Tuning slow queries after an upgrade
Who we are

Experts At Your Service

> Over 50 specialists in IT infrastructure
> Certified, experienced, passionate

Based In Switzerland

> 100% self-financed Swiss company
> Over CHF8 mio. Turnover

Leading In Infrastructure Services

> More than 150 customers in CH, D & F
> Over 50 SLAs dbi FlexService contracted
About me

Clemens Bleile
Senior Consultant

+41 78 677 51 09
clemens.bleile[at]dbi-services.com

Tuning slow query after an upgrade
Agenda

1. Upgrade problems
2. Addressing slow queries
3. Summary
Upgrade problems

- Challenges
- Best practices
Upgrade problems
Challenges

>90% of so-called upgrade problems aren't really upgrade problems but performance issues after the upgrade

• Typical problem areas:
  • Optimizer - execution plans
    • Queries are slow
    • Reports or Batches take longer to complete
  • Increased resource requirements
Upgrade problems
Best practices proactively 1

• Test, test, test:
  • Issues: Execution Plan change due to
    • Optimizer enhancements/changes.
    • New features.
    • Changed parameters.
Upgrade problems
Best practices proactively 2: Be prepared

• Use what’s available:
  • Automatic Workload Repository (AWR)/Statspack
  • Automatic Database Diagnostic Monitor (ADDM)
  • Active Session History (ASH)
  • SQL Plan Management (SQL Plan Baselines)
  • Real Application Testing (RAT), SQL Plan Analyzer (SPA)
  • SQLTXPLAIN and/or SQLd360

Use Active Session History (ASH) Analytics

SQL Tuning made much easier with SQLTXPLAIN (SQLT)
Upgrade problems
Best practices proactively 3

• Gather stats correctly (you may check the Statistics Advisor in 12.2.)

  The Optimizer Statistics Advisor

  Rules ➔ Findings ➔ Recommendations ➔ Actions

• Patch proactively (apply the recommended patches: MOS Note 756388.1, 756671.1)

• Check MOS Note “Things to Consider to Avoid Poor Performance or Wrong Results on 12.1.0.2 (Doc ID 2034610.1)” when upgrading to 12.1.

• Check the Alerts for your new release:
  • ALERT: Oracle 12c Release 1 (12.1) Support Status and Alerts (Doc ID 1565065.1)
  • ALERT: Oracle 12c Release 2 (12.2) Support Status and Alerts (Doc ID 2239821.1)
Upgrade problems
Best practices proactively 4: Be prepared

• 12.1.: Consider switching to the 12.2-behavior of adaptive optimizer features: See Note “Recommendations for Adaptive Features in Oracle Database 12c (Adaptive Statistics & 12c SQL Performance) (Doc ID 2187449.1)”

• Avoid migrating multiple components at a time
• Use OS watcher
• Keep the old environment for some time
• Know the new features (!!!)
Upgrade problems
Best practices reactively: Articles in MOS

• There are lots of MOS Notes about Upgrade issues and slow (Query-) Performance:
  • Troubleshooting a Server Upgrade Resulting in Slow Query Performance (Doc ID 160089.1)
  • Query with Unchanged Execution Plan is Slower than Previously (Doc ID 223806.1)
  • Avoiding and Resolving Database Performance Related Issues After Upgrade (Doc ID 1528847.1)

• BUT...

• BOTTOM LINE:
  • Proactively test your application
  • Be prepared for issues after the Upgrade. Especially know what tools (like SQLTXPLAIN) can do for you.
Adressing slow queries

> The usual reaction
> Quick wins: Overview
> How to do it right?
> SQL Plan Management
> SQLTXPLAIN
> SQLd360
Addressing slow queries

The usual reaction

• Gather fresh statistics using dbms_stats
• Run the SQL Tuning Advisor (when licensed)
• Switch off new features on system-level
  • OPTIMIZER_ADAPTIVE_FEATURES = FALSE in 12.1.
  • OPTIMIZER_FEATURES_ENABLE = <previous release>

• Possible issues with above approach
  • It often does not help (!!!)
  • It has a global affect
    • Change on system level affects all SQL
    • OPTIMIZER_ADAPTIVE_FEATURES in 12.1 covers adaptive plans AND statistics
    • Additional index suggested by the Tuning Advisor affects many SQL
Addressing slow queries
Quick wins: Overview

• **Fix the previous plan as a *temporary workaround***
  • IMPORTANT: Whenever possible change **locally** (on SQL-statement level), not globally (on system level)
    • Move a SQL Plan Baseline (or Outline) from the previous release database
    • Use SQLPatch (SQL Repair Advisor which does not require an additional license in EE) to fix a plan
    • SQL Tuning Advisor (if licensed)
    • Set the OPTIMIZER_FEATURES_ENABLE-parameter to the previous release. Use the hint
      /*+ OPTIMIZER_FEATURES_ENABLE('<version>') */ through a SQL Patch or SQL Plan Baseline.
    • Switch off new features (e.g. 12.1. adaptive features) in a test environment and create a SQL Plan Baseline.
Addressing slow queries
How to do it right?

• Identify the slow SQL
• Do an analysis
• Fix the issue
Addressing slow queries
How to do it right? Identify the slow SQL(s)

- **Identify the slow SQL**
  - Instrumented Application
  - Cloud Control Performance pages: Top Activity, ASH Analytics.
  - AWR/ADDM/Statspack

<table>
<thead>
<tr>
<th>Elapsed Time (s)</th>
<th>Executions</th>
<th>Elapsed Time per Exec (s)</th>
<th>%Total</th>
<th>%CPU</th>
<th>%IO</th>
<th>SQL Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.217.13</td>
<td>686</td>
<td>6.15</td>
<td>70.79</td>
<td>58.52</td>
<td>0.00</td>
<td>4e5tx2dagn5p63</td>
</tr>
<tr>
<td>424.76</td>
<td>199</td>
<td>2.13</td>
<td>7.13</td>
<td>82.08</td>
<td>0.09</td>
<td>f1rm1t1d03rwhz</td>
</tr>
<tr>
<td>121.96</td>
<td>232</td>
<td>0.54</td>
<td>2.10</td>
<td>69.98</td>
<td>0.00</td>
<td>0zuzqu6mnr12y</td>
</tr>
<tr>
<td>115.64</td>
<td>105</td>
<td>1.10</td>
<td>1.94</td>
<td>75.81</td>
<td>0.00</td>
<td>4rfr4rhdpgrk0</td>
</tr>
</tbody>
</table>

- **Shared Pool**

```sql
select sql_id, child_number, executions, elapsed_time/executions/1000000 elap_sec_per_exec from v$sql
where executions > 0
and last_active_time > (sysdate - 1/24)
and elapsed_time/executions/1000000 > 10;
```

- 10046 Trace
Addressing slow queries
How to do it right? Do an analysis

• Do an analysis
  • Doing an in-depth analysis usually takes too long:
    • Find the root cause
    • The solution may require application changes, additional objects (indexes, constraints) or additional statistics

• Find the “good” plan from before the upgrade
  • May still be in the AWR when it reached the threshold to be reported
  • Use the old or new environment to reproduce the good plan and get it from the Shared Pool
  • Use tools like SQLTXPLAIN (MOS Note 215187.1) and SQLd360 (http://mauro.pagano.com) to get all needed data
  • Find a “good” plan “quickly” using proven methods
    • E.g. visualize objects and statistics and hint the query with Dan Tow’s method
Addressing slow queries
How to do it right? Fix the issue

• Plan from before the Upgrade: Fix the issue locally without modifying the application
  • SQL Plan Management (SQL Plan Baseline)
  • SQL Profile
  • SQL Patch
  • Outline (in Standard Edition)
Addressing slow queries
How to do it right? Fix the issue: SQL Plan Baseline

• SQL Plan Baseline: Alternative 1
  • Get the good plan from the previous environment:
  • Manually
    • Create a SQL Tuning Set in your old environment and move it to the new version
    • Use dbms_spm.load_plans_from_sqlset to load a plan from the tuning set
    • See MOS “How to Load SQL Execution Plans in to a STS so that they can be Transferred (to 11g) and Loaded into a SQL Plan Baseline. (Doc ID 801033.1)”
  • With a script
    • SQLd360: sqld360-master/sql/sqld360_create_sql_baseline.sql
Addressing slow queries
How to do it right? Fix the issue: SQL Plan Baseline

- SQL Plan Baseline: Alternative 2
  - Get the good plan with a hint on the new version:
    - Manual steps:
      - Create a SQL Plan Baseline for the “bad” plan
      - Run the query with adjusted settings (hint, session-parameter) to get a good plan
      - Associate the SQL_ID and PLAN_HASH_VALUE of the good plan to the SQL_HANDLE of the Baseline

```
retval := dbms_spm.load_plans_from_cursor_cache(
    sql_handle => '<sql_handle baseline bad plan>',
    sql_id => '<sql_id good plan>',
    plan_hash_value => '<plan hash value good plan>');
```

Scripted: Use SQLTXPLAIN: sqlt/utl/coe_load_sql_baseline.sql
Addressing slow queries
How to do it right? Fix the issue: SQL Profile

• SQL Profile usually suggested by the Tuning Advisor:
  • Used when cardinality estimation is wrong
    • It’s a statement level statistic correction.
    • Reasons: data correlation, join estimation, etc.
  • It adds OPT_ESTIMATE() hints to the query
    • Table cardinality factor
    • Join cardinality factor
    • Index selectivity factor
    • It scales the estimation, so still reliable if volume grows
Addressing slow queries
How to do it right? Fix the issue: SQL Profile

• SQL Profile with SQLTXPLAIN: Alternative 3
  • Can be used to fix a plan
  • Interesting, because of FORCE MATCHING feature

• Scripts in sqlt/utl:
  • coe_load_sql_profile.sql → the same as coe_load_sql_baseline.sql, but as a SQL Profile
  • coe_xfr_sql_profile.sql → creates a script, which can run on any DB. Allows to adjust FORCE_MATCHING.
  • coe_gen_sql_profile.sql → creates scripts for all plans found in AWR and Shared Pool.
Addressing slow queries
How to do it right? Fix the issue: SQL Patch

- **SQL Patch (SQL Repair Advisor)**
  - May check if the advisor suggests a SQL Patch
  - Can be used to add other hints to SQL
    - When we cannot change the code
    - And the chosen execution plan is bad
    - Bug in an access path that leads to ORA-600
    - Bug in a transformation that gives wrong result
  
- See MOS Note “How To Use DBMS_SQLDIAG To Diagnose Query Performance Issues (Doc ID 1386802.1)”

- In 12.2, DBMS_SQLDIAG has an official method to add hints to SQL: `create_sql_patch/drop_sql_patch`. Apply patch 17203284 to get this functionality officially in 12.1.

- [https://oracle-base.com/articles/11g/sql-repair-advisor-11g](https://oracle-base.com/articles/11g/sql-repair-advisor-11g)
Addressing slow queries
How to do it right? Fix the issue: SQL Patch

- **SQL Patch: Alternative 4 (add a hint to a SQL)**
  - 12.2. (and 12.1. with patch 17203284):
    
    ```sql
    retval := dbms_sqldiag.create_sql_patch(
        sql_id => '<sql_id with bad plan>',
        hint_text => '<additional hint for the SQL>');
    ```

  - 11g and 12.1. (without patch 17203284):
    
    ```sql
    DECLARE
        cl_sql_text CLOB;
    BEGIN
        SELECT SQL_FULLTEXT INTO cl_sql_text from gv$sqlarea
        where sql_id = '<sql id of the query>';
        sys.dbms_sqldiag_internal.i_create_patch(sql_text=>cl_sql_text,
            hint_text=>'BIND_AWARE', name=> 'SQL_Patch_BIND_AWARE');
    end;
    /
Addressing slow queries
How to do it right? Fix the issue: SQL Patch

• **SQL Patch in SQLTXPLAIN**
  • coe_gen_sql_patch.sql
    • Create a SQL Patch with a specified hint and produce a 10053 Trace
    • → Do not use it in a Multitenant and/or 12.2.-DB

**SQL Tuning made much easier with SQLTXPLAIN (SQLT)**
Summary
Summary
Be prepared for the Upgrade

• Use new features to test the new release with your SQL/application if possible
• Become familiar with the tool SQLTXPLAIN. Alternatively/complementary use SQLd360.
• Go with the Oracle recommended patches.
• Review the alerts Oracle has provided for your target release.
• Use the OS watcher black box.
• Keep the old environment and thoroughly test the application before migrating.
Summary
Tune slow queries after the Upgrade

- For the “significant” slow queries: Focus is to get to previous plans asap
- Tune “local”, i.e. at statement level
- Many tools and utilities in SQLTXPLAIN and SQLd360 make your life easier to change plans without touching the SQL:
  - Generate a SQL Plan Baseline / SQL Profile with the “good” plan
  - Generate a SQL Baseline in the old environment and copy it
- If you cannot get back to a previous plan
  - Consider Dan Tow’s method as it is quite fast
- After implementing a workaround a deeper analysis HAS TO follow
Any questions?

Please do ask!

We would love to boost your IT-Infrastructure

How about you?