AMI, Metadata, and Software Infrastructure

David Malon
malon@anl.gov
30 August 2010
ATLAS AMI and Metadata Workshop

Introduction

- Assigned title seems to be “metadata: technology overview”
- Will not exactly be a technology overview
  - Lots of technologies related to metadata will not be discussed
  - But I do represent the software project side, not data preparation, not physics, not distributed analysis, ..., so I suppose I do bring a technological perspective
- Many technologies at work in support of metadata
  - ELSSI and TAGs for event-level metadata
  - COOL for interval-of-validity and lumi-block-level metadata
  - RunQuery as an interface thereto
  - XML DTDs as exchange formats (e.g.,for Good Run Lists)
  - Relatively sophisticated in-file metadata infrastructure (storage, incident handling, …)
  - AMI at the dataset level
  - Sources of metadata for AMI, tools that transport it, …
- Possible focus on technology related to getting metadata into and out of AMI
  - But Solveig, who knows many aspects of this work better (and who wrote her slides earlier!), will cover much of this

Earlier work

- Metadata and luminosity task forces
  - reasonable job of enumerating and classifying various kinds of metadata
  - and describing a few key requirements and use cases
- These task forces did not, however, provide much guidance on required tools or infrastructure or architecture
- Nonetheless, the collaboration has developed an array of tools that meet most of the needs foreseen in those efforts
  - Not always with an overarching architecture, but …

Sources of metadata

- General consensus after AMI review: physics metadata about datasets should be in or mediated by AMI
  - As opposed to in middleware and DDM catalogs/databases
  - Includes higher-level datasets (physics containers, super-datasets; we used to talk about hierarchical datasets when the meaning was less limited (i.e., before they existed))
- There are always items on the boundary (is this datum a physics metadatum?)
  - Provenance is an example, but clearly provenance has a physics aspect, and should be recorded in AMI
- And there are physics metadata about files
- Review followups required some reverse engineering by AMI team to extract data from other sources (task request databases, production system databases, T0 management databases, …)
  - Not necessarily intended as a long-term strategy
  - Perhaps this workshop provides an opportunity to think about longer-term strategy to accomplish our collaborative goals

Levels of metadata from production

- Some metadata known *a priori*
  - At task definition time
- Some metadata known only after the last job completes and output is checked/validated
- We mix these sometimes
- Some metadata are the same for all jobs in a task
- Some metadata are the same for all files written by a job
- Some metadata vary by file
- Currently, in distributed production, much of this metadata is reported redundantly, repeated for every file
  - We return far too much redundant metadata
- Improvements to reduce this were discussed and some were even coded, but apparently not put into production

Metadata transport

- Technologies for metadata transport are different for Tier 0 than for distributed production
  - Job report pickle files versus metadata.xml files, and so on
  - Metadata xml files are a legacy format: reuse of POOL file catalog DTD and its limited provisions for metadata
  - Alvin Tan proposed using Python dictionaries in place of xml files, but …
- Metadata packaging and transport from jobs should probably be revisited

Task-level metadata and AMI

- Discussion at the time of the AMI review of a tighter integration of task request infrastructure and AMI
- Much dataset-level metadata is known at the time the task that will create the dataset is defined
- Introduction of configuration tags (“AMI tags”) has been a major improvement in this direction
  - Configuration tags provide clear documentation of Athena configuration used to produce an output dataset

Transforms and metadata

- Metadata packaging for return to AMI (and elsewhere?) is handled largely at the transform level
- Transforms have grown into something of a control framework of their own
  - Production step sequencing, but also input file peeking for configuration, output post-processing and limited validation of sorts, metadata handling
- Plans years ago to rationalize the transform infrastructure
- Slow progress for various reasons, and now the developer is leaving ATLAS
- An important part of transform improvements will be ensuring that metadata are correctly and robustly packaged and returned
  - Have seen problems here

→ Transforms generally, and metadata handling by transforms, need work

Metadata and files

- If you have a file, can you figure out what's in it?
  - Which release produced it, which runs and streams and lumi blocks are in it, …
- If you don't have a file, can you figure out what you're missing?
- We're getting pretty good at the first of these
  - Lots of work on in-file metadata infrastructure and content
  - Even for eventless files (from sparse selections, etc.)
  - Example: ability to auto-configure jobs based upon file contents
- We're not quite as good at the second
  - But that's where AMI should help
- Lots of work on correct propagation of in-file metadata from input to output
  - {run, lumi block} ranges, for example
  - Can even merge N eventless files and get the metadata "right"

  → Still some work needed to make this robust, though most standard cases are handled reasonably well

Metadata and auto-configuration

- Can largely auto-configure jobs to process data files by peeking at contents
- Have worked hard to make this robust
  - Can usually do this even when the first input file, or several, is/are eventless
- Over-reliance, perhaps, though, on peeking
- In most cases, it should be possible to configure all jobs in a task from task- and dataset-level metadata, before the first file is opened

Executing jobs and metadata

- Athena jobs have little access to metadata
  - Metadata about the input dataset? About the task?
  - In a simple {dataset in, dataset out} task, try to find out from inside Athena the name of the dataset to which the job's input file belongs
- Writing metadata? Suppose you want to add a metadatum to the output
  - There are several ways that information can get into the metadata.xml file
  - From inside Athena, none is pretty
  - There are no Services for this
- Data currently written from executing jobs can be iffy
  - Event counting seems inconsistent and data-product-dependent, for example

→ This is an area that needs thought, and work (thought first)

Sundry issues

- Have seen some issues with how job status and metadata are reported
  - Particularly for compound (multi-step) jobs
- Implications of this and other factors on correctness of dataset “ready” status
- Others will speak of this, I think
- Provenance remains limited and somewhat fragile
- Have seen cases, e.g., of AOD containing more events than ESD
  - Would be helpful if information in AMI could help us detect this and track this down
  - And if usable metadata could record this and similar issues
  - Would be even more helpful if our validity checks at several levels would find this early
- Input of simulation metadata seems fragile
  - And there are recurring issues related to use of run number in simulation
- File naming differences between Tier 0 and distributed production
  - Metadata in the file names is different
- Are the ways people use AMI different than one might have anticipated?
  - If most people start with a stream and period, for example, or a related container dataset, and apply a Good Run List late, or if they use their working group’s D3PD, what is the role of AMI for them?

Would be nice …

- Stream-dependent offline “quality” flags?
  - More generally, a means to flag files that are problematic or interesting or anomalous in some way
  - There is a start on this
- Can we answer questions like, does the AOD for period E2 contain AntiKt6H1TopoJets?
  - What is the EDM content of a given dataset?
  - Some kinds of queries/provenance questions are difficult without reading (running?) all the nested job options that went into a job’s definition
- Remember History Objects?
    - Unless you are a core software or event data model developer, probably not
    - Object model support for event and object provenance
    - Probably overkill—but is there worthwhile middle ground here?