Oracle Database Performance Tuning Guide 阅读笔记

Managign snapshots

eg:

```
begin
    dbms_workload_repository.create_snapshot();
end;
/
```

Note: a snapshot is created immediately on the local database instance with the flush level specified to the default flush level of TYPICAL. To view information about the newly created snapshot (and any existing snapshots), use the DBA_HIST_SNAPSHOT view.

eg:

```
begin
    dbms_workload_repository.drop_snapshot_range(low_snap_id=>22, high_snap_id=32, dbid=> 3310949047);
end;
/
```

Note: snapshots with snapshot IDs ranging from 22 to 32 are dropped immediately from the database instance with the database identifier of 3310949047. Any ASH data that were captured during this snapshot range are also purged.

eg:

```
begin
    dbms_workload_repository.modify_snapshot_settings(retention => 4, interval => 30, topnsql => 100, dbid => 3310949047);
end;
/
```

Note: the snapshot setting for the database with the database identifier 3310949047 are modified as follows:
The retention period is specified as 43200 minutes (30 days).
The interval between each snapshot is specified as 30 minutes.
The number of Top SQL to flush for each SQL criteria is specified as 100.

Managing Baselines

eg:

```
begin
    dbms_workload_repository.create_baseline(start_snap_id => 270, end_snap_id => 280, baseline_name => 'peak baseline', dbid => 
3310949047, expiration => 30);
end;
/
```

Note: a baseline is created on the database instance with the database identifier of 3310949047 with the following settings:
The start snashot sequence number is 270.
The end snapshot sequence number is 280.
The name of baseline is peak baseline.
The expiration of the baseline is 30 days.

eg:

```
begin
    dbms_workload_repository.drop_baseline (baseline_name=> 'peak baseline', cascade => false, dbid => 3310949047);
end;
/
```

Note: the baseline peak baseline is dropped from the database instance with the database identifier of 3310949047 and the associated snapshots are preserved.

eg:

```
begin
    dbms_workload_repository.rename_baseline(old_basline_name => 'peak baseline', new_baseline_name => 'peak mondays', dbid => 
3310949047);
end;
/
```

Note: the name of the baseline on the database instance with the database identifier of 3310949047 is renamed from peak baseline to peak mondays.

eg: to display the summary statistics for metric values in a baseline period using the command-line interface, use the SELECT_BASELINE_METRICS function.

```
dbms_workload_repository.select_baseline_metrics(baseline_name in varchar2, dbid in number default null, instance_num in number default null) return awr_baseline_metric_type_table pipelined;
```

eg:

```
begin
    dbms_workload_repository.modify_baseline_window_size(window_size=>30, dbid=> 3310949047);
end;
/
```

Note: the default moving window is resized to 30 days on the database instance with the database identifier of 3310949047.

Managing Baseline Templates

eg:

```
begin
    dbms_workload_repository.create_baseline_template(
        start_time => '2012-04-02 17:00:00 PST',
        end_time => '2012-04-02 20:00:00 PST',
        baseline_name => 'baseline_120402',
        template_name => 'template_120402',
        expiration => 30,
        dbid=> 3310949047
    );
end;
/
```

Note: a baseline template named template_120402 is created that will generate a baseline named baseline_120402 for the time period from 5:00 p.m. to 8:00 p.m. on April 2, 2012 on the database with a dbid of 3310949047. The baseline will expire after 30 days.

eg: to create a repeating baseline template using command-line

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE (
                   day_of_week => 'monday', hour_in_day => 17,
                   duration => 3, expiration => 30,
                   start_time => '2012-04-02 17:00:00 PST',
                   end_time => '2012-12-31 20:00:00 PST',
                   baseline_name_prefix => 'baseline_2012_mondays_',
                   template_name => 'template_2012_mondays',
                   dbid => 3310949047);
END;
/
```

Note: In this example, a baseline template named template_2012_mondays is created that will generate a baseline on every Monday from 5:00 p.m. to 8:00 p.m. beginning on April 2, 2012 at 5:00 p.m. and ending on December 31, 2012 at 8:00 p.m. on the database with a database ID of 3310949047. Each of the baselines will be created with a baseline name with the prefix baseline_2012_mondays_ and will expire after 30 days.

eg: To drop a baseline template using command-line

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE_TEMPLATE (
                   template_name => 'template_2012_mondays',
                   dbid => 3310949047);
END;
/
```

Note: the baseline template named template_2012_mondays is dropped from the database instance with the database identifier of 3310949047.

Transporting Automatic Workload Repository Data
eg: to extract AWR data
@$ORACLE_HOME/rdbms/admin/awrextr.sql
and then enter the necessary parameter as prompted.

Loading AWR Data
eg: to load AWR data
@$ORACLE_HOME/rdbms/admin/awrload.sql

Using Automatic Workload Repository Views
DBA_HIST_ACTIVE_SESS_HISTORY
DBA_HIST_BASELINE
DBA_HIST_BASELINE_DETAILS
DBA_HIST_BASELINE_TEMPLATE
DBA_HIST_DATABASE_ISNTANCE
DBA_HIST_DB_CACHE_ADVICE
DBA_HIST_DISPATCHER
DBA_HIST_DYN_REMASTER_STAS
DBA_HIST_IOSTAT_DETAIL
DBA_HIST_SHARED_SERVER_SUMMARY
DBA_HIST_SNAPSHOT
DBA_HIST_SQL_PLAN
DBA_HIST_WR_CONTROL

Generating Automatic Workload Repository Reports
User Interface for Generating an AWR Report : Oracle Enterprise Manager
or
Generating an AWR Report Using the Command-Line Interface
The DBA role is required to run these scripts.
1. Generating an AWR Report for the Local Database
@$ORACLE_HOME/rdbms/admin/AWRRPT.SQL
2. Geberate an AWR report on a specific database instance using the command-line interface
@$ORACLE_HOME/rdbms/amdin/awrrpti.sql
3. Generating an Oracle RAC AWR Report for the Local Database
@$ORACLE_HOME/rdbms/admin/awrgrpt.sql
4. Generating an Oracle RAC AWR Report for a Specific Database
@$ORACLE_HOME/rdbms/admin/awrgrpti.sql
5. Generating an AWR Report for a SQL Statement on the Local Database
@$ORACLE_HOME/rdbms/admin/awrsqrpt.sql
6. Generating an AWR Report for a SQL Statement on a Specific Database
@$ORACLE_HOME/rdbms/admin/awrsqrpi.sql
7. Generating a Performance Hub Active Report Using a SQL Script
@$ORACLE_HOME/rdbms/admin/perfhubrpt.sql

ADDDM Analysis
An ADDM analysis can be performed on a pair of AWR snapshots and a set of instances from the same database.

Comparing Database Performance Over Time
User Interfaces for Generating AWR Compare Periods Reports
The primary interface for generating AWR Compare Periods reports is Oracle Enterprise Manager.

Generating an AWR Compare Periods Report Using the Command-Line Interface
Generating an AWR Compare Periods Report on the Local Database instance

@$ORACLE_HOME/rdbms/admin/awrddrpt.sql
Generating an AWR Compare Periods Report for a Specific Database
@$ORACLE_HOME/rdbms/admin/awrddrpi.sql
Generating an Oracle RAC AWR Compare Periods Report for the Local Database
@$ORACLE_HOME/rdbms/admin/awrgdrpt.sql
Generating an Oracle RAC AWR Compare Periods Reort for a Specific Database
@$ORACLE_HOME/rdbms/admin/awrgdrpi.sql

Analyzing Sampled Data
Generating Active Session History Reports
User Interfaces for Generating ASH Reports
The primary interface for generating ASH reports is Oracle Enterprise Manager.

Generating an ASH Report Using the Command-Line Interface
to generate an ASH report on the local database instance
@$ORACLE_HOME/rdbms/admin/ashrpt.sql
to generate an ASH report on a specific database instance using the command-line interface
@$ORACLE_HOME/rdbms/admin/ashrpti.sql
Generating an ASH Report for Oracle RAC
to generate an ASH report for Oracle RAC
@$ORACLE_HOME/rdbms/admin/ashrpti.sql

10 Instance Tuning Using Performance Views
Instance Tuning Steps
1. Define the Problem
2. Examine the Host System and Examine the Oracle Database Statistics
3. Implement and Measure Change
4. Iterate 1, 2, 3.

To reduce I/O load caused by Oracle Database, examine the I/O satistics collected for all I/O calls made by the database using the following views
V$IOSTAT_CONSUMER_GROUP : captures I/O statistics for consumer groups.
V$IOSTAT_FILE : captures I/O statistics of database files that are or have been accessed.
V$IOSTAT_FUNCTION : captures I/O statistics for database functions (such as the LGWR and DBWR).
eg:
select function_id, function_name from v$iostat_function order by function_id;
V$IOSTAT : contains I/O statistics for both single and multi block read and write operations.

Identifying I/O Problems Using Operating System Monitoring Tools
eg:
sar -d
or
iostat

Identifying Network Issues
V$IOSTAT_NETWORK

Examine the Oracle Database Statistics
Setting the Level of Statistics Collection
STATISTICS_LEVEL : the initialization parameter which controls all major statistics collection or advisories in the database.
three levels:
BASIC
TYPICAL
ALL

Dynamic Performance Views Containint Wait Event Statistics

V$ACTIVE_SESSION_HISTORY
V$SESS_TIME_MODEL : displays active database session activity, sampled once every second.
V$SYS_TIME_MODEL : contain time model statistics, including DB time which is the total time spent in database calls.
V$SESSION_WAIT : displays information about the current or last wait for each session.
V$SESSION : displays info. about each current session and contains the same wait statistics as those found in the V$SESSION_WAIT view.
V$SESSION_EVENT : provides summary of all the events the session has waited for since it started.
V$SESSION_WAIT_CLASS : provides the number of waits ant the time spent in each class of wait events for each session.
V$SESSION_WAIT_HISTORY : displays information about the last ten wait events for each active session.
V$SYSTEM_EVENT : provides a summary of all the event waits on the instance since it started.
V$EVENT_HISTOGRAM : displays a histogram of the number of waits, the maximum wait, and total wait time on an event basis.
V$FILE_HISTOGRAM : displays a histogram of times waited during single block reads for each file.
V$SYSTEM_WAIT_CLASS : provides the instnace wide time totals for the number of waits and the time spent in each class of wait events.
V$TEMP_HISTOGRAM : displays a histogram of times waited during single block reads for each temporary file.

System Statistics
V$ACTIVE_SESSION_HISTORY : displays active database session activity, sampled once every second.
V$SYSSTAT : contains overal statistics for many different parts of Oracle Database, including rollback, logical and physical I/O, and parse data.
V$FILESTAT : contains detailed file I/O statistics for each file, including the number of I/Os for each file and the average read time.
V$ROLLSTAT : contains detailed rollback and undo segment statistics for each segment.
V$ENQUEUE_STAT : contains detailed enqueue statistics for each enqueue, including the number of times an enqueue was requested and the number of times an enqueue was waited for, and the wait time.
V$LATCH : contains detailed latch usage statistics for each latch, including the number of times each latch was requested and the number of tims the latch was waited for.

Segment-Level Statistics
You can query segment-level statistics through the following dynamic performance views:
V$SEGSTAT_NAME : lists the segment statistics being collected and the properties of each statistic.
V$SEGSTAT : a highly efficient, real-time monitoring view that shows the statistic value, statistic name, and other basic information.
V$SEGMENT_STATISTICS : a user-friendly view of statisticvalues. In addition to all the columns of V$SEGSTAT, it has information about such things as the segment owner and table space name. It amkes the statistics easy to understand, but it is more costly.

Implement and Measure Change

Interpreting Oracle Database Statistics
Examine Load
Load-related statistics to examine include redo size, session logical reads, db block changes, physical reads, physical read total bytes, physical writes, physical write total bytes, parse count (total), parse count (hard), and user calls. This data is queried from V$SYSSTAT.

Changing Load

High Rates of Activity
1. A hard parse rate of more than 100 a second indicates that there is a very high amount of hard parsing on the system.
2. Check whether the sum of the wait times for library cache and shared pool latch events is significant compared to statistic DB time found in V$SYSSTAT. If so, examine the SQL ordered by Parse Calls section of the AWR report.
A high soft parse rate could be in the rate of 300 a second or more.

Using Wait Event Statistics to Drill Down to Bottlenecks
The most effective way to use wait event data is to order the events by the wait time. This is only possible if TIMED_STATISTICS is set to true. Otherwise, the wait events can only be ranked by the number of times waited, which is often not the ordering that best represents the problem.
To get an indication of where time is spent, follow these steps:
1. Examine the data collection for V$SYSTEM_EVENT. The events of interest should be ranked by wait time.

Alternatively, look at the Top 5 Timed Events section at the beginning of the Automatic Workload Repository report.
2. Look at the number of waits for these events, and the average wait time.
3. The top wait events identify the next places to investigate. It is ususlly a good idea to also have quick look at high-load SQL.
4. Examine the related data indicated by the wait events to see what other information this data provides. Determine whether this information is consistent with the wait event data.
5. To determine whether this theory is valid, cross-check data you have examined with other statistics available for consistency.

Table of Wait Events and Potential Causes
buffer busy waits : depends on buffer type, eg. waits for an index block may be caused by a primary key that is based on an ascending sequence. (V$SESSION)
free buffer waits : slow DBWR (possibly due to I/O), or Cache too small. (Examine write time using operating system statistics. Check buffer cache statistics for evidence of too small cache.)
db file scattered read : Poorly tuned SQL, or Slow I/O system. (Investigate V$SQLAREA to see whether there are SQL statements performing many disk reads. Cross-check I/O system and V$FILESTAT for poor read time.)
db file sequential read : Poorly tuned SQL, or Slow I/O system. (same as the last one.)
enqueue waits (waits starting with enq:) : Depends on type of enqueue. (Look at V$ENQUEUE_STAT)
library cache latch waits : library cache, library cache pin, and library cache lock : SQL parsing or sharing. (Check V$SQLAREA to see whether there are SQL statements with a relatively high number of parse calls or a high number of child cursors. Check parse statistics in V$SYSSTAT and their corresponding rate for each second.)
log buffer space: Log buffer small, or Slow I/O system. (Check the statistic redo buffer allocation retries in V$SYSSTAT. Check configuring log buffer section in configuring memory chapter. Check the disks that house the online redo logs for resource contention.)
log file sync : Slow disks that store the online logs, Un-batched commits. (Check the disks that house the online redo logs for resource contention. Check the number of transactions (commit + rollbacks) each second, from V$SYSSTAT.

Additional Statistics (There are several statistics taht can indicate performance problems that do not have corresponding wait events.)

Redo Log Space Requests Statistic
The V$SYSSTAT statistic redo log space requests indicates how many times a server process had to wait for space in the online redo log, not for space in the redo log buffer. Use this statistic and the wait events as an indication that you must tune checkpoints, DBWR, or archiver activity, not LGWR. Increasing the size of the log buffer does not help.

Read Consistency

Table Fetch by Continued Row
You can detect migrated or chained rows by checking the number of table fetch continued row statistic in V$SYSSTAT.
row migration
row chaining
$ORACLE_HOME/rdbms/admin/utlchn1.sql

Parse-Related Statistics
eg:
select name, value fromv$SYSSTAT where name in ('parse time cpu', 'parse time elapsed','parse count (hard)','CPU used by this session');

Wait Events Statistics

buffer busy waits
eg: to determine the possible causes, query V$SESSION to identify the value of ROW_WAIT_OBJ# when the session waits for buffer busy waits.
select row_wait_obj$ from V$SESSION where event = 'buffer busy waits';
eg: to identify the object and object type contended for, query DBA_OBJECTS using the value for ROW_WAIT_OBJ# that is returned from V$SESSION.
select owner, object_name, subobject_name, object_type from dba_objects where dba_object_id = &row_wait_obj;

If the contention is on the segment header,

eg: to find the current settign for free lists for that segment

select segment_name, freelists from dba_segments where segment-name = segmentname and segment_type = segmenttype;

Set free lists, or increase the number of free lists, or use free list groups.

If the contention is on tables or indexes (not the segment header)

Check for right-hand indexes. These are indexes that are inserted into at the same point by many processes

Consider using ASSM, global hash partitioned indexes, or increasing free lists to avoid multiple processes attempting to insert into the same block.

For contention on rollback segment header

If not using automatic undo management, then add more rollback segments.

For contention on rollback segment block.

If not using automatic undo management, then consider making rollback segment sizes larger.

db file scattered read

Inadequate I/O Distribution

Finding the SQL Statement executed by Sessions Waiting for I/O

eg: to determine which sessions are waiting for I/O

select sql_address, sql_hash_value from V$SESSION where event like 'db file%read';

Finding the Object Requiring I/O

eg: to identify the value of row_wait_obj# when the session waits for db file scattered read

eg:

select row_wait_obj# from V$SESSION where event = 'db file scattered read';

eg: to identify the object and object type contended for.

eg:

select owner, object_name, suboeject_name, object_type from dba_objects where data_object_id = &row_wait_obj;

db file sequential read

direct path read and direct path read temp