# CORAL & COOL
## A critical review from the inside

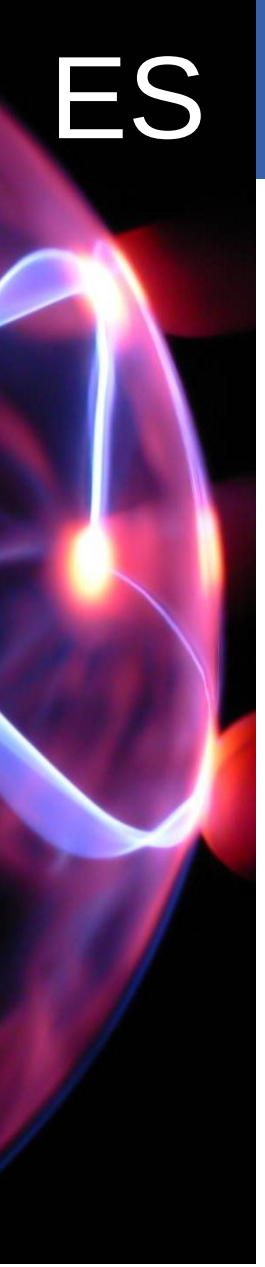
**Andrea Valassi (IT-ES)**

*WLCG TEG on Databases, 7th November 2011*

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Disclaimers





- **Much of what follows is my personal opinion**
- **I will sometimes state what is already obvious**
- **I will not attempt to give a revolutionary vision**

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Outline



- **Rationale and mandate for the TEG**
  - Set the *questions* that I will try to address (mentioned below)
- **CORAL, COOL – general comments**
  - *Standard solutions?*
  - *Common solutions?*
- **CORAL, COOL – comments on process and organization**
  - *Development model and time to release?*
  - *"First and foremost do not disrupt the large scale production service"*
  - *Operational effort?*
  - *Long term support*
- **CORAL, COOL internal details – good and not so good**
  - *Lessons learnt?*
  - *Needs for the next 2-5 years?*
- **CORAL, COOL – plans and work in progress**
  - Plans for the next few months
- **Wrap-up and conclusions**

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Rationale – a few quotes



- **From Ian's WLCG Strategy Rationale:**
  - While the service that we have today is running at a very large scale, it nevertheless takes some significant operational effort. Analysis of the failures shows that most failures are not in the grid middleware (software) at all […]. In addition database service problems cause significant operational effort.
  - [The middleware] shortcomings perceived by the experiments included the apparent slowness with which revisions and updates could be provided.
  - Finally, we must be concerned about the long-term support of the software that we rely on. We have made use of various funding sources until now. […] However, the core of what we rely on must be easily maintained, otherwise we risk to end up in a situation where something we rely on becomes unavailable to us.
  - […] We must of course note the real and important successes, and **ensure that first and foremost we do not disrupt the large scale production service and maintain the production throughput that we have**. We have also developed the operational and support frameworks and tools that are able to manage this large infrastructure. No matter how the underlying technologies evolve, this will remain crucial.
  - The goals of an activity to reassess our requirements and define our strategy include:
    - Re-building common solutions between the experiments. […] With many different experiment-specific solutions to problems is probably unsustainable in the future. Indeed sites are being asked to support different services for different experiments […].
    - The reassessment must take into account some of the lessons we have learned in not only functional aspects, but also operational and deployment problems.
    - There must be consideration for the long-term support of the software. If we can make use of standard solutions rather than writing specific software, we must do that. […] We must concentrate our development efforts on those areas where other solutions do not exist.

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# TEG mandate – a few quotes



- **From Ian's WLCG TEG Mandate:**
  - The work should, in each technical area, take into account the current understanding of:
    - Experiment and site needs in the light of experience with real data, operational needs (effort, functionality, security, etc.), and constraints;
    - Lessons learned over several years in terms of deployability of software;
    - Evolution of technology over the last several years;
    - Partnership and capabilities of the various middleware providers.
  - It should also consider issues of:
    - Long term support and sustainability of the solutions;
    - Achieving commonalities between experiments where possible;
    - Achieving commonalities across all WLCG supporting infrastructures (EGI-related, OSG, NDGF, etc).
  - Deliverables
    - Assessment of the current situation with middleware, operations, and support structure.
    - Strategy document setting out a plan and needs for the next 2-5 years.

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# CORAL - is there a standard solution?



- **CORAL is mainly a tool that allows us to develop C++ applications without knowing which backend will be used**
  - In particular: Oracle, SQLite, Frontier, CORAL server
    - Originally in line with the "3D" distributed database deployment model
    - Just change ~one line (of code or in an XML configuration file)
    - It is this feature (and its integration in COOL) that allowed ATLAS to quickly move to Frontier for distributed analysis
  - It is also meant to gives us some specific features (retrial, pooling) and a few handles for easier configuration

- ***IMO no standard tool can completely replace CORAL***
  - Assuming we do need relational databases (and access from C++)
  - For python, sqlalchemy is an interesting tool – but for C++?
    - CORAL is NOT "like ODBC" – CORAL largely writes the SQL for you
    - Even if we found one for Oracle and/or SQLite, is there a standard tool to replace Frontier and/or CORAL server for caching and multiplexing?
  - (I did not really look that much, though…)

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# COOL - is there a standard solution?



- **COOL is mainly the design and implementation of a data model and a relational database schema**
  - For specific "interpretations" of time-varying and versioned data
    - Specific to HEP, but even to a given experiment or different subgroups within one experiment (see next slides on "common project")

- ***IMO no standard tool replaces COOL (or CMS conditions)***
  - Unlikely to find ready-made applications with a suitable data model
    - And again, even if there was one for Oracle, what about caching and multiplexing as in Frontier and/or CORAL server (or even SQLite)?
  - One can/should rather look for more standard patterns to describe this data model – and for more simplicity and commonality
    - No interval support in Oracle, but do we really need it (e.g. CMS)?
  - (I did not really look that much, though…)

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# CORAL - is it a common solution?



- **CORAL is used by ATLAS, LHCb and CMS**
  - Most components are used by all three experiments (e.g. all basic packages and the Oracle, SQLite and XML plugins…)
- **Some components are (were) used only by one or two**
  - FrontierAccess was used only by CMS, but now also by ATLAS
    - And LHCb is now interested in testing it too
  - CORAL server is used only by ATLAS
    - With specific ATLAS contributions for its development and operation
    - (Was) meant for more general use – delayed mainly by low resources
  - LFCReplicaSvc was used only by LHCb, but has been dropped
  - PyCoral is (was) used only by CMS? (now using sqlalchemy?)
    - ATLAS and LHCb use PyCool instead?
- ***CORAL is already a very successful common project!***

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# COOL - is it a common solution?



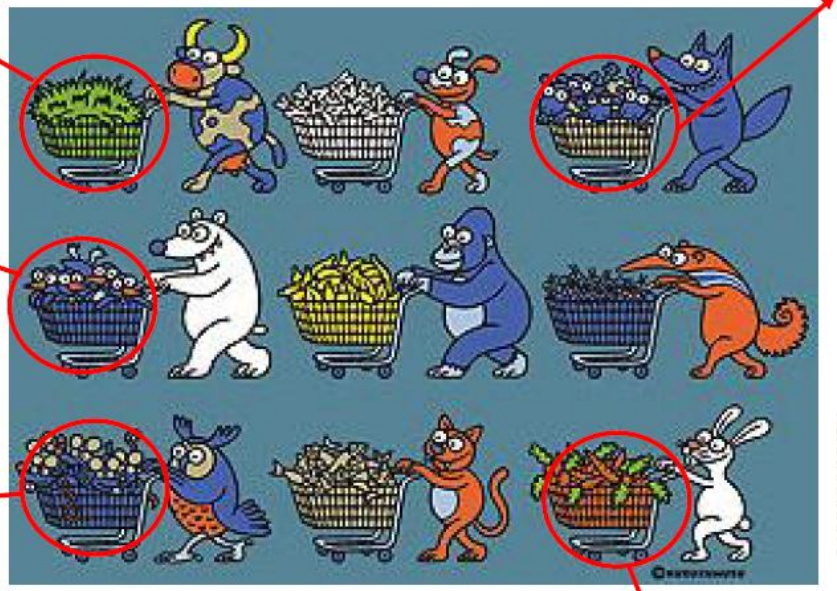
## About the "common" project…

- Is there anything *in common* between what you want?

Reading the XML BLOB containing the LHCb calibration data valid for the event processed

Retrieving the POOL alignment object for the run processed

Registering that the CMS detector geometry in a set of Oracle tables is valid for 2008 and 2009



© KUKUXUMUSU

Storing Atlas high voltages from PVSS into MySQL whenever the values change (every few seconds)

Reading Alice alignment from the ROOT files for the run processed



CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# COOL - is it a common solution?



- **COOL is used by ATLAS and LHCb**
  - CMS is using its own database design for conditions data
- **COOL is used in different ways by (within) ATLAS and LHCb**
  - What is common is the general infrastructure and the design of the *basic (common)* data model for validity intervals and their queries
  - COOL is complex because it offers very many features
    - Most of these features were requested by (and implemented with help from) ATLAS – while LHCb uses only a subset IIUC
    - HEAD, tags and "user" tags, single and multi version, channels…
    - Many use cases to maintain (e.g. many 11g queries to optimize…)
- ***There is some commonality, but we could do better***
  - IMO the homework is on (and within) the experiments
    - How much of COOL does ATLAS really use and need?
    - How little of COOL does LHCb use, is this almost an ATLAS-only tool?
    - How much simpler is the CMS data model in reality? Can we simplify COOL a la CMS? Would CMS benefit from a common solution instead?
  - Large effort, is it worth it and do we/you have the time?
    - (And I ignore the question of avoiding service disruptions)

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Collaboration and release model



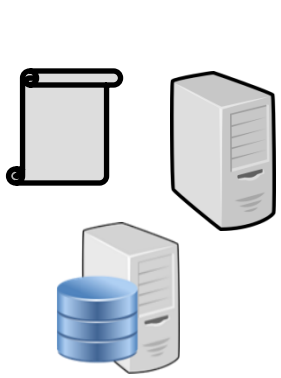
- **All activity is driven by experiment requirements and issues**
  - Support in operational issues (debug problems, plan migrations…)
  - Development of new features on requests from the experiments
  - Regular software releases
- **Software releases (one per month on average)**
  - Discussed with the experiments at LIM and AF meetings
    - Releases built by SPI for ATLAS and LHCb on AFS (and CVMFS)
    - Release tags for CMS that builds its own CORAL releases
  - No predefined release schedule
    - *Release on demand when the experiments need it*
  - Wide range of supported platforms and compilers
    - Not bound to the default system compiler
    - SLC5 (gcc43, gcc46, icc, llvm), SLC6, MacOSX, Windows (vc9)…
  - Each CORAL & COOL release has a set of recommended externals
    - Not bound to the default system "externals" (python, boost, oracle…)
    - Can re-release (rebuild) the same CORAL&COOL code base if externals change
    - ROOT is the main external dependency of the stack (only for PyROOT in COOL)
  - Further reduce the time to release by automatic nightly tests and builds
    - And post-install COOL & CORAL validation procedure is now outsourced to SPI
  - Group and delay API and schema changes to avoid service disruptions
- **A successful model? The experiments seem happy!**

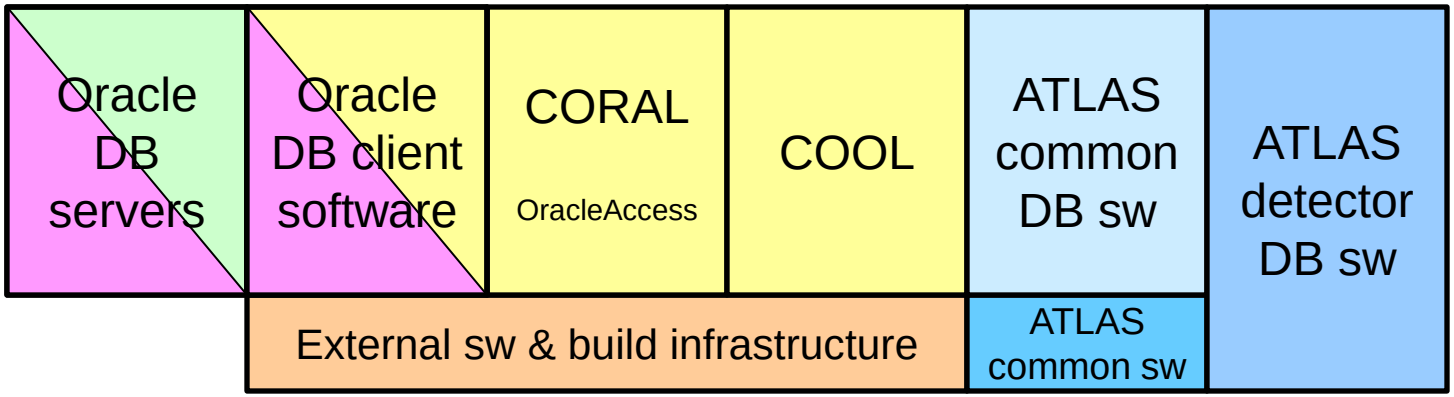
CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

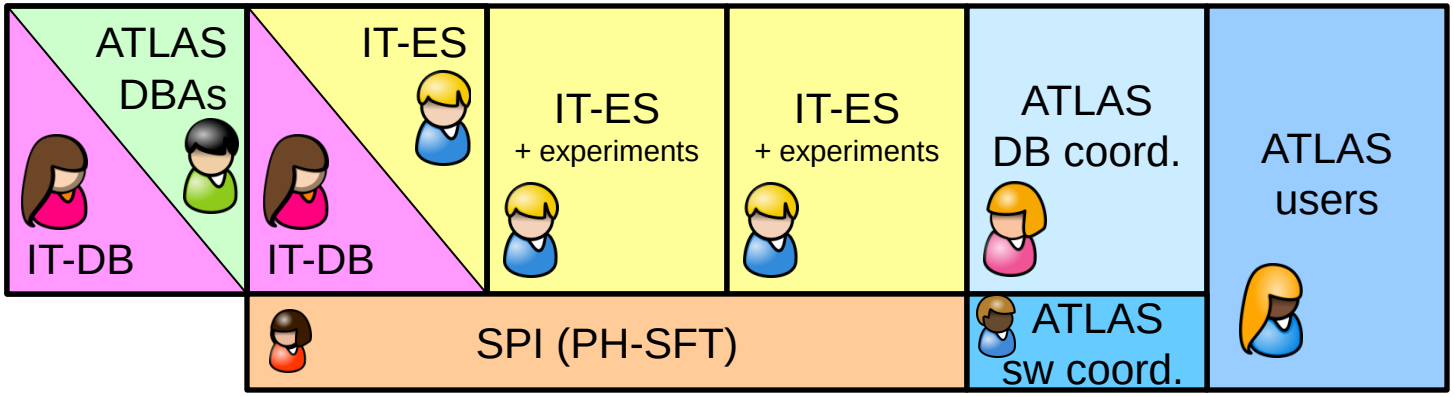
# Dependencies and collaborations

CERN IT Department

Example for ATLAS for Oracle (without taking into account Frontier or CoralServer)
*[NB box sizes are obviously not to scale!…]*





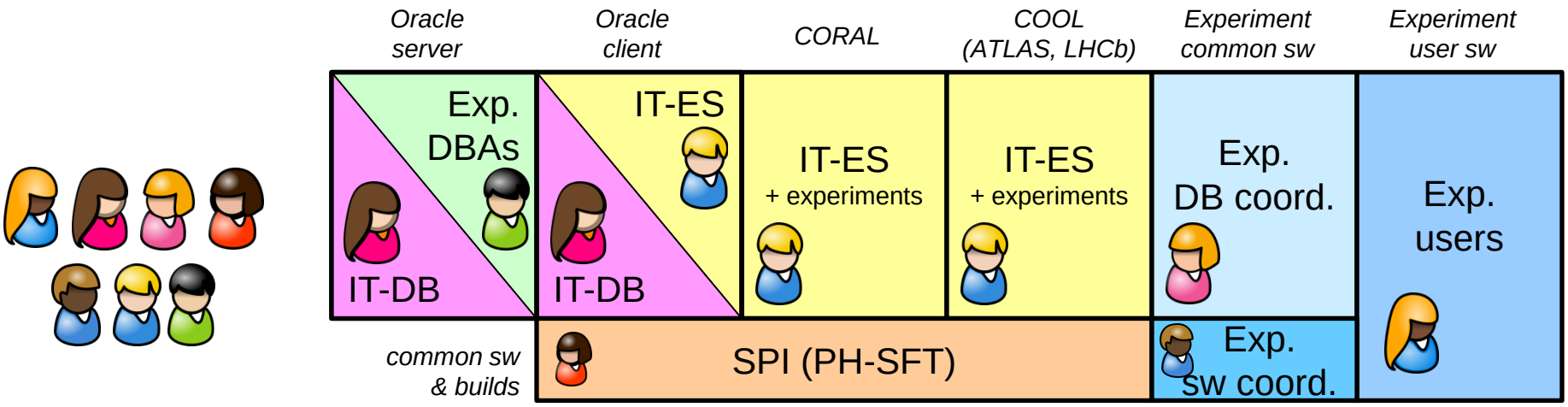




**Software interactions are mirrored by the collaboration of different teams**

***Thanks to the IT-DB physics team, to experiment users, developers, coordinators, DBAs and to the SPI team for their help and collaboration over many years!***

# Human aspects: lessons learnt



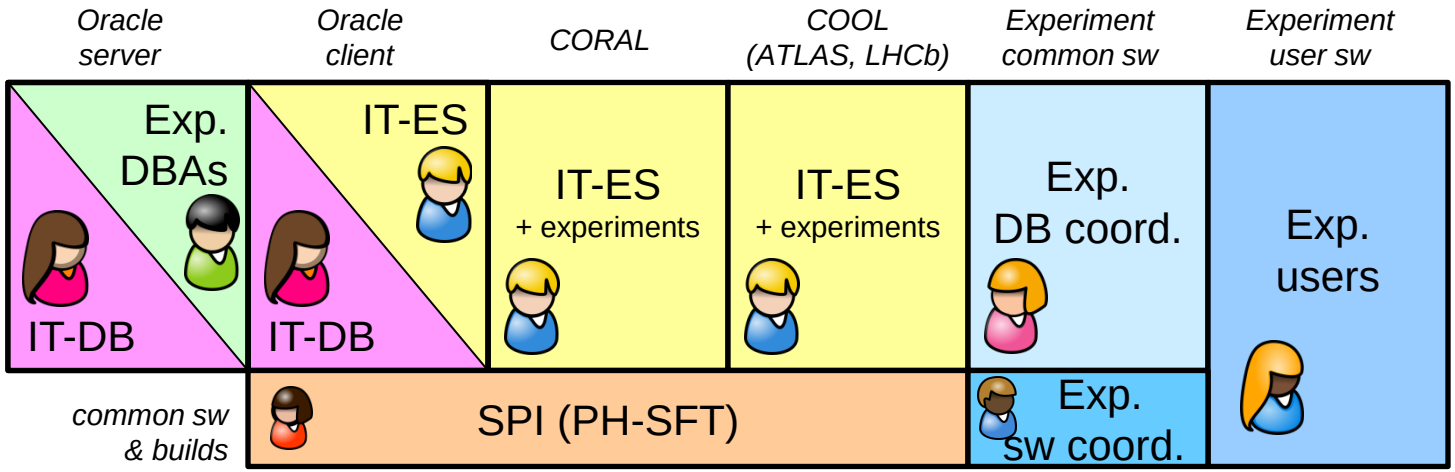



- **Experiment DB coordinators and DBAs are essential!**
  - Ensure uniform usage pattern for all experiment users and understand different needs
  - Filter user requests and prioritize experiment software needs to CORAL&COOL team
  - Provide central expertise and point of contact to follow up service operation issues
- **There is more to CORAL&COOL than developing new software**
  - Operation support and maintenance take more than 50% of the core team effort
    - Experiment support for service operation issues (firefighting and consultancy)
      - e.g. network glitches (all experiments), ORA-25408 (CMS), ATLAS Tier0 issues, 11g validation
    - Software maintenance, releases, testing
    - Oracle client software selection and installation (and patching in a few cases)
  - *Doing this in common for several experiments does reduce the overall efforts!*

# Human aspects: manpower concerns

CERN IT Department

The figure is for Oracle. The two boxes on the left should also contain: for Frontier, the work in CMS on client/server software, and that in ATLAS/CMS on its deployment; for CoralServer, the work in ATLAS on its deployment.




- **Manpower in the core CORAL&COOL team is critically low**
  - Experiment contributions to core team have been essential but decreased to << 1 FTE
    - And some are mainly targeted at the development of new features, rather than operations
    - Long-distance coordination of distributed effort also adds some overhead for a small team
  - Omitting students, IT-ES team is around 1.7 FTEs from one permanent staff (who will move to other projects eventually) and one fellow (on time-limited EGI project funding)
- **Expected result from this TEG: clarify the long-term support model**
  - Who should support CORAL & COOL (IT, experiments…) and for how long
  - Staffing level and profile should ensure a critical mass mainly for operations and expertise retention (new developments have lower priority if resources are limited)
    - Overhead from training any new team members should also be taken into account

# CORAL - lessons learnt?



- ***Idea and implementation of DB independence was the right thing***
  - In particular: *bringing Frontier into CORAL was the right choice*
- **Inside the database plugins**
  - *OCI was the right choice for OracleAccess*
    - Fewer issues than we had (would had) seen with the OCCI compiler dependency
    - And even so, we still get not-so-common software build issues (kerberos, gssapi, selinux…)
    - Not a very intuitive or practical API (led to some bugs in CORAL), but it's ok once you know it
  - Large duplication of code between different database plugins
  - Data dictionary queries are (were?) causing a performance overhead

# COOL - lessons learnt?



- ***Idea and implementation of DB independence was the right thing***
  - In particular: *basing COOL implementation on CORAL was the right choice*
- **Data model is (too) complex?**
  - See previous comments, is it all needed?
  - Really need [since, until] as independent variables?
  - Really need all those tagging features?
- ***Focus on performance optimization was essential***
  - Execution plans now well understood and table (indexes, hints…)
  - Requires Oracle expertise in core team and help from DBA teams
  - Good to focus on “direct lookup” (rather than on “payload queries”)
- **No software handling of data partitioning and/or archiving**
  - Managed by IT/DB using
- **Misleading terminology?**
  - Folders and folder sets – relique of the Objectivity days…
- **Internal separation of CORAL and ‘generic relational’ layer?**
  - Was useful initially but has become difficult to maintain

# CORAL+COOL – lessons learnt?



- ***Decoupling interface from the implementation was right for both***
- **Integration of CORAL & COOL could be improved?**
  - Good to have CORAL split from COOL (e.g. CMS) and CORAL not in POOL!
  - But a more uniform approach could ease their maintenance and support
- **Functional testing**
  - *COOL test-drive development is one of the reasons for its success*
  - The coverage of the CORAL test suite is still too poor
    - Many CORAL issues are effectively tested only inside COOL
    - Tests for some CORAL features (monitoring, reconnections) are now being written/improved
    - And the configurations of the two test suites are still a bit different
- **PyCoral, PyCool**
  - Having a *pythonized version of the API is very useful*
  - But PyCool (Reflex/PyROOT) and PyCoral (native) are incompatible
    - Rewrite PyCoral with PyROOT? ROOT dependency no longer an issue
- **Sessions and transactions**
  - COOL does not yet allow user control on sessions and transactions
- **coral::AttributeList and cool::Record**
  - COOL tried to avoid the CORAL issues in C++/SQL type mismatch
    - Force the use of defineOutput in CORAL queries (e.g. 32/64bit types)?
- **For both: could improve documentation, CPU performance, valgrind…**

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Overview of future plans



- **Software maintenance**
  - Regular software releases of the stack (~one per month)
  - New platforms/compilers, new external software versions
    - Including selection/installation/patching of Oracle (OCI) client
  - Infrastructure: repository, build, test (e.g. SVN, cmake, valgrind…)
- **Operation and support**
  - User support and bug fixing
  - Debugging of complex service issues (~from a client perspective)
- **A few enhancements and new features**
  - See next slides for details on CORAL and COOL

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Plans for CORAL



- **Reconnection after network/database glitches**
  - *Related to several recent incidents (ATLAS T0, ORA-25408 in CMS)*
- **More robust tests (spot bugs before users report them)**
  - Was heavily relying on COOL test suite so far
- **Performance studies and optimizations**
  - e.g. complete data dictionary query removal in the Oracle plugin
    - compare Oracle, Frontier, CoralServer to find any other such issue
    - also related to CORAL handling of read-only transactions (serializable)
- **Enhance/redesign monitoring functionalities**
  - Interest by ATLAS online (CORAL server) and CMS too
    - Student working on monitoring the CoralServerProxy hierarchy too
  - In practice: need better code instrumentation for DB queries
    - See Cary Millsap's recommendations at his seminar in June
- **In parallel: a few minor feature enhancements**
  - Support for sequences, FKs in SQLite, multi-schema queries…

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Plans for COOL



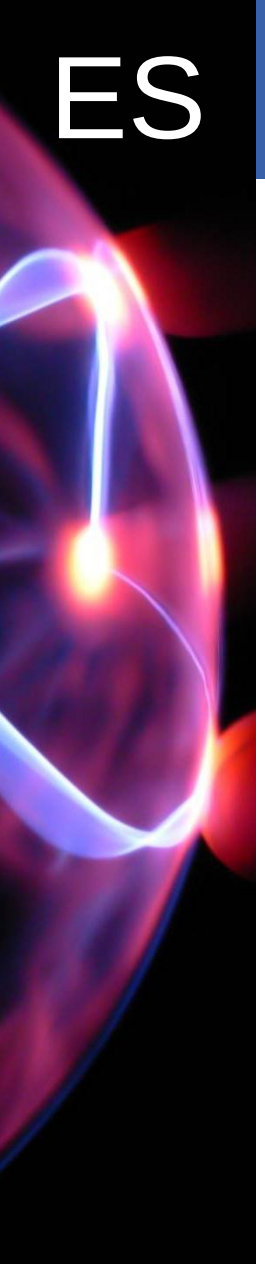
- **Performance optimizations and related new features**
  - Validate query performance on Oracle 11g servers
    - Major optimizations a few years ago (queries rewritten, indexes…)
    - Hints added (in dynamic query creation) to stabilize execution plans
    - Presently seems that the 10g optimizer is needed (report Oracle bug?)
  - New API (and SQL queries) for fetching first/last IOV in a tag
    - All access (read/write) to the COOL database goes via the API
- **New software features and database schema extensions**
  - e.g. COOL user control over CORAL sessions and transactions
  - e.g. better storage of 'vector' payload for IOVs and of DATE types
  - Some of these are already coded but need to be tested/released

# Wrap-up – standard & common solutions



- **Replacing CORAL or COOL completely by standard solutions does not seem realistic at the moment**
  - No single magic out-of-the-box bullet (not yet at least)
    - But one should keep an open eye on the market (e.g. sqlalchemy)
  - Even if we did find a standard solution, the point is not only software enhancement/maintenance, but also support and operation
- **I have not discussed alternative non-relational database models**
  - IMO this may be easy for a single developer to start, but the challenge is again large scale operation (e.g. query optimization will probably not come magically out-of-the-box and will need some brain work, like in Oracle…)
    - Some similarities with MySQL in CORAL (now essentially no longer used)?
  - Would also need a new distribution model in case (alternative to 3D/Frontier)
- **CORAL & COOL model did lead to successful common projects**
  - CORAL is largely a successful common tool for ATLAS, LHCb and CMS
  - Less commonality in COOL/conditions: the experiments should review this
  - Common operation and support do take a large effort, but this often benefits more than one experiment, improving the overall sustainability

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**



# Conclusions

- **ATLAS, LHCb use CORAL & COOL, CMS uses CORAL**
  - To access conditions and other data using Oracle, Frontier, SQLite
- **Common project development model has been a success**
  - Development is fully driven by the experiment requirements
- **CORAL (and COOL) support for many backends is essential**
  - This is what allowed such a fast switch of ATLAS to Frontier
  - This must be kept into account in any software rewrite or (r)evolution
- **Does the COOL data model really need to be so complex?**
  - How much does ATLAS really need? How simpler are CMS/LHCb?
- **Highest workload is operations: software & service support**
  - Releases, Oracle client issues, debugging of complex service issues
  - A few new developments too, in parallel
- **Expected result from this TEG: clarify long-term support**
  - Who should support CORAL & COOL and for how long
  - Staffing level and profile should ensure critical mass for operations and expertise retention (and for new developments if required)

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Reserve slides

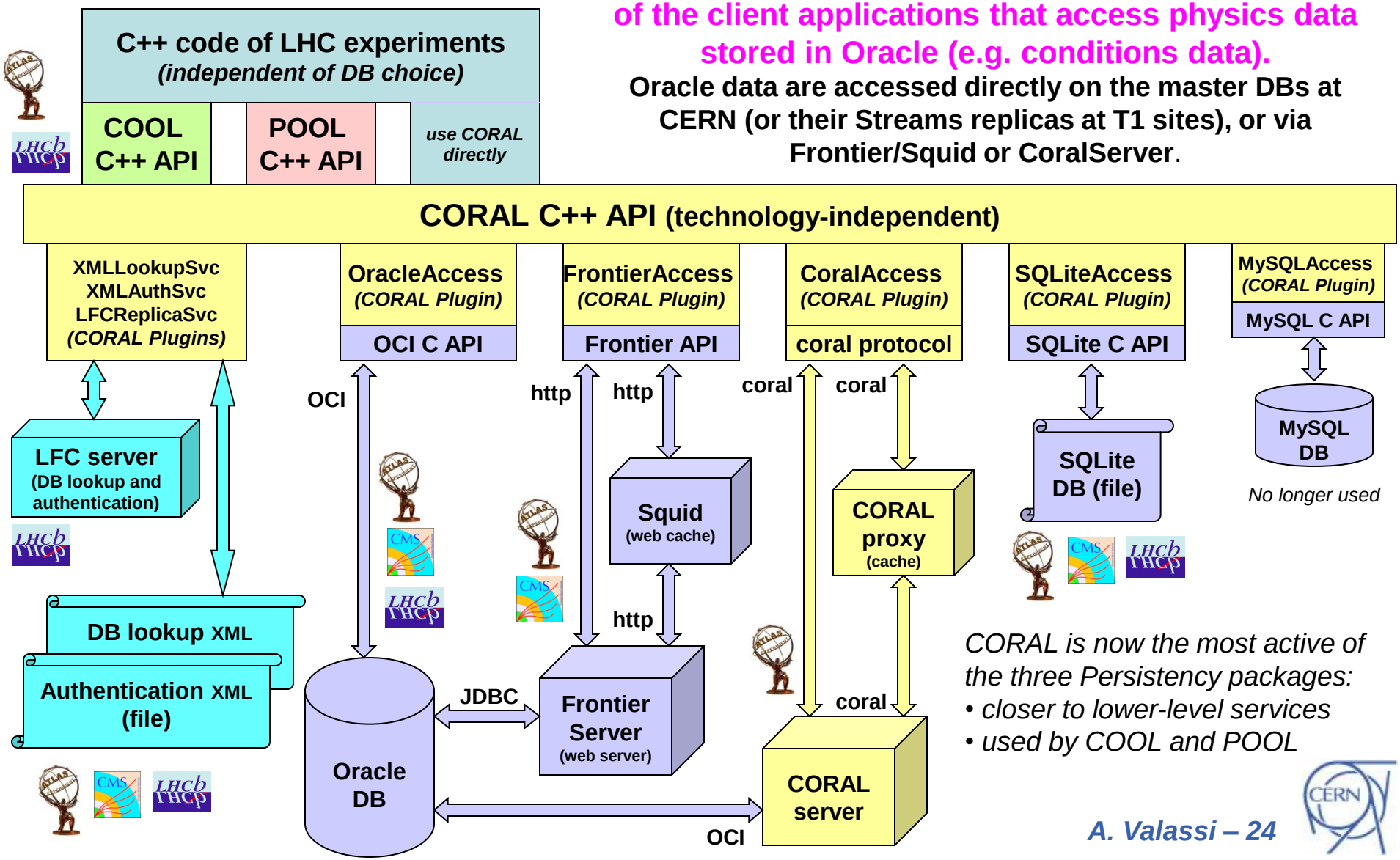
**(largely extracted from my talk at the June 2011 DB Futures workshop)**

# Overview of usage and architecture

CERN IT Department

**CORAL is used in ATLAS, CMS and LHCb by many of the client applications that access physics data stored in Oracle (e.g. conditions data).**

**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 is now the most active of the three Persistency packages:*

- *closer to lower-level services*
- *used by COOL and POOL*

# Details of CORAL, COOL, POOL usage

CERN IT Department

| Persistency Framework in the LHC experiments | ATLAS | CMS | LHCb |
|---|---|---|---|
| **CORAL** (Oracle, SQLite, XML authentication and lookup) | Conditions data (COOL)<br>Geometry data (detector descr.)<br>Trigger configuration data<br>Event collections/tags (POOL) | Conditions data<br>Geometry data (detector descr.)<br>Trigger configuration data | Conditions data (COOL) |
| **CORAL + Frontier** (Frontier/Squid) | Conditions data<br>*(R/O access in Grid)* | Conditions, Geometry, Trigger<br>*(R/O access in Grid, HLT, Tier0)* | — |
| **CORAL Server** (CoralServer/CoralServerProxy) | Conditions, Geometry, Trigger<br>*(R/O access in HLT)* | — | — |
| **CORAL + LFC** (LFC authentication and lookup) | — | — | Conditions data<br>*(authentication/lookup in Grid)*<br>*(only until 2010)* |
| **COOL** | Conditions data | — | Conditions data |
| **POOL** (relational storage service) | — | Conditions, Geometry, Trigger<br>*(only until 2010)* | — |
| **POOL** (collections – ROOT/relational) | Event collections/tags | — | — |

- *CORAL, COOL and POOL are a joint development of IT-ES, ATLAS, CMS and LHCb*
- *For POOL: only included the components using relational databases, relevant for this works*
- *CORAL and COOL are also used by non-LHC experiments (Minerva at FNAL, NA62 at CERN)*

# CORAL "network glitch" issues

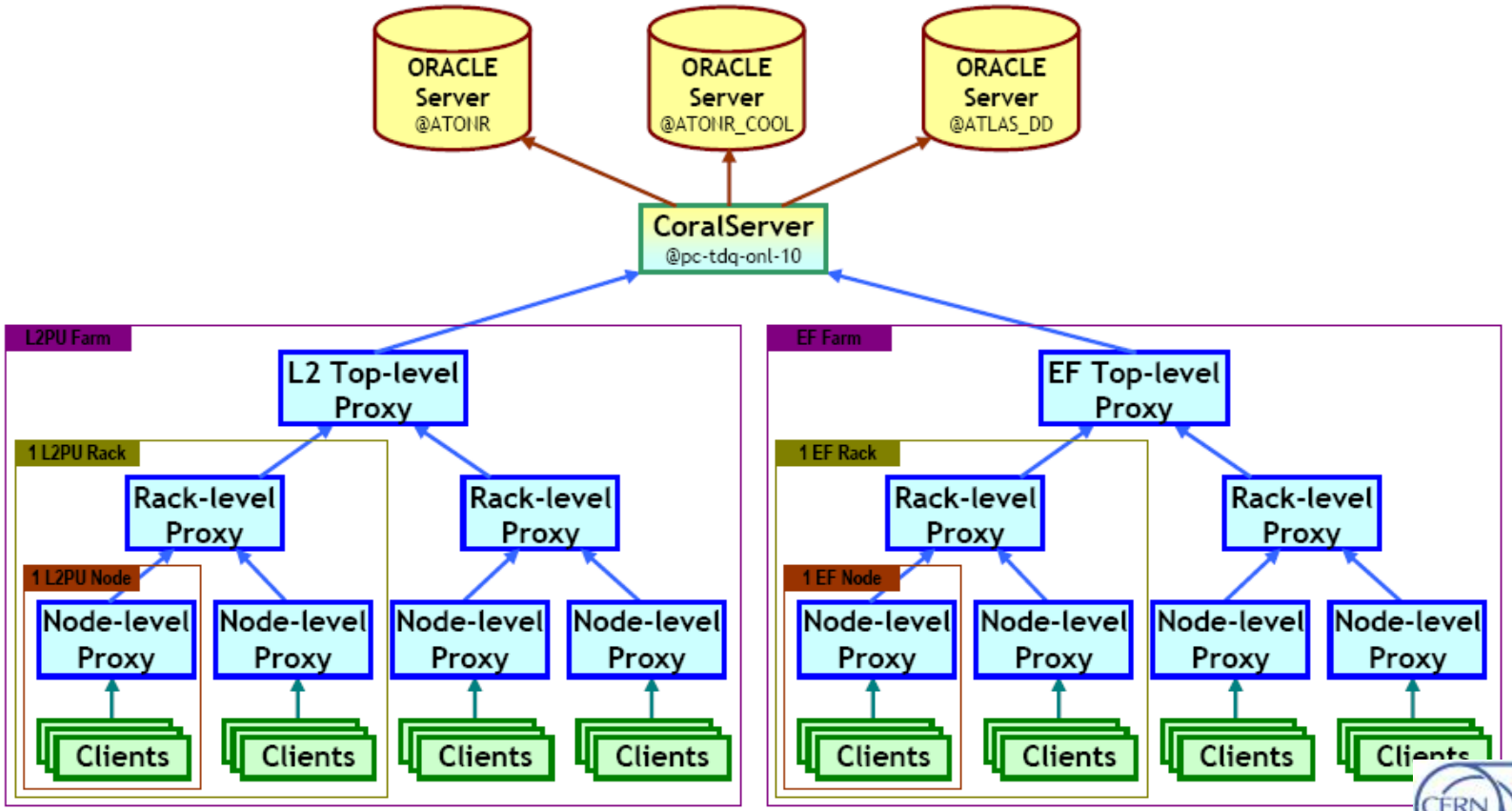


- **Different issues reported by all experiments**
  - e.g. ORA-24327 "need explicit attach" in ATLAS/CMS (bug #58522)
    - Fixed with a workaround in CORAL 2.3.13 (released in LCG 59b)
  - e.g. OracleAccess crash after losing session in LHCb (bug #73334)
    - Fixed in current CORAL 2.3.16 candidate (see below)
    - Similar crashes can also be reproduced on all other plugins
- **Work in progress since a few months** *(A.Kalkhof, R.Trentadue, A.V.)*
  - Catalogued different scenarios and prepared tests for each of them
  - Prototyped implementation changes in ConnectionSvc and plugins
- **Current priority: fix crashes when using a stale session**
  - May be caused both by network glitch and user code (bug #73834)!
  - A major internal reengineering of all plugins is needed (replace references to SessionProperties by shared pointers)
    - Done for OracleAccess ST in 2.3.16 candidate, pending for other plugins
    - The patch fixes single-thread issues; MT issues are still being analyzed
- **Next: address actual reconnections on network glitches**
  - e.g. non serializable R/O transaction: should reconnect and restart it
  - e.g. DDL not committed in update transaction: cannot do anything

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# CoralServer in ATLAS online

- **CoralServer deployed for HLT in October 2009**
  - Smooth integration, used for LHC data taking since then
  - No problems except for handling of network glitches




CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# Plans for CORAL server



- **Support usage in the ATLAS online system**
  - Requests and plans are in line with more general CORAL needs
    - Fix for the network glitch issue
    - CORAL monitoring of DB queries (both in server and proxies)
    - More detailed performance analysis and optimizations
- **Work on further extensions (e.g. for offline) is now frozen**
  - Interest from offline communities is limited or inexistent
    - Frontier is used for read-only use cases by both ATLAS and CMS
      - Also, likely synergy with CVMFS in the future for Squid deployment (http)
  - Possibly larger potential for extension than Frontier in other use cases, but no real need/request for these extended features
    - Authentication & authorization via X509 Grid proxy certificates
      - Already in CVS: will be released after cleanup of Globus integration
    - Database update functionalities (DDL/DML) – wont do
    - Disk resident cache (a la Squid) – wont do

# Oracle client libraries for CORAL



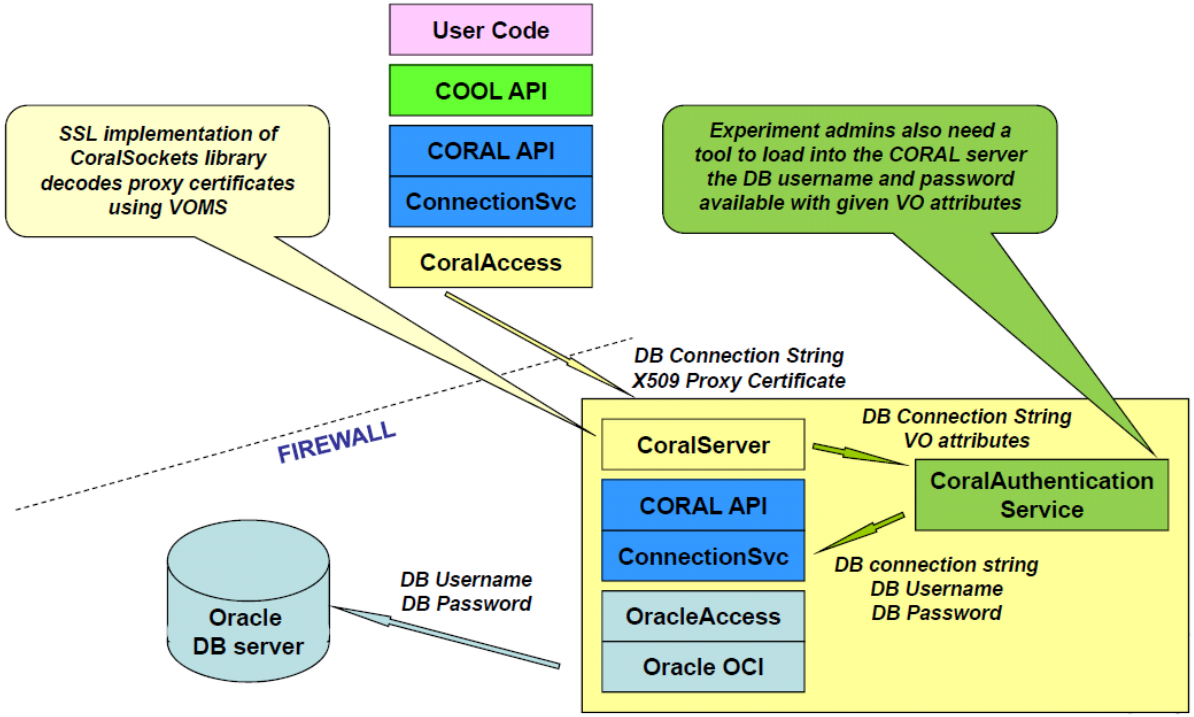
- **Oracle client for CORAL is maintained by the CORAL team**
  - Different installation (consistent with AA 'externals' on AFS)
  - Tailor-made contents
    - e.g. 11.2.0.1.0 patches to fix selinux and AMD quadcore bugs
    - e.g. sqlnet.ora customization for 11g ADR diagnostics
  - Close collaboration with Physics DB team in IT-DB on these issues

- **Two open issues in Oracle – plus a similar one in Globus**
  - All three are conflicts with the default Linux system libraries
    - Should either use the system libraries or use 'versioned symbols'
  - **Globus redefines gssapi symbols (bug #70641)**
    - Suggested to use versioned symbols: will be in the 2011 EMI release
    - Workaround: disabled gssapi from Xerces (used by CORAL)
  - **Oracle client redefines gssapi symbols (bug #71416)**
    - SR 3-1977807081 – gssapi in libclntsh.so conflicts with libgssapi_krb5.so
    - Suggested to use versioned symbols (Oracle bug 10184681)
    - No workaround needed so far (problem not yet seen in production…)
  - **Oracle client redefines kerberos symbols (bug #76988)**
    - SR #3-3620145421 – krb5 symbols in libclntsh.so conflict with libkrb5.so
    - Suggested to use versioned symbols (Oracle bug 12557209)
    - Workaround: will customize kerberos parameters in sqlnet.ora

CERN IT Department
CH-1211 Genève 23
Switzerland
**www.cern.ch/it**

# CoralServer secure access scenario






- **For comparison, if authentication uses the LFC replica service:**
  - Credentials are stored in LFC server (here: in Coral server)
  - Credentials are retrieved onto client by LFC plugin (here: stay in Coral server)
  - Credentials are sent directly by client to Oracle (here: sent by Coral server)
  - In both cases, credentials for Oracle authentication are username & password
    - No support of Oracle server for X509 proxy certificates
    - Could try using Kerberos authentication on Oracle server otherwise?