WORLDWIDE LARGE HADRON COLLIDER COMPUTING GRID

CORAL and COOL
during the LHC long shutdown

CHEP '13 AMSTERDAM

CORAL

- A software access layer for relational databases
- Same C++ or python user code for different relational back-ends – e.g. transparently switch Oracle/Frontier
- Schema design and SQL performance optimization are the responsibility of individual CORAL users

CORAL provides a DB-independent and largely (but not completely) SQL-free C++ API for accessing data stored using relational DB technologies. It takes care of executing user requests using the appropriate SQL.

```
coral::ConnectionService svc;
session = svc.connect( URL, coral::Update );
session.transaction().start( false );
coral::TableDescription tableDescription;
tableDescription.setName( “myTable” );
tableDescription.insertColumn( “ID”, “long long” );
tableDescription.insertColumn( “Data”, “double” );
tableDescription.setPrimaryKey( “ID” );
session.nominalSchema().createTable( tableDescription );
```

Example: table creation through the CORAL API

SQLite
URL = “sqlite_file:mydatabase.db"

```
CREATE TABLE "myTable"
( "ID" SLONGLONG ,
"Data" DOUBLE ,
PRIMARY KEY("ID") )
```

Oracle
URL = “oracle://server/myschema"

```
CREATE TABLE MYSCHEMA."myTable"
( "ID" NUMBER(20),
"Data" BINARY_DOUBLE,
CONSTRAINT "myTable_PK"
PRIMARY KEY ( "ID" ) )
```

MySQL
URL = “mysql://server/myschema"

```
CREATE TABLE “myschema"."myTable"
( "ID" BIGINT NOT NULL,
"Data" DOUBLE PRECISION,
CONSTRAINT  "myTable"
PRIMARY KEY  ( "ID" ) )
```

COOL

- A conditions database application to manage the time-variation and versioning of conditions data
- COOL offers a palette of configurable pre-defined relational schemas for different use cases
- Schema design and SQL performance optimization are the responsibility of the COOL development team

Each COOL conditions object has
- **Metadata** *(system-controlled)*
  - Data item identifier
  - Interval-of-validity [since, until]
  - Version information
- **Data “payload”** *(user-defined)*
  - Physics values (temperatures, calibration parameters…)
  - Separate columns or a CLOB

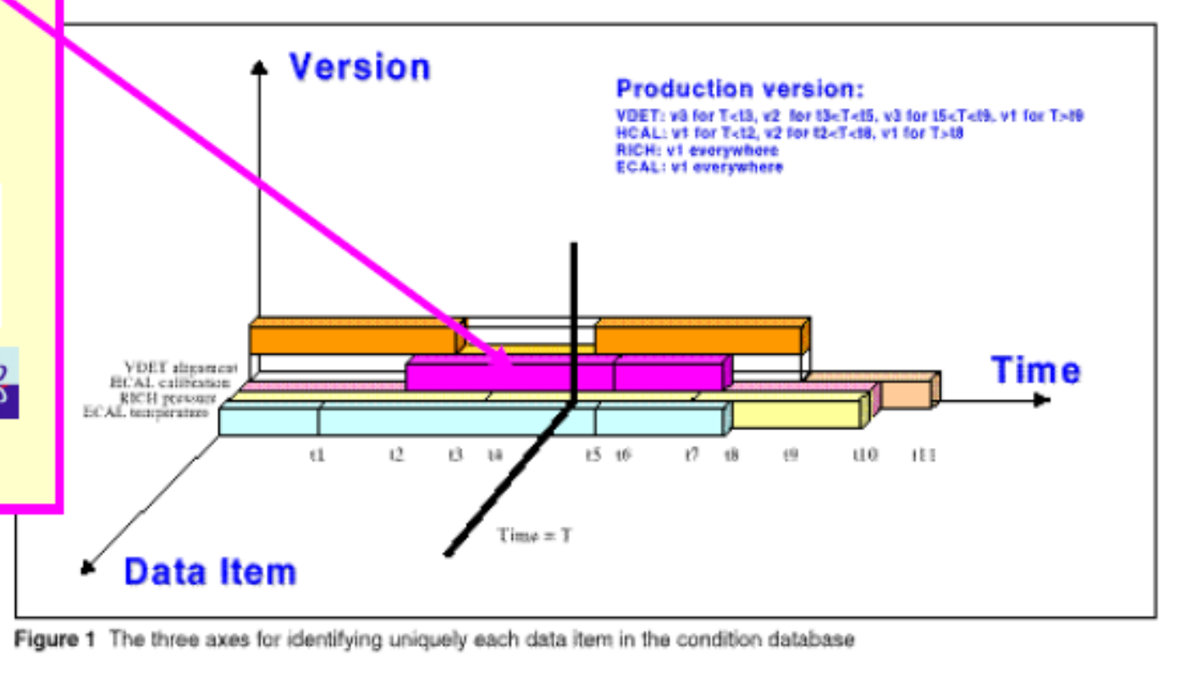


Figure 1 The three axes for identifying uniquely each data item in the condition database

COOL provides a technology-independent C++ API to handle the time variation and versioning of the conditions data of the LHC experiments. This is implemented using relational databases, based on CORAL.

General news

- Move repositories from CVS to SVN (CORAL and COOL for new releases, POOL for data preservation)
- 11 releases since CHEP12, support for gcc47/48 with c++11, port to clang33, icc13, Python2.7, Boost1.53
- Integration with profiling tools (valgrind, gperftools, igprof), memory and performance improvements
- Integration with Coverity code analyzer, several fixes
- POOL is maintained (and only used) by ATLAS

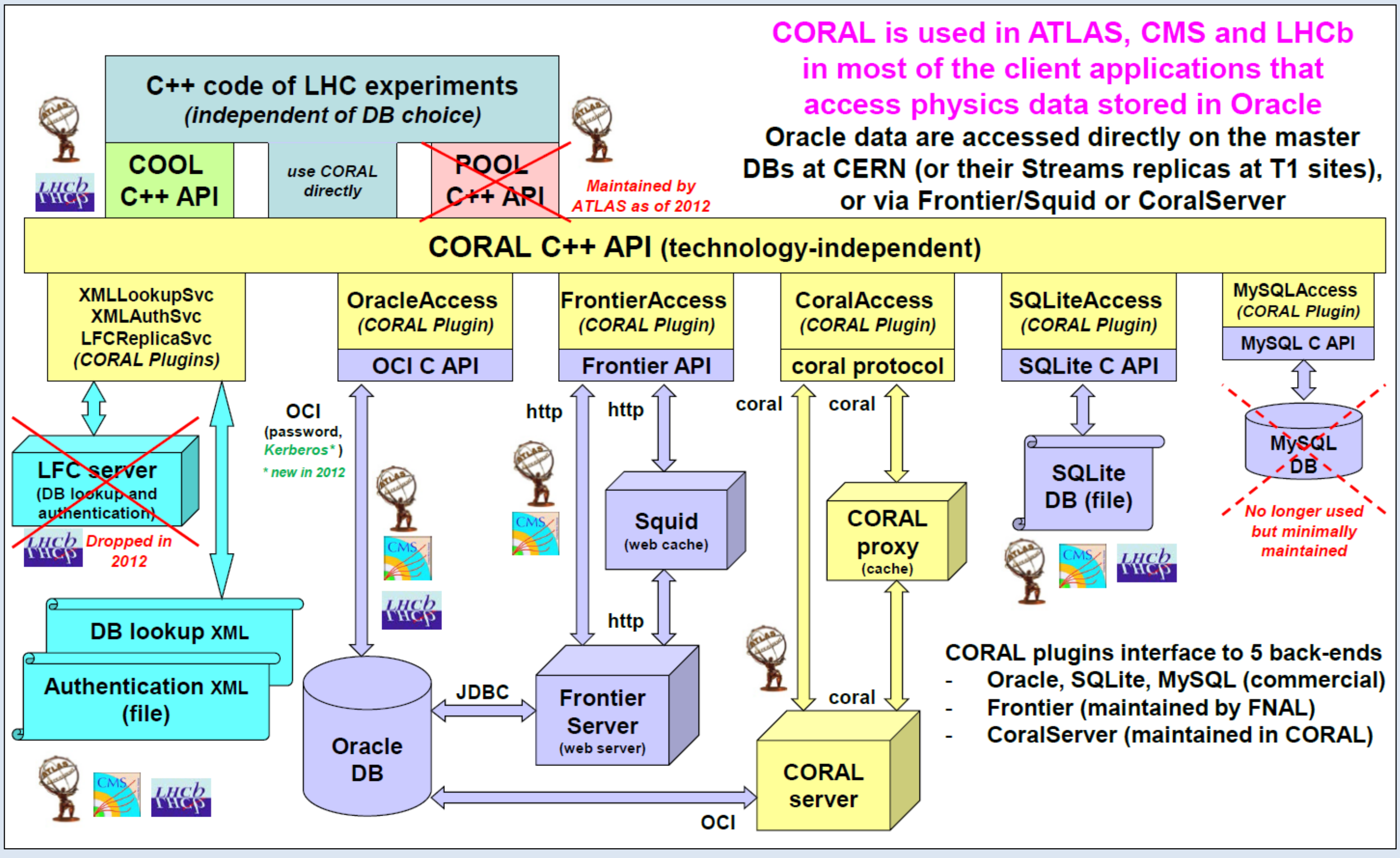
CORAL and COOL in the LHC experiments**	ATLAS	CMS	LHCb
CORAL (Oracle, SQLite, XML authentication and lookup)	Conditions data (COOL) Geometry data (detector descr.) Trigger configuration data Event collections/tags (POOL)	Conditions data Geometry data (detector descr.) Trigger configuration data	Conditions 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)*

** = new in 2012*

*** CORAL and COOL are also used outside LHC (NA62 at CERN and Minerva at FNAL)*

CORAL news

- Add support for Kerberos authentication (for new ‘external’ users or as proxy for existing Oracle users)
- Upgrade to oracle 11.2.0.3.0 client with security fixes
- Better connection management in CoralServerProxy and improved CoralServer resilience for ATLAS HLT


C++ code of LHC experiments
(independent of DB choice)
COOL C++ API
use CORAL directly
~~POOL C++ API~~
Maintained by ATLAS as of 2012
CORAL C++ API (technology-independent)
XMLLookupSvc
XMLAuthSvc
LFCReplicaSvc
(CORAL Plugins)
OracleAccess (CORAL Plugin)
OCI C API
FrontierAccess (CORAL Plugin)
Frontier API
CoralAccess (CORAL Plugin)
coral protocol
SQLiteAccess (CORAL Plugin)
SQLite C API
MySQLAccess (CORAL Plugin)
MySQL C API
~~LFC server (DB lookup and authentication)~~
Dropped in 2012
OCI (password, Kerberos*)
* new in 2012
http
http
http
coral
coral
coral
Squid (web cache)
CORAL proxy (cache)
SQLite DB (file)
~~MySQL DB~~
No longer used but minimally maintained
DB lookup XML
Authentication XML (file)
Oracle DB
JDBC
Frontier Server (web server)
CORAL server
OCI

CORAL is used in ATLAS, CMS and LHCb in most of the client applications that access physics data stored in Oracle

Oracle data are accessed directly on the master DBs at CERN (or their Streams replicas at T1 sites), or via Frontier/Squid or CoralServer

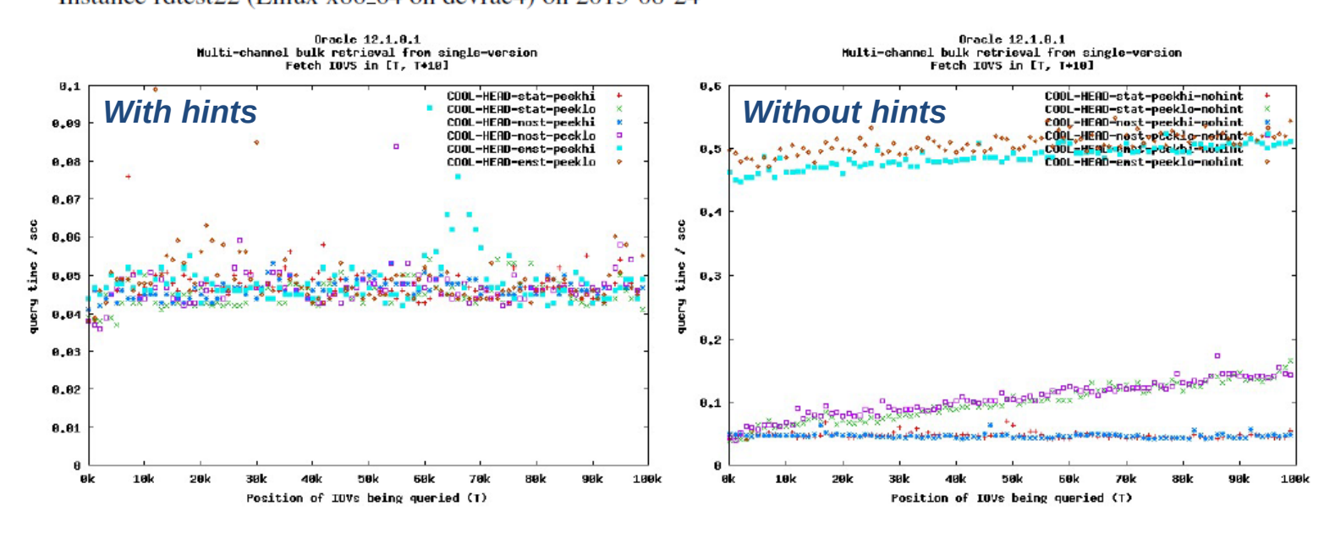
CORAL plugins interface to 5 back-ends
- Oracle, SQLite, MySQL (commercial)
- Frontier (maintained by FNAL)
- CoralServer (maintained in CORAL)

COOL news

- Performance validation on Oracle 12c servers (disable new ‘adaptive optimization’ features)
- ATLAS review of COOL usage – test vector payload

1. Use case SV_R (COOL-preview on Oracle 12.1.0.1.0)

Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
Instance rdtest22 (Linux x86_64 on devrac4) on 2013-08-24



Systematically test 9 read-back use cases of COOL:
- **SV_R**. Single version retrieval (with inline payload).
- **SP_R**. Single version retrieval (with a separate payload table).
- **MVUR**. Multi version retrieval from a user tag (with inline payload).
- **MPUR**. Multi version retrieval from a user tag (with a separate payload table).
- **MVHR**. Multi version retrieval from the HEAD (with inline payload).
- **MPHR**. Multi version retrieval from the HEAD (with a separate payload table).
- **MVTR**. Multi version retrieval from a normal tag (with inline payload).
- **MPTR**. Multi version retrieval from a normal tag (with a separate payload table).
- **SC_R**. Single version retrieval (with inline CLOB payload).

COOL performance report (for Oracle 12.1.0.1.0 servers)
- **Response time must be the same for retrieving old or recent data (no table/index scans)**
- **Demand good performance with COOL hints, cross-check without COOL hints**
- **Test 6 combinations of bind variable peeking (low/high) and statistics (good/stale/absent)**

Ongoing and future work

- PyCool migration to ROOT6 (without Reflex)
- Performance validation of COOL vector payload (several payload items associated to the same IOV)
- Release of CoralServer protocol enhancements
- Several API extensions and improvements
- Port to Oracle 12c client, follow up known issues
- Infrastructure migrations (cmake, jira, puppet…)
- Integration with multi-threaded frameworks

A. Valassi, R. Trentadue (CERN IT), M. Clemencic (CERN PH / LHCb),
D. Dykstra (FNAL / CMS), N. Goyal (Mody Institute / ATLAS),
A. Salnikov (SLAC / ATLAS), M. Wache (Univ. Mainz / ATLAS)

IT-SDC

https://twiki.cern.ch/twiki/bin/view/Persistency