ORACLE

ORACLE DOJO NR. 5

ULRIKE SCHWINN

Real Application Testing
Testen mit Database Replay und SQL Performance Analyzer

Oracle Dojo ist eine Serie von Heften, die Oracle Deutschland B.V. zu unterschiedlichsten Themen aus der Oracle-Welt herausgibt.

Der Begriff Dojo [ˈdo:dʒo] kommt aus dem japanischen Kampfsport und bedeutet Übungshalle oder Trainingsraum. Als „Trainingseinheiten", die unseren Anwendern helfen, ihre Arbeit mit Oracle zu perfektionieren, sollen auch die Oracle Dojos verstanden werden. Ziel ist es, Oracle-Anwendern mit jedem Heft einen schnellen und fundierten Überblick zu einem abgeschlossenen Themengebiet zu bieten.

Im *Oracle Dojo Nr. 5* beschäftigt sich Ulrike Schwinn, Senior Leitende Systemberaterin bei Oracle Deutschland B.V., mit der **Real Application Testing (RAT)** Datenbankoption und zeigt, wie diese Technologie eingesetzt werden kann, um eine effiziente und realitätsnahe Testdurchführung gewährleisten zu können.

ORACLE

Inhalt

ORACLE DOJO NR. 5

ULRIKE SCHWINN

Real Application Testing

Testen mit Database Replay und SQL Performance Analyzer

VORWORT DES HERAUSGEBERS

„Drum prüfe wer sich ewig bindet … Der Wahn ist kurz, die Reu ist lang“, schrieb schon Schiller in seinem *Lied von der Glocke* (1799) und meinte damit vieles, aber sicher nicht die heutige IT – und doch treffen diese Worte auch im IT-Umfeld den Nagel auf den Kopf. Testen und Prüfen von Anwendungen und Systemen gehört zu den Königsdisziplinen der IT. Einführung von Systemen im „Wahn“, ohne genügende und richtige Prüfung, führt immer zu einer langen Leidensphase für Anwender und für die IT-Verantwortlichen.

Dabei ist die beste Prüfung meist die, die am nächsten an die realen Anwendungsfälle und Lastprofile heranreicht. Nicht umsonst baute kürzlich ein renommierter Automobilhersteller eine der schlechtesten Straßen der Republik auf mehrere Kilometer als Teststrecke für das firmeneigene Testcenter nach. Auch hier gilt: Nichts ist realer als die Realität.

Dies ist auch das Grundprinzip und die Grundidee der **Real-Applikation-Testing-Option (RAT)**, die seit Oracle 11g zur Verfügung steht. Während eines normalen Arbeitstages – oder auch während der Höchstlastzeiten, wenn dies gewünscht ist – wird für ein oder zwei Stunden die Last aufgezeichnet und kann später auf einem Zielsystem mit anderer DB-Version, anderem HW-System, modifiziertem Plattenlayout etc. ausgeführt, also

„nachgespielt“ werden; auch mehrfach, jeweils mit unterschiedlichen Parametern.

Neben dieser **DB-Replay-Funktion** beschäftigt sich dieses Dojo auch intensiv mit der zweiten Komponente der RAT-Option – bei der es um das Testen von SQL-Befehlen geht – dem **SQL Performance Analyzer**.

Ulrike Schwinn, die für dieses Dojo verantwortlich zeichnet, hat in den letzten Jahren sehr viel Erfahrung mit RAT gesammelt. Sowohl kritische Systeme als auch größte SAP-Systeme wurden mit Unterstützung der RAT-Technologie getestet, bevor neue Versionen und neue Plattformen produktiv geschaltet wurden. Ich freue mich sehr, dass sie ihr Wissen für dieses Dojo zur Verfügung stellt und wünsche Ihnen viel Spaß beim Lesen und Testen.

Ihr Günther Stürner
Vice President Sales Consulting

PS: Wir sind an Ihrer Meinung interessiert. Anregungen, Lob oder Kritik gerne an barbara.frank@oracle.com. Vielen Dank!

1 Einführung

Testen ist unerlässlich und sollte zu jedem Software-Entwicklungszyklus beziehungsweise zu jedem Qualitätssicherungsverfahren gehören. Testen ist allerdings auch aufwendig: der Aufbau der Testumgebung und das zur Verfügung stellen von adäquaten Testprozeduren kann etliche Personen-Tage oder sogar -Monate kosten. Erschwerend kommt hinzu, dass man mit den meisten Testprozeduren nur eine Simulation durchführen kann, die in der Regel nicht dem realen Applikationscharakter entspricht.

Mit **Oracle Database 11g** steht eine neue Technik im Rahmen der **Real-Application-Testing-Option** (RAT-Option) zur Verfügung, die eine einfache Lösung bereitstellt, um Testabläufe zu vereinfachen und Systeme mit realen Applikationslasten zu testen. Bei Real Application Testing handelt es sich – ganz einfach formuliert – um ein Werkzeug für die Datenbank, das einen Workload aufzeichnen und in einer Testumgebung wieder abspielen kann. Der Ausdruck Werkzeug ist dabei etwas irreführend, da keine zusätzliche Installation einer separaten Werkzeug-Software für Real Application Testing notwendig ist. Die Nutzung erfolgt wie üblich über die Standardwerkzeuge Oracle Enterprise Manager oder PL/SQL beziehungsweise SQL-Aufrufe.

Welche Idee steckt hinter der Real-Application-Testing-Option? Ohne zusätzlichen Aufwand – wie beispielsweise dem Einsatz von speziellen Skriptsprachen – ist es möglich, eine Last oder einen SQL Workload aufzuzeichnen und in einer existierenden Testumgebung abzuspielen. Somit bietet sich Real Application Testing besonders bei Datenbank-Upgrades, Patch-Anwendungen, Konfigurationsänderungen wie zum Beispiel dem Umstieg auf RAC oder ASM, Änderungen an Datenbank-Sizing-Parametern, Änderungen an der Hardware-Plattform, am Betriebssystem oder am Storage an, um nur einige Beispiele zu nennen. Folgende Fragestellungen sind dabei typisch: Funktioniert meine Applikation nach dem Upgrade auf ein neues Datenbank-Release, nach dem Wechsel auf eine neuen Hardware oder nach Einspielung von Patches weiterhin wie gewohnt? Bleibt die SQL Statement Performance erhalten? Wie wirkt sich die Last gemäß I/O, CPU und Memory-Metrik auf das neue System insgesamt oder gar pro Statement aus?

Um die oben genannten Fragen zu beantworten, werden in diesem Dojo Technologien erläutert, über Voraussetzungen aufgeklärt, ein Verständnis für Funktionsweisen gegeben, Skripte für die „Linemode"-Nutzung zur Verfügung gestellt und von Erfahrungen aus vorherigen Projekten berichtet. In den folgenden Ausführungen wird bewusst auf die Nutzung der grafischen Oberfläche durch den Oracle Enterprise

Manager verzichtet, um der Lesbarkeit im Dojo-Format Rechnung zu tragen.

Real Application Testing beinhaltet zwei unabhängige sich ergänzende Lösungen, die je nach Charakteristik der Applikation oder Art der Analyse zum Einsatz kommen können. Es handelt sich dabei um das **Database Replay** (DB Replay) und um den **SQL Performance Analyzer** (SPA). Wie bei vielen Produkten oder Features ist der Name bezeichnend für den Charakter und die Eigenschaften der beiden verschiedenen Funktionalitäten.

1.1 DATABASE REPLAY UND SQL PERFORMANCE ANALYZER – EINE KURZBESCHREIBUNG

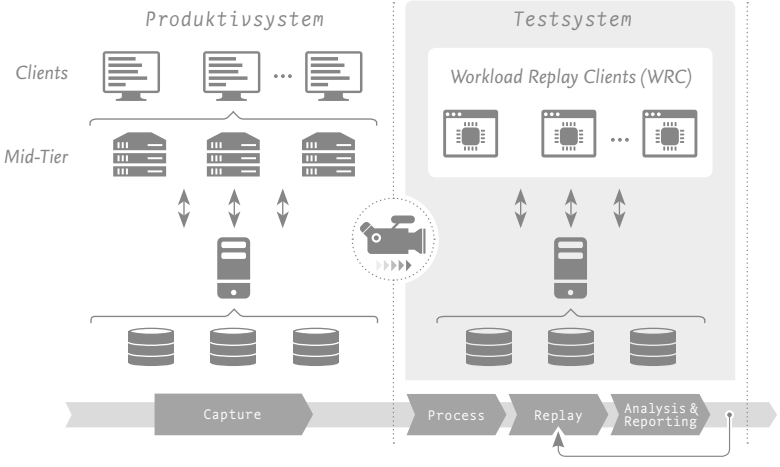
Database Replay zeichnet die gesamte Datenbanklast (Workload) unter Berücksichtigung aller konkurrierenden Zugriffe auf und führt eine Analyse bezogen auf die Gesamtlast der Datenbank durch (siehe Abb. 1). Das bedeutet, dass alle Datenbank-Calls (Ausnahmen sind im Application-Testing-Handbuch vermerkt) somit auch PL/SQL Calls im Workload enthalten sind. Der Ablauf sieht drei Verarbeitungsschritte vor:

- das Aufzeichnen auf dem Produktionssystem (Capture)
- eine spezielle Verarbeitung auf dem Testsystem (Processing oder Preprocessing)

- das eigentliche Abspielen auf dem Testsystem (Replay)

Dabei gibt es folgende wichtige Einschränkung zu beachten: Das Aufzeichnen (Capture) kann auf 9i-, 10g- und 11g-Datenbanken, das Abspielen (Replay) hingegen nur auf 11g-Datenbanken stattfinden. Des Weiteren ist es notwendig, ein passendes Datenbank-Testsystem – am besten SCN-genau – zur Verfügung zu stellen, damit das Replay nicht allzu viele Divergenzen produziert. Zudem sollte ein Mechanismus wie Flashback mit entsprechenden Restore Points implementiert werden, um mehrere Tests nacheinander auf dem System durchführen zu können.

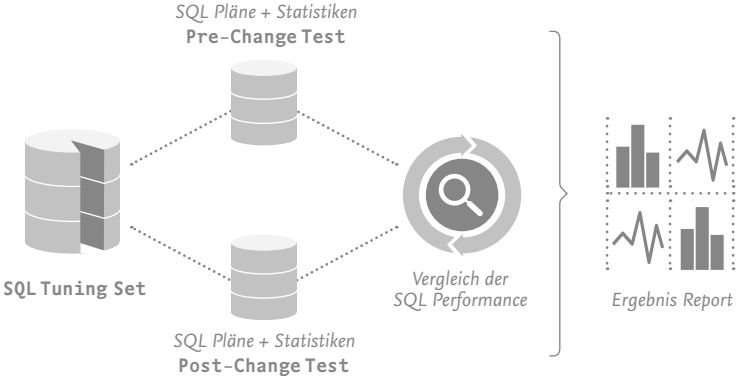
Abb.1: Database-Replay-Ablauf



Der Fokus von **SQL Performance Analyzer** (SPA) liegt auf der detaillierten Statement-Analyse eines definierten SQL Workloads (siehe Abb. 2). Ein SQL Workload besteht dabei aus SELECT- Statements beziehungsweise DML-Statements (nur im Linemode), die über ein SQL Tuning Set (STS) zur Verfügung gestellt werden. Der SQL Workload wird dabei zweimal abgespielt, vor und nach der Veränderung. Das Ergebnis ist eine detaillierte Vergleichsanalyse der einzelnen Statements, vor und nach der Veränderung, gemessen an verschiedenen Metriken, beispielsweise *elapsed time.* Möchte man nun einen Tuning-Prozess anschließen, ist dies einfach möglich durch die Integration des SQL Tuning Advisors oder durch die Nutzung von SQL Plan Baselines.

Vorteile von SPA sind die einfache Handhabung, Flexibilität und Verfügbarkeit. Da keine DML-Statements die Testumgebung verändern – DML-Statements werden automatisch zurückgerollt – ist kein Zurücksetzen nach dem Test erforderlich. Zudem kann die Analyse schon für 9i-Datenbanken erfolgen.

Abb.2: SQL-Performance-Analyzer-Ablauf



1.2 WICHTIGE VORAUSSETZUNGEN

Bevor Sie starten, sollten Sie unbedingt die **My Oracle Support Note 560977.1** lesen, um unter Umständen notwendige Patches einzuspielen. Folgendes Beispiel demonstriert die Patch-Überprüfung in einer 10g-Umgebung.

```
$. ./$ORACLE_HOME/OPatch/opatch lsinventory
Invoking OPatch 10.2.0.4.2
Oracle Interim Patch Installer version 10.2.0.4.2
...
There are 3 products installed in this Oracle Home.
Interim patches (1) :
```



```
Patch 10239989 : applied on Sat Aug 11 17:38:03 CEST 2012
Created on 7 Sep 2011, 01:47:54 hrs PST8PDT
Bugs fixed:
   10239989
--------------------------------------------------------
OPatch succeeded.
```

Überprüfen Sie, ob Real Application Testing auch wirklich **installiert** ist. Dies geschieht über die View V$OPTION. Der Wert der VALUE Column sollte dabei auf TRUE stehen. Folgende Abfrage überprüft die Installation:

```
select parameter, value from v$option
where parameter like 'Real Application Testing';
PARAMETER                      VALUE
-------------------------      ------------------
Real Application Testing       TRUE
```

Falls die Installation kein Real Application Testing enthält, kann mit dem Linemode-Werkzeug CHOPT schnell Abhilfe geschaffen werden.

```
$ chopt enable rat
```

Informieren Sie sich über die **Lizenzierung** Ihres Produkts. Real Application Testing ist eine kostenpflichtige Option der Datenbank. Möchte man das vollständige Report Framework beziehungsweise Tuning Framework nutzen, ist zusätzlich die Lizenzierung des Oracle Diagnostics Pack und des

Oracle Tuning Pack notwendig. Mehr Informationen dazu finden Sie auch im Licensing Guide (siehe Kapitel 6).

2 Oracle Database Replay

Wie funktioniert nun DB Replay? Dazu ist folgendes Drei-Schritt-Verfahren nötig:

- Das **Capture** stellt die Aufzeichnung auf dem Produktionssystem dar.
- Das Workload **Processing** (vgl. Abb. 1) ist die einmalige Aufbereitung der Dateien, um das nachfolgende Replay vorzubereiten.
- Das **Replay** ist dann das Abspielen auf dem Testsystem.

Auf dem Produktionssystem wird das sogenannte **Capture** in ein leeres logisches Verzeichnis (Directory) der Datenbank durchgeführt. Nutzt man das Capture in Version 10g, muss vorab der Parameter PRE_11g_ENABLE_CAPTURE eingeschaltet werden. Der Parameter ist dynamisch, sodass keine Downtime der Datenbank notwendig ist. Das Capture ist ein sehr einfach zu initiierender Prozess, der nur wenig Overhead auf dem Produktionssystem erzeugt. Das Capture sollte nicht zu lange dauern, ein Workload von circa ein bis zwei Stunden ist aber durchaus üblich. Dies gewährleistet

die Durchführung mehrerer Tests und erleichtert die Problemlösung nach dem Abspielen. Das soll nicht heißen, dass nicht auch Workloads von noch mehr Stunden machbar sind. Mehr hierzu finden Sie im Kapitel 4.

Hier schließt sich direkt die Frage nach dem Umfang der Capture-Dateien an. Eine einfache Antwort darauf gibt es nicht. Die Größe der Capture-Dateien steht zum Beispiel nicht in Zusammenhang mit dem Redo-Aufkommen, sondern sie errechnet sich in Abhängigkeit der Statistiken rund um die Statements. Dabei spielen unter anderem folgende Faktoren eine Rolle: Die Anzahl der verschiedenen Statements, die Verwendung von LOBs und Bulk Binds, die Anzahl der Datenbank-Connections und Parallelisierung, um nur einige Parameter zu nennen. Die Größenordnung kann am einfachsten über ein kurzes Test-Capture erfolgen. Möchte man den Automatic Workload Repository Report (AWR Report) zur Berechnung heranziehen, kann man sich an der Metrik *bytes received via SQL*Net from* orientieren.

Weniger Capture-Dateien kann man durch das Filtern – zum Beispiel nach User, Instance Id oder Services – beim Capture-Lauf erhalten. Dabei kann man einschließende oder ausschließende Filterkriterien definieren. So kann man sich beispielsweise über den Einsatz von Services nur auf einige wenige Applikationsserver konzentrieren oder aber bestimmte DB-User wie SYS oder SYSMAN ausschließen.

Die Capture-Dateien sind dabei binär und unabhängig vom Betriebssystem, sodass der Test auf unterschiedlichen Plattformen möglich ist. Abgeschlossen wird dieser Vorgang durch einen AWR-Export, um nach dem Replay einen AWR Difference Report produzieren zu können.

Hinweis: Hierzu muss das Oracle Diagnostics Pack lizenziert sein.

Fehler, die sich während des Captures ereignen werden aufgezeichnet, sie sind aber meist vernachlässigbar.

Das anschließende **Workload Processing** muss nur einmal durchgeführt werden. Dabei werden die Capture-Dateien analysiert und auf das Abspielen vorbereitet. Das Processing dauert je nach Menge und Komplexität des Captures unterschiedlich lang. Es kann auf der Testmaschine oder jedem beliebigen anderen System durchgeführt werden – einzige Voraussetzung: gleiches Datenbank-Release wie Testserver. Wenn grafisch gearbeitet wird, sollte man unbedingt parallel dazu den Workload Analyzer anstoßen. Er gibt darüber Auskunft, welche Auffälligkeiten und Charakteristiken der Capture Load aufweist und kann Hinweise auf die einzusetzende Replay-Option geben. Die Linemode-Variante des Workload Analyzers wird in den folgenden Skripten zur Verfügung gestellt und sollte nach dem Processing manuell gestartet werden.

Das **Replay** hingegen besteht aus mehreren Schritten. Zuerst werden ein Name und das logische Directory, in dem sich die Dateien befinden, bekannt gegeben. Dann wird ein Prepare durchgeführt, das über die Abspielart entscheidet. Abgespielt wird der Workload mit speziellen Clients – den sogenannten **Workload Replay Clients** (WRC). Die Anzahl der zu startenden Clients wird vorab über eine Kalibrierung festgelegt. Die WRCs können auf dem Testrechner selbst oder unabhängig davon auf beliebigen zusätzlichen Testrechnern gestartet werden. Dazu ist keine Oracle-Software-Client-Installation notwendig.

Bevor man das Replay startet, muss über dessen Art nachgedacht werden. Der Default ist ein COMMIT-synchrones-Abspielen (auch: preserve commit order) und eine Eins-zu-eins-Abbildung der *think* und *connection time*. Da die Abspiellogik einer eigenen Orchestrierung (auch: Regelwerk) folgt, kann in Abhängigkeit von der Workload-Charakteristik und des Testziels von diesem Default abgewichen werden. So kann zum Beispiel mit *synchronization* FALSE oder OBJECT_ID abgespielt werden sowie die *think* oder die *connection time* variiert werden. Im folgenden Kapitel wird noch genauer auf die Replay-Optionen eingegangen. Um eine einfache Integration in den SPA zu gewährleisten, ist es übrigens möglich, ein SQL Tuning Set während des Replays oder während des Captures anzustoßen.

Diese drei Schritte sind einfach in der grafischen Oberfläche zu nutzen. Intuitiv wird man zu den entsprechenden Schritten geführt. In Testumgebungen steht allerdings seltener eine grafische Oberfläche zur Verfügung. Im nächsten Abschnitt wird daher die Linemode-Variante vorgeführt.

2.1 CAPTURE, PROCESSING UND REPLAY IM LINEMODE

Damit ein einfacher Umgang mit dem Beispielcode möglich ist, stehen Ihnen die DB-Replay-Skripte für Capture, Processing und Replay-Nutzung mit den entsprechenden zusätzlichen Optionen zum **Download** zur Verfügung. Die Adresse für den Download finden Sie in Kapitel 6. Ein Readme, das schrittweise abgearbeitet werden kann, führt Sie dabei durch die Anwendung der einzelnen Skripte. So lässt sich beispielsweise das Capture in der einfachsten Variante mit einem einzigen Kommando durchführen. Hier ein kleiner Auszug aus dem Readme:

```
I) Voraussetzung
1) RAT Option überprüfen: OPTION.sql
2) Falls nicht gelinkt, dann $ chopt enable rat
3) Backup durchführen
II) Rechner 1: Durchlauf auf der CAPTURE Seite
-1) (falls 10g) PRE_11g_ENABLE_CAPTURE=TRUE Parameter setzen
0) (Optional) Directory leeren
```



```
1) Directory überprüfen          - Skript: DIR.sql
2) (Optional) Filter setzen      - Skript: CAPTUREFILTER.sql
3) Capture einschalten           - Skript: CAPTURESTART.sql
4) (Optional) Capture stoppen    - Skript: CAPTUREFINISH.sql
5) Capture überprüfen            - Skript: CAPTUREMONITOR.sql
6) AWR Export durchführen        - Skript: EXPORTAWR.sql
...
```

In der Regel handelt es sich um die Packages DBMS_WORKLOAD_CAPTURE und DBMS_WORKLOAD_REPLAY. Die wichtigsten Views sind DBA_WORKLOAD_CAPTURES und DBA_WORKLOAD_REPLAYS. DBA_WORKLOAD_CAPTURES gibt die Start-SCN und die Anzahl der aufgezeichneten Calls aus und spiegelt den Stand des Captures wider. Die View DBA_WORKLOAD_REPLAYS gibt Informationen über den Stand der Replays. Im Folgenden wollen wir den Ablauf schrittweise demonstrieren:

Zuerst wird das Capture auf der Produktionsseite durchgeführt. Ist das Produktionssystem ein Datenbanksystem der Version 10g muss zuerst folgender Parameter gesetzt werden:

```
alter system system set PRE_11g_ENABLE_CAPTURE=TRUE;
```

Folgende Skriptausführung stellt eine alternative Vorgehensweise dar:

```
@$ORACLE_HOME/rdbms/admin/wrrenbl.sql
```

Im nächsten Schritt muss ein (leeres) Directory zur Verfügung gestellt werden. Folgendes Kommando legt ein logisches Verzeichnis an und erstellt die entsprechenden Privilegien:

```
create directory capdir as '&pfad';
grant read, write on directory &dirname to &user;
```

Es kann sinnvoll sein, gleich bei der Aufzeichnung einen Filter zu nutzen, um die Capture-Menge auf den wesentlichen Testanteil zu reduzieren. Filterarten sind dabei unter anderem USER wie SYS und SYSMAN, MODULE, SERVICE. Das Filtern kann außerdem exklusiv oder inklusiv erfolgen. Folgendes Beispiel nutzt die Filterart USER. Es soll nur ein bestimmtes Schema aufgezeichnet werden.

```
execute dbms_workload_capture.add_filter(-
   fname       => '&filtername',-
   fattribute  => 'USER',-
   fvalue      => '&schema');
```

Die Überprüfung der Filter erfolgt mit folgendem Kommando:

```
select name, status from dba_workload_filters;
```

Nun wird die Aufzeichnung gestartet. Da wir einen USER-Filter nutzen und nur diese Information aufzeichnen wollen, müssen wir die Option EXCLUDE festlegen; die Defaulteinstellung ist INCLUDE.


```
execute dbms_workload_capture.start_capture(-
      name             => '&capturename',-
      dir              => '&dir',-
      default_action   => 'EXCLUDE');
```

Soll die Endzeit festgelegt werden, können wir zusätzlich eine Zeitdauer über den Parameter *duration* angeben.

```
execute dbms_workload_capture.start_capture(-
      name             => '&capturename',-
      dir              => '&dir',-
      duration         => &sec);
```

Ab Version 11.2.0.2, allerdings nur in Single-Instance-Umgebungen, gibt es sogar die Option, ein SQL Tuning Set aufzuzeichnen.

```
execute dbms_workload_capture.start_capture(-
      name             => '&capturename',-
      dir              => '&dir',-
      capture_sts      => TRUE);
```

Soll manuell gestoppt werden, ist folgender Aufruf erforderlich:

```
execute dbms_workload_capture.finish_capture();
```

Nun erfolgt noch ein Export des AWRs, damit am Ende des Replays ein Vergleich mit dem Capture AWR durchgeführt werden kann. Und fertig!

```
execute dbms_workload_capture.export_awr(capture_id =>
&captureid);
```

Um am Ende einen guten **Überblick über das Capture** zu bekommen, kann man entweder einen Capture-Bericht generieren oder eine Abfrage auf die View DBA_WORKLOAD_CAPTURES durchführen lassen. So erhält man beispielsweise Informationen über die Capture ID, die Größe des Capture, die Anzahl der User Calls, die aufgezeichnete DB Time und die Start-SCN für den Aufbau des Testsystems. Wenn in der Fehlerspalte ERRORS Fehler auftauchen, ist dies übrigens kein Grund zur Beunruhigung; dies kann in der Regel ignoriert werden. Die Datenbankzeit in Prozent (in unserem SQL-Beispiel *dbzeit_in_prozent*) gibt an, wie viel Datenbankzeit insgesamt aufgezeichnet worden ist. Da Background-Operationen, beispielsweise die Nutzung von DBMS_SCHEDULER-Jobs, nicht aufgezeichnet werden, kann diese Spalte in der Regel nie 100 Prozent erreichen.

```
select id, name, status, start_time, start_scn,
dbtime*100/dbtime_total dbzeit_in_prozent, user_
calls, user_calls_unreplayable, errors, awr_exported,
duration_secs, filters_used, capture_size/1024/102
from dba_workload_captures
where name like '&replayname';
```


Nun ist es an der Zeit, das Testsystem vorzubereiten. Dazu gehört das Einspielen des Backups, gegebenenfalls eine Migration, das Setzen von Flashback-Einstellungen – falls erwünscht – und die Kopie der Capture-Dateien.

Der erste Schritt nach dem Capture ist das Processing des Workloads. Dieser Vorgang muss nur genau einmal durchgeführt werden und kann auf dem Testsystem oder auf jedem anderen Server erfolgen. Damit es keine Engpässe gibt, sollte stets sichergestellt sein, dass ausreichend Platz im SYSTEM Tablespace vorhanden ist.

```
execute dbms_workload_replay.process_capture(-
        capture_dir          => '&dir',-
        parallel_level       => &parallel);
```

Möchte man den Fortschritt des Processing überwachen, kann folgendes Skript hilfreich sein:

```
set serveroutput on
declare
   p number;
   z number;
begin
   p:=dbms_workload_replay.process_capture_completion;
   z:=dbms_workload_replay.process_capture_remaining_time;
   dbms_output.put_line('in Prozent:'||p);
   dbms_output.put_line('in Zeit:'||z);
```

```
end;
/
```

Ist das Processing **erfolgreich** durchgeführt worden, ist die Spalte LAST_PROCESSED_VERSION aus DBA_WORKLOAD_CAPTURES mit der entsprechenden Versionsnummer gefüllt (zum Beispiel 11.2.0.3.0).

Nach dem Processing des Workload, sollte man, wie oben schon erwähnt, einen Workload Analyzer Report generieren. Der Workload Analyzer benötigt mindestens die Java Version 1.5 und besteht aus den zwei Teilen *dbranalyzer.jar* und *dbrparser.jar*, die im *$ORACLE_HOME/rdbms/jlib/* Verzeichnis einer 11.2.0.2-Installation zu finden sind:

```
java -classpath
$ORACLE_HOME/jdbc/lib/ojdbc5.jar:$ORACLE_HOME/
rdbms/jlib/dbrparser.jar:$ORACLE_HOME/rdbms/jlib/
dbranalyzer.jar:
oracle.dbreplay.workload.checker.CaptureChecker
<physikalisches Directory> jdbc:oracle:thin:@<maschine
>:<listenerport>:<SID>
```

Bevor wir nun mit dem eigentlichen Replay starten, generieren wir einen Restore Point, um die Datenbank später nach dem Replay-Lauf bis zu diesem Punkt zurückführen zu können:

```
create restore point pre_lauf guarantee flashback database;
```


Soll jetzt erst ein Filtering durchgeführt werden, kann dies mit zwei Kommandos erfolgen. Dabei werden die Filterattribute und der Parameterwert EXCLUDE und INCLUDE

(Default) für *default_action* wie beim Capture-Filter verwendet. Diese Schritte sind optional.

```
begin
   dbms_workload_replay.add_filter(
         fname          => '&filtername',
         fattribute     => 'USER',
         fvalue         => '&schema');
end;
/
begin
      dbms_workload_replay.create_filter_set(
         replay_dir             => '&replaydir',
         filter_set             => '&filterset_name',
         default_action         => 'EXCLUDE');
end;
/
```

Nun wird das Replay initialisiert. Das logische Directory und ein Name für das Replay müssen dabei bekanntgegeben werden.

```
execute dbms_workload_replay.initialize_replay(-
         replay_name    => '&replayname', -
         replay_dir     => '&dir');
```

Soll ein Replay-Filter verwendet werden, muss dies mit einem zusätzlichen Aufruf erfolgen:

```
begin
   dbms_workload_replay.use_filter_set(filter_set =>
'&filterset_name');
end;
/
```

Der nächste Schritt gibt die Replay-Optionen an. Mit *capture_sts* kann zusätzlich ein SQL Tuning Set aufgezeichnet werden.

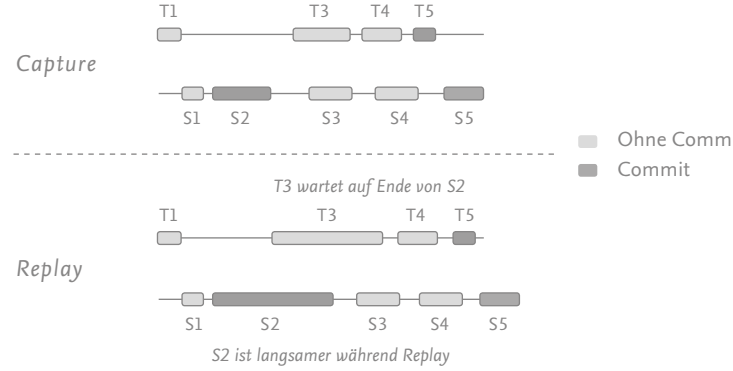
```
begin
   dbms_workload_replay.prepare_replay(
      synchronization          => &SYNC_TRUE_FALSE,
      connect_time_scale       => &CONNECT_TIME1_100,
      think_time_scale         => &THINK_TIME1_100,
      capture_sts              => &TRUE_FALSE);
end;
/
```

Der Parameter *synchronization* kann den Wert TRUE (Default-einstellung), FALSE oder OBJECT_ID haben. TRUE oder FALSE schalten die COMMIT-Synchronisierung (auch: preserve commit order) ein oder aus. Ist die Synchronisierung mit TRUE eingeschaltet, wird die COMMIT-Reihenfolge des Capture Workloads beibehalten. Jede Aktion wird erst dann abgespielt, wenn alle COMMIT-Aktionen, die vorher ausgeführt wurden, beendet sind. Dies kann jedoch zu einer Verzögerung beim Abspielen führen. Die Metriken *db time*, *cpu* usw. bleiben

davon unberührt. Allerdings kann es zu Irritationen bei der Testauswertung kommen, wenn das Replay im Extremfall ein Vielfaches der Capture-Dauer beträgt (siehe Abb. 3). Nutzt man hingegen den Wert FALSE kann es zu Divergenzen kommen. Sind diese gering, dann spricht jedoch nichts gegen diese Art des Replays, wie wir in mehreren Tests beobachten konnten.

Der Wert OBJECT_ID hingegen bedeutet, dass jede Aktion dann abgespielt wird, wenn die relevanten COMMIT-Aktionen beendet sind. Die relevanten COMMIT-Aktionen sind diejenigen, die mindestens ein Datenbank-Objekt verändert haben. Diese Art der Synchronisierung sollte unbedingt berücksichtigt werden.

Abb.3: Synchronisierungs-Regelwerk beim Replay



Die Parameter *connect_time_scale* und *think_time_scale* sind für die Skalierung von *connection* beziehungsweise die *think time* zuständig. Auf diese Weise kann der aufgezeichnete Workload mit unterschiedlicher „Geschwindigkeit" ablaufen. Folgendes Beispiel zeigt die Verwendung von *connect_time_scale*. Die Capture-Aufzeichnung sieht dabei folgendermaßen aus:

```
12:00 : Capture ist gestartet
12:10 : Erster Session Connect
12:30 : Zweiter Session Connect
12:42 : Dritter Session Connect
```

Mit *connect_time_scale* Einstellung 50 entsteht folgender Ablauf – die Angabe des Wertes erfolgt in Prozent:

```
12:00 : Replay ist gestartet
12:05 : Erster Session Connect
12:15 : Zweiter Session Connect
12:21 : Dritter Session Connect
```

Nun sind alle Vorbereitungen getroffen und die WRCs werden gestartet. Um Auskunft über die Anzahl der Clients zu erhalten, bietet sich die Nutzung des WRC-Werkzeugs an.

```
wrc user/password@<connection> replaydir=<phys.
directory> mode=calibrate
```


Als Alternative bietet sich folgendes Skript an. Die Ausgabe ist in XML und sehr nützlich. So wird nicht nur die Anzahl der WRCs aufgelistet, sondern es werden auch Informationen aus DBA_WORKLOAD_CAPTURES zum aktuellen Capture und Preprocessing-Status zusammengefasst.

```
set serveroutput on
declare
 result varchar2(4000);
begin
   result := dbms_workload_replay.calibrate('&replaydir');
   dbms_output.put_line(result);
end;
/
```

Nun wird das Replay mit einem einzigen Aufruf gestartet:

```
begin
   dbms_workload_replay.start_replay();
end;
/
```

Die View DBA_WORKLOAD_REPLAY gibt Auskunft über den Stand des Replays.

```
select name, id, status, user_calls, awr_exported,
start_time, end_time
from dba_workload_replays order by start_time;
```

Häufig stellt sich die Frage, ob der Anwender DBA-Privilegien zur Nutzung von DB Replay benötigt. Dies ist nicht erforderlich. Folgende Privilegien reichen zur Nutzung aus:

- EXECUTE für die Packages DBMS_WORKLOAD_CAPTURE und DBMS_WORKLOAD_REPLAY
- CREATE ANY DIRECTORY
- CREATE SESSION
- SELECT_CATALOG_ROLE
- BECOME USER
- ADMINISTER SQL TUNING SET

2.2 BERICHTGENERIERUNG IM LINEMODE

Nun können die entsprechenden Reports generiert werden, um eine Bewertung des Replay-Laufs durchzuführen. Alle Reports lassen sich im Linemode generieren und stehen häufig in TEXT-, HTML-oder XML-Format zur Verfügung. Einfacher zu erzeugen sind diese natürlich über die grafische Oberfläche, allerdings wollen wir uns in diesem Dojo auf die Linemode-Version konzentrieren. Generell sind folgende Reports möglich, die jedoch unterschiedliche Bewertungsmaßstäbe liefern:

- Der **Workload Analyzer Report** zeigt Merkmale und Charakteristiken der Capture-Dateien vor dem Replay (siehe Abschnitt oben) auf.
- Der **Replay Report** gibt erste Informationen zu Replay-Statistiken, Top-Events, Workload-Profil und Divergenzen.
- Der **Divergence Report** listet die Divergenzen im Detail.
- Der **Compare Period Report** gibt erste Informationen zu der Performance im Vergleich zum Capture oder einem weiteren Replay.
- Der **AWR Difference Report** beziehungsweise **AWR Report des Capture** und des **Replays**
- Der **ASH Report**

Die letzten beiden Reports sind gängige und bekannte Mittel beim Datenbank-Tuning. Die ersten vier hingegen sind typische Real Application Testing Reports. Konzentrieren Sie sich zuerst auf den Replay Report und den Divergence Report, damit Sie einen Überblick über das Ergebnis erhalten. Falls zu große Diskrepanzen bestehen, sollten Sie das Backup oder auch das Setup Ihrer Testumgebung überprüfen. Danach können Sie sich der Performance-Analyse über den Compare Period Report widmen. Dazu ist die AWR- Datei des Capture nötig, die falls noch nicht erfolgt, in einem zusätzlichen Schritt importiert wird. Der Compare Period Report liefert dabei einen guten Überblick über den Zustand der Performance und fokussiert auf die wichtigsten für die Performance relevanten Informationen. Die AWR Reports, insbesondere der AWR Difference Report, runden diese Informationen in einem letzten Schritt ab.

Der Import des AWRs erfolgt über die Funktion *import_awr*. Die *capture_id* kann dabei aus der View DBA_WORKLOAD_CAPTURES gewonnen werden.

```
declare
   dbid number;
   begin
      dbid := dbms_workload_capture.import_awr(
      capture_id               => &captureid,
      staging_schema           => '&schema',
```



```
      force_cleanup            => TRUE);
end;
/
```

Zur Kontrolle eignet sich die View DBA_HIST_SNAPSHOT. Hier müssten jetzt alle Snapshots aufgelistet sein.

```
select snap_id, instance_number, dbid,
   to_char(BEGIN_INTERVAL_TIME, 'dd.mm.yy hh24:mi'),
   to_char(end_interval_time, 'dd.mm.yy hh24:mi')
from dba_hist_snapshot order by begin_interval_time;
```

Der Replay Report lässt sich über einen einzigen Aufruf generieren. In unserem Beispiel wird das Ergebnis in der SQL*Plus Variable c gespeichert und über eine Spooldatei ausgegeben.

```
variable c clob
set long 100000 head off pagesize 0
execute :c:=dbms_workload_replay.report(-
         replay_id      => &replayid,-
            format      => 'HTML');
spool replayreport.html rep
print c
spool off
```

Im Replay Report werden generelle Informationen zum Replay und Capture, die Replay-Optionen und einige Replay-Statistiken (wie *db time*, *user calls*) aufgelistet und

die Replay-Divergenzen ausgegeben. Die Ergebnisse über Divergenzen können einen gewissen Überblick das Replay geben und beispielsweise grobe Verstöße und Probleme beim Abspielen aufdecken.

Nun bleibt noch die Aufgabe, die Performance genauer zu analysieren. Dies geschieht über den Compare Period Report und den AWR Difference Report. Der Compare Period Report wird ähnlich wie der Replay Report über einen Aufruf des Package DBMS_WORRKLOAD_REPLAY generiert und das Ergebnis in einer SQL*Plus Variable gespeichert.

```
variable ergebnis clob

begin
   dbms_workload_replay.compare_period_report(
         replay_id1     => &replayid,
+        replay_id2     => null,
         format         => 'HTML',
         result         => :ergebnis);
end;
/
set heading off set long 100000 set pagesize 0
spool comparereport.html rep
print ergebnis
spool off
```

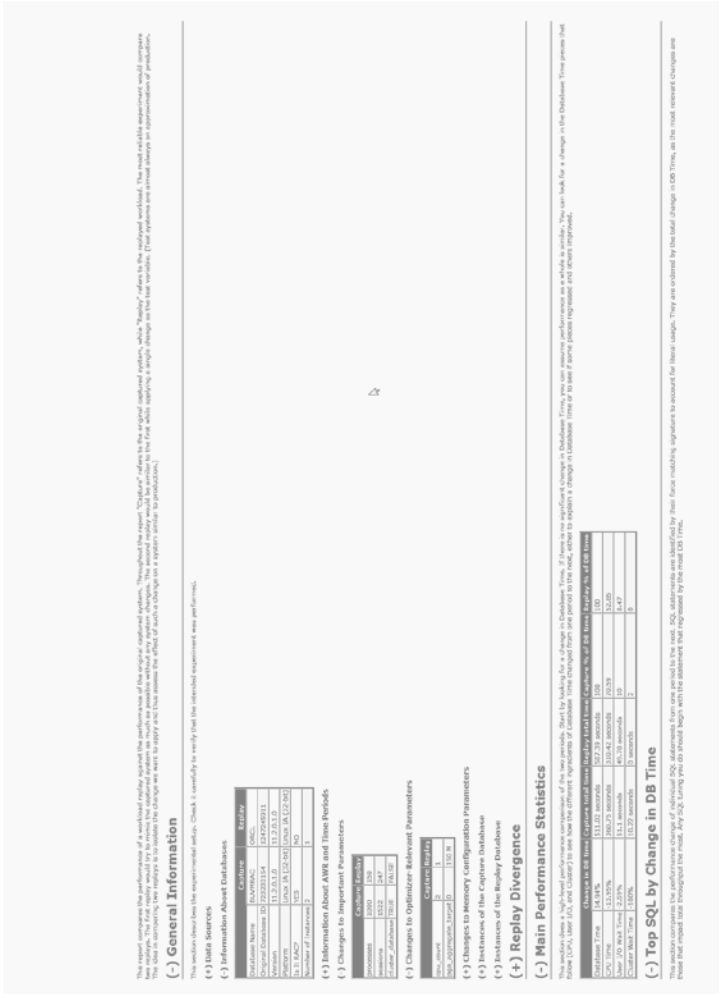


Abb. 4: Ausschnitt aus einem Compare Period Report

Dieser Report (siehe Abb. 4) kann normalerweise einige entscheidende Hinweise über die Performance-Unterschiede geben.

Es werden Optimizer- und Memory-Einstellungen, wichtige Initialisierungsparameter, Performance-Statistiken und Top-Statements miteinander verglichen und zusätzlich einige Hardware-Statistiken ausgegeben wie I/O- oder CPU-Nutzung. Am Schluss werden noch die Ergebnisse des ADDM-Laufs aufgelistet.

Möchten Sie genauere Informationen über die einzelnen Performance-Metriken erhalten, lässt sich zum Schluss noch ein AWR Difference Report mit dem Skript *awrddrpi* sowie einzelne AWR Reports generieren:

```
@$ORACLE_HOME/rdbms/admin/awrddrpi.sql
```

2.3 WICHTIGE TIPPS

Bevor Sie mit Database Replay starten, beachten Sie bitte noch folgende Hinweise:

- Überprüfen Sie, ob Ihr Capture Workload auch keine „**non supported Features**“ beinhalten wird (genauer: siehe Application-Testing-Handbuch).

- Planen Sie einen **ersten Testlauf** für Capture, Processing und Replay ein. Sie machen erste Erfahrungen und bekommen einen Überblick über den Capture-Umfang und über die möglichen Replay-Optionen.
- Überprüfen Sie Ihr Testsystem auf **Vollständigkeit** (Datenbank-User, Datenbank-Tabellen usw.). Abgespielt wird nämlich immer, auch wenn Daten nicht korrekt oder sogar unvollständig sind. Allerdings werden dadurch sehr viele Fehler entstehen, die das Replay-Resultat wertlos machen. Eine Kompromisslösung ist, das Capture bei geringer Last zu starten.
- Fahren Sie (wenn möglich) vor dem Capture die **Datenbank herunter**, um möglichst wenige sogenannte In-flight-Transaktionen, also noch offene Transaktionen, zu haben, die nicht aufgezeichnet werden können. Eine Kompromisslösung ist auch hier, das Capture bei geringer Last zu starten.
- Begrenzen Sie die **Capture-Dauer** auf wenige Stunden. Ideal sind ein bis zwei Stunden.
- Nutzen Sie beim ersten Abspielen die **gleichen Einstellungen** (zum Beispiel Initialisierungsparameter) wie beim Capture. Vergessen Sie dabei nicht die gesetzten Underscore-Parameter.

- Falls die Zeit und Ihr Workload es erlauben, ziehen Sie das **Tuning mit SPA** in Betracht. Am besten tunen Sie vor dem Replay zuerst die Applikation mit SPA. Sie ersparen sich dadurch unter Umständen lange Abspielzeiten.
- Nutzen Sie **Guaranteed Restore Points** für das DB Replay, um wieder einfach zu ihrem Testausgangspunkt zurückzukehren. Sorgen Sie dann allerdings dafür, dass ausreichend Platz für Flashback und Archive-Log-Dateien zur Verfügung steht.
- Verwenden Sie vor dem Abspielen die Informationen aus dem **Workload Analyzer Report** – entweder über die grafische Oberfläche (ab Version 11.2.0.2) oder über den Linemode-Aufruf (vgl. auch Skript-Download, siehe Kapitel 6).
- Überlegen Sie, wie Sie mit **externen Quellen** wie DB-Links, External Tables oder URLs umgehen, ob Sie diese zur Verfügung stellen können oder nicht.
- Planen Sie **ausreichend Zeit für das Replay** und die Nutzung verschiedener Replay-Optionen ein.

2.4 ANWENDUNGS- UND SPEZIALFÄLLE

DB Replay kann in Single-Instance-Umgebungen wie auch in **RAC-Umgebungen** Anwendung finden. Dabei macht es keinen Unterschied, ob das Aufzeichnen und Abspielen in einem homogenen RAC-Umfeld, Single-Instance-Umfeld oder in einem gemischten Umfeld erfolgt. Verwenden Sie zum Aufzeichnen, Processing und Abspielen am besten ein Shared Filesystem. Das Aufzeichnen kann zwar in unterschiedlichen physikalischen Directories erfolgen, aber zum Processing und Abspielen müssen alle lokalen Capture Directories von jeder Instance zur Verfügung stehen. Die Dateien muss man dann in ein gemeinsames Verzeichnis kopieren/konsolidieren.

Während des Aufzeichnens werden auch die Connection Strings aufgezeichnet. Möchte man auf dem Replay-System ein erfolgreiches Abspielen gewährleisten, muss unter Umständen ein **Mapping der Connections** erfolgen. Für RAC-Datenbanken kann man zum Beispiel alle Connection Strings auf einen Load Balancing Connection String abbilden. Ein anderer Anwendungsfall wäre das Umleiten eines speziellen Workloads auf eine spezielle Instance. Ein Mapping der Connection ID 101 könnte dann folgendermaßen aussehen:

```
execute DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION
(connection_id => 101, replay_connection =>
'<host>:<port>/SERVICE1');
```

Dabei kann man für den *replay_connection*-Parameter einen beliebigen Connection Identifier (wie Net-Service-Name, Database-Service oder Net-Service-Aliase) und eine beliebige Naming-Methode wählen.

Um das aktuelle Mapping zu überprüfen, verwenden Sie die View DBA_WORKLOAD_CONNECTION_MAP. Sie gibt Auskunft über die Connection und Replay IDs und den zugehörigen Connection Strings.

Einen weiteren Anwendungsfall stellt die **Verlagerung der WRCs** auf einen oder mehrere externe Server dar. Dazu ist keine Oracle-Software-Client-Installation notwendig. Es reicht aus, die WRC Instant Client Software von OTN (zum Beispiel „Instant Client Downloads for Linux x86-64“ unter *www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html*) auf den entsprechenden oder die entsprechenden Server zu kopieren. Die WRCs werden dann wie im Beispiel aus Kapitel 2.1 verwendet. Wichtig ist, dass allen WRCs zu jeder Zeit alle Processed-Dateien zur Verfügung stehen. Möchte man mehr über die Verwendung der WRC Clients erfahren, eignet sich der folgende Aufruf, der alle möglichen Einstellungen, Parameter und eine Kurzbeschreibung auflistet:


```
[oracle@sccloud003 ~]$ wrc
Workload Replay Client: Release 11.2.0.3.0 - Production
on Mon Jun 25 11:35:45 2012
FORMAT:
=======
wrc [user/password[@server]] [MODE=mode-value]
KEYWORD=value
...
Mode:
=====
wrc can work in different modes to provide additional
functionalities.
The default  MODE is REPLAY.
Mode         Description
-----------------------------------------------------
REPLAY       Default mode that replays the workload in
             REPLAYDIR
CALIBRATE    Estimate the number of replay clients
             and CPUs needed to replay the workload in
             REPLAYDIR.
LIST_HOSTS   List all the hosts that participated in
             the capture or replay.
Options (listed by mode):
=========================
MODE=REPLAY (default)
---------------------
```

```
Keyword      Description
----------------------------------------------------
USERID       username
PASSWORD     password
SERVER       server connection identifier (Default:
             empty string)
REPLAYDIR    replay directory (Default:.)
WORKDIR      directory for trace files
DEBUG        ON, OFF (Default: OFF)
CONNECTION_OVERRIDE TRUE, FALSE (Default: FALSE)
TRUE         All replay threads connect using SERVER,
             settings in DBA_WORKLOAD_CONNECTION_MAP
             will be ignored!
FALSE     Use settings   from DBA_WORKLOAD_CONNECTION_MAP
SERIALIZE_CONNECTS       TRUE, FALSE (Default: FALSE)
          TRUE           All the replay threads will
                         connect to the database in
                         a serial fashion one after
                         another. This setting is
                         recommended when the replay
                         clients use the bequeath
                         protocol to communicate to the
                         database server.
          FALSE          Replay threads will connect to
                         the database in a concurrent
                         fashion mimicking the original
                         capture behavior.
```



```
DSCN_OFF  TRUE,  FALSE (Default: FALSE)
      TRUE       Ignore all dependencies due to
                 block contention during capture when
                 synchronizing the replay.
      FALSE      Honor all captured dependencies.
```

2.4.1 CONSOLIDATED DATABASE REPLAY

Gibt es die Möglichkeit, mehrere Workloads von verschiedenen Servern auf einem Rechner abzuspielen? Typische Beispiele sind Szenarios, in denen man feststellen möchte, ob eine Konsolidierung erfolgreich sein wird. Als zentrales Verwaltungssystem sammelt beispielsweise Oracle Enterprise Manager Cloud Control 12c Daten hinsichtlich der Nutzung von IT-Ressourcen und nutzt diese Informationen als Planungsgrundlage für Konsolidierungsprojekte im sogenannten Consolidation Planner. Dieser ist im Plug-in Oracle Chargeback and Capacity Planning enthalten.

Unabhängig davon stellt DB Replay mit der neuen Technologie **Consolidated Database Replay** eine Möglichkeit zur Verfügung, ein Replay **mehrerer Capture Workloads** vom gleichen Server oder von verschiedenen Servern durchzuführen. So können Konsolidierungsprojekte verifiziert und Probleme und Engpässe vorab beseitigt werden.

Welche Voraussetzungen sind dazu notwendig? Die Datenbank-Version des Servers muss mindestens 11.2.0.2 sein. Zusätzlich ist ein Patch für das Replay-System erforderlich (siehe MOS Note 1453789.1). Das Capture-System bleibt allerdings unverändert; eine Aufzeichnung kann ab der Version 9.2.0.8 erfolgen. Bevor das Replay gestartet werden kann, sollten, wie schon erwähnt, die Applikationsdaten des Capture-Systems mit dem Startpunkt der Aufzeichnung übereinstimmen. Eine weitere Voraussetzung ist die Trennung der unterschiedlichen Capture Workloads in **unterschiedliche Schemata**. Dies bedeutet, dass vor dem Replay die einzelnen Schemata auf dem Zielsystem „restored" werden müssen. Es liegt in der Natur der Sache, dass auch die Processing-Dateien in unterschiedlichen Directories liegen. Für das Replay müssen diese Directories allerdings in ein gemeinsames Verzeichnis kopiert werden. Um diese Aktionen zu ermöglichen, wurde das Package **DBMS_ WORKLOAD_ REPLAY** durch einige Prozeduren erweitert. Das weitere Vorgehen beim Replay gleicht ansonsten dem Verfahren in Kapitel 2.1 Skripte und eine Dokumentation zum gesamten Ablauf finden sich in der MOS Note 1453789.1.

3 Die Komponente SQL Performance Analyzer

Der SQL Performance Analyzer (SPA) ist unabhängig von Database Replay nutzbar und ist eine gute Ergänzung und Vorbereitung für den DB-Replay-Testlauf, wie in Abschnitt 2.2 schon angedeutet. Wie DB Replay lässt sich der SQL Performance Analyzer einfach über die grafische Oberfläche des Oracle Enterprise Managers bedienen und auch im Linemode über Package-Zugriffe sowie SQL-Aufrufe verwenden. Auch hier sollte die Umgebung den Voraussetzungen aus Abschnitt 1.2 genügen.

Der Fokus von SQL Performance Analyzer liegt auf der detaillierten Statement-Analyse eines definierten **SQL Workloads**. Ein SQL Workload besteht dabei aus SELECT- beziehungsweise DML-Statements (nur im Linemode), die über ein SQL Tuning Set (STS) zur Verfügung gestellt werden. Der SQL Workload wird dabei zweimal abgespielt, vor und nach einer Veränderung. Das Ergebnis ist eine detaillierte Vergleichsanalyse der einzelnen Statements gemessen an verschiedenen **Metriken** beispielsweise *elapsed time.* Möchte man nun einen Tuning-Prozess anschließen, ist dies einfach möglich durch die Integration des **SQL Tuning Advisors** oder durch die Nutzung von **SQL Plan Baselines**.

Vorteile von SPA sind die einfache Handhabung, die Flexibilität und die Verfügbarkeit. Da keine DML-Statements die Testumgebung verändern, ist ein Zurücksetzen nach dem Test nicht erforderlich, denn DML-Statements werden automatisch zurückgerollt. Kennt man die Top-Statements, deren Performance beibehalten oder verbessert werden sollen, kann man auch nur diese testen, ohne den gesamten Workload abspielen zu müssen. SPA ist sehr flexibel bei der Auswertung und Wahl der Metrik und der Ausführung selbst. So können alle Statements ein oder mehrfach ausgeführt oder nur der Ausführungsplan erzeugt werden, um beispielsweise keine zusätzliche Last auf dem System zu erzeugen.

Kurz noch eine Definition von **SQL Tuning Sets**: Seit Oracle Database 10g und der Einführung des Tuning Advisory Framework spielen die sogenannten SQL Tuning Sets eine wichtige Rolle. Ein STS besteht dabei nicht alleine aus einer Ansammlung von gespeicherten SQL-Texten. Zusätzlich werden auch Execution-Kontext und Execution-Statistiken im Data Dictionary abgespeichert. Ein SQL Tuning Set besteht somit aus folgenden Komponenten:

- Einem oder mehreren SQL-Statements
- Execution-Kontext wie Bind-Variablen, Parsing-Schema, etc.

- Grundlegenden Execution-Statistiken wie *elapsed time*, *cpu time*, etc.
- Ausführungsplänen und Rowsource-Statistiken (optional)

Hinweis: SQL Tuning Sets (STS) können genutzt werden, falls das Oracle Tuning Pack oder die Oracle Real-Application-Testing-Option lizenziert ist.

Wie erzeugt man nun SQL Tuning Sets? Da eine gute Integration von SPA in DB Replay existiert, können SQL Tuning Sets zum Beispiel automatisch beim **DB Replay Capture** (ab Version 11.2.0.2) oder beim **Replay** erzeugt werden (siehe auch Code-Beispiele in Abschnitt 2.1).

Unabhängig davon ist es möglich SQL Tuning Sets über AWRs oder aus dem Cache zu erzeugen und gegebenenfalls über Data Pump Export und Import auf eine andere Datenbank zu verlagern. Wenn Sie sich genauer über SQL Tuning Sets informieren möchten, können Sie folgende Artikel aus der DBA Community (siehe Kapitel 6) zurate ziehen.

- SQL Tuning Sets im Einsatz Teil 1
- SQL Tuning Sets im Einsatz Teil 2

Hinweis: Im Fall von 9i-Datenbanken können die SQL Tuning Sets über SQL Traces erzeugt werden.

Wie funktioniert SPA? Nach der Erzeugung eines STS wird SPA für den ersten Testlauf gestartet, danach wird die Umgebung verändert und SPA zum zweiten Mal durchgeführt. Bei beiden Durchläufen werden die entsprechenden Statistiken (wie *elapsed time* (default), *cpu time*, *disk reads*, *direct writes*, *optimizer costs*, etc.) gesammelt, die Ausführungspläne gespeichert und der Anteil am Workload notiert, um eine Gewichtung des Statements durchzuführen. Um relevante Ergebnisse zu erzielen, erfahren die Statements ab Oracle Database 11g Release 2 ein „Warm-up" und werden mehrfach ausgeführt. Die Analyse ordnet dabei die Statements nach Relevanz für den Workload, listet die verschiedenen Metriken pro Statement auf, zeigt an, ob Ausführungspläne „gekippt" sind und gibt die Ausführungspläne aus.

Wie im Fall von Database Replay lässt sich SPA auch sehr einfach über die grafische Oberfläche wie Grid Control, Cloud Control oder Database Control anwenden. Auch hier werden wir uns auf die Linemode-Verwendung konzentrieren.

3.1 SQL-PERFORMANCE-ANALYZER-AUSFÜHRUNG IM LINEMODE

Zur Verwendung im Linemode stehen die entsprechenden PL/SQL Packages wie DBMS_SQLPA und DBMS_SQLTUNE zur Verfügung. Der folgende Abschnitt demonstriert die

Verwendung an einem Beispiel; dabei sollen auch DML-Statements berücksichtigt werden. Zur Erinnerung: DML-Statements werden bei Nutzung der grafischen Oberfläche noch nicht unterstützt.

Hinweis: Auch dieser Skript-Code steht zum Download zur Verfügung. Das Readme führt Sie auch hier durch die Anwendung der einzelnen Skripte (siehe Kapitel 6).

Im ersten Schritt wird eine **SQL Tuning Task** erzeugt. Für unseren Fall ist schon das SQL Tuning Set SPA_DML1 vorab generiert worden. Zur Kontrolle überprüfen wir die View USER_ADVISOR_TASKS.

```
variable tname varchar2(100)
execute :tname := dbms_sqlpa.create_analysis_task(-
                      sqlset_name => 'SPA_DML1');
select task_name, status from user_advisor_tasks where
task_name = :tname;

TASK_NAME                      STATUS
------------------------------ -----------
TASK_21137                     INITIAL
```

Da alle DML-Statements berücksichtigt werden sollen, muss eine zusätzliche **Parametereinstellung (EXECUTE_FULLDML)** vorgenommen werden. Der Default ist übrigens FALSE. Weitere Informationen zu den Einstellungen finden Sie im PL/SQL-Handbuch Table 138-10.

```
execute dbms_sqlpa.set_analysis_task_parameter(-
                task_name       => 'TASK_21137',-
                parameter       => 'EXECUTE_FULLDML',-
                value           => 'TRUE');
```

Nun wird der erste Test durchgeführt. Um ein „Warm-up" des Buffer Caches vorzubereiten und um möglichst aussagekräftige Run-Time-Statistiken zu erhalten, werden die Statements ab 11g Release 2 mehrfach ausgeführt (siehe Parametereinstellung DISABLE_MULTI_EXEC). Die Langläufer unter den Statements werden dabei nur zweimal ausgeführt.

```
execute dbms_sqlpa.execute_analysis_task(-
            task_name           => 'TASK_21137',-
            execution_type      => 'TEST EXECUTE',-
            execution_name      => 'Test1');
```

Nun kann die Umgebung entsprechend angepasst werden. Beispiele für Änderungen wären: Einstellungen der Initialisierungsparameter (OPTIMIZER_FEATURES_ENABLE etc.), Änderung der Speicherung (zum Beispiel Komprimierung, Hinzufügen von Indizes und vieles mehr). Danach erfolgt ein weiterer SPA-Testlauf.

```
execute dbms_sqlpa.execute_analysis_task(-
         task_name              => 'TASK_21137,-
         execution_type         => 'TEST EXECUTE',-
         execution_name         => 'Test1');
```


Nun kann die Auswertung erfolgen. Die Metrik *buffer_gets* ist für die Bewertung in unserem Beispiel ausschlaggebend. Andere mögliche Bewertungsmaßstäbe stehen über die Metrik-Einstellungen *cpu_time*, *disk_reads*, *optimizer_cost*, *elapsed_time (default)* oder *direct_writes* zur Verfügung.

```
execute dbms_sqlpa.execute_analysis_task(-
          task_name          => 'TASK_21137',-
          execution_type     => 'COMPARE PERFORMANCE',-
          execution_name     => 'Vergleich1',-
          execution_params   => dbms_advisor.arglist
                                ('comparison_metric',-
                                'buffer_gets'));
```

Seit 11g Release 2 gibt es auch die Möglichkeit **zwei SQL Tuning Sets** miteinander zu vergleichen. Dabei wird jedes SQL Tuning Set einzeln ausgeführt. Dies ist besonders dann interessant, wenn unterschiedliche Implementierungen einer Anwendung getestet werden sollen. Anstelle TEST EXECUTE als *execution_type* wird dabei der Wert CONVERT verwendet.

Nun werden die zugehörigen Berichte erzeugt. Mögliche Formate sind TEXT, HTML oder XML. Mit dem Parameter *level* lassen sich zusätzlich Detail Reports zu den einzelnen Themenbereichen generieren wie Berichte über Statements mit geänderten Plänen (CHANGED_PLANS), verschlechterte (REGRESSED) oder verbesserte (IMPROVED) Statements.

Folgende Beispiele geben eine Einführung in die Nutzung. Als Voraussetzung wurde eine SQL*Plus Variable *ergebnis* zur Verfügung gestellt.

```
execute :ergebnis := dbms_sqlpa.report_analysis_task(-
         task_name      => 'TASK_21137',-
         type           => 'HTML',-
         section        => 'SUMMARY');
```

Folgender Report zeigt Details zu Statements mit geänderten Plänen:

```
execute :ergebnis:      = dbms_sqlpa.report_analysis_
                          task(-
         task_name      => 'TASK_21137',
         type           => 'HTML',-
         level          => 'CHANGED_PLANS',-
         section        => 'ALL');
```

Folgendes Beispiel listet Statements auf, die sich verschlechtert (regressed) haben:

```
execute :ergebnis:      = dbms_sqlpa.report_analysis_
                          task(-
         task_name      => 'TASK_21137', -
         type           => 'HTML', -
         level          => 'REGRESSED',-
         section        => 'ALL');
```


Die Ausgabe erfolgt in allen drei Fällen über eine Spooldatei.

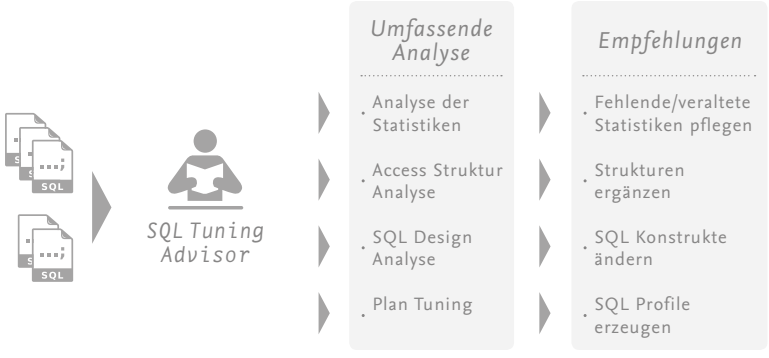
```
set long 1000000 longchunksize 1000000 linesize 200
head off feedback off
echo off
spool report.html
print ergebnis
spool off
```

3.2 TUNING NACH DER SQL-PERFORMANCE-ANALYZER-AUSFÜHRUNG

Nachdem wir im letzten Schritt einen Überblick über die Performance-Metriken und Pläne der Statements erhalten haben, heißt es nun, den Umfang der Performance-Unterschiede abzuschätzen. Bei großen Unterschieden und vielen betroffenen Statements sollte die Ursache in der Gesamtkonfiguration gesucht werden. Falls nur einige Statements betroffen sind, gibt es zwei Optionen wie man mit „regressed statements“ umgehen kann. Die erste Option wäre den **SQL Tuning Advisor** zu nutzen und die Empfehlungen – wie Profiles, Statementänderungen, Statistiken, alternative Pläne und ähnliches – zu implementieren (siehe Abb. 5).

Hinweis: Hierzu muss das Oracle Tuning Pack lizenziert sein.

Abb. 5: SQL Tuning Advisor



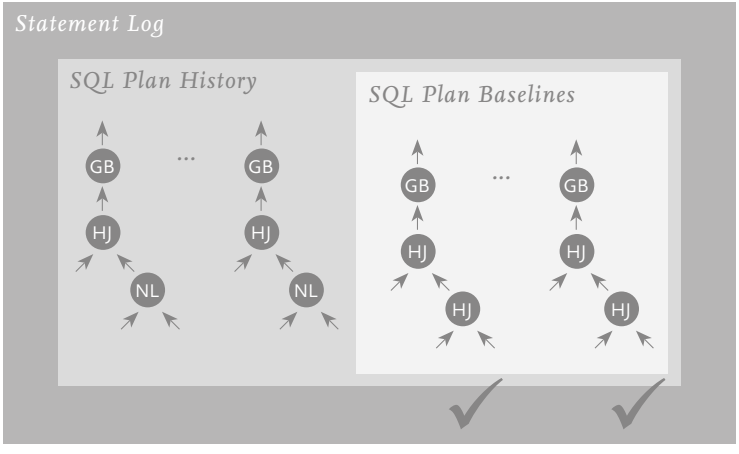
Eine weitere Option wäre, mit **SQL Plan Baselines** zu arbeiten. Die Idee ist, die Statements mit ihren originalen Ausführungsplänen als SQL Plan Baselines festzulegen. Kurz zur Erinnerung: Wie funktionieren SQL Plan Baselines? Voraussetzung für das SQL-Plan-Management ist das Aufzeichnen von SQL-Plänen sich wiederholender Statements im SYSAUX Tablespace. Nach wiederholter Ausführung des Statements wird der SQL-Plan in eine sogenannte SQL Plan Baseline aufgenommen und gilt somit als Maßstab für die folgenden Ausführungen. Ändert sich beim nächsten Parsen der Ausführungsplan, wird dieser neue SQL-Plan nicht genutzt, sondern in der sogenannten SQL-Plan-Historie

abgespeichert. Der initiale SQL-Plan, die SQL Plan Baseline, kommt zur Ausführung. So werden zu jedem Statement die SQL-Pläne in der SQL-Plan-Historie zusätzlich zu den SQL Plan Baselines gespeichert (siehe Abb. 6). SQL-Pläne aus der Historie kommen erst dann zum Einsatz, wenn sie für den Optimizer „akzeptabel“ und „enabled“ sind und zur SQL Plan Baseline gehören.

Da man existierende Ausführungspläne einfach aus einem SQL Tuning Set als Baseline importieren kann, muss **im Falle von SPA** das SQL Tuning Set nur auf die sogenannten „regressed statements“ reduziert werden. Danach reicht ein einziger Ladevorgang aus und die Statements nutzen die originalen Ausführungspläne als SQL Plan Baseline.

Hinweis: SQL Plan Baselines stehen ohne zusätzliche Installation in der Enterprise Edition der Datenbank zur Verfügung.

Abb. 6: SQL Plan Baselines



Beide Möglichkeiten – die Nutzung des SQL Tuning Advisors oder die SQL-Baseline-Methode – sind vollständig integriert und können entweder in der grafischen Oberfläche oder im Linemode verwendet werden. Nutzt man die grafische Oberfläche zum Beispiel im Enterprise Manager Database Control, lassen sich beide Möglichkeiten durch einen einfachen Klick auf den Button „Run SQL Tuning Advisor“ oder „Create SQL Plan Baselines“ implementieren.

Möchte man auch diese beiden Optionen im Linemode durchführen, bietet sich die Nutzung der Packages DBMS_SQLTUNE und DBMS_SPM an. Folgendes Beispiel zeigt die Implementierung des **SQL Tuning Advisors** im Linemode:

```
variable name varchar2(200)
execute :name:= dbms_sqltune.create_tuning_task(-
         spa_task_name          => 'TASK_21137', -
         spa_task_owner         => 'SYS', -
         spa_compare_exec       => 'Vergleich1',-
         task_name              => 'TUNING_SPA');
execute dbms_sqltune.execute_tuning_task(task_name =>
:name);
```

Das Ergebnis wird dann folgendermaßen ausgegeben:

```
set long 1000000 longchunksize 1000000 linesize 200
head off feedback off echo off
variable report clob
execute :report := dbms_sqltune.report_tuning_task(-
         task_name      => 'TUNING_SPA',-
         type           => 'TEXT',-
         level          => 'TYPICAL',-
         section        => 'ALL');
print report
...
```

```
-------------------------------------------------------
         Global SQL Tuning Result Statistics
-------------------------------------------------------
Number of SQLs Analyzed                        : 3
Number of SQLs in the Report                   : 3
Number of SQLs with Findings                   : 3
Number of SQLs with Statistic Findings         : 3
Number of SQLs with Alternative Plan Findings  : 3
Number of SQLs with Index Findings             : 2
-------------------------------------------------------
SQLs with Findings Ordered by Maximum (Profile/Index)
Benefit, Object ID

-------------------------------------------------------------------------------
object ID SQL ID          statistics  profile(benefit)  index(benefit) restruc
--------- --------------- ----------- ---------------- -------------- ------
        3 29d388ssdjrrj   1                            99.34%
        4 9t8bgrfusbfav   1                            99.34%
        2 cyzznbykb5O9s   1
-------------------------------------------------------------------------------
Objects with Missing/Stale Statistics (ordered by schema, object, type)
-------------------------------------------------------------------------------
Schema Name     Object Name     Type          State         Cascade
--------------- --------------- ------------- ------------- --------------
SH              T               TABLE         STALE         NO
...
```


Die Empfehlungen in unserem Beispiel weisen darauf hin, dass Statistiken veraltet sind, Indizes hinzugefügt werden könnten und alternative Pläne existieren. Dies sollte nun implementiert werden. Auch hier gibt es eine Linemode-Unterstützung über die Funktion SCRIPT_TUNING_TASK. In folgendem Beispiel werden die SQL-Statements zur Implementierung generiert:

```
SQL> select dbms_sqltune.script_tuning_task('TUNING_
SPA') from dual;
DBMS_SQLTUNE.SCRIPT_TUNING_TASK('TUNING_SPA')
-------------------------------------------------------
-- Script generated by DBMS_SQLTUNE package, advisor
framework --
-- Use this script to implement some of the
recommendations--
-- made by the SQL tuning advisor. --
-- --
-- NOTE: this script may need to be edited for your system --
-- (index names, privileges, etc) before it is
executed. --
-------------------------------------------------------
execute dbms_stats.gather_table_stats(ownname =>
'SH', tabname => 'T', estimate_percent => DBMS_STATS.
AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE
AUTO');
```

```
create index SH.IDX$$_52990002 on SH.T("PROMO_ID");
create index SH.IDX$$_52990001 on SH.T("PROMO_ID");
```

Möchte man hingegen **SQL Plan Baselines** für die Statements mit „gekippten" Ausführungsplänen nutzen, sind folgende Schritte erforderlich. Zuerst muss eine SQL-Tuning-Set-Untermenge (SUB_STS1) für die „schlechten" (regressed) Statements erzeugt werden.

```
execute dbms_sqltune.create_sqlset
      (sqlset_name => 'SUB_STS1',-
      description => 'Regressed Statements');
declare
   sqlset_cur dbms_sqltune.sqlset_cursor;
 begin
   open sqlset_cur for
   select value(p) from table(
dbms_sqltune.select_sqlpa_task('TASK_21137','Vergleich
l','REGRESSED')) p;
dbms_sqltune.load_sqlset(sqlset_name => 'SUB_STS1',
populate_cursor =>
sqlset_cur);
   close sqlset_cur;
end;
/
```


Dann können die entsprechenden (originalen) Ausführungspläne mit einem einzigen Aufruf als SQL Plan Baselines geladen werden.

```
declare
   my_plans pls_integer;
 begin
   my_plans := dbms_spm.load_plans_from_sqlset(sqlset_
name => 'SUB_STS1');
end;
/
```

4 Erfahrungen

An dieser Stelle sollen noch einige Erfahrungen mit DB Replay und SPA geschildert werden, die in mehreren ausgewählten Projekten gemacht wurden. Die Technologie wurde dabei für unterschiedlichste Zwecke wie Plattformwechsel, -Migration, Upgrade, Architekturwechsel (wie Partitionierung) oder Patch-Test verwendet. Die Zeit zum Erstellen eines Testsystems mit Backup, Upgrade, Migration und die Wahl der Methode zum Zurücksetzen müssen natürlich bei der Planung der Tests immer berücksichtigt werden. Nach einer anfänglichen Lernkurve, um sich mit dem Werkzeug vertraut zu machen, wurde in jedem Projekt schnell deutlich, wie gering der Testaufwand mit Real Application Testing ist und welche Vorteile ein solcher Test hat. Er kann Sicherheit und Vertrauen für Migrationen/Upgrade bringen, für ein besseres Verständnis der eigenen Applikationen sorgen oder auch die entscheidenden Argumente für einen Plattform- oder Architekturwechsel liefern.

Die wichtige Frage, ob und wie hoch der **Overhead beim Capture** ist, kann einfach beantwortet werden: Das Capture stellte in keinem Projekt eine Beeinflussung der Produktion dar! Diese Erfahrungen stammen aus den unterschiedlichsten Projekten – von DWH-Systemen mit Batch-Jobs und Reports bis zu OLTP-Systemen mit hoher Transaktionslast und hoher Concurrency. Auch große Standard-SAP-Systeme

waren unter den Testszenarios. Auch wenn die Empfehlung von Oracle lautet, die Datenbank vorab herunterzufahren, mussten die meisten **Tests im laufenden Betrieb** ohne jegliche Downtime der Produktion erfolgen. Das Einrichten von zusätzlichen Services im SAP-RAC-Umfeld oder das Setzen des Parameters PRE_11g_ENABLE_CAPTURE bei 10g-Datenbanken kann dabei auch im laufenden Betrieb erfolgen. In unseren Tests wurden **Zeiträume** von 30 Minuten bis zu mehreren Stunden (unser Maximum waren bisher 13 Stunden) aufgezeichnet. Die Applikationen zeichneten sich durch ganz unterschiedliche Lastprofile aus (8 Millionen User Calls in einer Stunde bis zu 600 Millionen User Calls in zwei Stunden). Die **Größenordnungen der Capture-Dateien** lagen dabei unabhängig von der Länge der Aufzeichnung zwischen 50 und 300 GB. Die Größe der Capture-Dateien steht dabei nicht im Zusammenhang mit dem Redo-Aufkommen, sondern errechnet sich in Abhängigkeit der Statistiken rund um die Statements. Dabei spielen folgende Faktoren eine Rolle: Anzahl der verschiedenen Statements, Verwendung von LOBs oder Bulk Binds um nur einige der Parameter zu nennen. Die Größenordnung des Capture kann am einfachsten über ein kurzes Test Capture erfolgen. Möchte man den AWR Report zur Berechnung heranziehen, kann man sich an der Metrik *bytes received via SQL*Net from* orientieren. Wichtige

Voraussetzungen für das Capture sind ein leeres logisches Directory und **ausreichend Platz** auf einem Storage mit **guter Schreibperformance**. Zusätzlich sollte die Dauer beziehungsweise die Menge des Capture begrenzt werden – ein Maximum von 4 Stunden Aufzeichnungszeit und eine Capture-Größe von unter 300 GB haben sich als sinnvoll erwiesen. Hintergrund ist einerseits die Handhabung beim Replay – ein kurzer Capture lässt sich immer leichter wiederholen und tunen als eine lange Aufzeichnung. Weitere Informationen dazu finden Sie auch im Kapitel 2.1.

In der PROCESS-Phase ist es wichtig, ausreichend Platz im SYSTEM Tablespace zur Verfügung zu stellen, da dieser wie wir in mehreren Projekten feststellten, stark anwachsen kann. Sorgen Sie am besten dafür, dass kein Engpass entstehen kann. Wie lange dauert nun ein Processing? Da das Processing die möglichen Replay-Optionen vorbereitet, müssen die Daten in den Capture-Dateien genau analysiert werden. Somit kann dieser Vorgang eine Stunde oder auch, wie in einem unserer Tests, mehrere Stunden dauern. **Die Dauer des Processing** hängt dabei von folgenden Faktoren ab:

- Anzahl und Größe der Capture-Dateien
- Abhängigkeiten im Workload
- Genutzte Hardware und Storage

Beim Processing können auch `ORA-15590` Fehlermeldungen auftreten, die Sie nicht weiter beunruhigen müssen. Sie weisen nur darauf hin, dass einige Capture-Dateien unvollständige Informationen enthalten. Beim Beenden des Capture sind Timeouts aufgetreten, somit konnten nicht alle aktiven Sessions ihre abschließenden Informationen in die Capture-Dateien schreiben.

Beim **Replay** müssen verschiedene Entscheidungen getroffen werden. So muss entschieden werden, ob eine nachträgliche Filterung stattfinden soll, wie viele WRCs pro Server gestartet werden sollen und mit welchen Optionen das Replay verwendet werden soll. Die ersten beiden Fragen lassen sich recht leicht beantworten. Die Entscheidung über die Art des Replay kann häufig nicht ohne verschiedene Testläufe beantwortet werden. Ideal ist ein kleiner Test vorab, um ein Gefühl für die richtige Wahl der Replay-Option zu bekommen. Ist dies nicht möglich, sollte man mehr Zeit für das Ausprobieren des Replays vorhalten. Eine Option sollte dabei immer die *synchronization*-Einstellung OBJECT_ID sein, allerdings spricht häufig auch nichts gegen die Einstellung FALSE, wenn wenige Divergenzen erzeugt werden.

Das Beispiel eines Tests auf einem **SAP-System** zeigt, wie leistungsfähig das DB Replay ist. In zwei Stunden wurden fast 600 000 000 User Calls aufgezeichnet. Obwohl kein Filtern

oder Herunterfahren der Produktionsumgebung möglich war, konnten die Tests ohne Probleme auf dem Testsystem ausgeführt werden.

Bei den Testverläufen wurde unter anderem auch eine Umstellung auf eine RAC-Architektur vorgenommen. Es wurden Services in der Produktionsumgebung genutzt, sodass im RAC-Test nur das Einrichten und Mappen der entsprechenden Services nötig waren (siehe Erläuterungen im Kapitel 2.4). Da es sich bei den Tests um Lasttests handelte, kamen die Aufzeichnungsarten *synchronization* FALSE oder OBJECT_ID in Betracht. Unterstützt wurde dieses Vorgehen noch durch das Abprüfen der Divergenzen nach dem ersten Test mit *synchronization*-Wert FALSE. Es zeigte sich, dass die **Divergenzen** unerheblich waren und sich in sehr kleinem Rahmen (< 1%) bewegten. Generell sollte man Zeit für die unterschiedlichen Prepare-Varianten einplanen. Spielt die gesamte Abspieldauer in der Bewertung eine große Rolle, sollten, wie in diesem Test, die Varianten OBJECT_ID und FALSE versucht werden. Das Ergebnis des Lasttests entsprach nach den Replays den Erwartungen.

Der **Compare Period Report** leistete bei der Interpretation gute Dienste. Die wesentlichen Ergebnisse zu den Top-5-Events, zu den Metriken *db time*, *cpu* und *i/o* sind übersichtlich, in aller Kürze und im Vergleich aufgelistet, sodass eine schnelle Einschätzung möglich ist. Weitere Details

konnten in den Capture und Replay AWR Reports (auch globalen) nachgeprüft werden.

Wie oben bereits erwähnt, kann **SPA** unabhängig von **DB Replay** genutzt werden und stellt eine gute Ergänzung und Vorbereitung für den DB-Replay-Testlauf dar. So kann zum Beispiel **während eines Replay-Laufs** automatisch ein SQL Tuning Set aufgezeichnet werden, um danach gleich zur Statement-Analyse überzugehen. SPA listet nicht nur die unterschiedlichen Metriken vor und nach der Veränderung auf, sondern gibt auch den Ausführungsplan und den Anteil des Statements am gesamten Workload an. Dabei werden die Statements nach verschiedenen **Kategorien** eingeteilt: *improved*, *regressed*, *unchanged*, *changed_plans*, *errors* usw. Auf diese Weise kann man sich beim Tuning sehr einfach den kritischen Statements mit veränderten Ausführungsplänen widmen. So konnte in einem unserer Projekte die Performance von kritischen Statements um 90 Prozent durch Einsatz von SQL Plan Baselines gesteigert werden. Zur Verifizierung wurde danach noch einmal ein Database-Replay-Lauf durchgeführt, um die Auswirkung auf die Gesamtperformance zu messen.

5 Fazit und Ausblick

Real Application Testing hat sich schon vielfach in verschiedensten Testkonstellationen bewährt. Steht ein geplanter Wechsel auf ein neues Datenbank-Release oder das Einspielen neuer Patchsets an, wird Real Application Testing häufig bei performance-kritischen Anwendungen eingesetzt. So können Probleme frühzeitig aufgedeckt und gelöst werden. Weitere Einsatzgebiete sind Tests beim Plattformwechsel oder reine Lasttests. Einmal ausprobiert, lernt man die Handhabung zu schätzen, die im Wesentlichen auf der Nutzung der Real-Application-Testing-Funktionen, der Erstellung des Testsystems und der Interpretation der Ergebnisse basiert. Kein kompliziertes Setup von Applikationsservern oder Installation von weiteren Werkzeugen ist erforderlich. Die folgende Übersicht zeigt die **Unterschiede von SPA und Database Replay** in der Zusammenfassung:

	SPA	**Database Replay**
Fokus	Einzel-SELECT-Analyse	Gesamte Datenbank-Workload-Analyse
Wiederholung	Ja	Ja, mit entsprechender Konfiguration
Installation	Keine; unter Umständen Patches notwendig Oracle Support Note 560977.1	Keine; unter Umständen Patches notwendig Oracle Support Note 560977.1
Nutzung vor 11g	In folgenden Fällen: 9.x => 10.x 9.x => 11g 10.x => 10.x 10.x => 11g 11g => 11g Oracle Support Note 560977.1	In folgenden Fällen: 9.2.0.8 => 11g 10.2.0.x => 11g Oracle Support Note 560977.1
Art des Workload	SELECT-Statements, DML über Package-Nutzung	Gesamter Datenbank-Workload wie z.B. SELECT, PL/SQL, DML, DDL, Transaktionen etc.
Voraussetzung	SQL Tuning Set	Keine
Ergebnis	Einzel-Ausführungspfad-Analyse	Capture und Replay Report, Compare Period Report, AWR und ASH Reports

Real Application Testing wird in jedem Datenbank-Release erweitert. Zusätzlich finden weitere Integrationen in andere Technologien über den Enterprise Manager statt. So steht zum Beispiel die Anwendung von Data Masking im Real-Application-Testing-Umfeld in Cloud Control zur Verfügung. Sensitive Daten, die in den Workload-Capture-Dateien und STS gefunden werden, werden innerhalb der Applikationsdaten maskiert, sodass das Testen im SQL Performance Analyzer oder mit DB Replay zu konsistenten und fehlerfreien Ergebnissen führt.

6 Weitere Informationen

- *Beispielskripte für Database Replay:*
https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_RESOURCE_BY_FNAME?P_TIPP_ID=221&P_FILE_NAME=rat_akt.zip

- *Beispielskripte für SQL Performance Analyzer:*
https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_RESOURCE_BY_FNAME?P_TIPP_ID=201&P_FILE_NAME=spa_akt.zip

- *Alle deutschsprachigen Tipps (Themen wie SQL Tuning Set, Plan Management etc.):*
http://tinyurl.com/dbainhalt

- *Application-Testing-Handbuch:*
http://docs.oracle.com/cd/E24628_01/server.121/e16540/toc.htm

- *Licensing Guide:*
http://docs.oracle.com/cd/E11882_01/license.112/e10594/options.htm#CJAGBGHH)

Herausgeber: Günther Stürner, Oracle Deutschland B.V.
Design: volkerstegmaier.de // Druck: Stober GmbH, Eggenstein

ORACLE

ORACLE®