LCG Persistency Framework (CORAL, COOL, POOL): Status and Outlook

A Valassi\textsuperscript{1}, M Clemencic\textsuperscript{2,10}, D Dykstra\textsuperscript{3,11}, M Frank\textsuperscript{2,10}, D Front\textsuperscript{4,12}, G Govi\textsuperscript{5,11}, A Kalkhof\textsuperscript{4}, A Loth\textsuperscript{13}, M Nowak\textsuperscript{6,12}, W Pokorski\textsuperscript{7}, A Salnikov\textsuperscript{8,12}, S A Schmidt\textsuperscript{8,12}, R Trentadue\textsuperscript{13}, M Wache\textsuperscript{8,12}, Z Xie\textsuperscript{9,11}

\textsuperscript{1} CERN, IT Department, CH-1211 Geneva 23, Switzerland
\textsuperscript{2} CERN, PH Department, CH-1211 Geneva 23, Switzerland
\textsuperscript{3} Fermi National Accelerator Laboratory, Batavia, IL 60510, USA
\textsuperscript{4} Weizmann Institute of Science, Rehovot 76100, Israel
\textsuperscript{5} Northeastern University, Boston, MA 02115, USA
\textsuperscript{6} Brookhaven National Laboratory, Upton, NY 11973, USA
\textsuperscript{7} SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA
\textsuperscript{8} Institut für Physik, Universität Mainz, D-55099 Mainz, Germany
\textsuperscript{9} Princeton University, Princeton, NJ 08544, USA

E-mail: andrea.valassi@cern.ch

Abstract. The Persistency Framework consists of three software packages (CORAL, COOL and POOL) addressing the data access requirements of the LHC experiments in different areas. It is the result of the collaboration between the CERN IT Department and the three experiments (ATLAS, CMS and LHCb) that use this software to access their data. POOL is a hybrid technology store for C++ objects, metadata catalogs and collections. CORAL is a relational database abstraction layer with an SQL-free API. COOL provides specific software tools and components for the handling of conditions data. This paper reports on the status and outlook of the project and reviews in detail the usage of each package in the three experiments.

1. Introduction – an overview of the Persistency Framework project

The Large Hadron Collider (LHC), the world's largest and highest-energy particle accelerator, designed to collide opposing beams of protons or lead ions, started its operations in September 2008 at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland. Huge amounts of data are generated by the four experiments installed at different collision points along the LHC ring. 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 on files. Relational database systems are commonly used to store several other types of data, such as the ‘conditions data’ that record the experimental conditions (like voltages and temperatures) at the time the event data were collected, as well as geometry data and detector configuration data. In three of the experiments (ATLAS, CMS and LHCb), some or all of these types of data are stored and accessed using the

\textsuperscript{10} The author is a member of the LHCb Collaboration.
\textsuperscript{11} The author is a member of the CMS Collaboration.
\textsuperscript{12} The author is a member of the ATLAS Collaboration.
\textsuperscript{13} Supported by EGI-InSPIRE grant INFSO-RI-261323.
The PF consists of three packages (CORAL, COOL and POOL) that address the data access requirements of the LHC experiments in different areas. POOL [4,5] is a generic hybrid store for C++ objects, metadata catalogs and collections, using streaming and relational technologies. CORAL [6,7] is a generic abstraction layer with an SQL-free API for accessing relational databases. COOL [8,9] provides specific software to handle the time variation and versioning of conditions data. All packages are written in C++, but Python bindings are also provided for CORAL and COOL. As shown in figure 1, all three packages are used directly by physics applications, but CORAL is also used internally by COOL and POOL to access relational databases (in fact, COOL and POOL include the design of specific tables and queries, unlike CORAL that allows users to design their own relational schemas).

Figure 1. CORAL, COOL, POOL serve physics applications using lower level computing services.

The PF software has been developed over several years (POOL since 2003, CORAL and COOL since 2004) through the well established collaboration of developers from the LHC experiments with a team in the CERN IT department, which has also ensured the overall project coordination. The PF also benefits from the close collaboration with other AA projects, especially ROOT (that provides the object streaming software for POOL and the Python binding software for COOL) and SPI (that provides and operates the build and test infrastructure for the software, described below). The development priorities to meet the requests of the LHC experiments are set with their representatives in the Architects Forum, where all AA projects are represented. The collaboration with the teams operating the relevant services, in particular the LCG distributed relational databases [10], has also been essential to ensure a better deployment of the PF software in a production environment [11].

The release process, which is well established, is slightly different in CMS (which performs its own builds using agreed tags) and in ATLAS and LHCb (which use the software libraries and binaries built and installed on shared disks by the SPI team). Regular production releases (the latest being the LCG60b configuration with CORAL 2.3.15, COOL 2.8.9a and POOL 2.9.13) are prepared whenever one of the experiments demands it, leading to one release per month on average [12]. This is generally motivated either by urgent bug fixes and functionality enhancements in the PF software, or by upgrades in the versions of the ‘external’ dependencies (ROOT, Boost, Python, Oracle...). These external versions vary quite frequently and may be different from those installed on a predefined O/S, as they must match those chosen by the experiments for their frameworks (Gaudi for LHCb, Athena for ATLAS and CMSWW for CMS), into which the PF packages are linked to build data-processing client applications. The software is supported on many production platforms on Linux, MacOSX and Windows, using one or more compilers on each O/S (e.g. gcc4.3 and icc11 on Linux SLC5). To
improve software quality and speed up the early adoption of new external versions, automatic builds and tests of CORAL, COOL and POOL are performed every night on all production platforms, as well as on a few test platforms using new compilers (such as gcc4.5 and llvm on Linux).

CORAL, COOL and POOL have been used in production by the LHC experiments since the first data-taking in 2008 (the following sections give details for each package, summarised in table 1). While the software is by now mature in its development cycle, the effort required for user support, service operation and software maintenance is still large (though expected to decrease with time, as the issues met during the LHC start-up phase are sorted out). User requests and service incidents normally result in bug fixes in the PF code, but often require a more global analysis involving other packages (such as Oracle, the Grid middleware or ROOT). In particular, while the PF only provides client components (with one notable exception, the CORAL server), understanding service operation issues often requires a detailed troubleshooting on the server side (typically, Oracle). Software maintenance includes the port to new externals and platforms, as well as internal tasks, like the recent consolidation of the CORAL and POOL test infrastructure. The development of new functionalities, due to explicit experiment requests, is also not over. Recent examples include the addition of a new ‘vector payload’ use case to the COOL relational schema and several enhancements in the POOL collections. Some R&D work is also in progress to evaluate new technologies relevant to data access optimization.

2. CORAL – the common abstraction layer for accessing relational databases
CORAL provides a set of libraries supporting data persistency for many relational database backends. Its API [6] consists of a set of SQL-free abstract C++ interfaces that isolate the user code from the specific implementation: users write the same code for all backends, as the SQL commands for each backend are executed by the relevant CORAL library, which is loaded at run-time by a special plugin infrastructure. As shown schematically in figure 2, direct remote access to Oracle servers and local access to SQLite files, the two main technologies supported by CORAL, are used by all of ATLAS, CMS and LHCb. The data stored in Oracle servers can also be read back by CORAL clients through a middle tier server and an optional data caching server in two ways: via the FroNTier/Squid web server/cache system [13], used since long in CMS [14] and recently adopted by ATLAS [15], or via the CORAL server/proxy technology, recently deployed in the ATLAS HLT as described in detail in...
another contribution to this conference [7]. A plugin for accessing MySQL servers also exists, but this is no longer used in production by any LHC experiment after being dropped by ATLAS [7]. To address the challenges of the LCG distributed database environment, CORAL also provides components that allow the retrieval of user credentials and database replica metadata, from XML files (used by all experiments) or from an LFC server (used by LHCb until it was dropped in 2010).

CORAL has recently been and continues to be the most active of the PF sub-projects. To start with, this is the only package used by all three experiments (for conditions data, via COOL by ATLAS [16] and LHCb [17] and directly by CMS [18]; for event tags, via POOL by ATLAS [19]; for geometry and configuration data, directly by ATLAS [20, 21] and CMS [22, 23]). The support load is high also because of CORAL’s role as the gateway to relational databases. Issues reported in COOL and POOL must often be addressed in CORAL, which they use internally. Also, bugs in the underlying Oracle server software (such as one recently triggering an ORA-07445 error) often show up in CORAL-based applications. CORAL support also involves selecting and maintaining the appropriate Oracle client library version, and debugging complex OCI issues (as in the current priority, the improvement of CORAL automatic reconnections to Oracle after a network glitch). Finally, this is the only PF package which required non-negligible effort to develop new components, the CORAL server and proxy [7].

![Figure 2. Overview of the CORAL plugins and of their use in ATLAS, CMS and LHCb.](image)

3. COOL – managing the time variation and versioning of conditions data

COOL [8] provides specific software components to handle the time variation and versioning of the conditions data of the LHC experiments. Each conditions data object is associated to its metadata (an interval of validity, a data item identifier and optionally a version) and its user defined data ‘payload’. The relational implementation of COOL, based on CORAL, fixes the relational schema of metadata tables and the SQL queries involved in payload lookup from metadata, whose optimization has been the project priority for many years [9]. COOL is the baseline conditions database implementation in ATLAS [16, 24] and LHCb [17, 25], where it is used to different degrees of complexity (payload is an XML string in LHCb, while it is a user-defined table row or an external object reference in ATLAS). Recently, COOL was also chosen as the conditions database of the Minerva [26] experiment at FNAL.
4. POOL – object persistency and collections using ROOT and relational databases

POOL is a generic hybrid store for C++ objects and object collections, using a mixture of streaming and relational technologies. The oldest PF package [4], now largely in maintenance mode, it consists of a large number of components that for simplicity have been split into three groups in table 1, according to the functionalities they provide and their usage in the experiments. The first group handles object streaming into ROOT files, one of the original motivations of the POOL and PF projects. It is used to store event data in ATLAS [27] and LHCb [25]; in ATLAS this is also used to store event collections [27], as well as conditions data payload whose metadata is in COOL [24]. This component often requires maintenance when a new ROOT version is released (as in the case for the recent ROOT I/O improvements [28]). A second group of components, dealing with object collections and their navigation, is now entirely developed and solely used by ATLAS [27], where it is the basis of the event ‘tags’ database [19]. POOL collections can be stored either in relational databases (using CORAL) or ROOT files. The final group deals with object streaming into relational databases [5]. This component, until recently maintained and solely used by CMS to store conditions data [18], is now no longer used, as CMS has reimplemented it (using CORAL) inside its software framework.

5. Conclusions

The Persistency Framework provides three packages (CORAL, COOL, POOL) that are essential ingredients in the data storage and access stack of the ATLAS, CMS and LHCb experiments at CERN and have been used for LHC data taking since 2008. While the software is by now mature in its development cycle, a large effort is still required for user support, service operation and maintenance tasks. A few new functionalities are also being developed, as requested by the experiments.

References


Acknowledgments
We are indebted to the former members of the Persistency Framework team for establishing the bases on which this project has grown. We especially wish to thank Radovan Chytracek, Dirk Duellmann and Ioannis Papadopoulos for their lead roles in the CORAL and POOL projects over several years.

We are grateful to the users of the CORAL, COOL and POOL software in the LHC experiments for their continuous feedback and suggestions for its improvement. All the former and current SPI team members, in particular Stefan Roiser, deserve special thanks for maintaining the development infrastructure and external software dependencies for the Persistency Framework. We are also grateful to the ROOT team for their help and suggestions. Finally, we wish to thank our colleagues from the Physics Database Team in CERN IT, together with the DBAs in the LHC experiments, for assisting us in understanding the subtleties and in fixing the issues in the Oracle database servers they operate.