ORACLE

What can we learn from how Oracle ADB auto-repairs SQL performance regressions?

Oracle Database 19c

Nigel Bayliss

Optimizer Product Manager

@vldbb

http://blogs.oracle.com/optimizer

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Introduction

- The deck content is…
  - Relevant to Oracle Database 11g and onwards
  - Relevant for Oracle Exadata, Autonomous Database and on-premises deployments
  - Not prescriptive or 'best practice' – it is food for thought
  - For environments that largely use *repeatable* SQL
    - Bind-variable SQL
    - Forced cursor sharing
- I will use 'Autonomous Database/ADB features' as shorthand for ADB *and* Exadata features
- I might only look at chat at the end
- Feedback encouraged/welcomed (email, Twitter etc)

Agenda

- Repairing SQL performance regressions in Oracle Autonomous Database
- The *automatic SQL tuning set* (ASTS)
- SQL plan management (SPM) in >=Oracle Database 19c non-ADB/Exadata
- SPM in pre-Oracle Database 19c
  - Especially with upgrade to Oracle Database 19c (non-ADB/Exadata) in mind
- The payoff: repairing SQL performance regressions quickly
  - Mainly aimed at Oracle Database 19c (non-ADB/Exadata)

SQL Plan Management

- SQL plan management (SPM) has been available since Oracle Database 11g
- Enforces SQL execution plans to *prevent* query regression
- 'Approved' (accepted) SQL execution plans are held in the form of SQL plan baselines
- SQL plan baseline plans are not set in stone: schemas change, data changes
  - Using evolution, the optimizer test executes alternative plans, and it can (if desired) enable them automatically when improvements are found
- Strategic/proactive use-case:
  - Capture *all* SQL reusable SQL statements
  - *Proactively prevent* query performance regressions
  - Test and evolve all plan changes
- Tactical/reactive use-case:
  - Tactically 'repair' SQL performance regressions

SQL Plan Management

Oracle Autonomous Database

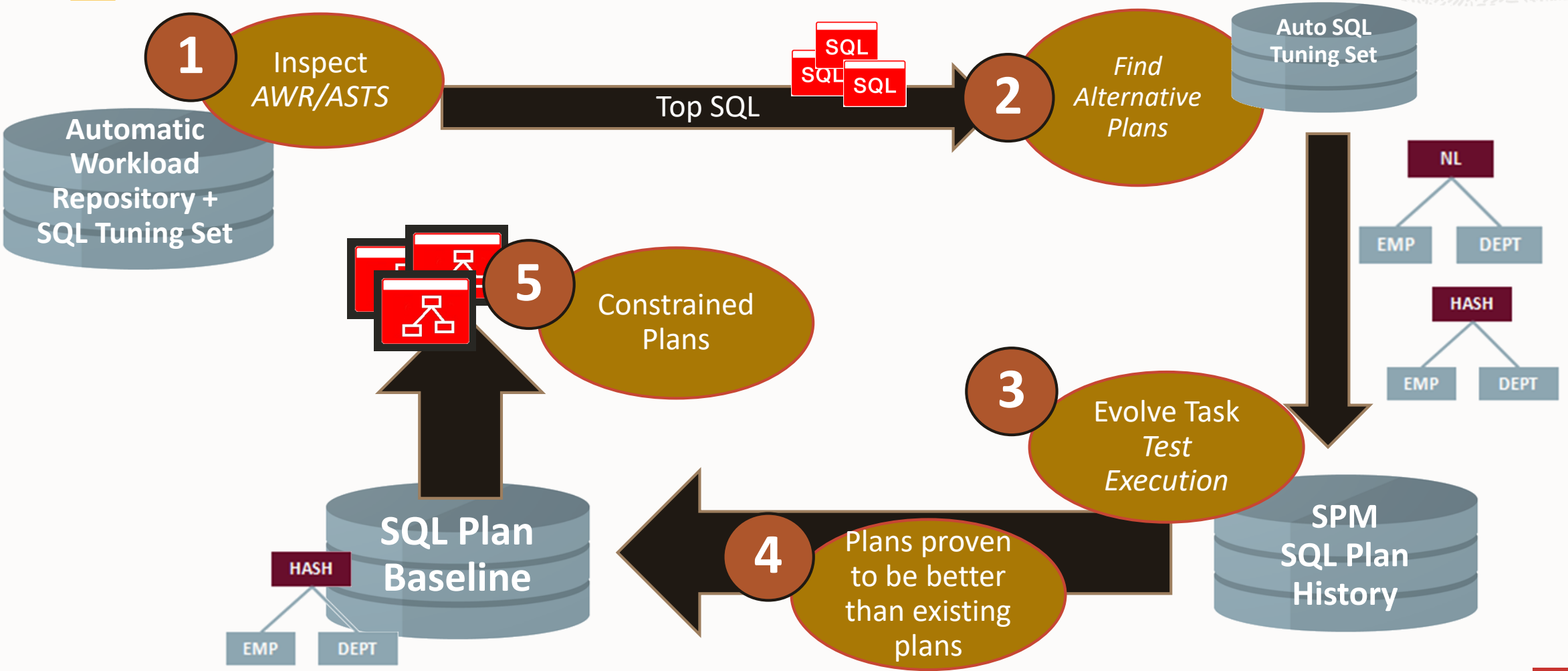
Automatic SQL Plan Management

Oracle Autonomous Database automates the tactical approach

- Automatic SQL plan management detects and repairs SQL performance regressions automatically
  - Automating the tactical approach
- 'AUTO' mode
  - AUTO setting for *alternate_plan_source* and *alternate_plan_baseline* SPM API settings
- AWR and ASTS* are used to target SQL statements and plans that consume significant system resources
  - *ASTS - Auto SQL Tuning Set is a system-maintained SQL tuning set - covered later in this presentation
- Alternative execution plans (for targeted SQL statements) are found in ASTS and stored in SPM for the SPM evolve advisor to test execute and compare
  - "Maybe an old plan is better than the new plan"

- https://blogs.oracle.com/optimizer/what-is-automatic-sql-plan-management-and-why-should-you-care
  https://blogs.oracle.com/optimizer/whats-new-in-the-oracle-optimizer-for-oracle-database-19c

Automatic SQL Plan Management



Summary: Automatic SQL Plan Management

Oracle Autonomous Database

- SPM proactively repairs SQL statement performance regressions
- For high resource-consuming SQL, alternative SQL execution plans are retrieved from ASTS and added to the SPM SQL plan history
  - ACCEPTED = NO in DBA_SQL_PLAN_BASELINES
- The performance of alternative execution plans is compared using test execution in SPM evolve
- 'Winning' SQL execution plans are added to the SQL plan baseline
  - ACCEPTED = YES in DBA_SQL_PLAN_BASELINES
- In Autonomous Database deployments, the *high-frequency SQL plan management* task executes evolution frequently to repair plans quickly outside maintenance windows

Automatic SQL Tuning Set

Automatic SQL Tuning Set (ASTS)

First available in Oracle Autonomous Database

- A SQL tuning set maintained by system (high-frequency) auto tasks
  - See DBA_SQLSET_STATEMENTS for SQLSET_NAME = 'SYS_AUTO_STS'
  - 'Auto STS Capture Task' details in DBA_AUTOTASK_SCHEDULE_CONTROL and DBA_AUTOTASK_SETTINGS*
- Intended to be fully self-managing and used in its default configuration
  - "ON or OFF": DBMS_AUTO_TASK_ADMIN ENABLE/DISABLE (client_name=> 'Auto STS Capture Task'…)
  - *Auto STS Capture Task* instantiates every 15 minutes
  - SQL statement retention is 53 weeks
- Dynamic SQL statement duplication is mitigated
  - There is a soft limit for capturing duplicate force matching signature SQL statements
- ASTS MOS note explains how to monitor resource consumption (Doc ID 2686869.1)
  - ASTS consumes space in SYSAUX
  - ASTS task consumes CPU to gather SQL from cursor cache

*In Oracle Database 20c preview

Automatic SQL Tuning Set (ASTS)

- ASTS was created to support automatic indexing in Oracle Database 19c, then exposed in RU 19.7 as an *infrastructure component*
- Let's address the dark side of the force:
  - Mike's Blog https://mikedietrichde.com/ :
    Do you love unexpected surprises? SYS_AUTO_STS in Oracle 19.7.0
  - See MOS note
    Automatic SQL Tuning Sets (ASTS) 19c RU 19.7 Onwards (Doc ID 2686869.1)
  - ON-by-default in ATP/ADW
  - OFF-by-default (but available) elsewhere
- Latest RUs for on-premises, task is OFF by default
- https://blogs.oracle.com/optimizer/the-auto-sql-tuning-set

SPM in non-ADB/Exadata

>=Oracle Database 19c

SQL Plan Management

>=Oracle Database 19c non-ADB/Exadata Environments

- Consider the significant advantages of keeping workload query plans
  - ASTS
- Ready to repair SQL performance (plan-related) regressions when they occur
- ASTS overheads are visible (Doc ID 2686869.1):
  - ASTS typically less that 1.5GB total but there is of course a wide spread of plan sizes and SQL statement counts
  - 12.2K statements ~540MB or about 45KB/statement but **mileage will vary**
  - Systems with very large number of queries may have shorter SQL statements and smaller plans, so overhead may drop under 10KB/statement
  - Typically 30-60sec capture task runtime for large ASTS (single thread)

SPM in non-ADB/Exadata

Pre-Oracle Database 19c

SQL Plan Capture

Pre-Oracle Database 19c non-ADB/Exadata Environments

- Create and maintain a SQL tuning set
  - Flexible API - DBMS_SQLTUNE.LOAD_SQLSET
- SQL Plan Management capture
  - >= Oracle Database 12.2 has filters for auto capture
  - DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE
    – can be very selective if driven from V$SQLAREA
  - DBMS_SPM.LOAD_PLANS_FROM_SQLSET is a good option too

SQL Plan Capture

Pre-Oracle Database 19c, non-ADB/Exadata Environments

- SPM example:
  - `optimizer_capture_sql_plan_baselines = TRUE`
  - `optimizer_use_sql_plan_baselines = FALSE`
  - `DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(` **>=12c**

```
    task_name => 'SYS_AUTO_SPM_EVOLVE_TASK' ,
    parameter => 'ACCEPT_PLANS',
    value => 'FALSE');
```

- Which means…
  - Capture SQL execution plans automatically in SQL plan history
  - Do not force plans to use SQL plan baselines (yet)
  - Do not allow SQL execution plans to be accepted automatically

SQL Plan Capture

Pre-Oracle Database 19c, non-ADB/Exadata Environments

- What about SPM SYSAUX 'flooding' of legend?
  - Only *repeatable* SQL captured
  - BUG 14054126 - SYSAUX FILL UP WITH SPM, CANNOT PURGE

  - `select * from v$sysaux_occupants where occupant_name = 'SQL_MANAGEMENT_BASE';`
  - `dbms_spm.configure('plan_retention_weeks', NN);` [53 week default]
  - `dbms_spm.configure('space_budget_percent', NN);` [Alert log message]
  - `ret := sys.dbms_spm_internal.auto_purge_sql_plan_baseline;`
  - `Reducing the Space Usage of the SQL Management Base in the SYSAUX Tablespace (Doc ID 1499542.1)`

The Payoff

Repairing SQL Performance Regressions Quickly

Recap: Oracle Autonomous Database

- SQL statement performance regressions are managed fully automatically using *automatic SQL plan management*
- Particularly useful in the context of upgrades
- You can use a manual/semi-manual approach if you want to
- https://blogs.oracle.com/optimizer/spm-auto-capture-vs-auto-spm

>=Oracle Database 19c Non-ADB/Exadata

- If you upgraded your database from an earlier release, you captured pre-upgrade SQL execution plans
- Whether you upgraded or not, you are capturing Oracle Database 19c SQL execution plans
- **You are in a position to address plan-related SQL performance regressions quickly…**

Repairing SQL Performance Regression

>=Oracle Database 19c, non-ADB/Exadata Environments

- https://blogs.oracle.com/optimizer/repairing-sql-performance-regression-with-sql-plan-management
- Assuming that a SQL statement has no SQL plan baseline already, and has suffered a performance regression…
- Step#1: Create a SQL plan history entry for the problem query plan:
  - dbms_spm.load_plans_from_cursor_cache

```
n := dbms_spm.load_plans_from_cursor_cache(
                sql_id          => sqlid,
                plan_hash_value => phv,
                enabled         => 'no');
```

  - dbms_spm.load_plans_from_sqlset

Repairing SQL Performance Regression

>=Oracle Database 19c, non-ADB/Exadata Environments

- https://blogs.oracle.com/optimizer/repairing-sql-performance-regression-with-sql-plan-management
- Step#2: Evolve:

```
tname := DBMS_SPM.CREATE_EVOLVE_TASK(sql_handle=>[handle for SPM SQL plan history entry]);

DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
   task_name => tname,
   parameter => 'ALTERNATE_PLAN_BASELINE',
   value     => 'EXISTING');

DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
   task_name => tname,
   parameter => 'ALTERNATE_PLAN_SOURCE',
   value     => 'CURSOR_CACHE+AUTOMATIC_WORKLOAD_REPOSITORY+SQL_TUNING_SET');

DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
   task_name => tname,
   parameter => 'ALTERNATE_PLAN_LIMIT',
   value     => 'UNLIMITED');

ename := DBMS_SPM.EXECUTE_EVOLVE_TASK(tname);
```

https://github.com/oracle/oracle-db-examples/tree/master/optimizer/spm_fix

Repairing SQL Performance Regression

>=Oracle Database 19c, non-ADB/Exadata Environments

- https://blogs.oracle.com/optimizer/repairing-sql-performance-regression-with-sql-plan-management
- Optional Step#3: Report:
  - `select DBMS_SPM.REPORT_EVOLVE_TASK(task_name=>tname) into from dual;`
- Optional Step#4: Implement:
  - `n := DBMS_SPM.IMPLEMENT_EVOLVE_TASK(tname);`

SQL Plan Baselines in General

- *Enable* the SQL plan baseline plan(s) you want
  - Has an advantage over FIXED, in that alternative (and potentially better) plans are captured in the SQL plan history
- As mentioned earlier, you can control plan acceptance (>=12c)

```
BEGIN
   DBMS_SPM.SET_EVOLVE_TASK_PARAMETER(
      task_name => 'SYS_AUTO_SPM_EVOLVE_TASK' ,
      parameter => 'ACCEPT_PLANS',
      value => 'FALSE');
END;
/
```

Summary

- SQL plan management is central to Oracle's Autonomous Database strategy
- Is it time to re-evaluate plan capture?
- You can use SQL plan management auto capture or bulk capture from multiple sources
- You can capture plans in SQL tuning sets
- You can use SPM to repair SQL performance regressions faster
- There are plenty of options available even if you're upgrading Oracle Database 11g, 12c or 18c non-Autonomous Databases to Oracle Database 19c
- If nothing else, historic plan capture (in addition to AWR) is a useful tool for addressing performance issues

More Information

- @vldbb
- Email Nigel.Bayliss [@oracle.com]
- Optimizer blog:
  - http://blogs.oracle.com/optimizer
  - https://blogs.oracle.com/optimizer/the-auto-sql-tuning-set
  - https://blogs.oracle.com/optimizer/repairing-sql-performance-regression-with-sql-plan-management
- Upgrade blog:
  - https://mikedietrichde.com/
- License guide:
  - https://docs.oracle.com/en/database/oracle/oracle-database/19/dblic/index.html

Thank you