following procedure

SQL> execute statspack.snap(i_snap_level => 10 , i_modify_parameter => 'true');

Others alternative for changing default values are

SQL > execute statspack.modify_statspack_parameter(i_buffer_gets_th => 20000 , i_disk_reads_th=>1000);

Running the Report

In statspack family there is one sql procedure, which generates report on the basis of any two snap id's data.

It is run from sql logging as user perfstat

SQL> @statsrep.sql

System asks for start snap id, end snap id and repot output file name.
Report is generated on the basis of data set stored against the snap id given in the parameter.

The report generated from statspack is real output on which analysis is done.

If there is database shutdown/startup between the two snaps then report will not show the correct picture. As information stored in v$ views gets reset when database is shutdown/startup and which is the base for statspack data collection that's why resulting information used in the comparison will show incorrect information.

In 9i, there are two additional reports, one is for SQL statistics for a particular sql and second one is for Real Application cluster.

Analyzing the Report

The output of the Statspack is the Report and analysis of the Report is the key of performance tuning using Statspack.

Report is divided into two sections. First section of the Report is summary page and next section shows detail level information.

Summary Information

```
STATSPACK report for

DB Name         DB Id     Instance    Inst Num  Release     OPS   Host
----------  -----------  ----------  --------  ----------  ----  ----------
IRS          2224803745  irs                1  8.1.6.3.0   NO    indorama

                                                            Snap Length
Start Id    End Id       Start Time             End Time      (Minutes)
--------  --------  --------------------  --------------------  -----------
     137       138  28-Feb-03 10:33:38    28-Feb-03 15:45:58         312.33
```

The above section of the report shows the Database name, instance name, host name etc.
It also gives the info of start time of snap and end time of snap.

Cache Size Information

```
Cache Sizes
~~~~~~~~~~~
           db_block_buffers:      120000
              db_block_size:        8192
                 log_buffer:    10485760
           shared_pool_size:  1153433600
```

This section shows the size of db buffer cache, shared pool size and log buffer size.
There could be multiple db buffers, but report shows only standard (default) db buffer information.

Load Pofile

```
Load Profile
~~~~~~~~~~~~
                                    Per Second      Per Transaction
                                   ---------------  ---------------
                  Redo size:          70,994.28        23,411.16
              Logical reads:          25,152.19         8,294.22
              Block changes:             505.22           166.60
             Physical reads:           3,436.92         1,133.36
            Physical writes:              57.56            18.98
                 User calls:             350.43           115.56
                     Parses:              76.59            25.26
                Hard parses:               0.26             0.09
                      Sorts:              82.48            27.20
               Transactions:               3.03

             Rows per Sort:       28.56
  Pct Blocks changed / Read:        2.01
        Recursive Call Pct:       75.38
 Rollback / transaction Pct:       15.76
```

This is the summary of database load between the two snaps.
In order to check any abnormal load baseline data should b e available.

Redo size – amount of Redo generation
Logical Reads – It is consistent Gets + DB block Gets
Block changes – Number of Blocks modified
Physical Reads – Number of request for a block, which needed physical I/O
Physical Writes – Number of Physical writes
User Calls – Number of queries generated

Instance Efficiency

```
Instance Efficiency Percentages (Target 100%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        Buffer Nowait Ratio:       99.70
        Buffer  Hit   Ratio:       86.34
        Library Hit   Ratio:       99.98
        Redo   NoWait Ratio:      100.00
       In-memory Sort Ratio:      100.00
           Soft Parse Ratio:       99.66
            Latch Hit Ratio:       99.68
```

These are the basically hit ratios, its values should be monitored to identify database trends. Tuning should not be done on the basis of these ratios.

Top 5 wait events

```
Top 5 Wait Events
~~~~~~~~~~~~~~~~~                                                 Wait     % Total
Event                                            Waits  Time (cs)   Wt Time
-------------------------------------------- ------------ ------------ -------
db file scattered read                        11,660,979    3,809,159   26.88
db file sequential read                       11,522,156    3,517,787   24.82
wakeup time manager                                  618    1,803,483   12.72
db file parallel write                            21,906      953,115    6.72
buffer busy waits                              1,418,206      934,099    6.59
         -------------------------------------------------------------
```

This is the key information of the Report as it indicates about the events for which database was waiting.
On the basis of this information further analysis of report should be done. For e.g. if db file wait ratios are there so details I/O section should be checked.
db file sequential and scattered read should be top wait events for a tuned instance.

For performance diagnosis, it is always suggested to set init ora's parameter **timed_statistics** to true. Incase it is set to false then wait events will be ordered by number of wait and not by the wait time.

Detail Information

This section provides details of the different areas of performance tuning.
Selected key areas are described below -

Foreground Wait Events for DB

Event	Waits	Timeouts	Total Wait Time (cs)	Avg wait (ms)	Waits /txn
db file scattered read	11,660,979	0	3,809,159	3	205.2
db file sequential read	11,522,156	0	3,517,787	3	202.8
wakeup time manager	618	591	1,803,483	#####	0.0
db file parallel write	21,906	0	953,115	435	0.4
buffer busy waits	1,418,206	1,246	934,099	7	25.0
log file sync	51,706	1,394	776,515	150	0.9
direct path read	250,223	0	451,488	18	4.4
PX Deq Credit: send blkd	1,365,167	151	381,677	3	24.0
SQL*Net message from dblink	2,997	0	317,940	1061	0.1
db file parallel read	47,668	0	206,313	43	0.8
log file parallel write	51,321	0	201,717	39	0.9
SQL*Net more data to client	267,133	0	146,081	5	4.7
direct path write	14,056	0	127,238	91	0.2
SQL*Net more data from dblin	2,313	0	99,437	430	0.0
PX Deq: Execute Reply	4,479	401	98,166	219	0.1

……..
…….. cont……..

These are the wait events associated with session process

Background Wait Events for DB

Event	Waits	Timeouts	Total Wait Time (cs)	Avg wait (ms)	Waits /txn
db file parallel write	21,906	0	953,115	435	0.4
log file parallel write	51,321	0	201,715	39	0.9
control file parallel write	6,280	0	68,821	110	0.1
direct path read	14,056	0	15,751	11	0.2
db file scattered read	7,520	0	8,483	11	0.1
log file sequential read	43,236	0	7,138	2	0.8
file open	23,415	0	2,092	1	0.4
db file sequential read	1,004	0	1,766	18	0.0

….. cont….

Background events are associated with system Background process and some non-system background process. System background process are Database Writer, Log writer etc. where as no-system background processes are parallel query slave etc.

Most of the idle wait events can be safely ignored from wait analysis.

SQL Information

```
* SQL ordered by Gets for DB
* SQL ordered by Reads for DB
* SQL ordered by Rows for DB
```

The above listing is applicable for 8i, in case of 9i one more additional listing of **SQL Ordered by Parse call** is also there.

Instance Activity Stats for DB

Statistic	Total	per Second	per Trans
CPU used by this session	9,557,701	510.0	168.2
CPU used when call started	9,555,560	509.9	168.2
CR blocks created	105,700	5.6	1.9
DBWR buffers scanned	4,318,888	230.5	76.0
DBWR checkpoint buffers written	161,688	8.6	2.9
DBWR checkpoints	14	0.0	0.0
DBWR free buffers found	4,156,606	221.8	73.1
DBWR lru scans	20,361	1.1	0.4
DBWR make free requests	23,475	1.3	0.4
DBWR revisited being-written buff	875	0.1	0.0

…. **cont** …..

This section shows the overall statistics of Database.

Tablespace IO Summary for DB

Tablespace	Reads	Avg Read (ms)	Writes	Total Wait Waits	Avg Wait (ms)
ERP_DAT1	14,815,500	2.4	20,615	1,249,138	4.2
ERP_DAT2	7,900,152	3.2	37,293	162,923	24.9
ERP_IDX1	578,145	14.1	31,470	311	72.7
ERP_IDX2	279,487	25.8	174,840	4,079	10.4
ONTD	207,535	6.9	963	985	5.3
ROLL_SEG	2,017	19.6	79,573	647	6.8
SYSTEM	60,057	18.6	1,580	69	68.6
MAXIMO	33,009	7.5	64	0	0.0
DATA	11,329	10.7	39	2	10.0
ONTX	6,709	17.1	1,045	0	0.0

…… **cont** ….

File IO Statistics for DB

Tablespace	Filename				
	Reads	Avg Blks Rd	Avg Rd (ms)	Writes	Tot Waits	Avg Wait (ms)
ABMD	/data11/irsdata/abmd01.dbf					
	36	1.0	0.0	9	0	
ABMX	/data11/irsdata/abmx01.dbf					

```
            36         1.0         0.0              9          0

AMSD                     /data11/irsdata/amsd01.dbf
            63         1.0         0.0              9          0

AMSX                     /data11/irsdata/amsx01.dbf
            63         1.0         0.0              9          0
…. Cont ….
```

The above two section shows information related to I/O, incase I/O waits are one of the wait event in Top 5 waits then this I/O section should be checked carefully. Average read time should be less than 20 ms, any value more than 20 ms indicates need of tuning in the I/O area.

Buffer wait Statistics for DB

Class	Waits	Tot Wait Time (cs)	Avg Time (cs)
data block	1,417,442	935,961	1
undo header	156	327	2
undo block	491	111	0
segment header	10	12	1
bitmap block	7	1	0
extent map	3	0	0
bitmap index block	2	0	0

This section shows the waits of different type of object in Database buffer.

Rollback Segment Stats for DB

RBS No	Trans Table Gets	Pct Waits	Undo Bytes Written	Wraps	Shrinks	Extends
0	69.0	0.00	0	0	0	0
2	13,737.0	0.00	2,801,646	0	0	0
3	13,812.0	0.00	4,521,540	1	0	0
4	305,870.0	0.00	1,878,058	0	0	0
5	3,602.0	0.00	2,091,862	0	0	0
6	10,171.0	0.00	46,827,194	4	0	0
7	4,746.0	0.00	5,057,378	5	0	0

The above section indicates about wait for rollback segments. Incase Pct waits are non zero then tuning of Rollback segments is needed.

Latch Activity for DB

Latch Name	Get Requests	Pct Get Miss	Avg Sleeps /Miss	Nowait Requests	Pct Nowait Miss
Active checkpoint queue latch	79,362	0.0	0.0	0	
Checkpoint queue latch	3,797,254	0.0	0.2	0	
JOX SGA heap latch	483	0.0		0	
Token Manager	50,859	0.0	0.0	1,440	0.0
archive control	18	0.0		0	
archive process latch	18	0.0		0	
begin backup scn array	5,990	0.0		0	
cache buffer handles	1,080,587	0.0	0.0	0	

Pct Miss (Pct Get Miss, Pct Nowait Miss) indicates the latch contention, if it's value is more than 1% then corrective action is needed. Latch contention occurs due to contention of the resources.

Dictionary Cache Stats for DB

Cache	Get Requests	Pct Miss	Scan Requests	Pct Miss	Mod Req	Final Usage	Pct SGA
dc_constraints	78	52.6	0		78	83	99
dc_database_links	444	0.5	0		0	10	91
dc_files	1,413	0.0	0		0	60	97
dc_free_extents	8,315	3.9	43	0.0	355	2,222	79
dc_global_oids	1,243	0.0	0		0	32	86
dc_histogram_data	188	0.0	0		0	80	88
dc_histogram_data_valu	236	0.0	0		0	39	95

Library Cache Activity for DB

Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali-dations
BODY	341,954	0.0	340,327	0.0	0	0
CLUSTER	331	0.6	270	1.5	0	0
INDEX	14,248	0.1	14,248	0.1	0	0
OBJECT	0		0		0	0
PIPE	76,987	0.7	80,782	0.7	0	0
SQL AREA	540,121	0.4	18,025,231	0.0	2,672	197
TABLE/PROCEDURE	2,887,350	0.0	16,522,088	0.0	990	0
TRIGGER	4,068	0.2	4,068	0.5	12	0

Both the above section gives information of Shared pool. If Pct Miss is higher in case of Dictionary cache then only solution is to increase the shared pool.

Methodology

In Order to use the report baseline should be there. In order to assign a report as baseline it is important to understand the load pattern of Database.

Baseline should be assigned on the basis of load pattern of individual database.

Assume a database where on Monday and Friday load is relatively low and rest of the working days load is higher. One baseline can be assigned for Monday/Friday and other for rest of the working days. If at the time of month end, database load increases due to month end processing then one more set of report should be made as baseline for the month end period.

Once baseline is assigned then on daily basis or every alternate days snap should be executed and report should be generated. This report should be compared with assigned baseline for any abnormality.

Missing information from Statspack

Statspack provides a good amount of information from performance tuning point of view, but there are few more critical information, which are not covered in Statspack.

e.g. –

-Operating System Related data
-Details of wait events etc.

In order to get complete performance data, some additional scripts should also be written to incorporate the missing data.

References –

- Diagnostic Performance with Statspack – Connie Dialeris and Graham Wood
- Performance Tuning with Statspack – Connie Dialeris and Graham Wood
- Analyzing a Statspack Report – Connie Dialeris and Graham Wood
- Systemwide Tuning using Statspack Reports - Metalink Document

About the Author

S. K. Srivastava: Bachelor of Engineering from Indian school of Mines (1995).

Oracle Certified Professional (9i), 3.5 years Oracle Application DBA (along with Solaris Sys Admin), and 3 years on Oracle Apps Techno functional. I have been involved in the following projects:

- Oracle apps Migration (10.7SC to 11.5.3 (11i)), role Oracle apps DBA (along with Solaris Administration), Company - Indorama Synthetics Tbk, Indonesia
  - Oracle Application Migration from 10.7 SC to 11.5.3 (11i) on Solaris Platform
  - Server Sizing for 11i
  - Oracle Database Tuning
  - Solaris Installation and Patch Application
  - Veritas Reconfiguration for optimum I/O
  - Database backup (Hot, Cold and Export Backup) and testing recovery

- Oracle apps 10.7SC Administration, role Oracle apps DBA, Company - Indorama Synthetics Tbk, Indonesia
  - Oracle Apps 10.7 SC Maintenance
  - Database Tuning (Wait Events, Hit Ratios and tkprof) – optimization of init.ora parameters
  - Veritas Installation and Raid (0+1) implementation

- Database backup (Hot, Cold and Export) and testing Recovery
- Oracle Multi Threaded Server Implementation

- Oracle apps 10.7SC (AR – OE implementation), role–Oracle Apps Techno-functional Team Leader, Company - Indorama Synthetics Tbk, Indonesia
  - Oracle Apps (Order Entry and Oracle Receivable) 10.7 SC Implementation
  - Customized New Modules in Oracle Apps (It Included Export Documentation, Packing, LC and Quality Control Module)
  - Developed multiple programs to use Oracle Open Interfaces (Order open Interface, Customer open Interface, Inventory Open Interface and Receivable open Interface)

- Oracle apps 10.6 (Inventory) and Maximo implementation, role – Programmer, Company – Punj Lloyd Limited, India
  - Oracle Apps Implementation (Oracle Inventory) 10.6
  - Maximo Implementation
  - Open Interface implementation between Maximo and Oracle

Email: - sksrivastava@indorama.com