POOL Based CMS Framework

Bill Tanenbaum

US-CMS/Fermilab

04/June/2003

CMS Framework Status

- **ROOT/IO** now used for persistency
- Dependent on ROOT technology
- Problems for large scale production
  - No concurrency (must use “winter” mode)
  - No file catalog (must keep track of files manually)

POOL Based Framework

- Use POOL file catalog on top
- Replace ROOT/IO with POOL storage manager for event data (ROOT/IO based, TTrees optional)
- For metadata, as above, or (future) use (RDBMS based) POOL collection manager, or RDBMS based storage manager

POOL Advantages w.r.t. ROOT

- File Catalog provided
- Object cache manager provided
- Dictionary generation (SEAL) easier
  - simple XML specification files
  - **lcgdict** simpler and more robust than **rootcint**
    - Does only what is needed for data dictionary
    - simple output – no compilation failures seen yet!

POOL Advantages w.r.t. ROOT

- Technology independent interfaces
- Modular Architecture
  - Components can be used independently
  - Storage Manager has layered architecture (e. g. ATLAS and LHCb will each use its own object cache manager)
- Class headers need no changes
  - No instrumentation (e. g. ClassDef)
  - No inheritance from base object (e. g. TObject)

POOL Advantages (cont.)

- Persistent Reference (pool::Ref<T>) can locate object in persistent store
  - POOL keeps track of file and container
- Can use ROOT trees or ROOT keyed objects
  - POOL handles details of ROOT trees
  - Simple to switch (one parameter)

POOL Advantages (cont.)

- Provides alternatives to ROOT/IO for metadata (providing atomic transactions, concurrency, and other RDBMS goodies)
  - RDBMS based collection manager
  - RDBMS based storage manager (future)

First Stage of Conversion

- Replacing ROOT with POOL Storage manager in COBRA/ORCA (plus adding File catalog)
- All data (including metadata) will still use ROOT storage manager (avoids redesign of metadata at this stage).
- Most time and effort will be debugging and testing.

First Stage Effort (est.)

- One developer nearly full time
- 3 more weeks coding (2 weeks coding done)
- Debugging and testing is unpredictable.
- Optimistically, debugging/testing begins late June, robust product in July.
- Could easily be August, though.

Current POOL/SEAL Status

- **Relevant Features recently released**
  - More STL support (e.g. map)
  - Transient members of persistent objects
  - Polymorphic access through pool::Ref<T>
  - Class name not needed to place object
  - Polymorphic access through C++ pointers
  - Dictionary Generation improvements

POOL/SEAL Status (cont.)

- **POOL 1_1_0 (30 June, prerelease 11 Jun)**
  - Update capability (needed for metadata)
  - Containers as “Implicit Collections”
  - More STL support (multimap)
  - Implicit Ref<T> interconversions
  - etc.
  - Summary: All we know we want at this point

POOL/SEAL Status (cont)

- Dictionaries generated (lcgdict) for all COBRA/ORCA persistent classes!
- All dictionaries compile!
- Conversion of SEAL dictionaries to CINT dictionaries by POOL, and run-time use, still untested for COBRA and ORCA classes

POOL concerns

- **Performance**
  - Storage Manager puts multiple software layers on top of ROOT
  - Good news: For production (writeAllDigis), ROOT takes only about 6% of the time, and most of that is data compression for writing. Still, performance is a concern.

POOL concerns (cont.)

- **Immaturity of POOL**
  - Pool is very young and unproven
    - Growing pains certain
    - But POOL team responds very quickly to requests

POOL Impact on Applications

- Very similar to ROOT!
  - A data dictionary must be generated for any persistence capable user-defined class.
  - A data dictionary must be generated for any user defined class used as a template parameter for a persistence capable class.
  - Therefore, the application and POOL cannot be totally decoupled.
  - Never any source code coupling for classes. Unlike in ROOT, **all** classes are “foreign”.

Future stages of conversion

- Internal cleanup/simplification of COBRA to eliminate vestiges of Objectivity
  - joint project with Vincenzo
  - as much as possible concurrent with stage one
  - data format must be stable soon
- Conversion of MetaData to RDBMS
  - Optional: only if driven by external need

Summary

- Straightforward conversion from ROOT/IO to POOL Storage Manager using ROOT
- Many advantages to POOL
- Major concern is POOL’s immaturity
- But: POOL progressing rapidly
- Performance is another concern