Table of Contents

CORAL fault tolerance to database and network glitches

Under construction!

This page is meant to document the status and plans for the CORAL handling of database and network glitches. This is mainly about the interaction of CORAL with Oracle through OCI and the various steps involved, within CORAL as well as inside Oracle itself, to ensure high availability connections to the CORAL client applications. As CORAL has to interact with (and generally complement and sometimes bypass) the HA features provided by the Oracle Transparent Application Failover (TAF) functionalities, we first document our tests with TAF.

Tests of Oracle TAF from sqlplus and CORAL applications

For additional and complementary details see also https://savannah.cern.ch/task/?25617.

These tests were done with TAF configured at TYPE=SELECT at all levels (in client and server). Within sqlplus: issue a very long SELECT (e.g. from DBA_OBJECTS) in session1, then in session2 kill session1.

- In 10g, within session1 we now get an ORA-03113. This seems a bug (we would expect the SELECT to continue transparently). However we are able to re-issue the SELECT, without manually reconnecting, because TAF seems at least to reconnect the session. (Is it to another node by the way?).
- In 11g, within session1 we now get a diffferent ORA-00028. And even if we re-issue the SELECT we still get an error (the same one?). This also seems a bug, worse than on 10g.

Note that the use of ALTER SYSTEM KILL SESSION (this is what was used in the tests above) seems to have a different effect from an OS-level kill of the server-side process (this is what we had used in previous tests and resulted always in ORA-03113 errors?).

Within CORAL, we did tests at a finer granularity, by glitching precisely before the first next() call, after the first next() call and in other places. In sqlplus this is not possible because the sqlplus application internally is doing all the prepare statement, execute and loop, and the glitch probably comes always within the loop, after a few iterations on the cursor.

- If we do something similar in CORAL as in sqlplus, i.e. in session1 execute the statement, call next() once which succeds, then glitch in session2, then the second next() in session1 gets ORA-03113 (as in sqlplus), while the third next still gets ORA-25401 (which we do not observe in sqlplus because we cannot re-iterate on the stale cursor there!).
- Anyway we get similar results also if the glitch happens before next is executed for the first time: the first next gets ORA-03113 and the second gets ORA-25401.
- This was on 10g or also on 11g? Did we try both?
- What happens if we try to re-issue the statement?

TAF configuration

Two useful notes on metalink:

- How to Configure Client Side Transparent Application Failover with Preconnect Option [ID 802434.1]
- How To Configure Server Side Transparent Application Failover [ID 460982.1]

TAF and keepalive

Note that in TAF it is the clients that probe the server, not the server that signals problems to the client. The actions are triggered by TAF when the client notices that the server has disapparead. In our tests, this is typically when we try to execute the first client action (e.g. get a cursor, increment a cursor) following a glitch.

(Is there also some probing of the server by the client at regular intervals? With which frequency? How is this related to keepalive?). A discussion of TCP keepalive for Oracle can be found in https://twiki.cern.ch/twiki/bin/view/PDBService/OCIClientHangProtection. See also the comments in https://savannah.cern.ch/bugs/?87164#comment51.

Open questions and to-do list

- Do we get sessions reconnected in CORAL by TAF (if we disable all CORAL handling of glitches)? We did not test to re-issue the SELECT from scratch in OCI (prepare it, execute it, loop on the cursor) after the glitch. **Do this test in CORAL**. We only observed that we get ORA-03113 and then ORA-25408 in the cursor loop. Re-issuing the SELECT would be like doing it in sqlplus (where it works in 10g - not in 11g). The question is essentially: do we need to change the OCI pointers and/or do we need to implement TAF callbacks in CORAL or not?
    - ♦ Related question: does sqlplus manage (in 10g only) to re-issue the SELECT statement without a manual reconnect because the OCI-based sqlplus executable internally implements TAF callbacks (which we would then need to implement also in CORAL) or is it more transparent?

- How does Oracle TAF handle read-only transactions? Are they restarted (at the same point in time) or not? Do the following test in sqlplus: create a table with 3 rows, then in session1 start a read only transaction and query the table; in session2, first insert another row and commit, then kill session1; then in session1 re-issue the select. How many rows do you get? **Do the test**.

- Is there a bug in Oracle 10g TAF? Why do we get ORA-03113 during a SELECT, even if TYPE=SELECT is specified? At least in 10g the session is reconnected and another SELECT can be issued without the need to manually reconnect. (Were these tests done with 10g or 11g client?)

- Is there a bug in Oracle 11g TAF (worse than that in 10g)? Why do we get ORA-00028 during a SELECT, and the session is not even reconnected, even if TYPE=SELECT is specified?

- We can also start with simpler tests: the tests above are SELECT tests, the glitch happens during a running SELECT. We can start to see if the glitch happens before any select but after a set transaction, or even before we set any transaction. And this can be done both in CORAL and sqlplus. (We should be able to do these tests without DBA help, using the web based session manager?)
    1. Glitch after connecting, before transaction - to do
    2. , Glitch after starting transaction, before selecting - to do
    3. Glitch while executing a SELECT - described above

CORAL handling of connection glitches

- Is there a way we can completely disable the current CORAL attempts to reconnect on connection glitches? We should add such an option (e.g. via an environment variable) to make tests easier and possible with every release without changing the code (currently we are commenting out three lines in SessionProxy in CORAL 2.3.21).

-- AndreaValassi - 17-Feb-2012

This topic: Persistency > CoralFaultTolerance
Topic revision: r1 - 2012-02-17 - AndreaValassi