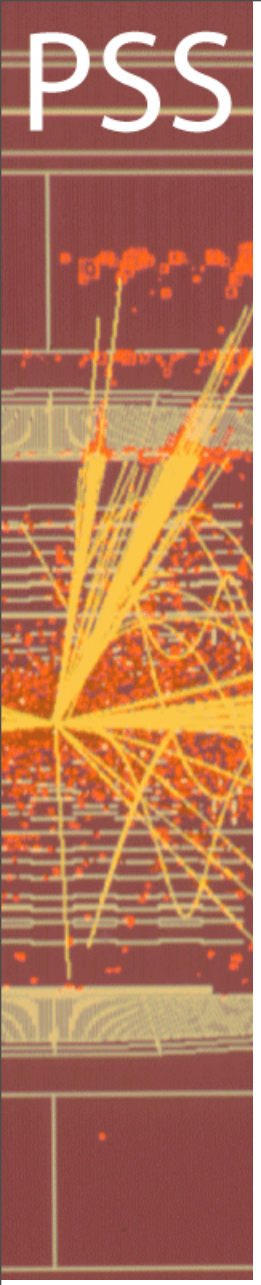
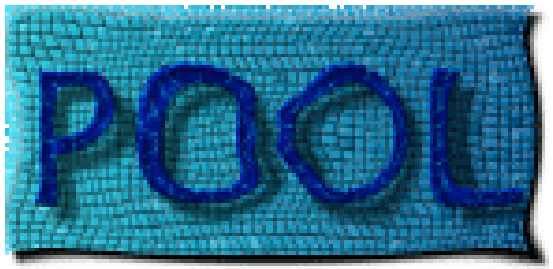




Persistency Framework Status

Dirk Düllmann, CERN IT
(on behalf of the persistency framework team)

LHCC Comprehensive Review
25-26 September 2006, CERN

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

Persistency Framework Parts

Three Software Packages

- CORAL
  - Abstraction of relational database access for Oracle, MySQL, SQLight and FroNTier
- POOL
  - C++ object persistency (via Root or Databases) and navigation (via catalogs and collections),
- COOL
  - Management of versioned conditions time series

- Layered and complementing each other
  - COOL and POOL use CORAL to access databases
  - Experiments use COOL to reference conditions objects stored in POOL

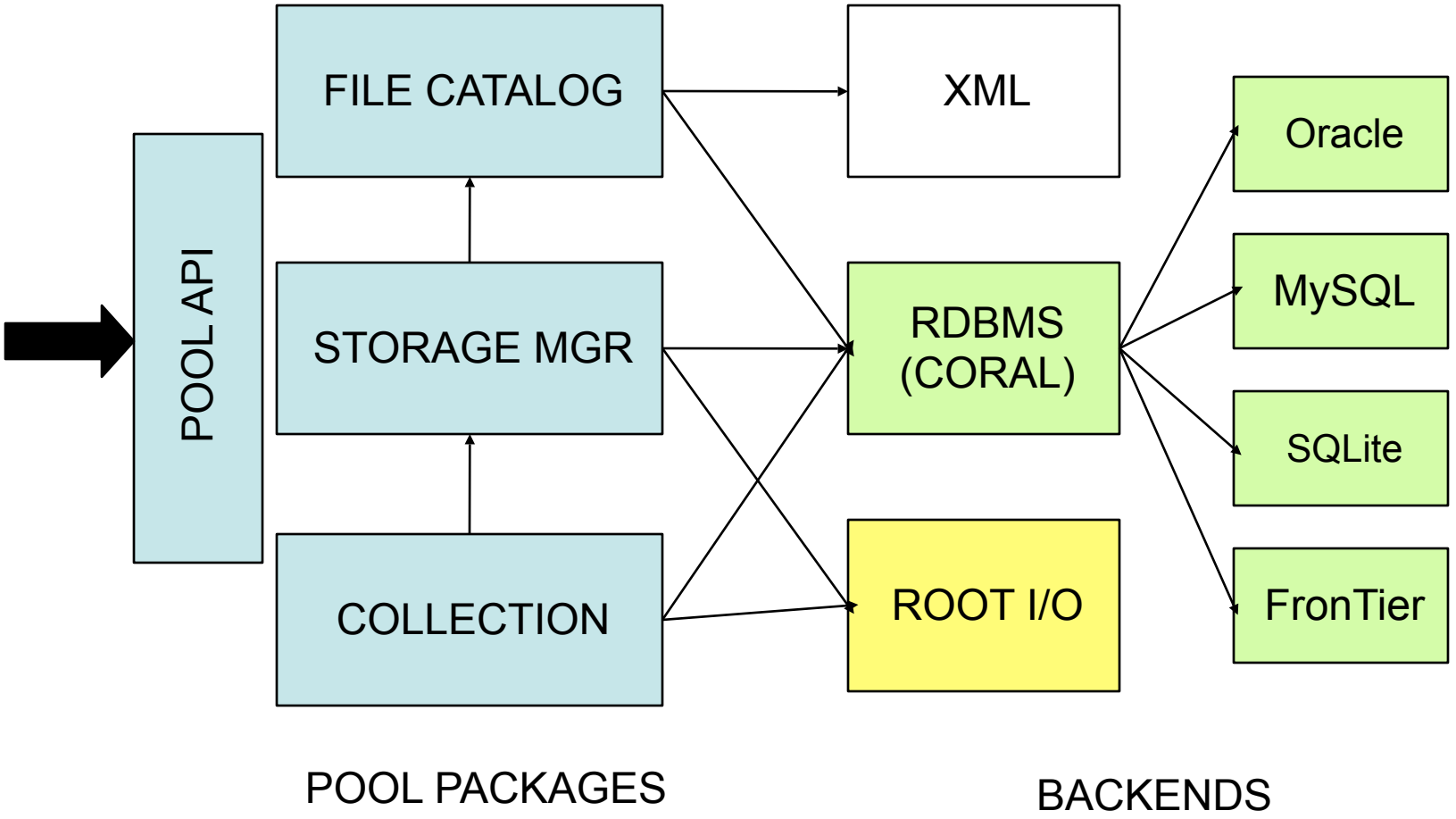
Dirk.Duellmann@cern.ch

PF - Main developments

- CORAL
  - Key importance for database deployment
    - DB lookup, monitoring, authorization, connection pool
    - encourage efficient db apps by API design (bind var's, effective use of db sessions)
- Allowed to factorise out database specific dependent code in POOL/COOL
  - Eg one COOL implementation instead of separate Oracle and MySQL flavours
  - Same code working now also against SQLight and FroNTier
- Picked up by experiment projects (offline and online)
- Integration with LCG 3D services
  - Database service at T0 and T1s
  - Replica management via existing LFC service

Dirk.Duellmann@cern.ch

Component interaction



Dirk.Duellmann@cern.ch

CORAL - Database Lookup and Indirection

- Database replica catalog mapping between
  - A logical db name: */my/conditions/data*
  - A physical connect string: *oracle://HostName/SchemaName*
- Until recently implemented based on XML file
  - Local or HTTP based access
- Now optionally hosted in an LFC catalog
  - Development in collaboration with RRCAT India
  - Ties in with existing LFC service
- Allows to manage available database replicas
  - command line tools to add/remove/change replicate
  - authorisation based on LFC ACLs (VOMS roles)

Dirk.Duellmann@cern.ch

Changes in PF Underpinnings

- POOL
  - Consolidation and maintenance phase
  - Few functionality request / bug reports
- Significant work went into absorbing changes of the core infrastructure
  - Components moving from SEAL into ROOT
    - Reflex support Dec 05
  - Interface changes because of different coding conventions between SEAL and ROOT
  - Dependency changes
    - Change of exception base class with ROOT Reflex
- POOL user code has stayed largely stayed unaffected

Dirk.Duellmann@cern.ch

PF and Grid Components

- File Catalogs
  - Good news: consolidation on fewer grid catalogs
    - LFC seems to be established as baseline
    - CMS also developed & uses “Trivial File Catalog”
  - POOL did not have to invest in new catalog integration
- Configuration management between apps and deployment area still an issue
  - Versions numbers & compatibility matrix need to be kept in sync (lfc, gfal, root and gfal plug-ins)

Dirk.Duellmann@cern.ch

POOL Storage Manager status

- New functionality added: dictionary auto-loading
  - Enables the loading-on-demand of the required dictionary libraries at run time
  - Code completed since POOL_2_2_6 (Atlas contribution)
- Cache and persistency service basically unchanged (apart from Reflex business)
  - Added command lines for the extraction/handling of file ID from POOL databases
- Root backend in maintenance mode
  - Adopted by Atlas, CMS and LHCb
  - Following up Root releases (from 4.X to 5.X)
  - Backward compatibility in reading mode tested in every release cycle (data regression test)
  - Few bug fixes

Dirk.Duellmann@cern.ch

Relational Storage Manager

- Relational StorageSvc (ORA) fully functional
  - Supports all of the CORAL backends
  - Adopted by Atlas and CMS
    - CMS in deployment phase
- New features added
  - Functionality to set up a POOL database from an existing set of relational tables
    - Command line tools based on XML driver file
  - Blob based storage for containers
    - Activated as an option, via user defined mapping
    - Using customizable streamer
    - Could be extended to arbitrary objects
- Next development priority
  - Schema evolution handling
    - Use cases need to be identified
    - Resource allocation and priority defined

Dirk.Duellmann@cern.ch

COOL – evolution 2004-2005

Initial developments

- *Nov-04: start of COOL development*
- Dec-04: single-version mode (bulk object read/write)
- Feb-05: performance studies (disable scrollable cursors)
- Feb-05: multi-version mode with tagging
- Mar-05: int64 validities, handling of SQL types

Software releases *(omitting most bug fixes)*

- COOL 1.0.0 (Apr-05): first public release
- COOL 1.0.1 (Apr-05): bug fixes
- COOL 1.0.2 (May-05): listChannels, listTags, new CVS
- COOL 1.1.0 (May-05): basic multi-channel bulk insertion
- COOL 1.2.0 (Jun-05): IFolderSet, setDescription, performance improvements, AuthenticationService
- COOL 1.2.1 (Jul-05): untag/retag, SQLite, Examples, Oracle privilege mgmt, RAL fixes (pthread lock, #open cursors)
- COOL 1.2.2 (Jul-05): bug fixes in SEAL
- COOL 1.2.3 (Aug-05): PyCool, multi-channel bulk retrieval, CLOB, user guide

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

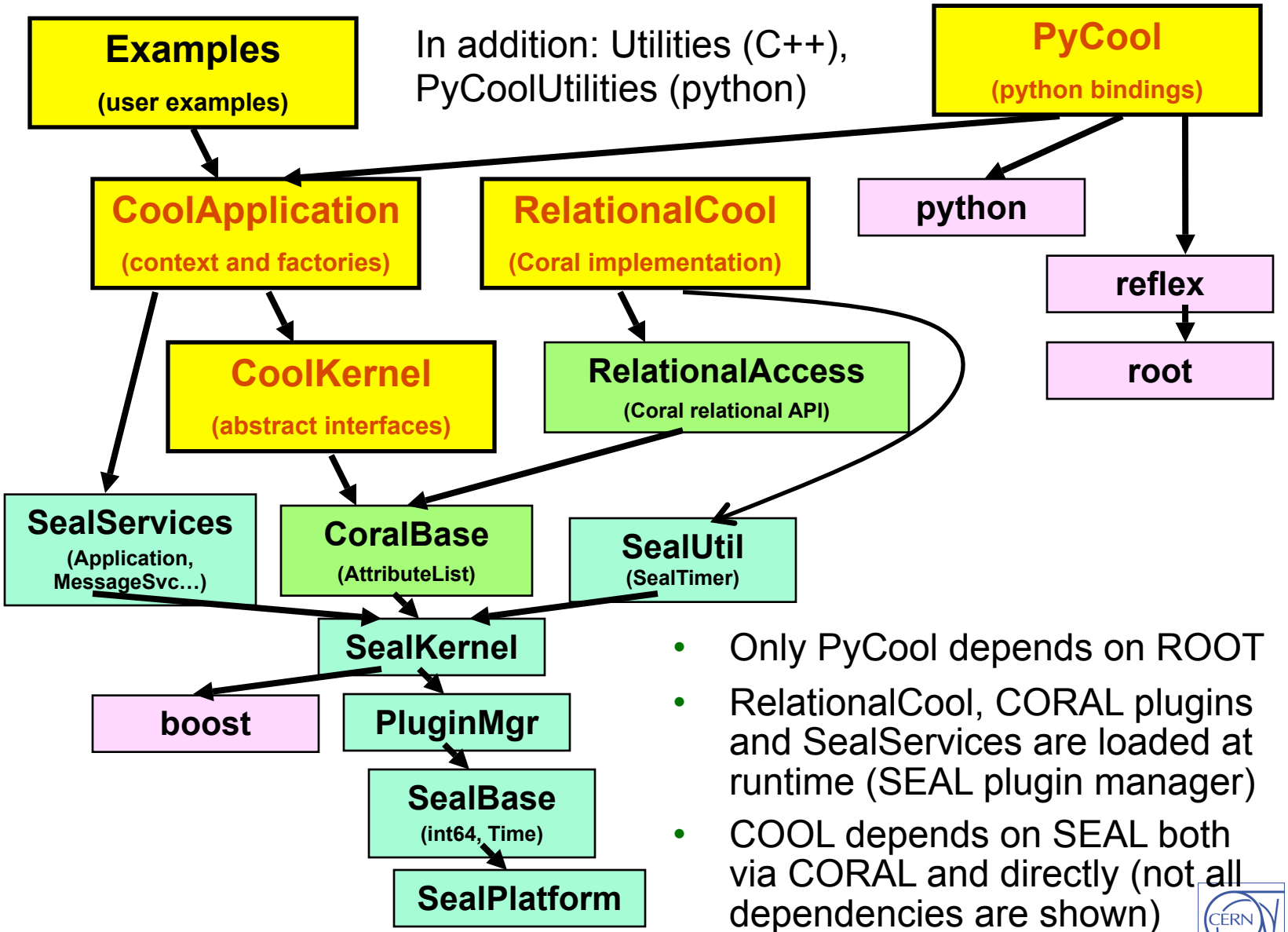
COOL – evolution 2005-2006

Software releases *(omitting most bug fixes)*

- COOL 1.2.4 (Sep-05): RAL bug fixes (memory leaks)
- COOL 1.2.5 (Oct-05): tools for Oracle table statistics
- COOL 1.2.6 (Nov-05): bulk retrieval with server cursor, PyCoolUtilities, closeDatabase, countObjects
- COOL 1.2.7 (Jan-06): RAL migration, Reflex migration, Wine
- COOL 1.2.8 (Jan-06): ROOT migration, gcc344, AMD64 test
- COOL 1.2.9 (Mar-06): new CORAL API
- COOL 1.3.0 (Apr-06): ConnectionService, HVS, user tags, schema evolution, performance fixes, std::exception
- COOL 1.3.1 (Apr-06): bug fixes
- COOL 1.3.2 (Apr-06): Frontier, bug fixes
- COOL 1.3.2a (May-06): rebuild
- COOL 1.3.2b (Jun-06): rebuild
- COOL 1.3.2c (Jul-06): AMD64 bug fixes in ROOT/CORAL
- COOL 1.3.3 (Aug-06): Frontier bug fixes, SEAL bug analysis, performance report

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

COOL - s/w dependencies (1)



- Only PyCool depends on ROOT
- RelationalCool, CORAL plugins and SealServices are loaded at runtime (SEAL plugin manager)
- COOL depends on SEAL both via CORAL and directly (not all dependencies are shown)

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

COOL - s/w dependencies (2)

- Fine granularity of package dependencies
  - Possible thanks to the use of SCRAM (or CMT)
    - 'scramShowUses' (COOL tool), 'cmt show uses'
    - *Please keep SEAL/CORAL substructure if migrate to CMT…*
  - *Only need to rebuild COOL if specific packages change*
    - E.g. no need to rebuild if CORAL plugins change
  - *Only need to build relevant SEAL/CORAL packages*
    - Private SEAL/CORAL builds routinely used for debugging
- ROOT dependency (only) via PyCool
  - COOL rebuild release often needed when ROOT changes
  - It would be nice to clarify which ROOT packages Reflex depends on - and also to minimise these dependencies
- SEAL plugin manager and component model
  - Positive experience (in spite of some bugs - being fixed)
  - Stability has been a plus - concern if ROOT migration

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

PF and AA Releases

- AA schedule driven by ROOT bug fix releases
  - which are driven by experiment requests
- Release frequency increased and latency decreased
  - Often just rebuild releases for POOL and SEAL
  - Also the warning time in the planning cycle decreased
    - CORAL and COOL have to fit new developments in
- Aim to further streamline the process
  - PF is high in s/w stack and exposed to code stability and release granularity of many lower level packages
  - Testing early (low down in the s/w stack) would help to further improve “time-to-user”
    - Moved some tests from COOL into CORAL
    - Propose to move some I/O related tests down into ROOT
- We can work with any proposed build-system, but expect the support of this system not to be in PF

Dirk.Duellmann@cern.ch

PF and SEAL components

- Preparing for the next round of core changes
- PF components rely on SEAL component model
  - Strategic choice to be prepared for a possible plug-in mgr change
  - Actively involved in SEAL bug fixing (eg threading issues)
- Will be affected by AA evolution in this area
  - Just one more standard compromise between
    - Removing dependencies by introducing code copies
    - Removing code duplications by introducing dependencies
  - PF has provide a short list of used SEAL components
- Propose to setup a table of SEAL components and plan their evolution

Dirk.Duellmann@cern.ch

PF Manpower

- POOL/CORAL - Moved to maintenance and improved service integration
  - Activity increasingly IT based - experiment fraction lowered
  - Need to insure sufficient experiment participation to insure efficient debugging inside experiment frameworks
  - Limited manpower left also in other areas
    - Collections - main ATLAS developer left and tasks moved to remaining person
- Short term - COOL manpower dropped to 1FTE
  - two very effective ATLAS contributors left the project
- Medium term - stability of expertise in the persistency framework
  - all IT based developers face a contract review next year

Dirk.Duellmann@cern.ch

Instead of a summary: some quotes from the internal review

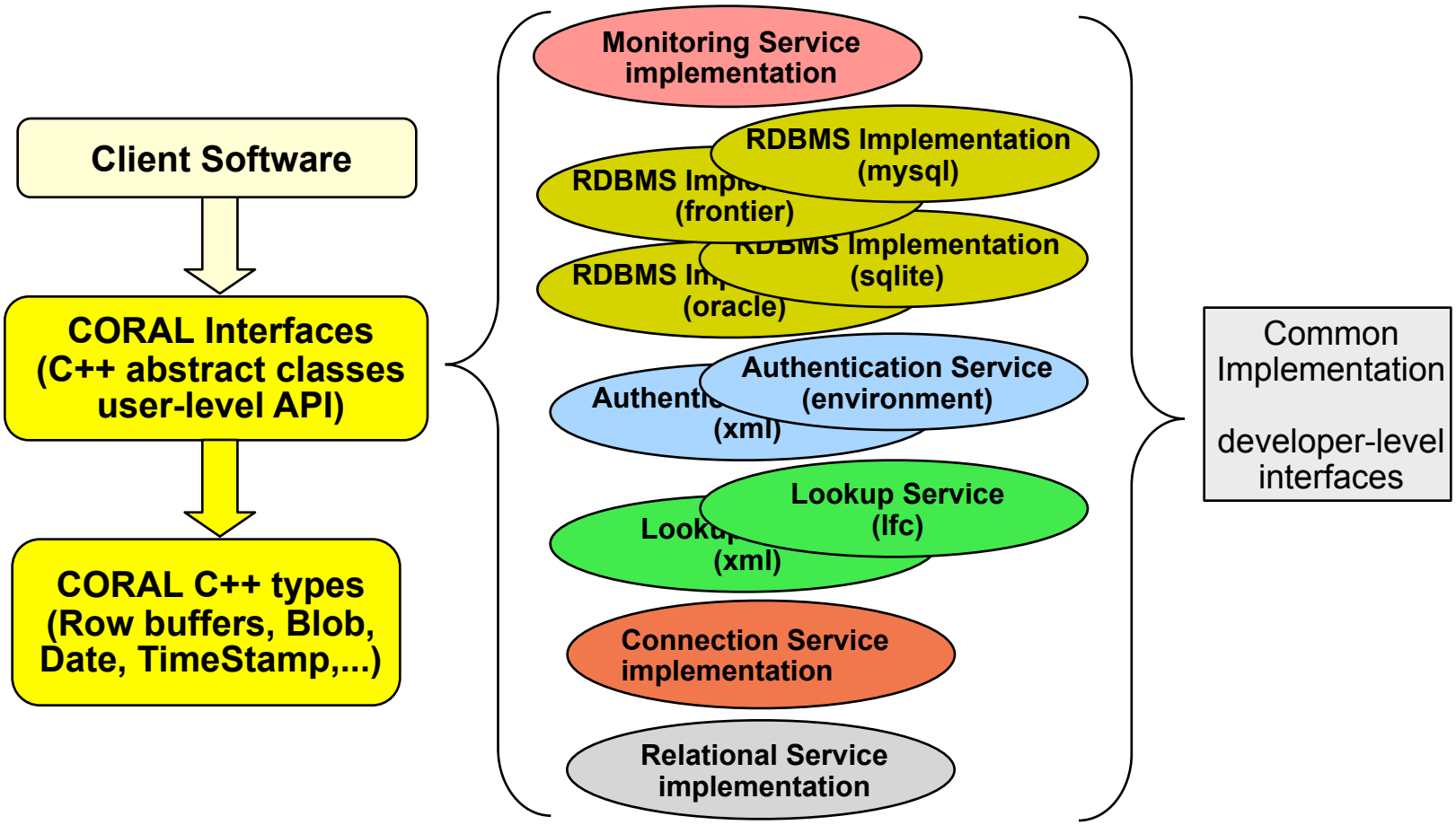
- POOL
  - Stable/mature product, continued maintenance is essential
  - Need to add schema evolution also for relational data
- CORAL
  - Separation from POOL works well, endorse new frontier back-end
  - Reaching maturity, emphasis should now be on stability
- COOL
  - Manpower situation critical (internal AA move recommended)
  - Continue work on remaining scalability issues

Dirk.Duellmann@cern.ch

Backup Slides

Dirk.Duellmann@cern.ch

Component Architecture (I)



Plug-in libraries, loaded at run-time, interacting only through the interfaces.

Dirk.Duellmann@cern.ch

RDBMS Back-End Technologies

- Oracle
  - Fully implements the CORAL API and all internal optimizations:
    - Row prefetching, bind variables, server-side cursors,…
  - Based on OCI (C client library) 10.2.0.2
- MySQL
  - Better suited where only a low level of administration can be afforded
  - Based on the C client library version 5.0
- SQLite
  - File-based; no administration required
  - Based on the C client library version 3.3.5
- FroNTier
  - Squid caches between client and Oracle database
  - Suitable for read-only data
  - Implies constraints on the data deployment model (as data may become stale)

Dirk.Duellmann@cern.ch

Component Architecture (II)

- Interactions across the various models is based on the SEAL component model.
- Advantages of this architectural choice
  - Client code depends on a very thin and lightweight software stack
    - CoralBase (Data Types, Row Buffers, 195 KB)
    - RelationalAccess (API, Exception Hierarchy, 115 KB)
  - Efficient unit testing and bug tracking
  - Most releases are binary compatible to the previous one:
    - The core of the implementation is in run-time loadable components
    - So far 16 releases
      - All of them backwards compatible (no need for change in user code, incremental extensions of the API)
      - 11 of them fully binary compatible (no need even to recompile the user code)

Dirk.Duellmann@cern.ch

From the last review until now

- Recommendations in the last internal review:
  - "The reviewers welcome the (proposed) split of the RAL and POOL release cycles"
    - CORAL is now released independently of POOL.
  - "Improve error handling; error reporting must propagate to end user with clear indication of which component in the complicated stack encountered the error, and provide sufficient description of the error"
    - A complete exception hierarchy in CORAL reporting on the error conditions and the modules throwing the exception.
    - CORAL architecture based on a thin component stack.
  - "(Security) In a distributed (grid) environment, POOL should not be the weakest point in the chain"
    - Using grid certificates in order to retrieve the authentication credentials (via LFC)

Dirk.Duellmann@cern.ch

Authentication Mechanisms

- CORAL connection strings tell only about the location of a data source
  - mysql://HostName/DatabaseName
- Data source access credentials not exposed
  - UserName="Me", Password="ThEpAsSwOrD"
  - Credential parameters are retrieved from different software components given the bare connection string
- Two simple implementations currently:
  - Credentials from environment variables
  - Credentials stored in XML files
  - Used for application development and prototyping
- Secure implementation based on LFC is being developed
  - Authentication based using GRID certificates
  - Credentials linked to "database roles" and controlled by the LFC ACL mechanisms
  - Extension of the LFCLookupService
  - No need for an extra service (again…)!

Dirk.Duellmann@cern.ch

Client-Side Monitoring

- Reasons for client-side monitoring
  - complements the server-side monitoring
  - assists application debugging and tuning
- Implementation strategy
  - CORAL API defines the interfaces for the call-back objects that are called by the RDBMS plugins
  - Information is pushed from the system to the monitoring implementation
    - Session and transaction time boundaries
    - Time duration that the client waits on the server to execute an SQL command
    - The SQL commands themselves
  - Current implementation serves as an example
    - Experiments are expected to implement their own plugins that are coupled to specific monitoring systems

Dirk.Duellmann@cern.ch

CORAL in Use and Outlook

- CORAL users
  - ATLAS
    - Via POOL and COOL
    - Direct usage from detector geometry and on-line applications
  - CMS
    - Via POOL (RelationalFileCatalog, Object-Relational Access)
    - Direct usage from conditions database and on-line applications
  - LHCb
    - Via POOL and COOL
- Main priorities
  - LFC-based component for authentication
  - Focus on connection handling policies
  - Make CORAL thread-safe
  - Follow requirements from the 3D activities and implement the necessary solutions
  - Python interface to CORAL
    - In cooperation with RRCAT, India

Dirk.Duellmann@cern.ch

Overall changes

- LCG dictionary evolution followed up (strong dependency)
  - Reflection replaced by Reflex
  - Reflex moved into Root
  - Storage Manager and Collection affected
- CORAL package factored out
  - Affected the Relational backends for all of the 3 domains
  - Migration transparent in the implementations
  - Transition phase keeping POOL::AttributeList in the public interfaces
    - Affected Catalogue and Collection
    - Will be soon replaced by CORAL::AttributeList
    - New code in the repository, needs to be validated from the experiments
  - Aligned with coral upgrades
    - Relational component adapted to use Connection Service

Dirk.Duellmann@cern.ch

File catalogue

- Key component, used from the experiment applications
  - Not only in the context of the POOL object storage
- Back-ends for grid connectivity evolved in parallel to middleware
  - EDG implementation phased out
  - LFC, Globus and Glite adaptors for POOL catalogue initially released
    - Implementation provided by the Grid developers
    - Performance comparison and benchmarks provided by the experiments (ARDA team)
  - LFC selected following the experiment recommendations
    - Built-in security based on grid certificates
    - widely used by the experiments
    - maintained actively by Grid deployment group
- Other back-ends
  - Generic RDBMS implementation based on CORAL still available
    - Supports all of the CORAL back-ends
    - SQLite and FronTier considered for caching
  - XML implementation being re-engineered

Dirk.Duellmann@cern.ch

Collections

- A common interface for two ways to define a set of persistent objects:
  - Implicit Collection
    - Defined 'by containment' in a POOL container.
    - Allows to navigate through the object stored in a given database (file or RDBMS table)
    - Adopted by Atlas and CMS
    - Developed and maintained by the POOL Storage Manager team
  - Explicit Collection
    - Defined externally, as a user-defined object set.
    - Convenient for metadata-based selections
    - Back-ends available in RDBMS, Root
    - Developed and maintained by Atlas, in scope with their specific requirements
- Some improvements completed
  - Back-end neutral utilities (command line) upgraded
    - Added new functionalities, improved parameter granularity
- Others are foreseen
  - Review of the API in order to allow better scalability (Explicit Collections)

Dirk.Duellmann@cern.ch

Issues from March2005 review

- **"COOL is very young and has yet to succeed"**
  - Several production releases since then
  - Being deployed in Atlas and LHCb (success in Pere's terms!)
- **"Commitment from Atlas and LHCb to use COOL; CMS is also considering using COOL"**
  - Continued commitment (~and manpower) from Atlas/LHCb
  - COOL not used by CMS afaik (but no clear statement yet)
- **"Experiments interested in COOL should commit more manpower to ensure project survival"**
  - Manpower was adequate (not abundant) in 2005-2006
  - *Present manpower situation critical again since June 2006* (both Atlas collaborators left - no more Atlas contribution)

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

COOL - backend support

- **Oracle**
  - From the start (replace old CondDBOracle)
  - *Main focus of development and performance optimization*
- **MySQL**
  - From the start (replace old CondDBMySQL – "Lisbon API")
- **SQLite**
  - Since COOL 1.2.1 (July 2005)
- **Frontier**
  - Since COOL 1.3.2 (May 2006)
  - Read-only backend - cannot reuse the same test suite
    - A lot of work in the last two months and still in progress
- Same relational schema for all backends
  - A choice (for COOL-independent copies), not a necessity
    - Cross-backend replication (e.g. Octopus) not tested yet

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

COOL - software releases

- **COOL development started in November 2004**
  - *First production release 1.0.0 in April 2005*
  - *Latest production release 1.3.3 in August 2006*
- Current release numbering *(after some evolutions)*
  - Minor release (e.g. 1.3.0) for important enhancements with schema changes and backward-incompatible API changes
  - Bug-fix release (e.g. 1.3.2) for enhancements with no schema changes and backward compatible API changes
  - Rebuild release (e.g. 1.3.2a) for rebuilds using the same source code and changes only in external dependencies
- 21 software releases so far – details later
  - 4 minor releases, 14 bug-fix releases, 3 rebuild releases
  - Rebuild releases are a very recent concept (1.3.2a-b-c)

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

COOL - CORAL feedback

- **Tight collaboration with CORAL**
  - Excellent interaction with CORAL team
  - Reuse of configuration infrastructure
  - COOL-oriented enhancements of CORAL
  - Fast support and bug fixes
  - *"Hidden" team – CORAL does a lot of the COOL job!*
- **CORAL thoroughly tested within COOL**
  - Same code tested against all 3 (or 4) backends
    - All available data types are routinely tested
  - Contributed many bug reports (and many fixes)
  - *Mutually beneficial collaboration!*
- **Benefit from separation of RAL from POOL**
  - *No software dependency of COOL on POOL*

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

COOL development team

Active manpower

- **A.V. (CERN - IT/PSS)**
  - 80% FTE since Oct. 2004
  - Project coordination, core development and release mgmt
- **Marco Clemencic (CERN - LHCb)**
  - 20% FTE since Nov. 2004
  - Core development and release mgmt
- With useful contributions from many other people
  - David Front (IT/LCG since April 2005) - stress tests
  - Richard Hawkings and other ATLAS users, testers and DBAs
  - The CORAL, SEAL/ROOT, SPI and 3D teams

Former collaborators

- **Sven A. Schmidt (Mainz - ATLAS)**
  - 80% FTE (Oct. 2004 to June 2006) - back at 20% FTE in October?
  - Core development
- **Uli Moosbrugger (Mainz - ATLAS)**
  - 80% FTE (Sep. 2005 to March 2006) – performance optimization

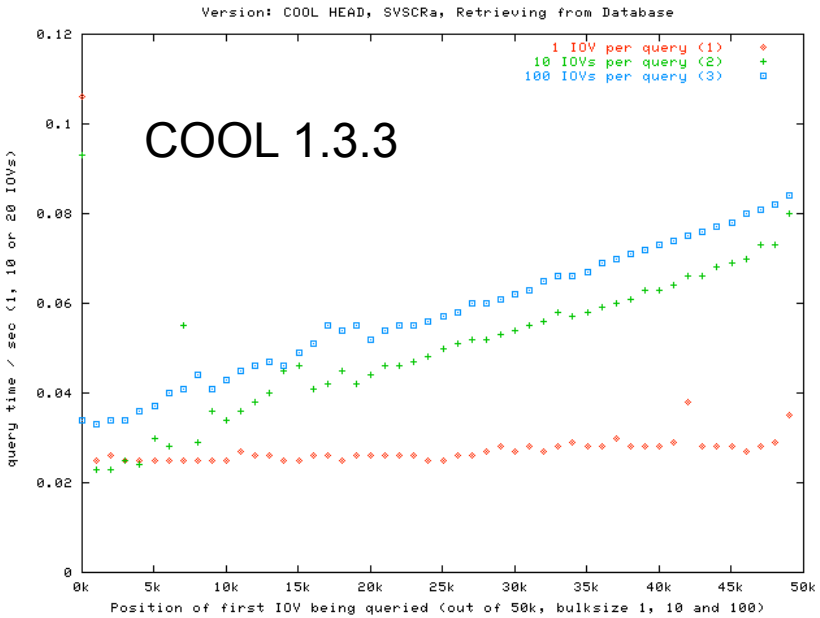
CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

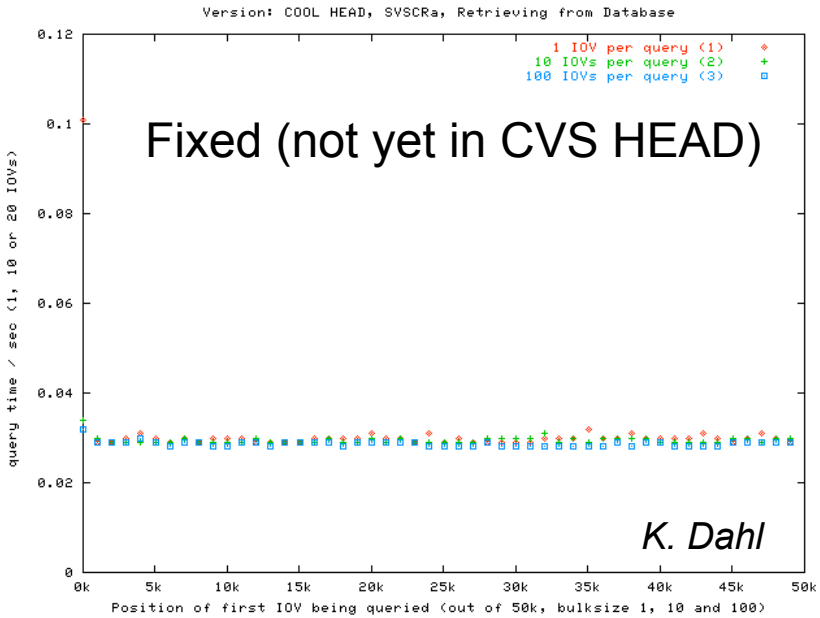
COOL - performance (1)

- Studied both query optimization and use case validation
  - Focus is Oracle only – not MySQL or SQLite (no manpower…)
- *Scripts to compute table statistics since COOL 1.2.5 (Oct 2005)*
  - Understood tricky features of Oracle execution plans
- Atlas prompt reconstruction validation studies (October 2005)
  - *Achieved 20 MB/s and 20k rows/s sustained read-back rates*
  - See CHEP2006 poster for details
- *Performance report since COOL 1.3.3 (Aug 2006)*
  - Example: report for COOL 1.3.3 linked to COOL web page
  - Includes plots for several pre-defined use cases (e.g. single/multi-version insertion/retrieval from single/multi-channel folders)
    - Most of the issues shown in the COOL 1.3.3 report are pending
    - Some instead are fixed in CVS HEAD or in private test code

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

COOL - performance (2)





- Example: query time for single-version single-channel retrieval of all IOVs between two time points t1 and t1+Δt
  - Query time increases with increasing values of t1
  - Fix prototyped in private COOL code, but not yet in CVS
  - Effect previously observed (now fixed) to fetch one IOV at time t1

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

COOL - data distribution

- Replication at the database backend level
  - **Oracle Streams** technology (tested and deployed within the 3D project) - see slides about Atlas and LHCb
  - Cross-technology replication is possible in principle (same schema for all backends), not really considered/tested yet
- Oracle remote access via **Frontier**
  - Intermediate Squid web caches
  - *Work in progress (functional/performance tests with Squid, cache optimization for specific use cases…) – Atlas tests*
- Replication using tools based on the COOL API
  - Data slicing/selection is also possible
  - Cross-technology replication (e.g. to **SQLite files**)
  - *Work in progress (dynamic replication) - COOL 1.4.0*

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

COOL (1/2)

- ATLAS making extensive use of COOL 1.3 in both online and offline
  - Store calibration, alignment, online configuration and DCS data
  - Over 50GB of COOL data from online detector commissioning (and reco/analysis), offline data challenges and legacy combined testbeam data
  - More and more ATLAS software is making use of COOL
  - A success!
- We're very concerned about COOL manpower situation
  - Several medium term requests, channels data, multichannel bulk insert, payload queries, full Frontier support (almost there), etc., have been around for 6-12 months but not yet addressed
  - ATLAS lost very active developer (Sven Schmidt) so some of the shortfall is result of this
- No realistic medium term plan/schedule
  - Mainly due to manpower shortfall
  - Many deliverables have been a few months away for 6 months or more
  - COOL development significantly affected by changes in LCG Infrastructure (SEAL, ROOT)
  - Concern about further disruptions
    - E.g. SEAL plug-in manager

COOL 2/2

- Some longer term concerns about scaling of COOL schema model
  - 1 COOL folder = 1-3 COOL tables
  - Optimizations proposed by ATLAS Oracle experts, but…
    - Lack of manpower again
    - Potentially very disruptive because of radically changed schema
- Bottom line(s):
  - ATLAS heavily committed to and dependent on COOL
  - COOL team heavily hampered by lack of manpower
  - ATLAS very concerned about rate of progress

CMS Condition DB

- **CMS Condition DB is based on Pool/ORA architecture**
  - Developed in close connection with the Pool team
    - Requirements, Design, Implementation
  - Rapid, positive bi-directional feedback
    - New features, performance improvements
    - bug hunting and fixes, support
- **Goal is an optimal blend of RDBMS tools and OO architecture**
  - Strength of RDBMS (back-end and front-end tools)
  - Persistent-object modeling common with event-data
  - Excellent results so far, very promising, architecture not frozen yet.
  - Must keep the flexibility of this “two faces” approach to allow evolution (and fall-back)

Persistency

- POOL
  - No need for File Catalog support besides XML (gfal ROOT plugin)
  - Stable and reliable
  - LHCb contributes to storage manager (M.Frank)
- CORAL
  - No direct dependency
- COOL
  - Basis for our Conditions DB
  - LHCb contributes to COOL Core development (M.Clemencic)
  - Framework completed for seamless update of calibrations and alignment from Conditions
  - Migration from XML files to CondDB to take place in coming months
    - SQLite slice used for simulation
    - ORACLE DB used at Tier1s & CERN for reconstruction / analysis