Framework for custom event sample augmentations for ATLAS analysis data

Peter van Gemmeren[1,*], *Alaettin Serhan* Mete[1], *Jackson Carl* Burzynski[2], *James* Catmore[3], *Lukas* Heinrich[4], *Marcin* Nowak[5], and *Nils* Krumnack[6]

On Behalf of the ATLAS Computing Activity[**]

[1]Argonne National Laboratory (US)[***]
[2]Simon Fraser University (CA)
[3]University of Oslo (NO)
[4]Max-Planck-Institut für Physik (DE)
[5]Brookhaven National Laboratory (US)
[6]Iowa State University (US)

Abstract. For HEP event processing, data is typically stored in column-wise synchronized containers, such as most prominently ROOT's TTree, which have been used for several decades to store by now over 1 exabyte. These containers can combine row-wise association capabilities needed by most HEP event processing frameworks (e.g. Athena for ATLAS) with column-wise storage, which typically results in better compression and more efficient support for many analysis use-cases. One disadvantage is that these containers, TTree in the HEP use-case, require to contain the same attributes for each entry/row (representing events), which can make extending the list of attributes very costly in storage, even if those are only required for a small subsample of events. Since the initial design, the ATLAS software framework features powerful navigational infrastructure to allow storing custom data extensions for subsamples of events in separate, but synchronized containers. This allows adding event augmentations to ATLAS standard data products (such as DAOD-PHYS or PHYSLITE) avoiding duplication of those core data products, while limiting their size increase. For this functionality, the framework does not rely on any associations made by the I/O technology (i.e. ROOT), however it supports TTree friends and builds the associated index to allow for analysis outside of the ATLAS framework. A prototype based on the Long-Lived Particle search is implemented and preliminary results with this prototype will be presented. At this point, augmented data are stored within the same file as the core data. Storing them in separate files will be investigated in future, as this could provide more flexibility, e.g. certain sites may only want a subset of several augmentations or augmentations can be archived to tape once their analysis is complete.

[*]e-mail: gemmeren@anl.gov

[***]Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357

1 Introduction

The ATLAS experiment [1] is a multi-purpose detector collecting data from the LHC at CERN. After the second Long Shutdown (LS2) to upgrade the LHC infrastructure and the detectors, Run 3 has started and ATLAS is taking data. Run 3 is expected to produce more than twice the collision data recorded by Run 1 and Run 2 combined, providing new opportunities for physics analyses while at the same time straining computing resources.

2 Run 3 Analysis Model, Data and Core I/O

The processing of ATLAS detector data and simulation is a multi-step procedure: Information detected by the different ATLAS sub-detectors for LHC collision events and simulation is reconstructed and stored in ROOT files [2]. A data product containing every event and all analysis objects is called Analysis Object Data (AOD) or primary AOD. The average size for the AOD is about 400 kB per event.

2.1 Analysis Data Model for Run 2 and Run 3

For Run 2, to streamline analysis, the AOD files were processed by the derivation framework [3] to produce about 100 different derived AOD (DAOD) streams that contain a subset of events and reduced reconstruction information customized for specific physics analyses. Figure 1 shows the Run 2 analysis model that has been very successful for analysis productivity, but expensive in terms of resource usage. Even as most DAOD streams reduce the event size significantly and write only a small fraction of the events, the large number of custom formats leads to data duplication due to non-negligible overlap between the streams and causes DAOD storage requirements similar to those of the primary AOD.

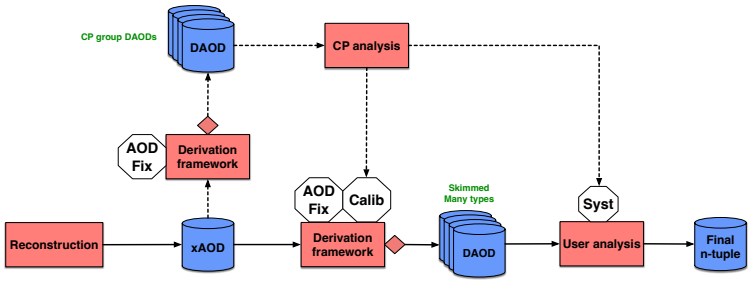

Figure 1. Run 2: Many [skimmed] DAOD streams, lots of data duplication

In Run 3, to limit the number of DAOD stream and significantly reduce the associated storage needs, ATLAS introduced a new common DAOD-PHYS format targeted for most physics analyses. To reduce needs for more expensive disk storage, some of the primary AOD will be stored on tape [4]. Figure 2 shows the Run 3 analysis model.

Introducing a new common analysis format is a big change, motivated by associated storage savings that will become mandatory for High Luminosity LHC. A common analysis format, that is written for every event requires a balanced consideration of what data attributes/columns are included. As these columns are written for every event, even if they are needed for a fraction of events only, they could potentially have a big cost in storage. However, if those attributes are not included in PHYS/PHYSLITE, some analyses cannot use that format and have to request their own custom streams, leading to data duplication.

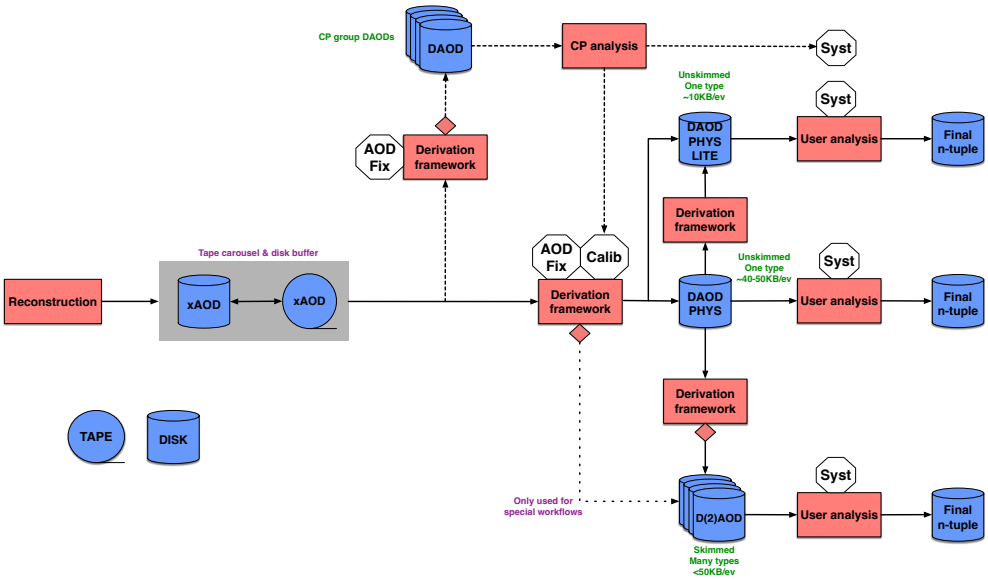

Figure 2. Run 3: Single [un-skimmed] DAOD-PHYS for most analyses

2.2 Data Model for DAOD

Similar to other processed ATLAS data, DAOD is written via ROOT, with most of the event data being stored in a single TTree. The ATLAS DAOD data model is separated into a large number of different physics attributes [5] which can be associated to higher level core objects. As events are processed row-wise by the derivation framework these objects are filled with data describing an ATLAS collision event. As the final step of processing a particular event, the data for every attribute is written into dedicated TBranches of a TTree, creating a single entry. In ROOT, many entries, configured via ATLAS software, are collected before the data for each TBranch is compressed individually and written to storage in a column-wise manner. Column-wise storage results in better compression and more efficient support for many analysis use-cases, where only a subset of attributes/TBranches are needed.

2.3 Core I/O developments in support of event augmentation

ATLAS' I/O uses navigational references that contain: Technology, DB/file id, container name and offset to access objects. In the past, the offset of objects stored in ROOT TTrees was given by their entry number, as this is simple, robust and unique. However, relying on the entry number means the references are not relocatable as they change and become invalid if objects were moved/merged into a different tree. They also are not customizable, e.g., can't be synchronized among trees. Therefore, for Run 3, ATLAS introduced a new column, with a unique 'index_ref', that is relocatable and can be synchronized between containers.

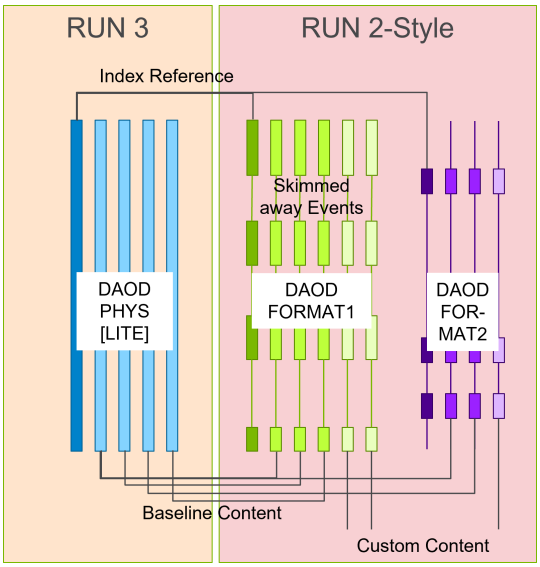

Figure 3. For Run 2, the ATLAS analysis model produced many streams of custom DAOD that contained attributes of interest for filtered events for analysis groups (as shown on the right). In Run 3, many of these custom formats are being replaced by the common DAOD-PHYS, that is written for all events (left). Many, but not necessarily all attributes from the typical Run 2 custom DAODs are also stored in DAOD-PHYS (medium shade), including a newly added index reference branch (dark shade).

3 Combining Run 3 and Run 2-style DAOD

Figure 3 shows how Run 3 and Run 2-style DAOD are written to ROOT. Custom (Run 2-style) DAOD contain attributes of interest for filtered events for analysis groups. The baseline Run 3-style common DAOD-PHYS is written for all events and contains many, but not necessarily all attributes from the typical Run 2 custom DAODs. All DAOD include the newly added index reference branch.

As shown in Figure 4, these DAOD's can be combined into a single stream, written into the same file. Attributes that are in the baseline DAOD-PHYS are written for every event and do not need to be duplicated into custom formats. Attributes that are only in a custom DAOD can be written into a separate tree, an operation called 'Event Sample Augmentation', containing the events of interest for that stream only. The index reference branch is written and used to link multiple trees (similar to primary/foreign keys in relational databases), even as they have different entry numbers.

4 Adding Event Sample Augmentation to Derivation

The command line for running Derivation to produce event sample augmentations is a simple extension (`--augmentations` option) of the standard command as shown:

```
# This command runs an example derivation production job
export ATHENA_CORE_NUMBER=8;
Derivation_tf.py \
  --CA 'True' --maxEvents '-1' \
  --multiprocess 'True' --sharedWriter 'True' \
  --inputAODFile '<file name>' --outputDAODFile 'pool.root' \
  --formats 'PHYS' 'LLP1' --augmentations 'LLP1:PHYS';
```

The details for event filtering and additional content are defined by Long Lived Particles (LLP) search, within the LLP1 Derivation kernel, similar to what would have been done for Run 2. This command will produce augmented DAOD-PHYS that can be used just like regular DAOD-PHYS and are not skimmed containing their baseline event data in single

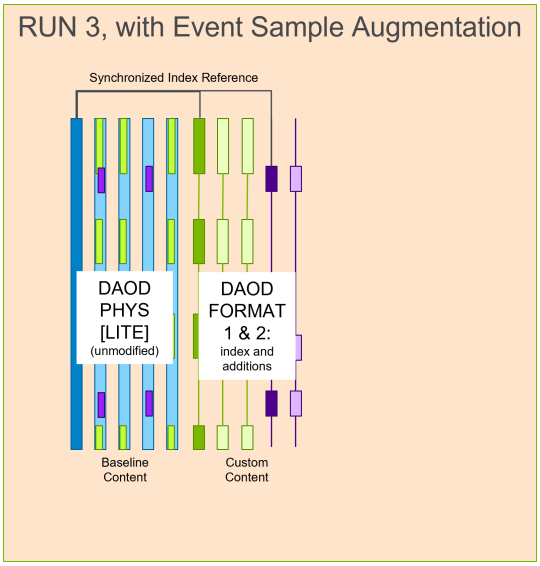

Figure 4. Combining Run 3 and Run 2-style DAOD: Write DAOD-PHYS, and different skims for DAOD-Format 1 and 2 into the same file. Columns for custom DAOD that are not in the baseline DAOD-PHYS can be augmented, by writing them into separate TTrees that only contain entries for the events of interest for that stream. The index reference branch in all trees is synchronized and allows association.

ROOT TTree. Within Athena, the augmented DAOD can also be used to read LLP1 additions for the selected event sample in a user transparent way, very similar to reading Run 2 style custom DAOD.

The LLP derivation introduces 6 new containers, in addition to the baseline DAOD-PHYS content, increasing the average event size by about 40%. The data is needed for a skim of about 40% of the events, which is a much larger fraction than was envisaged for the Run 2 analysis model.

- Old solution (Run 1): Add 40% to all (100%) data would increase the file size by 40% for everyone.
- Past solution (Run 2): Write out 140% extra for 40% skim, would cost 56% more storage due mainly increased duplication, but everyone gets their custom content.
- Event Sample Augmentation: Write out additional 40% for 40% of the events only, resulting in 16% file size increase, a much smaller increase. Since the augmentations are stored in separate trees they will not degrade I/O performance for clients reading only baseline DAOD-PHYS content.

The Event Sample Augmentation provides a solution that allows custom data for analyses, while mitigating resource impacts.

5 Summary and Outlook

ATLAS developed the infrastructure for in-file augmentation of custom event samples that combines advantages of having a single format, such as DAOD-PHYS with flexibility of (Run 2-style) physics tailored multiple-stream DAOD data products, while limiting additional resource needs. Core software support for allowing placement in separate files is in progress. Placing event sample augmentations in separate files increases flexibility of data placement substantially and completely eliminates size increase of baseline/common DAOD. However, this will require new data management/computational support to deliver augmentations when events are read. This kind of on-demand file delivery has been challenging in the past, but there are new technologies (such as object stores) which may be promising.

References

[1] ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider. J. Inst. 3, S08003 (2008) DOI: 10.1088/1748-0221/3/08/S08003

[2] R. Brun, F. Rademakers, ROOT: An Object Oriented Data Analysis Framework. AIHENP'96 Workshop Nucl. Inst. & Meth. In Phys. Res. A 389 81-86. (1996) http://root.cern.ch

[3] J. Elmsheuser, C. Anastopoulos, J. Boyd, J. Catmore, H. Gray, A. Krasznahorkay, J. McFayden, C. J. Meyer, A. Sfyrla, J. Strandberg, K. Suruliz, T. Theveneaux-Pelzer, Evolution of the ATLAS analysis model for Run-3 and prospects for HL-LHC. EPJ Web Conf. 245 06014 (2020) DOI: 10.1051/epjconf/202024506014

[4] X. Zhao et al., ATLAS Data Carousel, EPJ Web Conf. 245 04035 (2020) DOI: 10.1051/epjconf/202024504035

[5] T. Eifert, M. Elsing, D. Gillberg, K. Koeneke, A. Krasznahorkay, E. Moyse, M. Nowak, S. Snyder, P. Van Gemmeren Implementation of the ATLAS Run 2 event data model. Proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics J. Phys.: Conf. Ser. (2015) DOI: 10.1088/1742-6596/664/7/072045

[6] P. van Gemmeren, D. Malon, M. Nowak, Next-Generation Navigational Infrastructure and the ATLAS Event Store. J Phys Conf Ser 513:052036 (2014) DOI: 10.1088/1742-6596/513/5/052036

[7] M. Nowak, P. van Gemmeren, J. Cranshaw, ATLAS Event Store and I/O developments in support for Production and Analysis in Run 3. EPJ Web Conf. 245 02031 (2020) DOI: 10.1051/epjconf/202024502031