Parameter Dependency and the STATISTICS_LEVEL Parameter

By Yong Huang

Oracle initialization parameters greatly influence the behavior and performance of the Oracle Database. It is important for DBAs and Developers to understand the effects of setting various values in these parameters as well as the interdependencies amongst these parameters. In this article, we will see how certain parameters define the values for other parameters, and drill deeper using the STATISTICS_LEVEL initialization parameter.

Parameter Dependency

Many initialization parameters determine the values of other parameters. For instance, setting OPTIMIZER_FEATURES_ENABLE to a certain version will adjust many other Cost-Based-Optimizer (CBO) parameters, changing the NLS_LANGUAGE parameter also changes the NLS_SORT parameter, and increasing the value of the PROCESSES parameter also correspondingly increases the derived value of the SESSIONS parameter, which in turn increases the derived value of the TRANSACTIONS parameter. Wouldn't it be nice if Oracle could give us a view named $PARAMETER_DEPENDENCY so we can have this output? With this in mind assume that we have the following hypothetical view named V$PARAMETER_DEPENDENCY where Oracle exposes these dependencies:

```
SQL> desc V$PARAMETER_DEPENDENCY
 Name            Null?   Type
 --------------- ------- --------
 NUM                     NUMBER
 NAME                    VARCHAR2(80)
...
 PNUM                    NUMBER

SQL> select lpad(' ',2*(level-1)) || name name
  2  from V$PARAMETER_DEPENDENCY
  3 connect by prior num = pnum;

NAME
------------------------
...
nls_language
  nls_sort
...
processes
  sessions
    transactions
...
```

The pearl of this new view is not on the surface, but on its underlying X$ table, perhaps hypothetically called X$KSPPDP (name modeled after X$KSPPI and related view), where you'll find this:

(Warning: Query from the hypothetical fixed table!)

```
SQL> select lpad(' ',2*(level-1)) || ksppdpnm name
  2 from X$KSPPDP
  3 connect by prior indx = pindx;

NAME
--------------------------------
...
optimizer_features_enable
  _optim_peek_user_binds
  _optimizer_or_expansion
  _optimizer_undo_cost_change
...
statistics_level
  _ash_enable
  _rowsource_execution_statistics
  _ultrafast_latch_statistics
...
```

Besides satisfying our curiosity, this information can help us in our work. For instance, the Reference manual page for OPTIMIZER_FEATURES_ENABLE is supplemented with individual control knobs a DBA or developer can experiment with in SQL tuning. But the actual parameter names for these controls are not documented. The query on the hypothetical X$ table would have been able to help us identify the parameter we need.

Knowledge of parameter dependency can be used in two ways. For example, in the case of the STATISTICS_LEVEL parameter, you can either leave the parent parameter, (STATISTICS_LEVEL) at a higher level (TYPICAL), and disable certain parameters that would otherwise be enabled, or set the parent to a lower level (BASIC), and enable a few child parameters that would have otherwise been disabled. Needless to say, you should always consult with Oracle support before you change or set any underscore parameter.

Alas, the current version of Oracle does not have such a parameter dependency view. However, we can build this dependency list ourselves. More and more parameters are modified to be dynamically changeable without shutting down the instance. The dependency information for those parameters can be generated with code similar to the following (only code using undocumented tables and columns is shown):

```
--Prepare a parameter table to work with, based on X$KSPPI, table of
--all init params, and X$KSPPSV, table of their system level values
create table PARAM as
select a.ksppinm, a.ksppity, b.ksppstvl
          from X$KSPPI a, X$KSPPSV b
where a.indx=b.indx order by a.indx;

--Prepare a temporary table to store result of each run
create table TMP as select * from PARAM where 1=2;
```

continued on page 26

```
for x in (select * from PARAM) loop
  if (x.ksppity = 1) then -- boolean
    begin
      if (x.ksppstvl = 'TRUE') then
        execute immediate 'alter system set ' || x.ksppinm || ' to false
scope=memory';
        insert into TMP select select a.ksppinm, a.ksppity, b.ksppstvl
          from X$KSPPI a, X$KSPPSV b where a.indx=b.indx order by a.indx;
        execute immediate 'alter system set ' || x.ksppinm || ' to true
scope=memory'; -- change it back
      else
        execute immediate 'alter system set ' || x.ksppinm || ' to true
scope=memory';
        insert into TMP select select a.ksppinm, a.ksppity, b.ksppstvl
          from X$KSPPI a, X$KSPPSV b where a.indx=b.indx order by a.indx;
        execute immediate 'alter system set ' || x.ksppinm || ' to false
scope=memory'; -- change it back
      end if;
      --compare PARAM with TMP and store the diff somewhere
      truncate table TMP;
    exception when ORA-2095 then -- "specified initialization parameter cannot be
modified"
      --record the param in a table so we bounce DB later to change it
  elsif (x.ksppity = 2) then -- string
    --append a letter to the value, insert into TMP, remove the letter
...
end if
```

Some parameters will need a database bounce. Discovery of their dependency is better done manually for each of those parameters, although a sophisticated shell script may be able to automate the process.

STATISTICS_LEVEL

There are a few parameters whose dependency is well-known. But the details of the one about database statistics, STATISTICS_LEVEL, are not well recorded in the documentation and elsewhere. STATISTICS_LEVEL is a parameter that affects various types of statistics gathering and DBA advisories. Most shops rarely change its value from its default, TYPICAL value, to either BASIC or ALL. When they do, it is usually done using the ALTER SYSTEM command rather than ALTER SESSION to set it to ALL, since the intent is usually only temporary: i.e. gathering more extensive statistics for a short period of time and then turn it back to TYPICAL. Some companies have extreme demand on the raw horsepower of the database engine and so set it to BASIC. Regardless which value you turn it to, it is helpful to understand its implications. Like the OPTIMIZER_FEATURES_ENABLE parameter, the Reference manual has the details of the changes this parameter brings to the database. In addition, Oracle provides a view listing the details. The output below is formatted to be displayed in a pivoted form, from a Oracle Database 10.2.0.4 with STATISTICS_LEVEL set to TYPICAL, DB_CACHE_ADVICE set to OFF, with no changes made to any underscore parameter. You can see whether each performance feature (STATISTICS_NAME) is enabled, at what STATISTICS_LEVEL it will be enabled for you (ACTIVATION_LEVEL), and what dynamic performance view is relevant, among other things.

```
( select STATISTICS_NAME, SYSTEM_STATUS, ACTIVATION_LEVEL,
STATISTICS_VIEW_NAME, SESSION_SETTABLE, DESCRIPTION
from V$STATISTICS_LEVEL
order by 1)

STATISTICS_NAME      : Active Session History
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$ACTIVE_SESSION_HISTORY
SESSION_SETTABLE     : NO
DESCRIPTION          : Monitors active session activity using MMNL

STATISTICS_NAME      : Bind Data Capture
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$SQL_BIND_CAPTURE
SESSION_SETTABLE     : NO
DESCRIPTION          : Enables capture of bind values used by SQL statements

STATISTICS_NAME      : Buffer Cache Advice
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$DB_CACHE_ADVICE
SESSION_SETTABLE     : NO
DESCRIPTION          : Predicts the impact of different cache sizes on number
                       of physical reads

STATISTICS_NAME      : Global Cache Statistics
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : null
SESSION_SETTABLE     : NO
DESCRIPTION          : RAC Buffer Cache statistics

STATISTICS_NAME      : Longops Statistics
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$SESSION_LONGOPS
SESSION_SETTABLE     : NO
DESCRIPTION          : Enables Longops Statistics

STATISTICS_NAME      : MTTR Advice
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$MTTR_TARGET_ADVICE
SESSION_SETTABLE     : NO
DESCRIPTION
: Predicts the impact of different MTTR settings on number of physical I/Os

STATISTICS_NAME      : Modification Monitoring
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : null
SESSION_SETTABLE     : NO
DESCRIPTION          : Enables modification monitoring

STATISTICS_NAME      : PGA Advice
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$PGA_TARGET_ADVICE
SESSION_SETTABLE     : NO
DESCRIPTION          : Predicts the impact of different values of
                       pga_aggregate_target on the performance of memory
                       intensive SQL operators

STATISTICS_NAME      : Plan Execution Statistics
SYSTEM_STATUS        : DISABLED
ACTIVATION_LEVEL     : ALL
STATISTICS_VIEW_NAME : V$SQL_PLAN_STATISTICS
SESSION_SETTABLE     : YES
DESCRIPTION          : Enables collection of plan execution statistics

STATISTICS_NAME      : Segment Level Statistics
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$SEGSTAT
SESSION_SETTABLE     : NO
DESCRIPTION          : Enables gathering of segment access statistics

STATISTICS_NAME      : Shared Pool Advice
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$SHARED_POOL_ADVICE
SESSION_SETTABLE     : NO
```

```
DESCRIPTION          : Predicts the impact of different values of
                       shared_pool_size on elapsed parse time saved

STATISTICS_NAME      : Streams Pool Advice
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$STREAMS_POOL_ADVICE
SESSION_SETTABLE     : NO
DESCRIPTION          : Predicts impact on Streams performance of different
                       Streams pool sizes

STATISTICS_NAME      : Threshold-based Alerts
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : null
SESSION_SETTABLE     : NO
DESCRIPTION          : Controls if Threshold-based Alerts should be enabled

STATISTICS_NAME      : Timed OS Statistics
SYSTEM_STATUS        : DISABLED
ACTIVATION_LEVEL     : ALL
STATISTICS_VIEW_NAME : null
SESSION_SETTABLE     : YES
DESCRIPTION          : Enables gathering of timed operating system statistics

STATISTICS_NAME      : Timed Statistics
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : null
SESSION_SETTABLE     : YES
DESCRIPTION          : Enables gathering of timed statistics

STATISTICS_NAME      : Ultrafast Latch Statistics
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : null
SESSION_SETTABLE     : NO
DESCRIPTION          : Maintains statistics for ultrafast latches in the
                       fast path

STATISTICS_NAME      : Undo Advisor, Alerts and Fast Ramp up
SYSTEM_STATUS        : ENABLED
ACTIVATION_LEVEL     : TYPICAL
STATISTICS_VIEW_NAME : V$UNDOSTAT
SESSION_SETTABLE     : NO
DESCRIPTION          : Transaction layer manageability features
```

The default value TYPICAL is perfect for most cases. But if you want a little more to be done, say to just collect *timed OS statistics*, and do not want to have the overhead of other collections turned on by setting the STATISTICS_LEVEL parameter to ALL, you must find exactly what parameter to set. In this case, it is easy: Just set TIMED_OS_STATISTICS to TRUE. Going the other way, say you want to turn off *PGA Advice* because you do not want to monitor V$PGA_TARGET_ADVICE, you can set an underscore parameter _SMM_ADVICE_ENABLED to FALSE. Since the TYPICAL setting means most statistics are enabled (see SYSTEM_STATUS column above), it's likely you will individually turn some off more often than turn some on.

Back to the BASIC setting: Even though documentation sometimes gives advice such as "To disable monitoring of a table, set the STATISTICS_LEVEL initialization parameter to BASIC," the BASIC setting should not be taken lightly. If most team members understand the implications quite well and you have a good support contract with Oracle, you can do so, but remember that several parameters are essential and should be enabled. For example, to provide even basic performance tuning, TIMED_STATISTICS must be turned on (unless you run the buggy Oracle 8.1.5). Losing statistics on some critical latches can significantly limit your observation of the database health, so you may need to set the hidden parameter _ULTRAFAST_LATCH_STATISTICS to TRUE. Do you use ASH (Active Session History), the wonderful performance monitoring facility? Unless you do not have the license or have one similarly built in-house, perhaps with direct SGA attach to minimize overhead, you will need to set _ASH_ENABLE to TRUE. V$SESSION_LONGOPS is probably essential, particularly in a data warehouse, so you will specifically have to set _LONGOPS_ENABLED to TRUE. Do you take chances and check V$SQL_BIND_CAPTURE for captured bind variables, in case you are lucky? Set _CURSOR_BIND_CAPTURE_AREA_SIZE to some value. If you use V$SEGSTAT regularly, _OBJECT_STATISTICS needs to be TRUE. Everything else may not be essential.

Given the above, the following is a summary of the Oracle features or capabilities controlled by the STATISTICS_LEVEL parameter, followed by the precisely targeted child parameter. The parameters are listed in alphabetic order for easy reference. The findings are from research on Metalink, various sources on the Internet, and my own lab tests.

Active Session History: _ASH_ENABLE

Active Session History (ASH) is a great feature. See the Performance Tuning manual for details. Enable it unless you have a license issue or have your home-grown script.

Bind Data Capture: _CURSOR_BIND_CAPTURE_AREA_SIZE (set to a non-zero value)

V$SQL_BIND_CAPTURE only captures bind variable values during a hard parse, a soft parse that creates a new child cursor, or if the last capture was _CURSOR_BIND_CAPTURE_INTERVAL seconds or longer ago, column type is not LONG or LOB, and bind variables in the select list are ignored. If this is not the case and no values are captured at all, it is possible that _CURSOR_BIND_CAPTURE_AREA_SIZE needs to be increased. These are the limitations I know about Oracle's automatic cursor bind value capture. If you decide to not use this feature, set this parameter to 0, and use 10046 trace at level 4 or 12 only when you need it.

Buffer Cache Advice: DB_CACHE_SIZE

Oracle can predict how much buffer cache hit ratio will improve for a certain amount of increase in buffer cache size. But it comes with a price, not withstanding the usefulness of this hit ratio. The feature uses some CPU cycles as well as memory (100 bytes per buffer according to Metalink Note 148511.1). Generally, you give all remaining memory of the server to buffer cache after you consider other SGA components and predict how much total PGA could be. Isn't that what you're supposed to do regardless of buffer cache advice?

Global Cache Statistics: _GC_STATISTICS

On RAC, you almost certainly need this to be TRUE, or the statistics in GV$G% views will be frozen.

Longops Statistics: _LONGOPS_ENABLED

For a data warehouse database, you probably need to check V$SESSION_LONGOPS periodically. For OLTP, if you prefer, you can enable this parameter only for a while perhaps during data loading and disable it when done. But the inconvenience is probably not worth disabling it.

MTTR Advice: _DB_MTTR_ADVICE

Sets or disables the Mean-Time-To-Recover (MTTR) Advisor.

continued on page 28

Modification Monitoring: _DML_MONITORING_ENABLED

This Oracle Database 10*g* parameter allows you to do what you used to be able to do in 9*i* with the ALTER TABLE (NO)MONITORING command. However, this is not at the table level anymore: It is system-wide with monitoring enabled for all or none. I have observed that disabling table monitoring may not gain you much performance. To verify this, check buffer gets and executions of the recursive SQLs that update or insert into MON_MODS$ in V$SQLSTATS.

PGA Advice: _SMM_ADVICE_ENABLED

Self-explanatory.

Plan Execution Statistics: _ROWSOURCE_EXECUTION_STATISTICS

In Oracle9*i* Database, SQL trace will not give you buffer gets information for each step in the execution plan in the dump file or in V$SQL_PLAN_STATISTICS, unless you turn this option on (and hard parse the cursor). Most people use the ALTER SESSION command to enable it at session level during SQL tuning. There is little value in setting it permanently system-wide. If you want the effect on individual SQLs instead of the whole session you are in, use the `gather_plan_statistics` hint instead. In 10*g*, you can get these row level statistics by setting this parameter with ALTER SESSION without enabling SQL trace.

Segment Level Statistics: _OBJECT_STATISTICS

Object statistics are important to identify hot objects. But there is possible memory leak in querying V$SEGSTAT or V$SEGMENT_STATISTICS. See Bug 3519807 for versions in which the bug is fixed.

Shared Pool Advice: _LIBRARY_CACHE_ADVICE

If you check V$SHARED_POOL_ADVICE regularly, you may leave this on. Note that whatever `%simulator%` latch activity is mostly due to these nearly unused advisors. On top of that, Bug 6879763 affects versions 10.2.0.3 and 11.1.0.6. Consider setting it to FALSE if *shared pool simulator* latch is one of the top (say) 10 in latch gets. By the way, in spite of the name of the parameter, the view related to this parameter is not named V$LIBRARY_CACHE_ADVICE.

Streams Pool Advice: _DISABLE_STREAMS_POOL_AUTO_TUNING

Self-explanatory.

Threshold-based Alerts: _THRESHOLD_ALERTS_ENABLE

If you do not use EM (Enterprise Manager) and do not check views like DBA_OUTSTANDING_ALERTS and DBA_THRESHOLDS, you must have your home-grown monitoring scripts. Then you can consider disabling this parameter by setting it to 0.

Timed OS Statistics: TIMED_OS_STATISTICS

Oracle clearly warns us about the overhead. In spite of the name, V$OSSTAT is not related to this parameter and will always contain data. On pre-Solaris 10, this parameter turns on microstate accounting.

Timed Statistics: TIMED_STATISTICS

Must be true. Too well-known to discuss further.

Ultrafast Latch Statistics: _ULTRAFAST_LATCH_STATISTICS

If this parameter is false, you lose latch statistics in V$LATCH% views for a few important latches, including *cache buffers chains* latches, most likely not what you want. By the way, based on the name, it seems some latches are called *ultrafast latches*, and most are therefore non-fast latches by following this logic.

Undo Advisor, Alerts and Fast Ramp up: _DISABLE_TXN_ALERT

Fast ramp-up is explained in Metalink Note 396863.1. But I could not find more information about this parameter. If you set STATISTICS_LEVEL to BASIC, _DISABLE_TXN_ALERT is 659. When TYPICAL, it's 0. Even when the value is 659, V$UNDOSTAT.TUNED_UNDORETENTION still has numbers. Dennis Yurichev has found the parameter is related to the `ktsmgd_` variable and sets certain bits that may affect Oracle's alert.log recording and possibly transaction-related functions (Ref: http://blogs.conus.info/node/3).

The above paragraph is a rundown of the child parameters affected by the parent, namely STATISTICS_LEVEL. Most of the observations can be checked by a simple lab test. For instance, to check the effect of _ULTRAFAST_LATCH_STATISTICS, save a snapshot of V$LATCH, use the ALTER SYSTEM command to set the parameter to FALSE, run some SQLs or your user application, save another snapshot of V$LATCH and find the difference. You will see the freezing of some latches' statistics that are supposed to constantly incremented. Apart from self-explanatory ones, all parameters are explained beyond basics, but practical implications are not all found. For instance, although you can verify that _DISABLE_TXN_ALERT is `ktsmgd_` in Oracle kernel by running `oradebug dumpvar sga ktsmgd_` before and after you change the parameter, how the bitmap bits in this parameter affect our DBA work still needs more research.

New in Oracle Database 11*g*

Before we move on, we do need to understand what is coming in the future in this area. Oracle Database 11.1.0.6 has 24 rows in V$STATISTICS_LEVEL compared to 17 in 10.2.0.4. The additions are listed below, along with our best guess about their changed values when going from BASIC to TYPICAL values in the STATISTICS_LEVEL parameter.

- Adaptive Thresholds Enabled: _BSLN_ADAPTIVE_THRESHOLDS_ENABLED, FALSE -> TRUE
- Automated Maintenance Tasks: _ENABLE_AUTOMATIC_MAINTENANCE, 0 -> 1
- Plan Execution Sampling: _ROWSOURCE_PROFILING_STATISTICS, FALSE -> TRUE
- SQL Monitoring: _SQLMON_THRESHOLD, 0 -> 5
- Session Wait Stack: _DISABLE_WAIT_STACK, TRUE -> FALSE
- Time Model Events: _TIMEMODEL_COLLECTION, FALSE -> TRUE
- V$IOSTAT_* statistics: _IO_STATISTICS, FALSE -> TRUE

Summary

Oracle initialization parameters can be organized in a hierarchy such that changing values of some parameters changes the values of others. Knowledge of the parameter dependency helps DBAs and performance tuning analysts better judge the impact as well as provide them with additional tools. The STATISTICS_LEVEL parameter has not been extensively studied, compared to some other parent parameters such as OPTIMIZER_FEATURES_ENABLE, and yet it has a wide effect when its value is changed. The three values for STATISTICS_LEVEL are severely insufficient when you decide to enable or disable certain features in the database based on your business need. It is

important to be familiar with the individual control parameters and use them wisely, with Oracle's approval. We hope that this article provided some background that you can use while dealing with this topic.

Appendix:

SQL(s) to find the parameter difference for the three values of STATISTICS_LEVEL:

```
column parameter format a37
column "Instance Value" format a10
set trimspool on pagesize 10000
spool typical
select a.ksppinm "Parameter", c.ksppstvl "Instance Value"
from X$KSPPI a, X$KSPPSV c where a.indx = c.indx order by 1;
spool off
alter system set STATISTICS_LEVEL=basic;
spool basic
select a.ksppinm "Parameter", c.ksppstvl "Instance Value"
from X$KSPPI a, X$KSPPSV c where a.indx = c.indx order by 1;
spool off
alter system set STATISTICS_LEVEL=all;
spool all
select a.ksppinm "Parameter", c.ksppstvl "Instance Value"
from X$KSPPI a, X$KSPPSV c where a.indx = c.indx order by 1;
spool off
host diff typical.lst basic.lst | more
host diff typical.lst all.lst | more
```

About the Author

Yong Huang is a DBA at M. D. Anderson Cancer Center. Before joining M. D. Anderson, Huang worked as an Oracle DBA or consultant at Schlumberger, Unocal Oil, Nationwide Insurance, Electronic Arts, and eBay. Other than Oracle, his interest includes UNIX and Windows internals. Find more information at: **http://yong321.freeshell.org/computer.html**.