The LCG Pool Project and ROOT I/O

Dirk Duellmann

- What is Pool?
- Component Breakdown
- Status and Plans

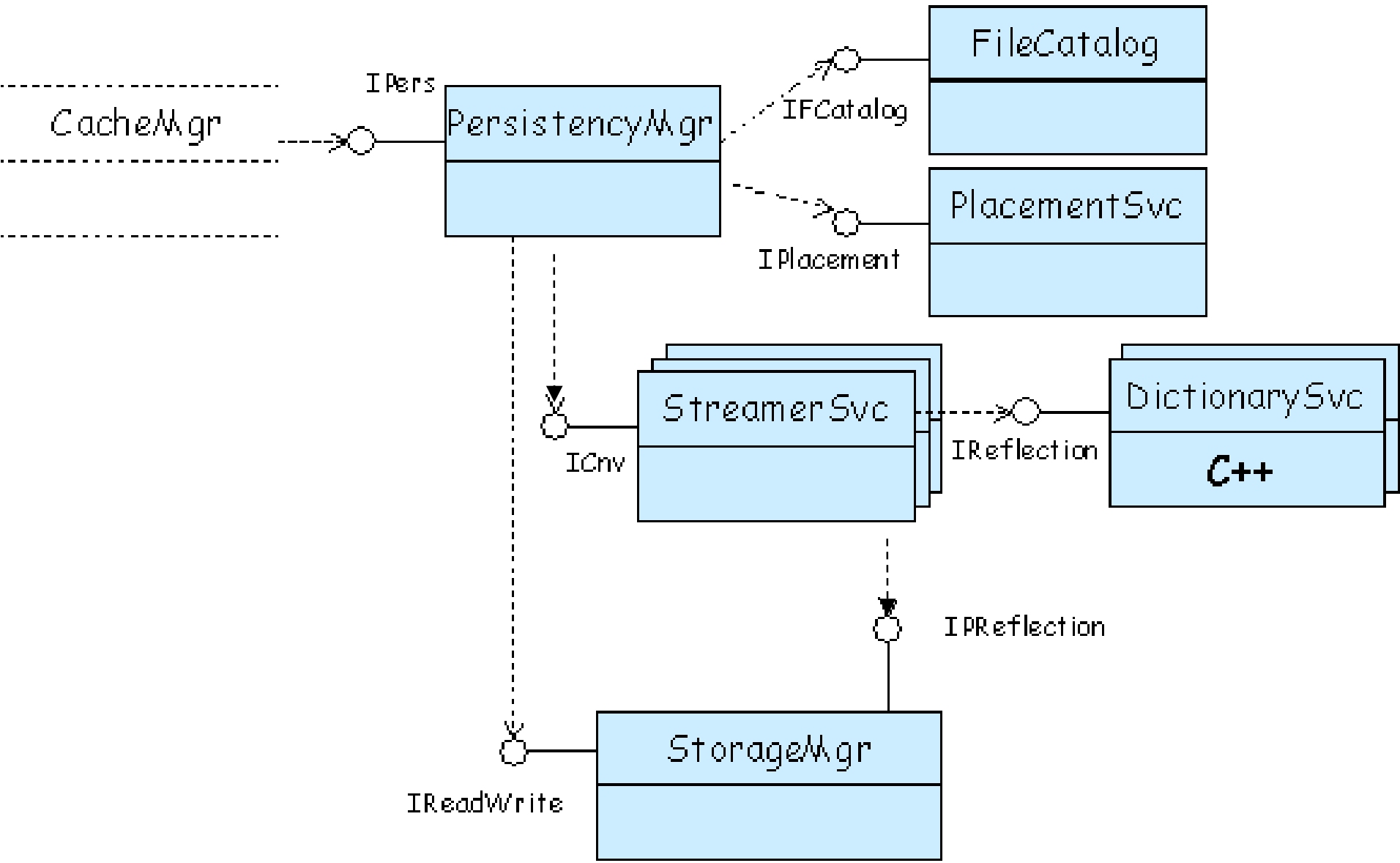
What is Pool?

- Pool of persistent objects for LHC
  - http://lcgapp.cern.ch/project/persist/
- LCG Framework for Object Persistency
  - Targeted at event data but not only event data
    - scalability, navigation, decoupling
  - Hybrid technology approach
    - RDBMS for consistent meta data handling (eg transactions)
      - eg file catalogs, object collections, extensible meta data
      - prototype based on MySQL (and InnoDB in particular)
    - Object persistency via extensible streamer/converter system
      - based on LCG Reflection component (transient object dictionary)
      - prototype uses Root I/O (Tree and TNamed mode)
  - Following the LCG component approach
    - components with well defined responsibilities
    - communicating via public component interfaces

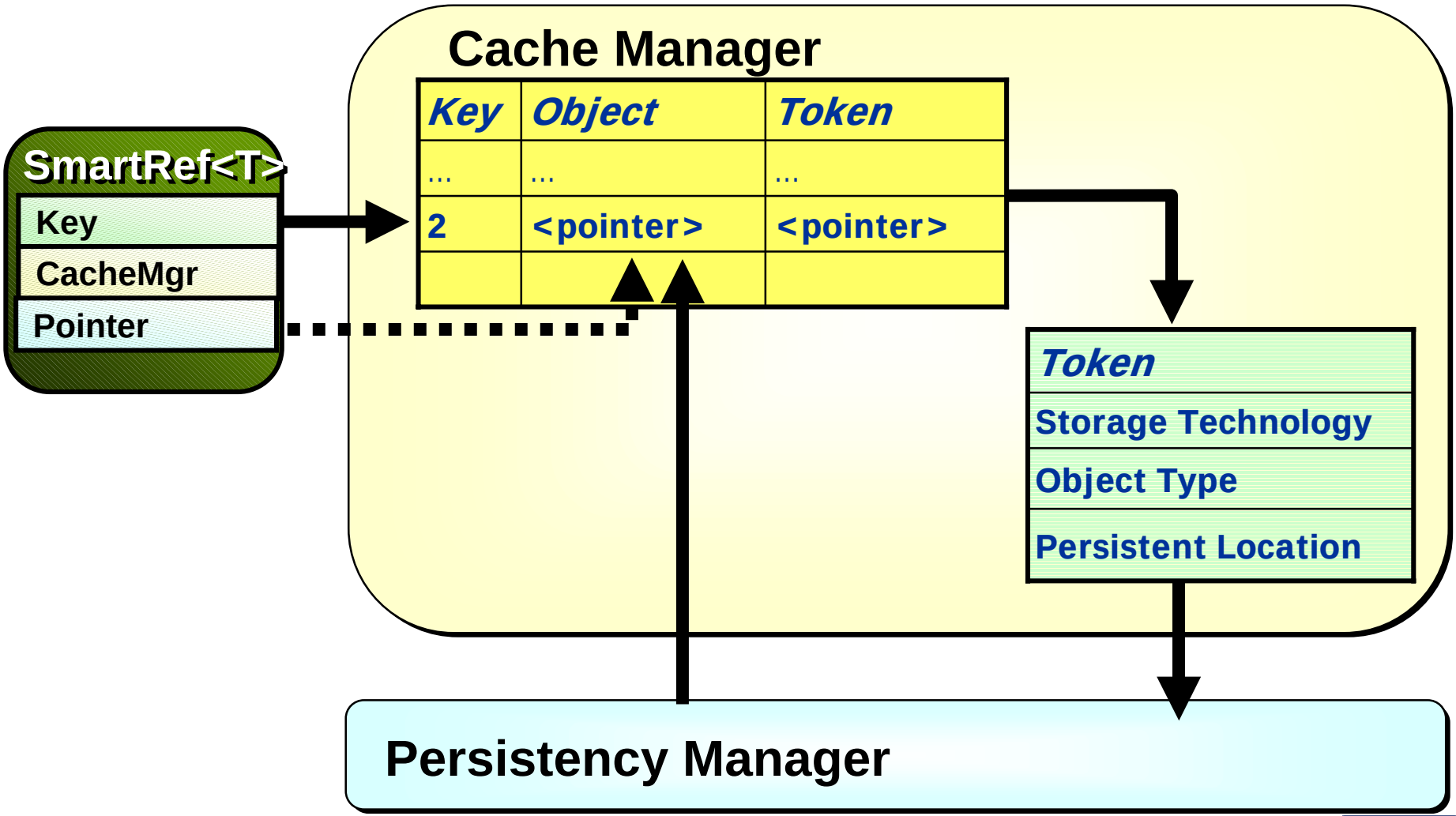
Pool as a LCG component

- Persistency is just one of many projects in the LCG Applications Area
  - Sharing a common architecture and s/w process
    - as described the Blueprint and Persistency RTAG documents
  - Persistency is important...
    - ...but not important enough to allow for uncontrolled direct dependencies eg of experiment code on its implementation
- Common effort in which the experiments take over a major share of the responsibility
  - for defining the overall and detailed architecture
  - for development of Pool components
- During the first internal release
  - contribution from several experiments and IT
  - total ~5 FTE mainly at CERN
- Additional contributors preparing components for upcoming releases

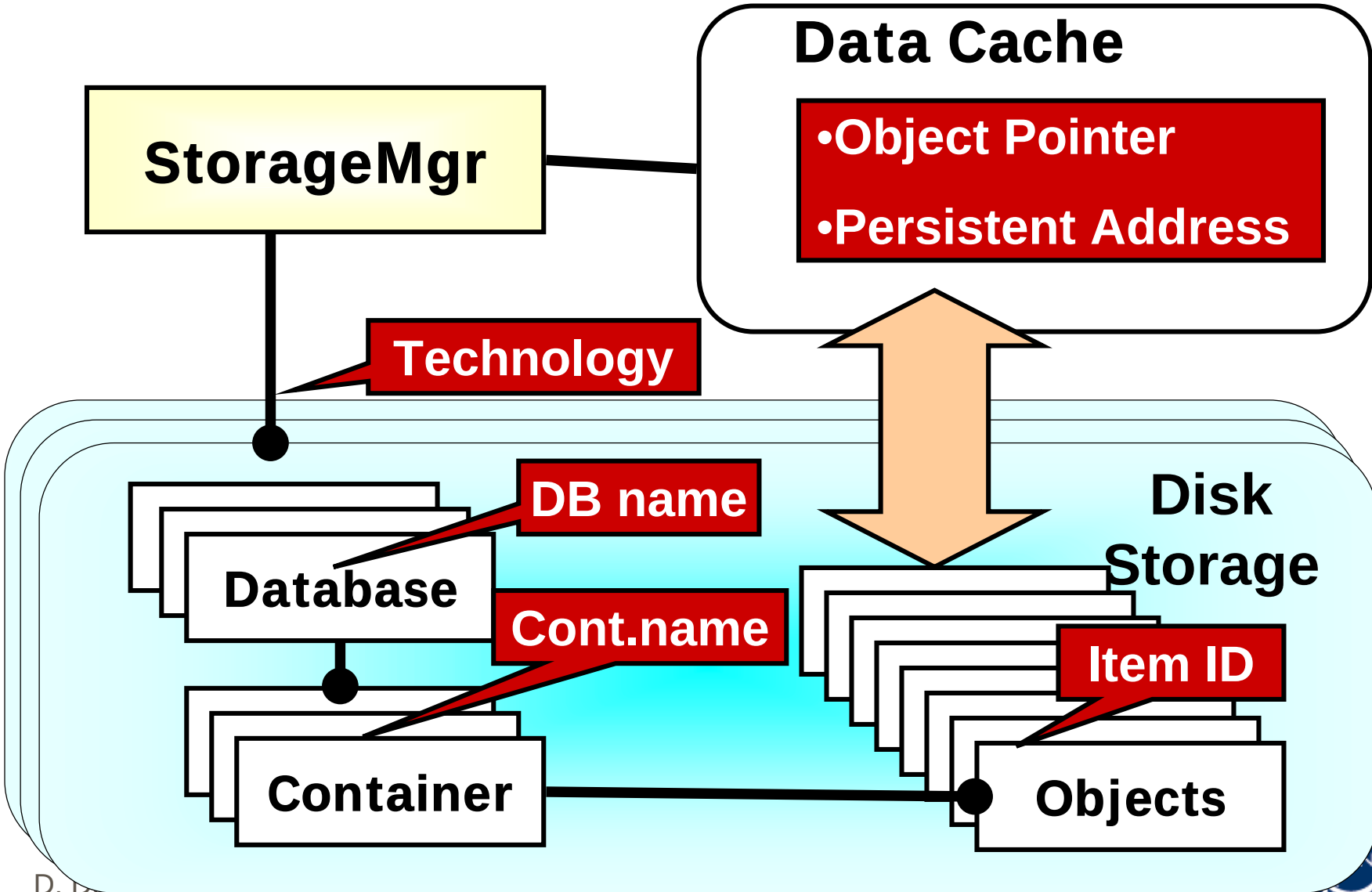
POOL Components



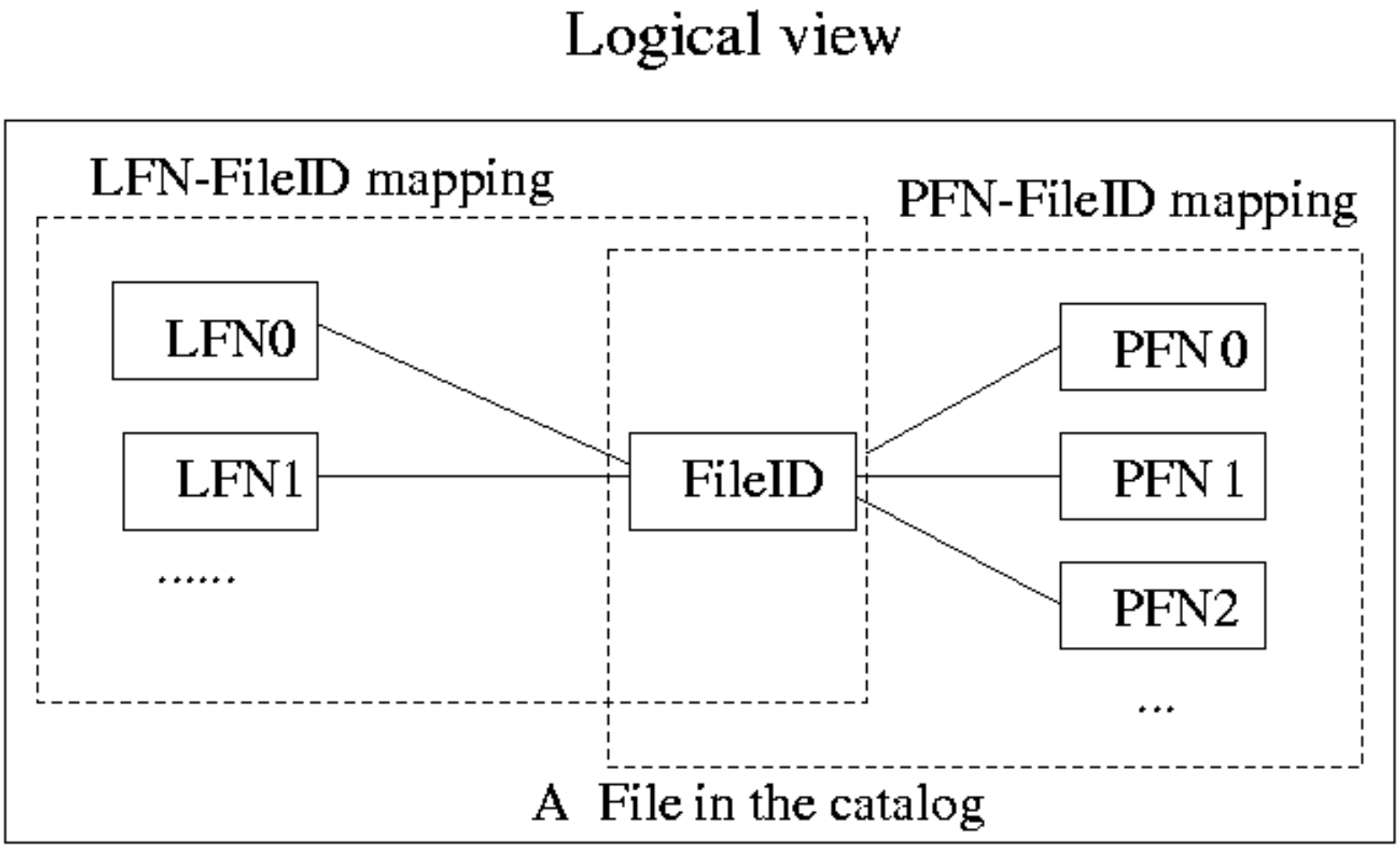
POOL Cache Access



Storage Hierarchy



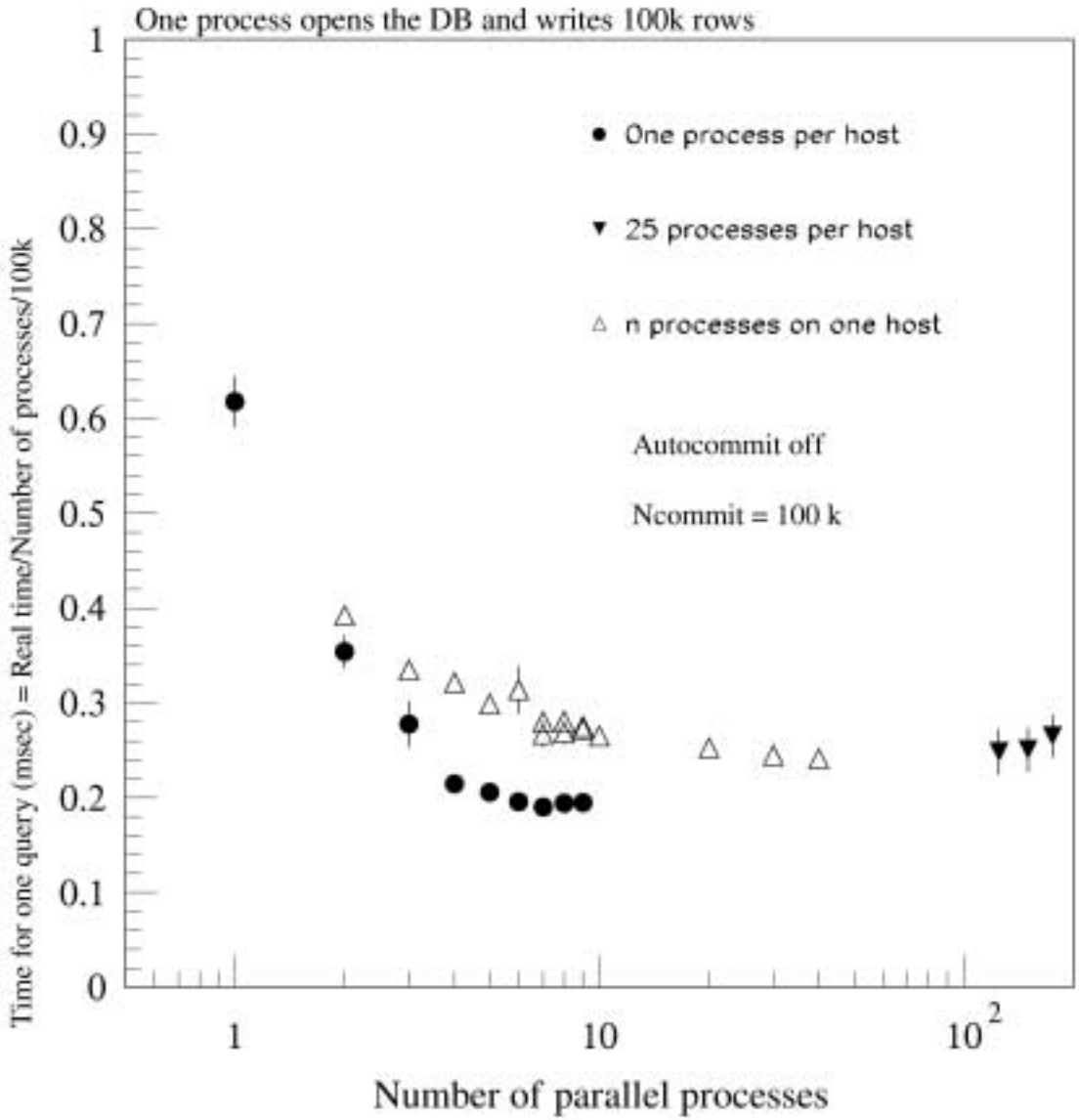
POOL File Catalog



Use Case: Farm Production

- Production manager may pre-register output files with the catalog (eg a “local” MySQL or XML catalog)
  - File ID, physical filename job ID and optionally also logical filenames
- A production job runs and may continue to add files and catalog entries directly.
- During the production the catalog can be used to cleanup files (and their registration) from unsuccessful jobs based on the job ID.
- Once the data quality checks have been passed the production manager decides to publishes the production catalog fragment to the grid based catalog.

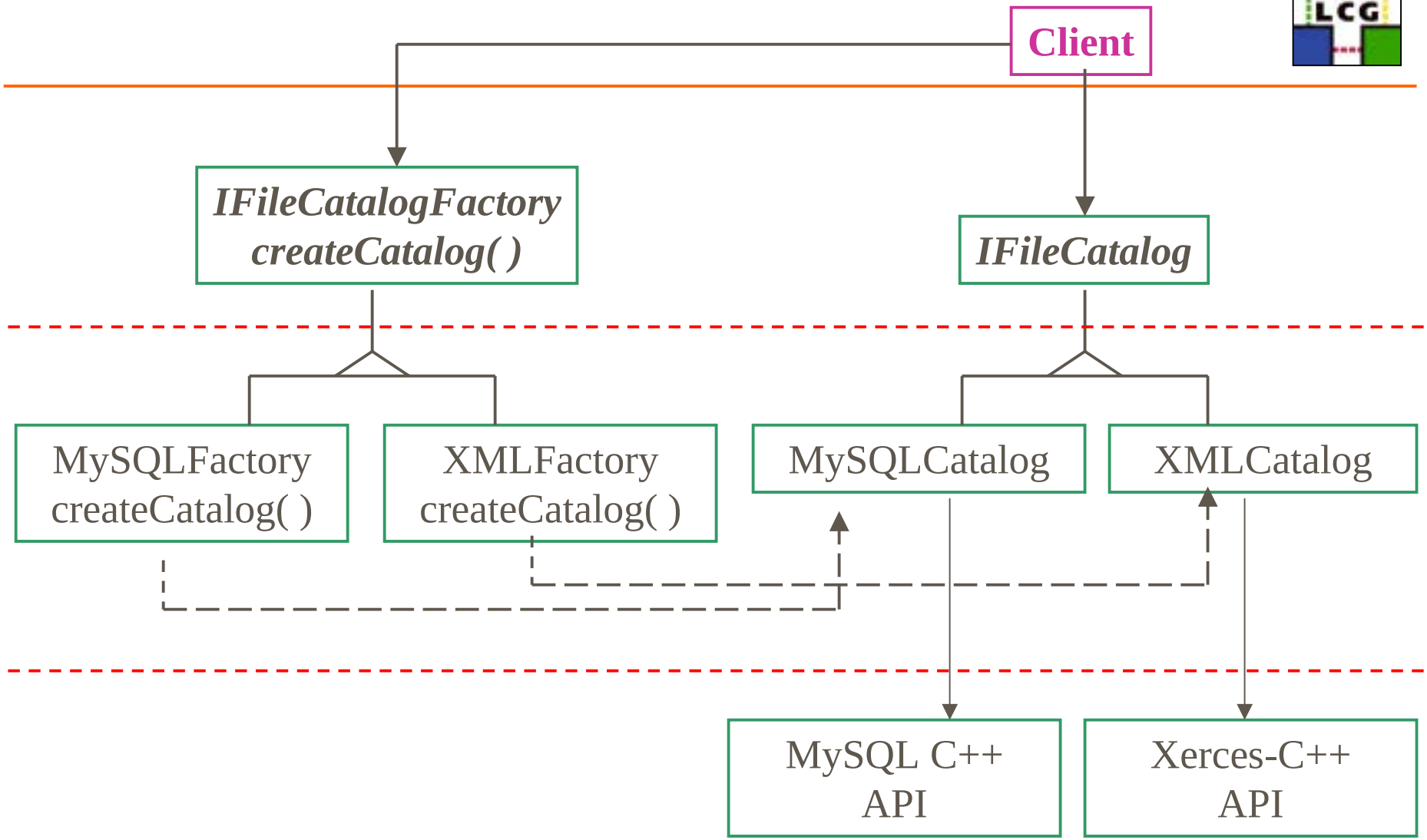
File Catalog Scaling Tests



- update and lookup performances look fine
- scalability shown up to a few hundred boxes
- need to understand availability and back in the production context

Use Case: Working in Isolation

- The user extracts a set of interesting files and a catalog fragment describing them from a (central) grid based catalog into a local (eg XML based) catalog.
- After disconnecting the user executes some unchanged jobs navigating through the extracted data.
  - Any new output files are registered into the local catalog
- Once ready for data publishing and connected again the user submits selected catalog fragments into the grid based catalog.


Client
IFileCatalogFactory
createCatalog( )
IFileCatalog
MySQLFactory
createCatalog( )
XMLFactory
createCatalog( )
MySQLCatalog
XMLCatalog
MySQL C++
API
Xerces-C++
API

Functionality in the V0.1 release

- RootI/O Storage Service
  - two operation modes objects in a tree or objects named into directory structure
  - provide coherent access to the two very different root optimisation models with the user same code
- File Catalog
  - two implementation based on MySQL (networked) and XML (local file)
  - provide catalog C++ API & command line administration tools
- Refs and Cache
  - simple cache implementation
  - mainly acting as test bed for Ref classes – example for integration with experiment caches
- Reflection & Conversion
  - Full reflection interface
  - Some re-factoring of the interface, template and namespace support are being put in place

Release Platform

- POOL V0.1 has been released on October 3rd
  - the cvs release tag is POOL_0_1_0
- Single supported platform so far
  - RedHat 7.2 using gcc 2.95.2
    - Code should also work on rh61 and (partially) on win32/VC6
- Require several external packages hosted by SPI
  - see /afs/cern.ch/sw/lcg/external
  - MySQL (4.0.1-alpha)
  - MySQL++ (1.7.9)
  - Root (3.03)
  - Xerces-C (1.6.0)
- Supported build systems
  - scram and (less complete) cmt

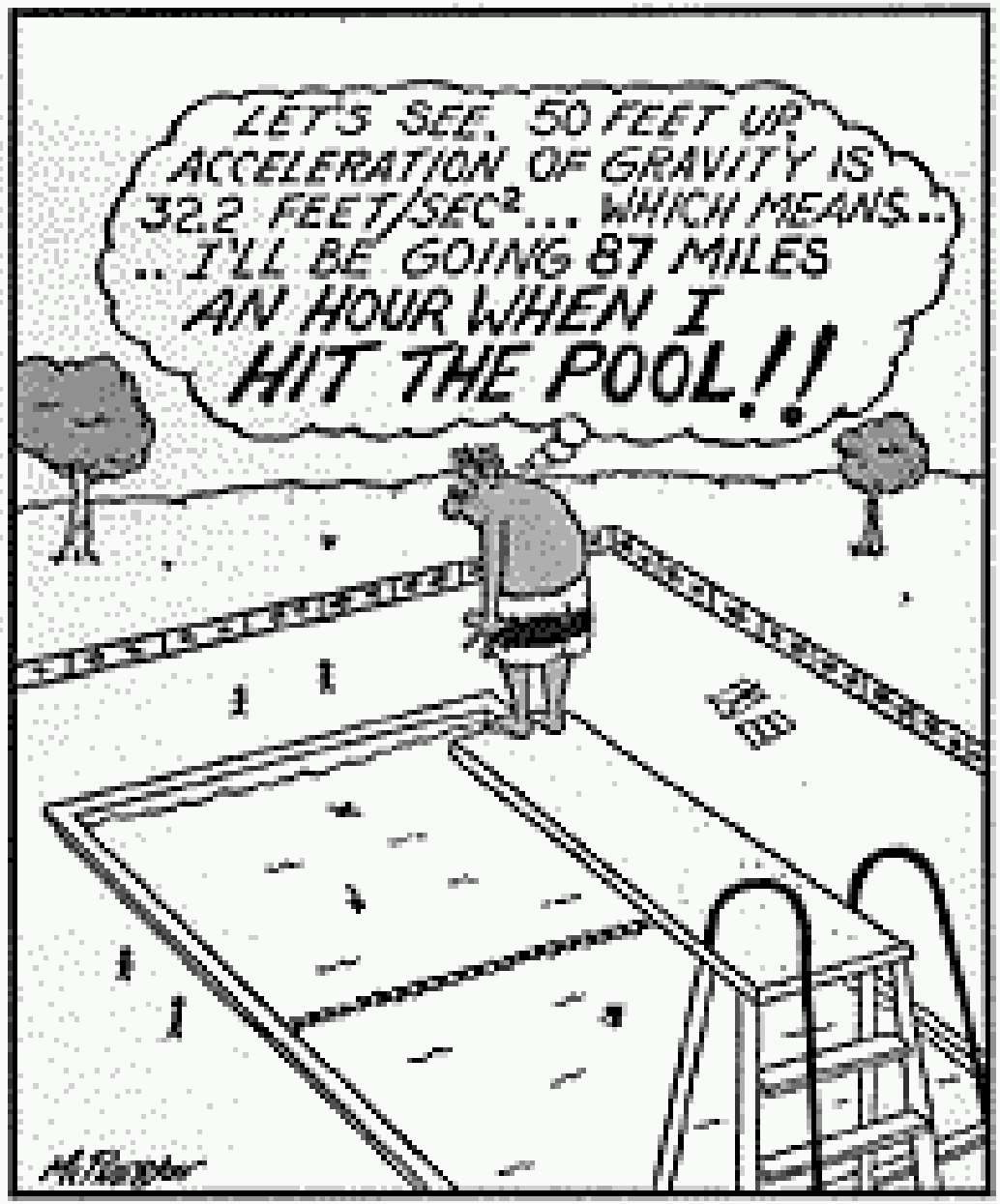
Pool Release Schedule

- September - V0.1 - Basic Navigation
  - all core components for navigation exist and interoperate
    - FileCatalog, Refs & Cache, Storage Svc, Reflection (limited)
  - some remaining simplifications
    - Assume TObject on read/write – simplified conversion
- October - V0.2 – Collections
  - first collection implementation integrated
    - support implicit and explicit collections on either RDBMS or RootIO
  - persistency for foreign classes working
    - persistency for non-TObject classes without need for user code instrumentation
- November - V0.3 – Meta Data & Query (first public release)
  - EDG/Globus FileCatalog integrated
  - Meta data annotation and query added
    - for event, event collection and file based meta data

Summary

- The LCG Pool project provides a hybrid store *integrating* object streaming like Root I/O with RDBMS technology for consistent meta data handling
  - Transparent cross-file (and cross-technology) object navigation via C++ smart pointers
  - Integration with Grid technology (eg EDG/Globus replica catalog)
    - network and grid decoupled working modes
  - Strong emphasis on component decoupling and well defined communication/dependencies
- Pool is still young and will need more time to mature to production quality
  - First internal release V0.1 has just been produced on schedule
  - Public release is expected for (end-of) November
- First larger scale production attempt summer ‘03

See you next summer in the POOL!



There are times when being a whiz at physics can be a definite drawback.