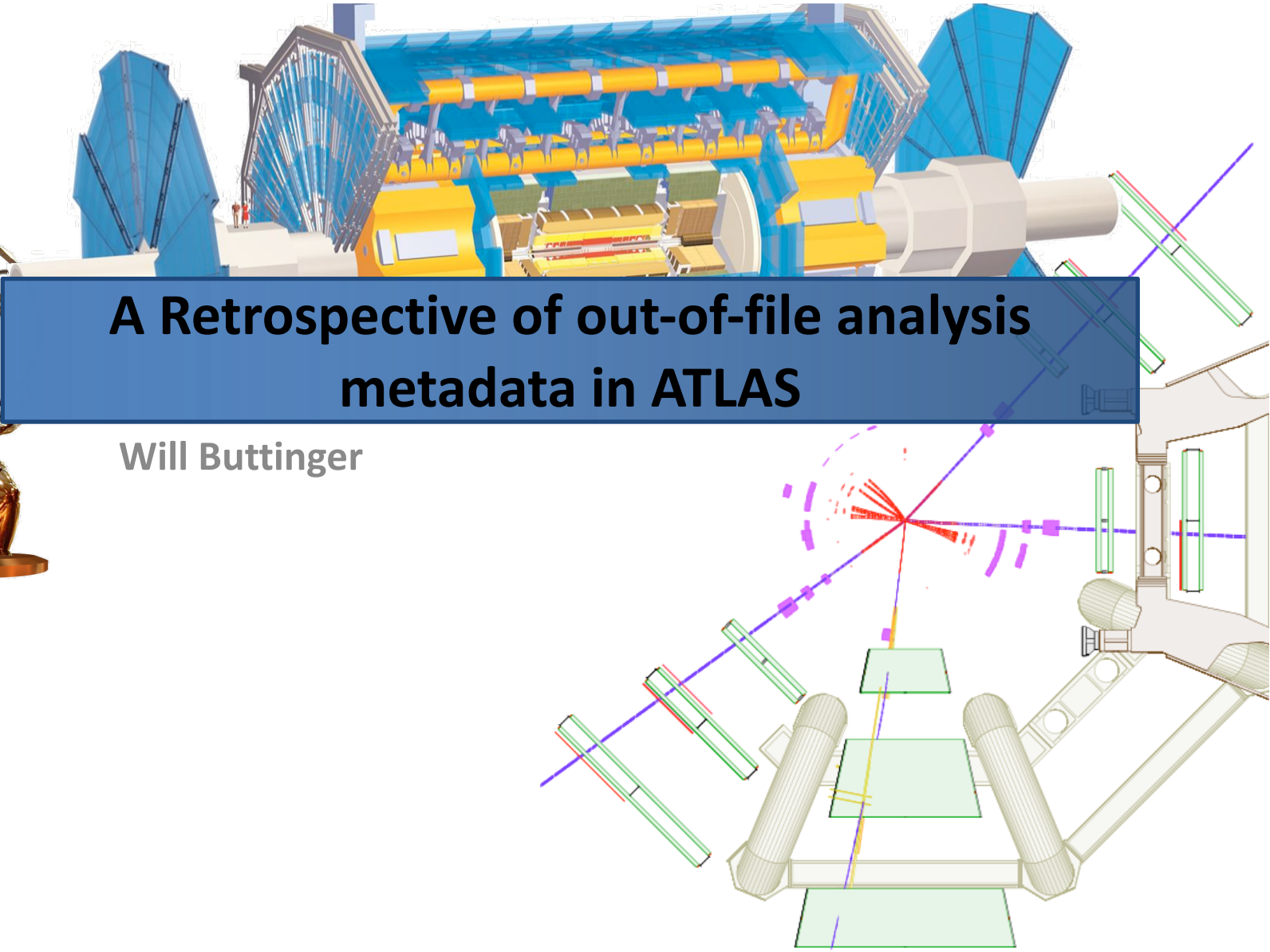


A Retrospective of out-of-file analysis metadata in ATLAS

Will Buttinger

Content

1. Define/describe analysis metadata in ATLAS
2. Outline some requirements for metadata storage
3. Compare and contrast the different systems in use by ATLAS
4. Case study of one system
5. Finish with some thoughts on future systems

Introduction

What?

- **External metadata:** Information that isn't available at the time the main data is produced, so is stored externally to that data
  - Our main data currently resides in files, in which case this is *out-of-file metadata*
- Examples:
  - Corrected/updated/improved cross-sections for MC datasets
    - Value extracted at generator-time is either superseded or just wrong
  - Other dataset-specific metadata that is (re)computed after full dataset has been built (extended)
    - E.g. a reweighting requires knowledge of a distribution over the full dataset
  - Calibration parameters/inputs calculated/updated by performance groups
    - Valid for "*campaigns*" (collections of datasets)

Some key system requirements

1. Association: Connect/relate external metadata to data and vice versa
2. Bookeeping: who provided the metadata, when, **versioning!**
   - Different users want to be using different versions of the same piece of metadata
3. Location+Access: How user tasks will access this information
   - On both local and distributed systems
4. Data Format: ranges from a single number to a very large file (order GB)

Solutions in use in ATLAS

Some possible places …

GitLab

Analysis Software Releases

- Put metadata in gitlab and have it distributed as part of weekly software release.
- Develop bespoke s/w tools to map data onto metadata for that particular type of metadata

cvmfs

Calibration File Area

- Filespace on afs (now eos) that is automatically replicated via cvmfs.
- Got a (large) file you want available everywhere with cvmfs access? Put it here

ami

ATLAS Metadata Interface (AMI)

- An automatically-populated database of all datasets produced by ATLAS.
- Has existed for well over a decade but only 5 years ago gained ability to have non-admins write additional metadata of datasets to it

System Comparison

...

	Pros	Cons
Analysis Software Releases (distributed ~ weekly in ATLAS)	• Bookkeeping and versioning (thanks to git) • Easy access • Coupled to compatible s/w	• Release bloat • Not automatically updated (but could be with a bot?) • No fine-grain control over mixture of metadata versions to use • No additional/updated metadata without s/w update too
Calibration File Area	• Easy access • Simple to understand -> popular • Good for large files • Access via s/w utility that user could override file area path of: use for local caching	• Manual bookkeeping and versioning – error prone • Had most of it be write-once but overwritable section is heavily used in analysis work
AMI Database	• Most powerful option for association, particular for dataset-level metadata • Was automatically updated with datasets as they were produced	• Hard to access (e.g. must be online, certificates etc) • Hard to update • Wasn't built with required bookkeeping in mind • Not good for file-based metadata • Confusing, not simple
CREST (a new database with a RESTful API)	• An actual database, like AMI, but better suited for blobs • Bookkeeping/versioning	• CREST not dataset-indexed like AMI • Access when offline? • API was still under development

A Case Study: MC Dataset Cross-sections

2015

- Twiki page from MC Gen team of cross-sections indexed by DSID
  - Analysis teams copied subsets of this info into text files
  - Some physics groups ‘centrally maintained’ text files
  - Teams/groups added/amended/deleted their text files at will
    - Adding metadata of DSIDs not on the twiki page (e.g. signal grids)
    - Altering cross-sections for some specific analysis requirsements
    - Augmenting with additional DS metadata if not readily available from in-file metadata
- A cross-section for each dataset was listed in AMI, but status quo was to “ignore it because it’s probably wrong”

Solution?

- Allow updated cross-sections and other dataset metadata to be appended to datasets in AMI
- Provide CLI to query database and produce a local cache of this metadata for required datasets

A Case Study: MC Dataset Cross-sections

Intended usage

- Admins maintained a list of allowed metadata fields
- Group contacts could submit (via CLI) values for these fields per dataset
  - Each group could have their own value
  - Additionally, values could be indexed by a subProcessID, in case dataset contained multiple processes that needed their own xsecs
- All inserted values timestamped and authored with a ‘reason’ (commit message): a full history of metadata values was kept
- Users would call CLI with a list of datasets, desired fields, and group preference order. Receive back a text file with all the metadata
  - File included comment at bottom with command to reproduce the metadata
  - Metadata values could be obtained from any point in time
    - Timepoint was like the “commit hash” you wanted to retrieve at

```
getMetadata.py --inDsTxt=my.datasets.txt --outFile=metadata.txt
```

- Intent was each analysis team would make+maintain+keep their cross-section file with their analysis code
  - CLI had features to *diff* two timestamps!

```
dataset_number/I:crossSection/D:kFactor/D:genFiltEff/D
303561  6.7901E+01      1.0     1.3175E-01
301016  8.9764E-10      1.0     1.0000E+00
361442  4.36e-09        0.99132 0.359953
361348  21.386  0.9083  0.891411
343432  9.5158E-08      1.0     1.0000E+00
361068  0.014022        0.91    1.000000
301137  1.4279E-10      1.0     1.0000E+00
301565  2.9636E-05      1.0     1.0000E+00
301120  2.2198E-02      1.0     1.0000E+00
303539  4.0325E-13      1.0     9.9088E-01
342561  4.4614E-04      1.0     1.0000E+00
302777  9.2630E-04      1.0     1.0000E+00
303452  2.4178E-09      1.0     1.0000E+00
361105  8.2826  1.035786        1.000000
341783  2.6920E-03      1.0     8.5170E-01
```

A Case Study: MC Dataset Cross-sections

What actually happened

- A daily cronjob queries AMI and produces a mega text file, which is copied to the overwritable section of the calibration file area
  - Cross-sections may change under analyzers feet if they stick with the 'latest' file rather than a particular date
- S/W tools available that parse that text file in analysis jobs
- Twiki pages are also automatically generated from that text file extraction
  - People still find it easier to search a twiki than query a database!
    - Would guess partly down to issue of needing grid certificate to query AMI
- Some metadata e.g. subprocess xsecs still managed by hand

Successes

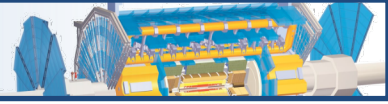
✓ Consolidated vast majority of cross-sections and associated simple dataset metadata in a single place

✓ Provided bookkeeping features for tracking changes to the metadata

Failures

✗ Getting people to query the database directly, and manage their own local caches

Thoughts for future systems …



An easily accessible database of metadata

- Clear/understandable versioning
  - Probably needs concept of tags, tagsets, tagset sets, …
  - Branches: parallel streams of different values of the same metadata
    - Possible to avoid the need for this with some good coordination?
- Concept of a local partial cache for offline work
- A fantastic API for users and developers alike
  - Developers and users loved file-based metadata because they can browse directories of files etc; ability to use familiar tools gives feeling of understanding
  - People will annoyingly fall back on what they know (files + a crude text-based index for association) if its not easy to use
- Must be easy for developers to add metadata to, including development (scratch) metadata
  - Utilise the local partial cache for this

- Is this git + a git filesystem + a database for data <-> metadata association?