Argonne NATIONAL LABORATORY

Potential for greater commonality in our approaches to I/O?

David Malon
malon@anl.gov

Gaudi Workshop
26 September 2017, CERN

U.S. DEPARTMENT OF ENERGY

Background

- It seems that I/O components and I/O design have not been central to discussions and action items in recent Gaudi workshops
  - Though there have been some discussions about thread scheduling for I/O-oriented and other “blocking” tasks
- Athena inherited from Gaudi its essential I/O infrastructure design—event selectors, outstreams, persistency services, conversion services, and converters
  - But implementations and extensions diverged early
- Lots of independent development since (shared readers and writers for multiprocessing, support for metadata reading/writing, …)
- Would it be worthwhile to explore joint I/O infrastructure evolution and the potential for greater commonality in our I/O components and strategies, or is this hopeless?
- It would be nice if progress in efficient multithreading for I/O and in leveraging developments in ROOT I/O (and possibly other advances) could be shared more directly among experiments. This might be easier if we shared more of our I/O components.

Prospects for a shared ROOT conversion service?

- Peter van Gemmeren has explored what it would take to use the Gaudi RootCnvSvc from Athena
- This work is in preliminary stages, but with a few changes to the Gaudi code and to Athena configuration Peter has succeeded in writing simple objects from Athena via the Gaudi RootCnvSvc
  - Not yet xAOD
- Some recent slides from Peter illustrating the changes are attached to the agenda
- My own high-level summary is this:
- The primary obstacles to a shared ROOT conversion service arise from LHCb-oriented assumptions made by the current Gaudi RootCnvSvc
  1. Regarding the transient store, and
  2. Regarding transient object naming and the mapping of transient names to persistent data organization
- Daya Bay too needed to develop its own ROOT conversion service, largely because of transient store issues and assumptions
- Differences among ATLAS, LHCb, and Daya Bay transient event stores are distinctly non-trivial, but perhaps not all of them need to be resolved in order to share at least some I/O services

LHCb-isms in Gaudi RootCnvSvc

- Gaudi RootCnvSvc expects everything to be a DataObject
- ATLAS can work around this by returning a DataObject* without “being” a DataObject
  - Certainly doable from an Athena DataBucket
  - Straightforward to change RootCnvSvc so that this is good enough, in a way that should work for both ATLAS and LHCb (cf. Peter’s code)
  - Is there a willingness to do this?
  - Is it worth doing?
- Is this dependence upon DataObject really required in the long run?
  - Is LHCb reliance on DataObject inheritance diminishing (cf. discussions of AnyDataHandle), or is the AnyDataHandle construct mainly intended to make certain functional programming interfaces a bit easier?

Other LHCb-isms in Gaudi RootCnvSvc

- Other LHCb-isms appear in assumptions about object names
- Conversion service assumes a hierarchical naming scheme, searching for “/” and so on
- Conversion service decides persistent data organization (branches, etc.) and naming based upon the transient object name in a way that is experiment-specific
- Related things done in implementation if not in design (e.g., packing transient object names into persistent references, …)

Worth doing?

- Is it worthwhile to try to provide a shared ROOT conversion service?
  - No immediate improvement in functionality or performance, but perhaps a starting point for jointly leveraging each other's developments in I/O multithreading and in exploiting emerging improvements in ROOT I/O
  - and ideally a slightly reduced code maintenance load
  - Probably no running experiment would want to change how it maps its transient event data model to ROOT storage at this point for the sake of a common approach to persistence
    - though it is perhaps premature to worry about whether this should be necessary
- Peter is has provided some diffs, and is willing to discuss these and supply corresponding merge requests if there is interest in adapting the current Gaudi RootCnvSvc to support simple object writing from Athena
- … and/or one could imagine a broader rethinking a common ROOT conversion service, one that is not a patch to one experiment's conversion service to support another
  - This may be called for in any case if we hope to support more than ATLAS and LHCb
- Note too Attila's and Martin's work presented yesterday—in which there is no need for a conversion service at all

Other components to explore?

- If there is a shared interest, there are other “obvious” candidates for commonality
  - FileCatalog infrastructure, for example
  - ATLAS uses something close to the original POOL file catalog infrastructure, from which LHCb diverged somewhat some time ago
  - Conceptually, though, perhaps not so different, and perhaps not so different in their requirements
- Is there in any case an interest in looking at the I/O-related components more generally and more systematically to explore potential commonalities?

A related topic for discussion: shared approach to serialization?

- Many use cases for object serialization—serialization is not just a precursor to writing to disk
- ATLAS has for years serialized high-level trigger results—objects in the Athena transient store--into a buffer stored with other (readout) buffers in its raw data files
- Object (and full-event) serialization are also useful and used for sharing data among processes
  - Which may in principle be local or remote
  - And ATLAS has an increasing number of use cases for this, with its distributed event service and event streaming service infrastructure
- Could we imagine a shared Gaudi design and implementation for this, or should this capability remain experiment-specific?
  - One possibility is to (re)use the conversion service and converter infrastructure for serialization (a streaming conversion service), factorizing it from writing, but there are others

An almost-unrelated thought

- We have adopted different approaches to in-file metadata and metadata handling.
- This is in flux in ATLAS; I'm not sure what is happening in LHCb
  - I remember hearing about LHCb File Summary Records at some point, though perhaps those are ancient history by now.
- Is there any interest in Gaudi-wide discussions of framework support for metadata, and possibly for in-file metadata handling?