The ATLAS ROOT-based data formats: recent improvements and performance measurements

W. Bhimji¹, J. Cranshaw², P. van Gemmeren², D. Malon², R. D. Schaffer³, and I. Vukotic⁴ for the ATLAS collaboration

¹ University of Edinburgh, School of Physics & Astronomy, James Clerk Maxwell Building, The Kings Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK
² ANL, Lemont, IL 60439, USA
³ Laboratoire de l'Accelerateur Lineaire, Universite Paris-Sud 11, Batiment 200, 91898 Orsay, France
⁴ University of Chicago, 5620 S Ellis Ave, Chicago IL 60637, USA

E-mail: wbhimji@staffmail.ed.ac.uk

Abstract. We detail recent changes to ROOT-based I/O within the ATLAS experiment. The ATLAS persistent event data model continues to make considerable use of a ROOT I/O backend through POOL persistency. Also ROOT is used directly in later stages of analysis that make use of a flat-ntuple based “D3PD” data-type. For POOL/ROOT persistent data, several improvements have been made including implementation of automatic basket optimisation, memberwise streaming, and changes to split and compression levels. Optimisations have also been made for the D3PD format. We present a full evaluation of the resulting performance improvements from these, including in the case of selected retrieval of events. We also evaluate ongoing changes internal to ROOT, in the ATLAS context, for both POOL and D3PD data. We report results not only from test systems, but also utilising new automated tests on real ATLAS production resources which employ a wide range of storage technologies.

1. Introduction
The ATLAS experiment stores data in a range of formats that have been described in detail in a previous CHEP proceeding [1]. The current ATLAS data processing workflow is to process RAW data (which is not stored in ROOT files) into the ESD and AOD formats which use ROOT I/O [2] via the POOL persistency framework [3]. The ESD format is larger and not designed for end-user analysis but to allow detector studies without needing to return to the RAW data. In many cases the AOD files are then processed into D3PD files, which are specific for analysis groups and contain ROOT trees with only simple variables (or vectors of such variables) and are therefore accessible directly via ROOT. In many cases these D3PDs are analysed by a number of users, and can have file sizes similar to the AOD, so require a further processing step to reduce the data volume. Both AOD and D3PD formats are designed to be used for “user analysis” (or “analysis”) activity that is done outside the experiment-wide coordinated, “production”. It is analysis activity that is the focus of this paper. The aim of much of the work described here, is to make more effective use of Atlas shared resources on the Worldwide LHC (WLCG) computing grid. Figure 1 shows the amount of activity classified as “analysis” on these resources both in terms of wallclock time and number of jobs run. As illustrated this is now a significant volume...
of activity, and it differs from production both in being less coordinated and less optimised, but also in reading in a significant volume of data unlike the generation of simulated data which contributes the largest portion to production activities. Therefore the per-event CPU time is small and the activity can be dominated by I/O overheads. The grid resources being used by these jobs also are backed by a range of storage technologies that differ in their response to changes, so we have constructed a testing framework that evaluates any optimisations on those resources themselves: providing both monitoring and fast feedback on developments. Despite analysis activity not being centrally coordinated, there are opportunities for optimisation across the experiment: for example, the writing of D3PDs is largely done by standard tools so properties of these files such as zip level and auto-flush settings can be adjusted. In addition, for the reading of D3PDs, there are centrally provided recommended tools into which optimisations could be placed. Finally we provide guidance for users writing their own analysis code.

Figure 1. Relative share of “analysis” (red) and “production” (yellow) ATLAS grid resources in terms of number of jobs (left) and wallclock time (right) for one year to April 2012.

In Section 2 we describe recent changes to the AOD format, particularly designed for performance in the case of selected retrieval of events. We then outline, in Section 3, a testing framework built to evaluate future changes in ROOT I/O on ATLAS. In Section 4.1 we provide some results from the framework including the evaluation of changes to the D3PD format, ROOT versions and wide-area-network tests.

2. AOD Optimisations

Figure 2 illustrates some salient features of ROOT streaming. Prior to the changes described here (and in more detail in [4]), ATLAS AOD data was "fully-split" so that the primitive data members of each object were stored in their own branch. Each branch will compress its data for multiple events into a basket, which is then flushed to disk. Full splitting was done primarily for compression and read-speed motivations but it created more than 10,000 branches for AODs and ESDs which increased the number of disk reads for retrieving a single object. As the speed advantages can also be achieved with ROOT’s “member-wise streaming”, a new event data layout is now used which switches off splitting, while continuing member-wise streaming, so each collection (POOL container) is stored in a single ROOT basket. This change was made for all but the largest container in the ESD (the Track container) and the two largest containers in the AOD (the TrackParticle and CaloCluster containers). Because of their large size, these
containers showed good performance with the previous persistent layout and not splitting them would cause a significant file size increase.

![Different ROOT streaming modes](picture_from_4.png)

**Figure 2.** Different ROOT streaming modes (picture from [4]).

This change decreased the number of baskets in the ESD and AOD to about 800, and therefore increased their average size by more than a factor of 10, which lowers the number of disk reads. Since the 2011 data, ATLAS have also used the ROOT feature to “optimize baskets” so that baskets are resized to share out 30MB and have approximately the same number of entries. With basket sizes optimized for 30 MB ROOT trees, approximately 50 to 200 entries (i.e., events) share the same basket. For sequential reading of all data objects in every event, this does not pose a problem, but for selective reading of a single event, this means that 30 MB of data have to be read from disk and decompressed, even when only a small fraction is needed. This behavior causes a large performance penalty on selective event retrieval. ATLAS physicists are able to read a small fraction of events by selecting on event tags (TAG) [5] which are event-level metadata which support efficient identification and selection of events of interest to a given analysis. The new data layout described here is therefore designed to minimize the performance penalty for selective event reading, without hurting overall I/O performance. The change outlined here reduces the number of disk reads by more than an order of magnitude, while increasing their size accordingly. This allows ATLAS to store fewer events per basket. In the new persistent data layout, automatic basket optimization is used to flush baskets every 10 events for AOD. The reduction in total basket size within the main event tree and switching off splitting for all trees allow ATLAS to reset the basket sizes for the other trees from 2 KB to 32 KB, which is the ROOT default.

### 2.1. AOD Performance Results

Two scenarios were considered to test read speed performance of the new data layout. First, reading all data objects of all events in sequence, in a way that is usually done during data processing campaigns, was studied. The results of CPU time measurements can be found in Table 1. These were performed on a controlled test machine, accessing files locally with care taken to avoid caching effects. They show that the new data layout not only matches the performance of the previous format, but improves it by over 30% for AOD. Improvements of 15% were seen for the ESD format [4]. The reason for the improvement is due to two factors. Firstly, there are a smaller number of required disk reads (without TTreeCache) because the
number of branches has been reduced by a greater factor than the number of entries in each basket. There is also a better sequencing of the baskets in the file due to the reduction in basket size meaning the branches are flushed more often, reducing the spacing within the file.

The main motivation for the layout changes, however, was the use case of selective reading. Analysis users of the AOD may read a sparse selection of events or may retrieve a particular data object only for events with certain characteristics. So the second use case is studied utilizing TAGs [5] as a mechanism to randomly read 1% of all events. For the selected events, all data objects are read. The TAG infrastructure poses only a very negligible overhead for event processing. However, when reading only a subsample of the events, a large per-event penalty is caused by the fact that the ROOT baskets contain data for many more events that need to be decompressed, but may not be part of any requested event. The per-event read speed times for selective reading 1% of events for AOD is given in Table 1. With the old layout of full splitting and 30 MB event tree, the selective reading slows down to 270 ms/event versus 55 ms/event when reading sequentially. Even though this is a very large penalty, the total read time for the job was still about 20 times faster. The reason for the large increase in per-event reading time is that even for a 1% selection most of the data will be read and decompressed. The new layout speeds up selective reading by a factor of 4 to 5 to just 60 ms/event to the point at which it is nearly as fast as the sequential per-event read time for the old format. Providing read performance for a selected subset of events that is similar to the sequential read performance of an extracted, materialized subset may reduce the number of extracted samples produced by the collaboration and it is expected that this will also benefit multi-processing jobs which use multiple workers on different computing cores to read and process event data and where each worker therefore reads events only from a non-sequential part of the input file.

### Table 1. CPU times (ms/event) for reading AOD events sequentially and selective reading of 1% of events. Fully-split means that the primitive data members of each object are stored in their own branch.

| Layout                  | All events | Selective 1% read |
|-------------------------|------------|-------------------|
| Fully split, 30 MB Auto-flush | 55 (±3)    | 270               |
| No split, 10 event Auto-flush     | 35 (±2)    | 60                |

Read speed was not the only metric studied. File size, virtual memory footprint and write speed measurements are important and performance degradation in these metrics needs to be avoided. For ESD and AOD files written in member-wise streaming mode, flushing every 5/10 events leads to no significant file size increase. Increasing the basket size for auxiliary trees, reduces some ROOT size overhead, but the effect on reducing file size is minimal. The virtual memory footprint with the new storage layout is reduced by 50 to 100 MB (from around 100MB to around 20 MB) for each reading and writing stage, caused by the fact that fewer baskets have to be loaded into memory. The write speed for ESD and AOD is increased by about 20% with the new format. For ESD the compression level can be somewhat relaxed, resulting in an overall write speed improvement of almost 50%.

This was implemented in production before the reprocessing of ATLAS data in autumn 2011, so the current ATLAS implementation is, as described here, to member-wise stream all data associated with a collection into a single branch and auto-flush it every 5 events (for ESD, 10 for AOD) into a compressed basket written to file. The observed read speed for reprocessed AODs in 2011 (which folds in the effect that the ROOT version also changed from 5.26 to 5.28) shows
the speed for ROOT reading improved from 4.6 MB/s to 10.3 MB/s while the Athena reading time (including T/P conversion and that to reconstruct certain objects) went from 2.9 MB/s to 4.9 MB/s (the Athena time excluding reconstruction of those objects went from 3.2 MB/s to 6.4 MB/s).

3. Atlas IO Testing Framework
In order to ensure that the effect of future changes within ROOT versions, or to the file layout are tested on all different storage system types, a testing framework has been developed. This utilises the Hammercloud [6] testing system to send continuous jobs to all large ATLAS Tier 2 sites. Hammercloud was modified to take tests directly from an SVN repository allowing a very quick development-test cycle for changes. The tests all run on an identical dataset for comparison. New datasets (written with new versions of ROOT for example) can be distributed by the Atlas Distributed Data Management system to all sites within a day. The tests run themselves are highly instrumented and collect a variety of performance data, such as ROOT diagnostics and the running conditions of the Worker Node, which are stored in an Oracle database for later analysis. This complements both controlled testing on dedicated machines and monitoring of real user analysis jobs. A web interface is provided for display of results - allowing for easy monitoring by ATLAS and WLCG sites; together with the ability to store as a ROOT TTree for more detailed analysis. A snapshot of the web interface is shown in figure 3.

![Figure 3. Snapshot of Web visualisation tool for testing framework](image)

4. D3PD Optimisations
The D3PD format is rapidly increasing in popularity. Figure 4 shows the fraction of jobs run with “prun” a tool that allows the submission of any user code not using the ATLAS Athena software framework. It is clear that prun jobs are generally running on D3PDs because running
on AOD or user MC production requires Athena. Figure 5 shows, from the filename of real users’ jobs, that most run on the primary D3PDs of 2 big groups Standard Model (SM) and TOP so those are measured within the testing framework. The results shown here are from the SM(WZ) D3PD unless stated otherwise.

**Figure 4.** Number of “prun” (D3PD) jobs (yellow) relative to other analysis jobs showing these now form the bulk of user activity by number of jobs.

**Figure 5.** Total number of jobs in the last year (to 1 May) running on D3PDs (here called “NTUP”) relative to those on AODs showing that the total jobs on D3PDs is greater than AOD and that SM and TOP are the most popular D3PD types.

### 4.1. D3PD compression and auto-flush

D3PDs are currently using a default compression level 1 and the default auto-flush value of 30MB shared among the written branches. Based on local testing in a controlled environment, it was considered that a compression (zip) level of 6 would be more optimal for a range of D3PD types. This was then tested on a variety of site storage systems using the testing framework described above along with a test of extreme values of the auto-flush of 3MB or 300MB. The
impact on total job (wall) time for a few sites with different storage technologies are shown in
figure 6 in terms of the storage system employed. While compression level 6 appears to offer small improvements (lower wall times) at some sites, this is not a significant difference within the spread of results seen. However, for those sites where files are copied to the worker node before local access, the time for the copy is not included in figure 6 but it is measured by the framework and as the file sizes are around 5% smaller there is also a small reduction in the time of that copy. There is also, of course, a corresponding saving in disk space, therefore the higher zip value will be used as default on D3PDs in the future. For autoflush, it is clear that a 3MB size has a significant performance penalty while 300MB does not seem to offer any significant advantages. We also measured the impact of using a larger TTreeCache value on reading these files which suggests that the larger TTreeCache has the same affect on 3, 30 or 300MB autoflush files and that a higher TTreeCache value may offer performance improvements of around 5-10% for all files on sites accessing the data from local worker-node disk.

**Figure 6.** Comparison of wall times for D3PDs written with different compression and autoflush values. The points are the mean values of all tests at a site with the storage type shown. CPU types will differ across sites so comparisons between storage types should not be drawn from this figure. Error bars are not plotted for clarity, but results on the same CPU type and site vary within around 5s.

4.2. Variations within sites and selective reading of events
There is considerable variation in the CPU efficiency (total CPU time divided by wallclock time) seen at various sites. However figure 7 shows that use of the TTreeCache generally improves CPU efficiency and on some storage systems considerably so. We have also found this is even more important in the case of reading a random 10% of events. The TTreeCache can only be set within the user job (and therefore is not by many Atlas users). This indicates that it will be important to explore ways in which it could be set by the job environment in ROOT itself or by introducing a wrapper around user jobs. Also standard D3PD-reading tools have been instrumented in the testing framework and recommendations for users will be provided.
Users of D3PDs do not generally select particular events from the file, however they often only use a small fraction of branches in the D3PDs. Figure 8 shows that for some sites this reduces the CPU efficiency. We observe that the storage systems where high CPU efficiencies are maintained also use a protocol that supports vector reading (dcap or xrootd in these cases).

**Figure 7.** CPU efficiencies for jobs reading 100% of events not using TTreeCache (black circles) or using it with a value of 300MB (red squares). Errors from RMS spread of results.

**Figure 8.** CPU efficiencies for jobs reading around 2% of branches with a TTreeCache value of 300MB (red squares) for 100% events read.
4.3. ROOT versions and developments

As mentioned in the introduction, the testing framework was also intended to track changes in ROOT. For changes within the ATLAS Athena framework this is easily monitored on the web interface and figure 9 shows an example site that no significant change in the running time was observed when ATLAS Athena releases transitioned to ROOT 5.30. However this framework also allows for changing the ROOT release to more recent versions than that employed within Athena. So figure 9 also shows the impact on wall time when the ROOT version from ATLAS is replaced with ROOT 5.32. This is averaged over all sites but indicates that there may be a small performance gain from v5.32. The testing framework can also use the latest trunk version of ROOT code and this will be used to provide a fast feedback mechanism to ROOT developers for improvements to ROOT’s basket optimisation algorithm.

![Figure 9](image)

**Figure 9.** Tracking change from ROOT 5.28 to 5.30 (Athena releases used for 2011 to 2012) (left). Looking ahead to impact of ROOT 5.32 on wall time (right), averaging over all sites for reading 100% of events with ROOT 5.30 (black) and ROOT 5.32 (red).

5. Wide-Area-Network Tests

As ROOT I/O, storage systems and networking hardware have improved in recent years, it is now possible to consider the reading of data over the wide-area network (WAN). Therefore we have begun monitoring this access with the testing framework and figure 10 shows CPU efficiencies for jobs analysing 100% of events on files that are located at remote sites. The CPU efficiencies are lower than when running on data at the same site (for example for the site “OU_OCHEP_SWT2” the drop is from 94% efficiency to around 60%-80%, for other US sites and around 45% for reading from CERN). However these results open interesting options to improve job efficiency in cases where job slots are not available at the sites where data is located while CPU resources at other sites are idle (and so the performance penalty for waiting would be greater then that from the decreased job efficiency). This kind of activity is only beginning to be explored on ATLAS and measurements such as these will provide valuable input into what could be complex scheduling decisions.

6. Conclusions

As described here, there has been considerable recent activity in tuning of ROOT I/O on ATLAS, on both its POOL based and pure ROOT formats, which have lead to significant performance improvements. We have also built a testing framework for monitoring and further tuning of analysis I/O on all ATLAS storage system resources. Using this, a series of improvements to the
Figure 10. CPU efficiency for analysis jobs running at US sites accessing data over the WAN (using xrootd) from CERN or other US sites vs ping time between those sites (top) and on average (bottom). All events are accessed with a TTreeCache of 30MB.

D3PD format have been made, but further monitoring and analysis of the data collected will be important, particularly towards establishing sensible default I/O choices for user analysis, improving ROOT basket optimisation, and enabling wide-area-network reading.

Acknowledgments
The authors wish to thank the help of Johannes Elmsheuser and Daniel van der Ster with Hammercloud; Paul Nilsson with ATLAS pilot jobs and Attila Krasznahorkay with D3PD settings.
References
[1] Vukotic, I, et al. 2011 Optimization and performance measurements of ROOT-based data formats in the ATLAS experiment. ATL-COM-SOFT-2010-081. J. Phys.: Conf. Series 331 032032
[2] Brun, R and Rademakers, F 1997 Nucl. Instrum. Meth. A 389 81. See also http://root.cern.ch/.
[3] Duellmann, D, The LCG POOL project: General overview and project structure, Computing in High-Energy and Nuclear Physics, 2003. See also http://pool.cern.ch.
[4] van Gemmeren, P, Malon, D, 2011 Persistent Data Layout and Infrastructure for Efficient Selective Retrieval of Event Data in ATLAS, arXiv:1109.3119v1.
[5] Cranshaw, J et al. 2010 Event selection services in ATLAS. J. Phys.: Conf. Series 219 042007
[6] Vanderster, D C et al. 2010 Functional and large-scale testing of the ATLAS distributed analysis facilities with Ganga J. Phys.: Conf. Series 219 072021