POOL and SEAL integration in ATLAS

- Contributions from
  - D. Adams, C. Arnault, V. Fine, H. Ma, RDS, A. Undrus
- Current overall status:
  - For the past two months we have been slowly adding read/write capabilities to different parts of ATLAS standard event model
  - A few "experts" set up the infrastructure using various examples from out event model.
  - For the past month, a documented procedure has been used a wider user community
  - We have so far only made limited I/O tests with these examples

Near term planning

- Focus in coming months: deploy and test a reasonable prototype
- ATLAS Computing Model in the time frame of Data Challenge
  - Model is still being defined - Computing Model working group preliminary report due in October
  - Note that DC2 is intended to provide an exercise of the Computing Model sufficient to inform the writing of the Computing TDR
- Ambitious development agenda required
  - See database slides from July ATLAS Data Challenge workshop: http://agenda.cern.ch/askArchive.php?base=agenda&categ=a032032&id=a032032s1t13/moreinfo
- Tier 0 reconstruction prototype is a principal focus, as is some infrastructure to support analysis
- DC2 dates:
  - Simulation/pileup April 2004, Tier 0 reconstruction June 2004

Near term planning (2)

- Another high-priority item is a combined test-beam run in summer 2004
  - Use of SEAL/POOL concerns only the conditions data
  - We are planning to use the ConditionsDB as an "interval of validity" storing intervals plus references to conditions objects stored in pool

Integration of the SEAL Dictionary

- For the SEAL dictionary, we follow the procedure elaborated for ADL (Atlas Dictionary and Description Language)
- Our classes packaged separately by DataObjects and Algorithms
  - Data pkgs: XXXEvent, YYYDetDescr, ZZZConditions
- In each data pkg, we build a DLL for the SEAL dictionary fillers
  - Clients specify a set .h files in a CMT pattern
  - Currently require users to list all classes needed for I/O in selection file
  - Loading of lib dictionary fillers is coupled to the loading of the AthenaPool converters (see later) - temporary solution
  - Have a simple AthenaSeal service which can load a set of dictionaries and check completeness

Integration of the SEAL Dictionary (2)

- No major problems generating dict-fillers for ATLAS data model since SEAL 0.3.3
  - GuineaEvent has checked major ATLAS data model features
    - Still need to provide SEAL/POOL test based on GuineaEvent
  - Small problem - support for enum's - have simple work-around
  - However, it is not sufficient to see that lcgdict can handle our classes, because there may POOL <=> SEAL Dict interaction that cause problems
    - GuineaEvent itself has not been written out
  - Missing support for StoreGate infrastructure:
    - Typedefs needed to support our "inter-object links"
      - I.e. references between objects below the "DataObject" level

Integration of POOL into Athena/Gaudi

- From the framework point of view, POOL is just a new I/O "technology"
  - This implies writing a new conversion service
  - Design work began in January, and has been integrated in releases following the POOL releases since February
  - Main components:
    - AthenaPoolCnvSvc - conversion service
    - AthenaPoolConverter - converter base class
    - AthenaPoolCnv<T> - templated converters
    - PoolSvc - Athena/Gaudi service interface to POOL
      - Allows jobOptions
    - EventHeader - stores the refs of the Event Data Objects
      - Ref to EventHeader is inserted in the event collection

Integration of POOL into Athena/Gaudi (2)

- We have simplified the user interface by allowing "generic" converters:
  - Use templated converter and we generate the necessary classes to create the converter
  - User just needs to specify a .h file for each DataObject (pool ref'ed object) to be stored
  - Expect that this should satisfy a large fraction of the I/O needs
- We are using explicit pool collections
  - Default user configuration is to use explicit root collections
  - Expect to explore the use of both explicit root collection and MySQL collections in DC2

General comments on integration

- Many nuisance technical obstacles to POOL integration into ATLAS
  - Not long-term concerns, but in the first year, they consume a great deal of time of time
- Integration of POOL into ATLAS/Athena has not been trivial
- Examples
  - Conflicts in how cmt/scram/ATLAS/SPI handle build environments, compiler/linker settings, external packages and versions, …
  - Conflicts between Gaudi/Athena dynamic loading infrastructure and SEAL plug-in management
  - Conflicts in lifetime management with multiple transient caches (Athena StoreGate and POOL DataSvc)
  - Issues in type identification handling between Gaudi/Athena and the emerging SEAL dictionary
  - Keeping up with moving targets (but rapid LCG development is good!)
  - Figuring out "gotchas" like inability to write classes with private constructors
    - Obscure error messages - had to read the code

Issues

- Initialization of transient members
  - There are different reasons for not writing out all data members
    - Optimized cache of information
      - E.g. HEPMC mapping of ids to particle ptrs
    - Connection to objects not being written out
      - E.g. CaloCell having ptrs to their "cell geometry"
    - Change of I/O granularity between online/offline
      - Three level containers: container -> collections -> objects
  - We are currently trying out "custom converters" where the generic converters are used for I/O but we allow hooks for "pre-write" and "post-read" modifications
- Need to fully understand and clearly spell out for users the restrictions/constraints on the C++ model
  - Default constructor now have "special meaning"
  - Pointers require virtual tables

Issues (2)

- Type id problem
  - Currently required to have UUID for persistent class
  - Athena/Gaudi classes already need a CLID
  - It would be nice to remove any requirement for these type numbers. When and how?
- Simplifying managing dictionary fillers
  - We are glad to hear that plugins will soon be used to simpify the loading of the dictionary fillers

Issues (3)

- The storage of our full data model is not yet complete
  - Internally, StoreGate uses "refs" called DataLinks and ElementLinks
  - We do not yet have a solution to this, but some examples have been provided from the POOL team
- Storage of CLHEP matrices
  - Matrices are stored as "double * m" which fails to be stored by pool
  - Need to provide the length, e.g. via the selection file?

Distribution

- ATLAS is currently finalizing its sw dist developer kits using pacman generated with CMT
  - Allows "one-line" download of a particular (release/sub-project) pair
    pacman -get 7.0.0/AtlasDistSetup
  - And running simple pool example:

```
export ATLAS_RELEASE=7.0.0
export SITEROOT=/afs/cern.ch/atlas/software/dist/trials/v-b
source ${SITEROOT}/setgcc32.sh
source ${SITEROOT}/setup.sh
export JOBOPTSEARCHPATH="`pwd`${JOBOPTSEARCHPATH}"

athena.exe AthenaPoolExample_WriteJobOptions.txt
athena.exe AthenaPoolExample_ReadJobOptions.txt
```

  - One can also check-out and develop against the "kits"

Distribution (2)

- CMT deduces the pacman statements from the requirements files
  - E.g. "depends" between kits - the installation unit, one per package
- "sub-projects" are the consistent down-loadable units
  - pacman -get SEAL_1_1_0/SEAL_Release
    - Gets SEAL plus all of its required external packages
    - Binaries/includes/data files
  - Easy to define new sub-projects with CMT, e.g. to have configurations with different dependencies
- Externals have two kits:
  - One with the dependencies of the external package (CMT glue package)
  - One with the re-locatable external itself

Distribution - what do we want from LCG?

- We can generate the pacman kits for POOL and SEAL
- If SPI were to do so, via CMT or other, we need
  - Agreement on naming conventions
    - <subprojectid>/<package> e.g. SEAL_1_2_0/…
  - For externals, to have two kits - one for the "glue" package, and one for the software
    - <subprojectid>/<gluepackage> e.g. SEAL-1-2-0/mysqlpp
    - <externalid> e.g. mysqlpp-v1
  - Relocatable kits depending upon on a single env variable:
    - (${SITEROOT} == ${PACMAN_INSTALLATION})
- More generally, it would probably be useful for ATLAS and LHCb to maintain the cmt "glue" packages for externals somewhere in the LCG afs area

Conclusions

- SEAL/POOL is beginning to be used by a small set of developers in ATLAS
- We have not yet been the most demanding client, but we have seen good respond for problems we see as important
  - E.g. solving DataLink problem or requiring transient attributes to have their type defined