Oracle Performance Management – A Radical Approach

Gaja Krishna Vaidyanatha
Director, Storage Management Products, Quest Software Inc.

StorageXpert for Oracle – Visit us http://www.quest.com/storage_xpert

Primary Author - Oracle Performance Tuning 101
A Minute of Silence...
Now, that sounds like fun!!!

For more information check out http://www.geekcruises.com
“It’s almost ‘duh!’ . What is common sense is not common practice.”  
- Steven Covey
Explain Plan for this Session

- The Problem
- The Effect
- Good Performance Habits
- The Method Behind the Madness
- Performance Benchmarking
- Identifying the Oracle Bottlenecks – Prong I
- Identifying the OS Bottlenecks – Prong II
- The Holy Grail of Oracle Performance Tuning
- Putting It All Together
- An Extreme Example
- Why are you tuning?
- That’s a Wrap
The Problem

- Perception – Oracle Performance Tuning is wizardry
  - Why? – Too many things to check...too many ratios to figure out...
- Erroneous and irrelevant printed, presented and spoken material correlating tuning with Oracle cache-hit ratios. Here are a few:
  - Ratios should be > 90%, some even > 99%, else performance is bad.
  - Any drop in ratios = Performance Loss
  - Larger the Oracle caches = Better System Performance.
  - Eliminate Physical I/O = Performance Gains
  - And more nonsense....
- Result - DBAs are now competing with one another to throw more memory at the various caches in the Oracle SGA
The Effect

- Arbitrary attempts to allocate more memory and endless editing of the init.ora in efforts to cure performance problems.
- Various “hunt and peck” expeditions based on “expert recommendations”.
- System performance problems start to feel like tunnels with no end in sight.
- System resource capacity wastage & misuse, lack of productivity and frustration.
Good Performance Habits

- Cache-hit-ratios are for losers [Mogens Nørgaard]
- Seek out the bottlenecks on your systems
- Cure the disease, not just the symptoms
- Tuning the resource hogs on your system will provide you the maximum benefit.
- Set tuning goals, stop tuning when you accomplish them
- Do not treat tuning as a contest or a never-ending laundry list
- Take expert recommendations with a grain of salt…no make that a bag of salt…;-).
- If you suffer from CTD, get help!!!
The Method Behind the Madness

- Seek out the “Oracle Bottlenecks” using the OWI – Prong I
- Seek out the “OS bottlenecks” when required – Prong II
- Move each prong towards the other
- When they meet – The real bottlenecks are detected and the performance problem is accurately defined
- Make calculated minimal changes as required
- Measure the effect of each change
The Method Behind the Madness

- The Oracle Prong (OWI) is comprised of `v$system_event`, `v$session_event`, `v$session_wait` and trace files generated by the `10046 event` - **Prong I**
- The OS prong is comprised of bottleneck checks for CPU, Memory, I/O and Network – **Prong II**
- The second prong is required for additional information that the first prong could not provide.
- When the bottleneck does not show up in Prong I, the use of Prong II is imminent.
- When the 2 prongs meet, you did it!
The Method Behind the Madness: Why is the OS Prong Needed?

- The OWI does not detect CPU bottlenecks
- The OWI does not detect memory bottlenecks
- The statistics are measured in centiseconds (1/100 sec.)
  - Until Oracle9i
- Even many sub-centisecond events need your attention
- I/O, Memory or CPU bottlenecks can cause many sub-centisecond events.
- Finally, it helps you find out “bottlenecks external to Oracle”
Performance Benchmarking

- Duration of measurement depends on the duration of the problem.
- Need to benchmark performance before and after any change implemented
- Set TIMED_STATISTICS=TRUE
- Run utlbstat/utlestat
  - Under $ORACLE_HOME/rdbms/admin
  - Generates a report called report.txt
Performance Benchmarking

- STATSPACK is preferred in Oracle 8i and beyond
- Check out spdoc.txt under $?$/rdbms/admin for more information
- Need help in analyzing your BSTAT/ESTAT and STATSPACK reports?
  - Check out the online analyzer at http://www.oraperf.com
- Need a performance data collector?
  - Check out Sparky at http://www.hotsos.com
  - SQLab Vision’s Stealth Collect - at http://www.quest.com
The Method Behind the Madness: The Oracle Prong (I)

- Start with `v$system_event` to determine the top wait events on your system
- Drill down to `v$session_event` which provides session-level events & `v$session_wait` which provides session-level waits
- Get all the SQL for the session in question - join `v$session_wait`, `v$session`, `v$sql`
- Get the bottlenecking segment for this SQL by joining `v$session_wait`, `dba_data_files`, `dba_extents`
The Method Behind the Madness: Using v$system_event

- Columns
  - Event
  - Total_Waits
  - Total_Timeouts
  - Time_waited – in centiseconds
  - Avg_wait – also in centiseconds
The Method Behind the Madness: Using v$session_event

- Same columns as in v$system_event with session information
- Information can change as the session metrics change
- As sessions log-off, the statistics are lost
- Remember to get the SQL executed by these sessions
The Method Behind the Madness: Using v$session_wait

- Complex view to understand
- Wait statistics are reported as they happen
- Drills down to the actual problem...
- Columns you should care about
  - sid, seq#, event, p1..p3
  - state, wait_time, seconds_in_wait
The Method Behind the Madness: Using v$session_wait

- **p1 - p3**
  - Provides details on the wait events
  - Example: For wait event “db file sequential read” - p1 is the file# and p2 is the block#
  - Example: For wait event “latch free” - p2 is the latch# related to the latch# in v$latch
The Method Behind the Madness: Understanding STATE

- **Waiting** - Currently waiting for the event
- **Waited Unknown Time** - timed_statistics is not set to TRUE, i.e., is set to FALSE
- **Waited Short Time** - Waited for an insignificant amount of time – Not really worth looking for round 1, take care of this in round 2
- **Waited Known Time** – If the resource that is waited upon is gained at a later time, the state changes from Waiting to Waited Known Time
The Method Behind the Madness: Understanding WAIT_TIME

- The value for this column is **STATE** dependent.
- If state = *(Waiting or Waited Unknown Time or Waited Short Time)* then
  
  
  **Wait_Time** = Irrelevant;

  End If;

- If state = *(Waited Known Time)* then
  
  **Wait_Time** = Actual wait time, in seconds;

  End If;
The value for this column is **STATE** dependent.

- If `state = (Waited Unknown Time or Waited Short Time or Waited Known Time)` then
  
  ```
  Seconds_In_Wait = Irrelevant;
  End If;
  ```

- If `state = (Waiting)` then
  
  ```
  Seconds_In_Wait = Actual Wait Time in Seconds;
  End If;
  ```
The Method Behind the Madness: The OS Prong (II)

- CPU Bottlenecks
- Memory Bottlenecks
- I/O Bottlenecks
The Method Behind the Madness:
Monitoring the OS: CPU

- `sar -u 5 10000`
  - `%usr`, `%sys`, `%wio`, `%idle`
  - `%wio` and `%sys` typically should be < 10-15%
  - 0% idle is OK, so long as `%wio` and `%sys` is not high and CPU run-queue < (2-3 * (# of CPUs))

- Determine CPU Run Queue – `vmstat -S 5 10000`
  (The column – “r” in the output)

- CPU Run Queue – `top – Runnable`
The Method Behind the Madness: Monitoring the OS: I/O

- `sar -d 5 10000`
  - I/O requests should be evenly balanced
  - I/O re-configuration should considered when only some devices get most of the I/O requests
  - High disk queue numbers + high service times
    - I/O contention
- Check out a state-of-the-art I/O bottleneck diagnostic tool - StorageXpert for Oracle at http://www.quest.com/storage_xpert
The Method Behind the Madness: Monitoring the O-S: Memory

- `vmstat -S 5 10000`
  - `sr` should be in and around 0
  - `swapins` and `swapouts` should be 0
  - Level of paging should also be at a minimum
- High I/O activity to the device where the swapfile or swap partition lives
- Do not go overboard on SGA sizing just to get high cache hit ratios
The Method Behind the Madness: Tools for OS Monitoring

- sar, mpstat, iostat
- vmstat
- Misc. Performance Monitors
- Sun’s Monitoring Toolkit
- Adrian’s Tool - Sun Solaris Only
- Hewlett Packard’s - MeasureWare, GlancePlus
- NT Performance Monitor
- Various freeware utilities for NT at SysInternals.com
  - http://www.sysinternals.com
The Holy Grail of Oracle Performance Tuning

/* For your own session */
alter session set timed_statistics=true; /* If not already set */
alter session set max_dump_file_size=unlimited; /* Prevent trace file truncation */

/* Set event for SQL Trace at the appropriate level
1 = Get Statistics -- Same as setting sql_trace = true
4 = Get Statistics and Bind Variable Values
8 = Get Statistics and Wait Event Information
12 = Get Statistics and Bind Variable Values, Wait Event Information */

alter session set events '10046 trace name context forever, level 12';
/* Trace session application for the duration of the problem */
/* Find trace file using the SPID in the USER_DUMP_DEST directory.
Scan the file for all lines that begin with the word WAIT. Turn off tracing completely */
alter session set SQL_TRACE=FALSE;
The Holy Grail of Oracle Performance Tuning

/* For someone else's session - Identify the session's process ID (SPID) */
select S.Username, P.Spid, S.Sid, S.Serial#
    from V$SESSION S, V$PROCESS P
    where S.PADDR = P.ADDR and S.Username like 'A%';

/* Turn on the 10046 wait event for that session's process */
oradebug setospid <SPID>
oradebug unlimit
oradebug event 10046 trace name context forever, level 12

/* Trace session application for the duration of the problem */
/* Find trace file using the SPID in the USER_DUMP_DEST directory.
   Scan the file for all lines that begin with the word WAIT. Turn off tracing
   completely */
exec dbms_system.set_sql_trace_in_session (SID, SERIAL#, FALSE);
The Method Behind the Madness: Putting it all Together

- You look at \texttt{v$system\_event}
- db file sequential reads is the main wait event
- This wait event is usually for “index reads”
- You drill down to \texttt{v$session\_wait}
- p1 is the file# and p2 is the block# waited on
- Using p1 and p2 join with \texttt{dba\_extents} and \texttt{dba\_data\_files}, the culprit segment is singled out
Joining `v$session_wait` to `v$session` to `v$sql` to get the SQL causing and experiencing the bottleneck(s)

Use the OS prong as appropriate for bottlenecks related to memory and CPU
The Method Behind the Madness: Need More?

- Read Craig Shallahammer’s paper on the subject
- Read the Anjo Kolk’s YAPP paper, study the “Wait Events” in the Oracle Reference Appendix
- Read Cary Millsap’s paper on the subject
- Read the Oracle Performance Tuning 101 (Osborne/McGraw-Hill - Oracle Press)
- www.oraperf.com - Anjo Kolk’s website and online report profiler
- www.orapub.com - Craig’s website for papers/scripts/tools
- www.hotsos.com - Cary Millsap’s website for papers/scripts/tools
- www.evdbt.com - Tim Gorman’s website for papers/scripts
- www.quest.com - Quest home page for various tools and solutions
An Extreme Example - I

- 8GB SGA (4.5GB Shared Pool)
  - Pre-tuning state – Init.ora was tuned like crazy. To attain a high library cache-hit ratio memory was periodically added (in vain). Bad response times with online queries. The system was experiencing severe parsing hiccups.
  - Bottleneck - Severe library cache latch contention
  - Cause - Lack of bind variable usage
  - Post-tuning state - On fixing the application, shared pool was shrunk to 256MB. Pre-tuning state symptoms vanished. System performance came right back.
An Extreme Example - II

- **6GB SGA (4GB Database Buffer Cache)**
  - **Pre-tuning State** - Init.ora parameters were all tuned. Cache-hit ratios were *nice and balmy* (90s). But response time was terrible. Very high CPU usage.
  - **Bottleneck** – Severe contention for the *cache buffers chains* and *cache buffers lru chain* latches. Moderate contention on *db file sequential read* and *buffer busy waits*.
  - **Cause** - Correlated sub-queries, queries forced to use indexes, lack of enough freelists on the tables with many concurrent inserts
  - **Post-tuning State** - Fixed the application, added more freelists to the relevant tables, shrunk DB buffer cache to 1GB, resurrected the system from the performance graveyard.
Why are you tuning?

- Is anyone complaining of performance?
- What are the wait events in your database?
- What are the bottlenecks on your system?
- Have you “unearthed” any performance problem?
- Are you running your life on cache-hit ratios?
- Are you suffering from CTD?
- Is it time for a drink or two?
That’s a Wrap

- Cache-hit ratios are useless numbers
- The 2-pronged approach provides you more information about system bottlenecks that any cache-hit ratio
- You need a method that provides repeated and consistent tuning success...you now know one!!!!
- Remember the 80-20 rule
- Setting specific tuning goals and ceasing the tuning effort when the goals are attained requires discipline
Gaja Krishna Vaidyanatha
Director, Storage Management Products, Quest Software Inc.
gaja@quest.com or gajav@yahoo.com
Questions and Answers
Business runs better on us