Integrating Pool in COBRA

AA Internal Review, 16 October 2003

Preliminary

Final Version on Sunday

Vincenzo Innocente

CERN/EP

From Objy to Pool: a short history

- **CMS decided to abandon Objy in Autumn 2001**
  - Workshop on Root (link)
  - Workshop on Persistency (link)
- **Oracle 9i**
  - Unsatisfactory C++ binding
- **Root Trees (chep`03)**
  - Exploiting the full power of all root paraphernalia
  - Not satisfying CMS use cases
- **ODBMS inspired (chep `03)**
  - Use Root Keyed-object and TRef
  - Essentially a prototype of POOL

Decision to go ODBMS way and then to POOL following internal review in September 2002 (link)

CMS & Pool

- **CMS has established a fruitful collaboration with the Pool team since the very beginning of the project**
  - Direct participation to the project itself: **2.6 FTE**
  - Efficient communication
    - Savannah Portal
    - Direct mail (and phone) exchange among developers
    - In person meetings when required
  - Continuous and prompt feedback
    - CMS typically feedbacks on any new pre-release in few hours
    - POOL responds to bug reports in 24/48 hours
      - Only few bugs took more than a week to be fixed in a new pre-release

Few old milestones

- **Dec 2002: dictionary built for typical CMS data classes parsing original header file with gcc-xml**
  - ❐ dictionary moved to SEAL, no further direct involvement of CMS
- **March 2003: first tests of FileCatalog**
  - ❐ Feedback on performances, API and command-line tools
- **April 2003:** POOL_0_5_0 **released**
  - ❐ First version able to support realistic use-cases
- **May 2003: first full scale integration completed**
  - ❐ 99% of persistent classes in lcg-dict
  - ❐ Missing features identified
    - – All about items already supported by “Vanilla” Root
- **14 June 2003: POOL_1_1_0-theta released**
  - ❐ satisfied most of the cms requirements
  - ❐ Start of full-scale realistic tests

Use of Pool in CMS: Current Status

- **COBRA 7.4.x OSCAR 2.4.y ORCA 7.5.w**
  - ❒ Based on POOL 1.3.z (now 1.3.3)
  - ❒ First public release on September 20
  - ❒ Under test in production
- **Usable for initial production**
  - ❒ 1-2 Million events produced with OSCAR (G4 simulation) each week
- **Essentially same functionality as Objectivity-based code but**
  - ❒ No concurrent update of databases
    - – No direct connection to central database while running
  - ❒ Remote access limited to RFIO or dCache
  - ❒ No Schema evolution
- **Still few bugs, missing-features, performance problems**
  - ❒ Affect more complex use-cases
  - ❒ Make difficult the deployment to a large developer/user community

Why so late?

On 2003/06/30 POOL 1.1 - First production release **was announced**

- In reality just a honest prototype with many bugs, missing-features, major performance problems.
- CMS realized (too late?) that pool internal unit and integration tests had a ver poor coverage and almost no complexity
  - Navigation features were essentially untested
  - Error conditions even less
  - Simple “chaining” of few tests in a single application caused crashes
- CMS decided to put debugging and integration of POOL as V.I. top priority

Early August a COBRA release based on Pool 1.2.0 was essentially ready for “Simulation” production

- It still shown unexplained error conditions and crashes
- CMS decided that was too risky to start production with such errors non cure

Bill T. and V.I. end spending last 10 weeks debugging, in close collaboration with the pool team, POOL software

What CMS use of POOL?

All objects (event and metadata) are stores as root keyed-objects (no root-tree)

Only object navigation is used, no other access mechanisms

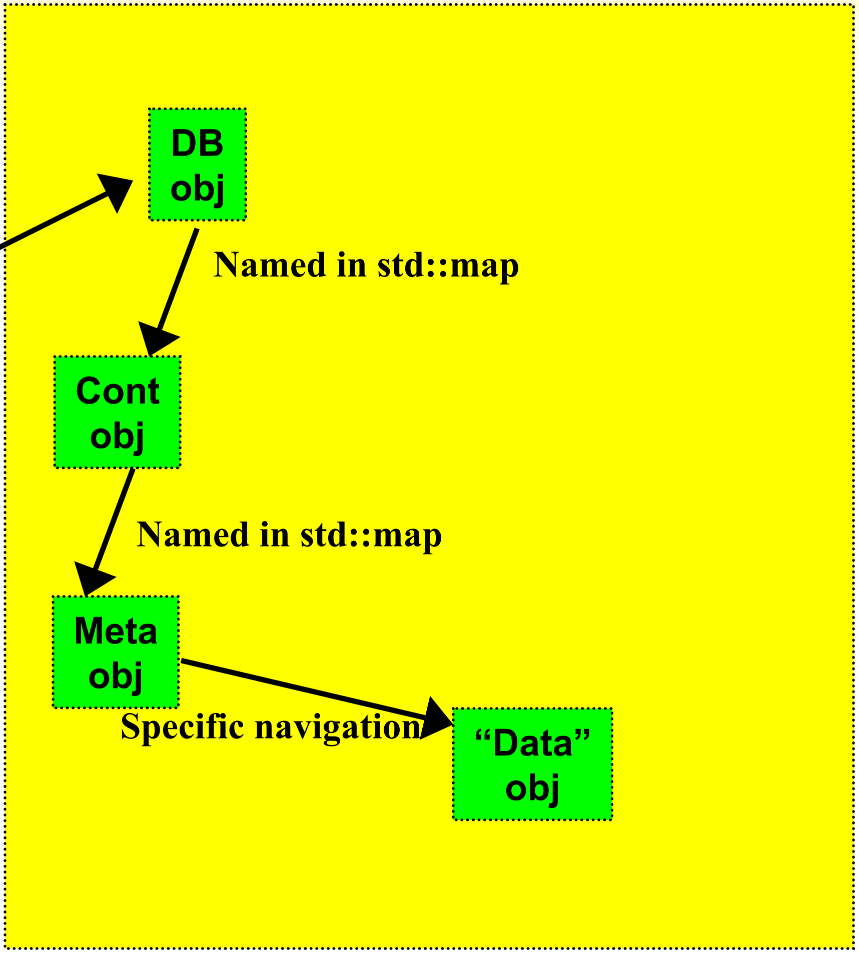
- **File Catalog**
  - Full interface
  - XML implementation in Physics Applications
  - MySQL & RLS under test for production use cases
- **Ref**
  - Full interface
- **Session**
  - Only Transaction Management
- **Few other classes and methods**
  - Mainly workaround to bug/missing-features
  - In test programs

CMS persistency paraphernalia

- Thread-safe proxy-wrappers to pool-interfaces
- Scoped (exception-safe) nested-transaction
- Context/Thread-specific Data Services
- Creation and management of DataBases and Containers
  - Including catalog, PFN, LFN and metadata
- Object (RefBase) -based placement hint
- Generic “named” navigation
  - Mono and bi-directional map<string,Ref>
- Specialized (base) classes
  - Smart-Proxies
  - Collections
  - …

CMS top level access

```
File ID="0C701391-3FE4-D711-801A-00D0B7B86D05">
physical>
pfn file_status="Fully-Registered" filetype="ROOT_All"
ib_status="1"
ame="rfio:shift20:/shift/shift20/data11/zh/cmsprod/OSCAR_2_4
0/mu03_mu_pt5_100/CARF_System.META.sw_Hit2402_g133"/>
physical>
ogical>
fn name="CARF_System.META.sw_Hit2402_g133"/>
logical>
metadata att_name="DBoid" att_value="[DB=0C701391-3FE4-
711-801A-00D0B7B86D05]
CNT=.master][CLID=7D721C8E-530D-608F-BFD9-
0E61D0F1EB5][TECH=00000201][OID=00000003-00000000]"/>
metadata att_name="DataType" att_value=“META"/>
metadata att_name="FileCategory" att_value=“System"/>
metadata att_name="dataset" att_value=""/>
metadata att_name="jobid" att_value=""/>
metadata att_name="owner" att_value="Hit2402_g133"/>
metadata att_name="runid" att_value=""/>
File>
```



A real catalog (test data)

CMS Data Model (same since `97)

- EventStructure498 (web)
- CARF1298 (web)
- Conditions (web)

Few Comments on SEAL

Future

- **Freeze schema now for next 18 months**
  - SEAL/POOL will not support schema evolution in near future
- **Follow a minimalist approach to avoid further confrontations with bugs, missing features, performance problems**
  - Use only what is really needed and produces major benefits to CMS use-cases
  - Avoid migration to LCG/AA software in areas were CMS has already deployed solutions
- **Focus on CMS near-term use-cases**
  - Develop/integrate only components with a wide use potential
  - Do not get involved in projects of unclear benefit to CMS

Concluding Remarks

CMS has ported to Pool all applications that were previously based on Objectivity for all previously supported use cases.

Still a long way ahead of us

- Some critical use cases not yet supported
- LAN and WAN data access/replication not fully tested
- Tuning of performances will require more work

Pool itself should not be considered anymore on the critical path toward CMS Data Challenge in 2004