POOL Release V1.0

Dirk Düllmann

LCG Application Area Meeting, 14th May 2003

POOL V1.0 Release

- POOL V1.0 has been released yesterday with ~2 weeks delay
  - Public release targeting expert developers expecting to use POOL directly (eg experiment framework developers)
    - CVS release tag "POOL_1_0_0"
    - Build against SEAL 0.2.1
  - Release notes at http://lcgapp.cern.ch/project/persist/relnotes.html
  - Please use savannah http://savannah.cern.ch/projects/pool/
- One supported platform so far
  - RedHat 7.3 using gcc 3.2
    - Preparations to extend the platform list for test builds underway in SPI
  - Upgrade to lastest SCRAM version
  - Aligned the release area with the agreed LCG policy SPI
- External packages hosted by SPI in /afs/cern.ch/sw/lcg/external

• MySQL	(4.0.4-beta)
• MySQL++	(1.7.9)
• ROOT	(3.05.04)
• Xerces-C	(2.1.0)
• Edg-rls-client	(1.2.11)

POOL and the SEAL Plug-in Manager

- POOL has already finished a “first round” of integration with the LCG plug-in manager (Radovan Chytracek)
  - Important advantages for POOL
    - No explicit linking (or runtime loading) of implementation libraries (eg ROOT, catalogs & collection implementations)
- For larger scale integration the issue of consistent sharing of some components eg in the experiment framework is still there
  - Who configures / initialises any commonly used component like ROOT, Message Service, …
  - Who keeps track of reference counts of shared components?
- This will need a second round of SEAL integration
  - Once the experiments have really accepted a common proposal

Storage Manager

- Ioannis Papadopoulos took over the Persistency Service
  - Stefan Roiser moved to SEAL already quite some time ago
  - (Almost) complete refactoring of the package
  - Removed the need to explicitly register classes in the user code
- Added (Markus Frank together with SEAL) support for more container types in persistent objects
  - Previously vector and list
  - Now also map, deque

POOL and Schema Evolution

- Very popular (and valid) question…
- Joint meeting between POOL, SEAL and ROOT developers about this issue
  - POOL relies on ROOT to do the (default) translation between disk representation and transient C++ objects
    - POOL instructs ROOT about the transient shape based on the SEAL dictionary
  - ROOT already provides support for evolving object shapes
    - This support currently relies on user provided version numbers which help ROOT to identify which C++ shape is used
    - These version numbers are not currently not provided by POOL or the SEAL dictionary
  - ROOT plans to remove the necessity for the user to provide the class version and discover it automatically

- Conclusion of the joint meeting
  - POOL should wait and use this new functionality and let ROOT deal with schema evolution rather that providing yet another (possibly inconsistent) implementation
  - Rene has announced this to become available not before this summer

Object Type and Placement Hints

- POOL allows to define the physical location where an object is stored via Placement objects
  - Keep track of storage technology, database file and container in that file to be used
- As of V1.0 we have removed the necessity to define the class of objects stored in the placement
- The real (runtime) object type is now determined later directly from the user object which is passed in

Object Ownership Policy Change

- So far POOL has expected the user to properly delete objects which have been marked for writing
  - Objects which were read were automatically garbage collected via reference counting
- As of POOL V0.5 this policy has changed
  - Any object which is passed to POOL for writing is “owned” by the POOL Cache and is destroyed after the commit automatically
  - References to these objects stay valid across transaction boundaries
- This new policy is much simpler and less error prone as the user code does not need to explicitly keep track of these written objects and synchronise their deletion with transaction boundaries

Collections & Meta Data

- Main collection developer left POOL (and HEP)
  - Kristo Karr only recently joined and will soon take over
- Outstanding issues
  - Agreement on Meta Data interface
  - Some error handling and scalability for large collections
  - Implicit collections (implementation not strictly speaking inside this work package)
- Complete reimplementation incorporating the new AttributeList interface and some changes wrt to iterator handling done by Jacek Wojcieszuk, Ioannis Papadopoulos and Helmut Schmuecker
  - POOL is now supporting collections and iterators of typed objects
    - Collection<T> and Collection<T>::Iterator
  - Collection and File Catalog now both using MySQL++
  - Not an ideal interface to MySQL (eg rather dangerous #defines) but currently the best we could find
  - Need to extract a common subset of required operations to prepare for a possible generic RDBMS layer (second half of the year)

Scalable Query Processing

- Have shown file lookup performance several times
  - Response time seems adequate for usecases we expect (pass in guid -> get one (or few) results)
- Queries with large result sets are more difficult
  - Eg query for all PFNs which start with “pfn:/cms*”
  - Eg query for all Events from the collection “LHCb2008”
- Two issues
  - Need to respond with first results quickly
    - Interactive programs tend to stop before the iteration is expired
  - Need to conserve memory on server and client side
    - Keeping very large result sets in memory on either side is wasteful/impossible
- Standard RDBMS problem: use a database CURSOR
  - MySQL has somewhat limited support in this area
  - Can fetch individual results
    - But either all or one – db roundtrips may affect performance
    - Need to retrieve the whole set even if one decides to stop earlier

ROOT / Hierarchical Collections

- Prototype by Helmut available in the POOL contrib area
  - ROOT tree based explicit collection
  - Supporting hierarchies of super-collections (other collections as leaf nodes)
  - Maintaining hierarchy of associated meta data (eg event tags)
  - Did not achieve complete integration of hierarchical collections with the new collection interface
- Currently collection models used by the experiments are quite different
  - Propose a AAM presentation of current state and request experiment guidance on which direction(s) to follow

Meta Data Interface - Attribute List

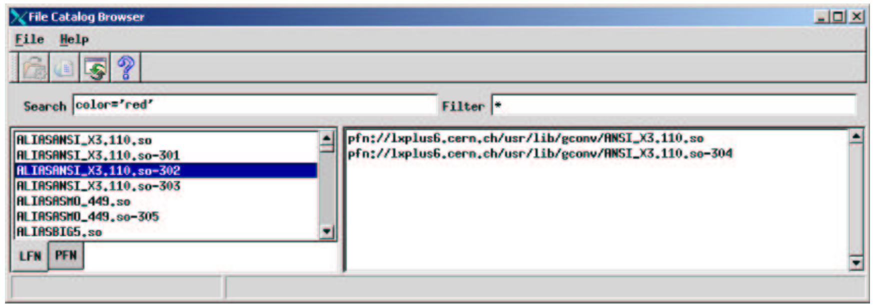
- Concluded long standing discussion on common interface meta data
  - Rather a simplified stop gap solution until complex object support on the RDBMS layer may become available via the standard storage manager interface
  - Two proposals (by S. Eckmann and J. Hrivnac) were included in parallel in earlier releases
  - New AttributeList class defined & implemented by Jakub Moscicki
- After investigating the AIDA Tuple interface we decided to use an interface close to Steve’s proposal
  - But using a more extensible templated mechanism to define and access attributes
  - Deployed with V1.0 for collections and file catalog

File Catalog Meta Data

- Maria Girone and Zhen Xie updated the File Catalog implementations to allow attaching simple meta data (C++ elementary types plus string) to each file
- File Catalog now supports ad-hoc queries based on this file level meta data
  - Goals is not to host all experiment meta data but rather the small subset which is relevant to define catalog fragments for export to another site, a laptop
  - Traditional approach of using grep eg on a logical file name does neither support complex queries, nor does it scale
- All three POOL file catalog back ends support the same interface
  - XML catalog can be used to define meta data at file creation time and carry it forward through the production process until it gets eg published into a larger grid-aware catalog
- Programmatic interface is the identical as for queries on eg POOL Event collections.

POOL File Catalog Browser Prototype

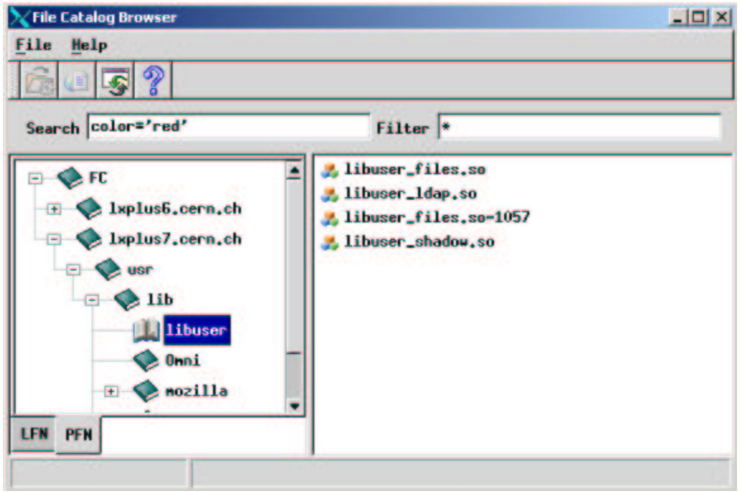
- Carmine Cioffi extended the python based prototype to access file meta data and support queries from the GUI interface



Search color='red' Filter *

```
ALIASANSI_X3.110.so
ALIASANSI_X3.110.so-301
ALIASANSI_X3.110.so-302
ALIASANSI_X3.110.so-303
ALIASASMO_449.so
ALIASASMO_449.so-305
ALIASBIG5.so
```

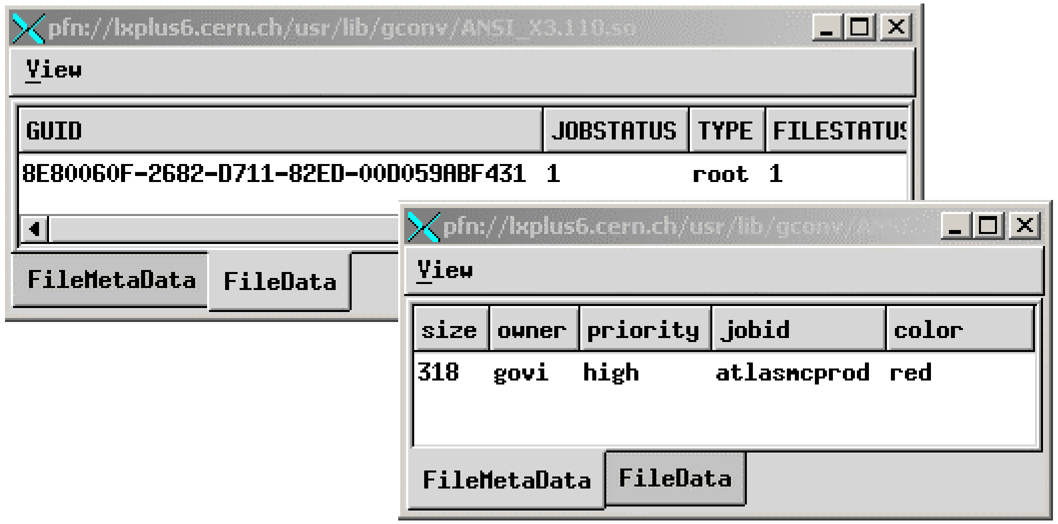
```
pfn://lxplus6.cern.ch/usr/lib/gconv/ANSI_X3.110.so
pfn://lxplus6.cern.ch/usr/lib/gconv/ANSI_X3.110.so-304
```



Search color='red' Filter *

```
FC
  lxplus6.cern.ch
  lxplus7.cern.ch
    usr
      lib
        libuser
        Omni
        mozilla
```

```
libuser_files.so
libuser_ldap.so
libuser_files.so-1057
libuser_shadow.so
```


pfn://lxplus6.cern.ch/usr/lib/gconv/ANSI_X3.110.so
View
FileMetaData
FileData
pfn://lxplus6.cern.ch/usr/lib/gconv/
View
FileMetaData
FileData

GUID	JOBSTATUS	TYPE	FILESTATUS
8E80060F-2682-D711-82ED-00D059ABF431	1	root	1

size	owner	priority	jobid	color
318	govi	high	atlasmcprod	red

Infrastructure and Testing

- The usual infrastructure upgrades
  - Upgraded to latest SCRAM version
  - Aligned release directory structure with proposal agreed between SEAL/POOL and SPI
    - Eg only one include directory for POOL users
- POOL test plan created and maintained by Giacomo Govi
  - Now running 20 integration tests (using Oval and CPPUnit) as part of the release
    - Some stress tests provided by Tim Barras
  - Many more unit tests added
  - Will keep reference output files for regression testing with upcoming releases

EDG RLS Test Service

- Together with the POOL V1.0 release EDG WP2 has started to provide a RLS test service for POOL
  - based on EDG-RLS running on Oracle 9iAS and 9i database
  - Thanks to the WP2 RLS team
- Please consult
  - http://edg-wp2.web.cern.ch/edg-wp2/replication/rls-for-pool.html

  for details on this service
- Basically
  - Point your POOL catalog to the service

```
POOL_CATALOG=edgcatalog_http://rlstest.cern.ch:7777/\
    edg-replicalocation/services/edg-local-replica-catalog
```

  - Run your unmodified POOL application
  - Report any service related problems to the RLS service support *hep-proj-grid-data-mgt-support@cern.ch*

Still in the queue…

- Implicit collections
  - Had to be dropped from the V1.0 list as we ran out of time because of the other changes in the collection area
- Performance optimisation
  - Almost no systematic optimisation attempt so far
- Work on the RDBMS storage manager prototype has started in Indore, India
  - Regular participation in work package phone meetings

More queue items …

- Support polymorphic containers
  - High priority request from CMS
  - vector<Track *> filled with different sub-types of Track
  - Container “owns” each element
- Update operations on streaming layer objects
  - High prio request from CMS
  - Append to an existing database
  - In place update (or delete) of existing objects (typically application meta data) without invalidating existing references
- Support for transient data members
  - Request from ALL, priority wrt. The others?
- Support for automated loading of LCG dictionaries
- Support for macroscopic transactions
  - Remove the burden from user code to keep track of individual component transactions

POOL V1.0, Functional Completeness and Interface Stability

- Did we achieve functional completeness with V1.0?
  - Maybe largely, but definitely not completely
  - Many important enhancement requests still come in as the experiments are really just starting to use POOL
    - Very useful feedback eg in the catalog area has resulted in significant changes to the interface in this release
- Need to be pragmatic and stay open for justified requests – especially if tagged as go/no-go features by at least one experiment
  - File Format - the answer to most questions will have to be “no” soon
    - But not before we have solid performance numbers
    - Only backward compatible changes - for which data types for how long?
  - Data Service and Refs (very exposed)
    - Changes are quite visible but luckily the interface is quite small and most anticipated interface changes (eg update/delete) are already in place with V1.0
  - File Catalog (visible to production – but not too much to readers)
  - Collections - essential concept in all experiment frameworks
    - But neither much agreement nor clear understanding how to integrate into a GRID environment yet
    - This will stay an area of discussion for the second half of the year

POOL needs for SPI deliverables

- Maintenance of LCG APPs external libraries for platform/compiler combination foreseen for regression test
  - also applies to SEAL releases

done. New platforms starting

- Automated ("Nightly") build system
  - Regression testing
    - subsystem & integration tests
    - platform/compiler regression
    - file format regression test
  - Can we get this in time for POOL V1.0 ?

POOL/SEAL will need to get in touch with NICOS people

POOL needs for SEAL deliverables

- Dictionary and Dictionary Generator
  - Pre-release of SEAL component is used now
  - Improvements for persistency support of more complex types, dictionary import export will likely be required
- Message Service and Exception Base
  - Centrally configurable diagnostic output
  - SEAL component used now, but very likely to change significantly
- Component Infrastructure
  - Component library loading and un/re-loading
  - Pre-release in time for LCG-1 ?
- Scripting Infrastructure
  - POOL component interface from scripting languages
  - Pre-release in time for LCG-1 ?
- Object Whiteboard
  - Relation to object cache to be clarified
  - To be addressed after LCG-1 ?

done

Exception done MsgSvc in 0.3.0

First round done.

POOL Experiment Integration…

- …has started already, but is expected to quickly become a significant work item for POOL
  - Individual POOL developers have already started to get directly involved in solving early deployment problems
  - Helps to cope with still insufficient POOL documentation and produces direct feedback to POOL
  - Propose to name a “link” person in POOL for each experiment
- Will soon open the POOL developer list to all developers who are involved in POOL integration
- For bug/support request please use savannah
  - Or even better – write a short POOL test which you’d like to have succeeding in the next release

Summary

- A bit later than expected, but still POOL V1.0 is out
  - Please use savannah as much as possible
- POOL V1.0 has significant improvements in several areas
  - Can cope with significant internal refactoring without affecting user code too much
- Still see continuous requirement/feature request flow into the project
  - Claiming full functional completeness or API stability before any pre-production with POOL has finished successfully seems premature
- We expect some very busy two months ahead of us.
  - Not that we don't already have them behind us.
  - Vincenzo will give some more reasons why…