CHAPTER 26

Integrating Extended SQL Trace and AWR

The Active Workload Repository (AWR) and SQL trace files both capture performance-relevant data on SQL and PL/SQL execution. It is undocumented that both data sources may be linked to answer questions that frequently arise during investigations of performance problems. By integrating SQL trace data with AWR, it is possible to find out whether different execution plans were used for a particular statement and at what time. Furthermore, contrary to EXPLAIN PLAN, the AWR is a reliable source for execution plans when plans for certain statements are absent from SQL trace files.

Retrieving Execution Plans

Unlike Statspack release 10.2, which captures the SQL_ID as well as the old (V$SQL.OLD_HASH_VALUE) and new hash value (V$SQL.HASH_VALUE) for SQL statement texts, AWR captures merely the SQL_ID from V$SQL. In Oracle10g, this poses a slight problem for the retrieval of past execution plans and statistics for statements captured by SQL trace files. The reason is that Oracle10g trace files do not include the SQL_ID from V$SQL. Oracle11g SQL trace files include the new parameter sql_id (see “PARSING IN CURSOR Entry Format” in Chapter 24), which corresponds to the SQL_ID from V$SQL. Thus, the issue of mapping a SQL statement text to the SQL_ID has been resolved in Oracle11g.

Note that execution plans are only emitted to SQL trace files when cursors are closed, such that it is possible to encounter trace files that do not contain execution plans for certain statements. If such statements have been aged out of the shared pool by the time the absence of an execution plan becomes evident, then AWR or Statspack (see “Execution Plans for Statements Captured with SQL Trace” in Chapter 25) are the only options for retrieving the plan. Occasionally the optimizer chooses different plans for the same statement over time. One execution plan might result in a very poor response time while another may cause an appropriate response time. The procedure presented next shows how to retrieve all plans for a SQL statement captured by SQL trace and AWR. The five-way join of tables in the sample schema HR depicted in “Retrieving the Text of Captured SQL Statements” in Chapter 25 is used as an example. In releases prior to Oracle11g, we need to start by determining the SQL_ID for a statement, since those releases do not emit it to the SQL trace file. Since AWR captures the SQL statement text as a character large object (CLOB) from V$SQL.SQL_FULLTEXT, this is accomplished by searching for the statement text or portions of it with DBMS_LOB. Statements captured by AWR are stored in
the data dictionary base table WRH$_SQLTEXT and may be accessed through the view DBA_HIST_ SQLTEXT (script awr_sqltext.sql).

```
SQL> SET LONG 1048576
SQL> COLUMN sql_text FORMAT a64 WORD_WRAPPED
SQL> SELECT sql_id, sql_text
FROM dba_hist_sqltext
WHERE dbms_lob.instr(sql_text, '&pattern', 1, 1) > 0;
Enter value for pattern: FROM hr.employees emp, hr.employees mgr
```

```
SQL_ID      SQL_TEXT
------------- -------------------------------------------------------------
yw85nghurbkk SELECT emp.last_name, emp.first_name, j.job_title,
d.department_name, l.city,
1.state_province, 1.postal_code, 1.street_address, emp.email,
emp.phone_number, emp.hire_date, emp.salary, mgr.last_name
FROM hr.employees emp, hr.employees mgr, hr.departments d,
hr.locations l, hr.jobs j
WHERE emp.manager_id=mgr.employee_id
AND emp.department_id=d.department_id
AND d.location_id=l.location_id
AND emp.job_id=j.job_id
```

Having retrieved the SQL_ID, we may now search for AWR snapshots that captured the statement. The view DBA_HIST_SQLSTAT not only contains the snapshot identifiers, but also gives access to execution statistics, hash values of execution plans, and the optimizer environment used (script awr_sqlstat.sql).

```
SQL> SELECT st.snap_id,
to_char(sn.begin_interval_time,'dd. Mon yy hh24:mi') begin_time,
st.plan_hash_value, st.optimizer_env_hash_value opt_env_hash,
round(st.elapsed_time_delta/1000000,2) elapsed,
round(st.cpu_time_delta/1000000,2) cpu,
round(st.iowait_delta/1000000,2) iowait
FROM dba_hist_sqlstat st, dba_hist_snapshot sn
WHERE st.snap_id=sn.snap_id
AND st.sql_id='yw85nghurbkk'
ORDER BY st.snap_id;
```

```
SNAP_ID BEGIN_TIME PLAN_HASH_VALUE OPT_ENV_HASH ELAPSED CPU IOWAIT
---------- -------------- -------------- ---------- -------- --------
72 13. Oct 07 21:39 4095786543 611815770 1.28 .05 .121
73 13. Oct 07 21:42 4095786543 611815770 .32 .06 .27
73 13. Oct 07 21:42 3985860841 3352456078 1.82 .38 1.60
81 15. Oct 07 11:24 4095786543 611815770 .16 .06 .10
```

The fact that the columns PLAN_HASH_VALUE and OPT_ENV_HASH in the query result are not unique for the single SQL_ID “yw85nghurbkk” proves that multiple plans for the same statement have been used and that the statement was run with different optimizer parameter settings. Actually, a single parameter used by the optimizer, namely OPTIMIZER_INDEX_COST_ADJ, which