Status and Future Evolution of the ATLAS Offline Software

This content has been downloaded from IOPscience. Please scroll down to see the full text.

(http://iopscience.iop.org/1742-6596/664/7/072044)

View the table of contents for this issue, or go to the journal homepage for more

Download details:

IP Address: 137.138.124.206
This content was downloaded on 24/02/2016 at 12:59

Please note that terms and conditions apply.
Status and Future Evolution of the ATLAS Offline Software

R Seuster\textsuperscript{1}, M Elsing\textsuperscript{2}, G A Stewart\textsuperscript{3} and V Tsulaia\textsuperscript{4} on behalf of the ATLAS Collaboration

\textsuperscript{1} TRIUMF, 4004 Wesbrook Mall, Vancouver, BC V6T 2A3, Canada
\textsuperscript{2} CERN, CH-1211, Geneva 23, Switzerland
\textsuperscript{3} School of Physics and Astronomy, University of Glasgow, University Avenue, Glasgow G12 8QQ, UK
\textsuperscript{4} Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, USA

E-mail: Rolf.Seuster@cern.ch

Abstract. These proceedings give a summary of the many software upgrade projects undertaken to prepare ATLAS for the challenges of Run-2 of the LHC. Those projects include a significant reduction of the CPU time required for reconstruction of real data with high average pile-up event rates compared to 2012. This is required to meet the challenges of the expected increase in pileup and the higher data taking rate of up to 1 kHz. By far the most ambitious project is the implementation of a completely new analysis model, based on a new ROOT readable reconstruction format, xAOD. The new model also includes a reduction framework based on a train model to centrally produce skimmed data samples and an analysis framework. These proceedings close with a brief overview of future software projects and plans that will lead up to the coming Long Shutdown 2 as the next major ATLAS software upgrade phase.

1. Introduction

The Large Hadron Collider (LHC) is a \textit{pp} collider at the European Organization for Nuclear Research, CERN, in Geneva, Switzerland. In early 2013, it stopped the very successful data-taking period from 2009 to 2013, called Run-1, to undergo an extensive upgrade and maintenance program. This maintenance period is called Long Shutdown 1, or LS1. The LHC now enters a new data taking period, Run-2, where an increase in the center of mass energy to about 13 TeV will expand the reach of the LHC experiments to explore new physics phenomenon at a variety of scales, such as insight into dark matter and dark energy phenomenon. After Run-2, in 2018, the machine will once again shutdown for major upgrades in Long Shutdown 2 (LS2).

The LHC will increase the center of mass energy at which it will operate during Run-2 to about 13 TeV. The data taking conditions will be substantially different from Run-1. The bunch-spacing will be 25 ns, the design value of the LHC. The instantaneous luminosity will well exceed the design value, and the ATLAS [1] experiment will see up to 40 interactions per bunch crossing, posing new challenges for the trigger systems, offline reconstruction and the analysis of the new data.

During LS1 ATLAS scheduled extensive detector upgrades. The tracking system was complemented by a fourth layer of the pixel detector, the insertable b-layer (IBL), closer to the beam pipe. Another change directly affecting the software is the increase of the final output
rate of the last stage of the trigger, the Event Filter (EF), to 1kHz, an increase of around a factor of 2.5.

In the ATLAS offline software several major upgrades were implemented during LS1 to cope with the harsh conditions in Run-2. Several deficiencies of the whole analysis model used during Run-1 were identified and rectified. These deficiencies result in changes for the whole chain of reconstruction of the initial data until the creation of the ntuples for the various working groups in ATLAS.

These proceedings will highlight the larger picture and summarise the progress achieved during Run-1. Other papers give more details on specific topics and are referenced in the corresponding section.

2. Reduction of Processing time

A direct consequence of the increase of the event filter output rate to 1kHz is, that the offline reconstruction software also needs to be improved significantly to allow a prompt first processing at the ATLAS Tier-0. Taking also the increased particle multiplicity due to the increased center of mass energy from 8 TeV to 13 TeV into account, an improvement in the reconstruction speed at an average pile-up events per collision, denoted with $\langle \mu \rangle$, of 40 of at least a factor of 3 is required.

![Figure 1. Reconstruction time versus releases. 17.2 was used at Tier0 in 2012, 19.X are intermediate snapshots during LS1 and 20.1 will be used at Tier0 in 2015. The plot is available from [14].](image)

In Run-1, the tracking in the Inner Detector dominated the event processing time, so an extensive upgrade program of its software was started. The seeding strategy for the tracking is now optimised for high pileup and leads to a speedup of about a factor of 2. Small, but relatively expensive parts of the tracking (so called back-tracking to find conversion tracks) is now done only in regions around electromagnetic showers in the calorimeter. An optimised tracking in dense environments gave another improvement of about 10%. On top of all that, more technical changes include the upgrade of the linear algebra library, previously provided by the CLHEP suite [2], to the modern Eigen library [3]. This package was shown to be fastest compared to other similar packages, as it partially uses single-instruction-multiple-data (SIMD) instructions, and at a later stage can provide opportunities for auto-vectorisation. In general,
as code was modified for the migration to Eigen, clean ups were done, unused functionality
removed and general improvements implemented like using new C++11 features. The service
providing the value of the magnetic field at any point inside the detector was converted from
Fortran to C++ and, together with intelligent caching, a significant speedup is achieved. This
caching is foreseen to be able to be modified to be thread safe, which is important for future
software upgrades. A few changes resulted in a speedup without any code changes, so called 'free
lunch' improvements: a more modern version of the gcc compiler suite, changing the default to
64-bit instead of 32-bit executable code. The change of the underlying operation system to SLC
version 6 also provided a speedup of about 10%, mainly from improved Linux kernel performance.
The last ‘free lunch’ was from using the optimised Intel mathematical library, which provides
highly optimised transcendental function implementations utilising SIMD instructions. Taking
all improvements together, ATLAS archived a factor of about 4 in speedup, well exceeding the
required factor of three, see Fig. 1. More details are provided in other proceedings for this
conference [4] and [5].

3. Memory usage reduction
During LS1, ATLAS moved to use 64-bit executables by default, as they provide a speed
improvement of about 25%. The cost of this is, however, a higher memory consumption of
between 25% to 50%, depending on the workload. This pushes the memory footprint of some
workloads to well above 2GB per core, which is the agreed minimum required by WLCG and
therefore a limit imposed by some Grid sites. (However, it should be made clear that athena
can swap out large parts of the memory and still run at full CPU speed as explained shortly.
athena is the name of the executable that is run for all workloads and has become a synonym
for the ATLAS software framework.)

During LS1 ATLAS introduced athenaMP for almost all workloads, which is a multi-process
version of our athena software framework. athenaMP relies on the copy-on-write feature of the
Linux system. Workers are forked off a mother process after the initialisation of the job in the
mother process. The events are then distributed to workers which process them concurrently
until each of the worker finalises. Each worker writes its own output file. The mother then
goes through a second, lightweight finalisation step. Forking during the first event results in
significant sharing of the worker and the mother memory pages.

This allows ATLAS to save large amounts of memory, e.g. a reconstruction job with 8 workers
can fit into about 15 GB of memory, so well below the 2GB per core, see Fig. 2.

For more details about the current status of athenaMP see [6].

A major problem for running athenaMP applications on the Grid is memory accounting as
most tools from the operating system do not report accurate numbers. Tools like “ps” or “top”
only take sharing of text code for libraries into account, not of heap of a program. This results
in a massive underestimate of the shared memory for athenaMP and a huge exaggeration of
memory usage, in extreme cases even reporting more RSS than the worker node has available
as real memory. Fig. 2 shows this problem. The black line, “PSS smaps”, shows the real
memory usage that should be used for accounting purposes. It is based on smaps, and bases
its memory calculation on PSS, the proportional set size. PSS is calculated by dividing RSS
by the number of processes which use this memory page, thus is the same as RSS in the case
of no sharing. The magenta line shows almost the same observable, but omits programatically
shared memory, which is being used for communicating between mother and worker processes
and also to transfer input events from the mother to the worker processes in the case of reading
in real data in byte-stream format. The red curve shows the RSS, also reported by smaps,
where sharing is explicitly not taken into account. This value is close to what the operating
system tools like “ps” and “top” would report and a massive overestimation of the real memory
consumption is seen. The green curve shows the virtual memory, without sharing taken into
Figure 2. Different measures to determine memory usage of athenaMP jobs. “RSS smaps” and “Vmem smaps” do not take sharing into account and overestimate largely the real value given by “PSS smaps” or also by “RSS cgroups”. The plot is available from [14].

account and the deviation from the true value is even more visible.

Figure 3. Same as Fig. 2 but with available memory artificially limited by cgroups settings to 12GB shown by the black line. The plot is available from [14].

Fig. 3 shows the time evolution of the various memory usage measurements for the same job run under artificially created memory pressure: Via utilising cgroups the total amount of the memory was limited to 1.5 GB per worker process, so in this case to 12 GB.

Both the black and the magenta stay below the limit. The red and green curves go well beyond the limit, as expected. The blue curve shows the amount of swap being used. The interesting feature of this plot is that the run time of the job is almost unaffected. Also athenaMP can swap out large parts of memory, which will only be used later in the finalisation of the job, where it then will be swapped in again. Decreasing the memory further can hurt the performance once
memory actively used in the event loop needs to be swapped out.

4. New Analysis Data Model
The event data model (EDM) used during Run-1 was very focused on the needs of reconstruction and was not easily readable in plain ROOT [7], requiring many support libraries consuming 100s of MB of memory. Physics groups therefore produced large ROOT-readable ntuples, “DPDs”, which contained almost the same information as the primary AODs. This duplication of information consumed many PB of disk space on the Grid and increased considerably the time data sets became available to the users.

![Figure 4. Sketch of the xAOD data layout.](image)

The new Run-2 EDM is called xAOD and is, with no extra code or libraries, readable in ROOT. In large parts, the reconstruction code is now also using this EDM, with the requirement that all classes that are used for analyses must be in the xAOD format. Thus reconstruction algorithms must directly write out and internally use these xAOD classes.

The performant access in ROOT and athena is based on a streamlined layout. A single base class is used to define the interface for all EDM classes and provides unique and simple interface for users, see Fig. 4. These lightweight objects contain only basic information like 4-momenta of the objects and only forwards access to additional information. Any further information is stored as such additional data in the AuxStore, a simple to use extension where arbitrary data can be stored and retrieved via string keys or enums. The new xAOD has no separation of the transient and persistent layers, but relies on ROOT schema evolution.

The memory layout is chosen to be structure-of-arrays instead of the array-of-structures as in the previous EDM. This enables a better data locality of when, e.g., looping over large number of objects and selecting only a very few. This memory layout offers by far more possibilities for auto-vectorization by the compiler, which will result also in local performance boosts.

More details about the new EDM can be found here [8].

5. Getting data to the users
With the new ROOT readable xAOD data format, users can directly start their analysis, without the need to produce a ROOT readable ntuple. However, with the huge data volumes, it is highly undesirable for anybody to run over the whole PB sized data set. Additionally, different analyses require different information from the events. The procedure for most analysis for Run-2 is the
following: the production system runs centrally over the whole data set and creates about 100 subsets, called derivations, which contain only selected events. Furthermore, just a subset of all collections per event can be selected. In these collections only selected objects can be written out. And last, one can also configure what additional information should be written out together with these objects. This additional information can be existing already in the primary xAOD or can be computed during the reduction step.

The reduction step done by the derivation framework has been introduced as an additional step in the processing of data and Monte-Carlo in the production system, thus the production of the derived xAOD formats can be largely automated.

To minimise the code users have to write for this, the configuration of the selection of events and its objects is text based. Special attention was paid to making this as efficient as possible. Also, the derivation framework takes care of including the right variables for each calibration or object selection that might be used in a later analysis of these written-out objects. A detailed monitoring closely follows sizes, overlaps between different selections of the various outputs.

Optionally, parts of the reconstruction can be re-run in the derivation framework in order to fix problems encountered in the initial processing similar to that provided by the AODFix framework used in Run-1.

More details about the derivation framework are given here [9].

6. Athena and ROOT: code re-usages

With the new xAOD, it is possible to finalise an analysis in either ROOT or in athena. To ensure the consistency of the analyses done in each framework, we now ensure that the same code is run in both frameworks, except for very framework specific code like logging. Typical steps in the final stage of an analysis are the final selection of objects, as recommended by performance groups, as well as the final calibration of many objects, like jets or electron and photon candidates, or corrections for identification and reconstruction efficiencies is performed. To achieve these steps correctly, in ROOT or athena, ATLAS has developed a dual-use tool model, where tools can be written that can be compiled for either framework and results are assured to be the same. Access to EDM objects, logging and retrieval of calibration data is possible. Calibration data is made available in a structured way using files stored in CVMFS, thus there is no need for a tool running in ROOT to require a database access layer.

To minimise the code users have to install e.g. on their laptop, dedicated analysis releases are distributed. These releases are lightweight and easy to install. Due to their small size, e.g. ROOT-6 was implemented there very quickly.

More details about the dual use tools are given here [10].

7. Future Framework: initial R&D

The most promising new framework for a multi-threaded athena application is GaudiHive [11], because athena is based already on Gaudi and therefore minimal changes are required to the overall software. Clearly, some of the algorithms, tools and services need to adapted to become multi-thread capable, and significant modifications will be necessary. Initial tests have been run on a subset of the full reconstruction, but look promising. Large parts of the calorimeter reconstruction is already running within GaudiHive/athena, also parts of the reconstruction in the Inner Detector. Code aimed at speeding up reconstruction in a single threaded Gaudi often needs to be modified. Examples are caching or memory pools. In the long term they need to be implemented in a thread-safe way, but have not been done at this stage, as the primary goal was to evaluate memory consumption, not speed of execution.

First results look very promising. For the calorimeter test bed serial bottlenecks were the limiting factor, however with 6 concurrent events a speedup of a factor of about 4 was achieved, whereas the memory increased by just 36%.
Certain design patterns common in some parts of the ATLAS reconstruction also have been shown to be difficult, at least with the forward scheduler in GaudiHive, e.g., the record-update pattern found often in the muon reconstruction. Here, one algorithm creates an empty collection, which is then filled and further modified by successive algorithms and tools. A newer scheduler for GaudiHive will correctly handle this use case, but remains to be tested. Another pattern very much used, which is difficult for multi-threaded frameworks, is “incidents”, which are unscheduled requests to execute code when a particular condition, e.g., BeginEvent, is encountered (a.k.a “firing an incident”). The problem here is that such incidents often lose their meaning when multiple events are in flight. Such patterns can often simply be replaced by scheduled work flows.

Another very interesting project is the hybrid execution of GaudiHive within athenaMP. Events are shared as for athenaMP, then per each worker process single events are processed within GaudiHive with multiple algorithms in flight. This reduces significantly the requirement of making code thread safe and provides a workable, intermediate solution until a significant amount of required code has been made fully thread-safe.

More details about the status of the R&D of a multi-threaded framework can be found here [12].

8. Future Framework: requirements
At about the same time as the initial R&D for the GaudiHive, ATLAS has collected the framework requirements from various domains, such as trigger, offline reconstruction, simulation and analysis. These have been summarised and are being considered by the collaboration, but some main points can be discussed here.

In order to avoid common threading problems, such as race conditions, tools need to be private to a particular algorithm. This is unlike the situation currently, where tools can be public and shared between algorithms. The scheduling of algorithms in a threaded framework depends on the EDM objects they read, write or update. The scheduler will take care of running algorithms in the correct order, but that requires that all algorithms declare what data they will create, consume or update, and that all algorithms pass this data via the whiteboard.

The same EDM objects from different events can exist at the same time in the whiteboard, so it is necessary to be aware of the event context when asking the whiteboard for data, which should be transparent to the algorithms and tools themselves. The whiteboard is a particular case of a “service”, which needs to be context aware and thread-safe. Note that caching, which is common in high-performance services, needs to be carefully evaluated and, if needed, re-implemented in a thread-safe way.

For code that is not thread-safe, either because it is hard to migrate, uses an underlying serial resource or simply that it consumes so little CPU time as to not be worth the effort to migrate, a legacy mode will be available that ensures that it will be run only once at any point in time. This feature is also vital to achieve an adiabatic migration of algorithm and tool code to the new framework.

In the new framework it is foreseen that any remaining incident based work flows should be implemented by having incidents simply add new tasks to the scheduler’s queue, thus overcoming many of the problems from unscheduled code execution.

The ATLAS trigger requires a new concept, event views. The Level-1 trigger system provides Regions of Interest, where only parts of a whole event need to be checked for interesting features (this hugely reduces the required bandwidth from the detector to the trigger farm and the CPU investment in events that are not accepted). For the new framework, a similar requirement is formulated, that the framework needs to support processing only parts of an event. This will require various extensions to the whiteboard service. It should be noted that this feature can also be used for simulation or reconstruction, where parts of the event will be fully processed in
full detail and other, less interesting parts like tracks from pileup-events will be processed only with a fast simulation or reconstruction. The framework will also allow the usage of accelerators, if these are present. More details about the requirements for a future framework can be found here [13].

9. Conclusions
The ATLAS offline software during LS1 has evolved significantly to match the new requirements the experiment is facing for Run-2. It covers all areas from data taking to analysis. The reconstruction software has been sped up by a factor of 4, matching the higher trigger output rate and the higher multiplicities at the higher center-of-mass energy in Run-2. The whole analysis model has been overhauled, from the EDM, to the distribution of the data and Monte-Carlo via the Derivation Framework and the analysis itself utilising dual use tools. The memory consumption has been significantly reduced by utilising a multi-processing software in full production.

For the future, research and development has started to transform the multi-processing framework further into a multi-threaded framework. Requirements for such a framework from all domains have been collected and will be taken into account in the future development of ATLAS software, preparing for LS2 and Run-3.

References
[14] https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults