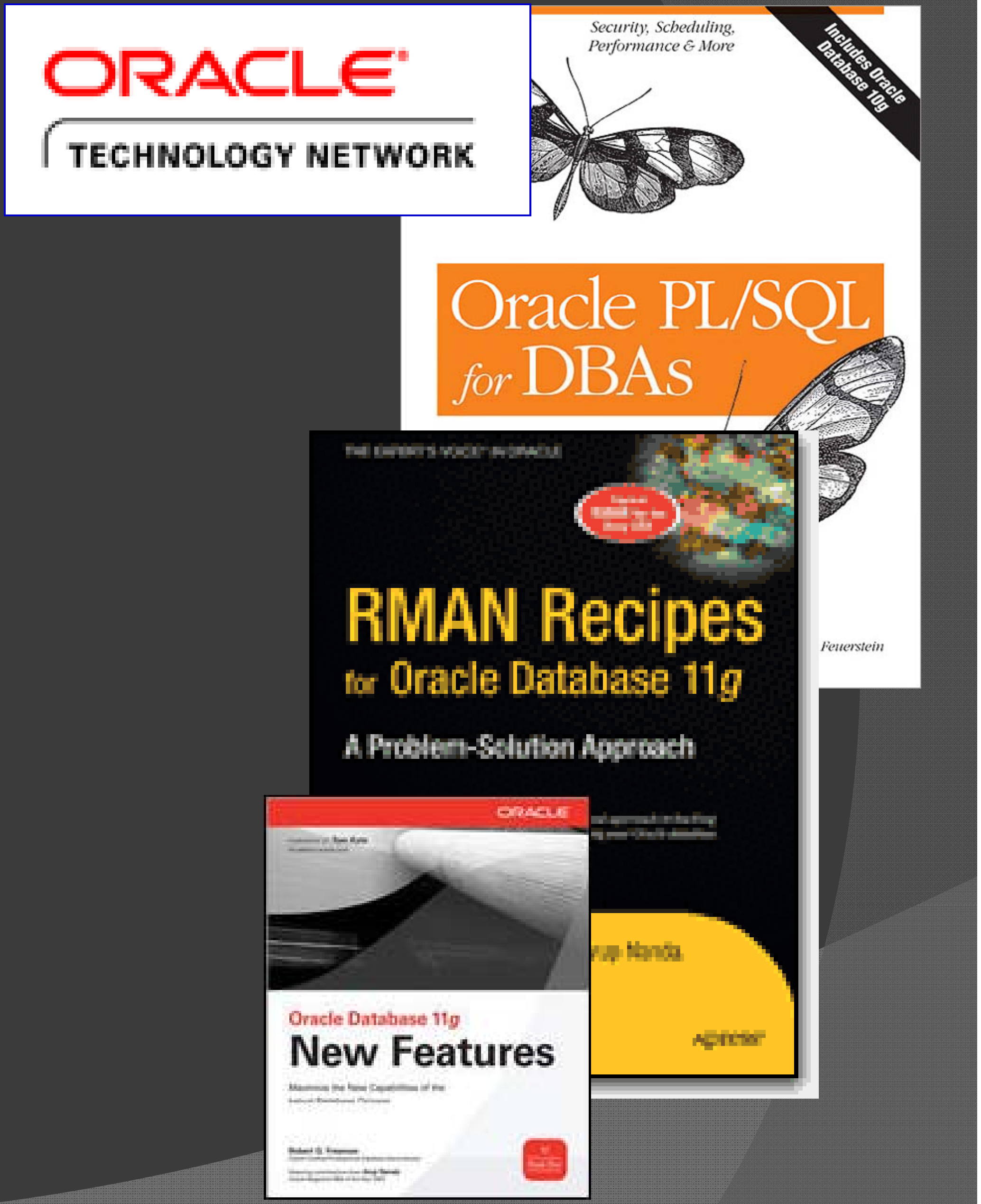
Arup Nanda

LOW-RISK 11G UPGRADE USING RAT, SNAPSHOT STANDBY AND PLAN BASELINES

About Me

- Oracle DBA for 16 years and counting
- Speak at conferences, write articles, 4 books
- Brought up the Global Database Group at Starwood Hotels, in White Plains, NY


ORACLE
TECHNOLOGY NETWORK
Security, Scheduling, Performance & More
Includes Oracle Database 10g
Oracle PL/SQL for DBAs
Feuerstein
RMAN Recipes for Oracle Database 11g
A Problem-Solution Approach
Apress
ORACLE
Oracle Database 11g
New Features

What You will Learn

- A rehash of our 11g Upgrade Experience
- What challenges lay during our upgrade
- What tools are available with Oracle
- How we used these tools to meet these challenges

The information is for educational purpose only; not professional advise or consultation. Starwood and the speaker make no warranty about the accuracy of the content and assume no responsibility for the consequence of the actions shown in the slides.

Must Read

- MetaLink Note 429825.1 shows the steps for a manual upgrade
- MetaLink Note 601807.1 Upgrade Companion for 11gR1: a one stop paper for upgrade.
  - Note 837570.1 for 11gR2
- A very important step [*Never* skip it] – check for dangling dictionary objects – MetaLink 579523.1

Database Details

- A lot of applications; not just one
  - A lot of business processes; not just a few
- Very critical business functionalities
  - A high $ amount attributed to downtime or slowness (which also translates to downtime since the apps time out)
- Version 10.2.0.4 was pre-upgrade

Pre/Post-Upgrade

Pre-Upgrade	Post-Upgrade
10.2.0.4	11.1.0.7
No Flash Recovery Area	Flash Recovery Area
Flashback not Enabled	Flashback Enabled
Non-OMF	Oracle Managed Files
Older hardware	Newer hardware
No partitioning	Partitioning
No Compression	Compression
Some parameters	Changed params
Linux RHAS 4	Linux RHAS 5

The $zillion Question

- *If it ain't broken don't fix it* – is generally the mantra
- *Must* Have Answers
  - What will happen – will the database at least perform as much as right now, or it might be worse?
  - How do we know?
  - How certain are we?

Why Worse?

- Optimizer Plans could be change for better (or, *worse*) – performance related
- Functionality may have changed, producing unexpected results
- New bugs may be encountered for which there will be no patches, at least not immediately
- Some new functionality may require further attention

The Usual Plan

- Create new environment (pre-prod) and run the production-*like* events there and examine the performance
- The key is it is “production-like”; not *actual* events that occurred in production.
- Usually synthetic, concocted

So, what’s Problem?

- Synthetic transactions are not faithful reproductions of the events that actually happened
- They are mechanized and repeatable, but do not capture production dynamics
- Concocted ones do not take into account unique data values.
  - Example: name searches are more on “Johnson” in New York while in Los Angeles, it’s “Lee”

The Ugly Truth

- Will the database work the same way (if not better) after the upgrade?
- Synthetic transactions will *not* give you the answer
- To get that, you must ask the users to redo the activities exactly how they did in real production
  - In the same order, using the same breaks in between!

Challenges

- Building a test system
  - quickly, easily, accurately, repeatedly
- Dry runs of Upgrades
- Ensuring performance
  - Repeating the activities of the production
    - accurately
    - without impacting production
- Impact of new parameters

Additional Challenges

- You want to change something else during the upgrade (since you have an outage)
  - Convert to RAC
  - Storage to ASM
  - Change buffer pools
  - Change some parameter, such as cursor_sharing
  - Take advantage of new features, e.g. LOBs to Securefiles

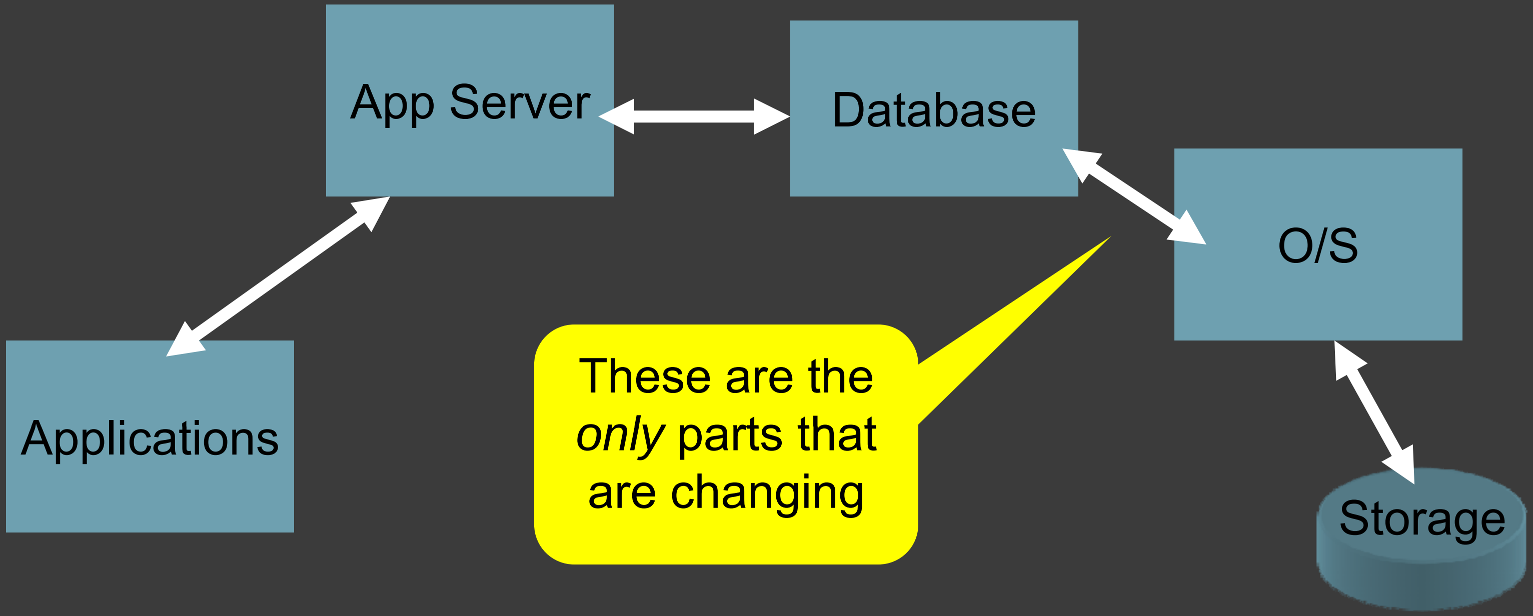
Tools at your Disposal

- Database Replay
- SQL Performance Analyzer
- SQL Tuning Advisor
- SQL Plan Management
- Easier Standby Building
- Snapshot Standby
- Switching between Physical and Logical

Concocting Prod Work

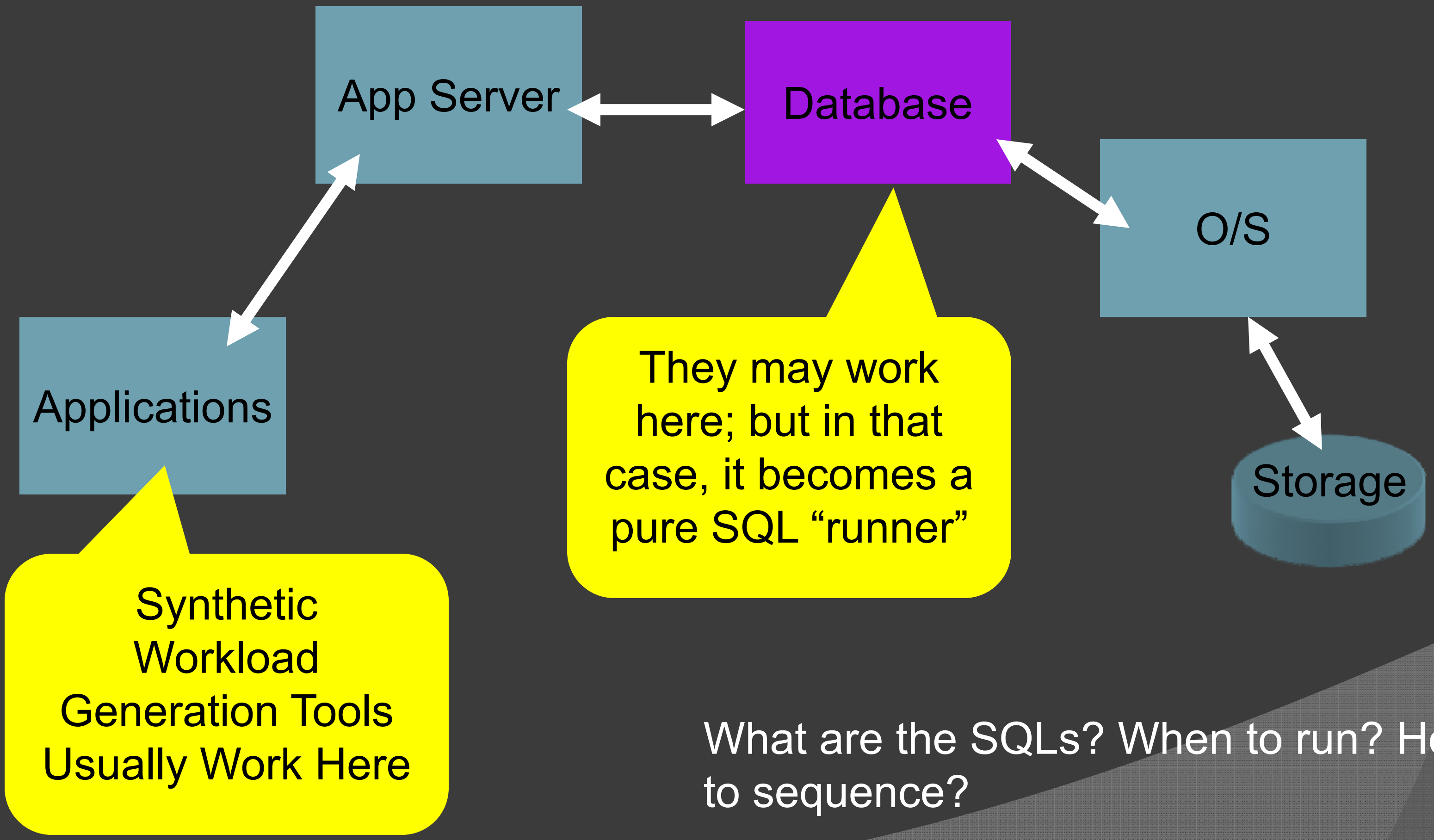
- Workload generator tools such as Load Runner can simulate user actions,
  - Capture clickstream on a webpage
  - Databank parameters to simulate load
  - Coverage for important workflows only
- Upgrade involves only one changed part
  - Application -> App Server -> Database
  - So, there is no need to test the entire stack
- Cost of QA is *not* insignificant
- Availability of QA is not automatic

Parts of a System



These are the *only* parts that are changing

Workload Generation Tools



App Server

Database

O/S

Storage

Applications

They may work here; but in that case, it becomes a pure SQL “runner”

Synthetic Workload Generation Tools Usually Work Here

What are the SQLs? When to run? How to sequence?

Questions for “SQL Running”

- What SQLs were executed
- How often was each one executed
  - Determines parsing, buffer cache hits, etc.
- In what order were they executed
  - Determines buffer hits
- How much was the time between them
  - Determines buffer hits, parsing
- What optimizer environment was in effect
  - Someone sets DBFMBRC before running an SQL and then resets it to default
- Are sequence numbers guaranteed?

Capturing the Work

- SQL Trace
  - Captures the SQL statements, in order, with plans
- 10046 Trace
  - Captures the SQL statements with timestamp
  - Bind variables
- 10053 Trace
  - Captures the optimizer environment
- But, how will you put the information from all this together to produce something that is:
  - Executable
  - Repeatable

Database Replay

- This is where Database Replay really shines
- It captures the actual transactions from the production system, in the same order, with the same breaks in between
- It's as if the users are redoing the same activities in front of the test system
- Even sequence numbers are fetched the same way they occurred in production
- No primary key violation

Workload Capture

- The package dbms_workload_capture captures workload from current production
- The package exists in 11g, so what about 10g?
- In 10.2.0.4 it exists
- For earlier versions, a patch needs to be applied
  - Refer to MetaLink Note 560977.1 for details
- The easiest is to use Enterprise Manager Grid Control
- Grid Control 10.2.0.5 has the toolkit

Steps

- Capture Workload
  - It produces a set of files with extension *.rec
- Move them to the 11g system
- Use Replay feature in command line or EM to replay the activities
- Both these activities take AWR snapshots before and after events.
  - Use AWR Compare Period Report to compare the performance.
- A complete detailed article on Database Replay is on OTN: http://www.oracle.com/technology/oramag/oracle/08-jan/o18dbasereplay.html

Capture from 10g

- Create a directory to hold the rec files

```
create directory RAT as ‘/oracle/rat’
```

- Add a Filter

```
BEGIN
  dbms_workload_capture.add_filter(
    fname      => 'abcd_filter',
    fattribute => 'USER',
    fvalue     => 'ABCD');
END;
```

- Allows you to capture only those for the user called ABCD.

- Start the Capture Process

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.START_CAPTURE (
      name               => ‘capture1',
      dir                => 'RAT',
      duration           => 3600,
      default_action     => 'EXCLUDE',
      auto_unrestrict    => TRUE);
END;
```

- It will generate a lot of files in the format wcr_*.rec in the /oracle/rat directory.

- Get the capture ID

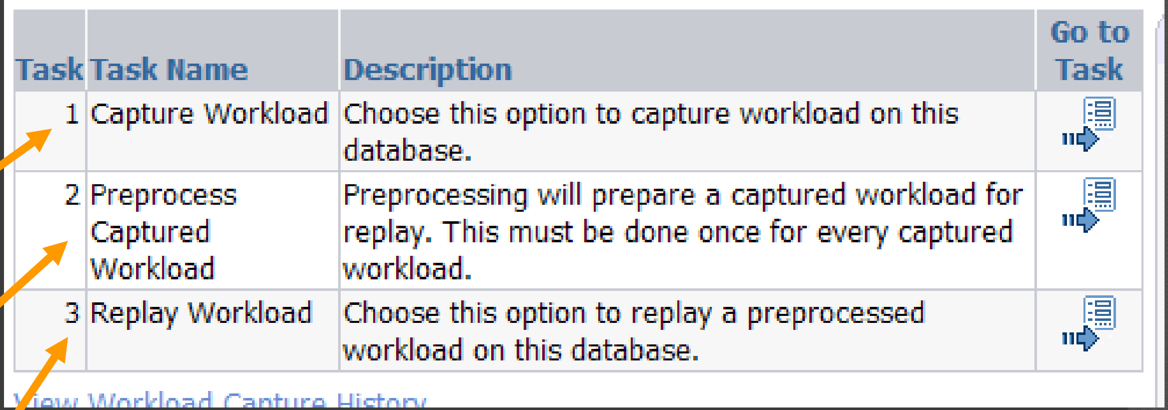
```
select ID from dba_workload_captures
where status = ‘COMPLETED’
```

- Export the AWR

```
begin
  dbms_workload_capture.export_awr
    (capture_id => <captureid>);
end;
/
```

- AWR will also be exported as a dumpfile in the /oracle/rat directory.
- Copy all the files in that directory to the target system

Replay Steps



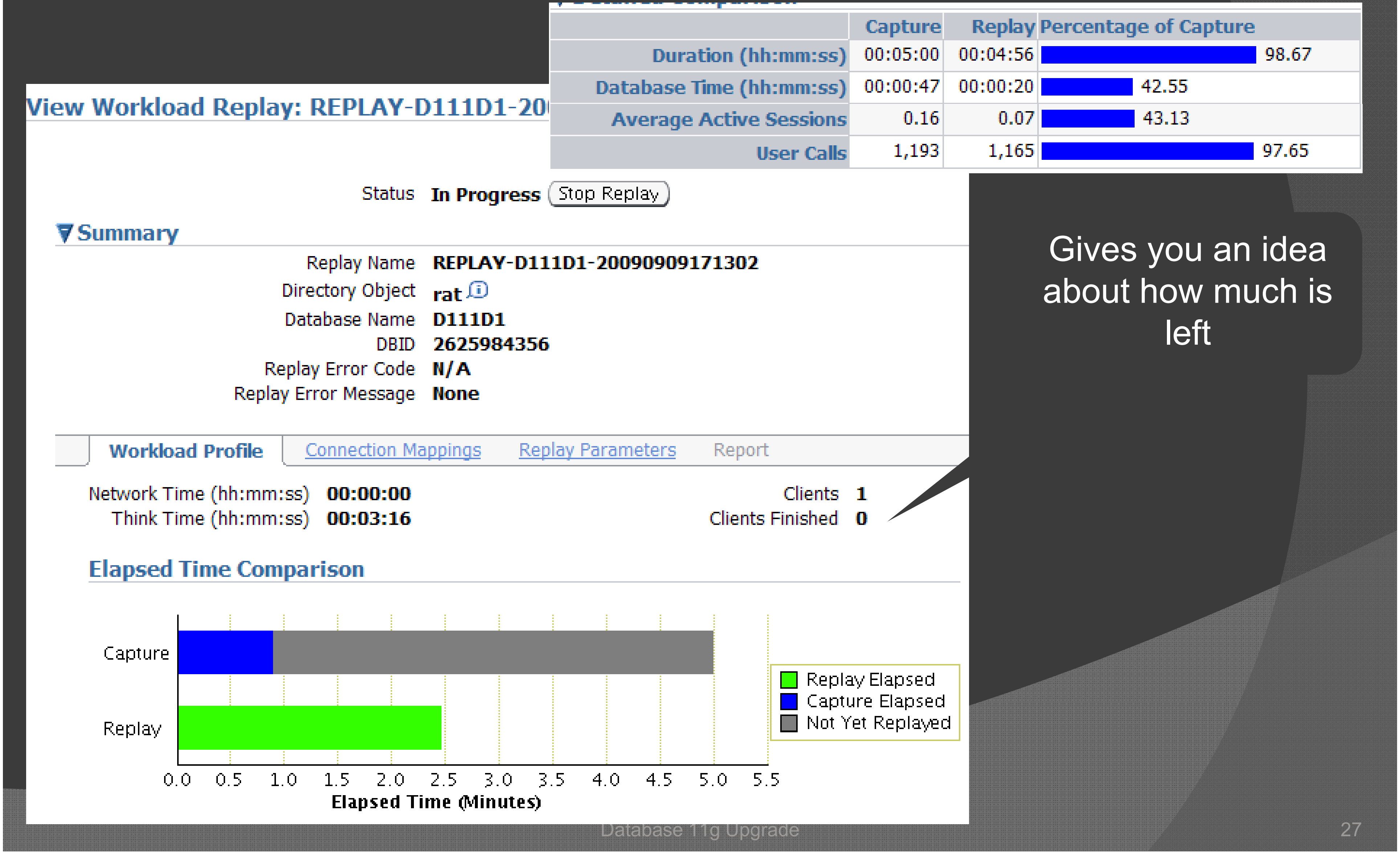
Task	Task Name	Description	Go to Task
1	Capture Workload	Choose this option to capture workload on this database.	➡
2	Preprocess Captured Workload	Preprocessing will prepare a captured workload for replay. This must be done once for every captured workload.	➡
3	Replay Workload	Choose this option to replay a preprocessed workload on this database.	➡

View Workload Capture History

1. Create directory on the target
2. Pre-process the captured workload
3. Replay the workload
4. From the command line

```
$ wrc system/manager replaydir=/u01/oracle/rat
```

During Replay



	Capture	Replay	Percentage of Capture
Duration (hh:mm:ss)	00:05:00	00:04:56	98.67
Database Time (hh:mm:ss)	00:00:47	00:00:20	42.55
Average Active Sessions	0.16	0.07	43.13
User Calls	1,193	1,165	97.65

View Workload Replay: REPLAY-D111D1-20

Status **In Progress** Stop Replay

Summary

Replay Name	**REPLAY-D111D1-20090909171302**
Directory Object	**rat**
Database Name	**D111D1**
DBID	**2625984356**
Replay Error Code	**N/A**
Replay Error Message	**None**

Workload Profile | Connection Mappings | Replay Parameters | Report

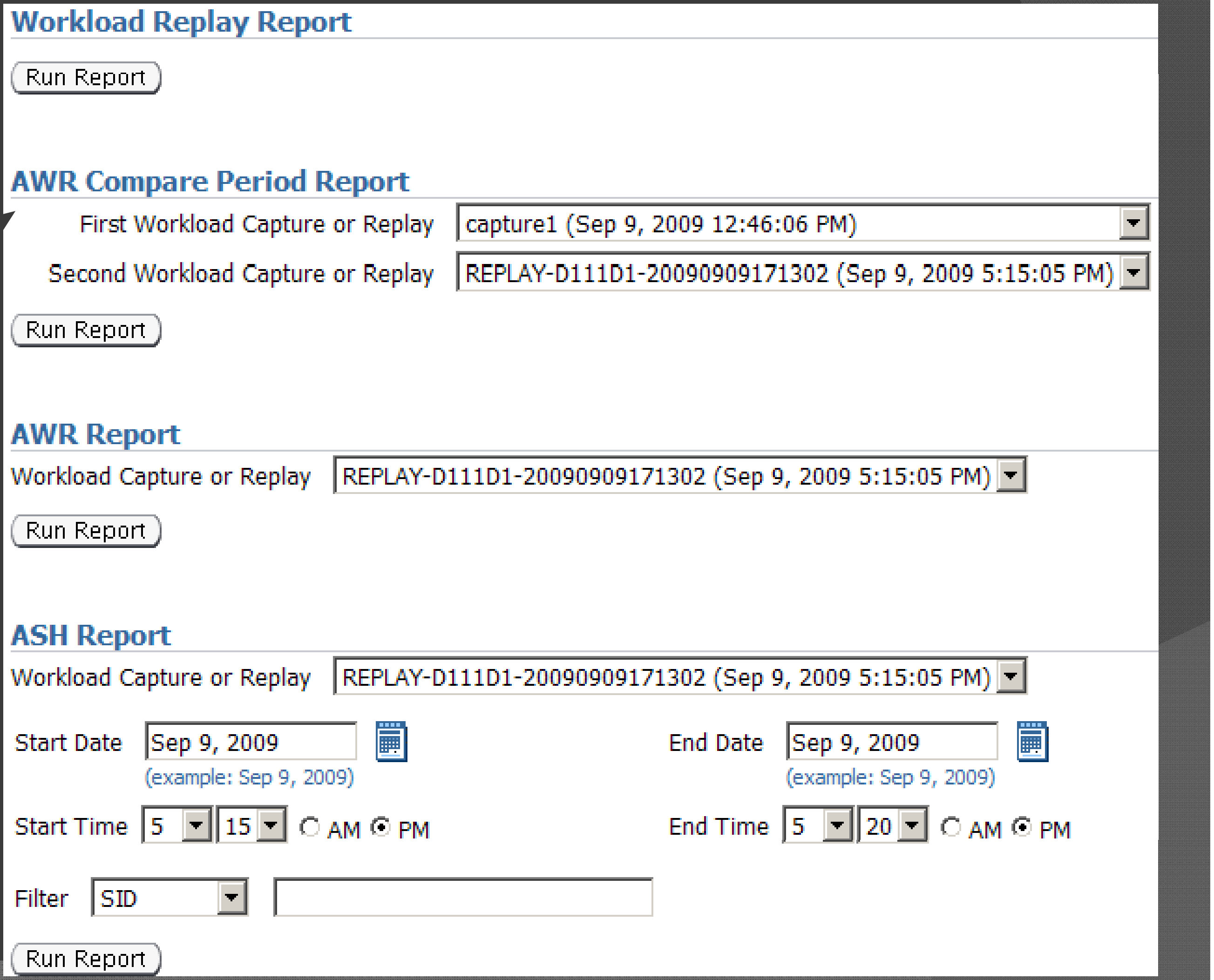
Network Time (hh:mm:ss)	**00:00:00**
Think Time (hh:mm:ss)	**00:03:16**
Clients	**1**
Clients Finished	**0**

Elapsed Time Comparison

Gives you an idea about how much is left

Get the Reports

This "compare" report, aka "Diff-diff Report" is the most important. It shows the system stats on the target and the source when the *same* activities were occurred there.



Workload Replay Report

AWR Compare Period Report

First Workload Capture or Replay: capture1 (Sep 9, 2009 12:46:06 PM)

Second Workload Capture or Replay: REPLAY-D111D1-20090909171302 (Sep 9, 2009 5:15:05 PM)

AWR Report

Workload Capture or Replay: REPLAY-D111D1-20090909171302 (Sep 9, 2009 5:15:05 PM)

ASH Report

Workload Capture or Replay: REPLAY-D111D1-20090909171302 (Sep 9, 2009 5:15:05 PM)

Start Date: Sep 9, 2009 (example: Sep 9, 2009)
End Date: Sep 9, 2009 (example: Sep 9, 2009)

Start Time: 5 15 ○ AM ● PM
End Time: 5 20 ○ AM ● PM

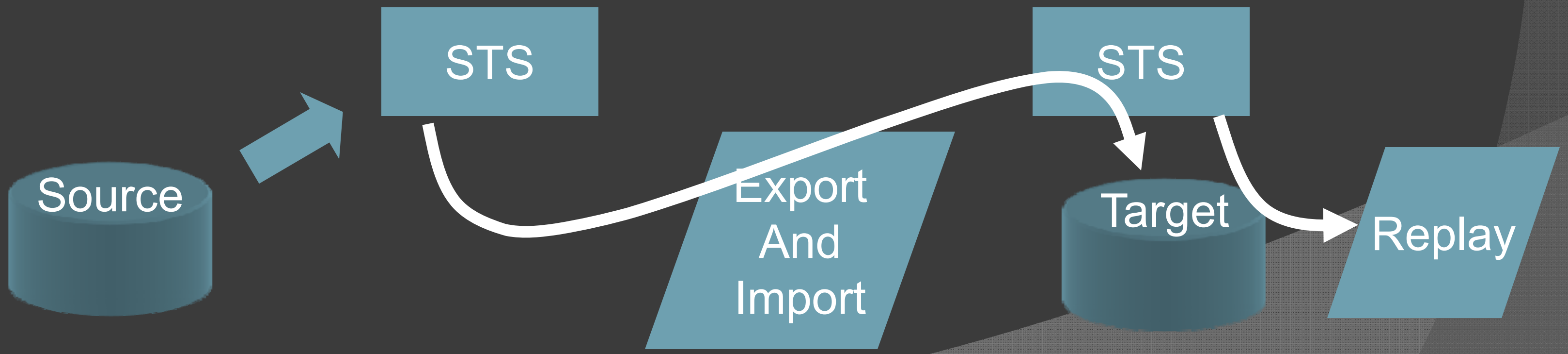
Filter: SID

SQL Performance Analyzer

- Some SQLs showed regression, i.e. they underperformed compared to 10g
- You need to know *why*
  - optimizer environment, bind variables, etc?
- SPA allows you to *run* captured SQLs in differing environments
  - In the same database but
    - Different optimizer parameters
    - Different ways of collecting stats,
    - With pending stats in 11g, can validate on PROD during maintenance windows/non-peak
    - Different indexes, or MVs

Source of SQLs

- Shared Pool
- Captured from Production during a workload
- Stored in a SQL Tuning Set (STS)
- Continuous Capture functionality to capture all SQLs



Capture from 10g

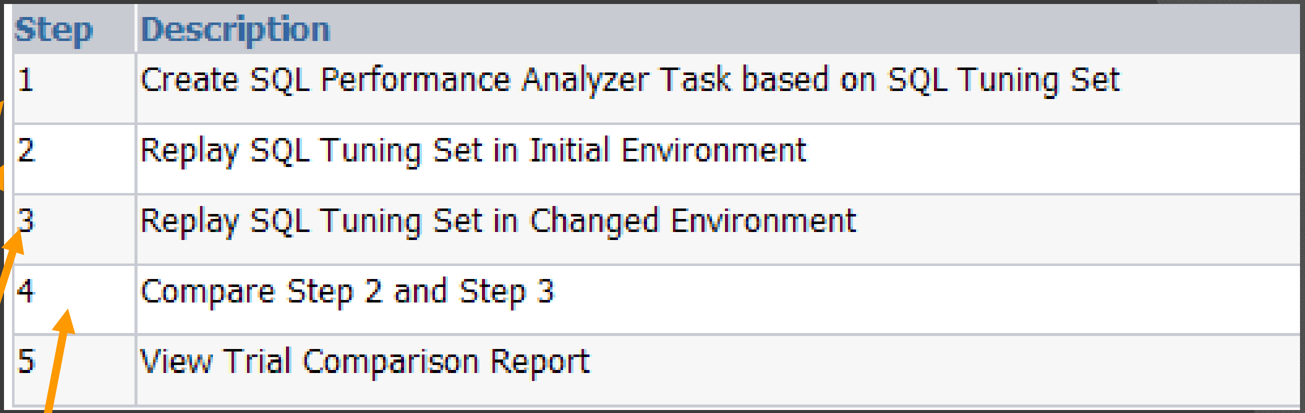
- The following captures the SQL Statements into a SQL Tuning Set (STS) in 10g.

```
BEGIN dbms_sqltune.capture_cursor_cache_sqlset(
     sqlset_name    =>'10GSTS',
     time_limit     => '3600',
     repeat_interval=>'300',
     sqlset_owner   =>'SYS');
END;
```

This incrementally captures the SQL statements every 5 mins for 10 hours.

- You can export this STS and import into 11g.

SPA Tasks

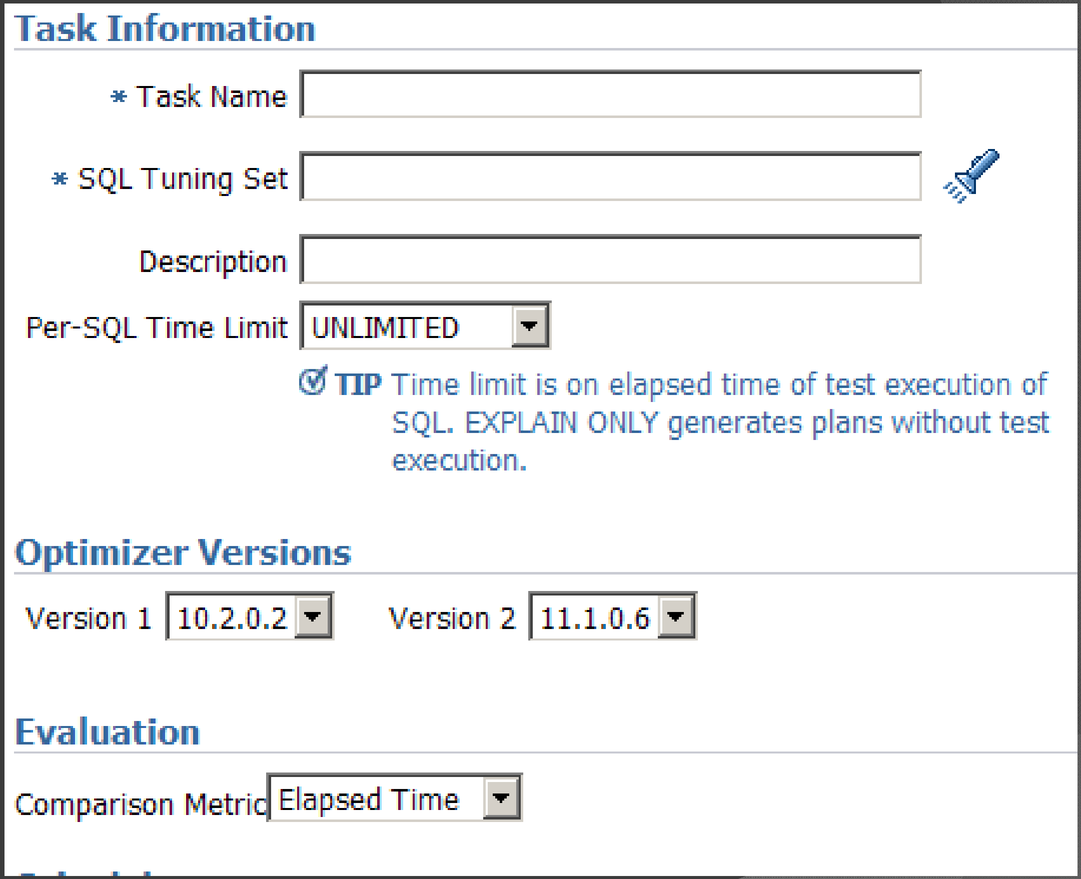


Step	Description
1	Create SQL Performance Analyzer Task based on SQL Tuning Set
2	Replay SQL Tuning Set in Initial Environment
3	Replay SQL Tuning Set in Changed Environment
4	Compare Step 2 and Step 3
5	View Trial Comparison Report

- Create an SPA Task on the STS imported
- Replay with Optimizer = 10.2.0.4
- Replay with Optimizer = 11.1.0.7
- Compare and make adjustments
- Repeat 2 through 4 as needed
- `http://www.oracle.com/technology/oramag/oracle/08-mar/o28sqlperf.html`

SPA Optimizer Change

Create an SPA Task on the STS imported



Task Information

* Task Name

* SQL Tuning Set

Description

Per-SQL Time Limit: UNLIMITED

TIP Time limit is on elapsed time of test execution of SQL. EXPLAIN ONLY generates plans without test execution.

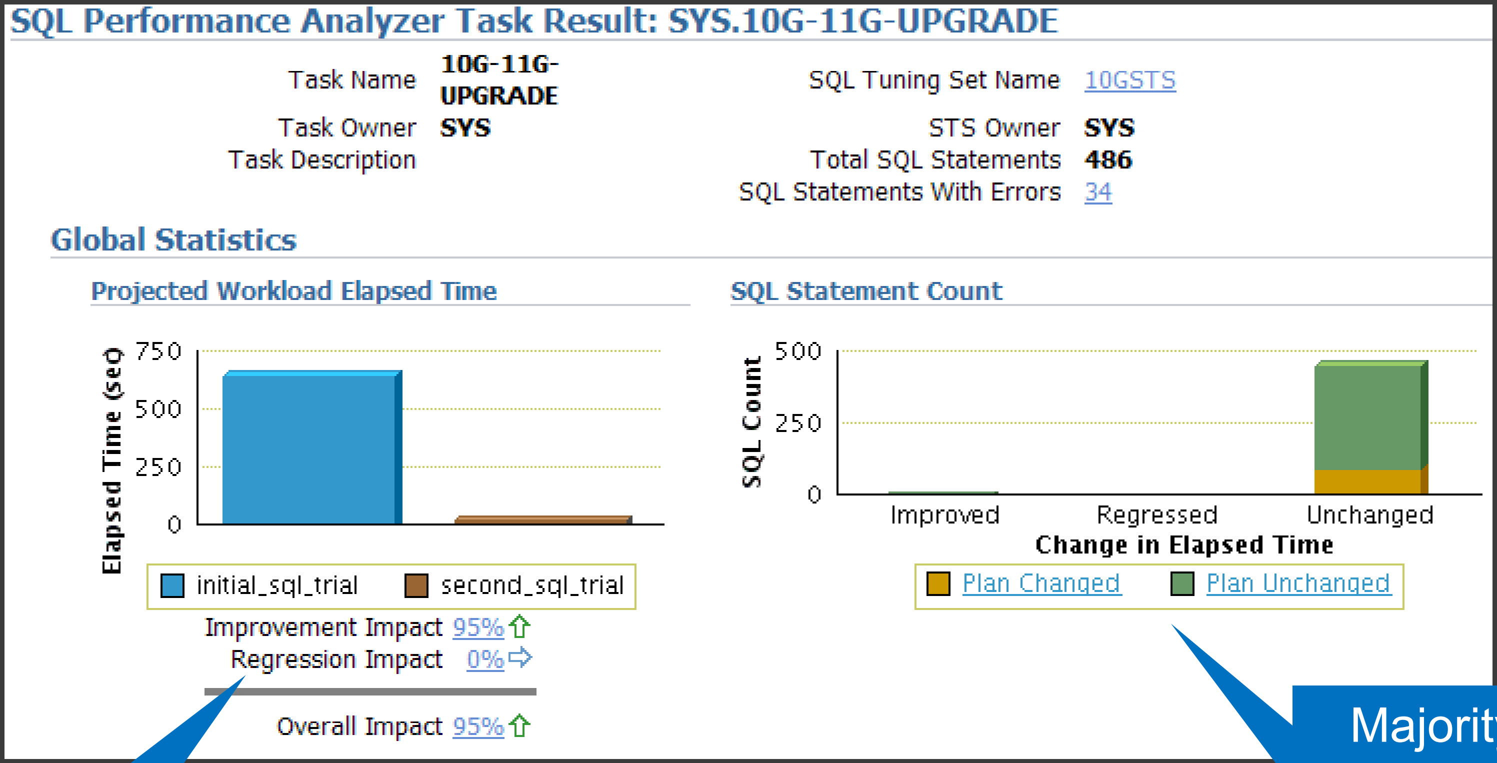
Optimizer Versions

Version 1: 10.2.0.2   Version 2: 11.1.0.6

Evaluation

Comparison Metric: Elapsed Time

Compare



SQL Performance Analyzer Task Result: SYS.10G-11G-UPGRADE

Task Name **10G-11G-UPGRADE**

Task Owner **SYS**

Task Description

SQL Tuning Set Name 10GSTS

STS Owner **SYS**

Total SQL Statements **486**

SQL Statements With Errors 34

Global Statistics

Projected Workload Elapsed Time

Improvement Impact 95% ⇧

Regression Impact 0% ⇨

Overall Impact 95% ⇧

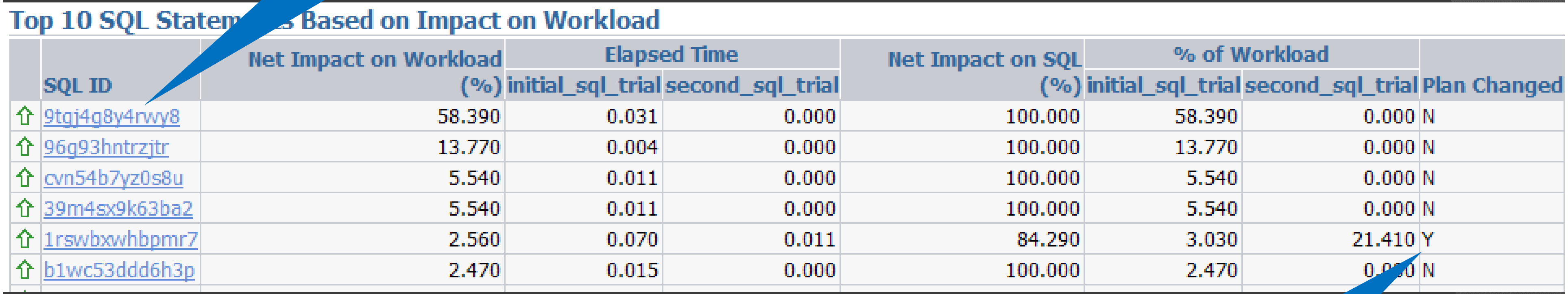
SQL Statement Count

Elapsed time significantly reduced

Majority of SQLs didn't see their plan changed!

Compare …

Shows the SQL_IDs, we can find from v$sql



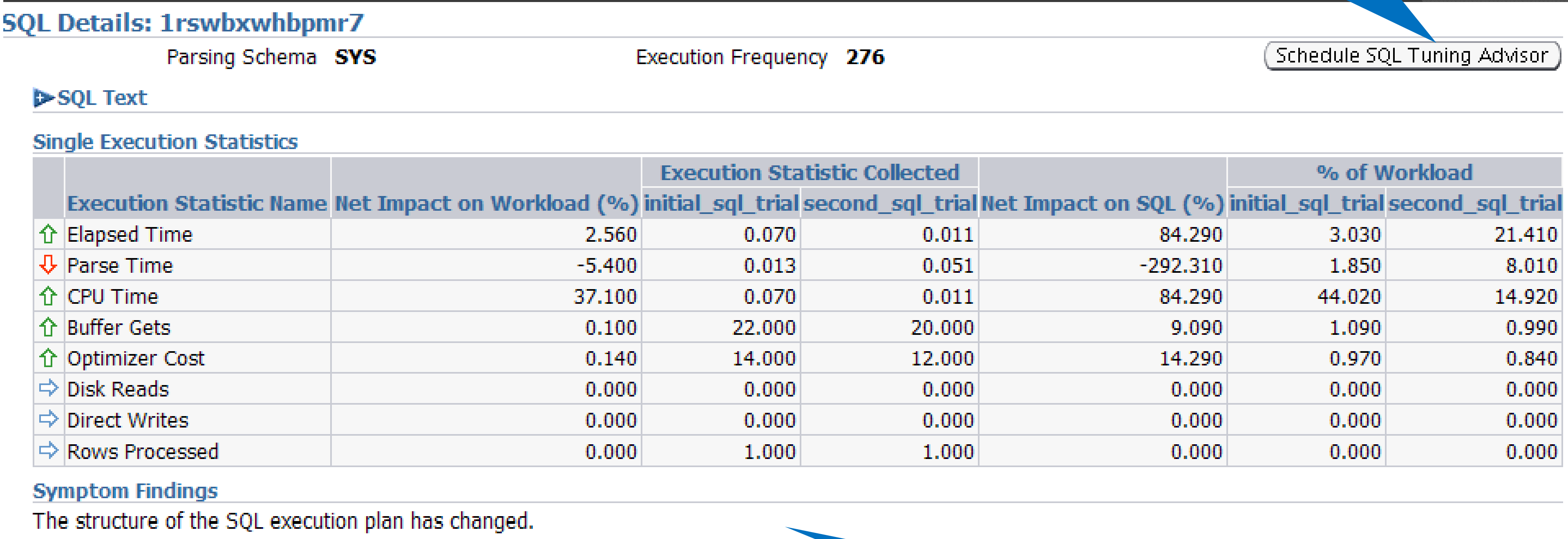
Top 10 SQL Statements Based on Impact on Workload

	SQL ID	Net Impact on Workload (%)	Elapsed Time initial_sql_trial	 second_sql_trial	Net Impact on SQL (%)	% of Workload initial_sql_trial	 second_sql_trial	Plan Changed
⇧	9tgj4g8y4rwy8	58.390	0.031	0.000	100.000	58.390	0.000	N
⇧	96g93hntrzjtr	13.770	0.004	0.000	100.000	13.770	0.000	N
⇧	cvn54b7yz0s8u	5.540	0.011	0.000	100.000	5.540	0.000	N
⇧	39m4sx9k63ba2	5.540	0.011	0.000	100.000	5.540	0.000	N
⇧	1rswbxwhbpmr7	2.560	0.070	0.011	84.290	3.030	21.410	Y
⇧	b1wc53ddd6h3p	2.470	0.015	0.000	100.000	2.470	0.000	N

Plan changed for this SQL, Using SQL_ID, check from v$sql

Clicking on the SQL_ID you can see the various stats on the SQL

You can call upon SQL Tuning Advisor to suggest possible tuning options on this SQL



SQL Details: 1rswbxwhbpmr7

Parsing Schema **SYS** Execution Frequency **276**

▶SQL Text

Single Execution Statistics

			Execution Statistic Collected			% of Workload	
	Execution Statistic Name	Net Impact on Workload (%)	initial_sql_trial	second_sql_trial	Net Impact on SQL (%)	initial_sql_trial	second_sql_trial
⇧	Elapsed Time	2.560	0.070	0.011	84.290	3.030	21.410
⇩	Parse Time	-5.400	0.013	0.051	-292.310	1.850	8.010
⇧	CPU Time	37.100	0.070	0.011	84.290	44.020	14.920
⇧	Buffer Gets	0.100	22.000	20.000	9.090	1.090	0.990
⇧	Optimizer Cost	0.140	14.000	12.000	14.290	0.970	0.840
⇨	Disk Reads	0.000	0.000	0.000	0.000	0.000	0.000
⇨	Direct Writes	0.000	0.000	0.000	0.000	0.000	0.000
⇨	Rows Processed	0.000	1.000	1.000	0.000	0.000	0.000

Symptom Findings

The structure of the SQL execution plan has changed.

The report continues with the plans before and after the upgrade, so you can compare them

SQL Plan Management

- What happens when the plan is actually worse?
- Perhaps the plan is better when a different optimizer environment parameter is used?
- In that case, we used SQL Plan Management to let the optimizer pick the right plan from the pool of plans

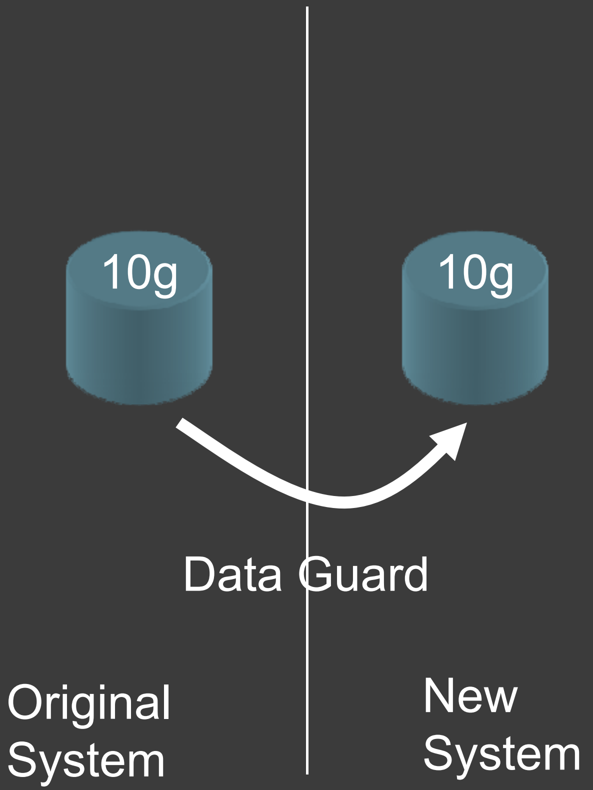
SPM

- Analogous to Stored Outlines
- But unlike outlines, baselines:
  - Calculate the plan anyway; but don't use it.
  - The DBA must check and mark a plan good by "accepting" it – a process called "evolving"
  - Have multiple plans in the baseline and choose the best
- So it is the best of both worlds

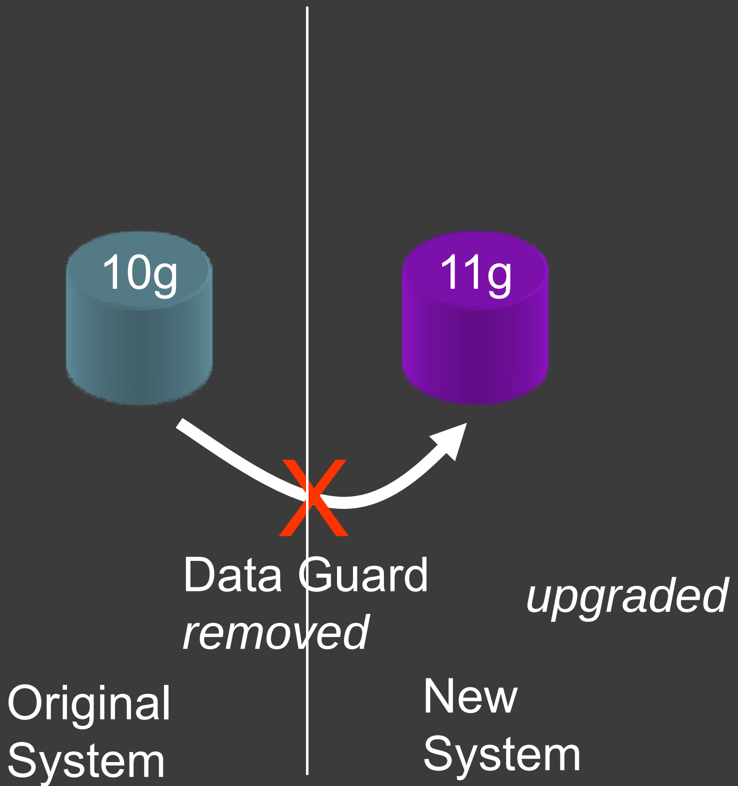
Strategy with SPM

- If a plan is “fixed”, that is used, regardless of the presence of other plans
- Capture all the plans from 10g to an SQL Tuning Set
- Load them to 11g after upgrade
- Mark all of them as fixed
  - So, the plans will be the same as 10g
- Turn on capture baselines; the new plans will be stored in the baselines
- Evolve them to see if any plan is better
- OTN Article explains it all: http://www.oracle.com/technology/oramag/oracle/09-mar/o29spm.html

Test System Creation



Test System Creation



1. Start the DB Workload Capture Process
2. Simultaneously break Data Guard
3. Convert the Standby to snapshot standby
4. Upgrade the standby to 11g

Converting 10gR2 Standby to RW

Primary	Standby
	1. alter database recover managed standby database cancel; 2. create restore point gold guarantee flashback database;
1. alter system archivelog current; 2. alter system log_archive_dest_state_2 = defer;	
	1. alter database activate standby database; 2. shutdown/startup mount 3. alter database set standby database to maximize performance; 4. alter system log_archive_dest_state_2 = defer; 5. alter database open;

Test System Creation

10g

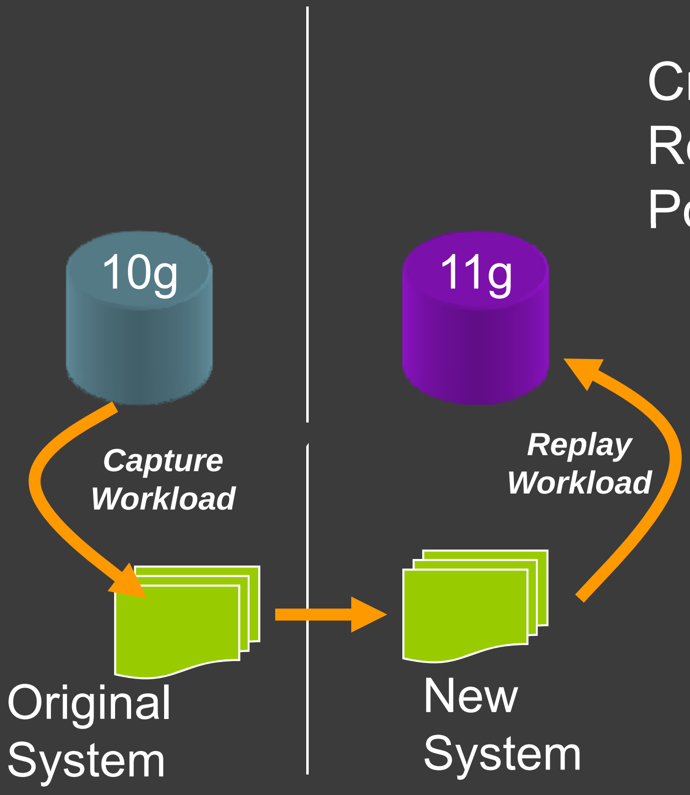
11g

Enable Database for Flashback

Original System

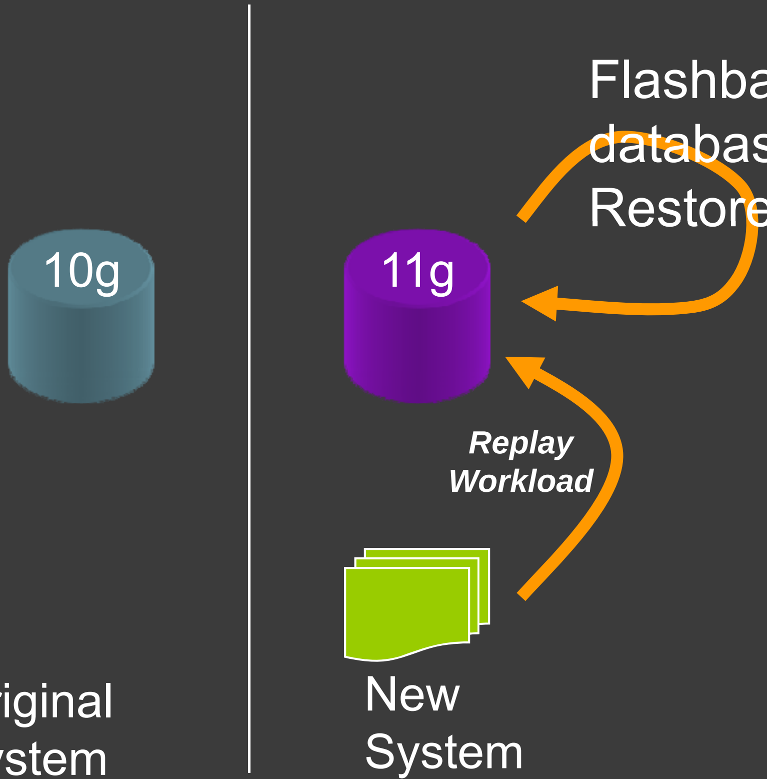
New System

Test System Creation



Create a Restore Point

Test System Creation

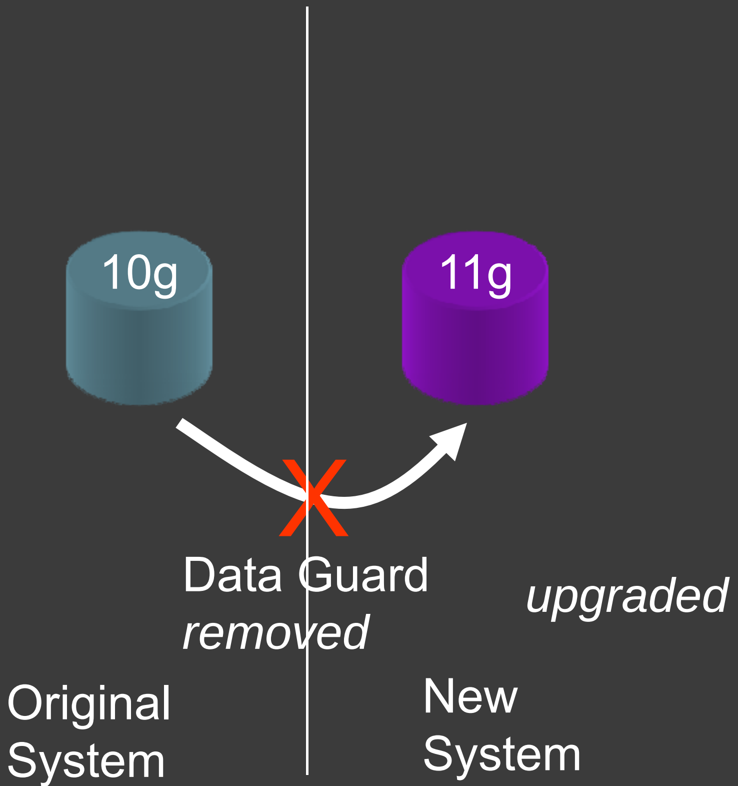


Flashback the database to the Restore Point

The workload has been captured only once; and replayed several times.

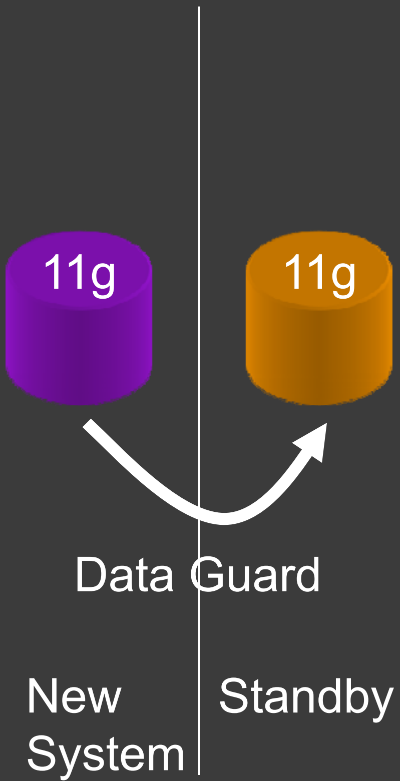
Repeat this as often as needed

Actual Upgrade



1. 10g → 10g Standby
2. Stop Data Guard
3. Upgrade the standby to 11g
4. This becomes the new production
5. The old prod is still available as of that point in time

Post Upgrade Tweaking



Data Guard

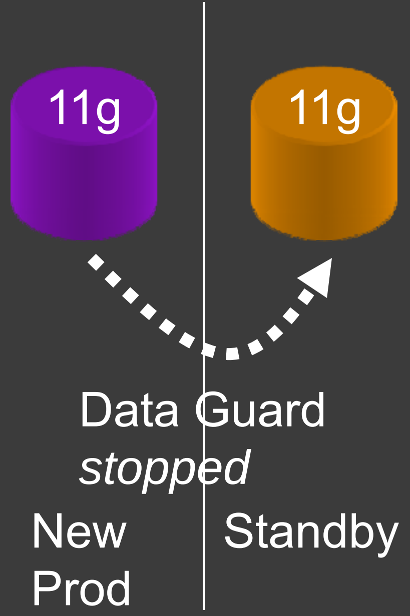
New System

Standby

Physical Standby, Converted from the old system

Post Upgrade Tweaking

1. What should the value of cursor_space_for_time should be?
2. What will be the effect of the I/O constraining Resource Manager?
3. What will be effect of the Patch Update?



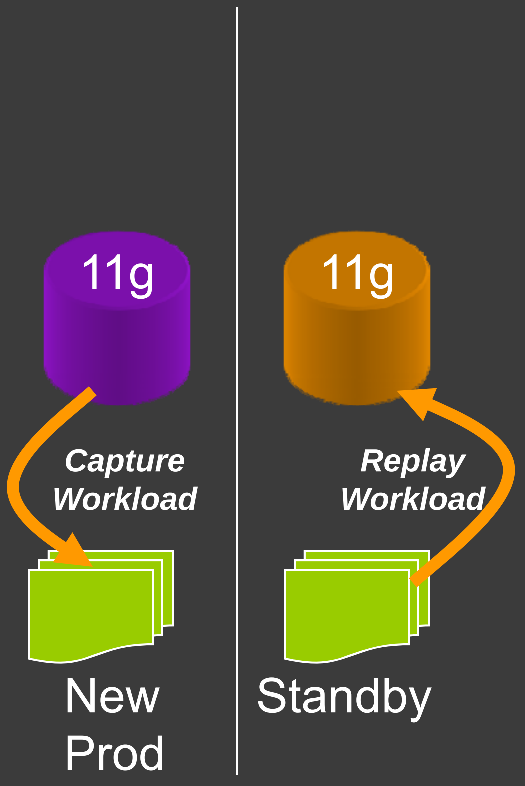
Data Guard *stopped*

New Prod

Standby

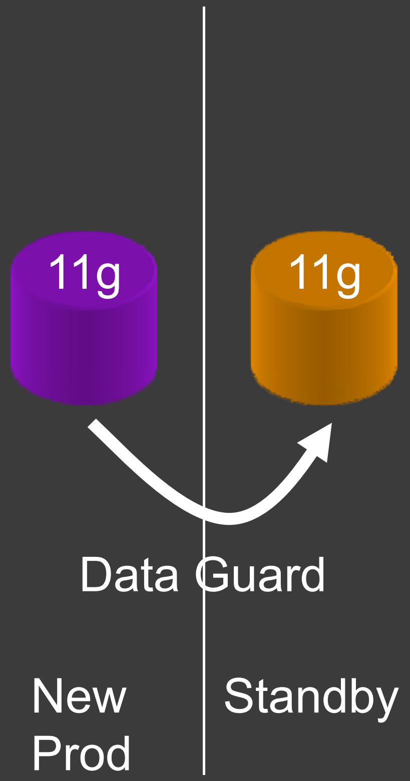
Convert to Snapshot Standby

Post Upgrade Tweaking



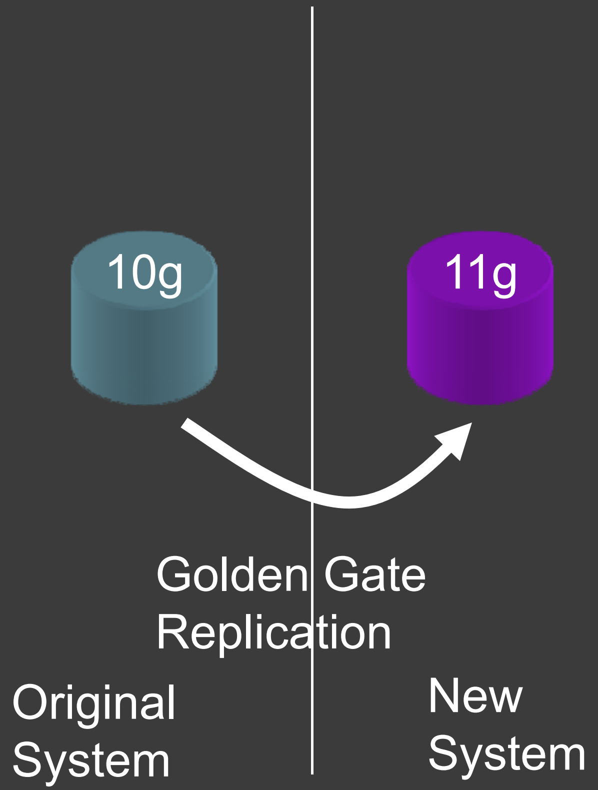
1. Take a Restore Point on the standby
2. Make changes on the standby
3. Capture workload from production
4. Replay against the standby
5. Flashback the standby to the Restore Point
6. Repeat steps 2-5

Convert to Active Data Guard



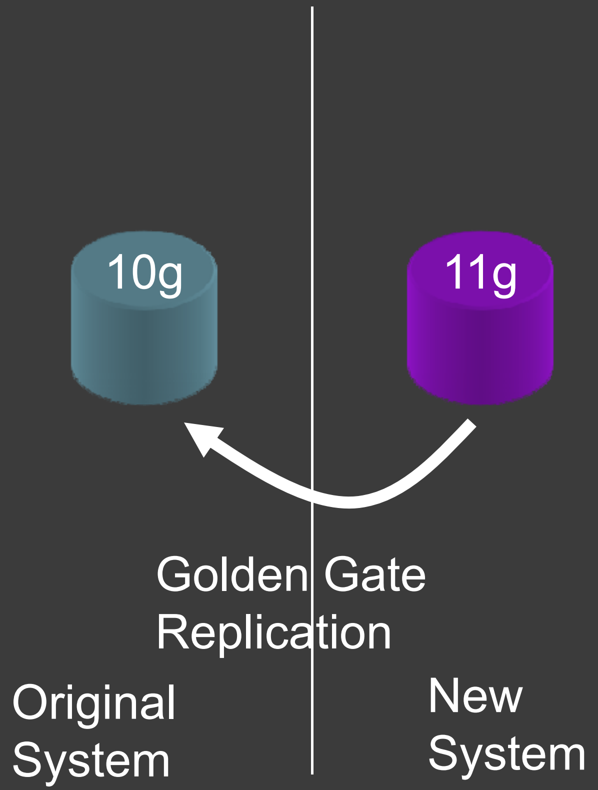
1. Convert the standby back to normal from snapshot
2. Stop Managed Recovery Process
3. Open the standby in Read Only mode
4. Restart the MRP
5. Pure Read Only queries can be directed at the Standby

Maintaining 2 Versions



1. 10g → 10g Standby
2. Break Data Guard
3. Upgrade the standby to 11g
4. This becomes the pre-production
5. Set up Golden Gate replication (or Streams) to apply SQLs to the 11g DB from 10g

Cutover



1. Stop the apply
2. Redirect clients to the new DB
3. Reverse the replication direction.

Tools Used

- Database Replay
- SQL Performance Analyzer
- SQL Tuning Advisor
- Snapshot Standby
- Active Data Guard
- Golden Gate

Conclusion

- Upgrade is just going to happen, you can't avoid it
- This is the best you can do to mitigate the risks, by replaying the activities as faithfully as you can
- Oracle's Real Application Testing Suite allows you exactly that – faithfully replaying the activities
- Using Standby database you can minimize the risk of failure during upgrade.
- Snapshot Standby allows you to tweak the parameters and sets the stage for future upgrades

감사합니다 Obrigado

Спасибо धन्यवाद

Danke תודה רבה

Thank you!

多謝 Grazie

Thank You

Merci

شكرا

ขอบคุณ நன்றி

Gracias

ありがとうございました