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

Peter van Gemmeren[1], *Attila* Krasznahorkay[2], *Alaettin Serhan* Mete[1], *Marcin* Nowak[3], and *Maciej* Szymański[1]*
On Behalf of the ATLAS Computing Activity†

[1]Argonne National Laboratory, Lemont, IL, United States**
[2]University of Massachusetts, Amherst MA, United States
[3]Brookhaven National Laboratory, Upton, NY, United States

Abstract. The High-Luminosity upgrade of the Large Hadron Collider (HL-LHC) will increase luminosity and the number of events by an order of magnitude, demanding more concurrent data processing. Event processing is trivially parallel, but metadata handling is more complex and breaks that parallelism. However, correct and reliable in-file metadata is crucial for all workflows of the experiment, enabling tasks such as job configuration, decoding trigger information, and keeping track of event selection. Therefore, ATLAS is enhancing its current in-file metadata system to support metadata creation and propagation in more robust ways. This work presents developments in the evolution of the metadata system. We investigate storage technologies tailored for in-file metadata payload, exploring advancements in the ROOT framework, which is used for storing data collected by the ATLAS experiment. We also discuss the challenging process of summarising the content of metadata objects when combining information from several sources.

1 Introduction

The in-file metadata system is a cornerstone of High Energy Physics (HEP) software, playing a critical role in ensuring the reliable description of events within files and the files themselves. This reliability is a prerequisite for all workflows within the HEP domain. Over the years, the in-file metadata infrastructure of the ATLAS [1] experiment at CERN's Large Hadron Collider has undergone significant evolution, adapting to the developments of Event Data Models (EDM) and needs of physics analyses. A notable advancement in this area has been the adaptation to multi-threaded (MT) processing [2].

As we approach the era of the High-Luminosity Large Hadron Collider (HL-LHC), the number of events is expected to increase by an order of magnitude, necessitating more concurrent data processing capabilities. Unlike event processing, metadata handling presents unique challenges, particularly in the merging procedure of metadata payloads. In response

*e-mail: mszymanski@anl.gov, speaker
**Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357
†Copyright 2025 CERN for the benefit of the ATLAS Collaboration. Reproduction of this article or parts of it is allowed as specified in the CC-BY-4.0 license.

to these challenges, ATLAS is actively enhancing its in-file metadata system to support more robust methods of metadata creation and propagation, ensuring the infrastructure can meet the demands of future HEP workflows.

2 In-file metadata in ATLAS

The ATLAS experiment employs a comprehensive metadata system that categorizes information into distinct domains. These domains can be broadly classified into event-specific and file-specific metadata, as detailed in Table 1.

Table 1. ATLAS Metadata Domains

Domain	Description
`EventStreamInfo`	Event sample description, used for production
`EventFormat`	Summary of event layout, used for analysis
`FileMetaData`	Event and provenance summary
`ByteStream`	Run parameters
`Interval of Validity`	Information with lifetime other than event or file
`BookKeeping`	Event selections, cuts
`LumiBlock`	Luminosity blocks stored in file
`TriggerMenu`	Trigger configuration
`Truth`	MC weights, generator details

The in-file metadata system plays a vital role in multiple aspects of ATLAS operations. It enables job configuration using input file information, facilitates software component initialisation, and enables the instantiation of `C++` objects using names stored in the file. Additionally, it handles trigger information decoding, monitors event selection and the integrated luminosity of the analysed data, and allows for user-specific annotations. These capabilities make the system fundamental to all ATLAS workflows, including reconstruction, simulation, derivation, and analysis. The example of the on-disk metadata content in a typical Monte Carlo file used as an input for physics analysis is shown in Listing 1.

The metadata processing and I/O infrastructure is deeply integrated within `Athena` [3], the software framework of the ATLAS experiment based on `C++` and `Python`. One of the main challenges of the in-file metadata design was to provide a practical way of caching the metadata in memory in a data store, which in principle is independent from the file [4]. At its core, the service component called `MetaDataSvc` orchestrates metadata propagation tools through file incidents (opening and closing the input files). Metadata describing the event sample is typically created via tools operated by `AthenaOutputStream` algorithm(s) (one for each output file), after each processed event. Such a design requires generally two tools per metadata domain that need to operate together. We evaluated a simplified design using a single tool with dual modes (creation and propagation), but found that the current approach of using two separate tools per category offers better flexibility with minimal additional complexity.

In-memory metadata employs two distinct stores. The input store is dynamically populated with content from each new input file and cleared when moving to the next one. In parallel, an output store accumulates new content by appending to output metadata containers throughout the workflow runtime. At the job's conclusion, metadata objects are written to the output stream(s) using tools implementing abstract interfaces. In practice, these abstractions are implemented using `TTrees` within `ROOT` [5] files, with metadata stored in a dedicated `TTree` containing a single entry and sharing much of its I/O infrastructure with

```
/Digitization/Parameters:
   DigitizedDetectors: ['pixel', 'SCT', (...)]
   IOVDbGlobalTag: OFLCOND-MC21-SDR-RUN3-07
   (...)
/Generation/Parameters:
   HepMCWeightNames: { 'AUX_bare_not_for_analyses' : 193 , (...)}
/Simulation/Parameters:
   ApplyPRR: True
   BeamPipeSimMode: FastSim
   (...)
EventFormatStreamDAOD_PHYS:
   AntiKt10LCTopoJets: DataVector<xAOD::Jet_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')
      (...)
   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
      (...)
auto_flush: 80
file_comp_alg: 5
file_comp_level: 5
file_guid: 5DB82173-BEB7-C24D-88C2-171E29F814E7
file_size: 20555186
file_type: POOL
```

Listing 1: The example of the on-disk metadata content in a typical Monte Carlo file used as an input for physics analysis. Note that ellipsis marks (...) are used for the sake of brevity.

event data. While this approach works reasonably well, it was not specifically designed for the metadata use case, making it somewhat of an imperfect fit since `TTree` was optimised for handling large datasets with multiple entries, whereas metadata, by its nature, consists of a single entry containing information about the event sample and its wrapper file.

The transient representation of metadata attributes across all domains includes simple types (Plain Old Data and `std::string`), containers (`std::vector`) of simple types, as well as nested vectors, `std::set`, `std::map`, and `std::pair`, with most of these data structures encapsulated within `xAOD` classes [6]. Despite the rich structure of metadata objects, their size and I/O performance in production workflows are not significant concerns, as they represent a relatively small fraction of the total file size - typically less than 1% of a Derived Analysis Object Data (`DAOD`) [6] file, which is the main data format for physics analysis.

To take full advantage of all available computing resources, `Athena` supports different concurrency modes. This includes multi-threading as well as multi-processing utilising shared I/O to avoid having to merge output data files [7]. As each event's data and processing is independent of each other, they are trivially parallelisable, and thus data can simply be appended. For metadata, however, the merge step requires domain-specific implementation to summarise metadata from different workers. Moreover, different metadata categories require specific handling at various stages of processing. For example, `ByteStream` metadata is created during reconstruction and subsequently propagated, while others may need to be updated at multiple steps of processing. This complexity necessitates the use of various `Gaudi` [8] components and requires sophisticated configuration approaches to support different running modes.

Recent improvements in `Athena` job configuration [9] have significantly enhanced the metadata system's configuration which has become more modular and explicit, leading to better maintainability and flexibility. Enhanced testing and validation procedures have been implemented, resulting in improved infrastructure robustness and ensuring healthy metadata content across all ATLAS workflows. These advancements represent a significant step forward in the system's evolution, providing a more reliable and efficient framework for metadata management in ATLAS operations.

3 Summarising metadata content

Merging metadata presents unique challenges compared to the straightforward concatenation of event data. In fact, it depends on the semantics of particular attributes. The merging process must account for different metadata types, where some values are expected to remain constant throughout the job execution, whereas others are mutable and can be updated according to various rules. Additionally, the merging strategy needs to consider how many events from the input files were processed, particularly for metadata categories such as luminosity information.

A robust merging procedure is essential not only for combining files but also for supporting concurrent workflows, in particular when summarising the information collected by the workers in the multi-processing mode. While `ROOT`'s `hadd` utility might seem like a natural solution for merging files, implementing it for metadata would require teaching metadata objects to merge themselves, detecting input compatibility, and determining appropriate handling strategies for different metadata types. However, this approach is limited to classes inheriting from `TObject`, making it unsuitable for our metadata system.

The metadata objects in ATLAS follow several distinct merging scenarios. In the case of unique accumulation, applicable for example to `EventFormat` and `TriggerMenu`, new values are appended to existing ones with deduplication. For example, merging collections containing `[electrons,muons]` and `[electrons,photons]` results in [electrons,muons,photons]. Natural addition applies to cases like event counts in EventStreamInfo, where values from the inputs are simply summed.

For certain metadata fields, particularly in FileMetaData, the system adopts a first-value priority approach. When values are identical (such as beam energy across files), no special handling is needed. However, when differences occur, the system uses the first encountered value. This may be acceptable in some cases (like different software release versions) but potentially problematic in others (such as mixing collision and MC data), necessitating appropriate warning or error mechanisms.

The most complex scenario involves processing-dependent metadata, exemplified by LumiBlock metadata. To obtain the meaningful luminosity information, one needs to keep track whether LumiBlocks are complete (all events processed), incomplete (partial processing), or suspect (more events processed than expected from the information taken from an external database).

Currently, metadata merging is handled by dedicated tools within the Athena framework. This approach presents challenges due to category-dependent handling requirements, particularly in shared I/O mode. Another critical design consideration is that metadata describing the event sample must remain readable even without events present, which is essential for workflows such as derivation production where the output event collections may be empty.

4 Storage technology

RNTuple is ROOT's future I/O system for HEP data [10]. As the successor to TTree, it offers a more performant, modern, and robust solution for data storage. As ATLAS transitions its event data storage to RNTuple [11], we are taking the opportunity to revisit the way we persistify the in-file metadata. Our goal is to establish a robust storage solution suitable for the HL-LHC era that is both appropriate and performant, while facilitating robust merging capabilities. This re-evaluation prompts us to reconsider our current approach, seeking an equivalent standard to event data tree storage and examining the concept of file granularity in the context of potential object store usage.

It is worth noting that there is no standardised approach to metadata storage across High Energy Physics experiments. The diversity of solutions used by experiments suggests potential benefits from cross-experiment collaboration and standardisation efforts.

In our investigation of alternative storage solutions, we have developed two prototypes. The first implements metadata storage in RNTuple, following our work with event data and including adaptations of auxiliary Python tools (used for e.g. file peeking and validation). Such an approach is a natural evolution of the currently employed solution which is our basic scenario for the future metadata storage technology. One needs to emphasise that since TTree will enter *legacy* phase, we need to take some action concerning in-file metadata storage anyway. However, while we have not found any showstoppers of persistifying metadata in RNTuple, there are also no significant advantages over the current solution based on TTrees. RNTuple was also not specifically designed for single-entry data samples. While we observed a tiny increase in on-disk size compared to TTree, this is entirely negligible given the small size of metadata compared to event data.

The second prototype explores metadata storage in ROOT-keyed containers based on TKey, reviving an implementation that dates back to the LCG POOL project [12]. However, this approach has proven suboptimal, relying on an old, low-level API and failing to address the fundamental challenge of metadata merging.

Looking forward, our preferred direction aligns with the planned user-provided metadata feature in RNTuple, as outlined in the RNTuple architecture document [13]. This upcoming capability appears to offer the most promising solution for our metadata storage needs, potentially providing both the flexibility and performance required for the HL-LHC era.

5 Conclusions

This paper presented our ongoing work to enhance the ATLAS in-file metadata system in preparation for the HL-LHC era. The metadata system represents an integral component of the ATLAS software ecosystem and is crucial for supporting a successful physics programme. Our investigations have focused on multiple areas of improvement, particularly regarding the in-file metadata storage mechanisms and their optimisation.

We are particularly interested in the development of user-defined metadata capabilities within the `RNTuple`. This advancement would open new possibilities for more flexible and efficient metadata handling. As we continue to develop these systems, we recognize the importance of collaboration within the broader high-energy physics community to establish common strategies for addressing the growing challenges of metadata processing.

Our efforts are directed towards creating more robust methods for summarising metadata objects, a task that has become increasingly challenging given the requirements for concurrent processing to achieve higher data rates. This work represents an important step forward in preparing the ATLAS metadata infrastructure for the demands of future experimental conditions at the HL-LHC.

References

[1] G. Aad et al. (ATLAS), The ATLAS Experiment at the CERN Large Hadron Collider, JINST **3**, S08003 (2008). 10.1088/1748-0221/3/08/S08003

[2] Berghaus, Frank, Krasznahorkay, Attila, Martin, Tim, Novak, Tadej, Nowak, Marcin, Schaffer, A.C., Tsulaia, Vakho, van Gemmeren, Peter, ATLAS in-file metadata and multi-threaded processing, EPJ Web Conf. **251**, 03006 (2021). 10.1051/epjconf/202125103006

[3] ATLAS Collaboration, Athena (2021), `https://doi.org/10.5281/zenodo.4772550`

[4] D. Malon, P. Van Gemmeren, R. Hawkings, A. Schaffer, An inconvenient truth: file-level metadata and in-file metadata caching in the (file-agnostic) ATLAS event store, J. Phys.: Conf. Ser. **119**, 042022 (2008). 10.1088/1742-6596/119/4/042022

[5] R. Brun, F. Rademakers, P. Canal, A. Naumann, O. Couet, L. Moneta, V. Vassilev, S. Linev, D. Piparo, G. GANIS et al., root-project/root: v6.18/02 (2020), `https://doi.org/10.5281/zenodo.3895860`

[6] G. Aad et al. (ATLAS), Software and computing for Run 3 of the ATLAS experiment at the LHC (2024), `2404.06335`.

[7] A. Serhan Mete, P. van Gemmeren (ATLAS), Shared I/O Developments for Run 3 in the ATLAS Experiment, PoS **ICHEP2022**, 219 (2022). 10.22323/1.414.0219

[8] LHCb Collaboration and ATLAS Collaboration, Gaudi (2024), `https://doi.org/10.5281/zenodo.14018447`

[9] W. Lampl, A new approach for ATLAS Athena job configuration, EPJ Web Conf. **214**, 05015 (2019). 10.1051/epjconf/201921405015

[10] J. Blomer, P. Canal, F. de Geus, J. Hahnfeld, A. Naumann, J. Lopez-Gomez, G.L. Miotto, V.E. Padulano, ROOT's RNTuple I/O Subsystem: The Path to Production, EPJ Web Conf. **295**, 06020 (2024). 10.1051/epjconf/202429506020

[11] A.S. Mete, M. Nowak, P. Van Gemmeren (ATLAS), Tech. rep., CERN, Geneva (2024), https://cds.cern.ch/record/2905189

[12] D. Duellmann, The LCG POOL project: General overview and project structure, eConf **C0303241**, MOKT007 (2003), physics/0306129.

[13] ROOT Project, RNTuple Architecture (2024), accessed: 11.12.2024, https://github.com/root-project/root/blob/v6-34-00-patches/tree/ntuple/v7/doc/Architecture.md#future-features