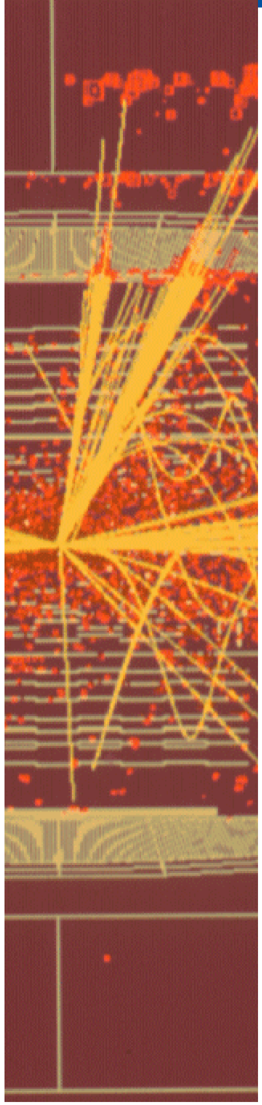


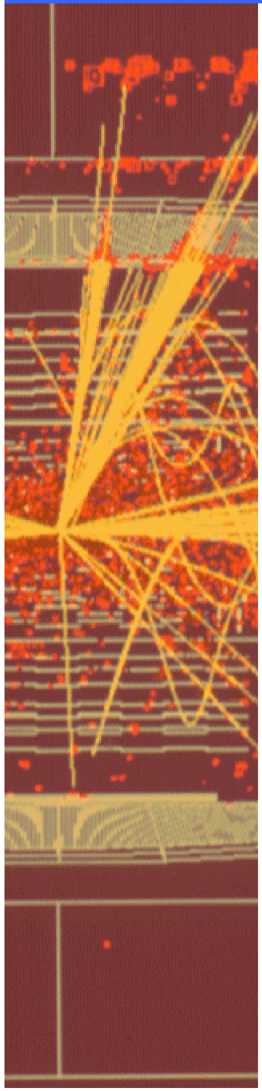
POOL
Object persistency for LCG

Giacomo Govi

CERN IT/PSS

Applications Area Internal Review, 18-20 September 2006

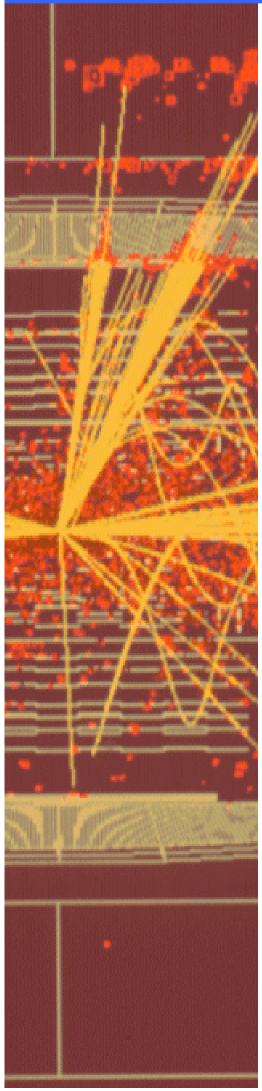
POOL Team



The Development Team

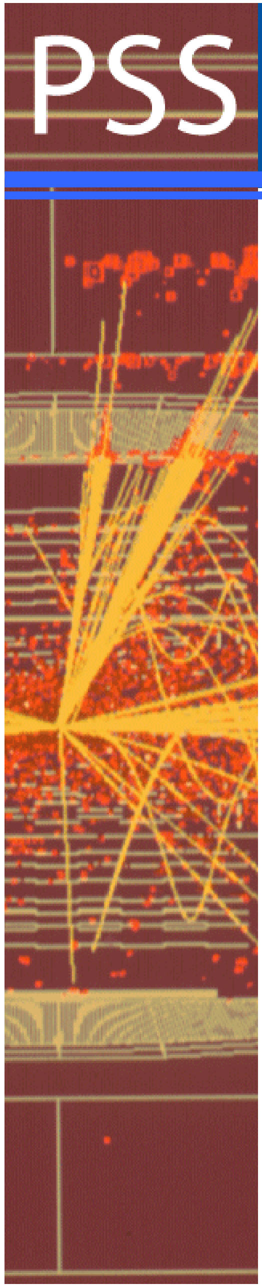
- IT-PSS-DP
  - Radovan Chytracek
    - PoolCore, XMLCatalog
  - Maria Girone
    - XMLCatalog
  - Giacomo Govi
    - DataSvc, ObjectRelationalAccess, RelationalStorageSvc, Testing
  - Ioannis Papadopoulos
    - PersistencySvc, ObjectRelationalAccess, RelationalStorageSvc, RelationalCatalog
- IT-GD
  - David Smith
    - LFCCatalog
- Experiments
  - Markus Frank (LHCb)
    - StorageSvc, RootStorageSvc
  - Marcin Nowak (Atlas)
    - StoragesSvc, Collections
  - Zhen Xie (CMS)
    - FileCatalog

Project Scope



- **Multi-PB experiment data** and **associated meta data** required to be stored in a distributed and Grid-enabled fashion
  - various types of data have to be handled
    - event data, detector data, calibration data, meta data
  - different volumes involved
  - different access patterns
- More **storage technologies** involved to support the various use cases and requirements
  - C++ object streaming, file-based persistency (**ROOT**)
  - **RDBMS** of different flavors (Oracle, MySQL, SQLite)
  - **web based caching** for read-only access
  - local catalog data extraction on **XML**-based files

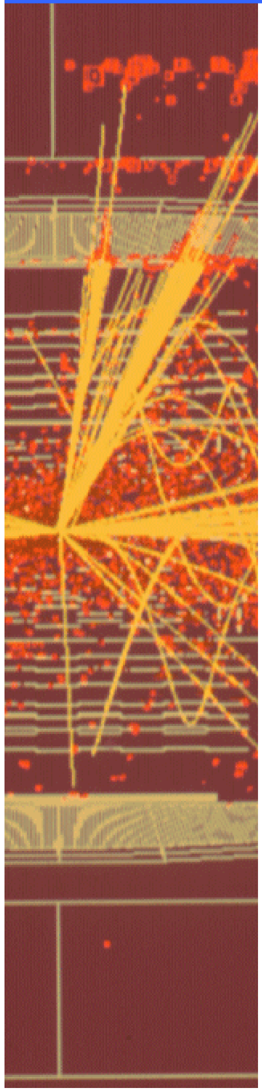
➢ Need for a **technology free** data access mechanism, shielding the physics software from the storage technology details.



Domain decomposition

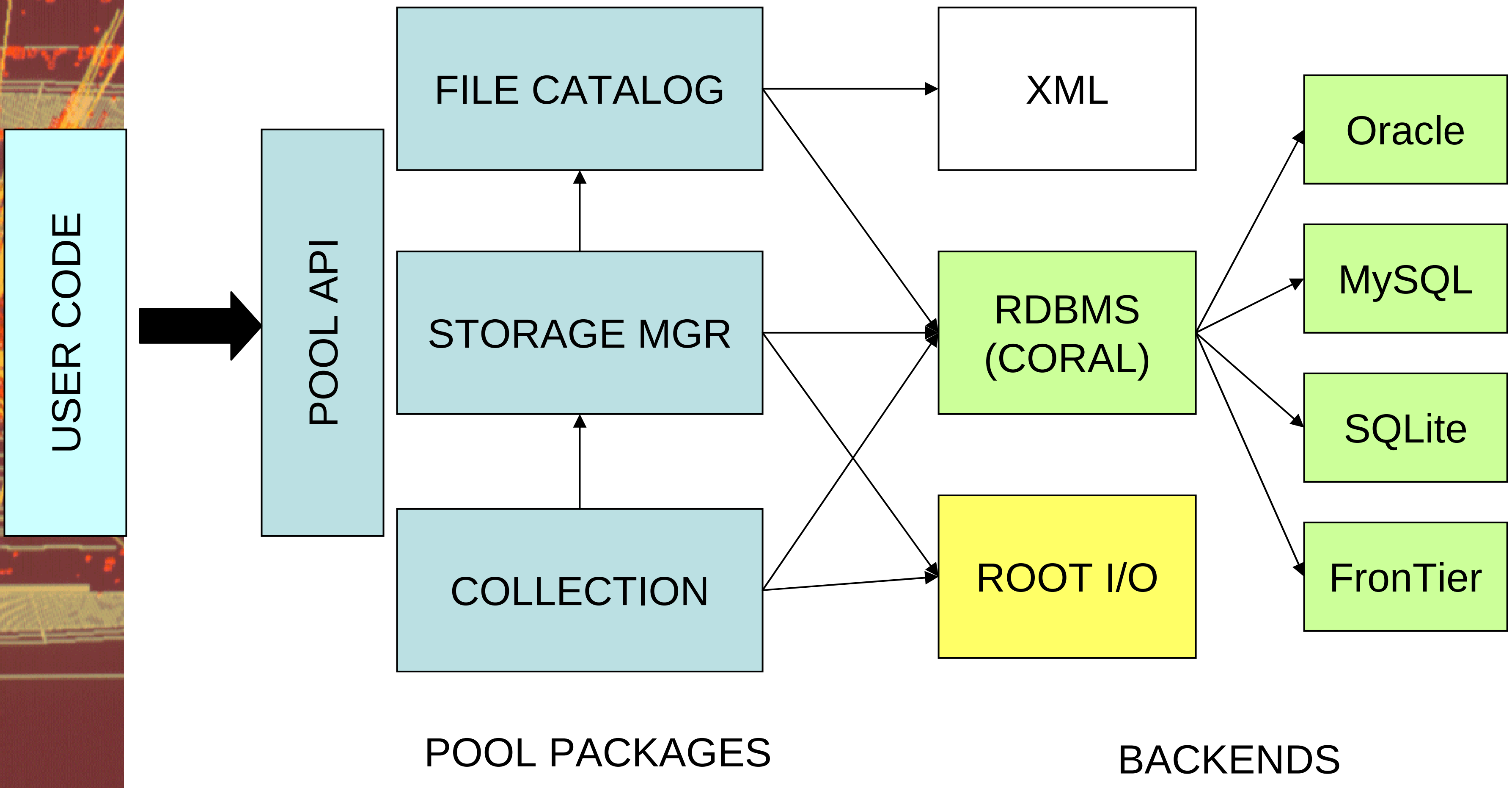
- **Storage Manager**
  - Store a transient C++ object of arbitrary complexity (including reference to other objects) into the persistent storage
  - Resolve logical object reference to a physical objects
  - Read objects from persistent storage
  - Object cache
    - Handle reading of hierarchies of related objects with a loading-on-demand mechanism
    - Keep track of already read objects to speed up repeated access
- **File Catalogue**
  - Keep track of files (physical and logical names) and their description
  - Resolve a logical file reference (FileID) into a physical file
  - Enables the integration with the Grid middleware
- **Collections**
  - Keep track of large object collections and their description with metadata
  - Allows navigation through the object stored in a database or subsets of them

Data categories & storage back-ends



- **ROOT I/O** based backend targeted for complex data structure
  - **event data**
  - Typically hundreds of structure types, many relations between them
  - LHC experiment applications deal with network of objects
  - References used to describe relations
  - analysis data
- **RDBMS** more natural choice for non-event data
  - **conditions, calibration, alignment, detector description**
  - possibly produced by online systems
  - frequently involved in selection queries
  - Oracle as a master storage – tier1
  - MySQL for local farms – tier2
  - SQLite for small exports
  - FronTier as a web-based cache to speed-up read-only access
- **XML** used for simple data structure in local computing environment
  - **catalogue metadata**

Component interaction



POOL external dependencies

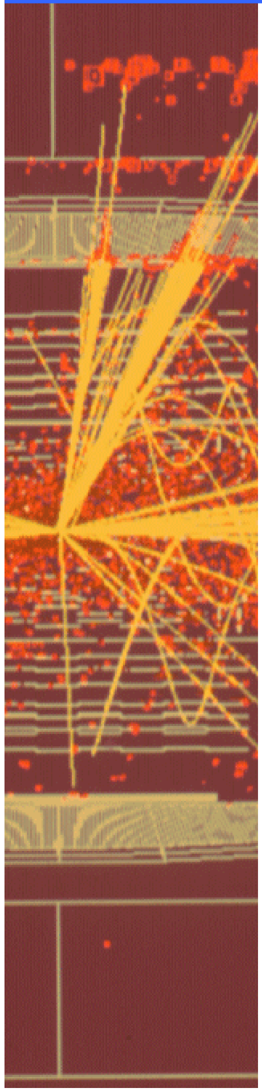
Public Interfaces:

- ROOT::Reflex::Type
  - Storage Manager
  - Visible on the middle and low level layers
- Coral::AttributeList
  - File catalogue, Collections
  - Visible as soon as migration to CORAL is completed
- Seal::Context
  - Storage Manager
  - Removed if component model is fully integrated

Implementations:

- ROOT::Reflex
  - Storage Manager, Collections
- ROOT: I/O
  - Root backends
- CORAL: AttributeList, Relational Access
  - FileCatalogue, Collections, Relational backends
- Seal: MessageStream, Plugin Manager, Component Model
  - Storage Manager, Relational File Catalogue, Relational Collections
- Xerces:
  - XMLCatalogue
- LFC Client:
  - LFC Catalogue

Overall changes

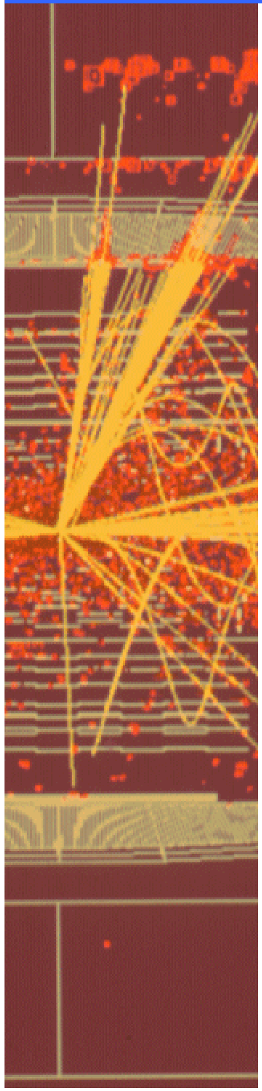


- LCG dictionary evolution followed up (strong dependency)
  - Reflection replaced by Reflex
  - Reflex moved into Root
  - Storage Manager and Collection affected
- CORAL package factored out
  - Affected the Relational backends for all of the 3 domains
  - Migration transparent in the implementations
  - 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

Storage Manager status

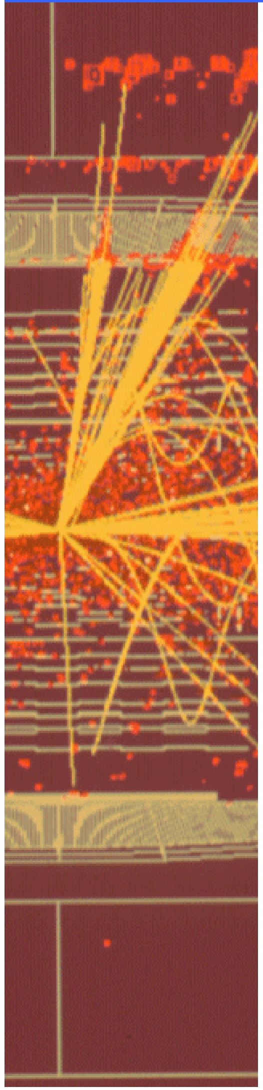
- New functionality added: dictionary auto-loading
  - Enables the loading-on-demand of the required dictionary libraries at run time
  - Code completed since POOL_2_2_6 (Atlas contribution)
- Cache and persistency service basically unchanged (apart from Reflex business)
  - Added command lines for the 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

Relational Storage Mgr status (I)



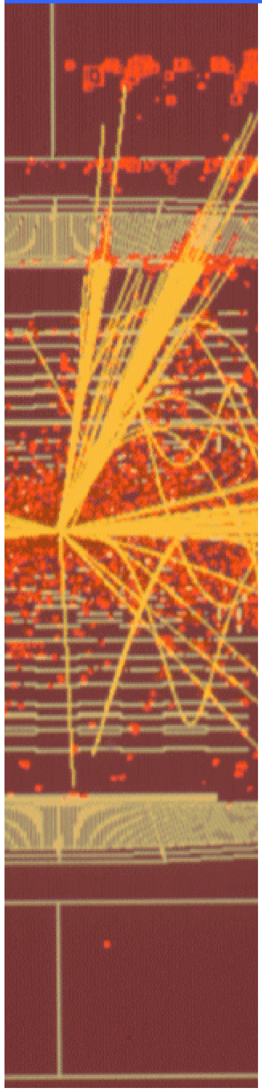
- Relational StorageSvc (ORA) fully functional
  - Supports all of the CORAL backends
  - Adopted by Atlas and CMS
    - CMS in deployment phase
- New features added
  - Functionality to set up a POOL database from an existing set of relational tables
    - Command line tools based on XML driver file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

Relational Storage Mgr status (II)



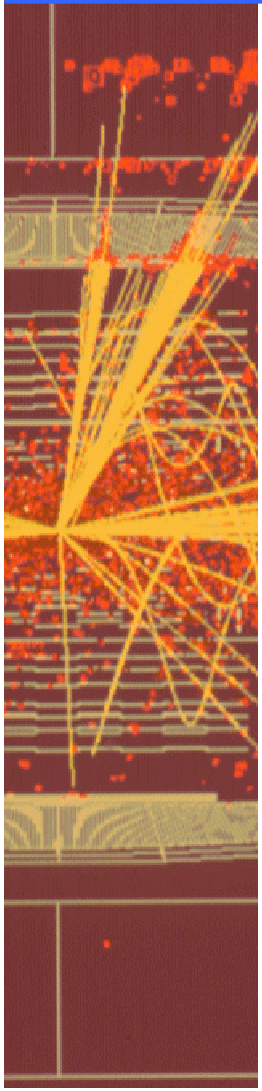
- Optimizations
  - – STL Containers handling with CollectionProxy
    - Eliminates the need of special dictionaries containing detailed container functions
    - Improves iteration performances
  - – Overhead in the ‘bootstrap’ phase reduced
    - POOL relational database needs to read the content of some special tables before end
      - – Object mapping and database structure info
    - Minimal number of necessary queries is performed
  - – Few bug fixes
    - Handling of keyed containers, transactions,…
    - Thanks to beta tester! (Zhen)

File catalogue



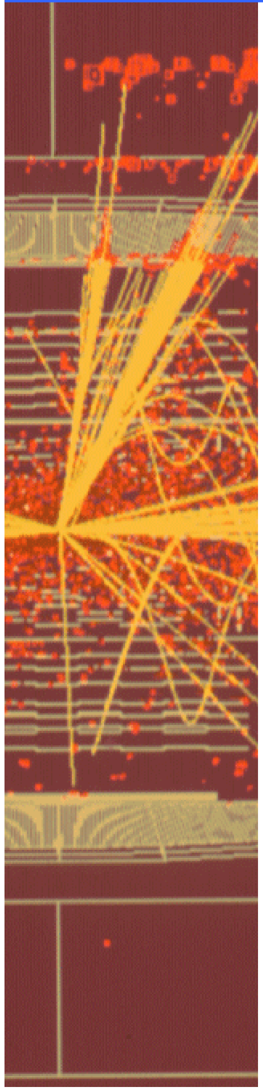
- Key component, used from the experiment applications
  - Not only in the context of the POOL object storage
- Back-ends for grid connectivity evolved in parallel to middleware
  - EDG implementation phased out
  - LFC, Globus and Glite adaptors for POOL catalogue initially released
    - Implementation 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

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)

Outlook



- All of the POOL component are currently integrated in the experiment software
  - Wide support experience gained over three years of production-like running condition in many of the areas covered.
- Most of the packages in maintenance mode
  - Following up changes of the dependencies
  - Bug fixes – especially in the packages more recently integrated
  - Most of the interface stables
  - Documentation maintained aligned with the releases
    - Workbook dropped
    - Little effort due to scarce manpower
- Developments 'on demand' focused in specific areas (e.g ORA)
  - Some consolidation might be required
  - Performance optimization might need more iterations