An Oracle White Paper
February, 2015

Oracle Database In-Memory Advisor Best Practices

Disclaimer

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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Intended Audience

Readers are assumed to have hands-on experience with Oracle Database technologies from the perspective of a DBA or performance specialist.

Introduction

Oracle Database 12.1.0.2 introduced Oracle Database In-Memory allowing a single database to efficiently support mixed analytic and transactional workloads. An Oracle Database configured with Database In-Memory delivers optimal performance for transactions while simultaneously supporting real-time analytics and reporting. This is possible due to a unique "dual-format" architecture that enables data to be maintained in both the existing Oracle row format, for OLTP operations, and a new purely in-memory column format, optimized for analytical processing. In-Memory also enables both datamarts and data warehouses to provide more ad-hoc analytics, giving end-users the ability to ask multiple business driving queries in the same time it takes to run just one now.

For complete details about Oracle Database In-Memory, see the Oracle Database In-Memory whitepaper and the Oracle Database In-Memory Page on oracle.com.

The Oracle Database In-Memory Advisor analyzes your workload and makes specific recommendations regarding how to size Oracle Database In-Memory and which objects would render the greatest benefit to your system when placed In-Memory. See the Oracle Database In-Memory Advisor whitepaper for details on the In-Memory Advisor.

This paper discusses best practices for using the Oracle Database In-Memory Advisor.

Database In-Memory Advisor

The goal of Oracle Database In-Memory is to optimize analytical processing in the database. The In-Memory Advisor analyzes the analytical processing workload present in your database to determine an estimated benefit for the database as a whole if that analytical workload is optimized.

The In-Memory Advisor differentiates analytics processing from other database activity based upon SQL plan cardinality, Active Session History (ASH), use of parallel query, and other statistics.

The In-Memory Advisor estimates the In-Memory size of objects based upon statistics and heuristic compression factors and, optionally, the DBMS_COMPRESSION package (in Oracle Database 12.1.0.2 and above).

The In-Memory Advisor estimates analytic processing performance improvement factors based upon the following:

- Elimination of user I/O waits, cluster transfer waits, buffer cache latch waits, etc.
- Certain query processing advantages related to specific compression types.
- Decompression cost heuristics per specific compression types.
- SQL plan selectivity, number of columns in the result set, etc.

The Advisor produces a recommendation report. The report lists a number of In-Memory sizes with estimated performance benefits, lists the objects which should be placed in the In-Memory column store for a given In-Memory size and the recommended compression factor for those objects.

In-Memory Advisor Best Practices

Below are some best practices to help you use the Advisor most effectively in your environment.

Use SQL Performance Analyzer to Validate Recommendations

SQL Performance Analyzer (SPA) allows you to validate the change in performance of individual SQL statements in your database. SPA can be used to measure the impact of any change to your database that could affect performance. Therefore, SPA is ideally suited for evaluating the impact of placing specific database objects in the In-Memory column store.

The input to SPA is a SQL Tuning Set (STS). A SQL Tuning Set is a database object that includes one or more SQL statements, along with their execution statistics and execution context. SQL statements can be loaded into a SQL Tuning Set from different sources, including the cursor cache, Automatic Workload Repository (AWR), and existing SQL Tuning Sets.

SPA creates two trials by executing every SQL in the SQL Tuning Set twice. SPA first executes every SQL in the pre-change environment. After the change has been made that will impact performance, SPA executes the second trial.

In this case, trial 1 will be executed using the traditional database access structures. Trial 2 will be executed with database objects in the In-Memory column store. After the two trials have been run, SPA produces a report highlighting the performance impact of the change – both for the workload as a whole and also the impact on each individual SQL in the SQL Tuning Set.

You can execute the various tasks for SPA from Enterprise Manager 12c or from PL/SQL. Both methods are documented in the Database Testing Guide available in the Oracle database documentation library.

Create a SQL Tuning Set

The first step is to create the SQL Tuning Set (STS) that contains the set of SQL to be tested for performance improvements. An STS can be created from polling the cursor cache, from existing AWR snapshots, or as a subset of an existing STS.

Database In-Memory improves the performance of analytic queries – queries that generally take a significant amount of time to execute. To measure the benefit of Database In-Memory, create an STS that consists of long running queries.

To create an STS from the cursor cache using PL/SQL, you use the DBMS_SQLTUNE. CAPTURE_CURSOR_CACHE_SQLSET procedure.. You can specify a SQL Tuning Set name, the duration of time in seconds for the capture to run and the polling interval. For example:

```
EXEC DBMS_SQLTUNE.CAPTURE_CURSOR_CACHE_SQLSET( -
                              sqlset_name     => 'IM_STS', -
                              time_limit      =>  300, -
                              repeat_interval =>  10);
```

This would create an STS named IM_STS by polling the cursor cache for SQL every 10 seconds over the next 300 seconds.

You can also create an STS from SQL captured in Automatic Workload Repository (AWR) snapshots. AWR automatically captures high load SQL at regular intervals in your database. For testing Database In-Memory, select long running queries based on elapsed time. For example, the following code snippet creates an STS named IM_STS. It then loads that STS with SQL from AWR snapshots 1000 through 1010 that have elapsed time greater than 60 seconds.

```
EXEC DBMS_SQLTUNE.CREATE_SQLSET (sqlset_name  => 'IM_STS');
DECLARE
  cur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
  OPEN cur FOR
    SELECT VALUE(P)
      FROM table(
        DBMS_SQLTUNE.SELECT_WORKLOAD_REPOSITORY(1000,1010,
                           'elapsed_time > 60000000') ) P;

  DBMS_SQLTUNE.LOAD_SQLSET(sqlset_name => 'IM_STS',
                           populate_cursor => cur);
END;
/
```

Configure Database In-Memory and Load Database Objects

To test Database In-Memory, SPA will execute the SQL in the STS using the standard database structures and then using Database In-Memory. The first step is to implement the In-Memory column store and populate it with the objects recommended by the In-Memory Advisor.

There are two steps to populating the In-Memory Column Store with database objects, such as tables and partitions.

The first step is to designate objects that should be placed In-Memory with the ‘`inmemory`’ clause. In addition, you can specify a priority for the objects to be populated In-Memory with the ‘`priority`’ clause. A priority of ‘`critical`’ means that the object should be loaded In-Memory immediately after the database is restarted and before objects with a lower priority. A priority of ‘`none`’, which is the default, means the object will be loaded upon access, and after any other objects of higher priority have been loaded. For example, the following DML would specify the `lineitem` table should be placed In-Memory immediately after the database has been restarted:

```
alter table lineitem inmemory priority critical;
```

The Advisor produces a script which modifies the recommended tables and partitions so they will be placed in the In-Memory column store. Execute that script to appropriately modify your key database objects so they will be loaded into In-Memory.

The second step to configure the the In-Memory Column Store is to set the inmemory_size init.ora parameter to the size suggested by the Advisor. In-Memory size can only be changed with a database shutdown and restart: For example

alter system set inmemory_size=10G scope=spfile;

Then shutdown and restart the database.

Allow sufficient time after the database has restarted for the In-Memory column store to be populated. If some objects have priority 'none', you may have to run a full table scan of the object for it to be populated into the In-Memory column store. For example:

```
select /*+ full(orders) */ count(*) from orders;
```

You can monitor the population of the In-Memory column store from the Enterprise Manager 12c In-Memory Central page or by querying the V$IM_SEGMENTS view in the database.

Run the SPA trials

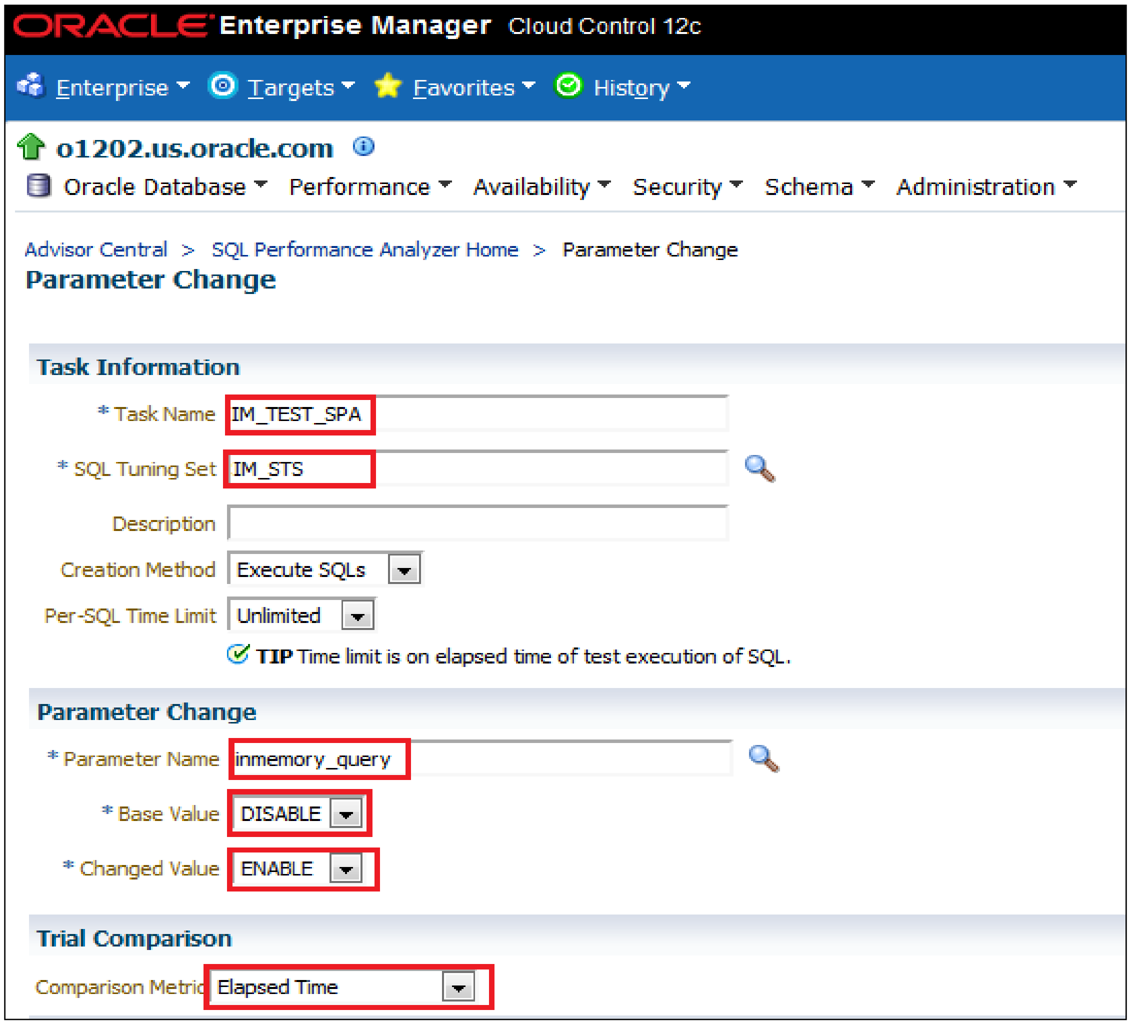
Once the column store is populated, run the SPA test. To test Database In-Memory, SPA executes the SQL in the STS, in one trial without using the In-Memory column store, and then a second time using the In-Memory column store. After the completion of the trials, SPA produces a report comparing the performance difference between the two.

On the SPA home page in Enterprise Manager 12c, there are a number of predefined workflows, along with a 'Guided Workflow'.

For testing Database In-Memory, the easiest workflow to use is the 'Parameter Change' workflow. When you choose the 'Parameter Change' workflow, you will be prompted for a few inputs:

- A name for the SPA Task
- The STS you've created containing the SQL to be tested with Database In-Memory
- The parameter you wish to test – 'inmemory_query'
- The two values of the parameter – 'DISABLE' and 'ENABLE'.
- The metric to be used for comparison between the two trials. Database In-Memory improves analytic query performance. A good metric to compare analytic query performance is 'Elapsed Time', since analytic queries typically take a substantial amount of time. Choose 'Elapsed Time' as the metric.

Your input screen should look similar to Figure 1, below:



Parameter Change

Task Information

* Task Name	IM_TEST_SPA
* SQL Tuning Set	IM_STS
Description	
Creation Method	Execute SQLs
Per-SQL Time Limit	Unlimited

TIP Time limit is on elapsed time of test execution of SQL.

Parameter Change

* Parameter Name	inmemory_query
* Base Value	DISABLE
* Changed Value	ENABLE

Trial Comparison

Comparison Metric	Elapsed Time

Figure 1. SPA Task Input

Click on the 'Submit' button to initiate the SPA task.

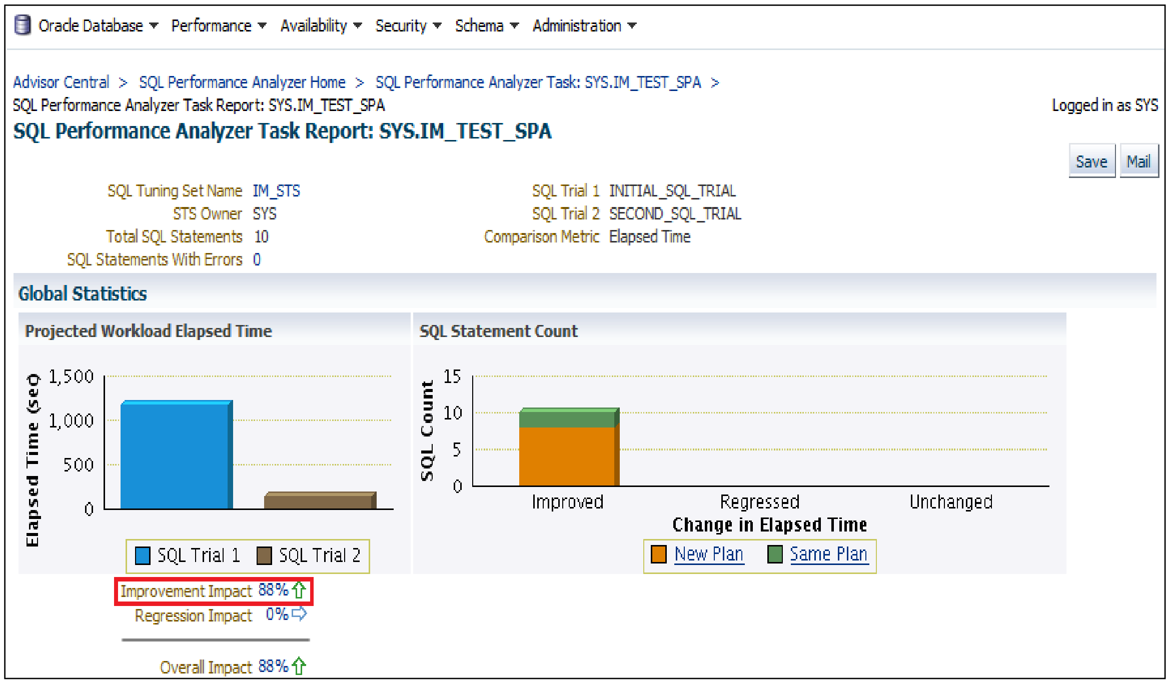
The SPA task should take some time to complete. You can monitor the progress of the SPA task from the SPA home page.

Evaluate SPA Results

When the two trials are completed, click on the 'View Latest Report' from the SPA home page. The report is a graphical report giving the results of the two trials.

In the top left of the report, is a summary result of the two trials, based on the chosen metric, along with the computed benefit of the change. To the right of the summary, is a bar chart showing the number of SQLs with improved, regessed and unchanged performance. The bar chart also indicates whether the SQLs have a new or unchanged execution plan.

Figure 2 below shows an example of the summary section:


Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾
Advisor Central > SQL Performance Analyzer Home > SQL Performance Analyzer Task: SYS.IM_TEST_SPA >
Save
Mail
Projected Workload Elapsed Time
Elapsed Time (sec)
1,500
1,000
500
0
SQL Trial 1
SQL Trial 2
SQL Statement Count
SQL Count
15
10
5
0
Improved
Regressed
Unchanged
Change in Elapsed Time
New Plan
Same Plan

SQL Performance Analyzer Task Report: SYS.IM_TEST_SPA

Logged in as SYS

SQL Performance Analyzer Task Report: SYS.IM_TEST_SPA

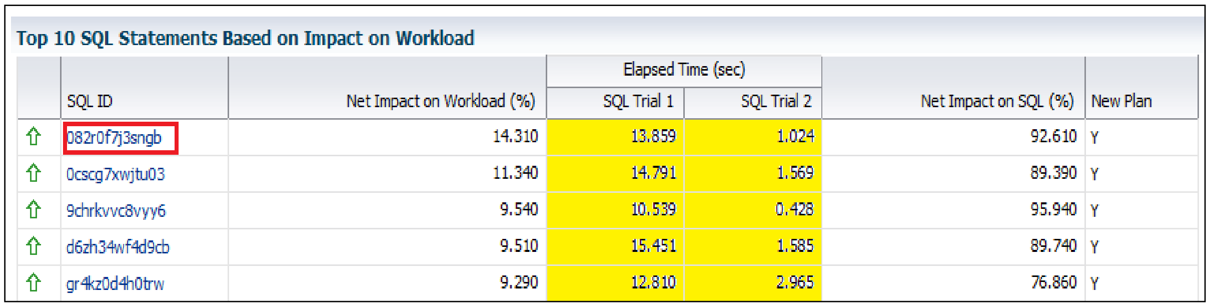
SQL Tuning Set Name	IM_STS	SQL Trial 1	INITIAL_SQL_TRIAL
STS Owner	SYS	SQL Trial 2	SECOND_SQL_TRIAL
Total SQL Statements	10	Comparison Metric	Elapsed Time
SQL Statements With Errors	0		

Global Statistics

Improvement Impact	88% ⇧
Regression Impact	0% ⇨
Overall Impact	88% ⇧

Figure 2. Summary of SPA results

Below the summary is a list of the top SQL that have the greatest impact on the workload. For each SQL, two measurements are listed. The first measurement is the change in performance of the individual SQL and the second is the net impact on the entire workload of the change in performance of this SQL. Figure 3 below shows an example of this section of the report:



Top 10 SQL Statements Based on Impact on Workload

			Elapsed Time (sec)			
	SQL ID	Net Impact on Workload (%)	SQL Trial 1	SQL Trial 2	Net Impact on SQL (%)	New Plan
⇧	082r0f7j3sngb	14.310	13.859	1.024	92.610	Y
⇧	0cscg7xwjtu03	11.340	14.791	1.569	89.390	Y
⇧	9chrkvvc8vyy6	9.540	10.539	0.428	95.940	Y
⇧	d6zh34wf4d9cb	9.510	15.451	1.585	89.740	Y
⇧	gr4kz0d4h0trw	9.290	12.810	2.965	76.860	Y

Figure 3. High impact SQL summary

You can drill down into more details of each SQL by clicking its SQL ID link.   That drill down will give you the execution plans and all metric details for that SQL.

To save the report to a shareable html format, click the 'Save' button at the top of the page.

Conclusion

The Database In-Memory Advisor makes specific, actionable recommendations about how to configure Database In-Memory and which database objects to place in it.   Use SQL Performance Analyzer for quick and easy validation that your workload will effectively use and benefit from Database In-Memory.

Oracle is committed to developing practices and products that help protect the environment

Oracle Database In-Memory Advisor Best Practices

February 2015
Author: Kurt Engeleiter

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0113

Hardware and Software, Engineered to Work Together