Advancements in the in-file metadata system for the ATLAS experiment

Peter Van Gemmeren (ANL), Attila Krasznahorkay (UMass), Alaettin Serhan Mete (ANL), Marcin Nowak (BNL), Maciej Szymański (ANL)
On behalf of the ATLAS Software & Computing Activity

U.S. DEPARTMENT OF ENERGY Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

CHEP 2024, Kraków, Poland, 19–25 Oct 2024

Argonne NATIONAL LABORATORY

Introduction

- The in-file **metadata** system is an **essential ingredient** of HEP software
  - Reliable description of events in files and files themselves are prerequisite for all workflows
- The ATLAS in-file metadata infrastructure has been evolving over the years in response to EDM and analysis developments and needs
  - Most notably enabling MT processing [Berghaus et al. CHEP 2021]
- In the HL-LHC, the number of events will increase by an order of magnitude, demanding more **concurrent processing**
- Unlike event processing, **metadata handling is not trivially parallel**
  - Due to the challenging nature of summarising metadata payload
- Thus, ATLAS is **enhancing its in-file metadata** system to support metadata creation and propagation in more robust ways

Metadata content

Software

Physics

Category	Content
EventStreamInfo	summary of the event content for production
EventFormat	summary of the event content for analysis
FileMetaData	event and provenance summary
ByteStream	run parameters
Interval of Validity	information with a lifetime other than event or file
BookKeeping	event selections, cuts
LumiBlocks	luminosity blocks stored in file
TriggerMenu	trigger configuration
Truth	MC weights, generator details

```
├── /Digitization/Parameters:
│   ├── DigitizedDetectors: ['pixel', 'SCT', …]
│   ├── IOVDbGlobalTag    : OFLCOND-MC21-SDR-RUN3-07
│       ...
├── /Generation/Parameters:
│   ├── HepMCWeightNames  : { 'AUX_bare_not_for_analyses' : 193 , 'Default' : 0 , ...}
├── /Simulation/Parameters:
│   ├── ApplyPRR          : True
│   ├── BeamPipeSimMode   : FastSim
│       ...
├── EventFormatStreamDAOD_PHYS:
│   ├── AntiKt10LCTopoJets: DataVector<xAOD::Jet_v1>
│   ├── AntiKt10LCTopoJetsAux.: xAOD::JetAuxContainer_v1
│       ...
├── FileMetaData      :
│   ├── amiTag            : e8453_s3873_r13829_r13831
│   ├── beamEnergy        : 6800000.0
│   ├── beamType          : collisions
│   ├── conditionsTag     : OFLCOND-MC21-SDR-RUN3-07
│   ├── geometryVersion   : ATLAS-R3S-2021-03-00-00
│   ├── isDataOverlay     : False
│   ├── mcCampaign        : mc21a
│   ├── productionRelease : Athena-24.0.22
│   ├── runNumbers        : [410000]
│   └── simFlavour        : FullG4_QS
├── StreamDAOD_PHYS   :
│   ├── eventTypes        : ['IS_SIMULATION', 'IS_ATLAS', 'IS_PHYSICS']
│   ├── itemList          :
│   │   ├── ('xAOD::TrigMissingETContainer', 'HLTNav_RepackedFeatures_MET')
│   │   ├── ('xAOD::BunchConfKey', 'BunchConfKey')
│   │       ...
│   ├── lumiBlockNumbers  :
│   │   ├── 1
│   │       ...
│   ├── numberOfEvents    : 401
│   ├── processingTags    : ['StreamDAOD_PHYS']
├── TruthMetaData     :
│   ├── evgenTune         : A14 NNPDF23LO
│   ├── generators        : Powheg+Pythia8(v.307)+EvtGen(v.2.1.1)
│   ├── mcChannelNumber   : 601229
│   └── weightNames       :
│       ├── MUR1_MUF2_PDF260000
│       ├── MUR1_MUF0.5_PDF260000
│           ...
├── auto_flush        : 80
├── file_comp_alg     : 5
├── file_comp_level   : 5
├── file_guid         : 5DB82173-BEB7-C24D-88C2-171E29F814E7
├── file_size         : 20555186
├── file_type         : POOL
```

Use cases

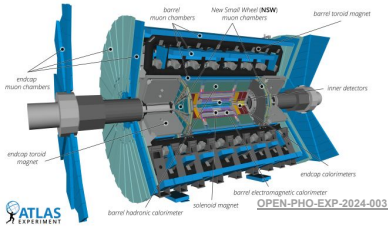
- **Configuration** of the job using info from the input files
- **Initialisation** of software components
- **Mapping** of names to data objects or values
- **Decoding** trigger information
- **Keeping track** of event selection and luminosity blocks
- **Annotations** added by users

Crucial for all workflows!

- including (but not limited to): reco, simulation, derivation, analysis

High-level overview of the architecture

- Athena-based infrastructure for metadata processing and I/O
  - `MetaDataSvc` orchestrates metadata tools through file incidents
  - *Domain–specific* tools create or propagate metadata on `beginInputFile` or after each event
  - (Transient) object stores make metadata available to clients
  - Metadata objects persistified in containers stored in database (abstractions of `Athena` I/O)
- We support metadata processing in a variety of running modes
  - Serial processing, multi-{threading, processing}, also with shared I/O [Mete et al. ICHEP2022]



creation of `TruthMetaData`

creation of `ByteStreamMetaData`

creation **and** propagation of `FileMetaData,EventFormat`

outside of production system

Generation → Simulation → Digitization → Reconstruction → Derivation → Analysis

propagation (in general **updating**, e.g. lumiblocks metadata depends on the knowledge if the input files were fully processed)

Restructuring metadata configuration

- Metadata configuration is quite *complex*
  - Various metadata categories are created or propagated at different stages
  - We use a variety of `Gaudi` components (tools, algorithms) to obtain the needed information
  - Handling depends on the mode (multi-threading, multi-processing w/ or w/o shared writer)
- Legacy configuration was difficult to maintain
  - `Athena` job configuration has recently been upgraded [Lampl, CHEP 2018]
- Hence, we have improved the metadata configuration
  - Making it modular and explicit which allows for better maintainability and flexibility
- Enhanced testing and validation enabled us to improve robustness of the infrastructure ensuring healthy metadata content
  - Difficult to quantify, but a number of tickets has seemingly dropped 😉

Metadata storage technology

- Revisiting the way we persistify the in-file metadata
  - In the context of migration of the event data to `RNTuple` [Mete et al. ACAT 2024]
- The goal is to find a solution to store metadata objects for HL-LHC
  - *Appropriate* and *performant*
- We may want to **rethink the approach**
  - Find the equivalent *standard* of storing event data in *trees*
  - Considering concept of a *file* granularity in the context of potential usage of object stores
- There is **no standard** way of storing metadata in HEP experiments
  - Examples of approaches: `TMap`, `TObject`, `sqlite` database, `JSON` string
- Arguably, there exists a **synergy across experiments** on this topic

Current state of ATLAS metadata I/O layer

- In-file metadata is stored in a dedicated `TTree` within a **single entry**
  - I/O infrastructure mostly shared with event data
  - Works reasonably well, but it's an imperfect fit (not designed for)
- Actual payload consists of:
  - Simple types (`POD` and `std::string`)
  - Containers (`std::vector`) of simple types
  - Nested vectors, `std:set`, `std::map`, `std::pair`
- Most of the above information is wrapped into `xAOD` classes [ATLAS S&C Run3 paper]
- **Size** & **I/O** speed are **not a concern**, being much smaller than event data
  - E.g. 100s of **kilobyte** within 100s of **megabyte** DAOD file (`TriggerMenu` is the largest contributor)

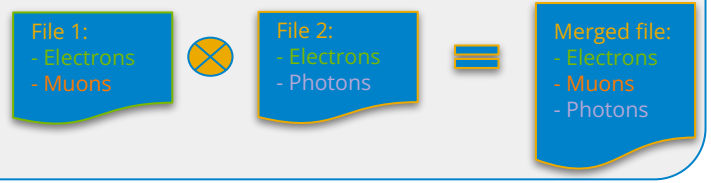
```
root [2] MetaData->Print()
******************************************************************************
*Tree    :MetaData  : MetaData                                                *
*Entries :        1 : Total =         4484849 bytes  File  Size =     168158 *
*        :          : Tree compression factor =  26.89                       *
******************************************************************************
*Br    0 :FileMetaDataAux. : xAOD::FileMetaDataAuxInfo_v1                    *
*Entries :        1 : Total  Size=        757 bytes  File Size  =        224 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :TriggerMenuJson_BGAux. : xAOD::TriggerMenuJsonAuxContainer_v1       *
*Entries :        1 : Total  Size=      26662 bytes  File Size  =       1558 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  16.76     *
*.............................................................................*
*Br    2 :TriggerMenuJson_HLTAux. : xAOD::TriggerMenuJsonAuxContainer_v1      *
*Entries :        1 : Total  Size=    2885794 bytes  File Size  =      81155 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  35.55     *
*.............................................................................*
*Br    3 :TriggerMenuJson_HLTMonitoringAux. : xAOD:                           *
*         | :TriggerMenuJsonAuxContainer_v1                                  *
*Entries :        1 : Total  Size=        807 bytes  File Size  =        206 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    4 :TriggerMenuJson_HLTPSAux. : xAOD::TriggerMenuJsonAuxContainer_v1    *
*Entries :        1 : Total  Size=     334396 bytes  File Size  =      29229 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  11.42     *
*.............................................................................*
*Br    5 :TriggerMenuJson_L1Aux. : xAOD::TriggerMenuJsonAuxContainer_v1       *
*Entries :        1 : Total  Size=    1092521 bytes  File Size  =      34198 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  31.93     *
*.............................................................................*
*Br    6 :TriggerMenuJson_L1PSAux. : xAOD::TriggerMenuJsonAuxContainer_v1     *
*Entries :        1 : Total  Size=      67979 bytes  File Size  =       3652 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  18.46     *
*.............................................................................*
*Br    7 :CutBookkeepersAux. : xAOD::CutBookkeeperAuxContainer_v1             *
*Entries :        1 : Total  Size=       1017 bytes  File Size  =        358 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.33     *
*.............................................................................*
*Br    8 :IncompleteCutBookkeepersAux. : xAOD::CutBookkeeperAuxContainer_v1   *
*Entries :        1 : Total  Size=       2008 bytes  File Size  =        612 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   2.33     *
*.............................................................................*
*Br    9 :StreamAOD : EventStreamInfo_p3                                      *
*Entries :        1 : Total  Size=      19569 bytes  File Size  =       4953 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   3.85     *
*.............................................................................*
```

Metadata merging (summarising)

- While merging events is a trivial concatenation, for metadata it is **not as straightforward**:
  - Depends on the metadata type, e.g. some values are assumed to be constant throughout the job
  - Must be handled differently knowing if the input file was fully processed, e.g. for lumi blocks
- Robust merging procedure is required not only to combine the files, but also to **support concurrent** workflows
- In principle, we could use popular ROOT's hadd utility to merge the files
  - Teaching metadata objects to know how to **merge themselves** by implementing `Merge()`
  - Detecting if the inputs are compatible and deciding how to deal with the given metadata types (and their constituents)
  - Unfortunately, this works only for the objects of classes inheriting from `TObject` 😕

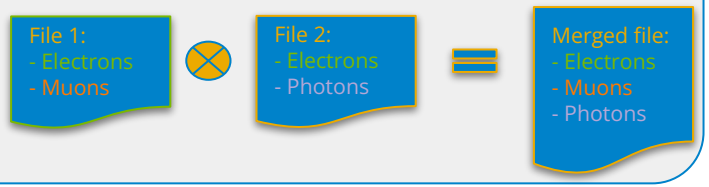
Merging scenarios

Unique accumulation: e.g. `EventFormat` (summary of event content), `TriggerMenu` configurations

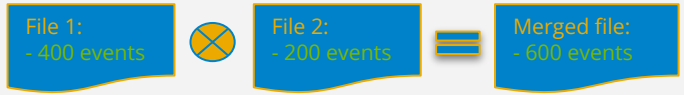


Merging scenarios

Unique accumulation: e.g. `EventFormat` (summary of event content), `TriggerMenu` configurations

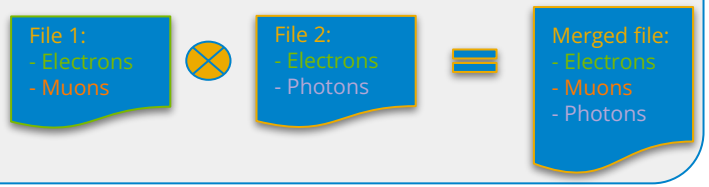


Natural addition: number of events in `EventStreamInfo`

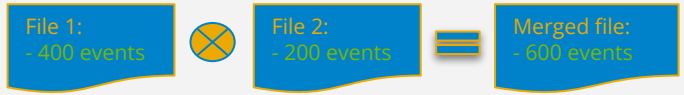


Merging scenarios

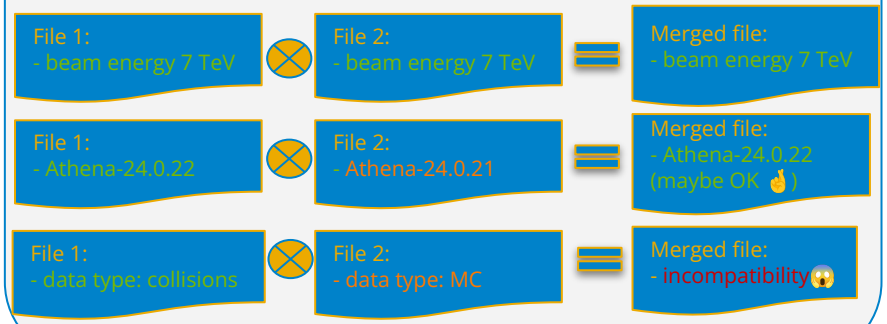
Unique accumulation: e.g. `EventFormat` (summary of event content), `TriggerMenu` configurations



Natural addition: number of events in `EventStreamInfo`

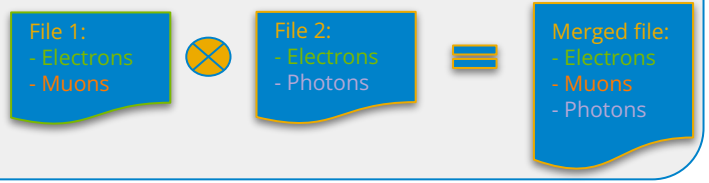


Use *first spotted value*: e.g. POD fields of `FileMetaData`

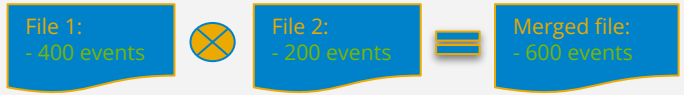


Merging scenarios

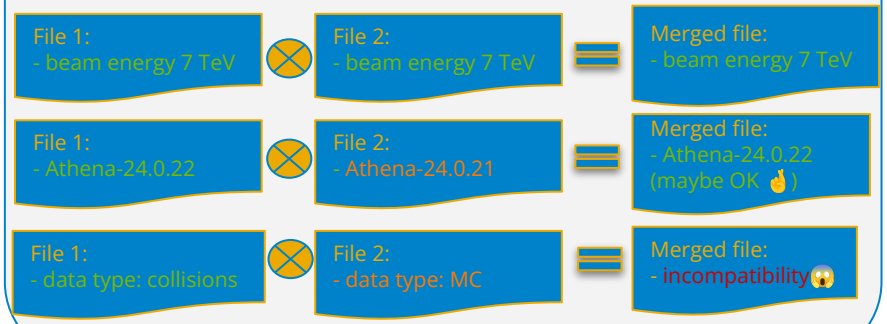
Unique accumulation: e.g. `EventFormat` (summary of event content), `TriggerMenu` configurations



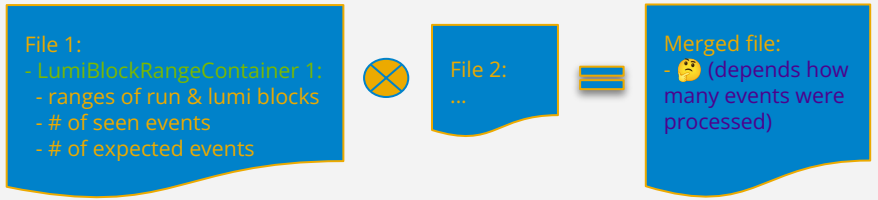
Natural addition: number of events in `EventStreamInfo`



Use *first spotted value*: e.g. POD fields of `FileMetaData`



Merged content *dependent on file processing* (`LumiBlocks`)



More on metadata processing

- As of now, merging is the responsibility of the metadata tools in `Athena`
  - Challenging *feature* due to type-dependent handling, especially for shared I/O mode
  - Metadata *linked* to the event tree has to be readable even w/o events (workers may not process any events, also the case of skimmed data)
  - We explored a single-tool with 2 modes (creation and propagation) design, but found not much gain wrt the current approach of 2 tools per category (simplicity vs flexibility trade-off)
- We investigate how *some* metadata propagation can be done **outside of event processing**
  - Make it possible to transmit some payloads (e.g. beam energy) w/o the need for an event loop
- We are also constantly improving `FileMetaData` (key-value store summarising properties of all events in the file)
  - Used primarily for job configuration and analysis, so we focus on ease of use outside `Athena` and convenient file peeking

Possible solutions for in-file metadata storage

- Store metadata in a separate `RNTuple`
  - Implemented, following the work for event data, also adapting auxiliary (Python) tools
  - However, `RNTuple` is not really designed for a single-entry *tree*
  - While not a crucial factor, slight size increase wrt `TTree` (less opportunities for compression?)
- Store metadata objects using `ROOT`-keyed container (based on `TKey`)
  - (Partially) implemented resurrecting an old implementation dating back to `LCG POOL`
  - Feels suboptimal to use an old, low-level API
  - Also, does not really help with solving the issue of metadata merging
- Leverage the future feature of user-provided metadata in `RNTuple`
  - Listed in the RNTuple architecture document
  - In essence, it would be beneficial to allow for a second *tree* associated with event tree
  - Favorable solution, looking forward to collaborating with `ROOT` team and other experiments!

Summary

- We presented the on-going investigations into **enhancing the ATLAS in-file metadata** system for HL-LHC
  - Integral part of the ATLAS software ecosystem crucial for successful physics programme
- Investigating **improvements** of the in-file **metadata storage**
  - Promising development of the user-defined metadata in `RNTuple`
- Looking forward to community collaboration on developing common strategies for coping with challenges of metadata processing
- Working towards more **robust summarising of metadata** objects
  - Challenging in face of the need for concurrent processing towards higher data rates

Thank you for your attention!

mszymanski@anl.gov