Reacting to database and network instabilities in CORAL

A distributed analysis model implies an inhomogeneity in the data storage infrastructure across the different institutes and across the long LHC lifetime.

The computing infrastructure has to be easily maintainable and adaptable.

POOL is a generic hybrid store for C++ objects, metadata catalogues and collections, using streaming and relational technologies.

CDOIL provides specific software to handle the time variation and versioning of conditions data.

CORAL is a generic abstraction layer with an SQL-free API for accessing relational databases.

CORAL provides a set of C++ libraries for several database back-ends:

- local access to SQLite files;
- direct client access to Oracle and MySQL servers;
- read-only access to Oracle through the FronTier/Squid or CoralServer/CoralServerProxy server/cache system.

A connection is a "physical circuit", a pathway to a database.

A session is a user process in communication with a server process. Users establish sessions against the instance, and the instance then manages the access to the database.

**ORA-03113!!!**

During the last two years the three experiments that make use of CORAL (ATLAS, CMS and LHCb) experienced a similar issue: an Oracle error appeared during the execution of some operations against the Oracle database, even though in different circumstances.

The Oracle error found is: ORA-03113

In some cases, this triggered an infinite loop and caused an application hang.

As reported by the Oracle site:

"ORA-03113: end-of-file on communication channel
Cause: The connection between Client and Server process was broken.
Action: There was a communication error that requires further investigation."

The most likely cause of this issue is an instability of the network, leading to a temporary connection break between client and server.

**Network glitch error reports**

During the analysis of the network glitch two other bugs have been identified, both related to the deletion order of the relevant CORAL objects (e.g. Session).

The first bug was that the destructors of many CORAL objects (e.g. Query) were using a Session already deleted. A shared pointer to the Session has been defined to keep it alive longer than all other objects.

The second bug was due to a wrong deletion order for OCI handles. Internally, most OCI handles (e.g. those for queries) use the service context handle when they are de-allocated. This was fixed by ensuring that the service context is the last OCI handle to be de-allocated.

**Preliminary bug fixes**

The validity of the connection and session is checked at the beginning of all crucial instructions (whenever the OCI service context handle is needed).

If a network glitch is detected, CORAL reacts in the following way:

- If the transaction is not active yet, CORAL triggers a reconnection for any type of session;
- If a transaction is already active, CORAL triggers a reconnection only if the transaction has been started in RO read committed mode.

The reconnection procedure creates a new physical connection and starts a new user session. CORAL achieves this by refreshing all OCI handles without de-allocating and re-allocating them.