Journal of Physics: Conference Series **664** (2015) 042033 doi:10.1088/1742-6596/664/4/042033

ATLAS I/O performance optimization in as-deployed environments

T Maier[1], D Benjamin[2], W Bhimji[3], J Elmsheuser[1], P van Gemmeren[4], D Malon[4] and N Krumnack[5] on behalf of the ATLAS Collaboration

[1]Ludwig-Maximilians-Universität München, Germany
[2]Duke University, USA
[3]University of Edinburgh, Scotland
[4]Argonne National Laboratory, USA
[5]Iowa State University, USA

E-mail: thomas.maier@physik.uni-muenchen.de

Abstract. This paper provides an overview of an integrated program of work underway within the ATLAS experiment to optimise I/O performance for large-scale physics data analysis in a range of deployment environments. It proceeds to examine in greater detail one component of that work, the tuning of job-level I/O parameters in response to changes to the ATLAS event data model, and considers the implications of such tuning for a number of measures of I/O performance.

1. Introduction

I/O is a fundamental determinant in the overall performance of physics analysis and other data-intensive scientific computing. It is, further, crucial to effective resource delivery by the facilities and infrastructure that support data-intensive science. To understand I/O performance, clean measurements in controlled environments are essential, but effective optimisation further requires an understanding of the complicated realities of as-deployed environments. These include a spectrum of local and wide-area data delivery and resilience models, heterogeneous storage systems, matches and mismatches between data organisation and access patterns, multi-user considerations that may help or hinder individual job performance, and more.

The ATLAS collaboration [1] has organised an interdisciplinary working group of I/O, persistence, analysis framework, distributed infrastructure, site deployment, and external experts to understand and improve I/O performance in preparation for Run 2 of the Large Hadron Collider (LHC) [2]. The working group has undertaken a program of instrumentation, monitoring, measurement, and data collection in both cleanroom and grid environments. The adoption of a new analysis data model [3] for Run 2 has afforded the collaboration a unique opportunity to incorporate instrumentation and monitoring into analysis code from the outset. The group's findings inform decisions on several fronts, including persistent data organization, caching, best practices, framework interactions with underlying service layers, and settings at many levels, from sites to application code. Increasing robustness and resilience in the presence of faults has been another focus of ATLAS work to improve I/O performance.

Components of this collaboration-wide effort include:

- an I/O performance test suite based upon ATLAS physics analyses;
- regular and automated I/O performance measurements in a range of local and distributed environments, exercising a variety of storage systems and access protocols;
- improvements to intelligent pre-fetching and caching;
- incorporation of data access and caching "best practices" into software releases, into sites' and jobs' default settings, into tutorial examples, and into shared analysis code;
- studies to determine the effects of persistent data organisation strategies upon a range of partial- and full-event retrieval use cases;
- instrumentation of analysis code to monitor which event data objects and attributes are touched;
- development of a reporting infrastructure to collect monitoring information in a distributed environment and connect it to ATLAS analytics;
- improvements to communication between frameworks and underlying infrastructure layers to improve robustness and resilience in the case of I/O faults;
- measurements in support of job-level tuning of I/O settings to improve performance.

It is the last of these that is the primary focus of this paper.
The ATLAS experiment has recorded many petabytes of data from proton and heavy ion collisions at unprecedented centre-of-mass energies during the data taking period from 2010 to 2012. Providing highly performant distributed data access at these scales was one of the many challenges which had to be overcome. In the past, a large amount of effort was put into optimising the various ways of accessing data for the range of formats used for data storage during the first data taking period of the ATLAS experiment [4][5].
With the upcoming start of the next LHC run period, at even higher centre-of-mass energies, a major re-design of the ATLAS data model has been done to address issues which arose during Run I and in preparation for the new data. The new xAOD data format [6] was designed to combine the old AOD(Analysis Object Data) and DPD(Derived Physics Data) formats. AODs were used as output of the data reconstruction in the Athena software framework [7]. Physics analyses preferred the use of the DPD format in the form of D3PDs, which are readable in plain ROOT [8] and more often performed better than AODs due to various Athena related overheads. For the new ATLAS analysis model, xAODs hold the full data information coming from data reconstruction, while being readable in Athena as well as in ROOT. For physics analyses, a centrally produced, trimmed down version (DxAODs) is provided. This format is customised to the needs of the different physics groups.
With the new data model, re-optimisation of old settings related to I/O performance as well as implementation of new ways to improve and monitor data access has become an important task. In the following sections some of the work done by the ATLAS I/O performance working group on job-level performance measurement and tuning is presented. The effect of the ROOT autoflush configuration will be shown, as well as some features connected to ROOT and the xAOD data model, which can be used to improve processing speed. The measurements shown were done in a cleanroom environment, running the respective jobs multiple times while clearing memory before each run to remove bias from cached data. The plots show the averages of the respective measurements. Fluctuations in processing time between runs have been found to be very low. For this reason no uncertainties on the results are shown.

2. Data storing in ROOT

In order to better understand the impact of the autoflush feature in ROOT, which will be discussed in the next section, a brief overview of the ROOT data storage structure is given.

ROOT introduced a new concept called Trees, which expanded the functionality of PAW Ntuples [9] with the use of complex objects and data structures. The data in Trees is stored in branches. Each branch carries its specific content in the form of simple variables, objects, arrays or more complex structures. The branches hold the information for their respective content for every entry stored in the Tree. Each branch has its own individual list of buffers, which means that variables associated to the branch are written into the same buffer.
During processing of data, the contents of the branches are collected in baskets, which are compressed and written to file when they are full. In D3PDs, all properties needed for analysis are directly stored in the branches of a Tree as simple variables (e.g. int, float, char, or vector). Reading these properties can be done directly in ROOT, without requiring additional class structures. The xAOD format adopted the object-based information storing of the old AOD format. General physics objects, like electrons or muons, are stored as xAOD objects in respective containers. Properties related to these objects are stored in separate branches which are accessed by retrieving the object beforehand. This requires a minimum amount of xAOD event data model libraries, which have to be overlaid on top of ROOT.

3. Autoflush as a handle on ROOT I/O

Baskets are self-contained buckets of data. This means that when a variable is accessed for a given entry, the whole basket, in which the information is stored, has to be loaded into memory. This is visualised in Figure 1. Given that every Tree entry is processed (which is common for physics analyses), the number of read operations, which have to be performed when reading the information from a branch, is equal to the number of baskets in the branch. The total amount of data stored in the branches can vary heavily, depending on the kind of information stored. A fixed, static basket size can lead to a very large number of baskets. This affects the time it takes to read the full, stored information. The ROOT Trees autoflush feature is a possible way to dynamically steer the size of baskets for a given branch. This has been found to be very effective for optimising ROOT I/O in the past [4][5].
Activating the autoflush feature before processing defines a threshold for the filling of the branch buffers. While data is processed, the content of the branch buffers are written to disk when this

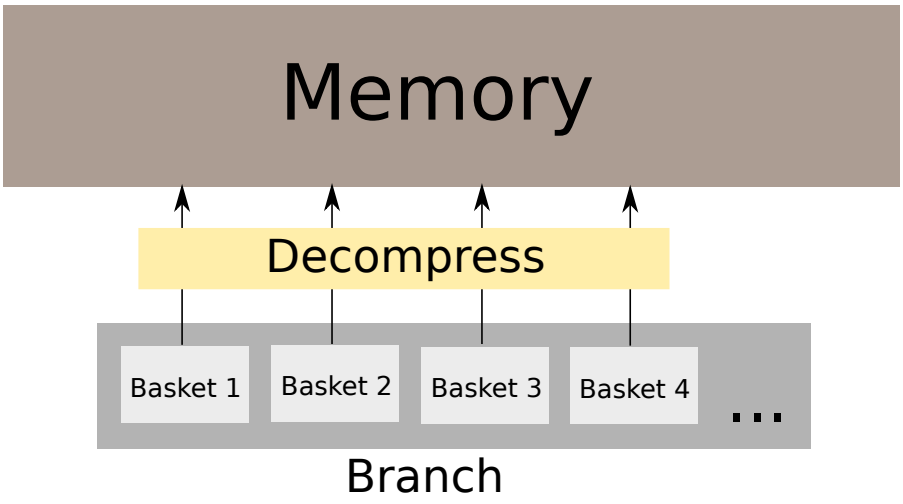

Figure 1. A simplified picture of the basket structure in ROOT branches. When information within a basket is accessed, the whole basket has to be loaded to memory.

threshold is reached. The basket size is then no longer defined by a fixed value, but rather by the amount of data in the buffer at the point of flushing. The threshold can be defined to either be a certain amount of processed entries or an amount of total processed bytes. For the latter case the autoflush value is set to the number of events which have been processed up to the point of the first flush.
For the old AOD data format an autoflush value of 10 has been found to be most suitable [5]. The vast changes in technical structure for the new xAOD data format necessitate re-optimisation of this setting. For this purpose xAOD files were generated for a range of different autoflush values, to study the behaviour on number of baskets and processing speed. Autoflush values of 10, 20, 50, 100, and 200 were taken into account, as well as the autoflushing steered by the total bytes processed (set to the ROOT default value of 30 MB). Additionally the performance of an xAOD with deactivated autoflush was looked at.
Measurements were done with a set of xAOD files, to test the effect on processing speed under common physics analysis conditions. That means all events were processed while accessing a subset of information. Figure 2 shows the measurements for the case that all branches related to the electron container are read, processed in plain ROOT. Figure 3 shows the measurements for a full analysis using the Athena software framework. Both measurements show a noticeable gain in processing speed when using higher values for the autoflush setting.
Additional to the effect on processing speed, another observation was made regarding the size of the xAOD files. Figure 4 shows the disk space consumption of the files against their respective autoflush setting. Additionally, the virtual memory consumption, when running the files in a merging job, which copies the full content to a new file, is shown. A decrease in file size

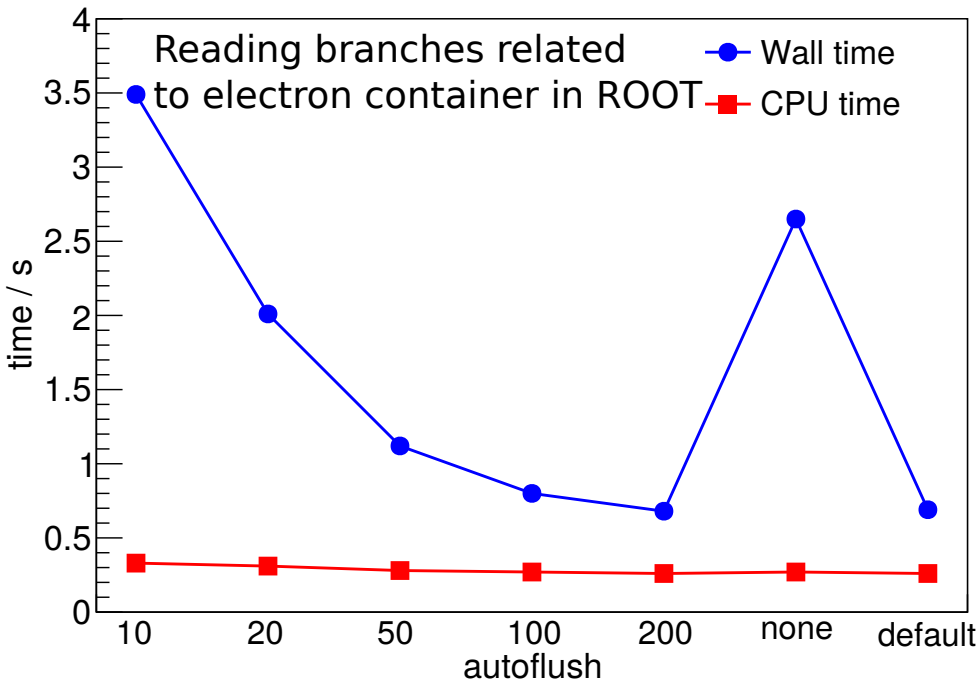

Figure 2. Measurements of processing time when reading branches related to the electron container for every event processed in ROOT, for different values of autoflush in the input xAOD file.

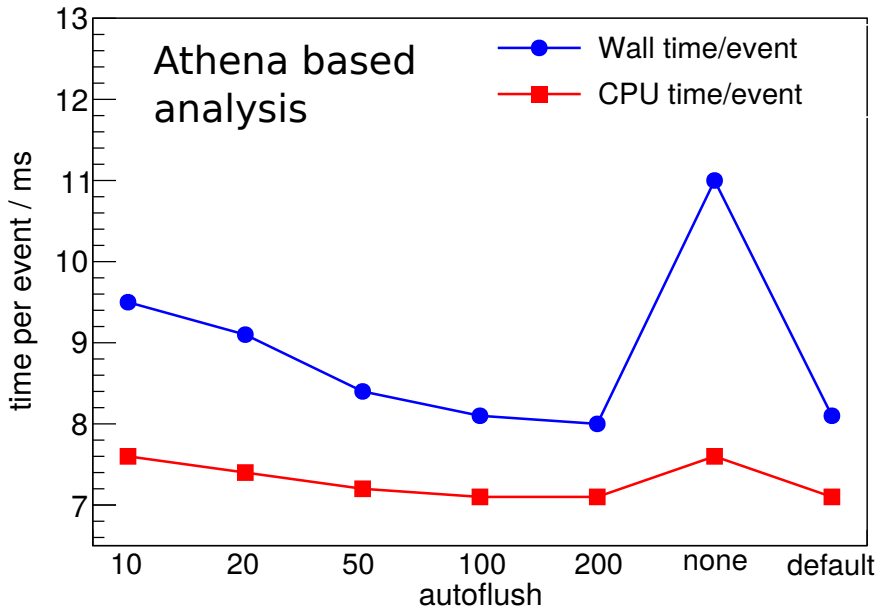

Figure 3. Measurements of processing time spend per event for a full analysis using the Athena software framework, for different values of autoflush in the input xAOD file.

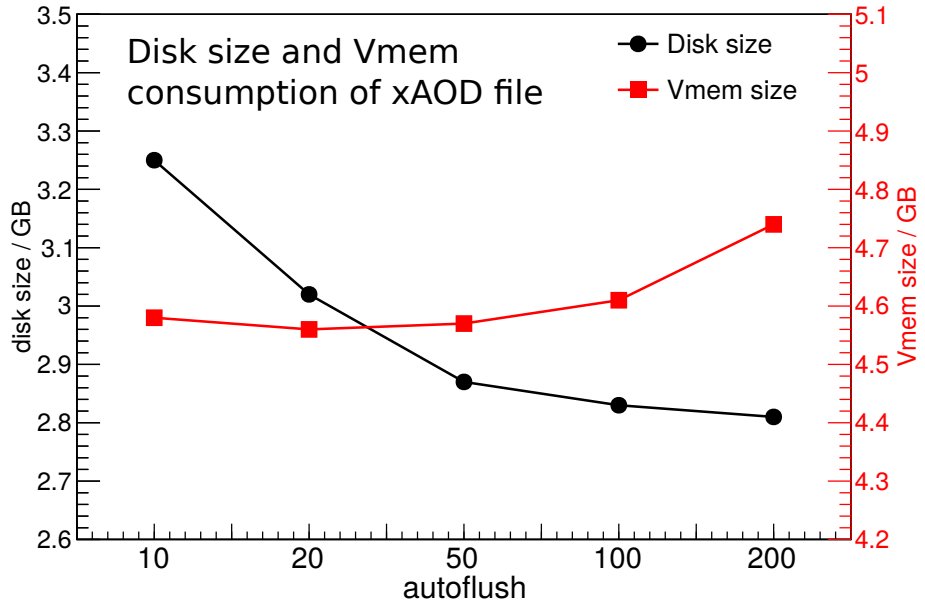

Figure 4. Disk space (black) and virtual memory (red) consumption of an xAOD file depending on the autoflush setting. The virtual memory footprint is measured by running the file in a merging job, which copies the full content to a new file.

can be observed when increasing the autoflush value, while only a very slight increase in the virtual memory footprint could be measured. This can be attributed to the increased amount of compressible data per basket, which allows for higher compression rates. According to the

observations made, the current autoflush setting for xAOD files was increased from 10 to 100.

4. Additional handles on reading speed

There are a couple of additional features, related to the new xAOD data model and ROOT, that can increase processing speed. The TTreeCache(TTC) feature of ROOT can be activated before processing to run in a pre-caching mode. During a defined number of events the TTC identifies the branches which are read by the users job. These branches are then read in advance and the content is cached. This can lead to considerable improvements in processing speed. Figure 5 shows the difference in CPU efficiency with TTC on and off. While it gives a slight improvement for locally read data, it is crucial for remotely accessed data (here shown by remote access using XRootD [10]).

When accessing information in plain ROOT, the new xAOD data model provides a feature which can reduce processing time. The user can decide to access the data in class-wise or branch-wise reading mode. The former will read all information connected to a container, when the container is accessed. The latter reads the data of a given branch only when the variable connected to this branch is accessed. This increases reading speed when only a small subset of variables of the accessed container is actually used. Figure 6 shows the effect of using this branch-wise reading, when iterating over muons, electrons, and jets while making common physics computations. For full xAODs the effect is quite large, since a lot of branches are not accessed by common physics analyses. The effect is a lot less significant for DxAODs, since their contents are already trimmed down for physics needs.

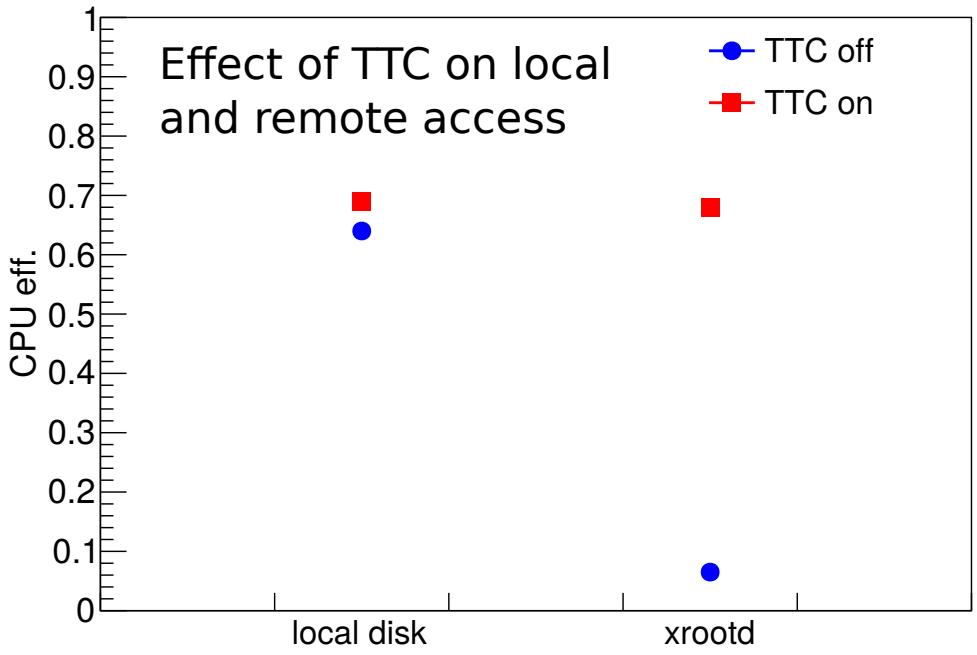

Figure 5. Difference in CPU efficiency for TTreeCache(TTC) turned on and off, measured for local and remote XRootD access.

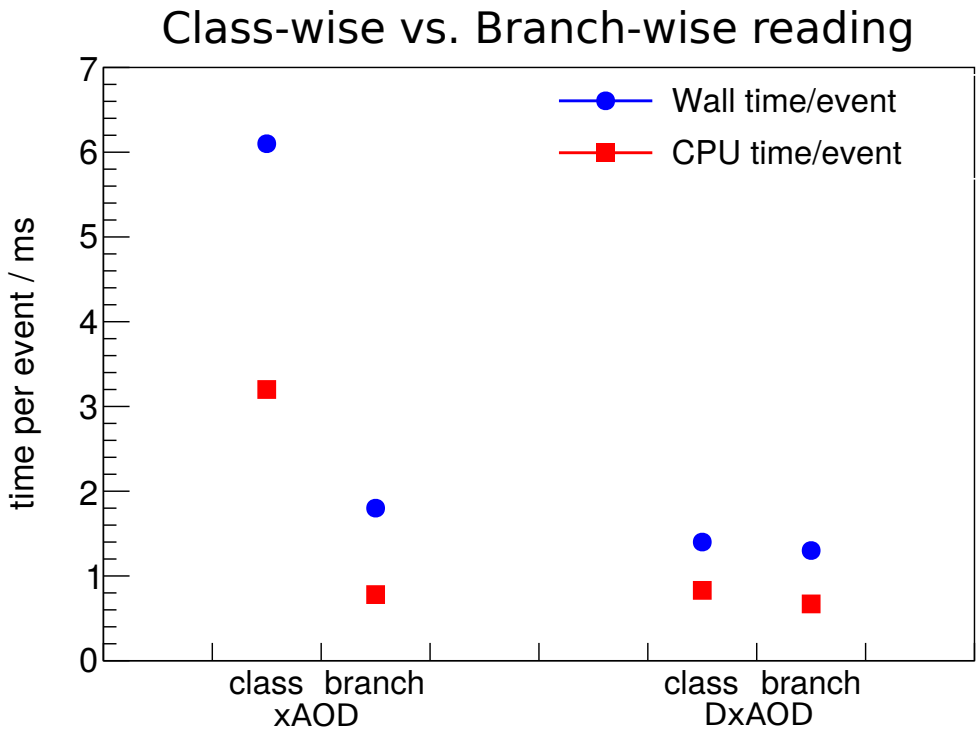

Figure 6. Difference in processing time spend per event between class-wise and branch-wise reading mode, measured for xAOD and DxAOD input.

5. Conclusions

The adoption of the new ATLAS analysis data model for Run 2 of the LHC induced the necessity of re-evaluation of known I/O improving ROOT features and configuration. First improvements were already made by re-optimising the ROOT autoflush setting in the new xAOD format. Old and new I/O improving features were studied to test their effect on xAOD reading. These improvements are one important component of an ongoing and broader program of work within the ATLAS collaboration to understand and address issues of I/O performance in distributed and local analysis environments.

References

[1] ATLAS Collaboration 2008. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST* **3** S08003
[2] Evans L and Bryant P 2008. LHC Machine. *JINST* **3** S08001
[3] Catmore J on behalf of the ATLAS Collaboration 2015. A new petabyte-scale data derivation framework for ATLAS. Proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) *J. Phys.: Conf. Ser.*
[4] Vukotic I on behalf of the ATLAS Collaboration 2011. Optimization and performance measurements of ROOT-based data formats in the ATLAS experiment. *J. Phys.: Conf. Ser.* **331** 032032
[5] Bhimji W on behalf of the ATLAS Collaboration 2012. The ATLAS ROOT-based data formats: recent improvements and performance measurements. *J. Phys.: Conf. Ser.* **396** 022006
[6] Snyder S on behalf of the ATLAS Collaboration 2015. Implementation of the ATLAS Run 2 event data model. Proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) *J. Phys.: Conf. Ser.*
[7] Calafiura P, Lavrijsen W, Leggett C, Marino M and Quarrie D 2005. The athena control framework in production, new developments and lessons learned. CERN-2005-002 p. 456-458, https://cds.cern.ch/record/688747

[8] Brun R and Rademakers F 1997. ROOT - An Object Oriented Data Analysis Framework. *Nucl. Inst. & Meth. in Phys. Res.* A **389** p. 81-86, http://root.cern.ch
[9] Brun R, Couet O, Vandoni C and Zanarini P 1991. PAW users guide, CERN Program Library Q121, http://paw.web.cern.ch/paw
[10] Gardner R on behalf of the ATLAS Collaboration 2014. Data federation strategies for ATLAS using XRootD. *J. Phys.: Conf. Ser.* **513** 042049