RAT-less Replay

Robert Barić
ITGAIN Consulting
Hannover

Tags
RAT, Trace, Unified Auditing, Replay, Workload, Benchmark

Introduction

Many times a new database system is measured by synthetic benchmarks. This kind of rating is limited as application requirements are often only partially simulated.

Application level benchmarks created by "capture and reply" rebuild requirements much better.

The workload of a database can be replayed to measure the new system.

Oracle provides such a functionality called Real Application Testing (RAT) for an additional cost.

Would it be possible to recreate such a feature?

Use Cases

If we talk about recording and replaying a real database workload we may ask about the reasons and benefits. Synthetic benchmarks have their place as they are able to messure a system with specific characteristics and make results compareable. So what are the benefits and use cases for real workload? I would like to explain these with a real world case.

Real World Case

Here is a real world example from a popular german car sharing company.

In terms of a consolidation project databases should be moved into the cloud.

Everything was set up, performance tests were made and a bunch of benchmarks were released on the database in the cloud. The results were great and the project was pushed forward for the movement of the production database.

However, the synthetic benchmarks did not cover the specific workload characteristics and the real workload lead to a business breakdown. True story as it may happen anywhere else.

What do we learn? The benefit about your real workload benchnmark is that it can answer questions about your workload. This can be crucial to your business.

The disadvantage about your real workload benchmark is these results are limited for comparison with other workloads. Maybe even less than synthetic benchmarks if the characteristics of your workload are not known. Three major test cases for real workloads have been identified: functional tests, migration cost and time tests and performance tests.

Funtional Tests Case

Your real workload could be used to answer the question if your code still works after transfering it to another system.

Especially when no seperate tests like unit tests are available then your workload could be a tests on its own.

However you should mind that your workload most likely is not covering your complete code and settings. Thus the functional test case should be considered incomplete for your application.

It may only be considered complete for this workload.

Migration Costs and Time Case

Especially when moving to a different database vendor using a real workload can help you to identify the amount and kind of code that needs to be adjusted and changed.

In business terms someone can get an idea about the time and the cost for such a migration.

Performance Tests Case

Performance tests are useable when ever a software or hardware change may question the performance stability. Using your real workload gives you assureance about the performance on a different system.

Real Application Testing (RAT)

Oracle itself defines RAT as a set of features comprised of "SPA Quick Check, SQL Performance Analyzer, Database Replay, Concurrent Database Replay and the Database Consolidation Workbench." [1]

Within the scope of this document when referring RAT it is in fact the Database Replay component referred to.

Functional Scope

Oracle Documentation [2] defines a (most likely incomplete) list features, abilities and limits to Real Application Testing and Database Replay. Considering to build a replica it is rational to verify to which extends these features may be copied. The list is shown in Table 1.

Reasons for a RAT-less replay

Oracle has built RAT as a comprehensive and accurate set of tools to replay a workload.

However, there may be some reasons searching for a RAT-less replay because of

- EE Option with Extra Cost
- Not licensable for Standard Edition
- Limited to Oracle Database
- Powerfull and feature rich, but for simple and small workloads rather complex

RAT Supported		NOT RAT Supported	
Complete SQL Code (DML,DDL, PL/SQL)	✓	Direct Path Loads, Import/Export	✗
Local Transactions	✓	Flashback (DB, Query)	✗
Sessionchange	✓	Distributed Connections	✗
Connects/Disconnects	✓	Shared Server	✗
Limited PL/SQL RPCs	✓	Streams (non PL/SQL based AQ)	✗
Complete Lob Functionality	✓	OCI Object Navigation / REF-Binds	✗

Abb. 1 RAT Features

Recoding Options with Oracle

To rebuild a RAT-less replay we need rebuild two components: the record component and the replay component.

Due to licensing restrictions, we cannot simply use the recording component of RAT. Therefor we must to a look what other options are open to use.

Considering possible data sources, the following list has been compiled:

- V$ Views / Diagnostic Views
  AWR, v$session, v$sql_text, V_$SQL_BIND_CAPTURE, etc..
- Standard Auditing
- FGA (Fine Grained Auditing)
- Unified Auditing
- Trace (10046 TRACE, SQL_TRACE, DBMS_MONITOR, ..)
- DB Triggers and manual logging

- JDBC Interface Logging
- Logminer   Reference →The LogMiner Utility (Doc ID 62508.1)
  Oracle 10g New Features Of LogMiner (Doc ID 249001.1)
- Oracle Streams

Evaluation

The aim for a suitable record option is a data source which

1. is useable with standard and enterprise version
2. offers an unchanged and complete user SQL Code
3. and finally, is “easy to use” or practical to implement

If we look at the first restriction, we must sort out FGA and Oracle Streams as these two are only useable with the Enterprise Edition. This also does apply for AWR and other diagnostic or tuning option views and procedures.

Applying the second restriction Logminer is sorted out as the information inside the logs does not necessary hold the original SQL. Querying session and db information like v$session and v$sql_text may not be complete as it only shows a time slice of actions.

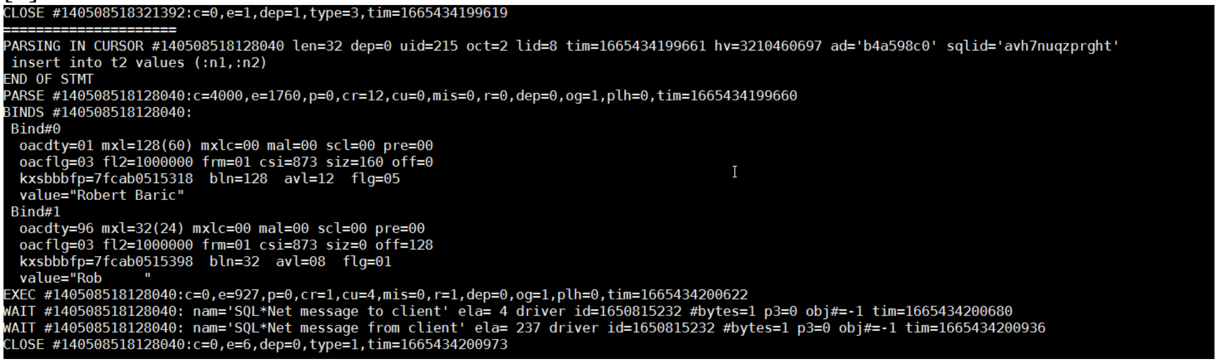
Restriction three should sort out sources that may be complex or hard to implement. Logging via JDBC Interfaces may be a good solution for a single application or an application server. Having hundreds of clients may be an impossible task.

Even so creating a manual logging through triggers could be a solution which could be automated, it needs changes to be done to the database. This “code” change may be restricted on production. Possible side effects in complex environments makes this option indeed rather “complex”.

This leaves three options: traces, standard audits and unified auditing

Trace Files

A very comprehensive way of recording activity and metadata information are traces. This feature is available for standard and the enterprise edition. Traces may be enabled session or database wide and may also be limited to services, modules and actions through the dbms_monitor package.
Trace files are text files. Usually each session creates one trace file. An example of such a trace is shown in *Abb. 2 Example of a dbms_monitor trace*. A detailed interpretation for trace files is described in [3].



```
CLOSE #140508518321392:c=0,e=1,dep=1,type=3,tim=1665434199619
=====================
PARSING IN CURSOR #140508518128040 len=32 dep=0 uid=215 oct=2 lid=8 tim=1665434199661 hv=3210460697 ad='b4a598c0' sqlid='avh7nuqzprght'
 insert into t2 values (:n1,:n2)
END OF STMT
PARSE #140508518128040:c=4000,e=1760,p=0,cr=12,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=1665434199660
BINDS #140508518128040:
 Bind#0
  oacdty=01 mxl=128(60) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=01 csi=873 siz=160 off=0
  kxsbbbfp=7fcab0515318  bln=128  avl=12  flg=05
  value="Robert Baric"
 Bind#1
  oacdty=96 mxl=32(24) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=01 csi=873 siz=0 off=128
  kxsbbbfp=7fcab0515398  bln=32  avl=08  flg=01
  value="Rob     "
EXEC #140508518128040:c=0,e=927,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=1665434200622
WAIT #140508518128040: nam='SQL*Net message to client' ela= 4 driver id=1650815232 #bytes=1 p3=0 obj#=-1 tim=1665434200680
WAIT #140508518128040: nam='SQL*Net message from client' ela= 237 driver id=1650815232 #bytes=1 p3=0 obj#=-1 tim=1665434200936
CLOSE #140508518128040:c=0,e=6,dep=0,type=1,tim=1665434200973
```

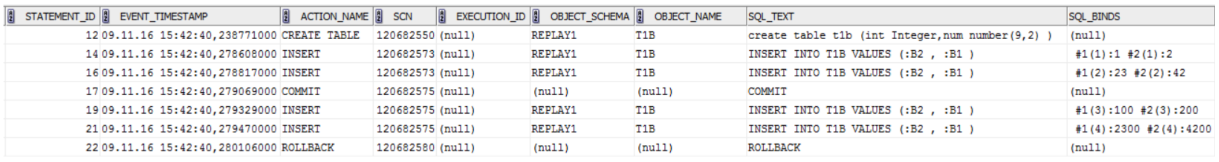
Abb. 2 Example of a dbms_monitor trace

Unified Auditing and Standard Auditing

Unified Auditing as a new auditing facility within Oracle 12c. The main benefits are the consolidation of auditing information into one place (one table) and Queued Write Mode. In Queued Write Mode audit records are batched in sql queue. Because of this a great performance boost is possible compared to standard auditing.

Additional benefits are the additional filtering options which allow a much finer grained auditing and the auditing of commits, savepoints and rollbacks. This enables transactions to be implemented within RAT-less replay. Standard auditing is missing these features. However, it standard auditing could be useable in this restricted fashion for versions below 12c. An overview about unified auditing is described in [4] [5].

Abb. 3 shows selected information collected by unified auditing.



STATEMENT_ID	EVENT_TIMESTAMP	ACTION_NAME	SCN	EXECUTION_ID	OBJECT_SCHEMA	OBJECT_NAME	SQL_TEXT	SQL_BINDS
12	09.11.16 15:42:40,238771000	CREATE TABLE	120682550	(null)	REPLAY1	T1B	create table t1b (int Integer,num number(9,2))	(null)
14	09.11.16 15:42:40,278608000	INSERT	120682573	(null)	REPLAY1	T1B	INSERT INTO T1B VALUES (:B2 , :B1)	#1(1):1 #2(1):2
16	09.11.16 15:42:40,278817000	INSERT	120682573	(null)	REPLAY1	T1B	INSERT INTO T1B VALUES (:B2 , :B1)	#1(2):23 #2(2):42
17	09.11.16 15:42:40,279069000	COMMIT	120682575	(null)	(null)	(null)	COMMIT	(null)
19	09.11.16 15:42:40,279329000	INSERT	120682575	(null)	REPLAY1	T1B	INSERT INTO T1B VALUES (:B2 , :B1)	#1(3):100 #2(3):200
21	09.11.16 15:42:40,279470000	INSERT	120682575	(null)	REPLAY1	T1B	INSERT INTO T1B VALUES (:B2 , :B1)	#1(4):2300 #2(4):4200
22	09.11.16 15:42:40,280106000	ROLLBACK	120682580	(null)	(null)	(null)	ROLLBACK	(null)

Abb. 3 Selected columns Unified Auditing

Recording Obstacles

While in theory traces and auditing can record a session, there are several restrictions in practice to encounter. Only some examples are listed here:

Unified auditing gathers most information needed for replay, including Sql text and binds. However, it misses the values types from the sql binds which trace files can record.

In comparison to trace files there are no additional or optional measurement values available like waits for cpu, I/O and network.

Unified auditing catches also CLOB bind variables. However, the maximum size is (currently) limited to 4000 bytes. Everything longer is truncated. Trace files (currently) do not collect certain types like lobs and user definied types.

The database generates additional code which is also recorded by traces and auditing. This may be trigger code or other internal actions during your workload. These extra code usually needs to be cleared by rules, especially when dealing with audits.

Especially with traces files “bugs” may restrain a “complete” workload recording.

Replaying Obstacles

Someone could tend to believe there should be no obstacles after successful recording. However, there may be limitations to face even with a “perfect” recording in place. Just some popular ones should be named:

A prominent problem may arise with individual values like system time being used within your workload. Even if the system time is being reset to the start values the replay will have a certain drift and will therefor differ.

The similar problem may arise with sequences or counters as replaying concurrent session may lead to a different value being used than before.

Usually replaying takes place with one user having all privileges needed to issue the workload recorded from different users. This simplification may lead to a different result. An example are logon triggers for certain users.

Replaying explained with SQLReplayer

The Itgain company has a own replay solution SQLRepalyer working with different databases which can be used to illustrate a RAT-less replay and some feature ideas for replaying.

The replay steps are illustrated with *Abb. 4 Replaying Operation Example using SQLReplayer*.

The obvious preparation step is to “clone” either the complete source database or the used structures and contents just before the workload is applied and recorded.

The recorded workload by trace or by auditing can be transferred to workload database for further analysis, reconstruction or combination with other workloads. A simplified approach is to use the recorded workload with export files directly with a replayer.

A replayer will have to process these files for proper replay. For examples using trace files only necessary data is compiled into new files which then can be replayed.

Especially for performance evaluation the target database may be monitored. This may be done with any suitable monitoring tool like Oracle own solutions statspack or AWR or with solutions from other vendors. The picture shows the companies own monitoring solution speedgain.

Feature ideas

Using a replay usually enables to tweak the recorded workload for optimized testing like

- Session and SQL Duplication
- Stress Mode by eliminating sql pauses
- Filtering of some kind

Other vendors may offer additional features like

- Out-off-the-box replay without software Installation
- Runable on every system via java / jdbc
- Easy to handle and transparent
- Different Input Types (csv, json, (trace), DB) which make changes very simple
- Support different db systems

Today you may choose to build a solution on your own or to buy one.

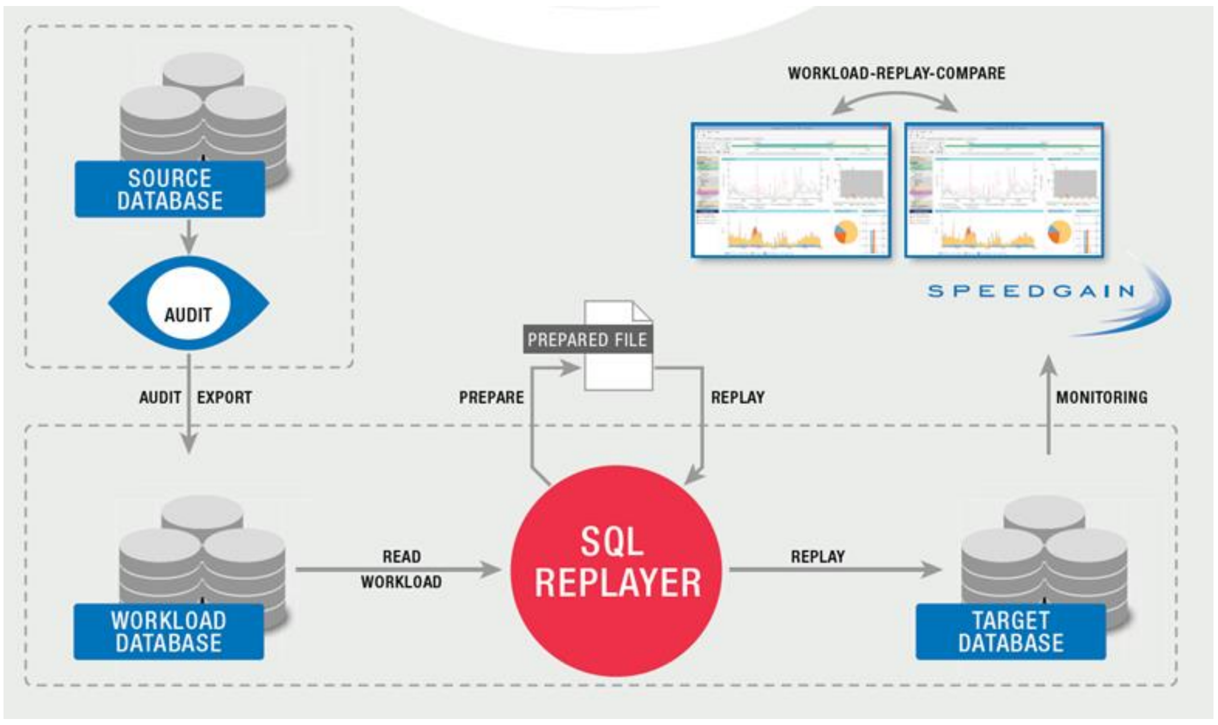


Abb. 4 Replaying Operation Example using SQLReplayer

[1] *Oracle Real Application Testing*. .
[2] "Database Replay - Workload Caputure Restrictions." [Online]. Available: https://docs.oracle.com/cd/B19306_01/server.102/b14211/wcr.htm#CJGHFCBE.
[3] Oracle, "Interpreting Raw SQL_TRACE output (Doc ID 39817.1)," *Oracle Support*.
[4] "Oracle 12c Unified Auditing - Part 1." [Online]. Available: http://www.carajandb.com/en/blogs/blog-jahrends-en/202-unified-auditing-en-1.
[5] Johannes Ahrends, "Oracle 12c Unified Auditing - Teil 1," 2015. [Online]. Available: http://www.carajandb.com/de/blogs/blog-jahrends/200-unified-auditing-de-1.

Contact:

Robert Barić
ITGAIN Consulting mbH
D-30173 Hannover

Telefon:	+49 (0)) 511-51513 700
Fax:	+49 (0) 511-51513 800
E-Mail	robert.baric@itgain.de
Internet:	www.itgain.de