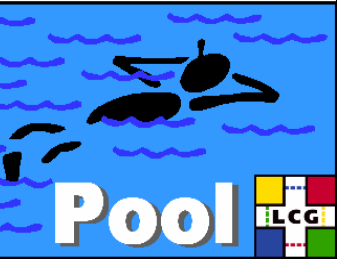

POOL Plans for 2004

ARDA / EGEE integration

Dirk Düllmann,

IT-DB & LCG-POOL

LCG workshop, 24 March 2004

POOL Focus in 2004

- Stabilise POOL s/w products
  - Focus on performance improvements rather than large functionality changes
  - In line with the experiments plans for the data challenges
- Help to simplify the integration into experiment frameworks
  - Tight coupling between POOL and experiment development and production teams
  - Automated schema loading, usability tools, documentation improvements
- Production release of Conditions DB
  - After a initial interface consolidation round
- Achieve POOL independence of the RDBMS backend
  - And extend the set of supported RDBMS systems

2004 Work Plan

- Draft document has been discussed in POOL and the Architect Forum
  - http://pool.cern.ch/POOL_Program_of_Work_20040307.doc
  - Based on WP work plans and experiment priorities
  - No significant objections received
- Hope to finalize the plan soon and present it to PEB and CS2
  - Significant overlap between the different experiment requests
  - Thanks to all experiments for their concrete and detailed input

POOL interest in ARDA/EGEE

- POOL will be one major client of ARDA/EGEE services
  - ARDA will be a framework for running POOL based (analysis) tasks
  - Currently POOL is mainly used in production activities
    - POOL will need to address analysis area this year
  - POOL needs to be able to integrate with middleware (EGEE) provided services
    - Mainly catalogs, file/database access, meta data
- POOL will align with the main ARDA concepts and implementations
    - In particular Collections, Filesets and File

ARDA/EGEE interest in POOL?

- POOL provides deployment models, interfaces and implementations as input to ARDA
  - Eg model for cross population between implementation using catalog meta data queries
- POOL file catalog may be seen as an implementation of file catalogs
  - See first successful use of POOL file catalogs in production activities
    - POOL use of cascading catalogs seems to be well accepted and should be considered is input to ARDA/EGEE
  - Also see first performance problems and design traps for backend services (eg the current RLS - see slides further on)
- A consistent distribution (extraction and publishing) mechanism will be required for all services which keep/provide meta data
  - This should be designed into the various catalogs, conditions db, …
    - Ie support for meta data based query and bulk update from eg XML
  - Could greatly profit from a single basic model and not a different treatment per meta data service

POOL Collections

- Role of event level collections and meta data in a grid environment needs clarification and prototyping
  - Expect active collaboration with ARDA to come up with a model for deploying collections in production and analysis environments
  - Integration of POOL collections with experiment frameworks just starting
  - Still many open questions about requirements
    - Need a collection catalog (with associated meta data)
      - Re-use of file catalog interface and implementations?
      - Or are collections/datasets just more general files?
    - Consistent catalog & meta data distribution – how ?
      - XML on application level? Directly between backend databases?
    - What collection meta level data needs to be kept?
- Collection implementation in POOL is only a first step
  - The real issue is not the implementation but a conceptual model which fits with the GRID provided services
  - Need active experiment involvement (and some agreement) in this area
- How do POOL collections tie in with ARDA and EGEE?

Collections as End-to-end Services

- End-to-end services in a layered system
  - May involve several concepts/services
  - Each of which may "live" on a different layer
- Example: POOL Collections
  - POOL (Event) Collection - end user concept
    - Access to collections of POOL objects
    - Integrated with PROOF analysis back end
  - Fileset – ARDA layer concept?
    - Access to abstract data from a set of closely related files
  - File – EGEE layer concept?
    - Access to a unstructured set of bytes in file somewhere on the grid

Collections & ARDA

- Joined Work Package between POOL and ARDA
  - Still some uncertainties concerning the ARDA side of the work plan
  - Will continue work to address the outstanding issues on the POOL side
  - POOL has asked experiments for principal contacts in this area
- Collection cataloguing, extraction and publishing tools
  - Can we achieve a single baseline model for distributed meta data catalogs?
    - File Catalog, Collection Catalog, Conditions Folder Catalog
    - One basic mechanism of data exchange across RDBMS vendor boundaries based on the POOL relation abstraction layer?
- Separation of logical and physical collection identification
  - Introduce a Collection (Fileset?) catalog
  - First implementation could simply be based on the existing File Catalog components, but as a separate service
- Integrate POOL collections with ARDA provided services
  - Migrate to ARDA/EGEE provided catalog and meta data(?) services

POOL Storage Manager Plans

- Optimisation required in several areas
  - Client side resource usage - memory, CPU, file handles (Q2)
  - Mass storage handling (minimise costly requests) (Q3)
  - "Transparent" double to float mapping (Q3)
- Automated schema loading (Q2)
  - Based on SEAL service
  - In cooperation with ROOT team to allow late integration of data types for already open files
- Bug fixes - more complex cases
    - Eg std containers with user defined allocators which define local data – aka CLHEP Matrix (Q1)
- RDBMS backend based on the RDBMS Abstraction Layer
  - Storage of simple data structures into a RDBMS via the same interface as for objects stored on the streaming layer
  - Two step plan:
    - First allow for objects which can trivially be mapped to SQL tables (Q2/Q3)
    - Possibly later an extension to more complex C++ objects (Q1 '05?)

POOL and Grid Files

- POOL needs to access data from the grid
  - POOL rarely deals with files directly - almost all I/O is delegated to the storage backend (ROOT or RDBMS)
  - Open problem of integration with storage element interface
    - How to access data from a file somewhere in the grid
      - Without hardcoding to the (mass) storage backend
    - Currently POOL can handle any files which ROOT can open
- SRM integration and data access to files on LCG storage elements needs to be provided
  - We share problem with vanilla ROOT
- Proposal: work together with ROOT team to make sure that the LCG SE integration into ROOT fulfills both and POOL needs

POOL and RDBMS

- Today POOL stores all user data in ROOT files
- During this year we plan to provide storage of objects also into relation database back ends
  - Required for frequently updated meta data, complex server side query support
- Will need to register backend database connections with associated meta data in a very similar way as files
    - In the file catalog?
    - In another dedicated database catalog?
- Other grid services may require access to registered logical/physical database connections as well
  - Is there a need for another catalog?

POOL File Catalog

- Significant developments completed already (Q1)
  - Support for LCG-2 (V1.5)
  - Support for Composite Catalogs (V1.6)
- File Catalog as model for handling and exchanging data could be a prototype for other (very similar) meta data catalogs (Q2/Q3)
  - Collection catalog and Collection entries
  - Condition Folder catalog and Condition Data
- Cataloguing, extraction based on meta data, publishing are all very similar
  - Even the catalog component implementation could be factorised out and shared
  - Performance of XML as exchange format for larger data amounts needs to be evaluated

POOL/RLS Experience

Current CMS Data Challenges shows clear problems wrt to the use of RLS

- Partially due to the normal "learning curve" on all sides in using a new systems
- Some reasons are
  - Not yet fully optimised service
  - Inefficient use of the query facilities
- POOL and RLS service people works closely with production teams to understand their issues
  - Which queries are needed?
  - How to structure the meta data?
  - Which catalog interface?
  - Which indices?

More POOL/RLS Experience

- But poor performance also due to known RLS design problems!
- File names and related meta data are used for queries
  - Current RLS split of mapping data from file meta data (LRC vs. RMC) results in rather poor performance for combined queries
  - Forces the applications (eg POOL) to perform large joins on the client side rather than fully exploit the database backend
- Many catalog operations are bulk operations
  - Current RLS interface is very low level and results in large overheads on bulk operations (too many network round-trips)
- Transaction support would greatly simplify the deployment
  - A partially successful bulk insert/update requires recovery "by hand"
- These are not really special requirements imposed by POOL
  - Still acceptable performance and scalability needs a catalog design which keeps the data which is used in one query close to each other
  - Try to work around some of this know issues on the POOL side

Summary

- POOL Focus for 2004
  - Consolidation and Optimisation
  - RDBMS vendor independence
  - Common model for distributed, heterogeneous meta data catalogs
  - ConditionsDB production release and integration with POOL
- POOL will be a major ARDA/EGEE client
  - Needs to stay aligned with ARDA concepts and EGEE services
  - Provider of persistent object storage and collections
- Joint work package between POOL and ARDA in particular in the Collections area
  - Need more active experiment involvement
- Gaining valuable real life (data challenge) experience with POOL/RLS as input for next round
  - Produces concrete experiment requirements as input to ARDA/EGEE
  - POOL may be able to workaround some of the RLS design problems
- A real solution will be required from ARDA/EGGE to achieve the performance and scalability goals

Input for a next software generation

- Catalogs of "things" annotated with their meta data exist all over the system
  - These catalogs services could/should share the implementation and distribution mechanism
- Separation of catalog mapping data from associated meta data makes meta data almost useless
  - Efficient queries require that mapping and meta data are handled by (in!) one same database backend
- Higher level interface for bulk insert and bulk query is required
  - The current use of SOAP RPC call for each individual data entry will not scale to larger productions
- Transaction concept is required for a maintainable stable production environment
  - User transactions may span span several services!