Maximizing Database Performance Using Database Replay

Kurt Engeleiter, Oracle
Arup Nanda, Starwood Hotels and Resorts
Safe Harbor Statements

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.
Program Agenda

- Database Performance Fundamentals
- Database Replay
- Maximizing Performance - Case Studies
- Capacity Planning
Database Performance Fundamentals
Database Tuning Methodology

Ensure host resources are not the bottleneck
- Check that memory is not exhausted (not swapping)
- Tune for CPU first when CPU constrained

Tune to reduce DB Time
- Performance improvement means accomplishing the same amount of work with less DB Time

Validate tuning
- Corroborate tuning results with Real Application Testing
DB Time Definition

- Total time in database calls by foreground sessions
- Includes
  - CPU time
  - IO time
  - Active wait (non-idle wait) time
- Basic unit for Oracle performance analysis
CPU Run-Queue and DB Time

- DB Time is inflated when CPU Bound
Database System Tuning Process

1. Identify the performance issue
2. Scope the issue
3. Set goals
4. Capture data (AWR does this)
Database System Tuning Process

1. Investigate DB Time distribution
   Identify the largest possible improvement

2. Make changes for the largest potential gain

3. Evaluate against goals
   SQL Performance Analyzer
   Database Replay
Database Replay
Database Replay

- Database load and performance testing with real production workloads
  - Production workload characteristics such as timing, transaction dependency, think time, etc., fully maintained
- Identify and remediate application scalability and concurrency problems in multitenant and non-CDB databases
- Allows scheduling, scaleup, subsetting, of multiple workloads
- Concurrent database replay available for 11.2.0.2 and above, MOS Note: 1453789.1
Database Replay Workflow

- Comprehensive interface for Database Replay in Enterprise Manager
- Database Capture page highlights performance data along with workload information
- DB Time displayed in Average Active Session graph
Database Replay Workflow

- Database Replay workflow monitoring:
  - User call progress
  - DB Time summarized in bar chart
  - Replay Divergence Summary enables easy analysis with click through links
Database Replay Workflow

- Rich reporting infrastructure leverages ASH and AWR data
- ASH Analytics predefined reports - new in Enterprise Manager 12c
Database Replay Workflow

- Predefined ASH Analytics reports by:
  - Wait Class
  - User
  - Service
  - Module

- Additional ASH Analytics reports can be generated at will
Maximizing Performance
Case Studies
Case 1
Tuning with DB Time – Case 1

- Current workload, two CPU system
- Average Active Sessions consistently greater than two for this workload
  - Average Active Sessions is DBTime over Time
- Workload to be tuned is captured with Database Replay
Tuning With DB Time – Case 1

- Many DB Time tuning options:
  - Manual Tuning – Read AWR Report
  - Automatic Database Diagnostics Monitor (ADDM)
  - SQL Tuning Advisor
  - SQL Access Advisor
Tuning With DB Time – Case 1

- In this case:

  ADDM
  • Recommends running SQL Tuning Advisor on high DB Time SQL

  SQL Tuning Advisor
  • Recommends additional access structures
  • Suggests running SQL Access Advisor to confirm

  SQL Access Advisor
  • Verifies the benefit for the whole workload
### Tuning with DB Time – Case 1

<table>
<thead>
<tr>
<th>Action</th>
<th>Object Name</th>
<th>Object Attributes</th>
<th>Indexed Columns</th>
<th>Base Table</th>
<th>Schema</th>
<th>Tablespace</th>
<th>Partition Key</th>
<th>SQL Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTITION_TABLE</td>
<td>H_LINEITEM</td>
<td></td>
<td></td>
<td>SALES</td>
<td></td>
<td></td>
<td>PARTITION BY RANGE (&quot;L_SHIPDATE&quot;)</td>
<td>INTERVAL 7 DAY</td>
</tr>
<tr>
<td>PARTITION_TABLE</td>
<td>H_ORDER</td>
<td></td>
<td></td>
<td>SALES</td>
<td></td>
<td></td>
<td>PARTITION BY RANGE (&quot;O_ORDERDATE&quot;)</td>
<td>INTERVAL 30 DAY</td>
</tr>
<tr>
<td>PARTITION_TABLE</td>
<td>H_CUSTOMER</td>
<td></td>
<td></td>
<td>SALES</td>
<td></td>
<td></td>
<td>PARTITION BY RANGE (&quot;C_NATIONKEY&quot;)</td>
<td>INTERVAL 7 DAY</td>
</tr>
<tr>
<td>PARTITION_TABLE</td>
<td>H_SUPPLIER</td>
<td></td>
<td></td>
<td>SALES</td>
<td></td>
<td></td>
<td>PARTITION BY RANGE (&quot;S_NATIONKEY&quot;)</td>
<td>INTERVAL 7 DAY</td>
</tr>
</tbody>
</table>

- SQL Access Advisor recommends partitioning tables as best option for this workload
Tuning with DB Time – Case 1

- Database copied to test system
- Partitioning implemented per SQL Access Advisor recommendation
- Workload replayed with Database Replay
  - Average Active Sessions reduced from two to one during replay
Tuning with DB Time – Case 1

- DB Time reduced from 298 minutes to 73 minutes
- Tuning was a success ✔️
Case 2
Tuning – Case 2

Should I use a new database parameter?

- In database 12c, a new parameter, THREADED_EXECUTIONS, was introduced.
- Definition: “THREADED_EXECUTION specifies whether to enable the multithreaded Oracle model.”
  - Most Oracle background processes run as threads.
- Question: Will this benefit my workload?
Tuning – Case 2
Should I use a new database parameter?

- Workload captured with Database Replay
Tuning – Case 2

Should I use a new database parameter?

- Database restarted
  - `threaded_executions` set to true
  - only six Oracle processes

```plaintext
ora_pmon_o12c
ora_psp0_o12c
ora_vktm_o12c
ora_u004_o12c
ora_u005_o12c
ora_dbw0_o12c
```
Tuning – Case 2

- Workload replayed with Database Replay
- No obvious performance changes
Tuning with DB Time – Case 1

- DB Time improved from 79 to 74 minutes
- Conclusion: THREAD_EXECUTIONS produced a minor improvement for this workload
Case 3
Case 3 – Impact of Encryption

- Customer: large non-profit education service provider
- For data security compliance, customer needs to implement Transparent TS Encryption
- Environment:
  - Solaris
  - Oracle 11g
  - Customer built application
- What will be the impact on performance with encryption enabled?
Case 3 – Impact of Encryption

- Peak activity captured and replayed.
- Very minimal CPU and DB Time impacts observed (less than 1%)
- TSE implementation – success. ✓
Capacity Planning: Database Replay
Capacity Planning Using Database Replay

- Comprehensive scale-up support and what-if scenarios testing
- Scale-up techniques superior to traditional methods
  - Zero-scripting approach extended for scale-up
  - Scales data and user population
  - Realistic data and bindsets
  - Flexible, supports custom workload creation
- Scale-up Strategies
  - Use think time, connect time replay parameters
  - Use scale-up multiplier replay parameter
  - Scale-up by scheduling concurrent replays
  - Scale-up by workload folding
  - Scale-up with multiple PDBs
Scale-up With Connect Time, Think Time

- Increase effective workload by reducing replay parameters - connect and think time scales
  - Same workload executed in shorter duration
  - connect_time: time between session connects
  - think_time: time between user transactions

- Pros
  - Good stress test
  - Works best for OLTP applications such as forms order entry

- Cons
  - Batch applications may not benefit as much due to minimal application latency
Scale-up with Multiplier Replay Parameter

- Increase read-only workload using replay parameter “SCALE_UP_MULTIPLIER”
- Multiplies workload specified number of times
- First replay full DML, all other replays are query only
- Pros:
  - Provides good coverage for databases with significant read portion of workload
- Cons:
  - Full DML is not replicated
Large Internet company captured peak production workload and replayed on Exadata

SCALE_UP_MULTIPLIER used to scale workload to 10x

Spare capacity remained on Exadata after 10x scale up
Scale-up with Workload Scheduling

- Consolidated Database Replay includes workload scheduling
- Workload peaks can be aligned to maximize stress on replay system
- EM Cloud Control 12c also now supports workload scheduling
Scale-up With Multiple PDBs

- Useful for scale-up testing when significant growth in the workload is expected or for testing Multitenant applications with identical schemas

- Process:
  - Set up test system with a single PDB duplicated from the initial database
  - Replicate this PDB as many times as required
  - Copy workload multiple times into consolidated replay directory and direct separate workloads to specific PDBs (services)
Scale-up with Multiple PDBs

- SQL to replicate a PDB:

```sql
CREATE PLUGGABLE DATABASE sales01 FROM sales
FILE_NAME_CONVERT = ('/u01/oracle/oradata/db12c/sales/',
'/u01/oracle/oradata/db12c/sales01/');
```
Scale-up With Multiple PDBs

- Workload to be evaluated for scale up captured.
- System has two CPUs.
- Based on Average Active Sessions, system has room for growth.
Scale-up With Multiple PDBs

- Database Replay executed in two identical PDBs.
- Based on CPU usage, system still has room for additional workload.
Scale-up With Multiple PDBs

- Database Replay executed in three identical PDBs.
- Result: system is CPU saturated.
- Conclusion: system can handle more than double the current workload, but less than triple.
Workload Subsetting

- New in Oracle Database 12c – creation of workload subsets (API)
- Select and replay the most interesting workload intervals
- EM Cloud Control 12c also now supports workload subsetting
Scale-up with Workload Folding

- Scale-up by combining subsetting and scheduling
  - Subset a longer workload into two or more intervals
  - Replay those subsets simultaneously as independent workloads
# Database Replay Scale-up Strategies

## Summary

<table>
<thead>
<tr>
<th>Scale-up Strategy</th>
<th>Workload Suitability</th>
</tr>
</thead>
</table>
| Think and connect time throttling       | - Workloads with think or connect latency  
                                        | - Suitable for OLTP workloads                                                      |
| Scale-up read-only multiplier           | - Workloads that are predominantly query only  
                                        | - Reader farms, Website read-only activity                                          |
| Concurrent Database Replay              | - Enables consolidation validation  
                                        | - Can be used for all workloads                                                    |
| Multiple identical PDBs                 | - Simultaneous workload and data scale-up  
                                        | - Multitenant what-if scenarios                                                    |
| Workload Folding                        | - N * Workload scale-up  
                                        | - Suitable for relatively stateless workloads                                      |
Conclusion

- DB Time is the fundamental metric in database tuning
  - Improve database performance by reducing DB Time

- Validate tuning with Database Replay

- Future proof your environment by using Database Replay for consolidation, capacity planning and scale up testing
Arup Nanda
Director - Principal Database Architect
Starwood Hotels & Resorts Worldwide, Inc.
Identifying Upgrade Issues
Challenge

- Upgrading from Oracle 10g R2 → 11g R2
- HPUX Itanium → AIX
- Across Data Centers: Houston → Phoenix
- Downtime expected: 2 hours max

**Question: What all can fail?**
What We Did

- **Methodology**
  - Captured workload for 1 month in 1 hour chunks
  - Replayed continuously against the new system

- **Result**
  - Discovered database bug (actually “fixed” an earlier bug 9824198)
  - "ORA-00979: not a GROUP BY expression"
  - Where, how many places?
Identification of SQLs

select service, module, action, sql_id, count(*)
from   dba_workload_replay_divergence
Where  observed_error# = 979
group by service, module, action, sql_id;

<table>
<thead>
<tr>
<th>SERVICE</th>
<th>MODULE</th>
<th>ACTION</th>
<th>SQL_ID</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>4st8fbfa5cpt2</td>
<td>132</td>
</tr>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>4karrnm9kt23k</td>
<td>13</td>
</tr>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>2syvktdcxxrsh</td>
<td>5</td>
</tr>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>2jku6m3bpk3yc</td>
<td>1</td>
</tr>
<tr>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
<td>cg7kdz4gsrnu5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Summary of Benefits

- Saved months of work for a team of 30 developers
- System tuning to perfection – in a week
- Reduction of the realm of the unknown