|  |  |  |
| --- | --- | --- |
| **Understanding and Troubleshooting Instance Load Balancing (Doc ID 263599.1)** | [To Bottom](https://support.oracle.com/epmos/faces/SearchDocDisplay?_adf.ctrl-state=pyrp7gr3i_9&_afrLoop=335957415329485%20\o%20To%20Bottom) |  |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **In this Document**   |  |  | | --- | --- | |  | [Purpose](https://support.oracle.com/epmos/faces/SearchDocDisplay?_adf.ctrl-state=pyrp7gr3i_9&_afrLoop=335957415329485%20\l%20PURPOSE) | |  | [Scope](https://support.oracle.com/epmos/faces/SearchDocDisplay?_adf.ctrl-state=pyrp7gr3i_9&_afrLoop=335957415329485%20\l%20SCOPE) |  |  |  | | --- | --- | |  | [Details](https://support.oracle.com/epmos/faces/SearchDocDisplay?_adf.ctrl-state=pyrp7gr3i_9&_afrLoop=335957415329485%20\l%20BODYTEXT) | |  | [References](https://support.oracle.com/epmos/faces/SearchDocDisplay?_adf.ctrl-state=pyrp7gr3i_9&_afrLoop=335957415329485%20\l%20REF) |     **APPLIES TO:**  Oracle Net Services  Information in this document applies to any platform.  Checked for relevance on 26-nov-2012  **PURPOSE**  To provide an understanding of instance load balancing architecture and to show how to further investigate issues in this area.  **SCOPE**  This document is intended for use by DBA's and support analysts attempting to understand and troubleshoot Load Balancing issues.  **DETAILS**  **Understanding Instance Load Balancing**  With server-side connection load balancing, the listener routes incoming client connections to instances on the least loaded nodes and then to the least loaded instance on that node. The listener determines connection distribution based on profile statistics that are dynamically updated by PMON. The higher the node load the more frequent PMON updates the load profile. Updates occur in as little as one minute on heavily loaded nodes and may take up to ten minutes on lightly loaded nodes.  Listener logs and PMON traces both offer valuable information when evaluating instance load balancing. We will review both areas in detail in this document.  Please note that the examples that follow utilize the Unix "grep" command to locate strings in the listener logs, Windows systems have a similar facility, "find", which can be used in the same manner.  For example:  **Unix**  grep establish listener\_A.log | wc -l  600  **Windows**  D:\oracle\ora92\network\log> type listener.log | find /c "establish"  600  **The Listener Log**  The listener log file contains audit trail information that allows you to gather and analyze network usage statistics, as well as information indicating the following:   * A client connection request * A start, stop, status, reload or service command issued by the LSNRCTL   Control Utility   * Service registers and service updates by PMON   For more detailed information on these and other listener log entry types please see "Analyzing Listener Log Files" in the "Oracle Net Services Administrators Guide"  **PMON entries in the listener log**  When an instance is started, PMON will register with listeners identified in the LOCAL\_LISTENER and REMOTE\_LISTENER parameter found in either the init.ora or spfile.ora. Once registered, PMON updates the listener with profile statistics that allow the listener to route incoming connections to the least  loaded instances.  PMON initial instance registrations (at instance startup) can be identified in the listener log by the string "service\_register".  A Unix grep (or Windows find) will extract the following entries for review.  $ grep service\_register listener1.log  13-FEB-2004 11:01:10 \* service\_register \* instance1 \* 0  13-FEB-2004 11:01:34 \* service\_register \* instance2 \* 0  13-FEB-2004 11:01:41 \* service\_register \* instance3 \* 0  PMON service updates (which provide the listener with instance load information) can be identified in the listener log by the string "service\_update".  A Unix grep (or Windows find) will extract the following entries for review.  $ grep service\_update listener1.log  13-FEB-2004 11:20:50 \* service\_update \* instance1 \* 0  13-FEB-2004 11:21:17 \* service\_update \* instance2 \* 0  13-FEB-2004 11:21:32 \* service\_update \* instance3 \* 0  **Understanding listener log connection entries with load balanced instances.**  Connection entries in an instance load balancing environment are unique in that multiple listener logs may update simultaneously for a single connection. Before we can use the data in the log we must understand how these entries are determined.  Consider the following configuration (Fig. 1)  Instance\_A is locally registered with Listener\_A and remotely registered with Listener\_B. Instance\_B is registered locally with Listener\_B and remotely with Listener\_A.      **Local instance connections (same node as listener)**  In figure 2 below, Client connection #1 goes to Listener\_A. Based on load information provided by PMON, Listener\_A routes the incoming client to the locally registered instance, Instance\_A.        This type of connection updates Listener\_A's log with a single entry.  **Listener\_A.log**  13-FEB-2004 11:05:02 \* (CONNECT\_DATA=(SERVER=DEDICATED)(SERVICE\_NAME=prod)  (CID=(PROGRAM=)(HOST=clienthost)(USER=oracle))) \*  (ADDRESS=(PROTOCOL=tcp)(HOST=10.10.10.10)(PORT=20108)) \* establish \* prod \* 0  **Remote instance connections (redirected connections from non-local listeners)**  In figure 3 below, client connection #2 goes to Listener\_A. With updated load information provided by PMON, Listener\_A routes the client to the remotely registered instance, Instance\_B.      This type of connection places an entry in BOTH listener\_A and listener\_B's logs, however, the listener\_B log entry has target instance information that identifies it as a redirected connection. Listener\_A's entry would not have instance information (same as Listener\_A.log example above).  \*\*Please note that when using the Shared Server model, connections may be routed to dispatchers on remote nodes without being logged in the listener.log. This analysis method is only applicable in the Dedicated server model.  **Listener\_B.log with instance information**  13-FEB-2004 11:05:02 \* (CONNECT\_DATA=(SERVER=DEDICATED)(SERVICE\_NAME=prod)  (CID=(PROGRAM=)(HOST=clienthost)(USER=oracle))(INSTANCE\_NAME=Instance\_B)) \*  (ADDRESS=(PROTOCOL=tcp)(HOST=10.10.10.10)(PORT=20110)) \* establish \* prod \* 0  Notice that line 2 of the Listener\_B.log entry has the additional string "INSTANCE\_NAME=Instance\_B" identifying it as a connection that was directed from a different listener.  Now that we understand the types of entries that are found within a listener log in an instance load balanced environment, we can use the data in the log to determine both connection load balancing and instance load balancing statistics.  **Interpreting listener log data**  While listener logs show redirected incoming connections (ie. INSTANCE\_NAME=xxx) they do not indicate from where the connections came. This makes it difficult to accurately analyze the logs of more than 2 listeners. If the environment can be limited to 2 listeners, then the logs can then be used to determine a number of connection statistics (since the redirecting listener must be the other listener). We can show how in the following examples:  **Testcase:**  Using a similar configuration to that in Figure 1, instances are registered locally and are also cross registered to listeners on each node.  With all instances and listeners stopped, the listener logs are cleared so we can start with fresh data. The listeners and instances are then started and a client script initiates constant connections/queries for 30  minutes. At the end of the testcase the logs are reviewed.  **Incoming connection entries**  Counting the occurances of the string "establish" in the listener logs will provide the total number of connections that the listener handled. This includes redirected connections from other nodes. Counting the number of occurances of the string "INSTANCE\_" shows the number of connections that were  redirected. The result of these counts are labled (C,D,E, & F) for use in later examples**.**  **Listener\_A**  grep establish listener\_A.log | wc -l ==> 800(C)  grep INSTANCE\_ listener\_A.log | wc -l ==> 300(D)  **Listener\_B**  grep establish listener\_B.log | wc -l ==> 600(E)  grep INSTANCE\_ listener\_B.log | wc -l ==> 130(F)  **Client load balancing breakdown**  To determine the number of client connections that were not redirected we subtract the total redirected from the total number of established connections. The initial client connection values are labeled (G & H) for use in later examples.  (total establish) - (redirected) = direct client connections  **Listener\_A**: 800(C) - 300(D) = 500(G)  Node\_A had 500 direct client connections  **Listener\_B**: 600(E) - 130(F) = 470(H)  Node B had 470 direct client connections  Client side connection load balancing results = 500 Listener\_A / 470 Listener\_B  This ratio of conections is an accurate representation of client side load balancing as the CSLB mechanism uses a randomization algorithm to determine connection routing.  **Redirection breakdown**  By taking the number of direct client connections and subtracting redirected connections we can determine how many connections the listener directed to a local instance. The resulting values are labeled (I & J) for use in later examples.  (direct client connections) - (redirected) = locally directed connections  **Node\_A**: 500(G) - 130(F) = 370(I)  **Node\_B**: 470(H) - 300(D) = 170(J)  Listener\_A directed 370 connections to the local instance  Instance\_A recieved 130 redirected connections  Listener\_B directed 170 connections to the local instance  Instance\_B recieved 300 redirected connections  **Connection distribution**  (local connections) + (incoming redirected connections) = instance distribution  **Node\_A**: 370(I) + 300(D) = 670  **Node\_B**: 170(J) + 130(F) = 300  Instance (Server side) load balance distribution  Instance\_A = 670  Instance\_B = 300  In conclusion, we had a relatively even distribution of connections from the clients to the listeners (500/470) but we had a percieved imbalance of connections routed to the instances (670/300). Why? There are a number of factors that can effect the distribution of connections with instance load  balancing.  **Call storms**  Remember that the shortest interval that PMON updates the listener is one minute. If a significant number of connections occurs between dynamic updates, connections will be routed to the last known least loaded node. This routing will continue until the next update and can sway the results.  (Call storms are better handled in Oracle 10.2, see "Oracle 10.2 changes" below.)  **Overall load on the system (in addition to Oracle)**  To determine load, PMON uses the 1 minute average obtained from the OS function kstat() and queries the system every 30 seconds. An overall high system load will effect PMON's update data. (In Unix, the uptime command can give an idea of the load information PMON is retrieving.)  **Length of testcase**  A short testcase and call storm combination has the most potential to show swayed results. Stopping the testcase before an important update will also sway the results. As a rule, the longer the testcase the more the numbers will even out, even with call storms.  Using PMON traces to check load balance behavior (Oracle 9 & 10.1)  While understanding connect behavior and the listener logs is important, the best way to determine whether or not instance load balancing is functioning properly is through PMON traces.  To trace PMON load information, set the following event in each instance's init.ora or spfile:  event="10257 trace name context forever, level 16"  After restarting each instance, a PMON trace file in the background\_dump\_dest will be generated and PMON load information should be available. At the beginning of the trace file, you will see the initial registration:  \*\*\* SESSION ID:(1.1) 2004-02-13 11:01:38.708  load balancing parameters set to [300, 192, 64, 3, 10, 10, 20480]  kmmgrg: add registrar (ADDRESS=(PROTOCOL=TCP)(HOST=racnodeA)(PORT=1521))  kmmlrl: initial registration  kmmlrl: 6 processes  kmmlrl: node max load 20480  kmmlrl: instance max load 1655  kmmlrl: block  kmmgrg: add registrar (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.1.1.1)(PORT=1521)))  kmmgrg: add registrar (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.1.1.2)(PORT=1521)))  kmmlrl: nsgr register returned 0  kmmlrl: update for unblock: 0 0 1 1655 0 20480  kmmlrl: 16 processes  kmmlrl: instance load 1  kmmlrl: unblock  kmmlrl: nsgr update returned 0  This is showing the number of processes on the instance and the current instance load.  After the initial registration, On an idle system, you may see messages like this:  \*\*\* 2004-02-13 08:42:14.090  kmmlrl: update for time delta: 60212  kmmlrl: 13 processes  kmmlrl: nsgr update returned 0  kmmlrl: nsgr register returned 0  When PMON is updating the listener with new load information, you will see messages like this:  \*\*\* 2004-02-13 09:00:38.382  kmmlrl: update for node load delta: 5092 4015 5120  kmmlrl: 371 processes  kmmlrl: node load 25  kmmlrl: instance load 356  kmmlrl: nsgr update returned 0  kmmlrl: nsgr register returned 0  \*\*\* 2004-02-13 09:01:40.039  kmmlrl: update for node load delta: 5053 5092 5120  kmmlrl: 355 processes  kmmlrl: node load 33  kmmlrl: instance load 340  kmmlrl: nsgr update returned 0  kmmlrl: nsgr register returned 0  This information will show why connections are being routed to certain instances.  'Processes' and 'instance load' give an indication of the number of connections that have been made to the instance.  'Node load' gives an indication of how busy the machine is. By default, 'node load' is the primary factor for determining where new connections will go. For example, if you have a 2-node RAC cluster and instance 1 has a node load of '25' and instance 2 has a node load of '50', new connections will be routed to  instance 1. You can use PMON tracing via Event 10257 at level 16 to confirm that connections are being routed based on 'node load'.  To get a graphical representation of load balancing, you can graph the 'processes', 'instance load', and 'node load' at time intervals with spreadsheet software such as MS Excel. An example format would be:  Instance Time        Processes Node\_Load Instance\_Load  1        8:55:27 AM  305       46        290  1        8:56:29 AM  204       38        189  1        8:57:31 AM  176       35        161  1        8:58:34 AM  246       23        231  1        8:59:36 AM  345       14        330  2        8:55:06 AM  133       15        118  2        8:56:16 AM  234       24        219  2        8:57:18 AM  318       38        303  2        8:58:20 AM  274       40        259  2        8:59:31 AM  222       37        207  In this example, from 8:55 to 8:57, new connections will go to instance 2 because it's node load is lower at that time. Also notice that during this time, the number of processes and instance load are increasing on instance 2. From 8:57 to 8:59, new connections will go to instance 1 because it's node load is lower at that time. Also notice that at this time, the number of processes and instance load are increasing on instance 1. This example shows node balancing working as designed.  **Oracle 10.2 changes**  In Oracle versions 10.2 and higher the listener uses a local value "lbscore" to determine the node and instance load. Lbscore is made up of two dynamic values "goodness" and "delta" that are continuously updated by pmon. When an instance is started PMON will contact the listener and provide starting values for goodness and delta...          Listener Lbscore = Goodness from PMON          Listener Delta = Delta from PMON  To account for changes in load between PMON updates the listener will increment the lbscore after each new incoming connection...          Listener Lbscore = Listener Lbscore(previous) + Listener Delta  With each periodic load update by PMON  the Lbscore and Delta are updated with current data.          Lbscore = Goodness (Received from PMON update)          Delta = New Delta (Received from PMON update)  Goodness, Delta, and Lbscore values can be observed in listener traces. Listener tracing is enabled by setting TRACE\_LEVEL\_LISTENER = 16 in the listener.ora or by dynamically enabling tracing using lsnrctl.  Trace examples:  Service registration  nsglgrDoRegister: Creating new service: "[RAC1.us.oracle.com](http://rac1.us.oracle.com/)".  nsglgrDoRegister: service:[RAC1.us.oracle.com](http://rac1.us.oracle.com/) flag:2 goodness:0 delta:1  Lbscore  nsglb: instance:repos6 new lbscore:50  Goodness and Delta  nsglgrDoRegister: service:[RAC1.us.oracle.com](http://rac1.us.oracle.com/) what:4 value:135  nsglgrDoRegister: service:[RAC1.us.oracle.com](http://rac1.us.oracle.com/) what:2 value:5800  what:4 is the Delta value  what:2 is the Goodness value    **Troubleshooting**  A significant amount of data needs to be carefully collected to properly overview instance registration behavior. Be sure to start with a clean slate prior to gathering data.  **Please note you should not use the default service for load balancing.**  See the following from information taken from the online documentation, chapter 6 Introduction to Workload Management from the Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide  The features discussed in this chapter do not work with the default database service. You must create cluster managed services to take advantage of these features. You can only manage services that you create. Any service created by the database server will be managed by the database server  <http://download.oracle.com/docs/cd/B19306_01/rac.102/b14197/hafeats.htm#BABBBGJB>  **1. Confirm the service information being used:**  Find the service ID being used    sqlplus /nolog  SQL> Connect / as sysdba  SQL> select name,service\_id from dba\_services where name = 'XXXX';  "XXXX" is the SERVICE\_NAME value your connections are using.  If no value returned, default service being used, goto step 2.  Take the ID returned and add to the following SQL  SQL>col name format a15  col failover\_method format a11 heading 'METHOD'  col failover\_type format a10 heading 'TYPE'  col failover\_retries format 9999999 heading 'RETRIES'  col goal format a10  col clb\_goal format a8  col AQ\_HA\_NOTIFICATIONS format a5 heading 'AQNOT'  SQL>select name, failover\_method, failover\_type, failover\_retries,goal, clb\_goal,aq\_ha\_notifications  from dba\_services where service\_id = X  Check to see if changing the CLB\_GOAL effects load balancing  [Note 552609.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=263599.1&id=552609.1) Load Balancing Advisory and Connection Load Balancing  **2**.**Backup and zero listener logs**:    $ cp listener.log listener.log.backup  $ cat /dev/null > listener.log  **3**.**CPU information utilities (gather at intervals**):  **A**. $ uptime  2:10pm up 180 days, 15:26, 18 users, load average: 1.14, 0.33, 0.16  Or use  **B**. OS Watcher. See [Note 301137.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=263599.1&id=301137.1) OS Watcher User Guide  **4.PMON tracing**:  Enable PMON trace via event for all nodes.  Find the process ID of PMON process  $ ps -ef | grep pmon  Then to enable event 10257  sqlplus /nolog  SQL> connect / as sysdba  SQL> oradebug setospid <PID\_of\_PMON>  SQL> oradebug Event 10257 trace name context forever, level 16  Following command will confirm the PMON trace filename and location  SQL> oradebug tracefile\_name  When ready to disable PMON  SQL> oradebug Event 10257 trace name context OFF;  Sometimes PMON tracing will not enable, thus will need to be set in spfile and the instances restarted.  Set the following event in each instance's init.ora or spfile:  ALTER SYSTEM SET event = '<event name> <action> [,<event name> <action>]...'COMMENT = '<comment>' SCOPE=SPFILE;  SQL> Alter system set event="10257 trace name context forever, level 16" scope=spfile;  After restarting each instance, a PMON trace file in the background\_dump\_dest will be generated.  SQL> Alter system set event="10257 trace name context forever off" scope=spfile;  **5.TNS listener trace**  Add to LISTENER.ORA file  TRACE\_LEVEL\_<listenername> = 16  TRACE\_FILE\_<listenername> = <FILE NAME>  TRACE\_DIRECTORY\_<listenername> = <DIRECTORY>  TRACE\_TIMESTAMP\_<listenername> = ON  TRACE\_FILELEN\_<listenername> = <SIZE in KB>  TRACE\_FILENO\_<listenername> = <NUMBER>  Re-start the listener. Cyclic tracing will ensure your can control the size and amount of trace produced.  The TRACE\_FILELEN parameter is for the size of a trace file.  The TRACE\_FILENO parameter is the number of traces per process.  Do this for all listeners and ensure if size used for cyclic is large enough not to be overwritten.  **6.Count of sessions**  Run a count of the sessions, with timestamp for all nodes, at the same time.  SQL> SELECT inst\_id, TYPE, service\_name, Count(\*) FROM gv$session GROUP BY inst\_id, TYPE,service\_name;  **7. If connections from clients using JDBC Thin:**  If new connections redirect to the different node using jdbc driver (e.g. jdbc thin), we cannot judge whether the redirection occurred or not from listener.log. The same message is logged in listener.log even if the connection is redirected to different node.  In this case, command lsnrctl services <listenername> contains useful information to identify redirections.  **-- InstanceA --**  Service "dbservice" has 2 instance(s).  Instance "instA", status READY, has 1 handler(s) for this service...  Handler(s):  "DEDICATED" established:39348 refused:0 state:ready <--(\*1)  LOCAL SERVER  Instance "instB", status READY, has 1 handler(s) for this service...  Handler(s):  "DEDICATED" established:26129 refused:0 state:ready <--(\*2)  REMOTE SERVER  (ADDRESS=(PROTOCOL=TCP)(HOST=ins02-vip)(PORT=1521))  **-- InstanceB --**  Service "dbservice" has 2 instance(s).  Instance "instA", status READY, has 1 handler(s) for this service...  Handler(s):  "DEDICATED" established:23445 refused:0 state:ready <--(\*3)  REMOTE SERVER  (ADDRESS=(PROTOCOL=TCP)(HOST=ins01-vip)(PORT=1521))  Instance "instB", status READY, has 1 handler(s) for this service...  Handler(s):  "DEDICATED" established:44428 refused:0 state:ready <--(\*4)  LOCAL SERVER  (\*1) : # of total connections on instA = (directly connections) + (redirected connections)  (\*2) : # of redirected connections to instB  (\*3) : # of redirected connections to instA  (\*4) : # of total connections on instB = (directly connections) + (redirected connections)  The number of local vs. redirected connections can then be calculated as follows.  # of local connections on instA = (\*1) - (\*3)  # of local connections on instB = (\*4) - (\*2)  All the diagnostics need to comes from the same time period and be small as possible.  Diagnostics needs to cover around 15-20 minutes of the problem.  Please confirm time load balancing problem was first noted.  Steps need to be crossed referenced via timestamp, please ensure O/S information  is gathering around one minute intervals and the timestamp is shown.  In Summary the recommended steps for capturing a problem are :  Step 1: Set the listener tracing on.  Step 2: Starting gathering O/S information and session data  If connections are JDBC thin, ensure listener service output gathered.  Then when load balancing problem spotted, note the time and ...  Step 3: Generate fresh listener log files and enable PMON tracing    **Known Bugs**  [Bug 6613950](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=263599.1&id=6613950%20\t%20_blank) RAC SERVERSIDE LOAD BALANCING NOT WORKING WHEN ONE SERVER IS HEAVILY LOADED  Details:Issue was due to differant number of CPU's on the node, Oracle was working as expected but code changed to ensure differances  like this taken into account for load balancing.  Fixed-Releases: 10.2.0.5 & 11.1.0.7 & 11.2  [Bug 5520002](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=263599.1&id=5520002%20\t%20_blank) RAC CLUSTER WITH SERVER SIDE LOAD BALANCING ALL CONNECTIONS ROUTED TO ONE NODE  Marked duplicate of [Bug 5755010](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=263599.1&id=5755010%20\t%20_blank) Listener registration never completes.  Details: Listener registration does not complete with some listeners even though there is no indication of registration failure.  Fixed-Releases: 10.2.0.4 & 11.1.0.6  [Bug 5593693](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=263599.1&id=5593693%20\t%20_blank) Connection load balancing does not work if db\_domain is set  Details:Connection load balancing may not work for a service if the network\_name is unqualified(may be seen iv$active\_services) and db\_domain is non-null.  Fixed-Releases: 10.2.0.4 & 11.1.0.6  [Bug 6083037](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=263599.1&id=6083037%20\t%20_blank) LOAD BALANCING PROBLEMS ON CRITICAL ODE DATABASE  Details:With serverside load balancing configured,the listeners might route all requests to the busiest node(s).  Workaround: Restart the listener on the lightly loaded node(s) but that may not fix the problem for very long.  Fixed-Releases: 10.2.0.4 and 11.1.07  [Bug 6442431](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=263599.1&id=6442431%20\t%20_blank)A node may stop receiving connections with RAC load balancing  Details: In a RAC cluster using server side load balancing, one of the nodes may stop receiving any new connections even though it is idle.  Fixed-Releases: 10.2.0.4 & 11.1.0.7 & 11.2 |