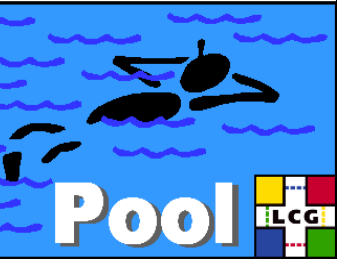


POOL Project Status & Plans

Dirk Düllmann,
IT-DB & LCG-POOL

LHCC Comprehensive Review of the LCG Application Area

25 November 2003

What is POOL?

- Project Goal: Develop a common Persistency Framework for physics applications at the LHC
  - Pool Of persistent Objects for LHC
- Part of the LHC Computing Grid (LCG)
  - One of the first Application Area Projects
- Common effort between LHC experiments and CERN IT-DB group
  - for defining its scope and architecture
  - for the development of its components

POOL Objectives

- To allow the multi-PB of experiment data and associated meta data to be stored in a distributed and Grid enabled fashion
  - various types of data of different volumes (event data, physics and detector simulation, detector data and bookkeeping data)
- Hybrid technology approach, combining
  - C++ object streaming technology
    - Root I/O for the bulk data
  - Transactionally safe Relational Database (RDBMS) services,
    - MySQL for catalogs, collections and meta data
- In particular POOL provides
  - Persistency for C++ transient objects
  - Transparent navigation among objects across file and technology boundaries
    - Integrated with a external File Catalog to keep track of the file physical location, allowing files to be moved or replicated

POOL Timeline and Statistics

- POOL project started April 2002
  - Ramping up from 1.6 to ~10 FTE
  - Persistency Workshop in June 2002
  - First internal release POOL V0.1 in October 2002

- In one year of active development since then
  - 12 public releases
    - POOL V1.4.0 is just being released
  - Some 60 internal releases
    - Often picked up by experiments to confirm fixes/new functionality
    - Very useful to insure releases meet experiment expectations beforehand
  - Handled some 165 bug reports
    - Savannah web portal proven helpful

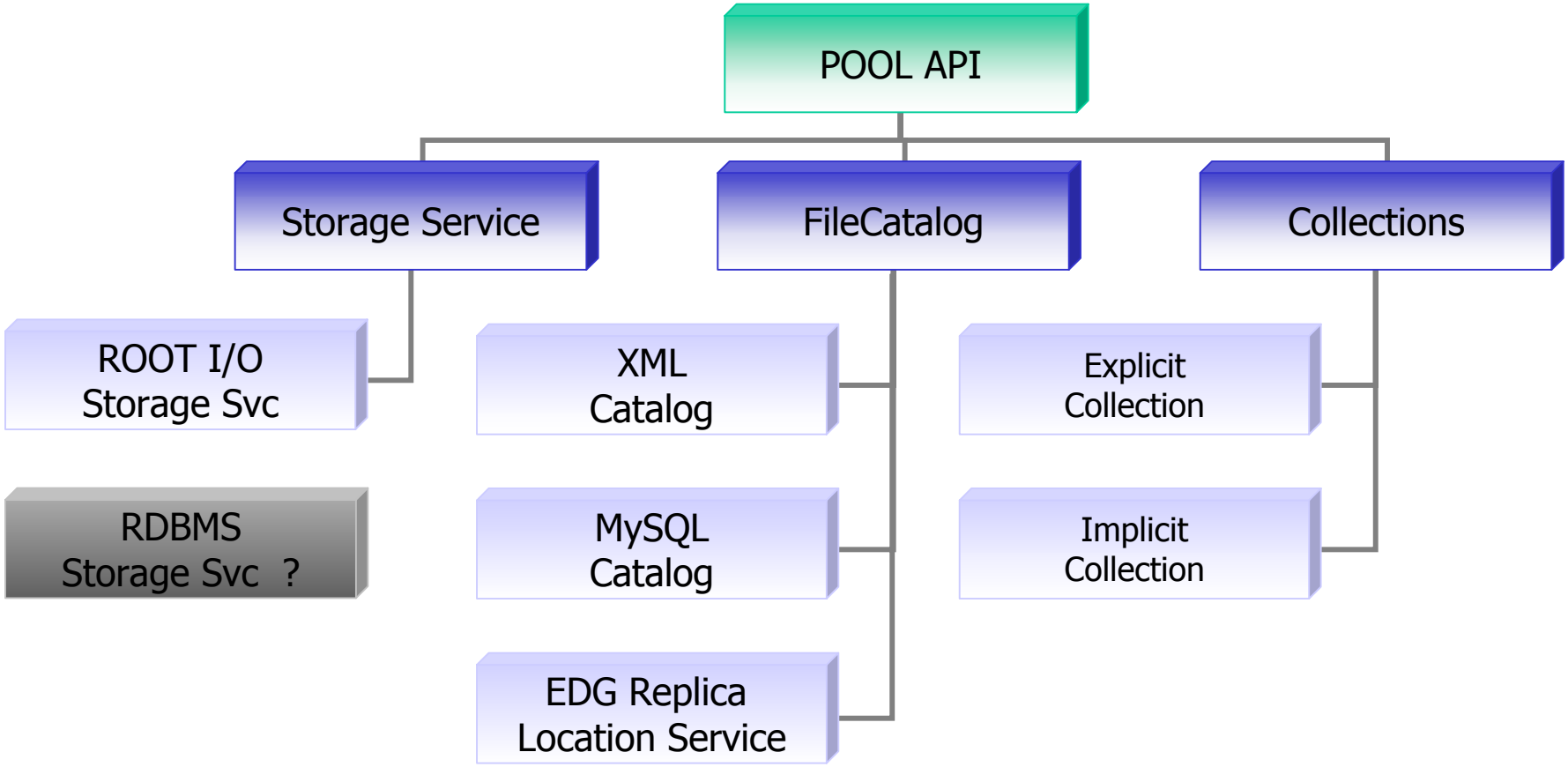
- POOL followed from the beginning a rather aggressive schedule to meet the first production needs of the experiments.

Component Architecture

- POOL is a component based system
  - follows the LCG Architecture Blueprint
- Provides a technology neutral API
  - Abstract component C++ interfaces
  - Insulates the experiment framework user code from implementation details of the technologies used today
- POOL user code is not dependent on implementation libraries
  - No link time dependency on implementation packages (e.g. MySQL, Root, Xerces-C..)
  - Backend component implementations are loaded at runtime via the SEAL plug-in infrastructure

- Three major domains, weakly coupled, interacting via abstract interfaces

POOL Component Breakdown



Work Package Breakdown

- Storage Manager
  - Streams transient C++ objects into/from disk storage
  - Resolves a logical object reference into a physical object
  - Uses Root I/O for event data, a proof of concept with a RDBMS storage manager prototype underway for other meta data
- File Catalog
  - Maintains consistent lists of accessible files (physical and logical names) together with their unique identifiers (FileID), which appear in the object representation in the persistent space
  - Resolves a logical file reference (FileID) into a physical file
- Collections
  - Provides the tools to manage potentially (large) ensembles of objects stored via POOL persistence services
    - Explicit: server-side selection of object from queryable collections
    - Implicit: defined by physical containment of the objects

POOL Milestones

- First "Public" Release - V0.3 December '02
  - Navigation between files supported, catalog components integrated
  - LCG Dictionary moved to SEAL – and picked up from there
    - Basic dictionary integration for elementary types
- First "Functionally Complete" Release - V1.0 June '03
  - LCG dictionary integration for most requested language features including STL containers
  - Consistent meta data support for file catalog and event collections (aka tag collections)
  - Integration with EDG-RLS pre-production service (rlstest.cern.ch)
- First "Production Release" - V1.1 July `03
  - Added bare C++ pointer support, transient data members, update of streaming layer data, simplified (user) transaction model
  - Due to the large number of requests from integration activities still rather a functionality release than the planned consolidation release.
  - EDG-RLS production service (one catalog server per experiment)
- Starting from POOL V1.3
  - (Being) Integrated with three experiment software frameworks
  - Successfully deployed in larger scale experiment productions

- Project stayed close to release data estimates
  - Maximum variance 2 weeks
  - Usually release within a few days around the predicted target date

POOL - Known Issues

- Need to improve on end-user documentation
  - Prepared a first user guide with V1.4
    - General overview of the POOL architecture
    - collecting some the experience gained during the framework integrations
  - Expanding the set of example programs and prepare a hands-on tutorial
    - POOL tutorial held in during the GridKa Computing school -> CSC 04
- Testing is not perfect..
  - .. and will probably never reach the complexity of the tests from within the experiment applications.
  - 60 functional and integration tests are executed in an automated way each release cycle
  - Feature requests now often come as a complete test case from the experiment. Thanks!
- Performance optimisation not yet fully addressed
  - Performance tests now exist for all components (addressed in June release)
  - External design and code reviews setup for use of ROOT I/O and for Object cache
- Schema Evolution and Stability of File Format
  - Current strategy relies fully on ROOT I/O facilities
    - The use of ROOT I/O as black box makes more generic schema evolution support non trivial
  - POOL does not fully control the file format, but can help to detect unwanted format changes during regression testing

Storage Manager

- All basic functionality is provided
  - Frequency of bug reports significantly dropped during the last months
- Mainly performance and consolidation, but…
  - Current dictionary loading creates deployment problems
    - All class dictionaries need to be loaded when ROOT file is opened
    - ROOT provides functionality to relax this constraint
    - POOL will work with ROOT team to make lazy dictionary loading available for POOL clients
  - Embedded pointer to non-polymorph type – POOL should store objects based on the pointer type
- Internal Review: Provide ROOT with POOL references and collection access
  - Looking at POOL plug-in for interactive ROOT
- Will demonstrate that POOL can expose the schema evolution facilities existing in ROOT

POOL Performance - first cut..

- POOL has not really been optimised systematically
  - Because many functional changes still late in the first experiment integration phase
  - Still first results look reasonable
    - We won't be faster than ROOT
    - We won't create smaller files than ROOT
  - But we want to control the overhead we put on top of ROOT - comparing to ROOT in areas where root offers similar functionality
  - POOL collection performance show clearly that POOL insulation overhead can be kept minimal (few percent level)
- POOL provides more functionality and flexibility than vanilla ROOT
  - comparing raw IO speed for very different operations risks to be comparing apples with pears

File Catalog Plans

- Used in the production environment
  - Several reports about successful use in experiment production chain
  - POOL waterfall model consisting of several catalog implementations to allow a large degree of decoupling and to cope with very different requirements is used and works
- Extension to allow for typical Meta Data evolution use cases
  - Eg new meta data elements are introduced during production
- Composite Catalogs
  - Accessing a single writable catalog together with several shared read-only catalogs
  - Eg a job reads some user files in addition to any file from the large experiment production
- Coming up
  - Upgrade to EDG-RLS 2.2 (required for LCG-2)
  - Integration/reimplementation with Globus and ARDA catalogs

POOL Collection Futures

- Several implementations exist and are used for prototyping
  - Integration with experiment frameworks just starting
  - Still many open questions about requirements
    - Is there a Collection Catalog (like the File Catalog)? A central one? What collection meta data needs to kept?
    - How do POOL collections tie in with grid middleware?
- Collection implementation in POOL is a first step
  - But the real issue is not the implementation but rather conceptual
  - Need active experiment involvement in this area
- Role of collections in a grid environment needs clarification and prototyping
  - Expect active collaboration with ARDA to come up with a model for deploying collections in production and analysis environments

RDBMS Independence

- POOL should not depend on a particular RDBMS
- In addition - MySQL++ is becoming a constraint
  - Need a replacement soon for several reasons
    - Performance constraint on collections implementation
    - Product does not seem to evolve anymore
    - Dependent on internals of the GCC compiler
      - Difficulties to port mysql++ based code to icc/ecc
- Propose to move to OTL after a "market survey"
  - Tests with OTL interfacing to MySQL, Postgres and Oracle suggest that a high level of independence can be achieved
  - Prototype implementations exist for MySQL FileCatalog and Collections
- Prototypes are now part of V1.4 internal releases cycles and expected to reach production quality soon

Infrastructure & Testing

- Move to AA testing tool - QMtest
  - Align with other LCG projects
- Several new platforms coming up
  - icc, ecc and VC for portability check of POOL code and also as additional development platform
- Automated data format regression tests
  - Highest priority now as experiment data is now being produced
  - Complex schema test cases in collaboration with experiments
- Add traceability between bug reports and release contents and release validation tests
  - In collaboration with SPI

Summary

- POOL has delivered a functional persistency framework and has been integrated into frameworks of CMS, ATLAS and soon LHCb
  - Currently used for test productions in CMS
  - Possibly with more effort than integration teams expected
- POOL as a development team works well and would profit more from insuring stability than additional manpower
  - Some central positions inside POOL are more difficult to back up, but we remained productive even through vacation periods overlapping with experiment integrations
- POOL operates close to its release plan
  - Following "release early, release often" strategy
  - Many experiment requirements have been clarified and agreed only during experiment integration phase rather than upfront
  - POOL has been validated on LCG-1
- POOL Workplan for 2004 is currently being defined
  - Validation of POOL for LCG-2 planned with V1,5

- Many thanks to
  - all developers working on the project for their commitment
  - all experiment integration teams for their patience and very constructive feedback!