Oracle Real Application Testing im Projekteinsatz – was hat sich bewährt, was nicht..

Konrad Häfeli
Trivadis AG
CH-3014 Bern

Schlüsselworte:

Oracle Real Application Testing, Capture, Replay, Sql Performance Analyzer, Sql Tuning Set RAT, SPA, STS

Einleitung

Migrationen sind immer mit Risiken verbunden, die Risikoanalyse und die Minimierung der Risiken ist ein Hauptteil der Migrationsvorbereitung und wiederspiegelt sich in den jeweiligen Migrationskonzepten. Je mehr Unbekannte im System sind, desto grösser sind die Risiken und demzufolge auch die Massnahmen die zur deren Minimierung getroffen werden müssen. Es gibt keine Veränderung ohne Problempotential, aber der Lebenszyklus von Hard- und Software sowie des Geschäftes allgemein, bedingt, dass wir uns mit Veränderungen auseinandersetzen.
„Never change a running system“ heisst die Devise, aber manchmal muss man halt doch Anpassungen vornehmen:

- Versionsupgrade
- Parameteränderungen
- DB-Architekturänderungen (Single Instance –> auf RAC)
- Plattformänderungen (Prozessor/Rechnerarchitekturen)

Der DBA muss sicherstellen, dass nach den Änderungen das System mindestens Gleich „gut“ läuft, wobei sich das „gut“ meistens auf die Durchlaufszeit von Applikationsabläufen bezieht. Testen ist aufwändig:

- Testsystem bereitstellen
- Reproduzierbare, produktive Last bereitstellen
- Testdaten passend für die Last bereitstellen
- Testiteration durchführen

Da kommt Real Application Testing (RAT) gerade richtig. Statements und Tätigkeiten können aufgezeichnet und auf diversen Umgebungen wiedergegeben und ausgewertet werden.

Übersicht

Real Application Testing ist ein Framework, das keine Lösungen anbietet, aber unterstützt solche zu erarbeiten. Die kostenpflichtige Option der Enterprise Edition des Oracle RDBMS wurde mit Oracle 11gR1 eingeführt und besteht aus zwei Komponenten:

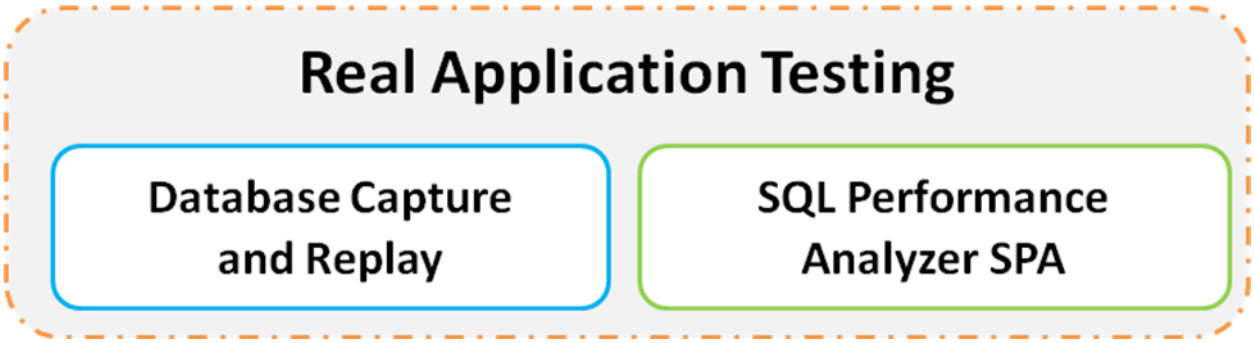

Abb. 1: Oracle Real Application Testing Komponenten

Während „Capture and Replay“ die Aufnahme und Wiedergabe der ganzen Datenbanklast macht, ist der „SQL Performance Analyzer (SPA)“ zuständig für die Analyse und Wiedergabe von SQL Statements.

Folgende Einsatzgebiete ergeben sich:

- SPA:
  - *Was*: SQL Antwortzeiten Analyse (nach Veränderungen)
  - *Wie*: Ausführung jedes Statements im Tuning Set, Vergleich des Ausführungsplanes und der runtime Statistiken
  - *Wann*: Unit Testing, Identifikation problematischer SQL

- DB replay
  - *Was*: Workload Durchsatz Analyse (nach Veränderungen)
  - *Wie*: Kompletter Workload replay mit “concurrency”, “synchronization” und “dependency”
  - *Wann*: Overall testing aller Sub-Systeme mit Produktionslast

Praxiserfahrung

Obwohl schon seit einiger Zeit verfügbar, ist die Option in Projekten eher spärlich eingesetzt worden. Wie immer kann man das mit den Kosten begründen, wobei die Reduktion der Veränderungsrisiken doch schon einiges wert sein sollte. Ausfallzeiten wegen unerwartetem Systemverhalten nach Änderungen sind richtig gerechnet in den meisten Fällen sehr teuer.

Ein Projektfeedback aus einem Sizingprojekt, war eher negativ, denn man hat in recht engem Zeitrahmen ein Vergleich einer bestehenden Platform mit einer Neuen machen wollen und war dabei überrascht vom grossen Datenvolumen der Workload (capture) Files und dem

Zeitaufwand diese vorzuverarbeiten (preprocessing). Jedenfalls trat beim Replay ein ORA-00600 Fehler auf. Der Fehler sollte in der Version 11.2 behoben sein, was sich aber nicht bestätigte, denn nun trat ein ORA-07445 Fehler auf. Dieses Erfahrungsfeedback zeigt auf, dass bei Capture/Reply nicht jede Applikationslast und Dauer ohne Probleme wieder abgespielt werden kann.

Eine weitere Unsicherheit in der Anwendung entstand, als in einem Migrations-Projekt, welches mit RAT abgesichert werden sollte. Die Datenstände nach dem Durchlauf nicht dieselben waren. Es stellte sich heraus, dass gewisse Statements die abgelaufen waren nicht in den Capture-Files enthalten waren. Das Verhalten konnte mit einer einfachen Sqlplus Session reproduziert werden, indem ein DML Statements mit commit abgesetzt wurde, welches aber bei einem Reply nicht ausgeführt wurde. Erst das Verlassen des Tools lies die Manipulation in das Capture-File schreiben. Auf einem produktiven System wollte man natürlich nicht die Applikation beenden, nur damit die letzten Statements sicher gecaptured werden konnten.
Die Anfrage bei Oracle ergab, dass das das erwartete Verhalten der Option ist, denn die Statements werden in Buffer zu 64KB gecached und die wurden nicht in das File heruntergeschrieben.
Bedingungen für das Schreiben der Buffer:

- Session logout
- Während finish_capture am Laufen ist (default Dauer 30 Sekunden)
- Buffer zu 75% voll
- Alle 5 Minuten

Für alle Bedingungen gilt aber die Session muss den Status „active“ haben und das haben Sessions nur wenn gerade ein Statements am Laufen ist. Reduktion des timeouts für den finish_capture resp. die Reduktion der Buffergrösse bringen da keine Verbesserung, im Gegenteil (mehr I/O Last auf das System).
Dieses Verhalten verunsicherte den Anwender, dass die Option schlussendlich doch nicht im Migrationskonzept integriert wurde.

Beispiel für ein Lasttest auf Oracle VM

In dem Projekt zieht der Kunde einen Wechsel von IBM P550 mit LPAR AIX 6.1 auf einen Oracle VM Server auf x86 mit Oracle Enterprise Linux 5.5 in Betracht. Dabei ist auch ein Setup für Oracle Live Motion[1] mit OCFS[2] auf dem Server implementiert. Dabei möchte man mit RAT feststellen ob es Leistungsunterschiede gibt zwischen den beiden Setups, ob I/O ein signifikanter Flaschenhals ist, bei welcher parallelen Last das System an seine Grenzen kommt und ob sich RAT überhaupt für solche Fragestellungen eignet.

[1] Mit Live Motion können virtuelle Maschinen von einem physischen Server auf einen Anderen verschoben werden
[2] OCFS ist das Clusterfilesystem von Oracle, welches einen shared Storage zwischen den Knoten bildet

Damit eine reproduzierbare Last auf das System gegeben werden konnte, wurde ein Tool mit dem Namen *swingbench*[3] eingesetzt, welches einen skalierbaren Benchmark auf Order/Entry Daten durchführt. Das Tool gibt es mit GUI (Java) und einem CLI mit dem Namen charbench. Das Setup ist Wizard unterstützt und installiert ein eigenes Datenbankschema:

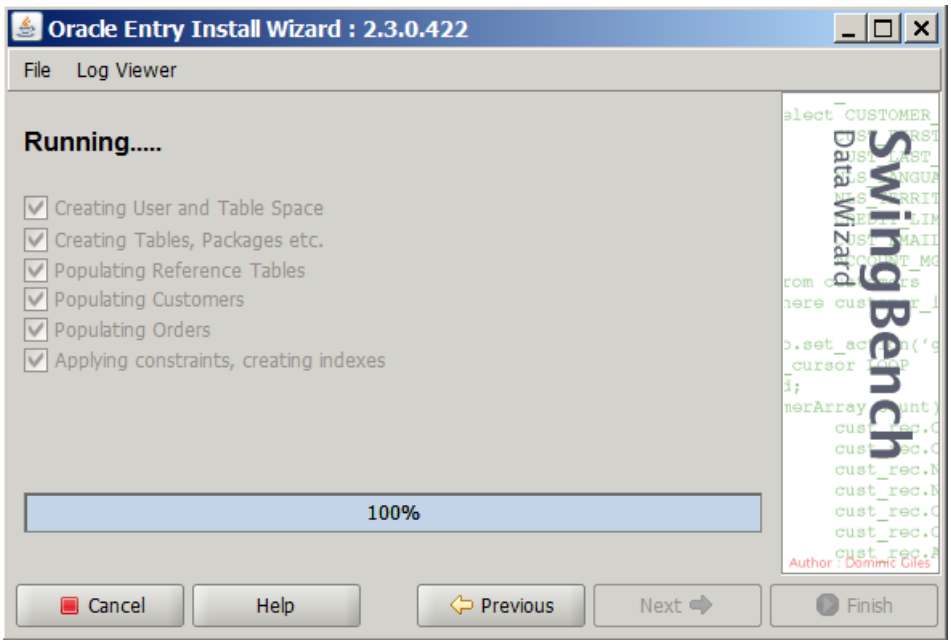

Abb. 2: Swingbench Data Wizard, Abschluss-Screen

Bevor nun die Last auf das Quell-System gegeben wird, muss das Capturing von RAT eingeschalten werden. Das Script:

$ORACLE_HOME/rdbms/admin/wrrenbl.sql

macht diese. Dabei empfiehlt Oracle den Restart der Datenbank, damit langlaufende Transaktionen ebenfalls korrekt gehandhabt werden können. Beim Capturen können Filter gesetzt werden die in unserem Fall nur auf das Testschema angesetzt sind. Mit den Prozeduren *start_capture* und *finish_capture* im Package *dbms_workload_capture* kann die Last aufgezeichnet werden. Die entstandenen Files müssen nun auf das Zielsystem übertragen werden, wo die Datenbank für den Replay vorbereitet werden muss. Der import des Testschemas zum Zeitpunkt vor dem Swingbench start, kann für mehrmaligen Durchlauf des Replays auch mit einem garantierten Restorepoint versehen werden, damit mittels flashback database sehr einfach die DB zurückgestellt werden kann.
Die Prozedur *dbms_workload_replay.process_capture* bereitet die Capturefiles für den Replay vor und legt ein eigenes Unterverzeichnis dafür an. Ab der Version 11.2.0.2 gibt es einen Capture-Checker, der die Files analysiert und Empfehlungen abgibt für den Replaydurchlauf. Dies ist sehr hilfreich, sind doch vor allem bei grosser PL/SQL Last die default Replayparameter nicht in jedem Fall optimal.

[3] Swingbench kann man downloaden unter: http://www.dominicgiles.com/swingbench.html

Das Utility „*wrc mode=calibrate*“ lässt eine sogenannte Kalibrierung machen, welche angibt mit wie vielen Clientprozessen die Last gefahren werden soll. wrc mit dem mode-Parameter *replay* startet die Client Prozesse und die Prozedur *dbms_workload_replay.start_replay* startet den Replay. Im Enterprise Manager kann der Fortschritt überwacht werden:

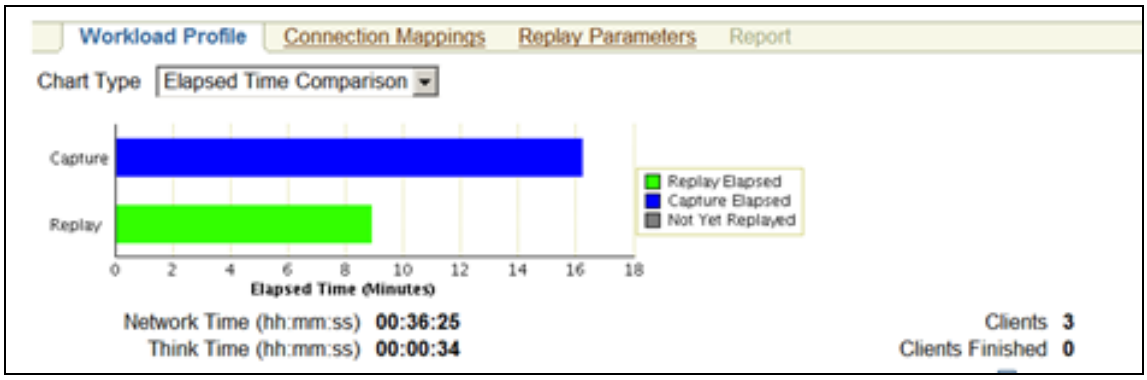

Abb. 3: Enterprise Manager Workload replay

Nach dem Durchlauf können im Reiter *Report* verschiedene HTML Reports generiert werden (AWR-, ASH-, Replay Report) die eine detaillierte Analyse des Durchlaufs ermöglichen. Im Testprojekt kam dabei klar zum Vorschein, dass im Vergleich der beiden Platformen das I/O System von Oracle VM um einiges langsamer war als IBM LPAR, was die schnelleren CPUs nicht kompensieren konnten.

SQL Performance Analyzer (SPA)

Mit dem SPA können Tuning Sets ab Oracle 9.2.0.8 (Produktion) erstellt und auf einer 11g Datenbank (RAT-DB) ausgewertet werden. In einen konkreten Projekt konnte sogar der Trial lauf via DB-Link auf einer 10.2.0.4 Datenbank (Test) durchgeführt werden, sodass RAT für die Migration von 9.2 auf 10.2 eingesetzt werden konnte.
Folgender Ablauf wurde gewählt:

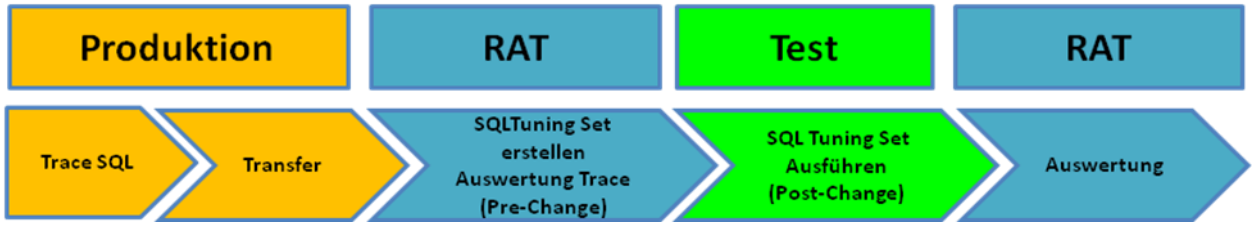

Abb. 4: RAT mit SQL Performance Analyzer auf SQL Tuning Sets

Durch die Komplexität der Applikation wurde in die Automatisierung der Abläufe einiges investiert, sodass für das Migrationsprojekt folgende Strategie umgesetzt wurde:

- Trace nach Applikation pro bestimmte Tageszeiten, Ablage in SQL Tuning Sets
- Trace von Top Sessions und relevanten Statements, Ablage in SQL Tuning Sets
- Bündelung nach Applikation, Ablage in SQL Tuning Sets

- Testdurchlauf (Trial)
- Analyse der schlechter laufenden Statements
- Verwaltung der Statements
- Rerun getunter Statements
- Regressionstests auf alle Statements bei globalen Parameteränderungen

Am Schluss wurden 450 Statements von 28‘000 analysiert und Massnahmen getroffen. Dabei wurden Init.ora Parameter im Bereich Optimizer angepasst, Indexstrukturen verbessert, Erkenntnisse an die Entwickler weitergegeben und die Statistiksammeljobs definiert. Nach dem Einsatz von RAT SQL Performance Analyzer war die Datenbankinfrastruktur bereit für den Upgrade.

Was hat sich bewährt…

Durch die Komplexität der Aufgabe empfiehlt es sich für den Einsatz von RAT einen eigenen Proof of Concept (PoC) zu machen, damit eine Kosten/Nutzen Analyse für die Lizenzkosten und den Einarbeitungsaufwand gemacht werden kann. Nebst Capture/Replay sollte unbedingt auch SPA im Einsatzkonzept sein, denn dieser bringt in vielen Fällen die Problempunkte schneller zu Tage. Durch das Vorhandensein von Softwareproblemen im Bereich RAT sollte unbedingt genügend Zeit eingeplant werden. Ebenfalls ist das Kennen der Applikationsart und deren Spezifikas wichtig, damit auch Kompromisse beim Setting der Replay Optionen gemacht werden können. Der Capture-Checker bringt da sehr hilfreiche Unterstützung. Die Automatisierung der Abläufe ist wegen der Testiterationen und der Notwendigkeitn von reproduzierbaren Ergebnissen wichtig. Tuning Kenntnisse sind unabdingbar, denn erst durch die Auswertung der Reports können sinnvolle Massnahmen zur Verbesserung getroffen werden.

Was hat sich NICHT bewährt…

Der Glaube an das perfekte Tool wird (auch) bei RAT enttäuscht. Es ist ein Framework mit dem man Arbeiten muss nur durchklicken bringt nicht die erhofften Erkenntnisse. Zeitdruck beim erstmaligen Einsatz bringt jedes RTA Projekt zum Scheitern. Grosse Lasten mit hoher Anzahl paralleler Sessions (ab ungefähr 200) und stark PL/SQL-lastig bringen beim Preprocessing vielfach Fehler zum Vorschein die nur durch Auslassen der problematischen Capturefiles umgangen werden können. Zudem sind lange Capturezeiten meist nicht sinnvoll, denn durch mehrmaliges Replay sind dann die Iterationszyklen auch sehr lange.

Fazit

Der Aufwand beim erstmaligen Einsatz von RAT ist nicht zu unterschätzen, auch muss das Einsatzkonzept gut auf die Gegebenheiten abgestimmt sein. RAT adressiert aber genau die Unsicherheiten bei Veränderungen der Datenbank-Infrastruktur.
Eine RAT Evaluation gehört in jedes Upgrade/Migrations Konzept!

Kontaktadresse:

Konrad Häfeli
Trvadis AG
Papiermühlestrasse 73
CH-3014 Bern

Telefon:	+41(0)31-928 09 60
Fax:	+41(0)31-928 09 64
E-Mail	konrad.haefeli@trivadis.com
Internet:	www.trivadis.com