POOL and SEAL integration in ATLAS

- Contributions from
  - D. Adams, C. Arnault, V. Fine, H. Ma, RDS, A. Undrus
- Current overall status:
  - We are currently able to write/read a small selection of ATLAS standard event model, so far done by “experts”
  - We have documented the procedure and are just now opening up to a wider user community
  - Have only made limited tests beyond the limited set of 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

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
  - Only (small) known outstanding is implementation of enum's
    - Have 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

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

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"
  - It is not clear how to best deal with this, and we are beginning to explore different possibilities
    - E.g. patterns for cache initialization on first access, or customized converters
  - Not clear if there are any requirements on SEAL/POOL
    - Understand that there are now some examples of "custom streamers"
  - It may be useful to somehow have "hooks" for prepareForWrite or prepareAfterRead
  - In any case, people must now be aware that the default constructor is used by POOL and thus has a special "meaning"

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
  - Would like to be able to load dictionary fillers "on-demand"
    - I.e if a class is not found in the reflection dictionary, then the library to fill it will be automatically loaded
    - Understand that there is work in progress in SEAL to do this
  - Simplify the selection file specification
    - Instead of specifying by hand all classes, should only have to say "all classes needed for these .h files", excluding those definitions defined elsewhere

Issues (3)

- The storage of our full data model is not yet complete
  - Internally, StoreGate uses "refs" called DataLinks
  - We do not yet have a solution to this, but some examples have been provided from the POOL team

Conclusions

- SEAL/POOL is just becoming functional 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
- We actually look forward to understanding in more detail the capabilities and working with the SEAL/POOL teams to extend them