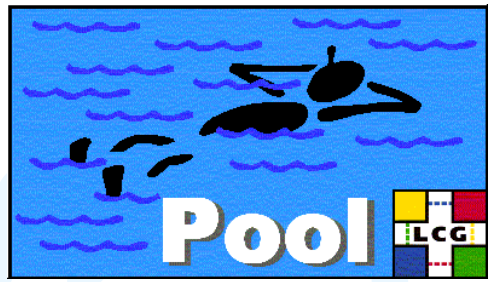


POOL persistency framework for LHC

Zhen Xie

Princeton University

(for the POOL project)

Outline

- The POOL project
  - overview, experience and new development
  - architecture, domains and software components
- Object Relational Mapping in POOL (POOL-ORA)
  - Object Relational Mapping (ORM)
  - POOL implementation of ORM
- CMS offline conditions database with POOL-ORA

POOL Project

- Pool Of persistent Objects for LHC
  - Framework for persistency of C++ objects
  - Started in 2002 as a project under LHC Computing Grid (LCG) Application Area. Collaboration of LHC experiments and CERN/IT
- POOL widely used in the LHC experiments
  - Hybrid approach: C++ streaming technology, such as ROOT/IO, for bulk data; RDBMS for catalogs, collections and metadata.
  - Established baseline for event data handling for ATLAS, CMS, LHCb
  - Data volume stored ~400 Tb!
- New development focus
  - RDBMS implementation of POOL API through Object Relational Mapping (ORM)
    - Started to proof POOL API is not bounded to ROOT
    - Interests in the product went immediately beyond the original scope
  - First functional release in Feb. 2005, chosen by CMS as the baseline for offline conditions database right afterwards
  - This is the focus of this talk

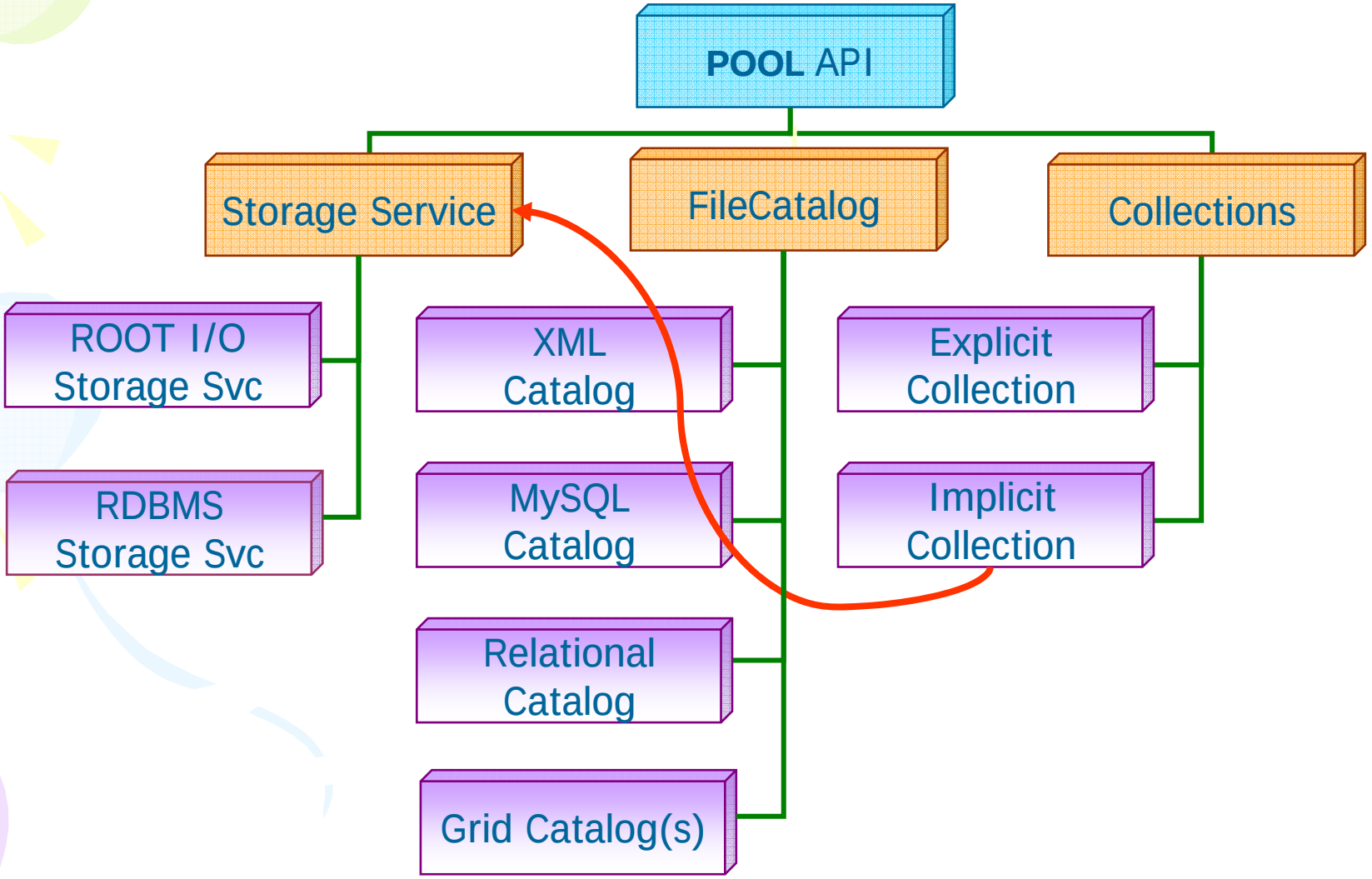
POOL architecture

- Storage technology neutral API
- Built from SW components
  - Implement pure abstract C++ interfaces
    - user code is insulated from concrete implementations and technologies
  - Minimal dependencies between components
    - Weak coupling ensured by interactions only via their abstract interfaces
  - Components are loaded on demand
    - via plug-in management and component model

POOL domains

- Storage Manager
  - Streams C++ objects into/from a storage
  - Resolves a logical object reference into a physical object
- File Catalog
  - Maintains the information about POOL accessible data files or database instances
  - Resolves a logical reference into a physical data source
  - Entry point to the Grid
- Collections
  - Provides tools to manage ensembles of objects stored via POOL persistence services
    - **Explicit: server-side selection of object from collections**
    - **Implicit: defined by physical containment of the objects**

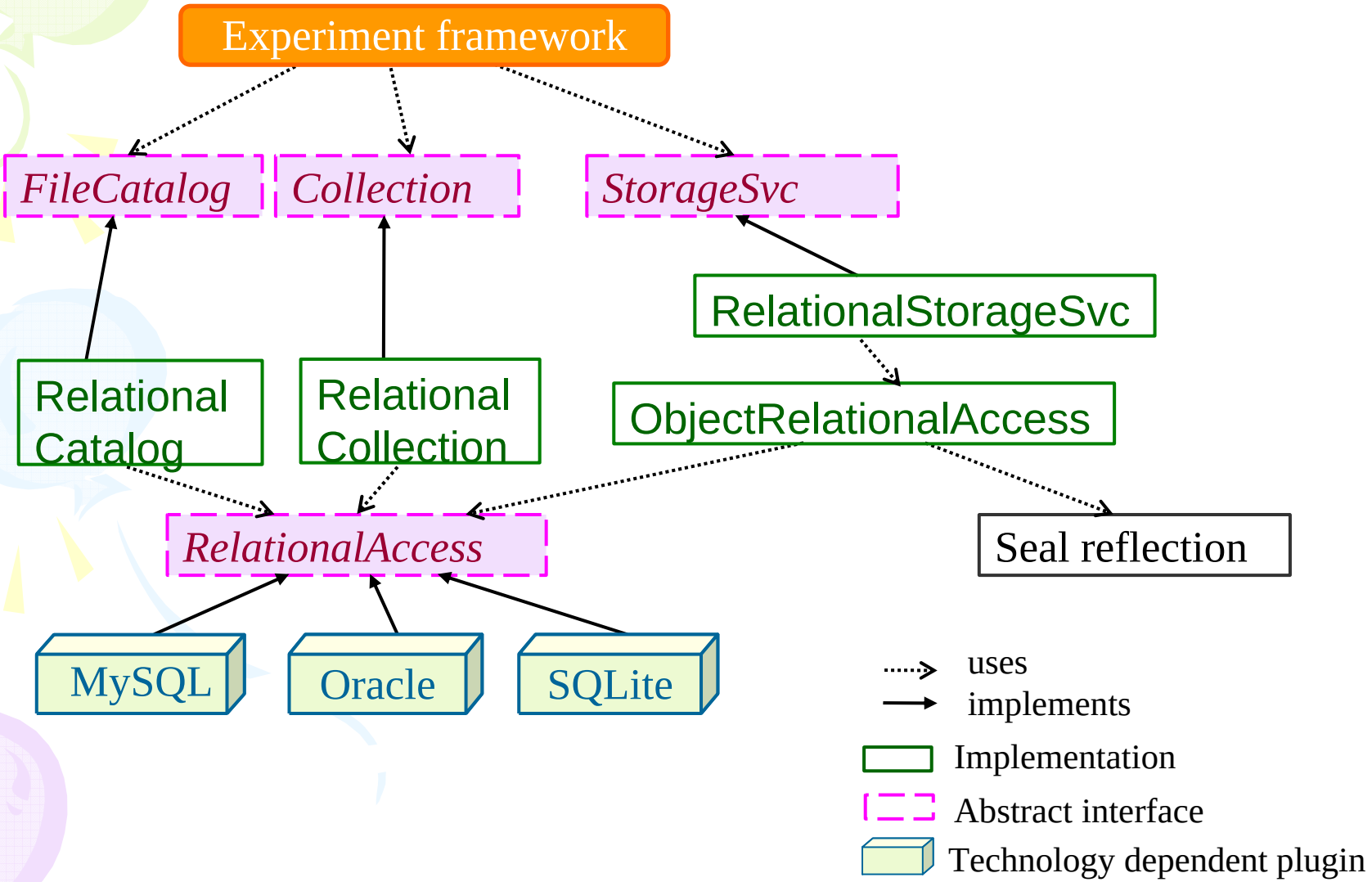
POOL components



Object Relational Mapping (ORM): what, why & how

- What
  - Automated and transparent persistence of objects to tables in a relational database by transform data from one representation to another
  - Bridge the "object/relation paradigm" mismatch
  - Research history back to 1980s, gained more and more popularity over other object persistency solutions
    - Hand-coding persistency layer with SQL or ODBC
    - Object-oriented databases
- Why
  - No need to bend/twist the object model until it matches the underlying database – comfortable for application developer
  - RDBMS vendor independent and SQL-free
- ORM implementations in the open software world
  - *Hibernate*: the most popular framework in Java world
  - Difficult to implement in C++ due to lack of standard reflection library
  - *POOL-ORA* in C++
    - LCG has its own run time reflection library: SEAL reflection
    - POOL has its own database abstraction layer

ORM in the overall POOL architecture



ORM and Object Storage in POOL

- Mapping rules implemented by ObjectRelationalAccess layer
  - Object id: generated from Primary Keys of the table
  - simple class ↔ one table
  - Embedded class: subset of the table row of the owning class
  - Class inheritance: more than one possibilities. The default rule is *"table-per-concrete-class"*
  - Object association ↔ table foreign key constraints
- Mapping metadata
  - which mapping rule is used; association of class/member names to table/column names, types; etc.
  - Mapping metadata is versioned and stored with the object data
  - User registers the mapping metadata in a XML file which steers the process; otherwise, default mapping takes effects.
- Guided object storage
  - Combined information from the SEAL dictionary and the mapping file guides the object streaming at runtime
- Tools allow user to transform existing relational tables to objects.

A Mapping example

```
Class A{
    int x ; float y ; vector<double>  v ;
    Class B { int i ; string s ; } b;
};
```

T_A

p.k.

ID	X	Y	B_I	B_S
1	10	1.4	3	“Hello”
2	22	2.2	3	“Hi”
.	.	.	.	.

f.k. constraint

T_A_V

ID	POS	V
1	1	0.12
1	2	12.2
1	3	4.1
1	4	5.452
2	1	32.1
2	2	0.1
2	3	0.1

This is only one of the possible mappings!

Typical use cases

- Write data as objects into RDBMS and read them back as objects from OO application
  - Transparent: no need of explicit database schema design; tables generated automatically following the ORM rules and metadata.
- Write data as relational tables and read them back as object
  - E.g. reading existing tables in the database as objects from OO application

CMS conditions database with POOL-ORA

- online and offline databases are clearly decoupled
  - Online: pure relational database
  - Offline: POOL-ORA database.
  - Online and offline databases have different schemas
- online to offline transfer procedure
  - extract and transfer necessary data from online database to offline
  - Use POOL-ORA command-line tools to make the transferred data readable as objects
- Offline software handles the event data and the conditions data through the same POOL API even though the underlying technologies are different.

Summary

- LCG-POOL has been integrated into the software framework of three LHC experiments and successfully deployed for data production
  - Provides hybrid store for event data integrating Root/IO with RDBMS technology
  - This area is migrating to support and maintenance phase
- POOL is focusing on developing RDBMS implementation of the POOL-API through ORM mechanism
  - ORM is a popular object persistency solution
  - POOL provides such a solution in C++ which is rare!
- CMS has chosen POOL-ORA as baseline technology for offline conditions database
  - Online and offline database technology compatibility
  - No worry about the object/relational mismatch in software modeling; no SQL vendor specific code in software
  - Event data and conditions data are handled through the same interface: POOL-API