Mining for Gold in the AWR

Yuri van Buren
Senior Oracle DBA Specialist
End-2-End Performance Engineer
Logica Nederland B.V.
Amstelveen – The Netherlands

Keywords: Pro-active Performance Management, Systematic Performance Tuning

Introduction

In this presentation Yuri shows his experience he gained from a real-life project. An extranet application was having big performance problems. The end-2-end response times were not what the customers expected. As Senior Oracle DBA Specialist Yuri was part of a performance team to once and for all get rid of the dreaded performance.

The stakes were high because this application was a sort of broker application which was used to serve 300 concurrent users of more than 50 different customers in the finance market.

Using data from AWR, the Automatic Workload Repository of Oracle Yuri could do a lot of data Mining on all available performance data.

After a short introduction into AWR, Yuri shows how he mined the AWR. This led him to several Golden Nuggets which you can easily use on your own performance projects.

Short introduction of AWR

The Automatic Workload Repository collects performance statistics (and the SQL text itself) for all SQL statements executed in the database. It is a historical performance data warehouse that stores SQL statement CPU, memory and I/O resource consumption.

The background server process (MMON) takes snapshots of the in-memory database statistics (much like STATSPACK) and stores this information in the repository. MMON also provides Oracle10g/11g with a server initiated alert feature, which notifies Grid Control of potential problems (out of space, max extents reached, performance thresholds, etc.).

A number of different statistics are collected by the AWR including wait events, time model statistics, active session history statistics, various system and session-level statistics, object usage statistics, and information on the most resource-intensive SQL statements.

Why use AWR?
The benefits:

- Works on every 10g/11g database that you review
- Easy configurable (interval & retention)
- Easy to find and drill down into recent load spikes
- Gives you the "facts" , no guessing needed
- Ideal for Systematic Performance Tuning
- Use it together with Load testing
- Use it to do capacity planning or forecasting

The investment:

- You need a Diagnostic Pack License

My Approach: How to Mine the AWR?
To be able to repeat your work it is best to use a performance methodology. For mining the AWR I have used the following steps:

Step zero (a to d) – First get overall instance data
Step 1 – Analyze Average Active Sessions of all snapshots
Step 2 – Zoom in on load spikes: find offensive SQL_ID's
Step 3 – Display performance of these SQL_ID's over time
Step 4 – Display Explain Plan data of these SQL_ID's
Step 5 – Find SQL BIND data for these SQL_ID's
Step 6 – Tune the queries

Lets have a closer look into all the steps.

Step zero a – "Spec" your instance
I still use my "Buffer Breakdown Methodology" (Presented on the DOAG2008 conference). Which starts with getting a "Spec", or specification of the instance its Logical IO and Physical IO average call times. It generally gives me a feel about how fast the available Hardware can work for this particular instance. Here are the definitions:

Average PIO call (ms) = (time waited for scattered and sequential reads) / physical reads count

Average LIO call (µs) = 10000 * CPU used by this session / session logical reads count

```
 scattered sequential  tot_waits  tot_waits  average_wait   average_wait
      secs       secs  scattered sequential scattered(ms) sequential(ms)
---------- ---------- ---------- ---------- -------------- --------------
    374416     526084   78482029   88730142         4.7707         5.9290

    Logical IO Average LIO  Physical IO Average PIO   DLF          estimated
         Count   Call (µs)        Count   Call (ms)   factor MBRC       MBRC
-------------- ----------- ------------ ----------- -------- ----- ---------
   25436743928        28.3   1346136403       .6690    23.67 16        16.02
```

In this example I measured an Average Physical IO Call time of 0.6990 milliseconds and a Average Logical IO call time of 28.3 microseconds.

Step zero b – Run sqlarea_top50_elapsed_time.sql
With a SQL script I can show the top 50 statements from v$sqlarea ordered by elapsed time. From this data a first rough performance response time profile can be derived.
{.. listing not provided to save space}.

Step zero c – Run top50lio.sql
From the v$segment_statistics view it is easy to query the top50 segments that were accessed by Logical IOs. Again it gives an opportunity to create a performance profile.

	A	B	C	D	E	F	G
1	OWNER	OBJECT_NAME	TABLESPACE_NAME	OBJECT_TYPE	LOGICAL__READS	PCT_TOTAL	TOTAL
2	----------	----------	----------	----------	----------	----------	----------
3	SAMCO_MAIN	EXCEPTIONLIST	PEARL_DATA	TABLE	1,10E+14	44	2,51E+14
4	SAMCO_MAIN	PK_EXT_INPUT_PORTFOLIO	PEARL_INDEX	INDEX	1113952784	4	2,51E+14
5	SAMCO_MAIN	MEAS_PFSEC	PEARL_DATA	TABLE	760649648	3	2,51E+14
6	SAMCO_MAIN	BP_EXTPF_SEC_INTPF	PEARL_DATA	TABLE	694366448	3	2,51E+14
7	SAMCO_MAIN	PK_MEAS_PFSEC	PEARL_INDEX	INDEX	661083328	3	2,51E+14
8	SAMCO_MAIN	PK_OUT_EXRATE_HEDGERET	PEARL_INDEX	INDEX	506448992	2	2,51E+14
9	SAMCO_MAIN	INP_SECPRICE	PEARL_DATA	TABLE	472759488	2	2,51E+14
10	SAMCO_MAIN	TD_SECURITY	PEARL_DATA	TABLE	472148176	2	2,51E+14
11	SAMCO_MAIN	EXT_INPUT_PORTFOLIO	PEARL_DATA	TABLE	399162144	2	2,51E+14
12	SAMCO_MAIN	PK_INP_SECPRICE	PEARL_INDEX	INDEX	374900880	1	2,51E+14

Step zero d – Run top5pio.sql
From the v$segment_statistics view it is easy to query the top50 segments that were accessed by Physical IOs. This also gives us a performance profile.

	A	B	D	E	F	G	H
1	OWNER	OBJECT_NAME	TABLESPACE_NAME	OBJECT_TYPE	PHYSICAL_READS	PCT_TOTAL	TOTAL
2	----------	----------	----------	----------	----------	----------	----------
3	SAMCO_MAIN	J_EXT_INPUT_PORTFOLIO	PEARL_DATA	TABLE	239045159	22	1066899452
4	SAMCO_MAIN	MEAS_PFSEC	PEARL_DATA	TABLE	145122511	14	1066899452
5	SAMCO_MAIN	MEAS_PFSEC_RETURNS	PEARL_DATA	TABLE	59966661	6	1066899452
6	SAMCO_MAIN	TW_OUTPUT	PEARL_DATA	TABLE	56388093	5	1066899452
7	SAMCO_MAIN	J_INPUT_MEAS_EXTPF_SEC	PEARL_DATA	TABLE	54959391	5	1066899452
8	SAMCO_MAIN	CB_OUTPUT	PEARL_DATA	TABLE	54123445	5	1066899452
9	SAMCO_MAIN	INPUT_MEAS_EXTPF_SEC	PEARL_DATA	TABLE	44601067	4	1066899452
10	SAMCO_MAIN	J_EXT_INPUT_PORTFOLIO_TMP	PEARL_DATA	TABLE	33263423	3	1066899452
11	FLEXIS_2008	OUT_FLEXIS_TW_PLAT_NEW	FLEXIS_PRLP_F	TABLE	29577509	3	1066899452
12	SAMCO_MAIN	EXCEPTIONLIST	PEARL_DATA	TABLE	29336249	3	1066899452

Step 1: Display the Average Active Sessions of all Available AWR snapshots

```
                                                                    DB_TIME
BEGIN_INTERVAL_TIME        END_INTERVAL_TIME          DIFF                    SNAPSHOTS    IN_SECONDS      AAS
-------------------------- -------------------------- ----------------------- --------- ------------- --------
07-OCT-10 10.00.55.417 PM  07-OCT-10 11.00.57.471 PM  +000000000 01:00:02.054   511,512           237      .07
07-OCT-10 11.00.57.471 PM  08-OCT-10 12.00.59.746 AM  +000000000 01:00:02.275   512,513           220      .06
08-OCT-10 12.00.59.746 AM  08-OCT-10 01.00.01.623 AM  +000000000 00:59:01.877   513,514           103      .03
08-OCT-10 01.00.01.623 AM  08-OCT-10 02.00.03.294 AM  +000000000 01:00:01.671   514,515            49      .01
08-OCT-10 02.00.03.294 AM  08-OCT-10 03.00.05.037 AM  +000000000 01:00:01.743   515,516           110      .03
08-OCT-10 03.00.05.037 AM  08-OCT-10 04.00.06.792 AM  +000000000 01:00:01.755   516,517            50      .01
08-OCT-10 04.00.06.792 AM  08-OCT-10 05.00.08.425 AM  +000000000 01:00:01.633   517,518            53      .01
08-OCT-10 05.00.08.425 AM  08-OCT-10 06.00.10.161 AM  +000000000 01:00:01.736   518,519           219      .06
08-OCT-10 06.00.10.161 AM  08-OCT-10 07.00.11.938 AM  +000000000 01:00:01.777   519,520          1420      .39
08-OCT-10 07.00.11.938 AM  08-OCT-10 08.00.13.784 AM  +000000000 01:00:01.846   520,521          3274      .91
08-OCT-10 08.00.13.784 AM  08-OCT-10 09.00.15.687 AM  +000000000 01:00:01.903   521,522          6875     1.91
08-OCT-10 09.00.15.687 AM  08-OCT-10 10.00.18.002 AM  +000000000 01:00:02.315   522,523          6318     1.76
08-OCT-10 10.00.18.002 AM  08-OCT-10 11.00.19.978 AM  +000000000 01:00:01.976   523,524          5791     1.61
08-OCT-10 11.00.19.978 AM  08-OCT-10 12.00.21.898 PM  +000000000 01:00:01.920   524,525          4261     1.18
08-OCT-10 12.00.21.898 PM  08-OCT-10 01.00.23.850 PM  +000000000 01:00:01.952   525,526          5764     1.60
08-OCT-10 01.00.23.850 PM  08-OCT-10 02.00.25.752 PM  +000000000 01:00:01.902   526,527          7021     1.95
08-OCT-10 02.00.25.752 PM  08-OCT-10 03.00.28.442 PM  +000000000 01:00:02.690   527,528         11159     3.10
08-OCT-10 03.00.28.442 PM  08-OCT-10 04.00.30.906 PM  +000000000 01:00:02.464   528,529          5119     1.42
08-OCT-10 04.00.30.906 PM  08-OCT-10 05.00.32.734 PM  +000000000 01:00:01.828   529,530          1428      .40
08-OCT-10 05.00.32.734 PM  08-OCT-10 06.00.34.371 PM  +000000000 01:00:01.637   530,531           240      .07
08-OCT-10 06.00.34.371 PM  08-OCT-10 07.00.36.188 PM  +000000000 01:00:01.817   531,532            75      .02
08-OCT-10 07.00.36.188 PM  08-OCT-10 08.00.37.872 PM  +000000000 01:00:01.684   532,533            57      .02
08-OCT-10 08.00.37.872 PM  08-OCT-10 09.00.39.674 PM  +000000000 01:00:01.802   533,534            60      .02
```

Step 2: Run several AWR reports (with awrrpt.sql) for High Average Active Session periods
A detailed method of how to read a Statspack/AWR report is given during the presentation. This leads to several SQL_ID's which need further investigation.

Step 3: Display the performance of SQL_ID's over time

SNAP_ID	SQL_ID	BEGIN_HOUR	EXECS_ PER HOUR	GETS_ PER HOUR	GETS_ PER EXEC	SECONDS_PER_HOUR	ELAPSED_PER_EXEC
2155	6v9cuv2uzm5tp	2010-12-14 22:00	230	652912	2839	4	.01739
2156	6v9cuv2uzm5tp	2010-12-14 23:01	128	357356	2792	2	.01563
2157	6v9cuv2uzm5tp	2010-12-15 00:00	208	543668	2614	3	.01442
2164	6v9cuv2uzm5tp	2010-12-15 07:00	198	918601	4639	6	.03030
2169	6v9cuv2uzm5tp	2010-12-15 12:00	4222	9138680	2165	53	.01255
2170	6v9cuv2uzm5tp	2010-12-15 13:00	4452	6869999	1543	43	.00966
2171	6v9cuv2uzm5tp	2010-12-15 14:00	4694	6581666	1402	41	.00873
2172	6v9cuv2uzm5tp	2010-12-15 15:00	2606	4831798	1854	30	.01151
2178	6v9cuv2uzm5tp	2010-12-15 21:00	682	1843704	2703	10	.01466
2187	6v9cuv2uzm5tp	2010-12-16 06:00	236	2774157	11755	59	.25000
2188	6v9cuv2uzm5tp	2010-12-16 07:00	1270	14949192	11771	325	.25591
2189	6v9cuv2uzm5tp	2010-12-16 08:00	2632	30916761	11746	685	.26026
2190	6v9cuv2uzm5tp	2010-12-16 09:00	3316	39745761	11986	923	.27835
2191	6v9cuv2uzm5tp	2010-12-16 10:00	4825	57040634	11822	1304	.27026
2192	6v9cuv2uzm5tp	2010-12-16 11:00	4832	57924321	11988	1327	.27463
2193	6v9cuv2uzm5tp	2010-12-16 12:00	2178	25709972	11804	589	.27043
2194	6v9cuv2uzm5tp	2010-12-16 13:00	4196	49747107	11856	1137	.27097
2195	6v9cuv2uzm5tp	2010-12-16 14:00	4446	53923822	12129	1252	.28160
2196	6v9cuv2uzm5tp	2010-12-16 15:00	1228	10654421	8676	318	.25896
2197	6v9cuv2uzm5tp	2010-12-16 16:00	2576	22370901	8684	648	.25155
2198	6v9cuv2uzm5tp	2010-12-16 17:00	1132	9873822	8722	276	.24382
2199	6v9cuv2uzm5tp	2010-12-16 18:00	398	3640249	9146	97	.24372
2200	6v9cuv2uzm5tp	2010-12-16 19:00	560	5089007	9088	136	.24286
2202	6v9cuv2uzm5tp	2010-12-16 21:00	36	333792	9272	9	.25000
2203	6v9cuv2uzm5tp	2010-12-16 22:00	48	418645	8722	12	.25000
2204	6v9cuv2uzm5tp	2010-12-16 23:00	360	3124232	8678	86	.23889
2211	6v9cuv2uzm5tp	2010-12-17 06:00	198	1375789	6948	8	.04040
2212	6v9cuv2uzm5tp	2010-12-17 07:00	866	3316171	3829	18	.02079
2213	6v9cuv2uzm5tp	2010-12-17 08:00	3022	10594345	3506	69	.02283

This gives you a clue during which hours the elapsed times per execute become too large.

Step 4: Display Explain Plan info of SQL_ID's
Using the dbms_xplan makes it easy to display plan info from a SQL_ID:

```
set lines 140 pages 1000
select * from table(dbms_xplan.display_awr('&your_sql_id')) ;
```

```
Plan hash value: 2545306363
-----------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                            |       |       | 31260 (100)|          |
|   1 |  MINUS                         |                            |       |       |            |          |
|   2 |   SORT UNIQUE                  |                            |     1 |  1313 |    46   (3)| 00:00:01 |
|   3 |    TABLE ACCESS FULL           | PLAUSCHECK                 |     1 |  1313 |    45   (0)| 00:00:01 |
|   4 |   SORT UNIQUE                  |                            |     1 |  1343 | 31214   (1)| 00:06:15 |
|   5 |    TABLE ACCESS BY INDEX ROWID| EXCEPTIONLIST              |     1 |    30 |     3   (0)| 00:00:01 |
|   6 |     NESTED LOOPS               |                            |     1 |  1343 |    48   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL         | PLAUSCHECK                 |     1 |  1313 |    45   (0)| 00:00:01 |
|   8 |      INDEX RANGE SCAN          | I_EXCEPTIONLIST_EXECTIME   |     8 |       |     2   (0)| 00:00:01 |
|   9 |       SORT AGGREGATE           |                            |     1 |    27 |            |          |
|  10 |        TABLE ACCESS FULL       | EXCEPTIONLIST              |     1 |    27 | 31165   (1)| 00:06:14 |
```

Step 5: Query the SQL_BINDS of a SQL_ID
From dba_hist_sqlbind it is easy to query the used bindvariables of SQL_ID's in question.

Step 6: Tune the queries with Dan Tow's approach.
A reference is made to the definitive guide for SQL Tuning. Creating a visual tuning diagram helps to fix queries fast.

Last but not least Yuri will show 6 golden nuggets from the AWR.

The queries mentioned in this paper can be obtained by sending me an e-mail.
See contact information below.

Summary
With AWR and my method you can do systematic performance tuning.
Just dare to "Enter the Gold Mine".

Contact address:

Name	**Yuri van Buren**
Company	**Logica Nederland B.V.**
Address	Laan van Kronenburg 2
	1183 AS Amstelveen
	The Netherlands
Phone:	+31(0)10-2537000
Email	yuri.van.buren@logica.com
Internet:	www.logica.com