

CORAL and COOL during the LHC long shutdown

**A. Valassi¹, M. Clemencic^{2,a}, D. Dykstra^{3,b}, N. Goyal^{4,c},
A. Salnikov^{5,c}, R. Trentadue¹, M. Wache^{6,c}**

¹ IT Department, CERN, CH-1211 Geneva 23, Switzerland

² PH Department, CERN, CH-1211 Geneva 23, Switzerland

³ Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

⁴ Mody Institute of Technology and Science, Lakshmangarh-332311, India

⁵ SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

⁶ Institut für Physik, Universität Mainz, D-55099 Mainz, Germany

^a The author is a member of the LHCb Collaboration.

^b The author is a member of the CMS Collaboration.

^c The author is a member of the ATLAS Collaboration.

E-mail: andrea.valassi@cern.ch

Abstract. CORAL and COOL are two software packages used by the LHC experiments for managing detector conditions and other types of data using relational database technologies. They have been developed and maintained within the LCG Persistency Framework, a common project of the CERN IT department with ATLAS, CMS and LHCb. This presentation reports on the status of CORAL and COOL at the time of CHEP2013, covering the new features and enhancements in both packages, as well as the changes and improvements in the software process infrastructure. It also reviews the usage of the software in the experiments and the outlook for ongoing and future activities during the LHC long shutdown (LS1) and beyond.

1. Overview

The Large Hadron Collider (LHC) at CERN, the world’s largest high-energy particle accelerator, successfully concluded its first three-year running period in February 2013. The LHC has now begun its first long shutdown (LS1), a period during which major consolidation and maintenance work is being carried out across the CERN accelerators and experiments, affecting both their hardware installations and their software and computing infrastructures.

Huge amounts of data have been recorded and analysed by the four LHC experiments, and many more will be generated when the LHC resumes operation in 2015. CORAL and COOL are two software packages that have been used by the ATLAS, CMS and LHCb experiments to store and access many types of data via relational database technologies, such as the “conditions data” that record the experimental conditions at the time the LHC beam collisions occurred, as well as geometry data and detector configuration data. The largest data volumes, coming from the “event data” that record the signals left in the detectors by the particles generated in the LHC beam collisions, are generally stored in files instead.

CORAL is a generic abstraction layer with an SQL-free API for accessing relational databases, while COOL, based internally on CORAL, provides specific software to handle the time variation and versioning of conditions data. These two packages have been developed over several years through the collaboration of developers from the LHC experiments and the CERN IT department

CORAL and COOL in the LHC experiments	 ATLAS	 CMS	 LHCb
CORAL (Oracle, SQLite, XML authentication and lookup)	Conditions data (COOL) Geometry data (detector descr.) Trigger configuration data Event collections/tags	Conditions data Geometry data (detector descr.) Trigger configuration data	Conditions data (COOL)
CORAL + Frontier (Frontier/Squid)	Conditions, Geometry, Trigger (R/O access in Grid, Tier0)	Conditions, Geometry, Trigger (R/O access in Grid, HLT, Tier0)	—
CORAL Server (CoralServer/CoralServerProxy)	Conditions, Geometry, Trigger (R/O access in HLT)	—	—
COOL	Conditions data (complex payloads)	—	Conditions data (string/CLOB payloads)

Table 1. Summary of CORAL and COOL usage in ATLAS, CMS and LHCb.

within the context of the Persistency Framework [1] common project of the LHC Computing Grid (LCG). Both packages are written in C++, but also provide Python bindings. In the past, the Persistency Framework used to deliver also a third software package, POOL, which the LHC experiments had been using to store conditions data as well as raw event data and event collection metadata; as of 2013, POOL has ceased to be a common project and is now being maintained by ATLAS [1, 2], the only LHC experiment that is still using some of its components.

The CORAL and COOL functionalities, software process and collaborations with other LCG projects have been previously described in the proceedings of the CHEP2012 conference [1, 3] and in the original references cited therein. The goal of this paper is to focus on the evolution since that time and to provide an updated summary of their usage in the LHC experiments and of the outlook for the future. The results achieved since CHEP2012 and work in progress in several different areas are briefly described in the next three sections, approximately covering common infrastructure issues and other areas of work more specific to CORAL and COOL.

Table 1 summarizes the current usage patterns of CORAL and COOL in the LHC experiments, which have not changed significantly with respect to those at the time of CHEP2012 [1]. It is worth mentioning, however, that a new mechanism for storing event indexes and collections using Hadoop is presently being developed within ATLAS [4], which may eventually replace the existing infrastructure for storing event tags based on CORAL. Also in ATLAS, the master copy of the conditions database is still hosted in an Oracle server at CERN, but its replication via the Oracle Streams technology is now limited to only four Tier1 sites [5], as the use of Frontier has become even more widespread. Finally, SQLite has gradually become the main deployment technology for the LHCb conditions database, while the use of Oracle has been progressively restricted to only a few very specific use cases [6].

2. Software process and infrastructure

Software development of CORAL and COOL continues to follow the well established release process described in Ref. [1]. Regular production releases are prepared whenever one LHC experiment demands it, leading to one release every 6 weeks on average. The software is supported on many production platforms, including several flavours of Linux (SLC5, SLC6) and MacOSX (10.6 Snow Leopard), using one or more compilers on each O/S. On SLC6, for instance, the reference builds are presently those using the latest gcc4.8 compiler with c++11 enabled; however, no c++11 extensions to the C++ language are used yet in CORAL or COOL, because another platform using the gcc4.6 compiler with c++11 disabled is still supported. The latest software versions recently released are CORAL 2.3.28 and COOL 2.8.19, which are included in the LCG66 configuration based on ROOT 5.34.10.

To improve software quality and speed up the early adoption of new external software versions, automatic builds and tests of CORAL and COOL are performed every night on all production

platforms, as well as on a few test platforms using new compilers and build options (including clang3.3 and icc13 on SLC6). An example of this continuous integration process concerns the ongoing tests of the beta version of the next major ROOT6 release: this is presently one of the most active areas of development for COOL, because the mechanism used to provide its Python bindings (the PyCool package) is heavily based on ROOT. More particularly, PyCool is based on ROOT’s own Python bindings (the PyROOT package) and on its C++ reflection functionalities, both of which have significantly changed from ROOT5 to ROOT6 with the replacement of CINT by cling as ROOT’s embedded C++ interpreter [7].

Two recent enhancements have further improved software quality assurance for CORAL and COOL. First, the two packages have been integrated into the CERN infrastructure for static code analysis using the Coverity tool [8], thanks to which a few issues in the code have been spotted and resolved. Second, several tools for memory checking and CPU/elapsed time profiling have been integrated into the test suites of CORAL and COOL, including valgrind [9], igprof [10] and gperftools [11]. The more systematic use of valgrind for memory checking, in particular, has made it possible to identify and fix several memory leaks, while the use of gperftools for elapsed time profiling has allowed the identification of a few inefficiencies in client-server communication via CORAL (one such issue being the overhead from the explicit probing of client connections to Oracle servers used in the CORAL handling of network glitches described in Ref. [3]).

On the infrastructure side, it should finally be noted that the CORAL and COOL code repositories (as well as the POOL repository, purely for “data preservation” purposes) were recently moved from CVS to SVN. More work is also expected in the near future to replace CMT by cmake in the software build infrastructure, to migrate active and past issues from the savannah tracking tool to Jira, as well as to migrate from quattor to Puppet the CORAL and COOL dedicated test nodes and servers in the CERN Computer Centre.

3. CORAL authentication and connectivity enhancements

One of the main recent enhancements of CORAL is the implementation and test of Oracle authentication via Kerberos. This represents a very interesting alternative to the customary username/password authentication mechanism, given the lack of support for X509 proxy certificates in Oracle. This is especially true for write access use cases, while read access is generally based on Frontier and delegates Oracle authentication to the Frontier server (CORAL back-end access patterns are schematically summarised in Fig. 1). Kerberos is already part of daily life at CERN, where users must hold a Kerberos ticket to be authenticated and authorized on the central Linux facilities and AFS storage. Two mechanisms for Kerberos-based connections to Oracle have been implemented in CORAL and COOL: one option allows a Kerberos principal to authenticate as an Oracle user with the same name on the database server, while another option allows proxy authentication of a Kerberos principal as an existing Oracle user with a different name on the database server. The latter option is the more interesting one because it would indeed represent an alternative authentication mechanism, based on Kerberos, to connect to existing Oracle schemas where access is presently controlled by the use of Oracle passwords. Both options would involve some user management work on the server-side to create and/or map Oracle users and Kerberos principals. To be used in CORAL and COOL, Kerberos authentication must be explicitly enabled on both client and server sides. While this mechanism is presently enabled only on a test database server, discussions have been started together with IT-DB about its possible deployment in the production environment of the LHC experiments.

Another recent enhancement in CORAL is the further consolidation of its handling of database and network instabilities, following up on the major reimplementation of this functionality previously described at CHEP2012 [3]: in particular, connection management and the handling of network glitches have been enhanced in the CoralServerProxy component to address limitations observed in its use in the ATLAS HLT system. It is also worth noting that

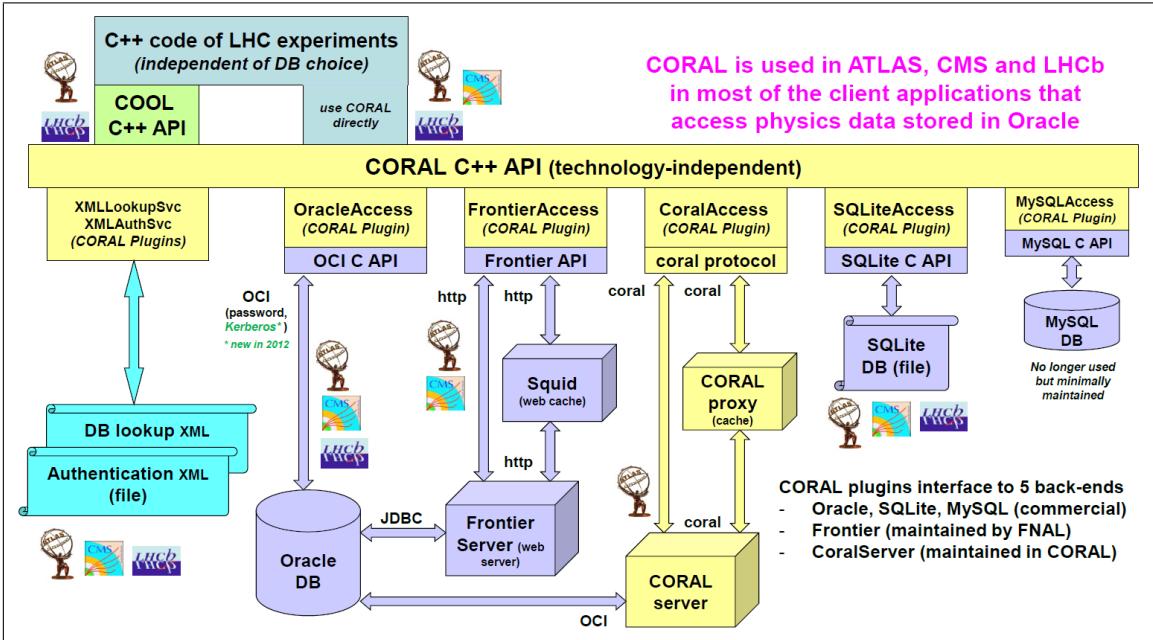


Figure 1. Back-ends supported by CORAL.

an upcoming new CORAL release will also include some enhancements in the CORAL server protocol, which were developed when LS1 had not yet started, but have not yet been added to the production ATLAS software so as not to disturb the smooth running of the ATLAS HLT during data taking. It should finally be observed that, while a large number of fixes and tests for the operation of CORAL in a multi-threaded environment were carried out in the context of the reimplementations of its handling of connection instabilities described in Ref. [3], both CORAL and COOL will need to be stress-tested in the future within a fully multi-threaded event processing framework such as those presently being developed by the LHC experiments [12].

4. COOL validation on Oracle 12c

The upgrade to the next major Oracle version 12c of the servers hosting LHC physics data at CERN, including those storing the ATLAS COOL conditions database, is currently being considered by CERN IT-DB, who are responsible for the operation of these installations. To proactively prepare for this migration, the performance of COOL queries for read access to conditions data has been tested against dedicated “integration” servers at CERN running the Oracle 12.1.0.1 server version. The automated testing tool previously developed for the Oracle 11g migration in 2011 [1] was used to prepare a detailed query performance reports for nine of the most important data retrieval use cases in COOL. As explained in Refs. [1] and references therein, the test strategy consists of creating COOL test tables containing only a few thousand “intervals-of-validity” (IOVs) and in measuring the query response time to retrieve COOL IOVs from those tables as a function of the validity time T used to look them up. Query performance and scalability are considered good if the query response time is flat and does not increase as a function of the validity time T . Two of the plots produced by the automated testing tool against Oracle 12c servers are shown in figure 2: the plot on the left represents query performance for the default COOL configuration where SQL hints are used to stabilise Oracle execution plans against unreliable statistics and bind variable peeking, while the control plot on the right represents query performance in the absence of hints. The fact that the curves on the left are indeed flat shows that the SQL hints developed for Oracle 10g and 11g server versions are still good enough to stabilize query performance against Oracle 12c. As expected, conversely,

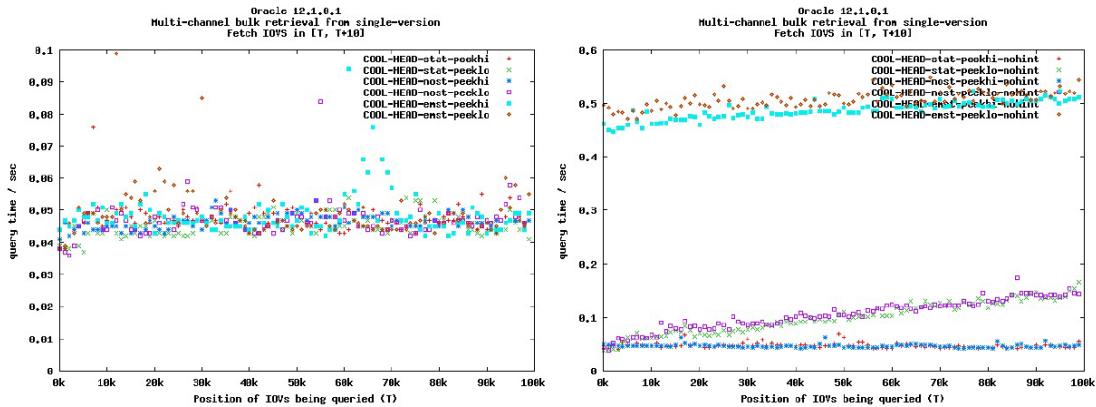


Figure 2. Performance plots for COOL SV queries on Oracle 12.1.0.1 servers with hints (left) and without hints (right). Oracle 12c “adaptive optimization” is disabled in both sets of plots.

only a few of the curves on the right are flat (such as that representing the test with reliable Oracle statistics and favorable bind variable values), while several others exhibit a query result time that increases significantly as the position of the IOV being looked up moves towards the end of the table (indicating that an alternative sub-optimal Oracle execution plan is used, for instance one involving a full index scan or a full table scan).

While COOL query performance against Oracle 12c servers was found to be satisfactory in all tested configurations when SQL hints were used (i.e. in the sets of curves on the left), it is worth mentioning that these tests also provided useful insight on how the Oracle “out-of-the-box” performance in the absence of SQL hints (i.e. in the sets of curves on the right) is affected by the new features of the Oracle 12c query optimizer. The plots in Fig. 2, in fact, represent COOL query performance for tests where the “adaptive optimization” new feature of Oracle 12c was explicitly disabled (using additional SQL hints inside COOL queries). Other tests where the Oracle 12c defaults were kept unchanged and adaptive optimization was enabled, conversely, show that COOL query performance in the absence of hints was sometimes bad (i.e. query response time increases) even in the presence of reliable statistics and favorable bind variable values. More generally, adaptive optimization was observed to lead to a large variation of results that is neither clearly reproducible nor easily predictable, i.e. different results were obtained from tests that were apparently performed in the same conditions. The precise mechanism that is responsible for this issue was not clearly identified, although the Oracle server-side trace files seem to indicate that the optimization of execution plans using bind variables that are completely different from those provided by the user may be one of the causes of this unpredictable behaviour. To improve stability and avoid any performance overheads, Oracle 12c adaptive optimization is now disabled by default for COOL at the SQL query level; so as to allow CORAL users to test it, hooks have also been added to CORAL to disable this feature at the Oracle session level.

It should also be observed that all tests above were performed using the Oracle 11g client libraries to connect to 12c servers. More precisely, the Oracle 11.2.0.3.0 client library (providing important patches for a known security vulnerability) is now used by default by CORAL and COOL applications on all Linux and MacOSX platforms. The validation of the new Oracle 12c client is currently ongoing but has not yet been completed. In particular, a few tests still need to be performed to understand how the new client libraries behave with respect to a couple of known issues observed in 11g: these include, for instance, the redefinition of Kerberos and GSSAPI symbols in the Oracle client library and some limitations in simultaneous support for Kerberos-based and password-based authentication.

Finally, it should be noted that work is in progress to repeat COOL performance validation

(against both Oracle 11g and 12c) and extend the automated performance reporting tool also for the new “vector payload” use case of COOL (where several “data payload” items are associated to the same IOV). This is particularly important in view of the upcoming use of this feature by ATLAS, who explicitly requested this new functionality and contributed to its implementation. An extensive internal review is, in fact, ongoing in ATLAS to rethink how to best match the available and upcoming COOL features to the needs of the detector groups in the experiment.

5. Conclusion

CORAL and COOL have been essential ingredients in the data processing frameworks of the ATLAS, CMS and LHCb experiments at CERN for many years. During the LHC long shutdown that started in February 2013, these two software packages and the way they are used in the experiments are undergoing extensive maintenance and consolidation work. This mainly concerns, on the one hand, the optimization of connectivity and query performance and the rationalization of the data schemas for the Oracle back-end and, on the other hand, the improvement of the software process and quality assurance as well as the consolidation of the code base and its port to new compilers and major updates of its external dependencies. Recent achievements include, in particular, the implementation of Oracle authentication via Kerberos, the systematic use of code analysis and memory/CPU profiling tools and the validation of COOL query performance against Oracle 12c servers. Work in progress and plans for the future include porting COOL to the ROOT6 major version upgrade, supporting ATLAS in their ongoing review of the conditions data storage as well as the integration and test of the software in the multi-threaded event processing frameworks that are being developed by the LHC experiments.

References

- [1] R. Trentadue et al., *LCG Persistency Framework (CORAL, COOL, POOL): Status and Outlook in 2012*, CHEP2012, NY, <http://iopscience.iop.org/1742-6596/396/5/052067>
- [2] P. v. Gemmeren et al., *Next-generation navigational infrastructure and the ATLAS Event Store*, CHEP2013, Amsterdam, <http://indico.cern.ch/contributionDisplay.py?contribId=248&confId=214784>
- [3] R. Trentadue et al., *Handling of network and database instabilities in CORAL*, CHEP2012, NY, <http://iopscience.iop.org/1742-6596/396/5/052068>
- [4] J. Cranshaw et al., *The future of event-level info repositories, indexing and selection in ATLAS*, CHEP2013, Amsterdam, <http://indico.cern.ch/contributionDisplay.py?contribId=289&confId=214784>
- [5] D. Barberis, *ATLAS Distributed Databases in LS1 and Run2*, Grid Deployment Board, CERN, December 202, unpublished, <http://indico.cern.ch/conferenceDisplay.py?confId=155075>
- [6] M. Clemencic et al., *LHCb Conditions Database operation assistance systems*, CHEP2012, NY, <http://iopscience.iop.org/1742-6596/396/5/052022>
- [7] V. Vasilev et al., *Cling, the new interactive interpreter for ROOT 6* CHEP2012, NY, <http://iopscience.iop.org/1742-6596/396/5/052071>
- [8] Coverity static code analysis service provided by CERN PH-SFT, <http://sftweb.cern.ch/coverity>
- [9] Valgrind, <http://valgrind.org>
- [10] IgProf, the Ignominous Profiler, <http://igprof.org> and references therein
- [11] gperf tools, <http://code.google.com/p/gperf-tools>
- [12] B. Hegner et al., *Introducing Concurrency in the Gaudi Data Processing Framework*, CHEP2013, Amsterdam, <https://indico.cern.ch/contributionDisplay.py?contribId=203&confId=214784>

Acknowledgements

We are grateful to the users of the CORAL and COOL software in the LHC experiments for their continuous feedback and suggestions for its improvement. We thank the SPI team in CERN PH-SFT for maintaining the development infrastructure and external software dependencies for CORAL and COOL. We are also grateful to the ROOT team for their help and support, particularly during the ongoing port of PyCool to ROOT6. Finally, we warmly thank our colleagues from CERN IT-DB, together with the DBAs in the LHC experiments, for assisting us in understanding the subtleties of the Oracle database servers they operate.