Optimization of the ATLAS beamspot measurement

Kyle Spurgeon

July 28th, 2014

Abstract:

The ATLAS experiment uses a large variety of technological subsystems to track and reconstruct the paths of charged particles from a proton-proton collision. The basis of this path reconstruction is based in the Inner Detector, where the tracking and vertexing algorithms occur. Every charged particle's path is measured in relation to the “beamspot” measurement, which is the location of the original proton-proton interaction. The Argonne ATLAS Support Center has taken measures to optimize the measurement of the beamspot to better suit the next installment of the ATLAS experiment, where there is expected to be an increase of events occurring in the detector on the order of 3 times previous experiments.

I. **Introduction**

The objective of the optimization project was to take the current parameters and equipment being used in the ATLAS detector at Cern and modify them to optimize the results for the next installment of operation at the Large Hadron Collider (LHC). The upcoming installment of operation includes an increase in the operating energy of the collider, from seven Terra-electronvolts (TeV) to fourteen TeV, and accompanying this increase in energy is an increase in the number of events, or collisions, occurring within the ATLAS detector. This increase in events will lead to an increase in collisions, tracks, vertices, and pileup of particles. Due to all of these increases, it is necessary to revise current tracking methods and optimize them for this increase in activity.

Previously, the detector saw $<\mu>\approx 19$, where $<\mu>$ is the average number of interactions, and the measurement of the beamspot, which refers to the point of interaction between the two proton beams, had been optimized for $<\mu>=1$. This number is expected to be closer to fifty interactions per crossing in the upcoming phase. With such an increase in interactions, the number of vertices and tracks will increase on the same scale, so the measurement of the beamspot will need to be as precise as possible to allow for the best possible resolution of the many particles present in the detector. The relation between beamspot resolution and overall particle resolution is directly related, the measurement of subsequent vertices will be only as precise as the original beamspot measurement. This is especially important for secondary vertices, or vertices that are located extremely close to a larger vertex.

The optimization of the beamspot is crucial to all experiments taking place at ATLAS. This measurement affects all subsequent measurements, and therefore it is important that this measurement be as accurate as possible to allow for the best possible results. The accuracy of the beamspot influences how other particles are seen, and as such a mis-optimization could lead to unfavorable experimental results for any of the experiments at the ATLAS detector. The major influences and tools for the optimization are introduced below.

A. The ATLAS Detector

The ATLAS detector is one of four detectors currently collecting data at the Large Hadron Clooider (LHC) (for more information, see http://home.web.cern.ch/topics/large-hadron-collider). The ATLAS detector is a general, in that it does not specialize in any specific types of particles, detector used for discovery and measurement of particles. The ATLAS detector is composed of three separate detecting sections: the Inner Detector (ID), the Calorimeters, and the Muon Spectrometer. This project is most concerned with the ID, so I will explain it in most detail last. The Calorimeters measure the energies of charged and neutral particles. They are made of metal plates and sensing elements, which are liquid argon in the interior and scintillating plastic on the exterior. Particles that reach the calorimeters create “showers” of particles, whose energies are then measured.

The Muon Spectrometer, the outer most portion of the detector, is located there due to the properties of muons. Muons interact very little with the other parts of the detector, essentially passing through the ID and the Calorimeters unaffected, and only interacting in the final stage, the spectrometer. The Spectrometer consists of thousands of charged particle sensors placed in the magnetic field which detect the momenta of Muons to very high precision levels.

The ID sits at the core of the detector, directly around the beam pipe, the vacuum pipe in which the protons travel and collide. The inner detector is composed of three technologies, with a fourth the be added for Run-II. The first, and most precise, layer is the pixel layer. The pixel layer contains three cylindrical layers of silicon detectors with alternating orientations. These inner most detectors withstand up to three-hundred kGy of radiation, and thus must be both very sturdy as well as highly accurate. The three cylindrical layers are accompanied by an equal number of disk layers located at both ends of the cylinders, to allow for pixel detection of particles traveling at large angles with respect

to the perpendicular. The second technology of the ID is the Semiconductor Tracker (SCT). This is laid out identically to the pixel layer, with the exception that it has four cylindrical layers and ten disk layers. The cylindrical SCT layer is also composed of silicon detectors, though of a lower resolution than the pixel layer.

The final layer of the ID is the Transition Radiation Tracker (TRT), which is composed of straw detectors which run parallel to the beam line. The TRT incorporates xenon gas to detect photons which are formed by electrons interacting with the material between the individual straws. A full diagram of the ATLAS Inner Detector can be seen in figure one, while figures two and three contain diagrams of the Pixel layer and a combination of all three layers. All three layers of the ID are under the influence of a high-power magnetic field, which influences any charged particles present in the detector.

The primary purpose of the ID is to detect the momenta of the particles given off in a collision, which is done by collecting a "hit" at every point that is triggered on each of the layers of the detector. These hits are then grouped using a long tracking algorithm, which will be discussed in some detail later, until the track that one particle takes is discovered. Every point is utilized in this way until a multitude of tracks are gathered and all points have been included or rejected. These tracks are then utilized by a vertexing algorithm, to be discussed in some detail later. These vertices and tracks are then used to find the curvature of the path taken out of the ID to formulate the momenta of the individual particles.

B. Tracking and vertexing

Both tracking and vertexing work in a similar fashion, here I will explain tracking first, and expand off of this process to explain vertexing. Tracking starts in the way described above, you have a large quantities of detector hits which need to be matched to a track. There are an initial set of parameters that are utilized so that the tracking algorithm only outputs what is classified as good tracks. These parameters are, at the basic level, simply a necessary number of hits in each layer that would eliminate any false or bad tracks. Using these parameters as well as pattern recognition, the algorithm starts to reconstruct tracks.

The pattern recognition program starts with the innermost hit, and then views a small window of the next layer of detectors and looks for a hit there, if found this process is repeated with increasingly smaller windows until the end is reached. If a point is not found in the next layer, the original point is abandoned and the algorithm moves on. The algorithm then removes the point that is farthest off of this ideal pattern fit. If the fit gets better, that point is forever removed from that track, however if the fit gets worse, they replace that point and move on. Once this is done with all points, the tracks that do not meet the original parameters are eliminated leaving only good tracks.

The vertexing algorithm takes over at this point. It analyzes the tracks and traces them back to where they intercept with one another. At this point, it behaves in the same manner as the tracking algorithm. The vertexing algorithm uses a cluster finding method to locate a vertex. It looks for tracks that have a very small difference in coordinates close to the center, and larger distances further from the center. Once grouped by these distances, the algorithm moves on to the next step. It looks at all of the tracks that point to one intersection spot, and, once these are isolated, it removes every other track. If the vertex gets better, then the tracks stay removed, if it gets worse then they return to the fit and the algorithm moves on. This process is repeated for all vertices.

The vertexing algorithm also has the task of finding secondary vertices, or vertices located very close to a larger, or primary, vertex, which originate from a decay or secondary interaction. These are found by taking a large amount of removed tracks from a vertex, or simply analyzing a vertex with unusual fits, and analyzing them to find whether they are genuinely bad tracks, or simply originating from a point very close to the primary vertex that was just analyzed. If it is found that a secondary

vertex is present, it filters the tracks in the way described above, finalizing the vertex. It is this secondary vertexing process that relies heavily on the optimization of the beamspot, without this optimization, these vertices may not be seen.

C. Software resources

The first major software resource used in the optimization project is Root. Root is the environment that hosts, creates, and manipulates histograms, which are the graphical output for much of the work done in ATLAS. This environment is used to browse, graph, fit, and in general manipulate these output histograms. The depth and possible uses of Root are vast, but for this project the main use is to fit the Histograms with a double-Gaussian curve and look at all, and mainly focus on one, chi squared over ndf, of the output characteristics, and find the right settings that output the most optimal numbers.

The second software resource is Athena. Athena is a software package which is used to run ATLAS specific code and simulations. Athena is host to a plethora of uses, and it is instrumental throughout much of the ATLAS experiment. In this case, Athena was used to run the trigger simulations on Monte Carlo, or simulation, and real data to output the beamspot measurements, so that the resolution in various settings could be measured.

II. **Experiment**

The first step in the process is to run the ATLAS High-Level Trigger (HLT) simulation. This was also the most challenging portion of the experiment, so the details for this portion will be described last. The second task was to develop a program to fit the outputs from the simulation processes and output the necessary characteristics to analyze and compare the various settings. This task was achieved with a fairly simple program developed using C++. The program utilizes a double-Gaussian fitting and a for loop. The for loop is used to access multiple data sets, for example the x, y, and z components of the beamspot measurement plot them on a single output window, having fit all of the chosen histograms with the double-Gaussian function.

This program was tested on plots from a previous analysis and, once deemed successful, the project moved towards the next goal. Still encountering issues running the HLT simulation, debugging the middle step in the process was the next obvious step. Utilization of a tool present in the trigger and data acquisition (tdaq) setup called “beamSpotTool.py” was deemed necessary to transform the output from the simulation into data pertaining to the beamspot. The tdaq system is a local cluster of computers simulating a similar device located at CERN, which is used to collect and output data from the collisions To run this tool, there had to be a connection between the local tdaq cluster and the users working space on the ATLAS cluster.

This connection was created only to cause an issue with the associated versions of Root. The current users system was utilizing a shared version of Root, which proved incompatible with beamSpotTool.py on the tdaq system. To remove this issue, the local shared version of Root was replaced. After these initial issue were resolved, an error arose with a python package “numpy”. This package was expected by the local team as well as ATLAS experts to be included in the tdaq software release, yet there was no success in getting the package to run with beamSpotTool.py. There were attempts made to obtain this package from ATLAS's global tdaq software release, and once obtained there more errors encountered of various origins.

Looking into the code, the uses of the “numpy” package where found, which included only finding a set of extrema for the histogram to decide if the histogram was going to be rebinned. This task was deemed unnecessary, and any call for the “numpy” package where removed from the code. In doing this, the beamSpotTool.py quit outputting errors and thus the second portion of the experiment

was successful. Later, focus returned to this project to rewrite the problematic code without utilizing the numpy package. This was successfully completed and has since been sent to the ATLAS collaboration. The focus now returned to running the HLT simulation.

In the beginning, the goal was to run the HLT simulation without utilizing Athena, as to decrease the overall complexity of the project. This was attempted for a short time, as it became evident very quickly that this process was going to be unsuccessful. This idea was then replaced with running with an old Athena release that had a reliable set of menus, and with which it was known that the beamspot measurements could be made. The version 18.1.0 was chosen, due partly to its compatibility with the chosen tool and also partly its age. To run the simulation in Athena, a setup file was created to find and access the correct version, and the tool chosen to run the simulation was "runHLT_standalone.py", a preconfigured tool.

To run "runHLT_standalone.py", a command file was created that simply called the data we wished to utilize in the simulation with the simulation command. Once shown successful, this command file was then modified to run the beamspot triggers, which would create the necessary output to optimize the beamspot when the settings were altered. This was the point where issues began to appear. Adding these triggers failed to alter the output of the simulation. To find the cause of this, many simulations were run including various commands to call these triggers. When all failed, it was attempted to find the overall cause of the inability to call the beamspot triggers. The seeding was changed in an attempt to access them, but none of the attempted seed changes altered the output.

At this point, help from experts was sought and, essentially, it was said that the Athena releases that were being worked with were too old and that, for most efficiency, it would have to be updated to the most current versions, and that a new environment "devval" should be utilized. The environment includes software packages that are updated on a daily basis, so it can be difficult to work with, which is the main reason it was originally avoided. The second issue with the devval environment was that the beamspot triggers had yet to be implemented, so some one would locally have to attempt to implement these in the correct manner, made more difficult by a recent change in convention for these types of implementation.

With much work from an experienced user, the beamspot chains were properly inserted into the environment in such a way that the errors that occurred when improperly implemented disappeared, but the simulation still appears unable to actually output the necessary beamspot information. This is due to a change in convention for the new software package and will require the local users to implement the chains themselves, which will be a time consuming process.

III. **Results and conclusions**

Currently, the project has produced no physics results, with progress still needing to be made concerning the simulation. Though this is the case, strides have been made with setting up the simulation and beamspot tools on the local ATLAS and tdaq clusters. The ability to run the ATLAS trigger simulations was a major step for the project and the issues running the beamspot triggers were unforeseen. Help from ATLAS experts has been requested and the issues are being resolved. There is an attempt at continuing the project past the end of the SULI program, in hopes to complete the original goal before the end of the year.

The project has not reached the point that was originally intended for the SULI program deadline, but, considering the multiple setbacks, it has not been unsuccessful. As stated previously, the ability to run the simulation locally is a major step forward and will make the continuation of the project in the future much simpler. This, paired with the satisfactory performance of the beamspottool and the fitting program will allow for quick analysis of the results once the beamspot chains are implemented in the future.

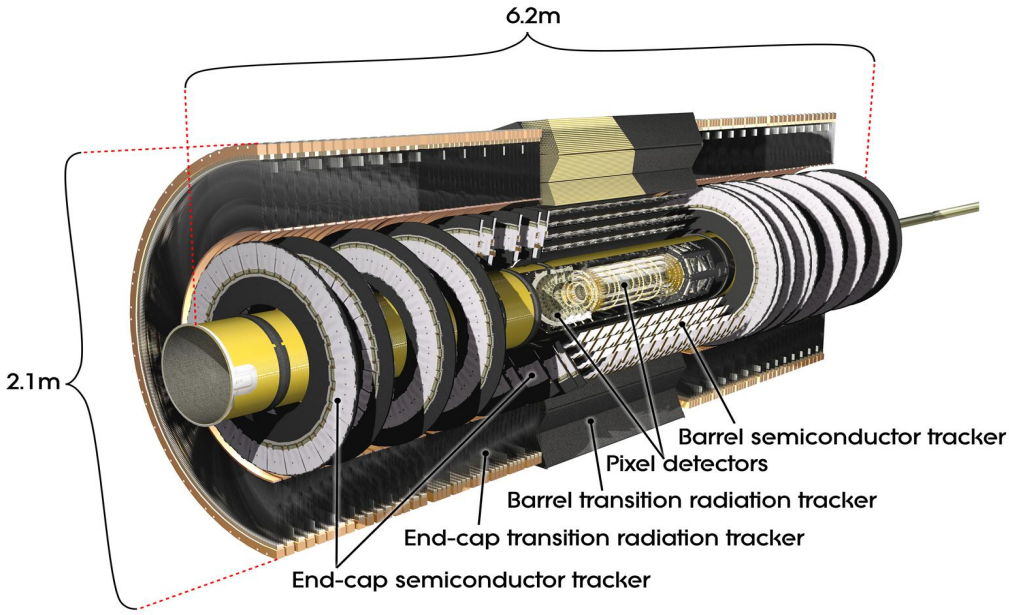

Figure 1: Above is a computer rendered cross sectional model of the ATLAS inner detector.

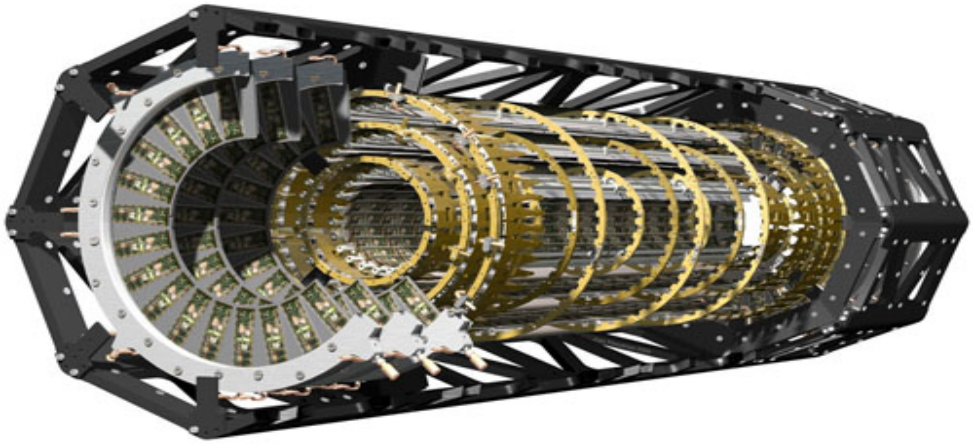

Figure 2: Shown above is a computer rendered model of the pixel detector layer from the ATLAS detector.

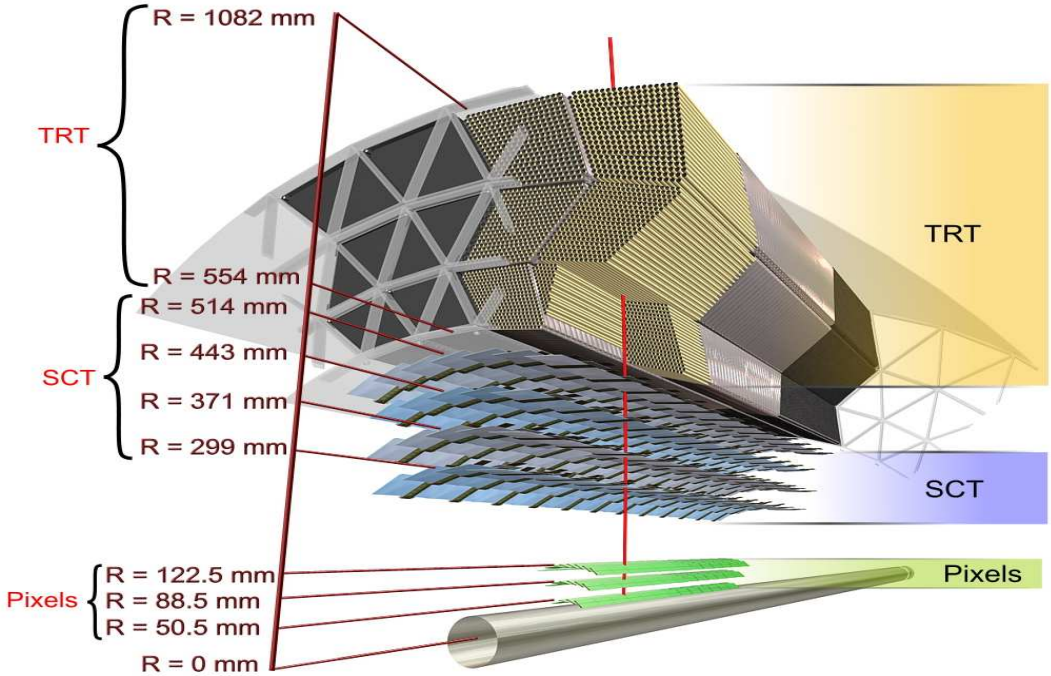

Figure 3: The above graphic shows the positions, sizes, and placements of each of the three technologies in the inner detector in relation to each other.