Optimisation of the ATLAS Track Reconstruction Software for Run-2

Andreas Salzburger, on behalf of the ATLAS collaboration
CERN, Switzerland
E-mail: Andreas.Salzburger@cern.ch

Abstract. The reconstruction of particle trajectories in the tracking detectors of experiments at the Large Hadron Collider (LHC) is one of the most complex parts in analysing the data from beam-beam collisions. To maximise the integrated luminosity during Run-1 of the LHC data taking period, the number of simultaneous proton-proton interactions per beam crossing (pile-up) was steadily increased. The track reconstruction is the most time consuming reconstruction component and scales non-linearly in high luminosity environments. Flat budget projections (at best) for computing resources during the upcoming Run-2 of the LHC together with the demands of reconstructing higher pile-up collision data at rates more than double compared to Run-1 have put pressure on the track reconstruction software to meet the available computing resources. The ATLAS experiment has thus performed a two year long software campaign which led to a reduction of the reconstruction time for Run-2 conditions by a factor of four: a major part of the changes were improvements to the track reconstruction, which was reduced by more than a factor of five without any loss of output information for subsequent physics analysis. We present the methods used for analysing the software, the planning and deployment of updates and new methods implemented to optimise both algorithmic performance and event data.

1. Introduction
The Run-1 data taking of the LHC from 2010 to 2012 marked a very successful period for high energy physics, concluding with the exciting discovery of the Higgs boson by both the ATLAS [1] and CMS [2] collaborations. The integrated luminosity needed for this discovery and many other outstanding physics measurements was only possible to be delivered by the LHC through a steady increase in the number of simultaneous proton-proton interactions per beam crossing, which will be in the following referred to as pile-up \( \mu \). The peak number of \( \mu \) reached up to 40 by the end of the Run-1 (Fig. 1), which put the event reconstruction under pressure to stay within the given computing resources. The CPU time used for track reconstruction depends particularly strongly on \( \mu \), since it is a pattern recognition problem that suffers from a highly non-linear increase of possible combinations for a linear increase in the number of read-out hits. Track reconstruction in ATLAS uses the Inner Detector (ID) and the Muon Spectrometer (MS). Large parts of the software are common to both and organised in a common tracking software project. However, the focus of this document is on the ID reconstruction since it takes the most significant part of the overall event processing time.
1.1. Run-2 projections
The increase of centre of mass beam energy from 8 to 13 TeV will lead to an increase of the event complexity, leading effectively to more particles being produced per proton-proton collision. At the same time, the average event pile-up $<\mu>$ is foreseen to increase to an average value of 40 (from about 20 at the end of Run-1). Finally, ATLAS will increase the high level trigger rate from 400 Hz to 1 kHz, essentially more than doubling the data rate to be processed. Projections based on these changes and under the assumption of flat budgeting of computing resources at best revealed the requirement to decrease the event processing by at least a factor of three to stay within the available computing resources.

2. The ATLAS Inner Detector track reconstruction
ATLAS is a general-purpose detector [3] designed to make precision measurement of known physics and to probe new physics at the energy frontier of the LHC. At the centre of the detector is the ID: an optimised, multi-technology inner tracking device embedded in an axial magnetic field with a central value of 2 T. The ID is designed to reconstruct trajectories (also called tracks) from charged particles. It has a silicon pixel detector at the innermost radii, which is surrounded by a silicon strip detector and a drift straw tube detector (TRT) that identifies particle types using transition radiation. During the shut down period from early 2013 to 2015, a new innermost pixel layer, the insertable b-layer (IBL) was installed and commissioned. An illustration of the ATLAS silicon barrel system including the newly installed IBL is shown in Fig. 2. The ID provides full hermetic $\phi$ coverage for particles produced within pseudo-rapidity $|\eta| < 2.5$.

![Figure 1. Peak interaction per bunch crossing during Run-1 of the LHC.](image1)

Track reconstruction uses local and global pattern recognition algorithms to identify measurements that stem from the same charged particle. The found measurements are then used in a track fit to estimate the track parameters. The ID track reconstruction consists of several sequences with different strategies as described in [4]. The main sequence is referred to as inside-out track finding, which consists of the following components that are executed in the following order:

- **Data preparation and space point formation:** The initial step of the ID reconstruction consists of the cluster and drift circle creation and the transformation of clusters in the silicon detectors into 3D space points. Clusters are formed by finding connected cells in the pixel and strip detectors. A neural network based cluster splitting module for the pixel detector [5] has been deployed during Run-1 to identify and resolve merged clusters from close-by particles in dense environments. Space points are built by using the module surface and applying the coordinate transformation from the measured local to the global coordinate system.
• **Space point seeded track finding:** track finding starts with the formation of space point triplets (seeds). Seeds can be built from space points in the pixel detector only (referred to as PPP seeds), the strip detector only (SSS) or any mixed setup (PSS,PPS). To reduce the number of potential seeds, initial cuts are applied and dedicated care is taken not to extensively use space points in multiple seeds. Seeds that pass the initial requirements are then input to a track finding algorithm that uses a combinatorial Kalman filter technique and aims to complete the track candidates within the silicon detector.

• **Ambiguity solving:** track candidates are then further processed in an ambiguity solving module that aims to eliminate track candidates from random hit combinations (often referred to as *fakes*) or track duplicates, which can be identified by measurements that are shared with other track candidates. The ambiguity solving relies on a scoring function applying positive scores for unique measurements and good fit quality, while penalising missing measurements where they would be expected (also called *holes*) or shared measurements with other track candidates.

• **TRT extension:** tracks that successfully pass the ambiguity solving stage and are within the coverage of the TRT detector are then extended into the TRT and completed for measurements in the outermost tracking detector. A successful TRT extension increases the momentum resolution significantly by exploiting the longer lever arm for field integration.

3. The LS1 software projects

The initial planning phase of the LS1 track software campaign started during the final phase of data taking during 2012, when the high peak values for $\mu$ revealed the increase of the necessary CPU time at larger scale. A thorough analysis of the track reconstruction software was started in preparation for an ATLAS Tracking software workshop (November 2012). Individual projects in the areas of geometry, the event data model (EDM), algorithms and framework were identified using mainly CPU performance tools such as kcachegrind [6], gperftools [7] and other common or ATLAS specific CPU monitoring tools. In addition, a review of the software structure in the light of lessons learned from the first data taking period was performed targeting code quality, maintainability, but also future use of modern computing techniques such as concurrent computing. In general, two main areas have been identified for improvement: updates and optimisations of the infrastructure (including common services and the EDM) and algorithmic improvements by change of strategy or optimisation of workflows. It became quickly evident that — given the limited time available during LS1 — the projects had to be deployed in parallel, making this software campaign particularly challenging. Some additional improvements came from purely technical infrastructure, e.g. switching from 32bit to 64bit architecture or changing to Scientific Linux 6, but those account for at most 15 percent and will not be presented further in this document.

3.1. Infrastructure and event data updates

3.1.1. Change of math and algebra library

Track reconstruction relies heavily on matrix manipulations and the use of standard math functions, which take a significant amount of CPU time. While the use of math library function such as trigonometric function calls, exponents or other standard operations can be changed even at runtime by preloading a different math library, the algebra library is usually a deeply integrated component of the EDM and algorithmic code. The ATLAS Run-1 track reconstruction software was based on the CLHEP library [8], which was used to express the geometrical and the algebraic components. Evidently the execution speed of general track reconstruction processes, such as e.g. the transformation of local measurement coordinates into a global representation, the transport of covariances using transport Jacobians, or Kalman filtering updates were dominated by the intrinsic performance
of matrix manipulations within the CLHEP library. These operations are in general within a dimensional space of one (as of a single one-dimensional measurement) up to five (for the full transport and update of track parameters). In order to investigate alternative libraries, ATLAS has produced a testbed mimicking typical operations that appear frequently in the track reconstruction process. Several libraries were tested, including MKL [9], the ROOT SMatrix library [10] and the open-source project Eigen [11]. As an outcome of the testbed comparison, the Eigen library has been chosen. Eigen is a heavily templated library that was written specifically in order to optimise the calculations for exploiting SIMD instructions and thus aiming to perform better when exploiting the increasing vector registers of modern CPUs. About 40 percent of the entire ATLAS software repository had been reworked, removing the explicit use of CLHEP and changing to the newly introduced ATLAS math and geometry (AMG) classes that are type definitions of the corresponding Eigen classes. Eigen offers a predefined preprocessor pragma that includes an additional header file to the library that allows to extend the user interface. This feature was used to adapt the initial Eigen interface to mimic CLHEP classes in order to simplify the software transition for the developers.

3.1.2. Cleanup of the tracking event data model  The change from CLHEP to the new AMG math library required a rework of the entire track reconstruction EDM. The EDM design patterns were redefined in the light of the Run-1 experience and associated MC campaigns. The ATLAS Tracking EDM [12] was initially designed in 2003 following the recommendations of the ATLAS Reconstruction Task Force [13]. The core of the Tracking EDM is a container track class that holds measurement representations, track parameterisations with associated covariances, and objects to define material effects and track quality descriptions. The chosen design was tailored around the blackboard data design pattern of the ATHENA framework [15] and was based upon the following considerations:

- **type safety**: code conventions are disfavoured relative to type conventions, e.g. a charged track parameterisation (with momentum parameterisation \(q/p\)) should be characterised by a different class than a neutral track parameterisation \((1/p)\) in order not to rely on client code checks before the momentum parameterisation becomes clear. Or, an expression of a track with respect to a certain surface type (e.g. planar surface) should be described by a different class than with respect to another surface type in order to avoid potential combination errors.

- **object constness**: a reconstruction job in ATHENA is realised as a sequence of algorithms that read and write from a common data store. Changing objects in the store should not be permitted in order to avoid uncontrolled behaviour in a downstream algorithm.

- **lazy initialisation**: in the initial design, derived class members were to be calculated only when first time requested, and cached then.

During the review of the Tracking EDM it became evident that the type safety and object constness were very useful and successful strategies and were desired to be kept for Run-2. The lazy initialisation, however, led to rather costly dynamic memory allocation and more importantly disfavoured data locality, since the allocation time for the contained members was uncontrolled. Besides, most of the lazily initialised members were requested sooner or later during the program flow and hence neither memory nor CPU time saving was guaranteed under this design. The initial Tracking EDM was designed for charged particles (see Fig. 3), but the requirement of neutral particles in e.g. conversion and V0 finding led to significant code duplication. When extended parameters that add the mass hypothesis for constraint fitting were introduced at the start of Run-1 another copy of the code was done, since the performance of CLHEP with non-fixed size vectors and matrices did not allow for dynamic dimensions in the
track parameterisation. In a reviewed design, the entire track parameterisation was rewritten
with a triple template structure, using the charge, surface type and dimension as template
parameters, see Fig. 4. In total, this reduced the lines of code by more than 90 percent
while keeping the same functionality within the EDM. Standalone tests of constructing track
parameters, determining their type, and retrieving associated covariances showed up to a factor
of three improvements in the new design.

Figure 3. Tracking EDM for charged tracks
during Run-1 based on object inheritance.

3.1.3. Change of magnetic field access
Predicting the track intersection with measurement
surfaces is essential for both track finding and track fitting. Due to the generally non-homogenous
magnetic field, this requires a numerical field integration for the transport of the track parameters
and their associated covariances. In ATLAS, this is done by a Runge-Kutta-Nystroem algorithm,
which has been highly optimised in the past and builds the backbone of the ATLAS track
reconstruction software [14]. The numerical field integration needs repeated access to the
magnetic field, which makes a significant contribution to the total CPU time. The field access
service used during Run-1 was not optimised: at first, a deep caller chain added unnecessary
time in navigating through the library table and the magnetic field was not stored in the native
units used in the field integration and hence unit conversions had to be applied at every call.
In addition, the field service was one of the remaining pieces in the ATLAS software that
had remained in FORTRAN90 and needed a wrapper to be called from the ATLAS offline
software. All of these points have been addressed in a newly designed field service and an
additional caching algorithm has been implemented: the adaptive Runge-Kutta method is a
stepwise numerical integration and for each step the field value has to be requested. However,
since the field is stored within a grid of a certain cell size, it frequently happens that the call
to the field service requests the field within the same or neighbouring cell with respect to the
prior call. A simple caching approach that stores the field parameters from the last call (and the
position and cell dimension of the last call) can therefore reduce the time for the field lookup
since it avoids the navigation into the memory blocks where the apparent field values are stored.

3.2. Algorithmic changes
3.2.1. Track seeding updates
The biggest single CPU consumers of the ID track reconstruction
is the seed building from space points and successive track finding within a road defined by
the seed. The latter needs track propagation through the ID and uses a combinatorial Kalman
filter technique to find all compatible hits following the track hypothesis from the initial seed
parameters. To optimise this very dynamic system, it is of particular interest that the initial seed collection is of high purity. This can be quantified by the probability that a seed leads to an accepted track candidate. Seeds of different composition (PPP, PPS, PSS, SSS) hereby show different purity. The optimal composition of the seed content is not trivial and has to be checked by Monte Carlo (MC) simulation. The performance of the Run-1 track reconstruction [16] and in particular the excellent description of the data by the full detector simulation has led to great confidence that the performance of the pattern recognition modules and their reliability can be studied using MC simulation to find optimal working points. For Run-2, the seed purity was reinvestigated and an additional component has been introduced: since the ID was equipped with a new innermost pixel detector, the Insertable B-Layer [17], seeds formed from three space points are now required to be confirmed by another space point in a different detector layer before the road search in the track finding module starts. In total, these changes yield more than 25 percent speed-up and at the same time increase the track finding efficiency and improve the scaling of the ID track reconstruction with event pile-up. Table 1 shows the comparison between the Run-1 and Run-2 setups in the seed finding strategy: for Run-2 seeds are searched primarily in the strip detector, but requiring a fourth space point confirmation in order to start the track finding process. Then, clusters found and assigned in this first search are removed and a second search is evoked using PPP seeds, again requiring a fourth space point confirmation before starting the track finding process.

<table>
<thead>
<tr>
<th>&lt;μ&gt; / type</th>
<th>PPP</th>
<th>PPS</th>
<th>PSS</th>
<th>SSS</th>
<th>PPP+1</th>
<th>PPS+1</th>
<th>PSS+1</th>
<th>SSS+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>57</td>
<td>26</td>
<td>29</td>
<td>66</td>
<td>79</td>
<td>53</td>
<td>52</td>
</tr>
<tr>
<td>40</td>
<td>17</td>
<td>6</td>
<td>5</td>
<td>35</td>
<td>39</td>
<td>8</td>
<td>16</td>
<td>70</td>
</tr>
</tbody>
</table>

### 3.2.2. Tracking in dense environments

At the end of Run-1, ATLAS successfully deployed a cluster splitting module to identify and split clusters that stem from multiple particles in dense environments: all pixel clusters were tested with a set of artificial neural networks (NNs) to check whether they are compatible with being caused by a single, double or multiple particles. The call to the NNs with its 21 hidden layers is rather CPU costly and not really necessary for the majority of the clusters. For Run-2, the clusters are only investigated for potential splitting if they appear as shared clusters on several track candidates, i.e. there exists a good chance that these actually need to be split. At the same time, when the NNs estimated a high probability that a certain cluster was created by more than one particle, the tight hit requirement for shared hits on tracks is relaxed [18]. In total, this reduces the ID reconstruction time by about 10 percent while at the same time increasing the efficiency to find tracks within dense environments, as shown in Fig. 6.

### 3.2.3. Task optimisation

Additional CPU reduction was achieved by task optimisation. As part of many combined object reconstruction algorithms, tracks are extrapolated to different measurement layers in the calorimeter to allow for track-cluster matching. This is done as part of particle flow, combined muon reconstruction (using the ID and MS information), τ and e/γ reconstruction algorithms. In standard reconstruction jobs, this operation was repeated up to six times without caching the result. In a recent redesign, a common service has been introduced to calculate the extension of both charged and neutral particles from combined reconstruction and reuse the result if a subsequent request is made by another algorithm. The freed CPU cycles are used to calculate the track intersection with the calorimeter more precisely: while
Figure 5. Schematic view of the magnetic field lookup during Runge-Kutta integration with cached cells.

Figure 6. Improvement of the $\tau$ reconstruction efficiency in the $3\pi$ decay after applying the new dense environment tuning.

in the former model only the intersection with a rather crudely defined centre position of the different calorimeter layers was calculated, the new setup yields the entrance and exit position, the centre position within each calorimeter cell and additionally the path length within the cell, which allows for comparison of expected to measured energy deposits.

4. Deployment and Results
In total, the ATLAS software optimisation campaign during the shutdown period yielded a factor 4 speed up of the reconstruction time on a simulated $t\bar{t}$ pair production benchmark sample with an average of 40 pile-up events. The main improvement was achieved by the ID track reconstruction which itself improved by more than a factor of 5 and reduced the relative contribution to the total event reconstruction time from 70 to approximately 50 percent. At the same time, basically all components of the ID reconstruction were improved in terms of the efficiency and quality for physics performance. Figure 7 shows the CPU consumption of reconstructing the benchmark sample starting from the last data taking release from Run-1 to the start-up for Run-2 together with a schematic view of the deployment timeline; besides the technical challenges that were related to the different projects, this LS1 software campaign also marked a difficult planning exercise. Many of the described projects required intervention and changes to the repository at rather large scale and within the manpower and time constraint of the LS1 a parallel integration had to be performed. Where possible, certain projects were encapsulated and deployed using dedicated release builds in parallel to the major release, but most of the projects had a very high overlap and could not be disentangled. As a result, ATLAS was more than half way through LS1 without an executable offline development release. Furthermore, the quantification in speed-up or timing improvements for the single projects is not possible and therefore only a rough estimate can be given.

5. Summary
ATLAS has led a very successful software campaign during the last two years, which reduced the CPU consumption for event reconstruction by a factor of four for a benchmark sample with
pile-up $< \mu > = 40$. This was largely achieved by updates to the ID track reconstruction, which was reduced by a factor of five without loss of output information.

References
[16] The ATLAS Collaboration, Performance of the ATLAS Silicon Pattern Recognition Algorithm in Data and Simulation at $\sqrt{s} = 7$ TeV, ATLAS-CONF-2010-072,

Figure 7. Evolution of the total and fractional ID-only CPU consumption for reconstructing the same benchmark sample from the last data taking release of Run-1 to the start-up release in Run-2. The planning and deployment phases for certain projects are also shown in a schematic way.