SQL Plan Stability – Post 11g Upgrade at Verizon Wireless

NYOUG – Dec 13,2011

Sridhar Doraikannu
Sr. Member Oracle DBA Group

verizon wireless

Agenda

Preupgrade

Issues faced after Database upgrade

Lessons learned

Q & A

Who we are

- ✓ #1 wireless carrier with largest 3G network in the U.S.
- ✓ Most reliable wireless voice and data network in the U.S.
  - ✓ 104 million customers
- ✓ 87,000 employees
- ✓ More than 2,000 retail stores & kiosks
- ✓ Joint venture of Verizon Communications (NYSE: VZ) + Vodafone (NYSE: VOD)

Sql plan stability – prior to 11g

verizon wireless

Sql Plan stability

- Stored outlines to preserve execution plans
  DBMS_OUTLN.CREATE_OUTLINE
- Locking of statistics
- Using hints
- sql profiles

Issues

- Both stored outline and sql profile depend on the Hints
- Optimizer still can come up with different plan
- Licensing requirement for using sql profiles
- No new plans is used – potential better plans ignored
- Outlines take precedence over SQL Profiles

Sql plan Management – SPM

verizon wireless

- First execution of any sql's execution plan is treated as optimal and executed
- SPM keeps track of this first executed sql by adding it to the sql log
- Sql log is compared during the second execution of the same sql
- If it is same, plan is added to the plan history
- Sql is executed and than it is added to the sql plan baseline as accepted plan
- Any future new plans for the same sql statement is added to the plan history as non-accepted plan
- For sql's with baseline, only accepted plan get executed
- SPM stores all new plans in the sql management base (SMB)
- It is possible that one of these new plans can become better plan in the future

Managing SQL Plan Baselines

verizon wireless

SQL Plan Baselines

- •Maintains history of plans for individual SQL statements
- •Plan history is maintained only for repeatable SQL statements
- • Default uses 10% of SYSAUX

Automatic plan capture

- • OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES=TRUE

Manual plan capture

- •Loading of plans from SQL Tuning Sets and AWR Snapshots
- •Loading of Plans from the Cursor Cache

Selecting SQL Plan Baselines

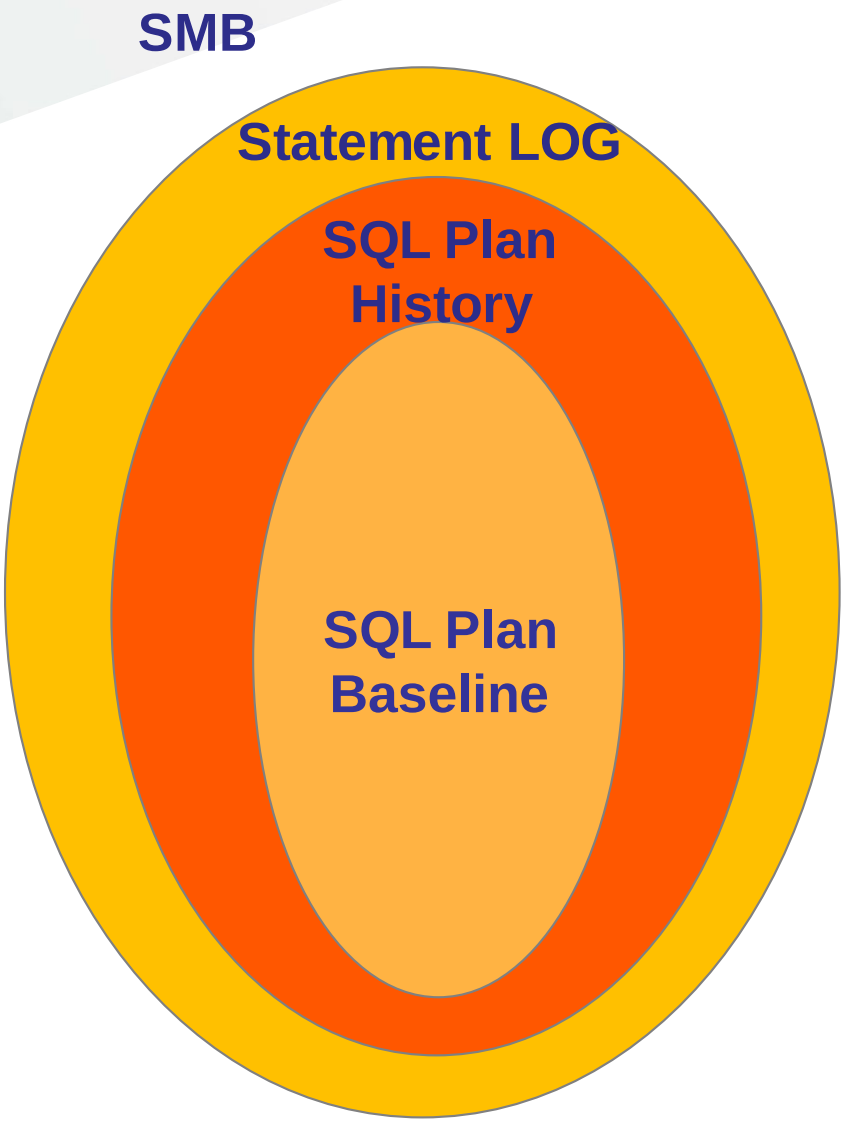
- •optimizer comes up with a plan
- •Checks if the plan is already available and accepted in Baseline
- •New non accepted plan is added to plan history if not found in baseline
- •change in the system (such as a dropped index) causes all accepted plans to become non-reproducible

Evolving SQL Plan Baselines

- •Evolving Plans With Manual Plan Loading (STS , Cursor cache)
- •Evolving Plans With DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE
- • Evaluate **an** unverified execution plan for a given statement in the plan history to become either accepted or unaccepted.

http://docs.oracle.com/cd/B28359_01/server.111/b28274/optplanmgmt.htm

SMB – SQL MANAGEMENT BASE



- SMB is part of the data dictionary
- SMB stores the SQL plan baselines and plan history in the SYSAUX tablespace.
- The SMB also contains SQL profiles

SQL Management Base - SMB

verizon wireless

- By default SMB uses 10% of sysaux space
- By default SMB stores the plans for 53 weeks

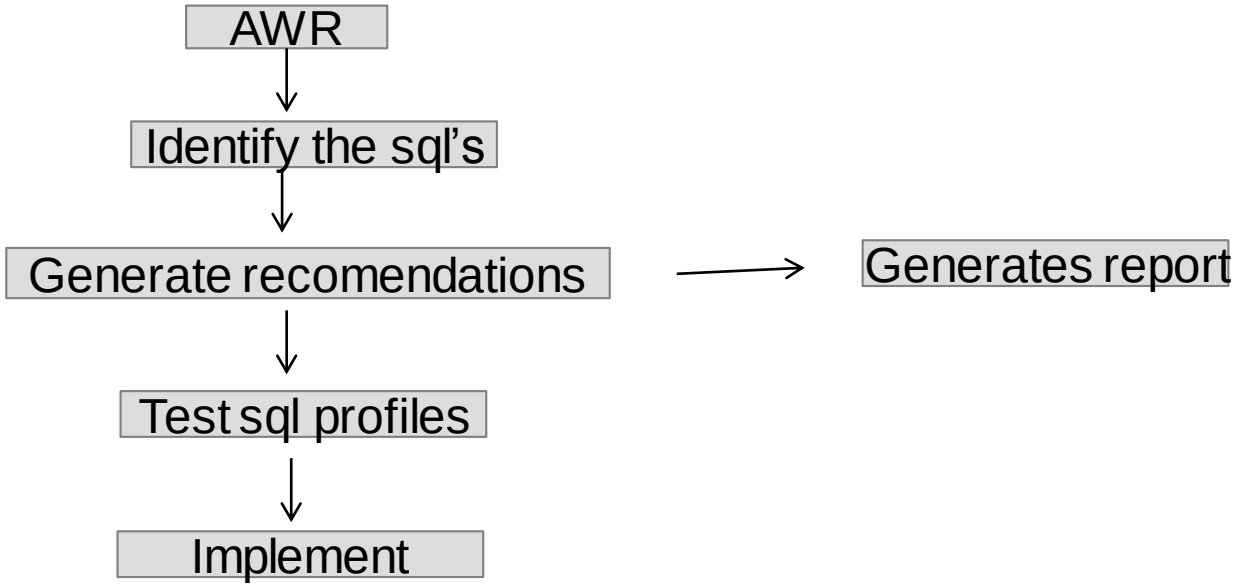
```
SQL> Select * from dba_sql_management_config;

parameter_name          parameter_value     last_modified      modified_by
space_budget_percent          10
Plan_retention_weeks          53
```

- Alert log is written when there is a space issue.
- Space % can be modified using

```
begin
    dbms_spm.configure('space_budget_percent',25);
  end;
```

SQL Plan Baselines with SQL Tuning Advisor



- Identifies the sql candidates from AWR
- Tests the sql profile by executing the sql statement
- if ACCEPT_SQL_PROFILES =TRUE and performance improves three folds than the profile is accepted
- If SPB exists than a new plan is added
- This enables to use the new plan immediately

Evolution of New Plans

- Dba invokes the dbms verify plan (exec dbms_spm.evolve_sql_plan_baselines)
- Optimizer verifies if the new plan can perform better than other baseline plans
- If it is, than the new plan is accepted
- If not it remains in the plan history.

SPM - Plan Selection Process

The optimizer selects a SQL plan baseline based on:

- A plan must be ENABLED
- A plan must be ACCEPTED
- A plan *can be FIXED*
- FIXED plans have precedence

```
SQL_HANDLE          PLAN_NAME                       ORIGIN        ENA ACC FIX AUT
------------------  ------------------------------  ------------  --- --- --- ---
SQL_0014cb1fa90d6612  SQL_PLAN_0056b3ynhuthka1376729  AUTO-CAPTURE   YES  YES  NO  YES
SQL_006478a4387ff2b0  SQL_PLAN_00t3snhw7zwpha26aeb8a  AUTO-CAPTURE   YES  YES  NO  YES
```

In case of multiple choices:

- A SQL profile will be taken into account when available
- The optimizer environment is taken into account
- The costs are taken into account

Note: a stored outline takes precedence over a SQL plan baseline

Upgrade scenarios and SQL Plan Management

verizon wireless

Capture sql plans before upgrade

- Create a SQL Tuning Set on the current release
- Upgrade to the new release
- Load plans from the SQL Tuning Set into the SMB

Capture current plans after upgrade

- Upgrade database to 11g
- Set OPTIMIZER_FEATURES_ENABLE to the old release
- Turn on the capture. optimizer_capture_sql_plan_baselines = TRUE
- All the sql's will execute as it was in the previous release.
- All the plans captured will be same as previous release
- Set OPTIMIZER_FEATURES_ENABLE to the new release

Migrate existing Stored Outlines to SQL Plan Baselines

- DBMS_SPM.MIGRATE_STORED_OUTLINE

Pre Upgrade Tasks

Create Sql Tuning set - STS

verizon wireless

How to create SQL tuning set

```
BEGIN dbms_sqltune.create_sqlset(sqlset_name =>
    'MCS_10G_03052011', description =>'10g STS Before Upgrade',
    sqlset_owner =>'DORAISR'); END;
```

How to capture SQL plans in Tuning set (capture every 5 minutes for 1 complete day)

```
Begin DBMS_SCHEDULER.CREATE_JOB(job_name =>
    'CREATE_STS_Mcs', job_type => 'PLSQL_BLOCK', job_action =>
    'DECLARE bf VARCHAR2(61);  BEGIN bf :=
    q"#UPPER(PARSING_SCHEMA_NAME) NOT IN ("SYS", "SYSTEM")
    #";dbms_sqltune.capture_cursor_cache_sqlset( sqlset_name=>"
    MCS_10G_03052011 ", time_limit=>"86400", repeat_interval=>"300",
    basic_filter=>bf, sqlset_owner=>"DORAISR"); END;', enabled =>
    TRUE); End;
```

Create staging table - STS

- **Store Tuning Set data into staging table**

```
exec DBMS_SQLTUNE.create_stgtab_sqlset(table_name =>
    'STS_10G',schema_name => 'DORAISR');

exec DBMS_SQLTUNE.PACK_STGTAB_SQLSET
   (sqlset_name=>' MCS_10G_03052011 ',
    sqlset_owner=>'DORAISR',staging_table_name=>'STS_10G',
   staging_schema_owner=>'DORAISR');
```

- **Verify data in staging table**

```
SELECT  SQL_ID, PARSING_SCHEMA_NAME,  ELAPSED_TIME,
    CPU_TIME, ROWS_PROCESSED, EXECUTIONS  FROM  STS_10G
    ORDER BY 3;
```

Capture 10G SQL Plans

```
create table SQL_PLANS_BEFORE_11G as
select sysdate, a.*, indi from  sys.GV_$SQL_PLAN a ,dual where a.sql_id in(
   SELECT distinct sql_id FROM sys.gv_$sql WHERE executions > 0
    AND (elapsed_time / 1000) / (DECODE (executions, 0, 1, executions)) > 1
     AND parsing_schema_name in
         ('Schema_name'));
```

Capture SQLs by Elapsed Time before upgrade

verizon wireless

```
CREATE TABLE sql_perf_data_before_11g
( ts, parsing_schema_name, inst_id, executions, bufgets, diskreads,
    elapsedtime, cputime,
cluster_wait_time, rows_processed, sql_id, sqltext, INDICATOR)
AS SELECT SYSDATE, parsing_schema_name, inst_id, executions,
    buffer_gets / executions,        disk_reads / executions,
    elapsed_time / 1000 / executions,
    cpu_time / 1000 / executions,        cluster_wait_time / 1000 / executions,
    rows_processed / executions,
     sql_id,  DBMS_LOB.SUBSTR (sql_fulltext, 4000, 1),  'Ora_10g'
 FROM gv$sql
  WHERE    executions > 0
  AND parsing_schema_name  not IN ('SYS','SYSTEM','DBSNMP')
 AND (elapsed_time / 1000) / (DECODE (executions, 0, 1, executions)) > 0;
```

Post Upgrade Tasks

Parameters for SQL Plan Baseline

Parameters

- optimizer_capture_sql_plan_baselines
  - To capture SPM (False by Default)
- optimizer_use_sql_plan_baselines
  - To use by SPM (True by Default)

Loading of plans into Base lines

- **Do you want to load ALL plans after the upgrade?**

  If performance is good after the upgrade to 11g don't load all plans. Load only the plans for the sql's that is performing bad.

- **Loading all the plans from STS**

  Execution plans can be bulk loaded from an STS into SPM using the PL/SQL procedure

  DBMS_SPM.LOAD_PLANS_FROM_SQLSET or through Oracle Enterprise Manager (EM).

```
SQL> Variable cnt number
        execute :cnt := DBMS_SPM.LOAD_PLANS_FROM_SQLSET(
          sqlset_name => ' MCS_10G_03052011 ');
```

Capture SQL Plans – After upgrade

```
create table SQL_PLANS_AFTER_11G as
select sysdate, a.*, indi from  sys.GV_$SQL_PLAN a ,dual where a.sql_id
    in(
  SELECT distinct sql_id FROM sys.gv_$sql WHERE executions > 0
AND (elapsed_time / 1000) / (DECODE (executions, 0, 1, executions)) > 1
    AND parsing_schema_name in   ('Schema_name'));
```

Comparing SQL – pre and post upgrade

```
SELECT --pre
   pre.PARSING_SCHEMA_NAME   who,  pre.inst_id Inst_id, pre.SQL_ID sql_id,
pre.EXECUTIONS preexec,
  pre.DISKREADS preDisk,
   pre.BUFGETS preBuff_gets,   pre.ROWS_PROCESSED "preROWS",   pre.CPUTIME preCPU,
   -- Diff
  round(pre.ELAPSEDTIME/1000,2) Pre_ELA_TIME,
  round((POST.ELAPSEDTIME/1000-pre.ELAPSEDTIME/1000),2) "Elap Time Diff",
  round( ( ( (POST.ELAPSEDTIME-pre.ELAPSEDTIME) / (pre.ELAPSEDTIME) )/1000 ) * 100 , 2)
"% Diff",
  round(post.ELAPSEDTIME/1000,2) Post_elapsed_time,
   -- Post
post.EXECUTIONS postexec,   post.DISKREADS postdisk,   post.BUFGETS postbuff_gets,
post.ROWS_PROCESSED postrows,
  post.CPUTIME postcpu
  FROM  sql_perf_data_before_11g pre, sql_perf_data_after_11g post
 WHERE
 pre.sql_id = post.sql_id
  and post.executions  > 1
  and pre.inst_id=post.inst_id
and pre.sql_id in ('gn7t6kvt2409g')
 order by 10 desc
```

Identify performance problem by SQL_ID

sql_perf_data_**before**_11g pre,
sql_perf_data_**after**_11g post

SQL_ID	PREEXEC	PREDISK	PREBUFF_GETS	preROWS	PRECPU	PRE_ELA_TIME	Elap Time Diff	% Diff	POST_ELAPSED_TIME	POST EXEC	POST DISK	POST BUFF_GETS	POST ROWS	POST CPU
gn7t6kvt2409g	3	8	63.3333333	35.3333333	6.66666667	.07	**7.93**	**10.81**	8	3	8	63.3333333	35.3333333	6.66666667

Sql_id - gn7t6kvt2409g is performing bad after the upgrade

Loading Plans for only affected sql's

- Assume only one particular sql_id ' **gn7t6kvt2409g** ' is having problem
- Load the plan for that particular sql_id into Baseline

```
begin
:cnt := dbms_spm.load_plans_from_sqlset
(sqlset_name => ' MCS_10G_03052011,
  basic_filter => 'sql_id='' gn7t6kvt2409g '');
end;
/
```

Migrate to SPM from Stored Outlines

verizon wireless

- Turn on the auto capture create_stored_outline = TRUE
- Run all the sql or execute the work load
- If you manually execute the sql's , use the exact sql text (with bind variables)
- Turn of the auto capture. create_stored_outline = FALSE
- Outlines are stored in the OUTLN schema, verify with following sql
  - select name, sql_text, category from user_outlines;
- Export the schema and import it into 11g (or)
- Use EM or DBMS_SPM.MIGRATE_STORED_OUTLINE

Example

Migrate the Stored Outlines based on category, sql text …

```
SQL> exec
    :report:=DBMS_SPM.MIGRATE_STORED_OUTLINE(attribute_name=>'OUTLINE_
    NAME', attribute_value => 'sql_stmt');
-- Migrate all Stored Outlines
SQL> exec
    :report:=DBMS_SPM.MIGRATE_STORED_OUTLINE(attribute_name=>'ALL');
```

SQL Plan Baselines from Dev/Test to Prod

verizon wireless

1) **Source database: Non Prod (QA)**

1.1) **Find out *sql_handle for the stored basline***

```
SQL> SELECT SQL_HANDLE, sql_text, PLAN_NAME, ORIGIN,created, ENABLED,
ACCEPTED, FIXED, MODULE
    FROM   DBA_SQL_PLAN_BASELINES
    where origin like 'STORED%‘
    and sql_text like ‘select  * from sales_tmp where %’
    order by created desc;
```

1.2) **Create the staging table to hold the baseline information to exp.**

```
SQL> set serveroutput on
    BEGIN
      DBMS_SPM.CREATE_STGTAB_BASELINE(
        table_name      => 'spm_sri',
        table_owner     => 'doraisr',
        tablespace_name => 'USERS');
    END;
    /
```

SQL Plan Baselines from Dev/Test to Prod

verizon wireless

1.3) **Pack the baseline into staging table**

```
SQL>
    DECLARE
    l_plans_packed  PLS_INTEGER;
    BEGIN
      l_plans_packed := DBMS_SPM.pack_stgtab_baseline(
      table_name      => 'spm_sri',
      sql_handle => 'SYS_SQL_1835a435979d3240',
      table_owner     => 'DORAISR');
     DBMS_OUTPUT.put_line('Plans Packed: ' || l_plans_packed);
   END;
   /
```

1.4) **Verify data is captured in the table created**

```
SQL> select SQL_HANDLE,CREATED
     from DORAISR.spm_sri;
```

Export this table spm_sri and import it into the target database (prod)

Migrate the plan to Prod

2.1) **Import the table into prod database**

2.2) **Check baseline exist or not for the same sql_handle**

```
SQL> select SQL_HANDLE, PLAN_NAME, ENABLED, ACCEPTED, FIXED,
        substr(CREATED,1,18) created, OPTIMIZER_COST, SQL_TEXT
    from dba_sql_plan_baselines
    where sql_handle='SYS_SQL_1835a435979d3240';
```

2.3) **if exist, drop it or you can disable it.**

```
SQL> set serveroutput on
    DECLARE
        l_plans_dropped  PLS_INTEGER;
    BEGIN
      l_plans_dropped := DBMS_SPM.drop_sql_plan_baseline (
        sql_handle => 'SYS_SQL_1835a435979d3240',
        plan_name  => NULL);
      DBMS_OUTPUT.put_line(l_plans_dropped);
    END;
```

Migrate the plan to Prod

2.4) **unpack baseline data**

```
DECLARE
          l_plans_unpacked   PLS_INTEGER;
        BEGIN
         l_plans_unpacked  := DBMS_SPM.unpack_stgtab_baseline(
           table_name      => 'spm_sri',
           table_owner     => 'DORAISR',
           creator         => 'DORAISR');
         DBMS_OUTPUT.put_line('Plans  Unpacked:  ' || l_plans_unpacked);
        END;
         /
```

2.5) **verify baseline created with enabled=yes, accepted=yes**

```
select SQL_HANDLE,  PLAN_NAME,  ENABLED,  ACCEPTED,   FIXED,
      substr(CREATED,1,18)  created, OPTIMIZER_COST, SQL_TEXT
from dba_sql_plan_baselines
where sql_handle='SYS_SQL_1835a435979d3240';
```

Preferred - Upgrade Strategy

verizon wireless

- Keep (OFE) optimizer_features_enable=10.x after the upgrade to 11g
- Set capture_sql_plan_baselines =true
- Set use_sql_plan_baselines =false
- This will enable optimizer to capture all the 10g plans into sql plan baseline
- Set OFE to 11.x after few days
- Set use_sql_plan_baselines=**true** to start using the 10g plans instead of new 11g
- Set capture_sql_plan_baselines =false (if you don't want to capture new 11g plans)

Lessons Learned

1. Optimizer_capture_sql_plan_baseline

Disabled optimizer_catpure_sql_plan_baseline on may 26 but still it is capturing the sql plans. What is the reason ?

```
 SQL> show parameter sql_plan_baselines
 NAME                                 TYPE                           VALUE
------------------------------------ ------------------------------ -------------
optimizer_capture_sql_plan_baselines boolean                         FALSE
optimizer_use_sql_plan_baselines     boolean                         TRUE
```

```
  1  select count(*),trunc(created),accepted,fixed from DBA_SQL_PLAN_BASELINES
  2  group by trunc(created),accepted,fixed
  3* order by 2;
```

```
  COUNT(*)     TRUNC(CREATED)           ACCEPTED  FIXED
----------    ------------------  --------- ---------
       191 05-26-2011 00:00:00           NO        NO
        50 05-26-2011 00:00:00           YES       NO
        23 05-27-2011 00:00:00           NO        NO
         2 05-29-2011 00:00:00           NO        NO
        10 05-30-2011 00:00:00           NO        NO
       387 05-31-2011 00:00:00           NO        NO
```

Once a plan exist in the base line it will continue to capture , however they will be marked as not accepted.

2. Disabling the BAD plan

Use DBMS_SPM.ALTER_SQL_PLAN_BASELINE to disable the bad plan

```
SQL> variable cnt number;
SQL> exec :cnt :=DBMS_SPM.ALTER_SQL_PLAN_BASELINE( -
SQL_HANDLE => 'SYS_SQL_bf5c9b08f72bde3e', -
PLAN_NAME => 'SQL_PLAN_byr4v13vkrrjy42949306' -
ATTRIBUTE_NAME => 'enabled', -
ATTRIBUTE_VALUE => 'NO');

SQL> SELECT sql_handle, sql_text, plan_name, enabled FROM dba_sql_plan_baselines
Where sql_handle = 'SYS_SQL_bf5c9b08f72bde3e'
And plan_name = 'SQL_PLAN_byr4v13vkrrjy42949306';

SQL_HANDLE SQL_TEXT                              PLAN_NAME            ENABLE
---------- ------------------------------ ------------------        ------
SYS_SQL_bf5c9b08f72bde3e SELECTPROD_NAME,SUM SQL_PLAN_byr4v13vkrrjy42949306 N
```

3. Why optimizer not using the new loaded plan

verizon wireless

Solution - Flush the sql_id

- PURGING A SQL FROM SHARED POOL
- select ADDRESS, HASH_VALUE from V$SQLAREA where SQL_ID ='3hs5fvu5mhmw9'
- exec sys.DBMS_SHARED_POOL.PURGE ('00000008739FEBD8, 2335723401', 'C');
- Run the sql again after flushing from the shared pool.
- Check the note of the plan to verify if it used baseline plan.

```
SQL> select * from table(dbms_xplan.display_cursor('3hs5fvu5mhmw9',0));
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID  3hs5fvu5mhmw9,child  number  0
-------------------------------------
Plan hash value: 3401323945  ( removed the plan to fit the slide)
Note
-----
  - SQL plan baseline SQL_PLAN_8q2x2hscyvhyad01cf9be  used for this
    statement
```

4. Selection criteria for multiple Fixed Plans

If Multiple Plans are FIXED with sql plan baseline, what criteria is used by the optimizer to choose amongst the FIXED plans

If there are multiple FIXED plans, then the Optimizer will cost each of the plans and pick the one with the lowest cost.

Why it is not using the sql profile generated by tuning advisor

When there is a fixed plan in the baseline , the accepted sql profile is added to the baseline as non fixed plan. Optimizer will not use the tuned plan when reproducible fixed plan is present.

Note

- Database does not evolve a fixed SQL plan baseline when you execute DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE.
- You can evolve a fixed SQL plan baseline by manually loading new plans into it

5. Where is sql_id in baseline

SQL_ID is used everywhere else like Vsql, vsql_plan to analyze a specific SQL . However, SQL PLAN BASELINE table does not have Sql_id column. Any easier way to locate a specific SQL in SQL PLAN BASELINE table

The dictionary view DBA_SQL_PLAN_BASELINES does not contain sql_id. Matches need to be made using the SIGNATURE column

For example

```
SELECT s.sql_id, s.sql_text
FROM   v$sql s, dba_sql_plan_baselines b
WHERE  s.exact_matching_signature=b.SIGNATURE;
```

Or you can use the the PLAN_NAME column in DBA_SQL_PLAN_BASELINES.

```
SELECT s.sql_id, s.sql_text
FROM   v$sql s, dba_sql_plan_baselines b
WHERE  s.sql_plan_baseline=b.plan_name;
```

6. Why it is not using the fixed plan

verizon wireless

We exported the good plan and imported it into this prd database (11.2.0.1)

1. We made this new plan as fixed.
2. We notice there are multiple accepted for the same sql but only the one we imported is fixed.
3. Oracle picks up one of these accepted plans stored instead of the fixed plan.
4. We remove all the other accepted plans and keep only the fixed alone .
5. Optimizer now picks up this plan and performs good.

Reason from oracle (probably a bug we believe)

1. Sql baseline contained several accepted and one fixed plan for that sql_id
2. Query issued – Matches with one of the accepted plan but not the fixed plan
3. Goes ahead and execute the accepted plan even though it is not fixed.
4. Marking a plan as fixed did not prevent the optimizer from using other accepted plans in the SQL plan baseline.

According to oracle document - 11.2.0.2

1. Optimizer gives preference to fixed plans over non-fixed ones
2. Optimizer will pick the fixed plan even though non accepted plan cost is less
3. If the fixed plans are not reproducible than the optimizer picks non-fixed accepted plans

7. Issues faced after Database upgrade

verizon wireless

SQL Plans

- The SQL Plans that were captured in the export were SQL Plans from the AWR for the past 30 days.
- Sql plans that were not part of the captured plans started behaving bad.
- These are the plans that were good in 10g but misbehaved in 11g

Solution :

- STS set should span across months to capture sql's from month end processing.
- Capture another set of STS from cursor cache also.
- Keep at least one non prod still in 10g this will help in bringing the affected sql_plan alone from 10g to 11g

.

8. Our strategy capturing from load test

verizon wireless

Issue

- Upgraded our QA box to 11.2
- Load test with Optimizer_features_enable set to 10.2.0.4
- optimizer_capture_sql_plan_baselines = TRUE
- We assumed that we got all the sql
- Many sql's were missing in baseline since SPM captures only repeatable sql statement

Solution

- Populate baselines from cursor cache or use a SQL Tuning Set

9. Issues faced after Database upgrade

verizon wireless

SQL Plans

- Optimizer NOT using SQL PLAN baselines even with FIXED or ACCEPTED plans
- We had this problem with 11.2.0.1

Solution :

- Due to Oracle cardinality feedback feature it was not using our fixed sql plans.
- We turned off this by setting *._optimizer_use_feedback=FALSE
- You can also disable this by

  select /*+ opt_param('_optimizer_use_feedback' 'false') */
- It is suppose to be fixed in 11.2.0.2 but we didn't try changing it back yet

Useful Metalink Notes

- Note.789888.1 - HOW TO LOAD SQL PLANS INTO SPM FROM AWR
- Note 801033.1 - HOW TO MOVE 10gR2 EXECUTION PLANS AND LOAD INTO 11g SPM
- Note 787692.1 - HOW TO LOAD HINTED EXECUTION PLANS INTO SQL PLAN BASELINE
- Note.790039.1 - HOW TO DROP PLANS FROM SPM REPOSITRY
- Note.456518.1 - SQL PLAN MANAGEMENT

Reference

Oracle® Database Performance Tuning Guide 11*g* Release 2 (11.2)
http://docs.oracle.com/cd/E11882_01/server.112/e16638.pdf

Upgrading from Oracle Database 10g to 11g: What to expect from the Optimizer – Maria Colgan

SQL Plan Management in Oracle Database 11g - Maria Colgan

What to expect from the Optimizer when upgrading from Oracle Database 10g to 11g - Maria Colgan & Mohamed Zait

http://kerryosborne.oracle-guy.com/

http://oracle-randolf.blogspot.com

Czuprynski, J. 2008. Oracle Database 11g: SQL Plan Management, Part 1 & 2

http://www.databasejournal.com/features/oracle/article.php/3723676/Oracle-Database-11g-SQL-Plan-Management-Part-1.htm

http://www.databasejournal.com/features/oracle/article.php/3730391/Oracle-Database-11g-SQL-Plan-Management-Part-2.htm


QUESTIONS
ANSWERS