Oracle Maximum
Availability Architecture

Exadata Health and Resource Utilization Monitoring

System Baselining for Faster Problem Resolution

ORACLE WHITE PAPER | JANUARY 2017

Introduction

One of the most common causes of performance degradation is change - change in application logic and code, change in software and hardware/infrastructure configuration; even change in application usage patterns. Identifying that change, tracking it and correlating it with changes in application performance is a key technique in deterministically solving performance degradation.

To understand change, a solid baseline foundation is key and Oracle Enterprise Manager Cloud Control (Enterprise Manager) has a suite of tools for capturing current baseline information and using those captured metrics to quickly solve system performance problems. In general, the process records and validates system configuration data at all levels of the Exadata stack, validates databases schemas, and verifies application performance and resource usage is within defined ranges.

Configuration Management

An Exadata Machine encompasses many Enterprise Manager targets at multiple levels of the hardware and software stack, and when investigating performance problems within this stack, quickly identifying configuration changes can be a big time saver, especially when those changes may have happened without an administrator's knowledge. Enterprise Manager’s Configuration Comparison and Drift Management tools can help to set up saved configurations and baselines for any target of the Exadata machine, from top level software down to the lowest hardware component.

With Configuration and Drift Management administrators can troubleshoot or investigate problems where a change is suspected, ensure target configurations remain the same as reference or saved targets, and ensure targets of a similar target type within a system remain the same.

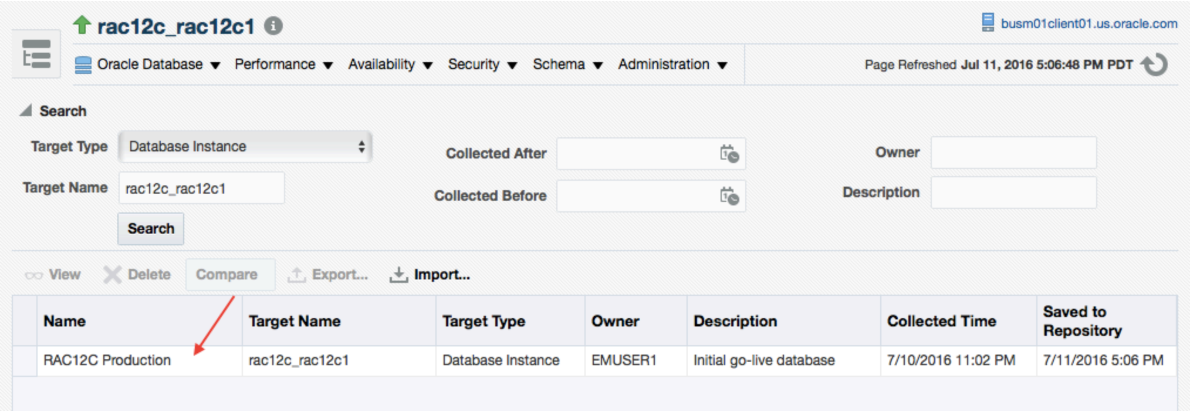
Configuration Management allows administrators to run manual one-time comparisons, run automated comparisons with email notification of changes, and save configurations for later use. Saved configurations can be useful to capture a “Gold Copy” of a full system stack, which would represent the standard configuration for a company’s architecture. When new systems come online, these new systems can be compared against the Gold Copy to ensure compliance with company standards. It is important to note that as systems change and evolve, such as with software or hardware upgrades or tech refreshes, new baselines should be captured to represent current proper configuration.

The basic Enterprise Manager setup captures a lot of configuration data to cover every target in the Exadata software and hardware stack, but it’s also very extendable and customizable for customer environments. Configuration Extensions allow this customization through SQL queries, file parsing and custom host executed commands. If you need a tailored environment, Configuration Extensions will further extend Enterprise Manager’s comprehensive functionality.

When running target comparisons, Templates define which configuration items to include or ignore. A template is associated with a specific target type, which determines the properties to be compared and also define constraints of acceptable values for specific properties. Included with Enterprise Manager are a set of comprehensive templates that can be copied and customized as needed to filter out any unwanted noise from known or expected differences in your architecture. A set of custom Exadata templates is also available through the following support document. These templates can also be used to compare Exadata targets across different Exadata machine versions (example: comparing X3-2 with X4-2)

- Enterprise Manager Configuration Comparison Templates for Exadata (Doc ID 2181282.1)

Every target in Enterprise Manager has a configuration, which is tracked by default with Drift Management. Administrators can use this Drift Management data to see what changes were made to a system over a defined window of time. When there is a need to be more focused on a particular aspect of your system, Saved Configurations allow administrators to be more specific with comparisons in conjunction with Comparison Templates. The following is an example of a saved configuration for a database instance, which among other items, will help keep track of changes to database initialization parameters.



Page Refreshed **Jul 11, 2016 5:06:48 PM PDT**

Target Type: Database Instance

Target Name: rac12c_rac12c1

Collected After:

Collected Before:

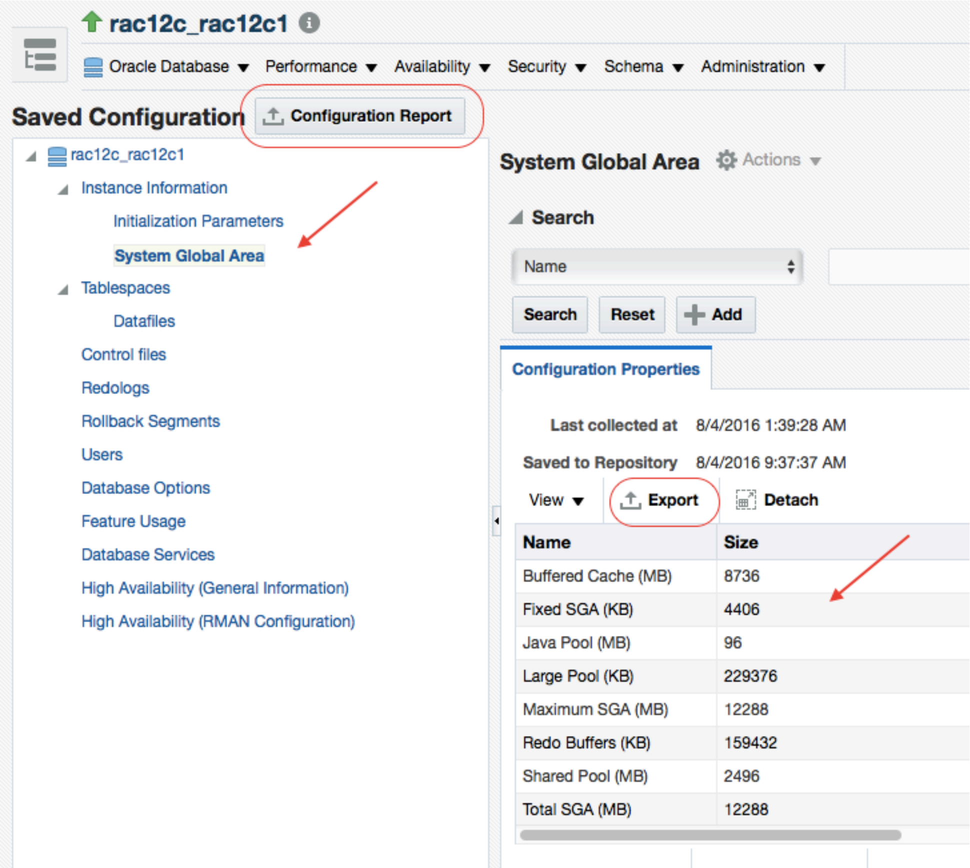
Owner:

Description:

Name	Target Name	Target Type	Owner	Description	Collected Time	Saved to Repository
RAC12C Production	rac12c_rac12c1	Database Instance	EMUSER1	Initial go-live database	7/10/2016 11:02 PM	7/11/2016 5:06 PM

Figure 1

Viewing a saved configuration presents a tree list of configuration categories. Selecting a category will display the individual category properties. In the example below, System Global Area properties are selected. A saved configuration can also be exported to an Excel spreadsheet for additional analysis using the “Configuration Report” button for a full report, or the “Export” button for the properties of the selected category only.



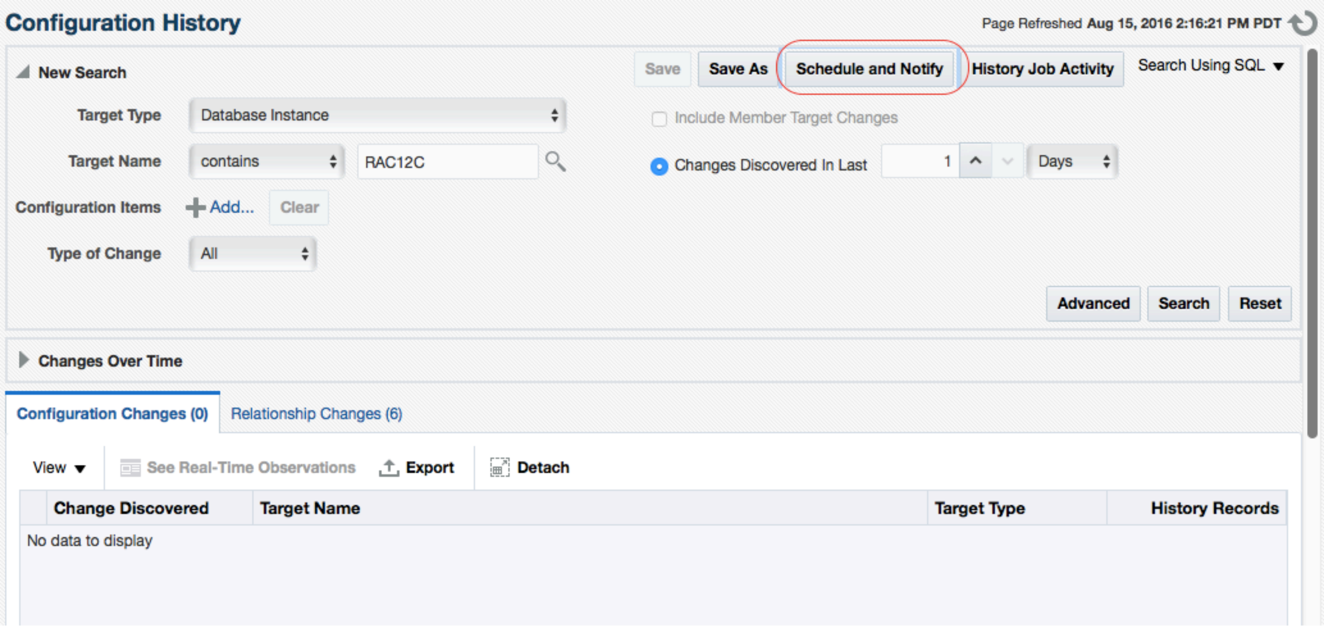
Last collected at 8/4/2016 1:39:28 AM

Saved to Repository 8/4/2016 9:37:37 AM

Name	Size
Buffered Cache (MB)	8736
Fixed SGA (KB)	4406
Java Pool (MB)	96
Large Pool (KB)	229376
Maximum SGA (MB)	12288
Redo Buffers (KB)	159432
Shared Pool (MB)	2496
Total SGA (MB)	12288

Figure 2

The Configuration History screens in Enterprise Manager can be used to actively query for any configuration changes that may have occurred in the past, track those changes over time, and configure email notification for any changes that have occurred. The example below tracks Database Instance changes which may have happened in the past 24 hours. Clicking on the “Schedule and Notify” button brings up a dialog box to schedule the running of this check and define email recipients.



Configuration History

Page Refreshed Aug 15, 2016 2:16:21 PM PDT

New Search

Target Type: Database Instance

Target Name: contains RAC12C

□ Include Member Target Changes

● Changes Discovered In Last 1 Days

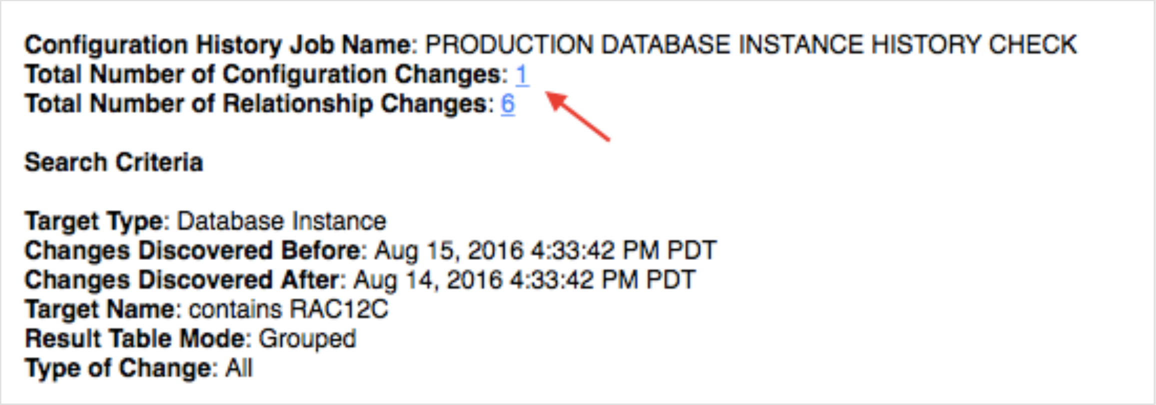
Type of Change: All

Configuration Changes (0)

Change Discovered	Target Name	Target Type	History Records
No data to display			

Figure 3

The email notification will give a summary of changes detected with a link to the Configuration History for the defined query.



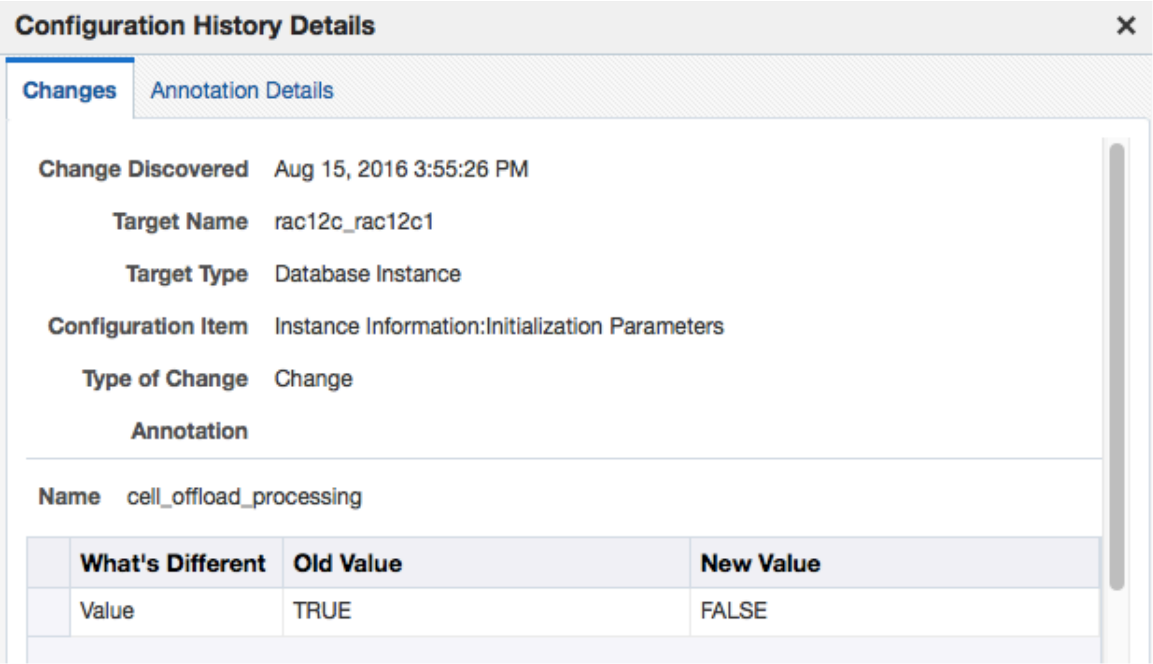
Configuration History Job Name: PRODUCTION DATABASE INSTANCE HISTORY CHECK
Total Number of Configuration Changes: 1
Total Number of Relationship Changes: 6

Search Criteria

Target Type: Database Instance
Changes Discovered Before: Aug 15, 2016 4:33:42 PM PDT
Changes Discovered After: Aug 14, 2016 4:33:42 PM PDT
Target Name: contains RAC12C
Result Table Mode: Grouped
Type of Change: All

Figure 4

The History screen provides the ability to drill down to identify the change that was made. In this example a database initialization parameter was altered.



Configuration History Details

Change Discovered Aug 15, 2016 3:55:26 PM

Target Name rac12c_rac12c1

Target Type Database Instance

Configuration Item Instance Information:Initialization Parameters

Type of Change Change

Annotation

Name cell_offload_processing

What's Different	Old Value	New Value
Value	TRUE	FALSE

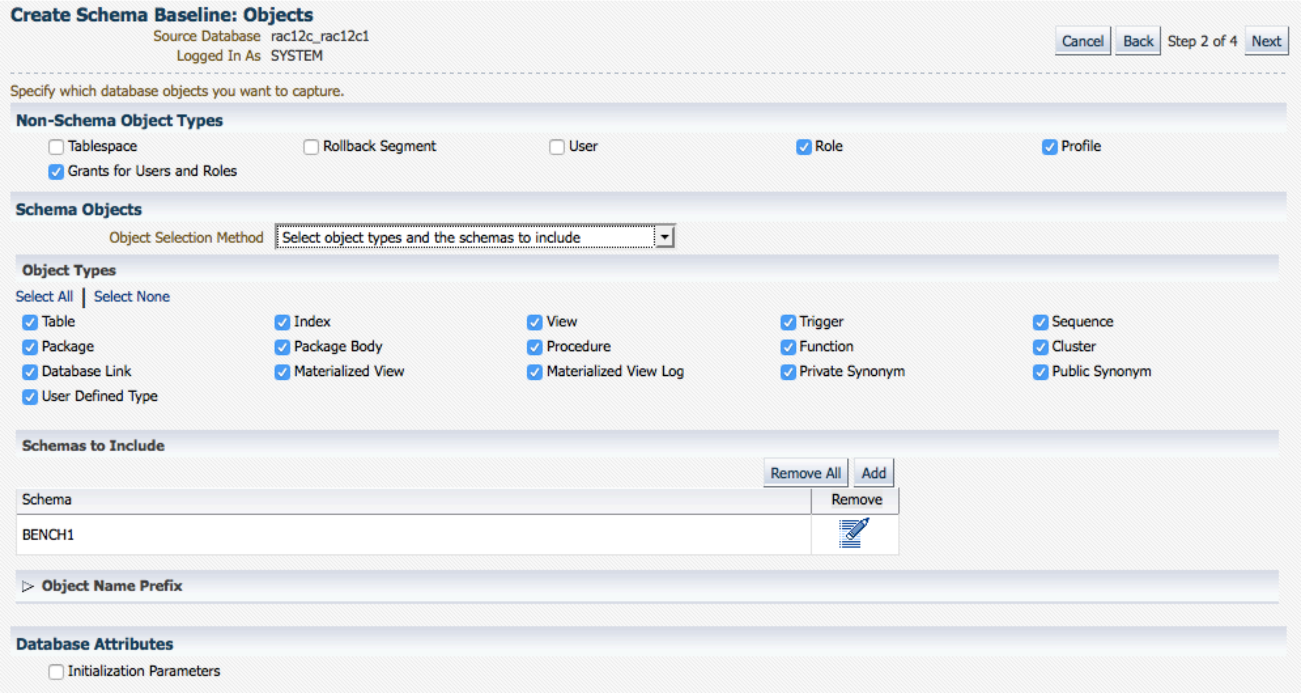
Figure 5

In summary, the Configuration Management tools within Oracle Enterprise Manager provide the means to track, record, notify and analyze changes for any target across the Enterprise.

Schema Baselines

Saving database schema baselines as an application evolves, from initial go-live to the current release version, is a great tool to track and report on schema changes over time. When comparing schemas, there are no Comparison Templates to use. Instead, the schema objects to compare are selected when creating the baseline. These schema objects will be used when running the compare. Items like indexes, packages and triggers, and also some supporting objects like profiles and roles can be included. As new versions of an application are released and it’s associated schema evolves, additional baselines should be captured to represent and keep up with the latest version of the application. Eventually you will have a documented history of schema changes, which can be used for other reporting purposes as well, like security and regulatory audits. To accommodate applications, which span schemas, multiple schemas can be included in a single baseline capture.

The following example creates a baseline for the schema BENCH1.



Create Schema Baseline: Objects

Source Database rac12c_rac12c1
Logged In As SYSTEM

Specify which database objects you want to capture.

Non-Schema Object Types

□ Tablespace □ Rollback Segment □ User ☑ Role ☑ Profile
☑ Grants for Users and Roles

Schema Objects

Object Selection Method: Select object types and the schemas to include

Object Types

☑ Table ☑ Index ☑ View ☑ Trigger ☑ Sequence
☑ Package ☑ Package Body ☑ Procedure ☑ Function ☑ Cluster
☑ Database Link ☑ Materialized View ☑ Materialized View Log ☑ Private Synonym ☑ Public Synonym
☑ User Defined Type

Schemas to Include

Schema	Remove
BENCH1	

▷ Object Name Prefix

Database Attributes

□ Initialization Parameters

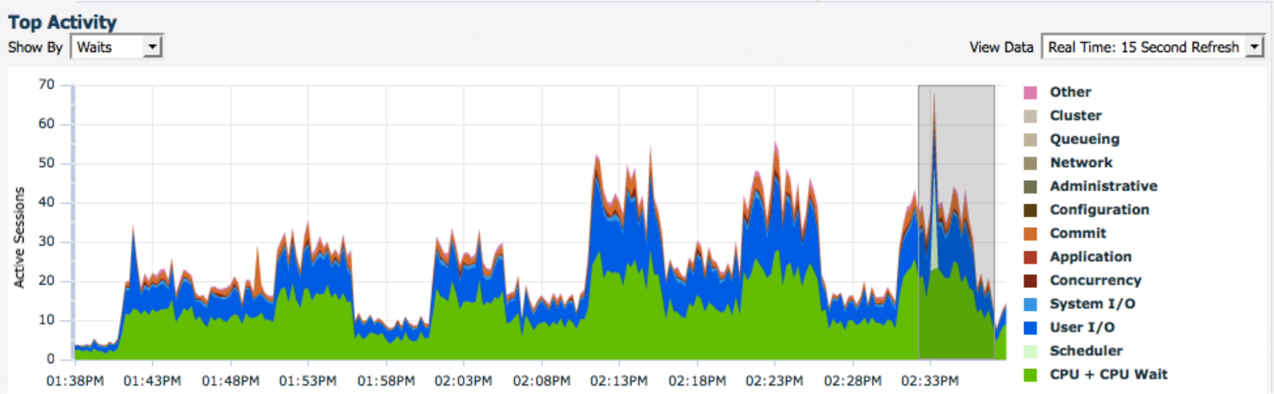
Figure 6

Workload Baselines

In Enterprise Manager, AWR Baselines can capture database resource usage for a defined time window, using static one-time baselines, a regularly scheduled repeating baseline or a rolling window. Using all three methods can provide additional options when comparing AWR reports.

Understanding application behavior will help in deciding which baseline methods will be most effective. The start and end times of the AWR Baselines will depend on this behavior. It may be fine to have a small baseline window of 10 minutes for consistent steady applications, but application behavior that might last for hours, like an extract, transform, and load (ETL) job, would require a larger window. When there are longer running regularly scheduled jobs with defined start times, capturing a static baseline of normal behavior and then setting up repeating scheduled baselines is a great way to look at past performance in comparison to recent activity. This can be useful for database loads which start late in the evening or early morning when administrators may not be regularly available.

As an example, for a steady workload that rises and falls depending on the number of users running queries, a baseline lasting 30 minutes would be sufficient to capture a typical load.

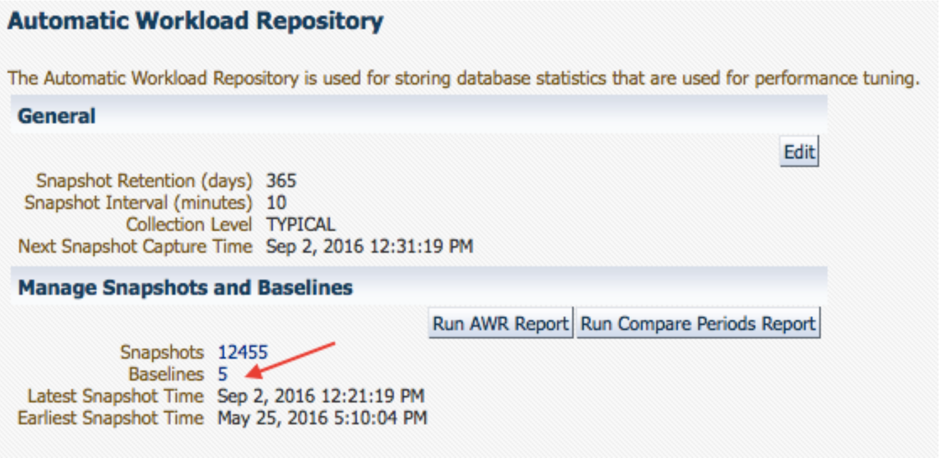

Top Activity

Show By Waits

View Data Real Time: 15 Second Refresh

Figure 7

To create AWR baselines, navigate to the AWR Administration screen and click on the baseline link.


Automatic Workload Repository

The Automatic Workload Repository is used for storing database statistics that are used for performance tuning.

General

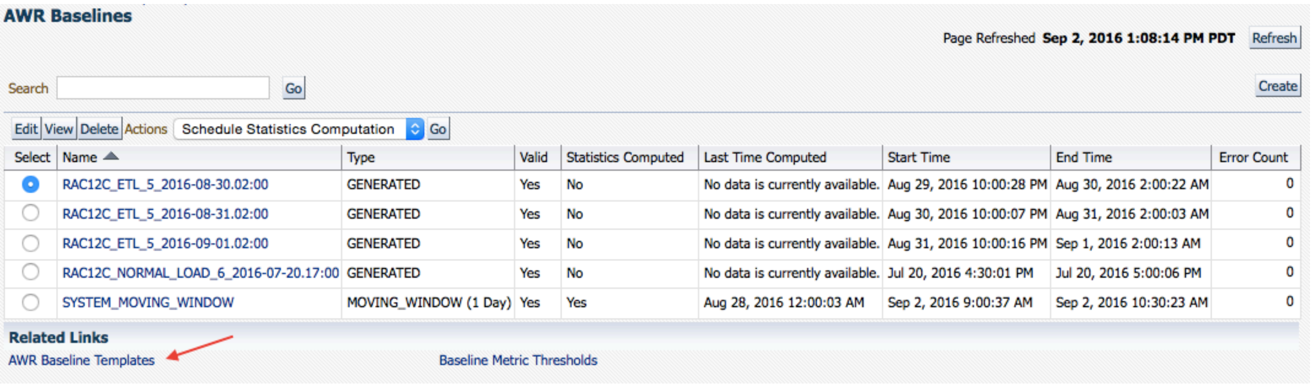
Snapshot Retention (days) 365
Snapshot Interval (minutes) 10
Collection Level TYPICAL
Next Snapshot Capture Time Sep 2, 2016 12:31:19 PM

Manage Snapshots and Baselines

Snapshots 12455
Baselines 5
Latest Snapshot Time Sep 2, 2016 12:21:19 PM
Earliest Snapshot Time May 25, 2016 5:10:04 PM

Figure 8

The example in the following figures consists of three rolling baselines which track nightly ETL jobs, a one-time baseline that captures 30 minutes of a typical day, and a daily rolling window, which is a default baseline provided by Enterprise Manager. These various baselines are defined by clicking on the Related Link at the bottom of the screen.



AWR Baselines

Page Refreshed **Sep 2, 2016 1:08:14 PM PDT**

Select	Name ▲	Type	Valid	Statistics Computed	Last Time Computed	Start Time	End Time	Error Count
●	RAC12C_ETL_5_2016-08-30.02:00	GENERATED	Yes	No	No data is currently available.	Aug 29, 2016 10:00:28 PM	Aug 30, 2016 2:00:22 AM	0
○	RAC12C_ETL_5_2016-08-31.02:00	GENERATED	Yes	No	No data is currently available.	Aug 30, 2016 10:00:07 PM	Aug 31, 2016 2:00:03 AM	0
○	RAC12C_ETL_5_2016-09-01.02:00	GENERATED	Yes	No	No data is currently available.	Aug 31, 2016 10:00:16 PM	Sep 1, 2016 2:00:13 AM	0
○	RAC12C_NORMAL_LOAD_6_2016-07-20.17:00	GENERATED	Yes	No	No data is currently available.	Jul 20, 2016 4:30:01 PM	Jul 20, 2016 5:00:06 PM	0
○	SYSTEM_MOVING_WINDOW	MOVING_WINDOW (1 Day)	Yes	Yes	Aug 28, 2016 12:00:03 AM	Sep 2, 2016 9:00:37 AM	Sep 2, 2016 10:30:23 AM	0

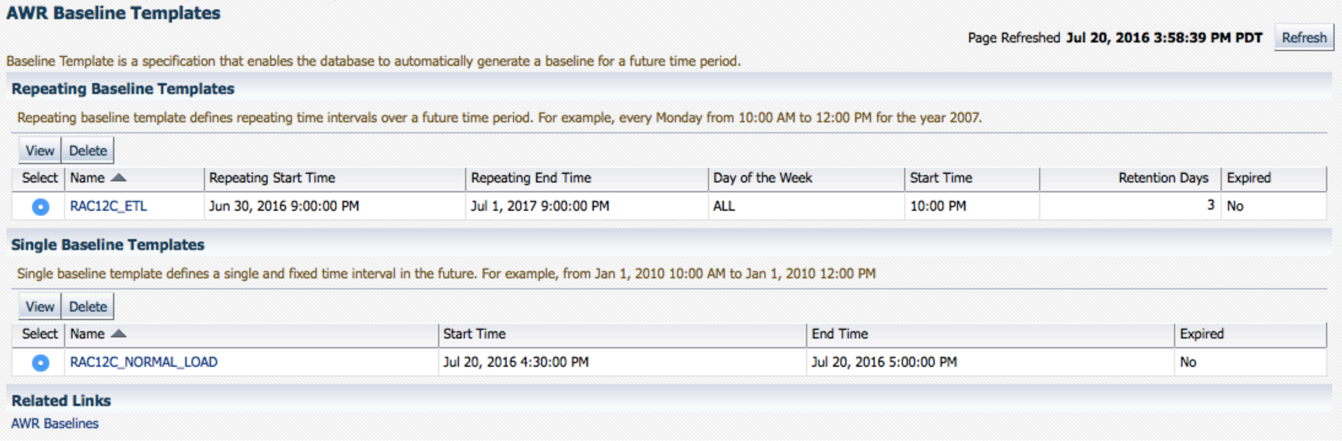
Related Links

AWR Baseline Templates

Baseline Metric Thresholds

Figure 9

The following example shows a repeating baseline template that captures an ETL job and a one-off static baseline template that is scheduled to capture a typical day's load over a 30-minute window.



AWR Baseline Templates

Page Refreshed **Jul 20, 2016 3:58:39 PM PDT**

Baseline Template is a specification that enables the database to automatically generate a baseline for a future time period.

Repeating Baseline Templates

Repeating baseline template defines repeating time intervals over a future time period. For example, every Monday from 10:00 AM to 12:00 PM for the year 2007.

Select	Name ▲	Repeating Start Time	Repeating End Time	Day of the Week	Start Time	Retention Days	Expired
●	RAC12C_ETL	Jun 30, 2016 9:00:00 PM	Jul 1, 2017 9:00:00 PM	ALL	10:00 PM	3	No

Single Baseline Templates

Single baseline template defines a single and fixed time interval in the future. For example, from Jan 1, 2010 10:00 AM to Jan 1, 2010 12:00 PM

Select	Name ▲	Start Time	End Time	Expired
●	RAC12C_NORMAL_LOAD	Jul 20, 2016 4:30:00 PM	Jul 20, 2016 5:00:00 PM	No

Related Links

AWR Baselines

Figure 10

Using multiple static and rolling window baselines, it is possible to capture typical application behavior at various times in the application life cycle and to use that information for comparison against encountered anomalies. These baselines should also be used to help configure monitoring and alerts for various Exadata targets. For more information on Exadata monitoring, see References.

Compliance With Exachk

Compliance Management in conjunction with Exadata's Exachk tool adds significant value by verifying your full Exadata stack is configured correctly based on Exadata Best Practices complied directly from Oracle's development team. Although not a true baseline tool, it does guarantee your systems will remain within Oracle's recommended Best Practice framework as new patches and software upgrades become available. Critical issues and related patches are also tracked for your system so you know right away if your systems are vulnerable and allow you to be proactive and plan upcoming maintenance accordingly.

Compliance results can be access through the Enterprise menu as shown in figure 11

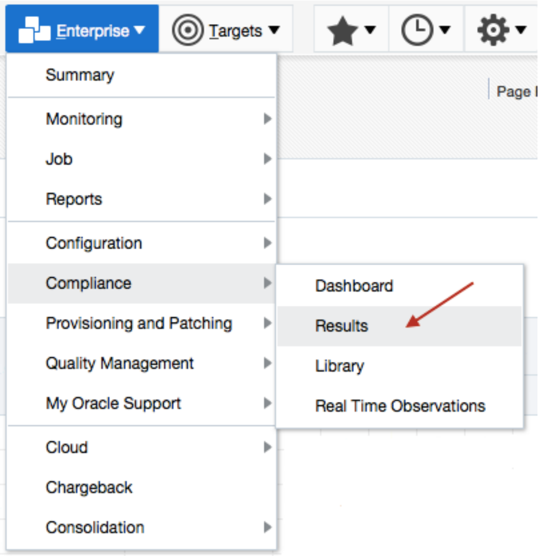
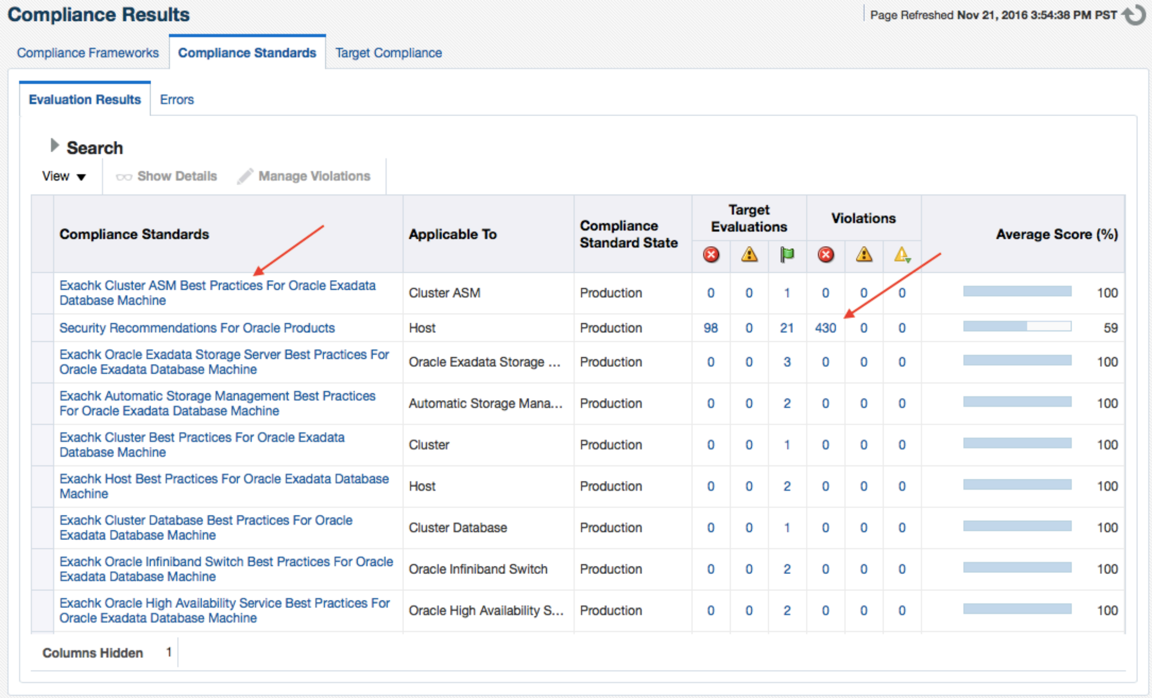


Figure 11

Compliance Standards are grouped by target type. You can further drill down into any problem areas by selecting a listed Standard, Target Evaluation or Violation.



Compliance Standards	Applicable To	Compliance Standard State	Target Evaluations			Violations			Average Score (%)
			✖	⚠	⚑	✖	⚠	⚠	
Exachk Cluster ASM Best Practices For Oracle Exadata Database Machine	Cluster ASM	Production	0	0	1	0	0	0	100
Security Recommendations For Oracle Products	Host	Production	98	0	21	430	0	0	59
Exachk Oracle Exadata Storage Server Best Practices For Oracle Exadata Database Machine	Oracle Exadata Storage ...	Production	0	0	3	0	0	0	100
Exachk Automatic Storage Management Best Practices For Oracle Exadata Database Machine	Automatic Storage Mana...	Production	0	0	2	0	0	0	100
Exachk Cluster Best Practices For Oracle Exadata Database Machine	Cluster	Production	0	0	1	0	0	0	100
Exachk Host Best Practices For Oracle Exadata Database Machine	Host	Production	0	0	2	0	0	0	100
Exachk Cluster Database Best Practices For Oracle Exadata Database Machine	Cluster Database	Production	0	0	1	0	0	0	100
Exachk Oracle Infiniband Switch Best Practices For Oracle Exadata Database Machine	Oracle Infiniband Switch	Production	0	0	2	0	0	0	100
Exachk Oracle High Availability Service Best Practices For Oracle Exadata Database Machine	Oracle High Availability S...	Production	0	0	2	0	0	0	100

Columns Hidden 1

Figure 12

When Performance Degrades

In the lifetime of an application, it is inevitable that there will be system problems at some level, so it is important to have a plan in place to quickly identify where the problem exists and work towards resolution. The Exadata Health and Resource Usage Monitoring whitepaper covers many aspects of this performance troubleshooting process, which includes:

1. **Hardware Alerts** - With Oracle's Auto Service Request technology, hardware problems will be quickly identified and scheduled for diagnostics or replacement if needed.
2. **Configuration Changes** - Enterprise Manager's Configuration Management tools expose any unknown differences in a system configuration for quick identification and resolution.
3. **Application Behavior** - AWR Report Baselining and comparisons help to identify changes in application behavior which result in changes to system resource usage and potential changes in performance.

Additionally, system problem details need to be tracked and reported, and Enterprise Manager's "Incident Manger" provides a central location from which to view, manage, diagnose, and resolve incidents. Details on the Incident Manager can be found in the Enterprise Manager Cloud Control Administrator's Guide.

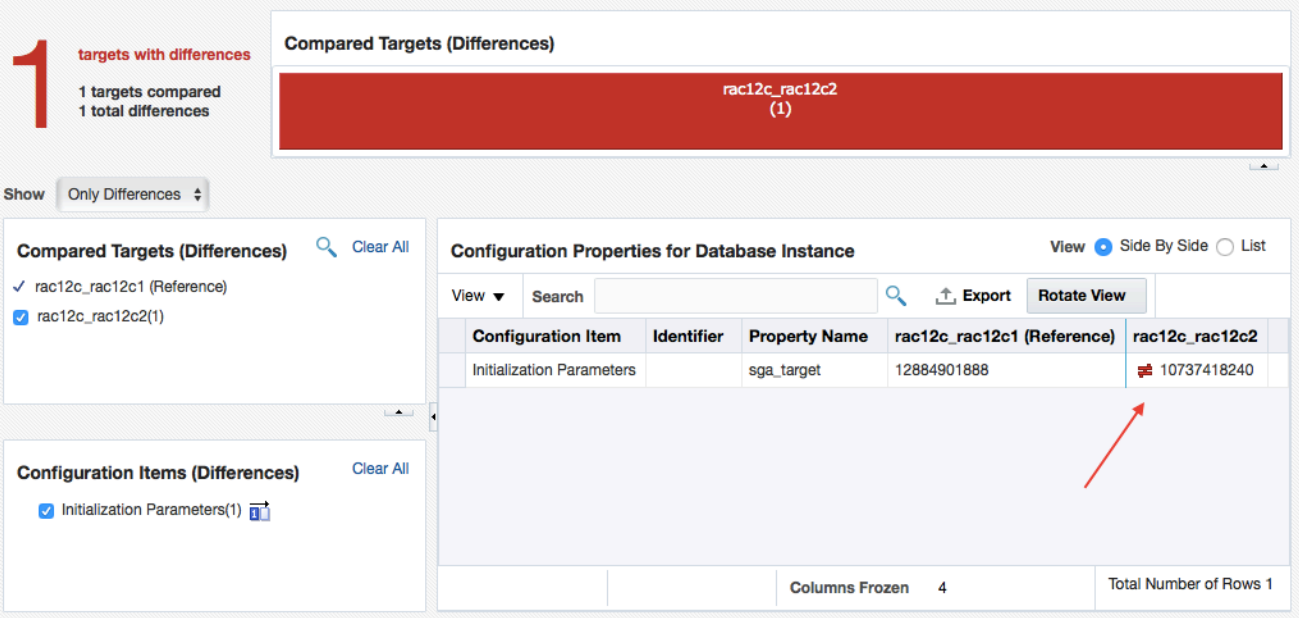
Capturing a system baseline allows for a repeatable documented approach to diagnosing performance problems when they occur. In addition to ruling out hardware faults, identified via ASR or through the steps defined in the Exadata Health and Resource Usage Monitoring whitepaper, comparing the current system to saved baselines allows fast identification of changes in configuration and workload.

When comparing baselines, the recommended approach is to:

1. Validate the current system against saved configuration baselines
2. Validate current schemas against schema baselines
3. Validate application behavior against AWR baselines

Validating configurations can be automated through Enterprise Manager's Drift and Consistency Management tools with a recommended proactive frequency of once a week.

The following is an example comparison using the Databases Instance Target Saved Configuration. There was a change made to the initialization parameter sga_target, but the change was not propagated to both database instances.



1 targets with differences

1 targets compared
1 total differences

Compared Targets (Differences)

rac12c_rac12c2
(1)

Show: Only Differences

Compared Targets (Differences)

✓ rac12c_rac12c1 (Reference)
☑ rac12c_rac12c2(1)

Configuration Items (Differences)

☑ Initialization Parameters(1)

Configuration Properties for Database Instance

View ● Side By Side ○ List

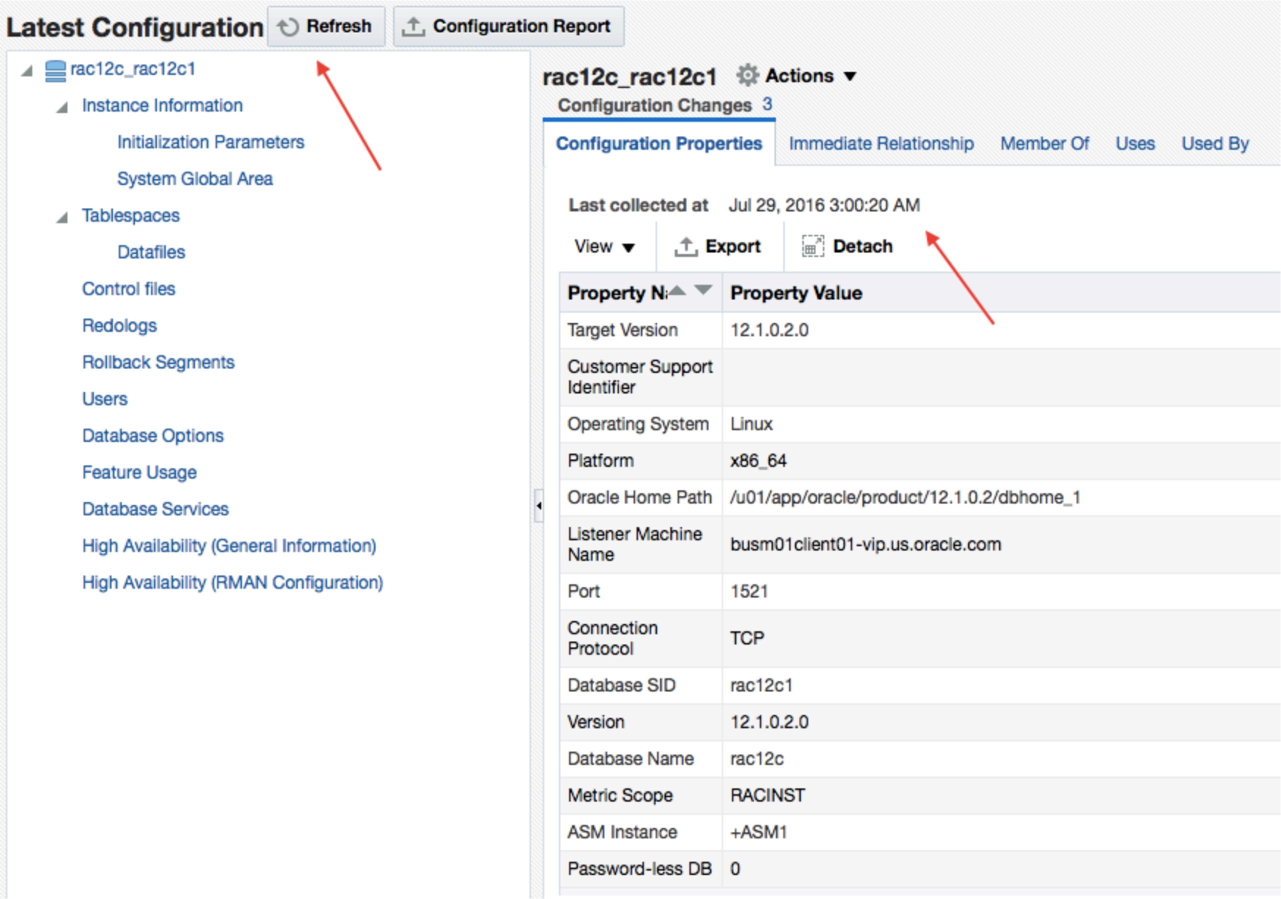
Configuration Item	Identifier	Property Name	rac12c_rac12c1 (Reference)	rac12c_rac12c2
Initialization Parameters		sga_target	12884901888	≠ 10737418240

Columns Frozen 4

Total Number of Rows 1

Figure 13

It's important to note that target configuration data is refreshed on a periodic basis, so any system configuration changes will not be immediately visible within Enterprise Manager. A manual refresh would be required to make those changes visible and to use them for any comparison. The Latest Configuration screen for a target includes the Last Collected date/time and also provides a means to refresh the configuration data.

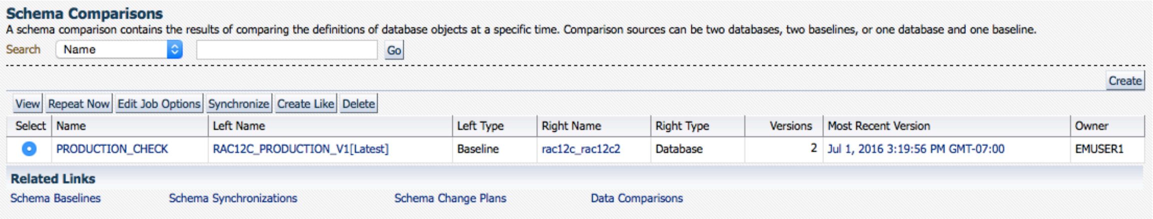


Last collected at Jul 29, 2016 3:00:20 AM

Property N	Property Value
Target Version	12.1.0.2.0
Customer Support Identifier	
Operating System	Linux
Platform	x86_64
Oracle Home Path	/u01/app/oracle/product/12.1.0.2/dbhome_1
Listener Machine Name	busm01client01-vip.us.oracle.com
Port	1521
Connection Protocol	TCP
Database SID	rac12c1
Version	12.1.0.2.0
Database Name	rac12c
Metric Scope	RACINST
ASM Instance	+ASM1
Password-less DB	0

Figure 14

Using the previously saved Schema Baseline, administrators can easily check at the schema level for any changes that may have occurred to cause performance problems. Each schema comparison is saved for historical tracking and later review. The example below depicts a schema comparison using the saved baseline against the current live schema.



Schema Comparisons

A schema comparison contains the results of comparing the definitions of database objects at a specific time. Comparison sources can be two databases, two baselines, or one database and one baseline.

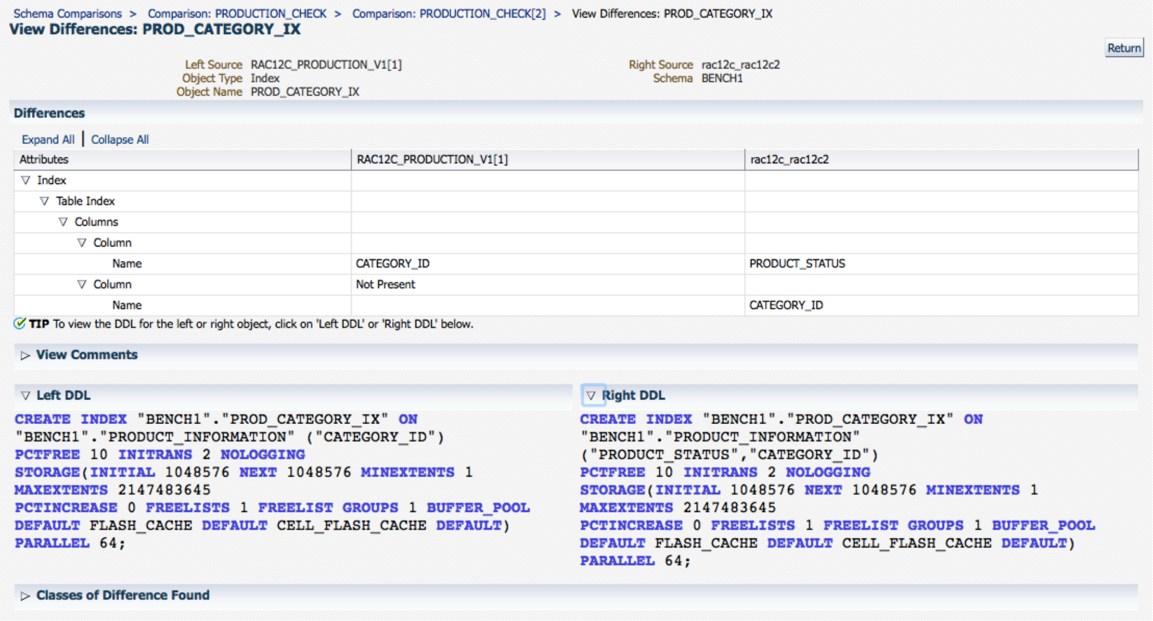
Search: Name

Select	Name	Left Name	Left Type	Right Name	Right Type	Versions	Most Recent Version	Owner
●	PRODUCTION_CHECK	RAC12C_PRODUCTION_V1[Latest]	Baseline	rac12c_rac12c2	Database	2	Jul 1, 2016 3:19:56 PM GMT-07:00	EMUSER1

Related Links

Figure 15

Drilling down further exposes an index with changes made to the leading column, potentially changing an optimizer execution plan. This level of detail also allows quick development of a rollback or roll-forward strategy if necessary, based on SQL which is easily accessible.



Schema Comparisons > Comparison: PRODUCTION_CHECK > Comparison: PRODUCTION_CHECK[2] > View Differences: PROD_CATEGORY_IX

View Differences: PROD_CATEGORY_IX

Left Source: RAC12C_PRODUCTION_V1[1]
Object Type: Index
Object Name: PROD_CATEGORY_IX

Right Source: rac12c_rac12c2
Schema: BENCH1

Differences

Expand All | Collapse All

Attributes	RAC12C_PRODUCTION_V1[1]	rac12c_rac12c2
▽ Index		
▽ Table Index		
▽ Columns		
▽ Column		
Name	CATEGORY_ID	PRODUCT_STATUS
▽ Column	Not Present	
Name		CATEGORY_ID

✓ **TIP** To view the DDL for the left or right object, click on 'Left DDL' or 'Right DDL' below.

▷ **View Comments**

▽ **Left DDL**

```
CREATE INDEX "BENCH1"."PROD_CATEGORY_IX" ON
"BENCH1"."PRODUCT_INFORMATION" ("CATEGORY_ID")
PCTFREE 10 INITRANS 2 NOLOGGING
STORAGE(INITIAL 1048576 NEXT 1048576 MINEXTENTS 1
MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
PARALLEL 64;
```

▽ **Right DDL**

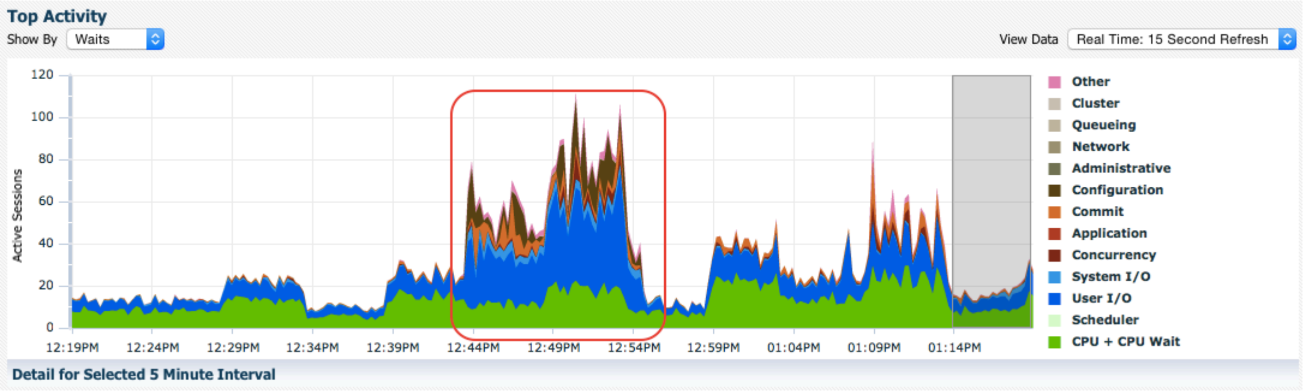
```
CREATE INDEX "BENCH1"."PROD_CATEGORY_IX" ON
"BENCH1"."PRODUCT_INFORMATION"
("PRODUCT_STATUS","CATEGORY_ID")
PCTFREE 10 INITRANS 2 NOLOGGING
STORAGE(INITIAL 1048576 NEXT 1048576 MINEXTENTS 1
MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
PARALLEL 64;
```

▷ **Classes of Difference Found**

Figure 16

After ruling out any changes to the system's configuration, compare the load characteristics. Using saved AWR Baselines can help identify key areas of the system in which load characteristics have changed significantly and the resultant Workload Repository Compare Period Report highlights key areas in system resource usage to help easily identify those differences.

As an example, in this Top Activity graph something changed over a 10-minute window.



Top Activity

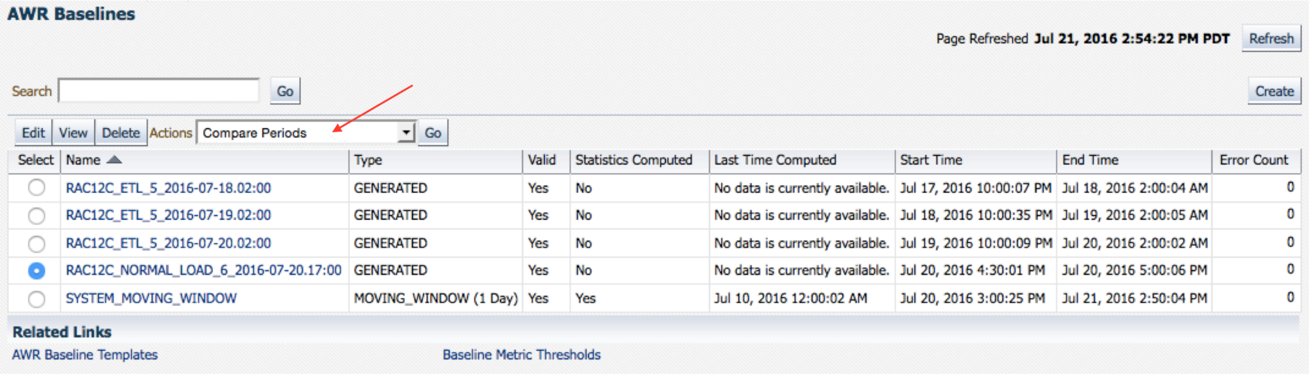
Show By Waits

View Data Real Time: 15 Second Refresh

Detail for Selected 5 Minute Interval

Figure 17

To investigate further, use the saved AWR Baseline of a normal load to make a comparison.



AWR Baselines

Page Refreshed **Jul 21, 2016 2:54:22 PM PDT** Refresh

Search Go Create

Edit View Delete Actions Compare Periods Go

Select	Name ▲	Type	Valid	Statistics Computed	Last Time Computed	Start Time	End Time	Error Count
○	RAC12C_ETL_5_2016-07-18.02:00	GENERATED	Yes	No	No data is currently available.	Jul 17, 2016 10:00:07 PM	Jul 18, 2016 2:00:04 AM	0
○	RAC12C_ETL_5_2016-07-19.02:00	GENERATED	Yes	No	No data is currently available.	Jul 18, 2016 10:00:35 PM	Jul 19, 2016 2:00:05 AM	0
○	RAC12C_ETL_5_2016-07-20.02:00	GENERATED	Yes	No	No data is currently available.	Jul 19, 2016 10:00:09 PM	Jul 20, 2016 2:00:02 AM	0
●	RAC12C_NORMAL_LOAD_6_2016-07-20.17:00	GENERATED	Yes	No	No data is currently available.	Jul 20, 2016 4:30:01 PM	Jul 20, 2016 5:00:06 PM	0
○	SYSTEM_MOVING_WINDOW	MOVING_WINDOW (1 Day)	Yes	Yes	Jul 10, 2016 12:00:02 AM	Jul 20, 2016 3:00:25 PM	Jul 21, 2016 2:50:04 PM	0

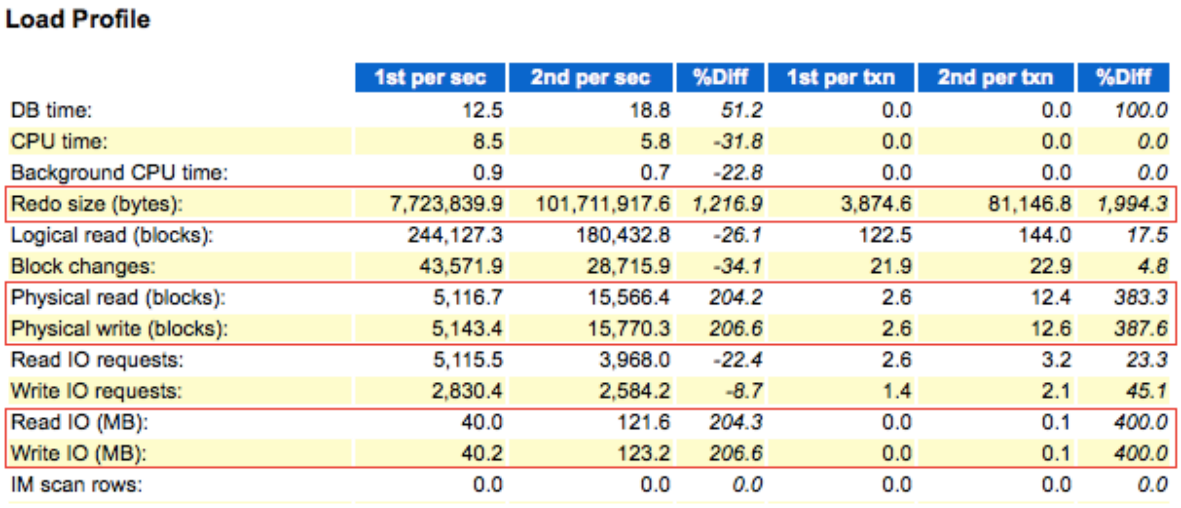
Related Links

AWR Baseline Templates

Baseline Metric Thresholds

Figure 18

Although the Workload Repository Compare Period Report contains a lot of information, the Top Timed Events and Load Profile sections are the best places to start identifying differences at a high level. In this example using the Load Profile, differences in IO usage are clearly visible.

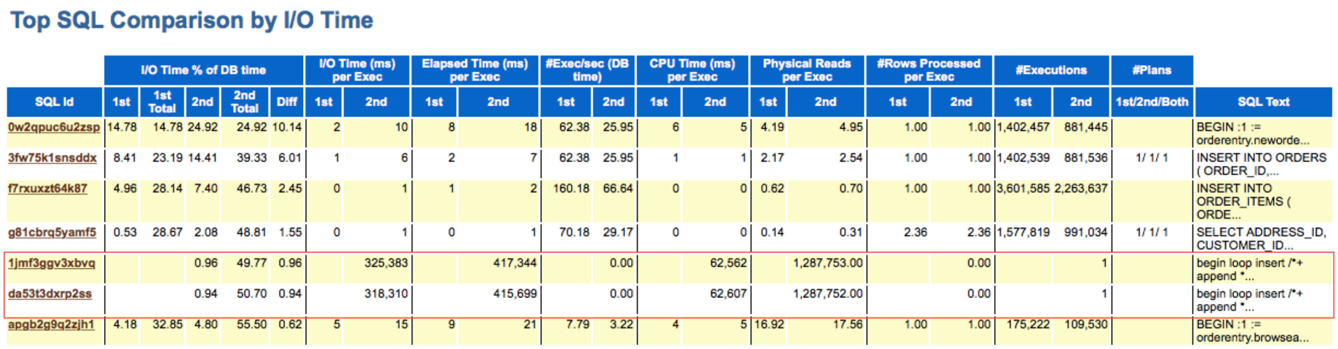


Load Profile

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	12.5	18.8	*51.2*	0.0	0.0	*100.0*
CPU time:	8.5	5.8	*-31.8*	0.0	0.0	*0.0*
Background CPU time:	0.9	0.7	*-22.8*	0.0	0.0	*0.0*
Redo size (bytes):	7,723,839.9	101,711,917.6	*1,216.9*	3,874.6	81,146.8	*1,994.3*
Logical read (blocks):	244,127.3	180,432.8	*-26.1*	122.5	144.0	*17.5*
Block changes:	43,571.9	28,715.9	*-34.1*	21.9	22.9	*4.8*
Physical read (blocks):	5,116.7	15,566.4	*204.2*	2.6	12.4	*383.3*
Physical write (blocks):	5,143.4	15,770.3	*206.6*	2.6	12.6	*387.6*
Read IO requests:	5,115.5	3,968.0	*-22.4*	2.6	3.2	*23.3*
Write IO requests:	2,830.4	2,584.2	*-8.7*	1.4	2.1	*45.1*
Read IO (MB):	40.0	121.6	*204.3*	0.0	0.1	*400.0*
Write IO (MB):	40.2	123.2	*206.6*	0.0	0.1	*400.0*
IM scan rows:	0.0	0.0	*0.0*	0.0	0.0	*0.0*

Figure 19

Drilling down to the SQL layer in relation to IO, we see two queries are identified that are not part of the normal AWR baseline and thus not part of a normal daily workload. If these SQL statements were discovered to be a new addition to the daily load profile, a new AWR baseline should be captured to include this new load.



Top SQL Comparison by I/O Time

	I/O Time % of DB time					I/O Time (ms) per Exec		Elapsed Time (ms) per Exec		#Exec/sec (DB time)		CPU Time (ms) per Exec		Physical Reads per Exec		#Rows Processed per Exec		#Executions		#Plans	
SQL Id	1st	1st Total	2nd	2nd Total	Diff	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st/2nd/Both	SQL Text
0w2qpuc6u2zsp	14.78	14.78	24.92	24.92	10.14	2	10	8	18	62.38	25.95	6	5	4.19	4.95	1.00	1.00	1,402,457	881,445		BEGIN :1 := orderentry.neworde...
3fw75k1snsddx	8.41	23.19	14.41	39.33	6.01	1	6	2	7	62.38	25.95	1	1	2.17	2.54	1.00	1.00	1,402,539	881,536	1/ 1/ 1	INSERT INTO ORDERS (ORDER_ID,...
f7rxuxzt64k87	4.96	28.14	7.40	46.73	2.45	0	1	1	2	160.18	66.64	0	0	0.62	0.70	1.00	1.00	3,601,585	2,263,637		INSERT INTO ORDER_ITEMS (ORDE...
g81cbrq5yamf5	0.53	28.67	2.08	48.81	1.55	0	1	0	1	70.18	29.17	0	0	0.14	0.31	2.36	2.36	1,577,819	991,034	1/ 1/ 1	SELECT ADDRESS_ID, CUSTOMER_ID...
1jmf3ggv3xbvq			0.96	49.77	0.96		325,383		417,344		0.00		62,562		1,287,753.00		0.00		1		begin loop insert /*+ append *...
da53t3dxrp2ss			0.94	50.70	0.94		318,310		415,699		0.00		62,607		1,287,752.00		0.00		1		begin loop insert /*+ append *...
apgb2g9q2zjh1	4.18	32.85	4.80	55.50	0.62	5	15	9	21	7.79	3.22	4	5	16.92	17.56	1.00	1.00	175,222	109,530		BEGIN :1 := orderentry.browsea...

Figure 20

In this example the newly added SQL is easily identified, but there will be times when the differences are subtle and Enterprise Manager has additional tools to help with this analysis. This toolset includes the SQL Tuning Advisor, the SQL Access Advisor, SQL Tuning Sets and SQL Plan Control. More information on these tools can be found in the Database SQL Tuning Guide as part of the larger Database Administration documentation for your databases.

Summary

Managing consistency and compatibility across large hardware and software deployments across the enterprise at Cloud scale can be a difficult task. Enterprise Manager helps to simplify this task and enables administrators to view, save, track, and compare configuration information for all managed targets, and provide auditing as an automated process which can be further extended for incident ticketing systems. Using this information and taking time to capture the physical architecture, work load characteristics, and resource usage metrics of the Exadata Database Machine will greatly enhance the speed at which administrators can solve performance discrepancies. Proactive monitoring of configuration changes will also further reduce unforeseen downtime and potential outages, providing the benefit of long term stability in a controlled stable environment.

References

This paper is part of the larger Maximum Availability Architecture document series for Exadata available at: http://oracle.com/goto/maa.

For more information, see the following related publications:

Enterprise Manager Exadata Management - Getting Started Guide

Exadata Health and Resource Usage Monitoring

Exadata Health and Resource Utilization Monitoring - Exadata Storage Server KPIs

Exadata Health and Resource Utilization Monitoring - Adaptive Thresholds

Additional Exadata Target Comparison Templates

Enterprise Manager Configuration Comparison Templates for Exadata (Doc ID 2181282.1)

ORACLE®

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

B blogs.oracle.com/oracle

f facebook.com/oracle

twitter.com/oracle

oracle.com

Integrated Cloud Applications & Platform Services

Exadata Health and Resource Utilization Monitoring: System Baselining for Faster Problem Resolution January 2017
Author: Kurt Schmutz