WHY USING HANGANALYZE

A “true” hang in the Oracle database can be defined as an internal deadlock or a cyclical dependency between two or more processes. When dealing with DML locks (i.e., TM), Oracle is able to detect this dependency and rollback one of the processes to break the cyclical condition. On the other hand, when this situation happens with internal kernel-level resources (e.g latches or pins), Oracle is usually unable to automatically detect and resolve the deadlock.

A severe performance problem may easily be mistaken for a hang. This usually happens when contention is so bad that it seems like the database is completely hung. Previous to HANGANALYZE (Available since Oracle realease 8.0.6), the only tool available to diagnose this type of problem was the well-known “SYSTEMSTATE dump”. A SYSTEMSTATE dump has the following disadvantages when dealing with hanging issues:

- SYSTEMSTATE dump reads the SGA in a “dirty” manner, so it may be inconsistent when the time to dump all the process is long.
- SYSTEMSTATE dump usually dumps a lot of information (most of which is not needed to determine the source of the hang), which makes difficult to determine the dependencies between processes quickly.
- SYSTEMSTATE dumps do not identify “interesting” processes on which to perform additional dumps (ERRORSTACK or PROCESS STATE).

HANGANALYZE use internal kernel calls to determine if a session is waiting for a resource, and reports the relationships between blockers and waiters. In addition, it determines which processes are “interesting” to be dumped, and may perform automatic PROCESSSTATE dumps and ERRORSTACKS on those processes, based on the level used while executing the HANGANALYZE.

Note: HANGANALYZE is not intented to replace a SYSTEMSTATE dump, but may serve as a road map to interpret a SYSTEMSTATE while diagnosing complex issues.

USING HANGANALYZE

The “HANGANALYZE” command is available since Oracle Release 8.1.6. In Oracle9i it was enhanced to provide “cluster wide” information in Real Application Cluster (RAC) environments on a single shot. The meaning of this is that it will generate information for all the sessions in the cluster regardless of the instance that issued the command.

HANGANALYZE may be executed using the following syntax:

```
ALTER SESSION SET EVENTS 'immediate trace name HANGANALYZE level <level>';
```

Or when logged in with the “SYSDBA” role,

```
ORADEBUG hanganalyze <level>
```

To perform cluster wide HANGANALYZE use the following syntax:

```
ORADEBUG setmypid
ORADEBUG setinst all
ORADEBUG -g def hanganalyze <level>
```

The <level> sets the amount of additional information that will be extracted from the processes found by HANGANALYZE (ERROSTACK dump) based on the “STATE” of the node. See further detail in the “STATE DESCRIPTION” section of this document.

The levels are defined as follows:

10	Dump all processes (IGN state)
5	Level 4 + Dump all processes involved in wait chains (NLEAF state)
4	Level 3 + Dump leaf nodes (blockers) in wait chains (LEAF,LEAF_NW,IGN_DMP state)
3	Level 2 + Dump only processes thought to be in a hang (IN_HANG state)
1-2	Only HANGANALYZE output, no process dump at all

INTERPRETING HANGANALYZE DUMPS

HANGANALYZE has several sections, which may or not be reported in the trace file. They are only generated when HANGANALYZE finds information related to the section.

Sections where session information is reported always have a header that details the information that is extracted. Previous to Oracle9i the information extracted is always consistent, while in Oracle9i, especially when getting “cluster wide” HANGANALYZE trace files, it may gather information related to the operating system only on the node where the HANGANALYZE was issued.

Examples of session “headers” :

Oracle 8.x chain header: <sid/sess_srno/proc_ptr/ospid/wait_event>
Oracle9i chain header: <cnode/sid/sess_srno/proc_ptr/ospid/wait_event> :

Where:

sid	= Session ID
sess_srno	= Serial#
proc_ptr	= Process Pointer
ospid	= OS Process Id
wait_event	= Waitevent
cnode	= Node Id (Only available since Oracle9i)

State of Nodes headers are explained in the section called: STATE OF THE NODES DESCRIPTION

The following example shows how to interpret each of the sections of a HANGANALYZE trace file:

```
*** 2002-05-13 17:02:30.937
==============
HANG ANALYSIS:
==============
```

CYCLES: This section reports the process dependencies between sessions that are in a deadlock condition. Cycles are considered “true” hangs.

```
Cycle 1 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <980/3887/0xe4214964/24065/latch free>
 -- <2518/352/0xe4216560/24574/latch free>
 -- <55/10/0xe41236a8/13751/latch free>
```

BLOCKER OF MANY SESSIONS: This section is found when a process is blocking a lot of other sessions. Usually when a process is blocking more that 10 sessions this section will appear in the trace file.

```
Found 21 objects waiting for <sid/sess_srno/proc_ptr/ospid/wait_event>
    <55/10/0xe41236a8/13751/latch free>
Found 12 objects waiting for <sid/sess_srno/proc_ptr/ospid/wait_event>
    <2098/2280/0xe42870d0/3022/db file scattered read>
Found 12 objects waiting for <sid/sess_srno/proc_ptr/ospid/wait_event>
    <1941/1783/0xe41ac9e0/462/No Wait>
Found 12 objects waiting for <sid/sess_srno/proc_ptr/ospid/wait_event>
    <980/3887/0xe4214964/24065/latch free>
```

OPEN CHAINS: This section reports sessions involved on a wait chain. A wait chains means that one session is blocking one or more other sessions.

```
Open chains found:
Chain 1 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <2/1/0xe411b0f4/12280/db file parallel write>
Chain 2 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <3/1/0xe411b410/12282/No Wait>
Chain 6 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <18/1631/0xe4243cf8/25457/db file scattered read>
 -- <229/1568/0xe422b84c/8460/buffer busy waits>
Chain 17 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <56/11/0xe4123ce0/13755/latch free>
 -- <2384/599/0xe41890dc/22488/latch free>
 -- <32/2703/0xe41fa284/25693/latch free>
```

OTHER CHAINS: It refers to chains of blockers and waiters related to other sessions identified under “open chains”, but not blocked directly by the process reported on the ″open chain″.

```
Other chains found:
Chain 676 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <20/93/0xe411d644/13597/latch free>
Chain 677 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <27/1201/0xe41d3188/15809/latch free>
Chain 678 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <36/1532/0xe428be8c/4232/latch free>
 -- <706/1216/0xe4121aac/23317/latch free>
Chain 679 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <43/12/0xe4122d54/13745/latch free>
Chain 680 : <sid/sess_srno/proc_ptr/ospid/wait_event> :
    <80/2/0xe41290d4/13811/library cache pin>
 -- <1919/1134/0xe421fdbc/3343/enqueue>
```

Additional DUMP detail: HANGANALYZE generates errorstack dumps for individual processes based on: the LEVEL used when the command is issued, and the STATE of the session. This part of the HANGANALYZE trace indicates which processes will be dumped if higher levels are used. It also indicates how many nodes will be dumped so you can measure the impact to the system (if too many errorstack dumps occur simultaneously, the disks serving the trace destination may become saturated). The “STATE DESCRIPTION” section of this document further explains the meaning of the states.

```
Extra information that will be dumped at higher levels:
[level  4] :  23 node dumps -- [LEAF] [LEAF_NW] [IGN_DMP]
[level  5] :  36 node dumps -- [NLEAF]
[level 10] : 130 node dumps -- [IGN]
```

STATE OF NODES: This section might be considered as the main section of the report. It shows all the sessions connected when the HANGANALYZE trace file was generated. This section essentially describes a dependency graph between nodes(known as an “adjacency list”), where each session is considered a node, and each node may have an “adjacent” node or a “predecessor” node related to it. Depending on the state of the node it may be a blocker, a waiter or a hung node. The “STATE DESCRIPTION” section of this document explains further the meaning of the states.

```
State of nodes
([nodenum]/sid/sess_srno/session/state/start/finish/[adjlist]/predecessor):
[0]/1/1/0xa6f8b0/IGN/1/2//none
```

```
[1]/2/1/0xa70230/IGN/3/4//none
[3]/4/1/0xa71530/IGN/5/6//none
[4]/5/1/0xa71eb0/IGN/7/8//none
[5]/6/1/0xa72830/IGN/9/10//none
[6]/7/1/0xa731b0/IGN/11/12//none
[7]/8/1/0xa73b30/IGN/13/14//none
[8]/9/1/0xa744b0/IGN_DMP/15/18/[130]/none
[9]/10/1/0xa74e30/IGN/19/20//none
[10]/11/4202/0xa757b0/IGN/21/22/[130]/none
[11]/12/1196/0xa76130/NLEAF/23/28/[49]/none
[12]/13/1/0xa76ab0/IGN/29/30/[130]/none
[37]/38/37/0xa85830/NLEAF/73/76/[50]/46
[46]/47/15/0xa8adb0/NLEAF/91/92/[37][50]/none
====================
END OF HANG ANALYSIS
====================
```

STATE OF THE NODES DESCRIPTION

As described before, HANGANALYZE use the model of “Adjacency Lists” to report the sessions found when the HANGANALYZE command was issued. In Graph Theory, an adjacency list represents a list of nodes that are connected to a given node. For each node, a linked list of nodes connected to it can be set up to represent the neighbor node connected. To be able to interpret a HANGANALYZE trace file it is required to relate the “STATE” of the node with the “adjacent” node to it.
A typical entry in the state of the nodes section will be as follow:

Oracle 8.x :
[nodenum]/sid/sess_srno/session/state/start/finish/[adjlist]/predecessor
Oracle9i:
[nodenum]/cnode/sid/sess_srno/session/ospid/state/start/finish/[adjlist]/predecessor

Where:

Nodenum	= This is sequential number used by HANGANALYZE to identify each session
sid	= Session ID
sess_srno	= Serial#
ospid	= OS Process Id
state	= State of the node
adjlist	= adjacent node (Usually represents a blocker node)

```
  predecessor = predecessor node (Usually represents a waiter node)
  cnode              = Node number (Only available since Oracle9i)
```

The following describes the important states to be considered:

IN_HANG: This might be considered as the most critical STATE. Basically a node in this state is involved in a deadlock, or is hung. Usually there will be another "adjacent node" in the same status. For example:

```
  [nodenum]/cnode/sid/sess_srno/session/ospid/state/start/finish/[adj
list]/predecessor
  [16]/0/17/154/0x24617be0/26800/IN_HANG/29/32/[185]/19
  [185]/1/16/4966/0x24617270//IN_HANG/30/31/[16]/16
```

In this example the node [16] is waiting for node [185], and the other way around; this is a cyclical condition (deadlock).

LEAF and LEAF_NW: Leaf nodes are considered on top of the wait chain (usually blockers). They are considered "Blockers" when there is another session waiting. This can be easily identified using the "predecesor" field. If there is a node referenced in the 'prdecessor' field, the node is considered as "blocker", otherwise it is considered as a "slow" session waiting for some resource.
The difference between LEAF and LEAF_NW is that LEAF nodes are not waiting for something, while LEAF_NW are not waiting or may be using the CPU. A typical representation of these nodes when they are considered blockers is:

```
  [  nodenum]/cnode/sid/sess_srno/session/ospid/state/start/finish/[a
djlist]/predecessor
  [16]/0/17/154/0x24617be0/26800/LEAF/29/30//19
  [19]/0/20/13/0x24619830/26791/NLEAF/33/34/[16]/186
```

In this example, node [16] is blocking node [19]. Notice that node [16] has node [19] in the predecessor field.
Also notice that node [19] has node [16] in the adjacent list.

NLEAF : These sessions are usually considered as "stuck" sessions. It means that there is another session holding a resource needed by the session in this state. By using the adjlist, you can determine which node is the blocker of this process. When many sessions are found in this state, it is likely the database is experiencing a performance problem rather than a hang problem.

IGN and IGN_DMP : Sessions in this state are usually considered as IDLE sessions, unless they reference a node in the " adjlist" field. In this case, the node is waiting for another node, so it will be considered as a 'stuck' session as well.
Extending the previous example,

```
  [nodenum]/cnode/sid/sess_srno/session/ospid/state/start/finish/[adjlist]/predecessor
  [16]/0/17/154/0x24617be0/26800/LEAF/29/30//19
  [19]/0/20/13/0x24619830/26791/NLEAF/33/34/[16]/186
  [189]/1/20/36/0x24619830//IGN/95/96/[19]/none
  [176]/1/7/1/0x24611d80//IGN/75/76//none
```

You may notice that node [189] is waiting for node [19] which in turn is waiting for node [16], while node [176] is an IDLE session . This maybe the case when a session has a DML lock but never finished the transaction.

In Oracle9i, two new states were introduced to differentiate between LEAF nodes that have other nodes waiting behind them (i.e., LEAF nodes that are blockers) vs. LEAF nodes that are not affecting other nodes.

SINGLE_NODE and SINGLE_NODE_NW:
This can be described the same as LEAF and LEAF_NW, except that they don't have processes depending on them.

LEVEL USAGE RECOMMENDATIONS

It is advisable not to use levels higher than 3 due to the potentially large number of trace files that may be produced (and could overwhelm the I/O subsystem). Since HANGANALYZE will be mostly used to diagnose "true hangs" , a level 3 will dump the processes that are involved in a the hang condition - this usually involves fewer than 4 processes.
Level 4 may be used with caution when dealing with performance scenarios where sessions are being stuck for a long period of time, and no other information can be gathered. Be carefull that a HANGANALYZE level 4 will dump all the processes in the following states:
[LEAF] [LEAF_NW] [IGN_DMP]. Before doing so, perform a HANGANALYZE level 3, and check the section: "Extra information that will be dumped at higher

levels”. This section will tell you exactly how many sessions will be dumped at each level. It is diffucult to know how many dumps are an acceptable number to keep from saturating the disks due to the differences in disk implementations (5 to 10 is probably OK for most systems; larger systems may be able to handle 20 to 40 dumps).

Level 5 and 10 are typicaly useless and generate a huge amount of trace files that may severely impact the performance of the database.

HANGANALYZE AND OTHER TRACING FACILITIES

When diagnosing problems using HANGANALYZE it is recommended also to generate a SYSTEMSTATE dump level 2 to be able to see all the session details. HANGANALYZE together with the SYSTEMSTATE dump will help Oracle Support to understand better the status of the database when the problem ocurred and get a detailed view of each of the process connected to the database.

RELATED DOCUMENTS

Note:175006.1 : Steps to generate HANGANALYZE trace files