Performance of the ATLAS Reconstruction Software with high level of Pileup in 2011

Rolf Seuster, for the ATLAS Collaboration
Max-Planck-Institut für Physik (Werner-Heisenberg-Institut), Föhringer Ring 6, D-80805 München
E-mail: Rolf.Seuster@cern.ch

Abstract. In 2011 the LHC provided excellent data, the integrated luminosity of about $5\,fb^{-1}$ was more than what was expected. The price for this huge data set is the in- and out-of-time pileup, additional soft collisions overlaid on top of the interesting collision. The reconstruction software is very sensitive to these additional particles in the event, as the reconstruction time increases due to increased combinatorics. During the running of the experiment in 2011, several successful changes to the software were made that sped up the reconstruction. Pileup has different effects on the various detector technologies used in ATLAS and a general recipe for all subdetectors is not applicable.

1. Introduction
The large hadron collider, in the following called LHC, produced excellent data in 2011, with unprecedented luminosity. The daily luminosity seen by the ATLAS experiment is shown in the left plot of Fig. 1. A steady increase in luminosity is seen over the course of the year 2011. This resulted in a total data set of ATLAS recorded data of well over $5\,pb^{-1}$, see right plot in Fig. 1.

This paper presents the computational challenges the ATLAS collaboration was facing during

![Figure 1: The daily (left) and the integrated (right) luminosity delivered and recorded by ATLAS in 2011.](image)
the operation in the year 2011, the software developed and utilised for the monitoring of the performance and the procedures to implement the necessary speedups in the software.

2. Short Review of 2011 Data Taking

The price of the enormous data set collected during the year 2011 was a huge computing challenge: much higher levels of pileup than projected in the official road-maps for the LHC luminosity. ‘Pileup’ describes the fact that many soft underlying interactions are hiding below the hard interaction, which usually fires the trigger of this event.

Each of these soft underlying events adds additional particles to the event of interest. Due to finite integration times in the sub detectors, these additional events might even come from neighbouring bunch crossings. As these have different effects on the different sub detectors, these events are subdivided into two classes:

- in-time pileup means that the additional collisions happen during the same bunch crossing as the hard scattering which triggers the event.
- out-of-time pileup means that the additional collisions come from other bunch crossings than the hard scattering which triggers the event. Depending on the technology of the subsystem, a different number of events before or after the bunch crossing of the hard scatter can contribute.

During the 2011 data taking the amount of pileup changed dramatically, shown in Fig. 2 as number of interactions per bunch crossing. After a quick ramp-up at the beginning of the data taking, LHC delivered stable conditions with about 7 interactions per crossing for a large part of 2011. After a break in June, the luminosity and with it the interactions were rising slowly but steadily until about 15 interactions per crossing and reaching a peak of almost 20 at the end of the year. These numbers are averaged over all bunches, single bunches had a interaction rate of almost 25 per crossing, see the left plot in Fig. 2.

A typical pileup interaction adds about 10 to 20 additional tracks to the event, depending on transverse momentum thresholds, quality requirements, etc. Often, the amount of pileup is characterised by the variable $\langle \mu \rangle$ which denotes the average number of pileup events overlayed over the hard scatter. The right plot in Fig. 2 shows a nice candidate of a $Z \rightarrow \mu \mu$ event with 20 reconstructed vertices.
3. The Different Sub Detectors and their Behaviour under Pileup
The ATLAS detector [1] consists of several different detector technologies which behave differently under pileup. The inner detector consists of pixel detectors, a semiconductor tracker (SCT) and transition radiation tracker (TRT). Due to their fine time resolution, these are mostly sensitive to in-time pileup, the effect of out-of-time pileup is very limited. The calorimeter system has longer integration times and is therefore more affected by the out-of-time pileup than the inner detector. The integration times vary from three to many bunch crossings. The hardware and software of these detectors is designed such that the out-of-time pileup averages over many bunch crossings, adding noise to the measurements. The muon system is largely unaffected by the pileup except for neutrons coming from the cavern background. This background is expected to be proportional with the luminosity.

4. Data Reconstruction in 2011
Recorded events are processed promptly at the local dedicated computing center for ATLAS data processing at CERN, in the following named Tier0 [2]. First, about 10% of all data is reconstructed in the express stream, which determines the beam spot and serves as input for the estimation of the data quality. The whole data is processed about 48 hours later, after conditions like beam spot are determined but also hot or damaged channels are masked based on the results from the first processing of the express stream.

Large reprocessing campaigns as well as Monte Carlo production are done on the GRID.

4.1. The Reconstruction Software in 2011
At the beginning of data taking in 2011, ATLAS was using reconstruction software of release version "16.6". This software was with little modifications used until August, when ATLAS switched to release version "17". This new release had improvements mostly in the inner detector software, where a better pixel clustering was implemented as well as several software speedups. Further improvements were a better alignment of several sub detectors as well as an improved treatment of temporary hardware defects like spikes and bursts affecting the data quality. All 2011 data taken until then was reprocessed very fast on the GRID with the new release 17 to form a unique dataset.

A typical reconstruction job with detector data as input has about 300 different reconstruction algorithms with widely varying levels of complexity. Clearly, the key to improving the performance of the software is to identify out of the hundreds of algorithms the ones which take the longest, and in which parts of the code they spend the CPU time. The different approaches ATLAS implemented to do this profiling will be explained in more detail in section 5.

4.2. Frozen Tier0 Policy
The frozen Tier0 policy was implemented to ensure an as uniform data set as possible and ensure that the large Monte Carlo production was done with a consistent reconstruction software as the data. Higher luminosity will require higher prescales for trigger and even during a run the trigger configuration can change to ensure maximum output of high quality data. Such changes are explicitly excluded from the frozen Tier0 policy. The frozen Tier0 policy applies to the reconstruction software and ensures that new code will give identical results when run on the same data or Monte Carlo. This policy has only few exceptions. Allowed changes are for fixing deficiencies in the software as bugs, memory overwrites, floating point exceptions and use of uninitialised values, given that these affect only a very limited number of events. This policy is tested every time the new software is compiled in the nightly builds by several jobs run on each of these nightly builds and comparing the outputs.
This places stringent constrains on the performance improvements which can be applied. Algorithmic changes are mostly ruled out as it usually changes the results. This leaves for the performance improvements of an release having to obey frozen Tier0 policy only little room.

4.3. Different Strategies for Improvements
Several different types of improvements were possible at different stages of an release. Before an release is deployed at Tier0, the frozen Tier0 policy does not apply therefore leaving much more room to improve the software. The release 17 saw several of improvements of the physics output and a lot of work was done to ensure that the CPU consumption was not increased with respect to the previous release. All this was implemented before the frozen Tier0 policy was enforced. Other changes which are usually not frozen Tier0 compatible are quality cuts which reduce the input size of objects entering a reconstruction chain to reducing combinatorics for e.g. tracking algorithms.

In the following discussion, two slightly different definitions for the CPU performance of the software are given. These different definitions are due to the way these numbers are derived. The first definition of CPU time per event describes the true CPU time spent processing an event, in the following called "true CPU time per event". Overhead of job initialisation and other processing steps during a job are not included. This leads to the second definition of CPU time per event, which is derived from the total time a job is running divided by the number of events it has processed, in the following called "total CPU time per event". Both numbers are important and useful. The first definition can be used to estimate true speed improvements due to software changes. The second number is more important for operational planning, because the total time of a job is more important in this respect than the time spent in just the event loop. In the following discussions about the performance of the reconstruction software the focus lies mostly on reducing the true CPU time per event but sometimes the total CPU time is shown.

5. Software Tools and Automated Tests
The problem ATLAS was facing during the 2011 data taking was approached on different levels. Two software profilers were extensively used to analyse test jobs. Apart from that automated performance tests were developed as well as parsing of logfiles from the initial processing of the data at the Tier0 computing center, both published automatically on dedicated web pages.

5.1. valgrind/callgrind
The valgrind [3] tool suite simulates a CPU in software and thus allows for a wide range of software analysis. Well known is the tool memcheck, which detects errors in memory usage such as reading or writing into non-accessable memory locations, using undefined values and memory leaks. This tool is also widely used in ATLAS but not the focus here. For the performance analysis we used extensively the callgrind tool and its graphical profile data visualiser kcachegrind. As the CPU and memory overhead is huge compared to running the same job outside of the profiler, the profiling jobs were running mostly interactively on dedicated machines. To shorten the overall running time, only very few events were processed. The main focus of these profiling jobs was to optimise the event loop, so the callgrind tool was run such that it would start profiling after the first event was fully processed. This was done because in the ATLAS software, much of the job initialisation is delayed until the first event when the exact conditions for this job are known. Due to the memory overhead, the profiling had to be done in a 64 bit version of the software where as the normal processing of the data is normally done in 32 bit to reduce the memory footprint. Due to the overhead from simulating and monitoring in valgrind, a typical reconstruction job running inside callgrind approximately doubles the memory consumption and slows down the job by a factor of 5-10.
5.2. **INTEL VTune Amplifier XE**

Only late in the year 2011, ATLAS also started to use the INTEL Performance analyser VTune Amplifier XE [4]. The huge advantage of this tool over valgrind/callgrind is that it runs with only very little overhead. The measured slowdown compared to a run outside of the profiler was found to be less than 5% and the memory overhead was also much smaller than with valgrind. This allowed ATLAS to run jobs in a very similar configuration as the normal processing of the data jobs, i.e. jobs could run in the 32 bit version, and many events could be processed. Despite the fact that ATLAS used it only towards the end of the 2011 data taking, it helped identifying a piece of code which was using up to almost 10% of the walltime of the job. This was then confirmed with valgrind/callgrind but the percentage was estimated to be only about 4%. After identifying this inefficiency, ATLAS was able to reduce this percentage to well below 1%. The available license restricts the usage of this tool to the CERN site.

5.3. **Nightly Performance Tests**

In order to monitor the performance changes more closely, ATLAS implemented automated tests of the CPU and memory consumption by re-running the same job over new versions of the software produced every night. For the CPU consumption, it was found necessary to run on the same dedicated machine as the normalisation between the different CPU types found in the computing clusters at CERN were difficult to interpret. This restricts the number of jobs for the CPU time measurements to only a few. The jobs measuring the memory consumption are fine grained, the jobs are run in many different configuration with varying parts of the reconstruction switched off. This way, the memory consumption is broken down in detail. In order to get detailed knowledge of the CPU time spent in different domains, detailed fine grained jobs can be run by hand when necessary. The left plot in Fig 3 shows the scaling of the CPU time per event measured on different Monte Carlo samples produced with different levels of pileup. The label "rel16" denotes the release used at the beginning of 2011 for the data processing at Tier0, "rel17" denotes an early version of the software used in the second part of 2011 for data processing. This plot shows the performance of the "rel17" software after implementing an improved inner detector software, but before many speed-up improvements were implemented. It shows results from reconstruction of different Monte Carlo samples generated with different \( \langle \mu \rangle \) of 0, 10, 20 and 40. Overlayed are also events from one run taking in 2011 which had a \( \langle \mu \rangle \) in the range of 5-8. The scaling is different because in this run there was a trigger bias towards events with many pileup events, thus artificially increasing the \( \langle \mu \rangle \). This bias was identified and removed shortly after.

At different levels of pileup different algorithms might dominate the time spent in the event loop. It is therefore crucial to look also at performances at different luminosity points, i.e. at different values for \( \langle \mu \rangle \). Often, it is sufficient to look at Monte Carlo events where the \( \langle \mu \rangle \) is precisely known. Such a scaling of CPU consumption per domain with \( \langle \mu \rangle \) is shown in the right plot of Fig. 3. Similar plots for algorithms per domain exists and were used to identify the biggest CPU consumers.

It has been proven useful to have data points at multiple \( \langle \mu \rangle \) in order to detect at an early stage algorithms which would become dominant at a later time when the accelerator was delivering a higher luminosity. The actual algorithms can then be profiled and improved before a problem manifests itself.

5.4. **Performance numbers from data processing at Tier0**

At the Tier0 computing center ATLAS processes all data shortly after data taking. This accumulates huge amounts of performance data which can also be analysed at various levels of detail. Fig. 4 shows a similar plot as Fig. 3 but now for some data run taken towards the end of 2011. The average value \( \langle \mu \rangle \) was given by the LHC accelerator. The CPU times shown here

---

International Conference on Computing in High Energy and Nuclear Physics 2012 (CHEP2012) IOP Publishing
Journal of Physics: Conference Series 396 (2012) 022043
doi:10.1088/1742-6596/396/2/022043
include the overhead of the job. Clearly visible are the different performances for the different CPU types being in use at the Tier0 center. Included is also a special run taken at the end of 2011 with only a few but very intense bunches. This run showed a much higher in-time pileup than any other run taken in 2011 but it had no out-of-time pileup.

Figure 4: The scaling of the CPU consumption of the Tier0 processing with respect to different levels of pileup. Here, the luminosity is given by the accelerator. Included is one run with very high in-time but no out of time pileup taken towards the end of the data taking in 2011.
5.5. CPU Time in TAG files
The latest addition to the profiling capabilities has been the addition of the CPU time estimate to the TAG files. The TAG files are short root files containing only a few variables like trigger decision, and basic quantities of an event like number of certain objects in an event, the 4-momentum of the hardest of these objects etc. The estimate of the CPU time for an event has to be taken while the event is processed, so the start time is taken by the first algorithm in the sequence of reconstruction algorithm and the stop time is taken while this event is written out to disk. For reconstruction jobs where most time is spent in the reconstruction and less so in the writing of events, this is a approximation accurate to about 0.5% in worse cases when buffers are physically written to disk. Generally it is better than 0.1%, typically when no buffers are written. Having the CPU time in the TAG files allows to easily select event which have a long processing time or search for correlations of CPU time and other event quantities. This shows that the pileup in terms of overlaid collisions is a bad estimator for the CPU time of this event, the number of reconstructed tracks shows a much better relation. This is due to the fact, that track finding is the dominant contributor to the event processing time and its time is dominated by combinatorics, see Fig. 5.

![Figure 5](image)

Figure 5: The correlation between the reconstruction time of an event versus the number of primary vertices found in an event (left) and versus number of tracks (right). At a high number of vertices the efficiency of finding a vertex decreases. The variable containing the real reconstruction time per event was implemented only at the end of data taking in 2011, so here a $t\bar{t}$ Monte Carlo sample is shown, the luminosity follows the observed luminosity in 2011.

6. Results
Once an algorithm has been identified as being slow at a certain $\langle \mu \rangle$ or in a certain configuration it will be profiled and a close investigation starts. A valgrind profile, possibly of a much simplified job disabling most other algorithms that are not needed for this investigation, is often the first step. Once the bottleneck is identified, the code needs to be improved and replaced with a functionally equivalent but faster version. Occasionally, numeric results will slightly change. In such cases, the physics output of large Monte Carlo samples reconstructed with the old and the improved version of the software are compared in detail. Only if these results change marginally (if at all), such an improvement can then be accepted into the release. In should be noted, that most speed improvements found in 2011 did not change the results at all.

The most significant performance changes in the reconstruction software in 2011 were:

- Release 17 introduced a new improved pixel clustering software for the inner detector with no visible affect on the CPU consumption. Additionally, tighter quality cuts on the seeds
into the track finding algorithms reduced CPU consumption and showed an improved scaling under high pileup. More details can be found in the contribution [5] of this conference.

- An accessor to the elements to a CLHEP vector was using an old and slow interface, which was the only one available at the time the software was written. A newer interface with much improved performance was introduced later, but the reconstruction code was not rewritten to use it. Doing so gave an improvement of almost 10% CPU time at intermediate \( \langle \mu \rangle \).

- An improved implementation for a test of hardware defects for a sub detector gained another 5% CPU time at intermediate \( \langle \mu \rangle \).

- An association of a jet to a vertex via tracks found close to the jets was redoing extrapolations for each vertex. Caching results and optimising the algorithm improved the CPU time significantly for this algorithm. This problem would have been significant at high pileup, but the algorithm was improved well before the accelerator delivered such data.

- An access to elements in a large vector was inefficient. This gave a huge speedup in this algorithm with only marginal effect on the CPU time for the whole event loop.

Many further small improvements which were done during the data taking 2011 are not listed here. Further improvements giving better CPU speed and reducing the memory footprint were done for the software which is monitoring the data quality of the subsystems. This software is also run during the reconstruction of the recorded data at Tier0, but is not part of the reconstruction software itself. Its inefficiencies of course have negative effects on the overall performance.

7. Conclusion

Despite the huge increase in luminosity the ATLAS experiment observed during the data taking in 2011, the Tier0 computing center kept up well and reconstructed all incoming data promptly. The CPU time of the reconstruction was steadily decreased due to detailed investigations based on profiling as well as on detailed automated nightly tests. This speed-up also helped ATLAS to produce the huge Monte Carlo samples on the GRID needed for many precision analysis in 2011 and later. The frozen Tier0 policy has been proven to work well and provide homogenous data sets and corresponding Monte Carlo samples.

For 2012 ATLAS continues to closely monitor the performance of the release used at Tier0 as well as the development release.

8. Bibliography

[1] ATLAS Collaboration, J. Inst. 3, S08003 (2008)
[2] Markus Elsing et. al, 2010 J. Phys.: Conf. Ser. 219 072011
[3] see http://www.valgrind.org for manual and documentation
[4] http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/
[5] Christoph Wasicki et al, 2012 "Track and Vertex Reconstruction Strategies in the ATLAS Inner Detector in the High Multiplicity LHC Environment", J. Phys.: Conf. Ser. At press.