Database Replay

An Oracle White Paper
November 2007

ORACLE

NOTE:

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Database Replay

Agile businesses want to be able to quickly adopt new technologies, whether it's operating systems, servers, or software, to help them stay ahead of the competition.

However, change often introduces a period of instability into mission-critical IT systems. Real Application Testing—with Oracle Database 11*g* Enterprise Edition—allows businesses to quickly adopt new technologies while eliminating the risks associated with change. Real Application Testing combines a workload capture and replay feature with an SQL performance analyzer to help you test changes against real-life workloads, then helps you fine-tune them before putting them into production.

INTRODUCTION

System changes such as hardware/software upgrades, patch application, etc. are essential for businesses to maintain a competitive edge for compliance/security purposes. Businesses spend significant time and effort evaluating and testing system changes in test environments before introducing them in production systems. However, despite such testing, using various scripts and simulation tools, many issues often go undetected until production deployment and negatively impact system performance and availability. The main reason for low success rate of testing is the inability of existing tools to test using real production workloads.

Database Replay, one of the solutions within the Oracle Real Application Testing option enables realistic testing of system changes by recreating the production workload on the test system while maintaining the unique characteristics of the workload. It does so by capturing the live workload on the production system and replaying it on the test system with the exact timing, concurrency and transactional properties of the original workload. This cost-effective solution does not require any test scripts to be generated and maintained. As a result, the testing cycle for complex applications that previously took several months of effort can now be done in a matter of days.

DATABASE REPLAY WORKFLOW

The workflow for Database Replay consists of four main steps:

1. **Workload capture**
   When workload capture is enabled on the production system, all external client requests directed to the Oracle Database are tracked and stored in binary files, called capture files, on the file system. The user specifies the location of the capture files as well as the duration of capture. These files contain all relevant information needed for replay, such as SQL text, bind values, wall clock time, system change number etc. The capture is database wide and RAC aware, and hence it needs to be initiated on one instance only. You can view the progress of capture and the system activity via Enterprise Manager. Oracle recommends having a backup and restore strategy in place prior to the workload capture so that a test system can be recreated with the same data as of the start of capture. To get maximum benefit from this feature users should enable capture during

their peak workload since this is the period that would be most interesting for testing purposes.

2. **Workload Processing**
   Once captured, the workload files have to be processed before replay. This creates the necessary metadata needed for replaying the workload. It is important that the processing be done once on the same database version as the Replay system. Once processed the captured workload can be replayed repeatedly on the same version. It is generally recommended that this step be performed on the test system where the workload will be replayed

3. **Workload Replay**
   After the workload has been processed it is ready for Replay. The test system should have the change applied and the database should reflect the same application data state as of the start of capture. A special client called the Replay Client is provided to issue the workload to the database server with the same timing and concurrency characteristics as in the capture system. A calibration tool is provided to help determine the number of replay clients since more than one may be needed to drive the workload. The workload files have to be made available to the database server as well as to all the Replay clients.

4. **Analysis and Reporting**
   Extensive reports are provided to enable detailed analysis of the capture and replay. Any errors or divergence in data encountered during replay are reported. It is highly recommended to have an Application level validation script to assess the state of the application data after replay. Reports using the AWR data are also available for detailed performance analysis. AWR data is automatically exported into the workload directory after the replay allowing comparison between replays.

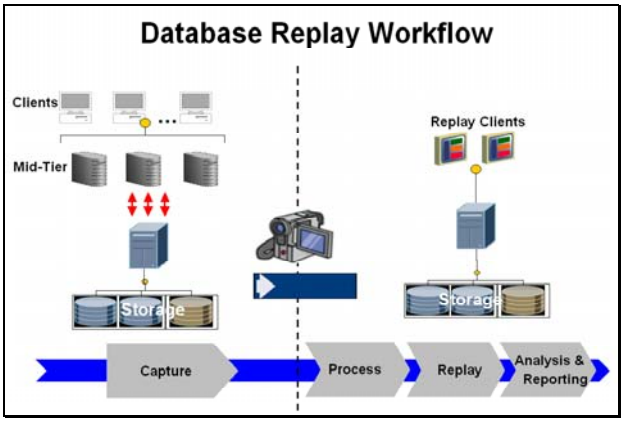


Figure 1: Database Replay Workflow

Oracle's Enterprise Manager provides complete support for the all the steps in the above workflow. Shown below is the page within Enterprise Manager that launches any of the above steps.

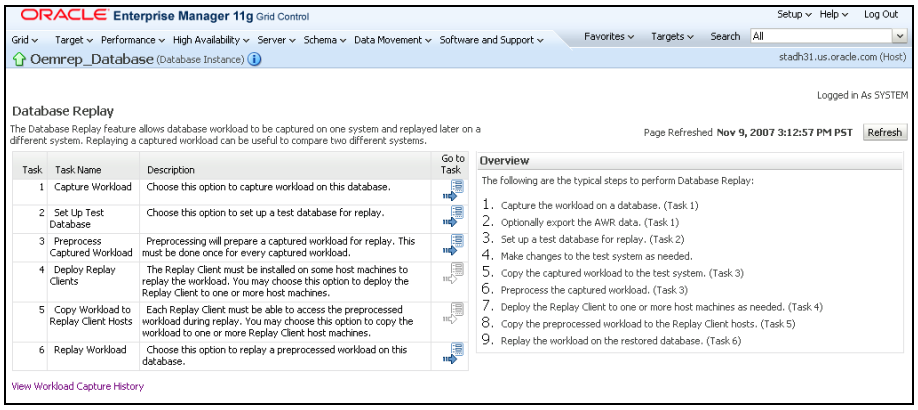


Figure 2: Enterprise Manager Database Replay Launch Page

COMMON USAGE SCENARIOS

Database Replay can be used to test any system change at or below the database level. This includes configuration and hardware changes. Listed below are some of the common usage scenarios for Database Replay

- Database upgrade including patch deployments.
- Configuration changes such as conversion from single instance to RAC, usage of ASM.
- Experimenting with load balancing techniques and RAC services.
- Changing database configuration parameters eg: sizing parameters
- Operating system and hardware platform migrations.
- Storage, network, interconnect changes

The rest of the paper explains the various steps in more detail along with the best practices that Oracle recommends.

CAPTURE

All external client requests directed to the Oracle database are tracked and stored in binary files in the directory specified as input. Background and system activity are not captured. The workload capture infrastructure supports both local and shared file system. The size of captured file also depends on the workload and is proportional to the data that needs to be captured. Note that the format of the captured data is platform independent; this allows you to test migration between hardware platforms.

The workload directory over time will contain the workload files, metadata necessary for replay, and AWR snapshots of capture and all subsequent replays (to allow for comparison). It is recommended that this workload directory be moved to a separate location after capture and every replay. Every replay will be adding more contents to the directory thus it is important to keep using the same set of files to preserve replay history and comparison.

We currently support workload capture on Oracle Database versions 10.2.0.4 and higher. Please refer to the section on restrictions for a list of unsupported features.

Setting up Capture

Before starting the capture, a directory object needs to be created. Ensure that the underlying directory is empty and has ample storage. If the system runs out of space the capture will automatically stop without affecting the user workload. Workload capture can be run on demand or scheduled to run at a certain time, and the duration should cover interesting periods of the application such as peak times and maintenance windows AWR snapshots will be taken automatically during the capture for the purpose of analysis and reporting. The user is expected to export the snapshots to the directory containing the workload files at the end of capture. Enterprise Manager provides full support for these operations.

Specifying Filters

Capture infrastructure provides the ability to include or exclude specific workload based on workload attributes: user, program, module, action, service and session ID. Since many applications might be sharing the same database, filters can be used to test only some specific applications that need to be upgraded instead of the whole system. Filter can also be useful to filter out administrative workload that the DBA might want to do during the time of capture.

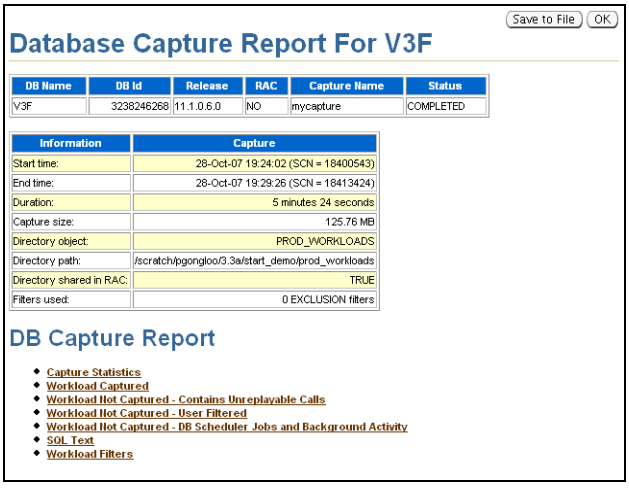


Figure 3: Capture Report

Capture Monitoring and Report

Enterprise Manager provides extensive system monitoring capabilities. You can graphically see the amount of workload being captured in relation to the entire database workload. You can also get a report of the state of the capture at any time. AWR snapshots are taken automatically during the capture, and the performance data within AWR is used to provide a more detailed capture report. An AWR export is recommended for further analysis and comparison with replay. In addition to performance data the report characterizes the workload that has been captured, the amount that has not been captured as well as the amount that will be unreplayable.

Overhead

The workload capture process has been highly optimized to make sure it incurs negligible overhead even on a busy system. The overhead primarily depends on the workload and is proportional to the data that needs to be captured. For example, if the workload consists mainly of calls that are long-running queries, the overhead is going to be minimal, since the data captured per time unit is small: Only the text of the complex query once per session for each call that executes this query. On the contrary, if the workload is insert-intensive, the overhead will be higher. For example, we have observed a throughput degradation of 4.5% on a system running a highly concurrent TPCC workload.

TEST SYSTEM SETUP AND WORKLOAD PROCESSING

Once the workload has been captured, the next step is to setup the test system and apply the changes that the user wants to test, such as a database upgrade or a system configuration change. The test system should have database restored to the point in time before the capture started to reflect the same application data state or else the database might encounter data divergence during the replay. Duplication or restoration of the application data can be done using RMAN, snapshot standby, import/export tools, etc. The start SCN of the capture period can be retrieved from the capture Report. (see figure 4). For instance, if database upgrade is being tested, you would restore the database on the test system to a particular SCN or point in time using RMAN and then perform the upgrade.

The capture files need to be transported and consolidated into one directory before being processed on the test system. Processing the workload capture results in the creation of the necessary metadata needed for the replaying the workload. Processing needs to be done once on the same database version as the Replay system. Once processed, the captured workload can be replayed repeatedly on the same version.

REPLAY

Once captured workload has been processed, it is now ready for replay on the test system. It is assumed that the test system has been appropriately setup. A special client called the "replay client" replays the workload from the processed files. The replay client submits calls to the database with the exact same timing and concurrency as in the capture system and puts the exact same load on the system as seen in the production environment. This allows the identification of any instability caused by the change and their subsequent remediation in test environment before the introduction of the change in production.

It is important to notice that neither the middle-tier nor the application layer is required for replay. Only the database and the replay clients are necessary to faithfully reproduce the production workload and test the system change.

Replay Client

The replay client is the multi-threaded OCI client that replays the capture files. Each replay client thread reads *one* capture file, and interprets the recorded calls as a sequence of OCI calls that it submits to the database.

"Three new features in Oracle Database 11g that really jump out are Automatic SQL Tuning, Partition Advisors, and Real Application Testing. Those three things alone are reason enough to upgrade."

—Arthur Fleiss,
IT Architect, Colgate-Palmolive

The replay client uses the capture processing output to know when to start replaying a capture file. Once the replay of a file has been started, the replay client thread issues the calls based on the recorded timing: it honors the think time that was recorded between two consecutive calls. Practically, this means that there is no coordination between the different replay threads on the client side. All the synchronization (if enabled) is done on the server side, which yields a very scalable architecture where we can have as many replay clients as needed without any overhead.

Although the replay client is a multi-threaded application that can replay multiple sessions at the same time there are practical limitations due to the number of threads per process, open file descriptors, etc.. It is recommended that a replay client should not replay more than 50 concurrent sessions by itself. If the recorded workload has more concurrent sessions, multiple replay clients should be started in order to have a faithful replay. We provide a calibrate utility that estimates these numbers (see section on Setting up Replay Clients for more information)

The replay client binary, named `wrc`, is part of the Oracle Client and Oracle Instant Client.

Replay Technology

Synchronization

One of the key breakthroughs is the ability to replay the workload while maintaining the original concurrency and transactional characteristics. We refer to this a "synchornization". If synchronization is turned on during replay (default behavior), the server honors commit ordering to preserve transactional characteristics.

We classify calls during captures into commit operations and non-commit operations. This distinction is based on the fact that a commit operation changes the database state and makes the new state visible to all subsequent operations. Since the result of any call depends on the database state, the replay has to ensure that each call follows the appropriate commit operation so as to satisfy result consistency and ultimately database end state consistency. To enforce that requirement, replayed calls that arrived too "early" at the server to be replayed wait on the following wait events until the appropriate commit operation is replayed:

- *WCR: replay lock order*. A session will wait on this event during replay if it sees some lock contention during capture with a transaction whose commit operation has not been replayed yet.
- *WCR: replay clock*. This is an idle event. A session will wait on this event during replay if the database state that it needs to see depends on a commit operation that has not been replayed yet.

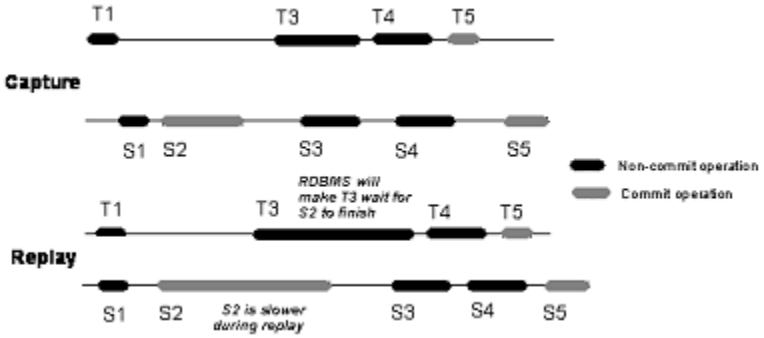


Figure 5: Replay Synchronization Example

A simple synchronization example is shown in figure 6. During capture, the non-commit operation T3 was executed after the commit operation S2. Therefore, changes made to the database state by S2 were visible to T3. Now, we can imagine that, during replay, S2 is slower and takes more time to finish than during capture. Even though T3 has been issued by the replay client at the same time it was issued during capture, it shouldn't be replayed by the database until S2 is done, or it will miss S2's changes. Thus, it will be put on the *WCR: Replay Clock* wait event until S2 has finished.

If synchronization is disabled, we do not enforce any logical dependencies between calls, in which case, the calls are replayed based only on the captured timing.

Data Remapping

Even if we enforce logical data dependencies, there are cases where a replayed call will not do the same work it did during capture. This happens when a call contains system dependent data that are invalid in the replay system. Examples of such data

in the Oracle Database include row identifiers (*rowids*), large object locators (*lob locators*) and result references (*ref cursors*). If they are part of the bind values, the replay clients re-map them to runtime correct values.

Runtime re-mapping is illustrated by the following example (fig 7). The rowid that is returned to the client as part of the select list is captured (1). Then the same rowid is captured as an bind (2). During replay the captured rowid associated with the select statement (3) makes the replay client expect the replay time value of the rowid. At this point the association is made between the captured rowid and the one seen during replay. When the update is replayed (4) the replay time rowid is used. This will make the update succeed and update the employee row.

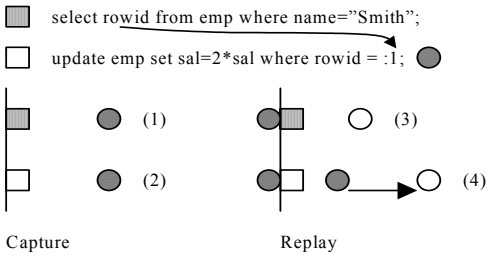


Figure 7: Runtime remapping of system dependent values

Remapping also needs to be done for identifiers that are generated within the server, an example of which includes sequence values. When a replayed call uses a sequence value, the replay infrastructure looks up the respective sequence values used by the workload during the capture phase. The replayed call is then furnished with the exact same values it saw during capture. If the sequence code were not altered during the replay, a replayed call would most probably use different sequence values than those used during capture, resulting in data divergence.

Causes of Data Divergence

A replayed call is considered to *diverge* if

- it affects different number of rows during replay than it did during capture;
- it sees an error during replay and there was none or one with a different error code during capture.

Since the captured data does not contain any results, we cannot know if a call during replay actually has the same results as it did during capture. But for the purposes of replay, row counts and error values are sufficient to indicate whether a replayed call performed similarly as its captured counter part.

Although the Replay technology is advanced enough to deal with synchornization and data remapping, result consistency is not always possible. Some corner cases do exist and in all the workloads we experimented with, these cases appeared in a small percentage of calls and did not invalidate the value of replay as a testing tool. Some example causes of divergence are the following:

- *Multiple commits within PLSQL scripts* - In this case we consider the entire PLSQL block as a single commit operation and don't enforce commit ordering between the statements inside the PLSQL block and the calls from other sessions.
- *User actions that are not synchronized during replay* - This includes, for instance, calls to dbms_pipe, user locks, dbms_lock.sleep, …
- *Use of non-repeatable functions* – Examples of such functions are PLSQL's RANDOM() and SYSDATE.
- *In-flight sessions at the time capture starts* - These sessions may contain calls that operate on data that was not committed before capture started. These calls will diverge because the missing part of the in-flight sessions will not be replayed.
- *Reference to external entities* - When a captured call uses facilities such as database links that point to remote databases it will fail during replay if the remote link does not exist.
- *Interaction with scheduled jobs* - If there are scheduled jobs executing in the test system that interact with the same user data as some replayed sessions, there maybe divergence because we don't order commits within scheduler jobs.

Replay strives to minimize replay divergence in most practical situations. Since it is a testing tool we assume some divergence due to corner cases can be tolerated. This also underscores the necessity of having an Application level validation script to assess the quality of the replay

Replay Options

Server Options

By default, the replay is done in a fully synchronized mode. It has the same transaction commit ordering, the same concurrency and the same timing as the captured workload. Each replay thread is started at the same time it was started at capture time (relatively to the beginning of capture/replay), and the think time recorded during capture between any two consecutive calls in the same session is also honored during replay. This mode ensures minimal data divergence by design. However, it may add some overhead (mainly due to the commit ordering) that could invalidate performance testing in some cases. Therefore we provide options to tweak the replay behavior suitably for each test case.

Commit ordering can be turned on or off. Ignoring commit ordering may be suitable in cases where, during replay, some data divergence can be tolerated. If your application has minimal dependencies between sessions then this mode will not produce significant data divergence and could be useful. It can also be used for stress and load testing.

The remaining three replay options (connect time scale, think time scale and auto-rate) tweak the request rate of the replay. By default connect time scale and think time scale are set to 100%, which is required for request rate consistency. They can be adjusted depending on the situation. For example, when trying to replay a single instance capture on a 4-node RAC system that could handle up to 4 times the request rate compared to a single instance setup, one could set think time scale to 25% thus quadrupling the request rate of the workload issued by the replay clients. Auto-rate is on by default, in which case the replay thread will attempt to maintain the captured request rate by reducing the think time appropriately. If auto-rate is turned off the captured think time is not decreased to compensate for longer executing calls.

Connection Remapping

At capture time, the connection strings used by recorded sessions to connect to the database are recorded. However, the database used for replay is usually not the one used during capture. Therefore, these connection strings need to be remapped by the user before starting the replay. Some possible remappings situations are:

- *One-to-one* – simple instance-to-instance remapping. One can also remap a single connection string to a load-balancing listener in order to test an upgrade from single instance to a RAC system.
- *Many-to-one* – remap several connection strings to a single service in the test system (e.g. load-balancing listener).

Setting Up the Replay Clients

The number of replay clients used during replay is configurable by the user. However, it is recommended that the user run the WRC replay client in calibration mode (`wrc mode = calibrate`) against the processed captured workload. The calibration report provides recommendations on the number of replay clients to be used, as well as the number of CPUs that will be needed. It also gives some information on the number of captured sessions and the maximum concurrency seen during capture. It is strongly recommended to use at least the number of replay clients and CPUs suggested by the calibration report. Failure to do so might result in an unfaithful replay and/or replay client errors.

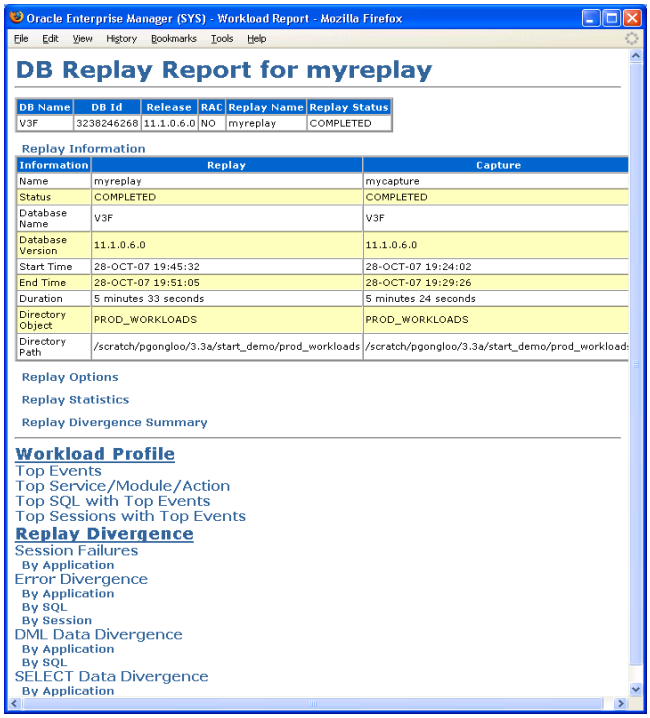
Additionally, the user has to make sure that all the replay clients can connect to the replay database, and that they can access the directory containing the processed capture files. This can be done either by copying the directory to the client machine, or by using a shared file system. If the former approach is followed the contents of the directory on the client machine should not be copied back to the original directory after the replay. This would lead to overwriting vital replay data.

REPLAY ANALYSIS AND REPORTING

After a replay is done, users are able to evaluate the quality of the replay compared with the workload capture. Typical questions answered include:

- What is the general replay information such as start time, finish time, replay options, number of replay clients used etc.?
- Has the replay run into completion? Was it cancelled?
- How fast or slow is the replay compared to the capture, or compared to previous replays?
- Did the replay actually run the expected workload?
- What error and data divergence was seen during replay?
- What were the performance characteristics of the replay?

The Database Replay feature provides replay report to help users to analyze replays. For further investigation, other performance reports (AWR, Compare Period, ASH…) and performance advisors (ADDM) can be used.



Replay Report and Monitoring

Users can use the Enterprise Manager to monitor the progress and system activty of the Replay, and generate a report during or at the end of replay. The replay report includes replay information, replay statistics, workload profile, and replay divergence such as errors encountered during replay and data divergence in rows returned by DML or SQL queries. The above figure is an example of replay report. Due to space limitation, we only show the outline and detailed information for "Replay Information."

The following sections outline how to interpret the various sections of the report.

Session Failure

The replay of a user session might fail and the replay of the remaining calls in the session will be skipped. It is shown as session failure in the replay report. Session failure can be caused by improper system configuration. (eg: max process limit). It can be also caused by crashes from shadow processes or replay threads, in which case, incidents should be found in the client or server log directories.

Error Divergence

An error divergence is reported if the replay of a specific call returns with a new error or a different error. There are three cases:

- *Not Found:* Error encountered during capture not seen during replay
- *New error:* Error encountered during replay not seen during capture
- *Mutated:* Different error produced in replay than during capture

Data Divergence

During capture and replay, the row count of every query (SELECT) or DML (INSERT, UPDATE) is recorded. Differences in row counts are reported as divergence. It is extremely important that users not rely on this data alone.

User-Supplied Data Validation Scripts

Due to the complexity of workload replay, users might easily get lost in the detailed divergence analysis that is provided. It is highly recommended to evaluate the replay quality with application-specific validation scripts.

There are two ways to develop validation scripts. First, users can inject in the workload custom designed scripts that validate key aspects of the data. Outfitting such scripts with the appropriate "module name/module name action" attributes allows the reporting infrastructure to produce a detailed replay specific report on them. The premise is that if such custom designed scripts don't diverge then the replay has done a good job of reproducing the captured workload.

The other way is to develop a script to assess the overall success of the replay. For example, if 10,000 orders are processed during workload capture, you should validate that similar number of orders are also processed during replay. Note in this case, the verification script is not part of the workload capture.

Performance Comparison

Only once the data quality of the replay has been assessed satisfactorily, is it valid to do a performance analysis. The capture and replay reports can be used to do a simple performance comparison. For example, users can compare the duration, wait class/event times etc. Unless the replay was run in unsynchronized mode, significant error or data divergence indicates that the replay did not exercise the database appropriately.

Due to the nature of synchronization an exact db time comparison (in case sycnhronization is turned on) may not be possible. This is due to the fact that synchronization will induce waits on the "wcr: replay clock" idle wait event. Also note that synchronization will eliminate certain events that occurred during capture; these include row lock waits, and a few buffer busy and global cache busy waits events. The time that was attributed to these wait events during capture will appear, for the most part, under the "wcr: replay lock order:" replay wait event.

For in-depth non-replay specific performance study of the replay, existing tools are available to monitor and measure performance. These include AWR, ASH, and AWR compare period report.

Restrictions

Database Replay can capture and replay all the SQL-related and PLSQL calls, including PLSQL RPC. However, the following features are not supported in the current Database release (11.1):

- SQL Loader direct path load, import/export
- OCI based object navigation (ADTs) and REF binds
- Streams, non PLSQL based AQ
- Distributed transactions, remote describe/commit operations
- Flashback queries
- Shared server

BEST PRACTICES

This section summarizes the list of practices that user should consider to effectively use the Database Replay feature.

Capture Planning

- Storage overhead: Storage size needed primarily depends on the workload. User needs to provide adequate disk space for workload capture files. Oracle recommends estimating the size based on running capture for a short period of time.
- Capture period: Capture period should contain interesting workload such as peak time.
- Restarting Database: Restarting the database is optional however if there are in-flight sessions during the start of the capture, the data changed by those sessions before capture is started might cause divergence during the replay. It is recommended to restart the database when possible to minimize the data divergence.
- File System for RAC: When possible, it is recommended to use shared file system.

- AWR Data Export: Exporting AWR provides the statistics needed to enable in-depth replay performance analysis. User should consider the impact on the production system before exporting AWR.

Test System Setup and Workload Capture Processing

- Data Setup: Application Data needs to be identical to the production system in order to minimize the replay divergence. It is recommended to have a plan to ensure the identical duplication.
- Processing: Processing should be done on the test system instead of on the production system as the processing has performance overhead and can possibly take a long time.

Replay Planning

- Isolate Test System: The test system should be isolated from the production system so the test would not interfere the production. Database Replay infrastructure provides flexible directory object creation, and connection remapping for this purpose.
- System Clock: If the application logic involves SYSDATE usage, the user should consider resetting the system clock at the start of the replay to the time at the start of the capture.

Here is another area for sidebar comments. Note that sidebars have no visible borders.

PLSQL PACKAGES

While the primary interface for Database Replay is the Oracle Enterprise Manager, a command line interface is provided through some PLSQL packages and DBA views. Please note that the DBA role is needed to use those commands.

The PLSQL package used to control the workload capture is `dbms_workload_capture`. Here are the main commands that need to be entered to do a capture. Please refer to the documentation for more details.

- `create directory "CAPTURE_DIR" as '/captures/cap1';`
- `dbms_workload_capture.start_capture(name => 'capture1', dir => 'CAPTURE_DIR', duration => NULL);`
- `dbms_workload_capture.finish_capture;`

The PLSQL package used to control the workload replay is `dbms_workload_replay`. In order to process and replay the captured workload, the following commands will be needed:

- `dbms_workload_replay.process_capture('CAPTURE_DIR');`
- `dbms_workload_replay.initialize_replay(replay_name => 'replay1', replay_dir => 'CAPTURE_DIR');`
- `dbms_workload_replay.remap_connection`

- `dbms_workload_replay.prepare_replay(synchronization => TRUE, connect_time_scale => 100, think_time_scale => 100, think_time_auto_correct => TRUE);`
- `dbms_workload_replay.start_replay;`

On the client side, the user will have to use the wrc binary to get a calibration report on the captured workload (in order to know how many replay clients are needed) and to start the Replay cients:

- `%> wrc mode=calibrate replaydir=/captures/cap1`
- `%> wrc user/passwd@inst replaydir=/captures/cap1`

DBA Views

The following views can be queried to get all the information regarding capture and replay in the target databases:

- `dba_workload_captures`
- `dba_workload_replays`
- `dba_workload_replay_divergence`
- `v$_workload_replay_thread`

Please refer to the documentation for a detailed description of these views.

CONCLUSION

Change is relentless in today's rapidly evolving IT environments but it doesn't have to be difficult for data center managers and administrators. Thanks to the new Database Replay feature within the Real Application Testing capabilities in the Oracle Database 11g, database administrators can adapt to changes easily while keeping their undesired effects to a minimum. The capabilities achieved by this unique technology, unmatched by any testing tool in the market, allows the DBA to capture live production workload and replay the same on a test system while maintaining the timing, concurrency and transactional properties of the original workload. The replay system doesn't require the DBA to setup the entire application stack thus further saving time and money for the organization. Changes can be evaluated quickly with high confidence, and corrective action can be taken before business users are negatively impacted by the change. DBAs can finally "Change without risk".

Database Replay
November 2007
Author: [OPTIONAL]
Contributing Authors: [OPTIONAL]

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com