Performance Improvements for the ATLAS Detector Simulation Framework

August 2013

Author:
Yasin Almalioglu

Supervisors:
Andreas Salzburger
Elmar Ritsch

CERN openlab Summer Student Report 2013

CERN openlab

Abstract

Many physics and performance studies carried out with the ATLAS detector at the Long Hadron Collider (LHC) require very large event samples. A detailed simulation for the detector, however, requires a great amount of CPU resources. In addition to detailed simulation, fast techniques and new setups are developed and extensively used to supply large event samples. In addition to the development of new techniques and setups, it is still possible to find some performance improvements in the existing simulation technologies.

This work shows some possible ways to increase the performance for different full and fast ATLAS detector simulation setups, using new libraries and code improvements in the ATLAS detector simulation framework. Besides of the improvements, measured time consumptions of different setups are shown and possible further improvements are the other main focuses of this project.

Table of Contents

Performance Improvements for the ATLAS Detector Simulation Framework.....1

Abstract.....2

1 Introduction.....4

2 ATLAS Experiment.....6

2.1 Inner Detector.....7

2.2 Calorimeter System.....8

2.3 Muon Spectrometer.....8

3 Simulation Strategies in ATLAS.....9

3.1 Event Generation.....10

3.2 Physics and Detector Response.....10

3.2.1 Geant4.....11

3.2.2 ATLFASII.....11

3.2.3 ATLFASTIIF and Fatras.....12

3.3 Digitization.....12

3.4 Reconstruction.....12

4 Test Results and Performance Improvements.....12

5 Summary.....17

6 Reference.....18

7 APPENDIX.....19

7.1 ATLFASTIIF CPU Profiling Results.....19

7.2 ATLFASTII CPU Profiling Results.....21

1 Introduction

Particle physics is one of the fundamental approaches to be able to understand the laws of the smallest parts of the matter, called *particles*. Several particle physics experiments have been organised to have a deeper understanding of the particle world.

Between 1998 and 2008, the LHC[1] was built to reach higher energies in particle collisions. It is the highest-energy (14TeV) particle collider ever built so far. It has a circumference of 27 km and lies 100 m underground. Protons are first accelerated in a linear accelerator and are subsequently carried to the LHC via a chain of circular accelerators. Each proton beam consists of ~3000 bunches and each bunch contains ~10^{11} protons. Then, they are collided in four different experiment points. To keep the beams on track, ~1200 superconducting dipole and ~400 quadruple magnets producing an 8.33 Tesla magnetic field are used.

There are four main experiments on the LHC. ATLAS[2] and CMS[3] are the two largest symmetric general-purpose particle detectors. Furthermore, the other two non-symmetric detectors are LHCb[4] and ALICE[5]. The LHCb experiment mainly investigates the properties of b-quarks and CP Violation. The focus of ALICE experiment is to measure the collision of heavy ions and is to analyse behaviour of quark gluon plasmas.

All the four detectors and the LHC were successfully constructed and the first beam was circulated in 2008. However, they do not function at full capacity, upgrades are still going on to catch the maximum centre of mass energy of 14 TeV for proton-proton collisions. A schematic view of LHC and how the experiments are located are shown in Figure 1.

Modern particle physics relies as much on computer simulations as on the physical experiments. Before the construction of the detectors, it has to be ensured that high quality physics studies are feasible with the intended detector design. That is to say,

[1] **LHC**: **L**ong **H**adron **C**ollider
[2] **ATLAS**: **A** **T**oroidal **L**HC **A**pparatu**S**
[3] **CMS**: **C**ompact **M**uon **S**olenoid
[4] **LHCb**: **L**arge **H**adron **C**ollider **b**eauty
[5] **ALICE**: **A** **L**arge **I**on **C**ollider **E**xperiment

particle identification and measurements will be done at high efficiencies and with high precision. Detector simulation plays an important role to assure that all the requirements will be met. Moreover, complex particle interaction processes with the detector material make the calculations by hand impossible. As a particle traverses the detector, it interacts with the detector material in many ways and it may decay into other particles. Thus, the detector output is non-trivial and software tools are used to simulate the most relevant processes and their impacts on the measurements.

To allow for highly accurate results in physics studies, very large simulated collision event samples are required. The traditional way to simulate particles traversing the detector is a full, very accurate simulation which models the particle interactions with the detector material, taking into account various material and particle properties. This simulation approach is used, regardless of whether collision or cosmic ray events are simulated. Since this full detector simulation takes time in the order of minutes per event, it is necessary to investigate faster and feasible ways to generate billions of simulated events per year.

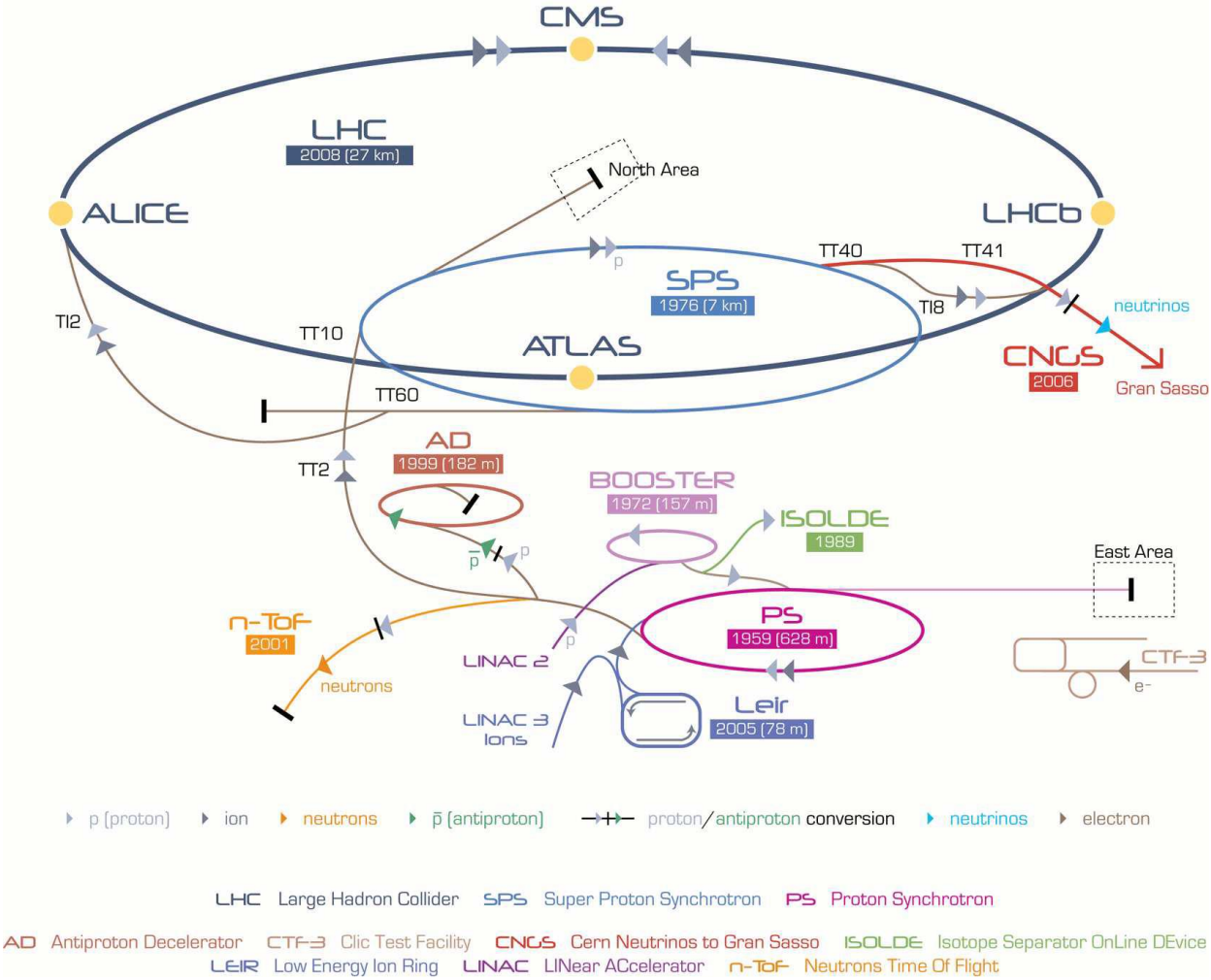


Figure 1 A schematic view of LHC experiments and the CERN accelerator complex [1]

2 ATLAS Experiment

The ATLAS [1] detector is the biggest particle detector ever constructed. As of today, the collaboration consists of more than 3000 scientists. The detector measures 25m in height and 44m in length and weighting about 7000 tons. It is built to measure the particles produced by proton-proton collisions of up to 14 TeV centre of mass energy as well as heavy-ion collisions up to 5.5 TeV. The purpose has been to verify the existing Standard Model physics and search new phenomena in particle physics as well as other physics scenarios like Supersymmetry and "exotic" particles. The detector consists of

three sub systems: Inner detector, calorimeters and muon system. The cross-section of the detector and main parts of it are shown in Figure 2. Each detector sub-system has its own purpose and functionality. When combining the information gathered by the individual systems, a complex picture of the collision event is obtained.

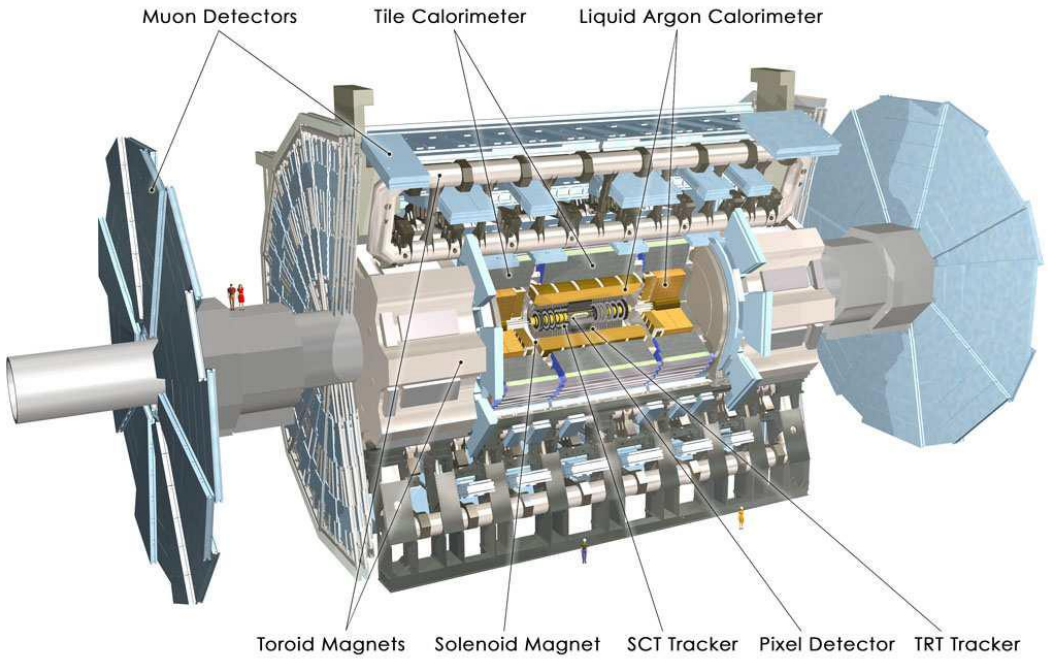


Figure 2 A cut away view of ATLAS detector measuring 25m in height, 45m in length and weighting about 7000 tons.

2.1 Inner Detector

The inner detector is the sub-detector closest to the collision point and, thus, the inner-most part of the ATLAS detector. The inner detector is a tracking detector, consisting of three components: three layers of silicon pixel detectors providing three space points per charged particle, four stereo layers of silicon strip detectors (SCT) providing four-space points and a transition radiation tracker (TRT) providing 36 hits and additional electron – pion separation capabilities.

2.2 Calorimeter System

After the particles have passed through the inner detector, they enter the calorimeter system. The main purpose of the calorimeter system is to measure the energies of the particles. It is composed of two sub-calorimeters. The electromagnetic calorimeter is responsible for measuring photon and electron energies, while the hadronic calorimeter measures energies of hadrons. In order to measure these energies, incident particles are forced stop inside the calorimeter. Thus, they are provoked to cause a particle shower such that energy deposited by the shower can be measured. As different showers shapes are caused by different type of particles, the shower shapes can be used to determine the particle types. Muons, however, only interact very weakly with the calorimeter much so that they easily pass through the calorimeter system.

2.3 Muon Spectrometer

The muon spectrometer is the outer-most part of the ATLAS detector and has the most significant signatures for some of the most interesting processes at the LHC. Like the inner detector, the muon system is responsible for particle tracking, whereas only muons will reach the muon spectrometer. Throughout the muon system, a 0.3 T magnetic field is applied to measure each particle's charge over momentum ratio. Combining this to that of inner detector, very accurate muon momentum measurements are obtained.

As mentioned above, each particle leaves a distinct signature in the detector. Figure 3 shows a few examples of particle signatures as they appear in the ATLAS detector.

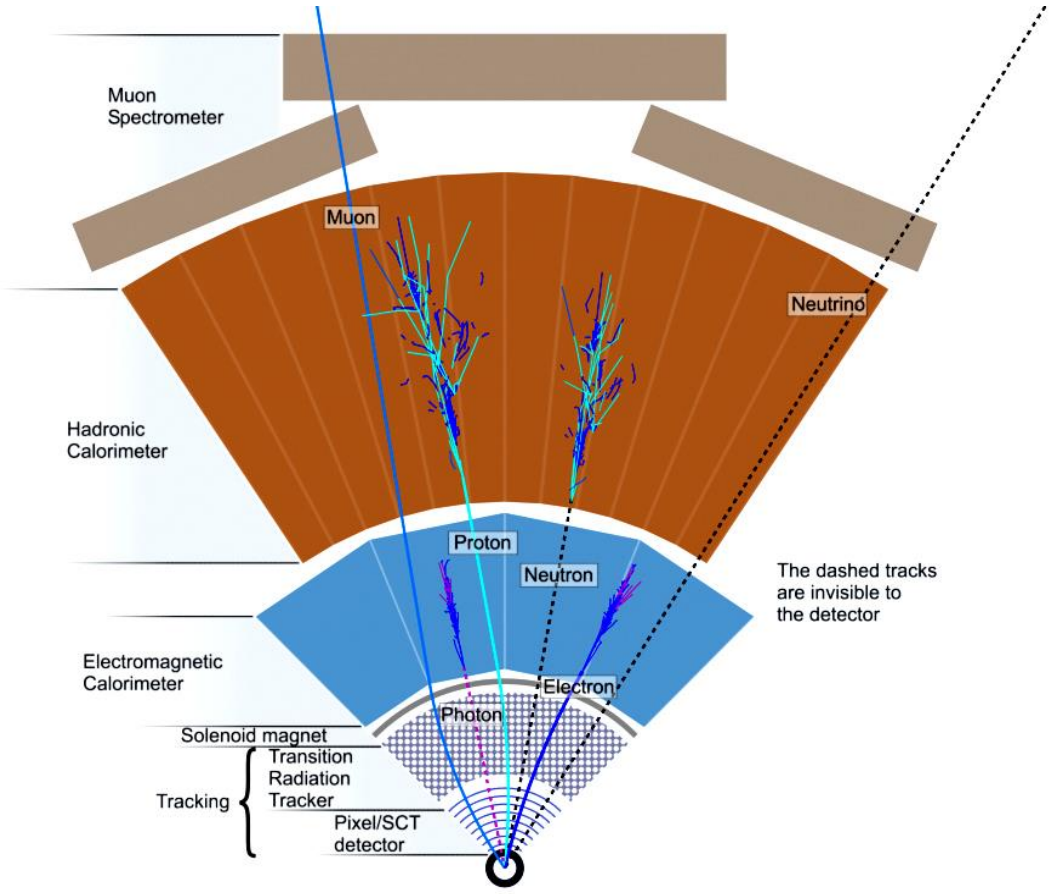


Figure 3 Particle signatures of particles traversing the ATLAS detector. The distinct signatures of different particle types allow for efficient particle identification with the ATLAS detector.

3 Simulation Strategies in ATLAS

Detector simulation is required to qualify the detector response (and its uncertainties) to particles stemming from proton-proton collisions. Due to in-time and out-of-time pileup, measurements of signal events are overlaid with background processes. Thus, both background and signal particles are simulated in the detector. During the design phase of the ATLAS experiment, detector simulation played a major role in finding the most suitable detector layout for physics studies at the LHC. Besides, in order to prepare

physics analyses and software tools prior to recording data, a great number of simulated event samples was necessary.

The modelling of detector responses and the simulation of physics events with Monte Carlo methods is a basic technique in particle and high energy physics. The main input for the testing and validating the performance of the reconstruction software is simulated data as most of the tools and software are developed during the construction of the detector.

In order to compare recorded and simulated data, after some point, it necessary to have a common output format. However, this comparison is valid only on a statistical basis for significant distributions due to the intrinsic randomness of the Monte Carlo method.

There are four steps to produce Monte Carlo event samples for the ATLAS experiment.

3.1 Event Generation

The physics event generator is the basis for particle simulation. The events consist of stable particles with specified creation point in space and time, four momenta and particle type. Stable particles from a single proton-proton interaction at the vertex (0,0,0) are computed on the theoretical basis. These events are either directly passed to the detector simulation or written into a file.

3.2 Physics and Detector Response

The next step in producing the Monte Carlo events is to simulate interactions of the particles with the detector. In a fast manner, this is done by modelling the energy depositions of particles inside the detector. Yet, in a detailed manner, further on called *full simulation*, the particles are processed through the detector and the most significant interactions with the detector are emulated. Energy depositions in sensitive detector elements, further on called *hit*s, are saved to a file for further processing in the digitization step (see section 3.3). In the ATLAS experiment, there are three detector simulation methods available. All three of them are analysed and tested on this work: Geant4, ATLFASTII, and ATLFASTIIF. These approaches can be categorized into

two groups: Geant4 in full simulation; ATLFASTII and ATLFASTIIF are in fast simulation.

3.2.1 Geant4

Geant4 is a generic toolkit to simulate particles traversing matter. It can simulate different types of particle interactions with different materials over a very wide variety. Its main applications are found to be several fields such as medical science, high energy physics and astrophysics.

Due to very high precision capacity of Geant4, the detailed full simulation in ATLAS is based on Geant4. Geometrical description and the desired accuracy parameters are passed to Geant4 for particle simulation. While it provides very accurate results, the drawback for that is the immense computational power and a great amount of time required to simulate collision events in the ATLAS detector. Due to this disadvantage, it is not feasible to use Geant4 for the cases that require great amount of event samples. Thus, it is necessary to seek for faster ways of doing this job. The relative Geant4 simulation time contribution per ATLAS sub-detector is shown in Figure 4.

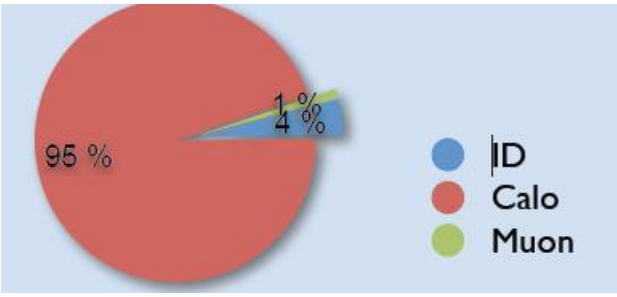


Figure 4 Relative Geant4 simulation time per sub-detector [2].

3.2.2 ATLFASII

About 95% of the overall detector simulation time is spent in the calorimeter simulation in Geant4. Thus, a fast calorimeter simulator (FastCaloSim [3]), which is a parameterised simulation based on Geant4, is developed to reduce total time. ATLFASTII is a fast ATLAS detector simulator which uses FastCaloSim for the calorimeter simulation and Geant4 for the inner detector and muon spectrometer. This way, overall job time is reduced by the factor of 10.

Muons are the only particles simulated by Geant4 outside of the inner detector. The rest of the particles leaving the inner detector are simulated by FastCaloSim. Thus, the only particles that can pass to muon spectrometer in ATLFASTII are muons.

3.2.3 ATLFASTIIF and Fatras

ATLFASTIIF is a fast simulation for ATLAS detector. It is an upgrade to ATLFASTII in a sense that ATLFASTIIF uses the fast ATLAS Tracking Simulation (Fatras [4]) for the inner detector and muon spectrometer instead of Geant4. With ATLFASTIIF, a factor of ~100 in CPU time is gained compared to full Geant4 detector simulation. Fatras uses the simplified reconstruction geometry and physics parameterizations so that an additional factor of ~10 is obtained compared to full Geant4 simulation.

3.3 Digitization

The digitization software converts the simulated energy deposits inside sensitive detector material into the detector response. Hence, the next step in the simulation chain is to simulate the detector response and obtain an output format, comparable to data retrieved from the detector. This includes conversion of the previously computed energy deposits into measurable quantities. The output format is RDO (RAW Data Objects). Thus, the steps after this point are not specific to the simulation.

3.4 Reconstruction

After the RDO output format is produced, the next step both in the simulation chain and real detector data chain is to interpret the file. Different algorithms are used to trace the particles, to measure momentum and energy and to identify individual particles.

4 Test Results and Performance Improvements

Different setups and approaches for detector simulation in the ATLAS experiment explained in the section 3.2. Each of these setups takes different amount of time for the

simulation. In order to test and improve these different setups in terms of CPU time consumption, there are several practical techniques.

The first technique is to measure simulation time per event. When the main concern is time consumption of different setups, this technique is very feasible to apply. Another technique that can be employed in performance tests is to use CPU profiling. The CPU profiling technique that was used in this project takes samples of the call stack of the program within the specified time intervals and, thus, gives sufficiently enough results for average CPU time consumptions of the different functions called by the program. With the profiling, it is easier to analyse which part of the program takes most of the time and, hence, to find possible candidate functions to be improved. KCachegrind [5] is a visualization tool for the profiling data, which is used in this work. Each colored rectangle represents a function; its size tries to be proportional to the cost spent therein while the active function is running. The profiling tool catches all memory accesses for the trace. The trace includes the number of instruction/data memory accesses and 1st/2nd level cache misses, and relates it to source lines and functions of the run program. Thus, the numbers just under the percentages in the profiling diagrams give us a good approximate values proportional to the exact run time of functions.

In order to test and improve the simulation, the standard mathematics library is replaced by the Intel® Math Kernel Library and an improved ATLAS magnetic field interpolation is used. Test results for time per event statistics are shown in the following table and CPU profiling diagrams are in the appendix.

	Geant4 **(7 Events)** **(Average ±Standard Deviation) /(Minimum)/ (Maximum) per event in sec.**	**ATLFASTII** **(80 events)** **(Average ±Standard Deviation) /(Minimum)/ (Maximum) per event in** **sec.**	**ATLFASTIIF** **(500 Events)** **(Average ±Standard Deviation) /(Minimum)/ (Maximum) per event in sec.**
Standard	**308(± 146)** / 151 / 628	**33.9(± 11.8)** / 9.77 / 67.6	**1.45(± 0.452)** / 0.503 / 3.3
Intel® Math Kernel Library 10.3 update 2	**297(± 128)** / 153 / 572	**29.6(± 10.4)** / 11.2 / 64.2	**1.28(± 0.395)** / 0.429 / 2.76
Intel® Math Kernel Library 11.0 update 3	**274(± 115)** / 136 / 514	**29.4(± 10.4)** / 10.2 / 66.8	**1.18(± 0.361)** / 0.423 / 2.67
Intel® Math Kernel Library 11.0 update 3 + Improved ATLAS Magnetic Field Interpolation	**270(± 115)** / 142 / 512	- (Unable to test due to temporary incompatibilities)	**1.26(± 0.388)** / 0.442 / 2.84

All values are (Average ± Standard Deviation)/(Minimum)/(Maximum) per event in seconds, tested in the ATLAS offline software release 17.7.X.Y with ttbar samples and geometry version ATLAS-GEO-20-00-01.

Another way to improve execution performance is based on a deeper understanding of low-level CPU features such as instruction level parallelism. In the programming, branch prediction strategy plays an important role. The *if-else statements* in a code are one of the important parts of the branch prediction. In the CPU, multiple instructions are pipelined. If the next instructions that will be issued can be predicted correctly, no instruction would be wasted. That is to say, the better a branch is predicted, the more efficient the CPU is utilized. A mechanism to predict a branch is to count how many times a branch is taken or not. Thus, reducing the number of branches caused by the if-else statements is important in terms of CPU time. On this work, thanks to the CPU profiling explained previously, the *straightLineDistanceEstimate()* function is chosen and, then, improved, using the branch prediction strategy. The code before and after the modification is as shown:

```
0209   if(A==0.) {                      // direction parallel to cylinder axis
0210     if ( fabs(currDist) < tol ) {
0211       return Trk::DistanceSolution(1,0.,true,0.);   // solution at surface
0212     } else {
0213       return Trk::DistanceSolution(0,currDist,true,0.); //point of closest approach without intersection
0214     }
0215   }
0216
0217   // minimal distance to cylinder axis
0218   double rmin = C - B*B/A < 0. ? 0. :  sqrt(C - B*B/A);
0219
0220   if ( rmin > radius ) {          // no intersection
0221     double first = B/A;
0222     return Trk::DistanceSolution(0,currDist,true,first); //point of closest approach without intersection
0223   } else {
0224     if ( fabs(rmin - radius) < tol ) {     // tangential 'intersection' - return double solution
0225       double first = B/A;
0226       return Trk::DistanceSolution(2,currDist,true,first,first);
0227     } else {
0228       double first  = B/A - sqrt((radius-rmin)*(radius+rmin)/A);
0229       double second = B/A + sqrt((radius-rmin)*(radius+rmin)/A);
0230       if ( first >= 0. ) {
0231         return Trk::DistanceSolution(2,currDist,true,first,second);
0232       } else if ( second <= 0. ) {
0233         return Trk::DistanceSolution(2,currDist,true,second,first);
0234       } else {      // inside cylinder
0235         return Trk::DistanceSolution(2,currDist,true,second,first);
0236       }
0237     }
0238   }
0239 }
```

Figure 5 Unmodified part of the straightLineDistanceEstimate function in CylinderSurface.cxx [6] code

```
//just tools for quick calculation
  double oneOvera = 1./A;
  double bOvera = B*oneOvera;

  //Arguments of DistanceSolution function
  double first = bOvera;
  double second = bOvera;
  int num=0;

  if(A!=0.) {
      double temp2 = C - B*bOvera; //just used in the following
      double rmin = temp2 < 0. ? 0. : sqrt(temp2); // minimal distance to cylinder axis
      double rDiff = rmin-radius;

      if( rDiff > 0. ){ // no intersection
      second=0.; // point of closest approach without intersection
      }else if( fabs(rDiff) >= tol ){   // tangential 'intersection' - return double solution
      double mySqrt = sqrt( (-1*rDiff)*(radius+rmin)*oneOvera );
      second -= mySqrt;
      first += mySqrt;
      num = 2;
      if(second >= 0.){
          double temp = first;
          first = second;
          second = temp;
      }
      }
  }else{    // direction parallel to cylinder axis
      first = 0.;
      second = 0.;
      if( fabs(currDist) < tol ){
      num=1;
      currDist = 0.;
      }
  }

  return Trk::DistanceSolution(num, currDist, true, first, second);
}
```

Figure 6 Improved part of the straightLineDistanceEstimate() function in CylinderSurface.cxx [5] code

After the improvement, a 1% overall performance gain is achieved. Reduced actual CPU time consumption from 7.44%, in Figure 7, to 6.52%, in Figure 8 ;

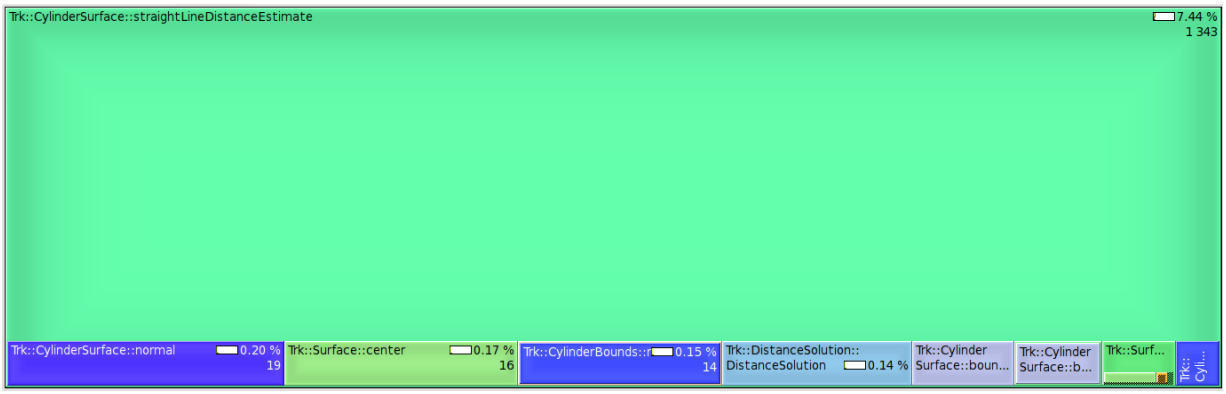


Figure 7 Actual CPU time consumption of unmodified straightLineDistanceEstimate() shown in the CPU profiling software with ATLFASTIIF detector simulation setup, ATLAS offline software release: 17.6.0.5 & ttbarSample

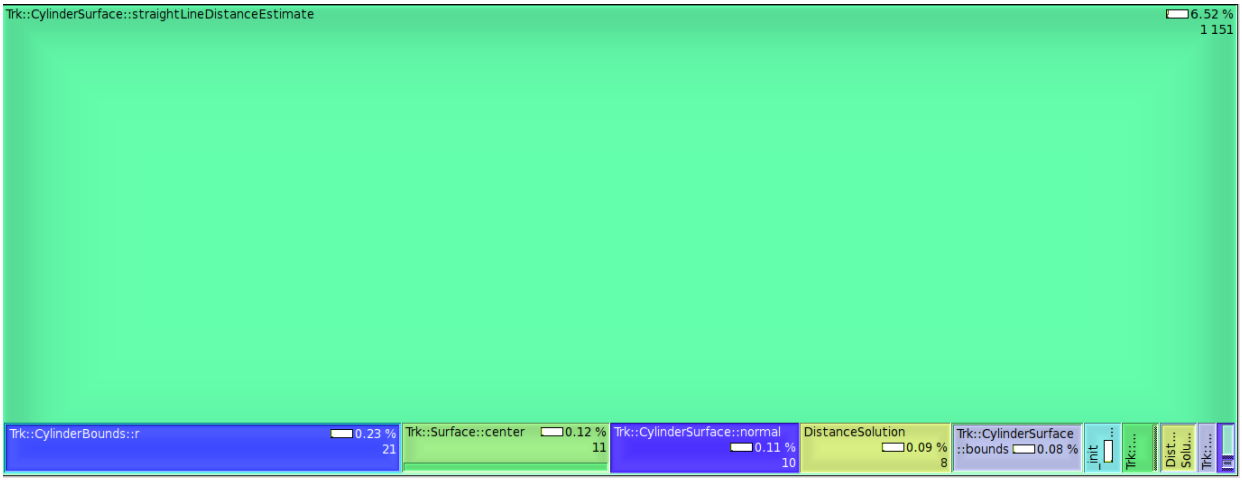


Figure 8 Actual CPU time consumption of improved straightLineDistanceEstimate() shown in the CPU profiling software with ATLFASTIIF detector simulation setup, ATLAS offline software release: 17.6.0.5 & ttbarSample

5 Summary

Detector simulation is one of the biggest challenges for CERN LHC experiments. There are several detector simulation setups varying from full, detailed simulation to very fast, comparatively less accurate simulation. Three different detector simulation approaches are used by the ATLAS Collaboration and studied in this work: Geant4, ATLFASTII and ATLFASTIIF. Improving and testing these detector simulation setups are challenging jobs in addition to creating new simulation techniques. Measuring the time consumption of different simulation setups, creating CPU profiles and improving the code of simulation setups are described in this work. With these techniques, certain performance improvements are gained and test results of the three simulation setups are obtained.

6 Reference

[1] The ATLAS Collaboration, «The ATLAS Experiment at the CERN Large Hadron,» Aug 2008.

[2] W. Lukas, «Fast simulation for ATLAS: Atlfast-II and ISF,» ATL-SOFT-SLIDE-2012-196, 2012.

[3] W. Lukas, «Fast Simulation for ATLAS: Atlfast-II and ISF,» ATL-SOFT-PROC-2012-065, 2012.

[4] K. Edmonds, S. Fleischmann, T. Lenz, C. Magass, J. Mechnich ve A. Salzburger, «The Fast ATLAS Track Simulation (FATRAS),» ATL-SOFT-PUB-2008-001 ; ATL-COM-SOFT-2008-002, 2008.

[5] G. G. P. L. v. 2. (GPLv2). [Online]. Available: http://sourceforge.net/projects/kcachegrind/. [Accessed: 21 November 2013].

[6] ATLAS Detector Software, [Online]. Available: http://acode-browser.usatlas.bnl.gov/lxr/source/atlas/Tracking/TrkDetDescr/TrkSurfaces/src/CylinderSurface.cxx. [Accessed: 12 September 2013].

[7] M. Duehrssen ve K. Jakobs, «Study of Higgs bosons in the WW final state and development of a fast calorimeter simulation for the ATLAS experiment,» CERN-THESIS-2010-061, 2009.

[8] E. Ritsch, A. Salzburger ve E. Kneringer, «Fast Calorimeter Punch-Through Simulation for the ATLAS Experiment,» CERN-THESIS-2011-112, 2011.

[9] A. Rimoldi, «Simulation Strategies for the ATLAS Experiment at LHC,» ATL-SOFT-PROC-2011-016, 2011.

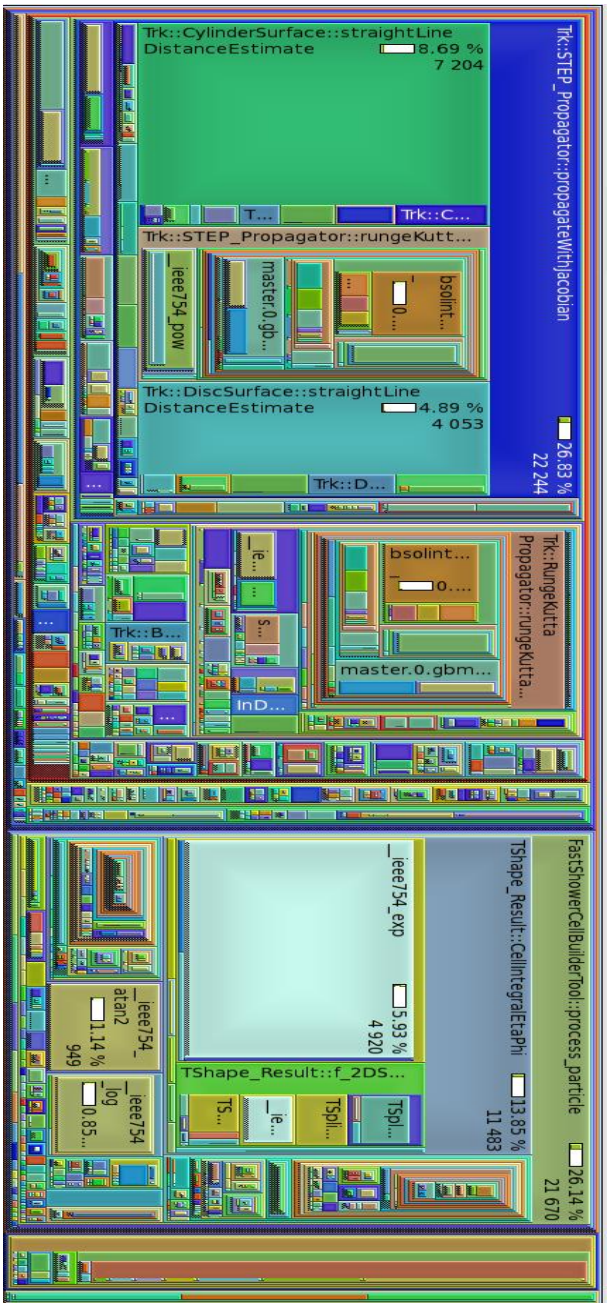
[10] A. Salzburger, «Track Simulation and Reconstruction in the ATLAS experiment,» 2008.

7 APPENDIX

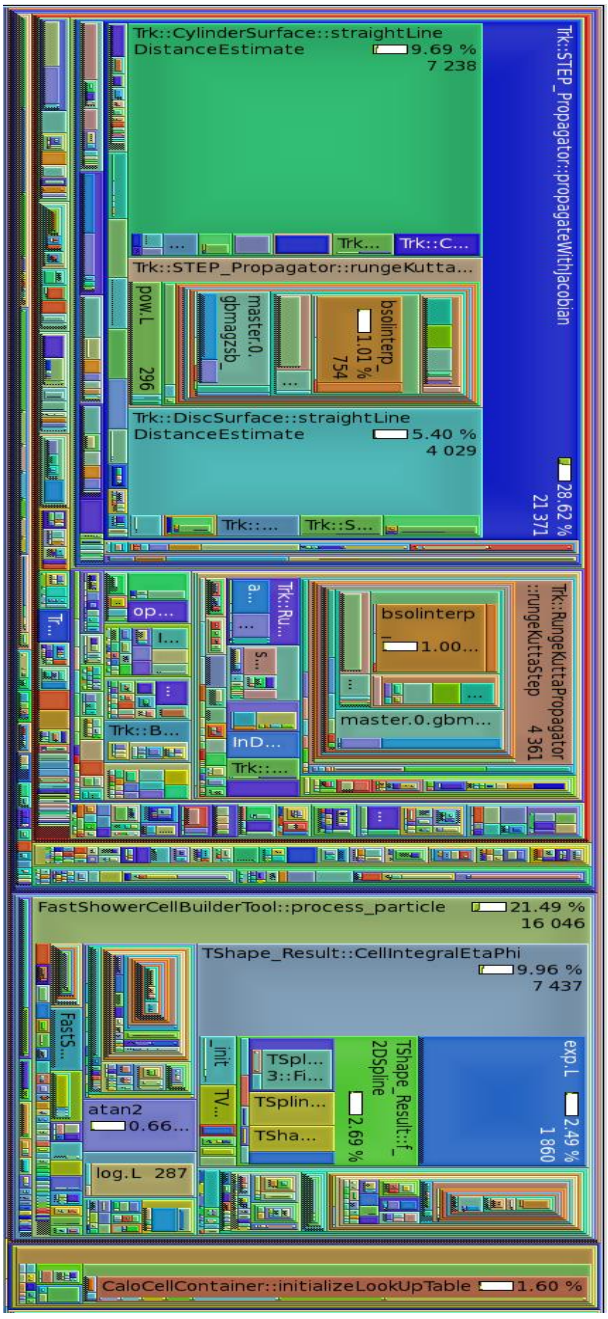
All the results are obtained in the ATLAS offline software release 17.7.X.Y with ttbar event samples and detector geometry version ATLAS-GEO-20-00-01. The percentage values show the actual CPU time consumption in the overall simulation job.Each colored rectangle represents a function; its size tries to be proportional to the cost spent therein while the active function is running.(For further explanation see [5])

7.1 ATLFASTIIF CPU Profiling Results

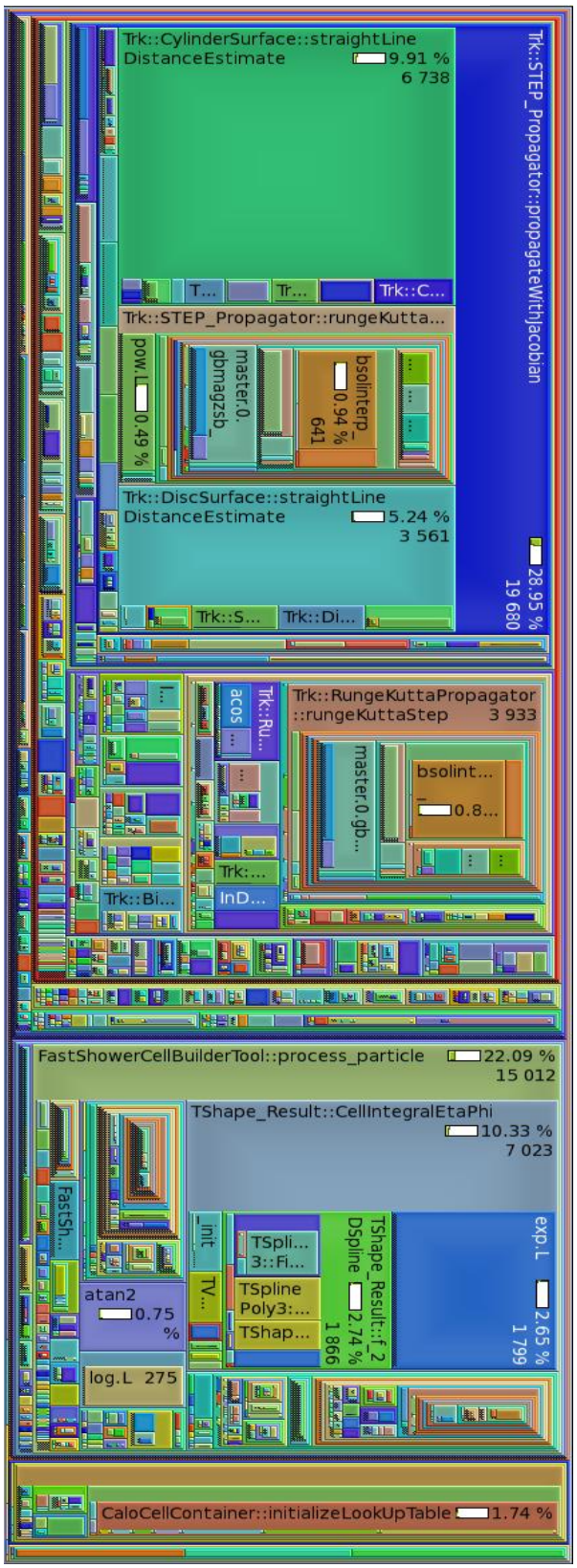
2 Original ATLFASTIIF setup



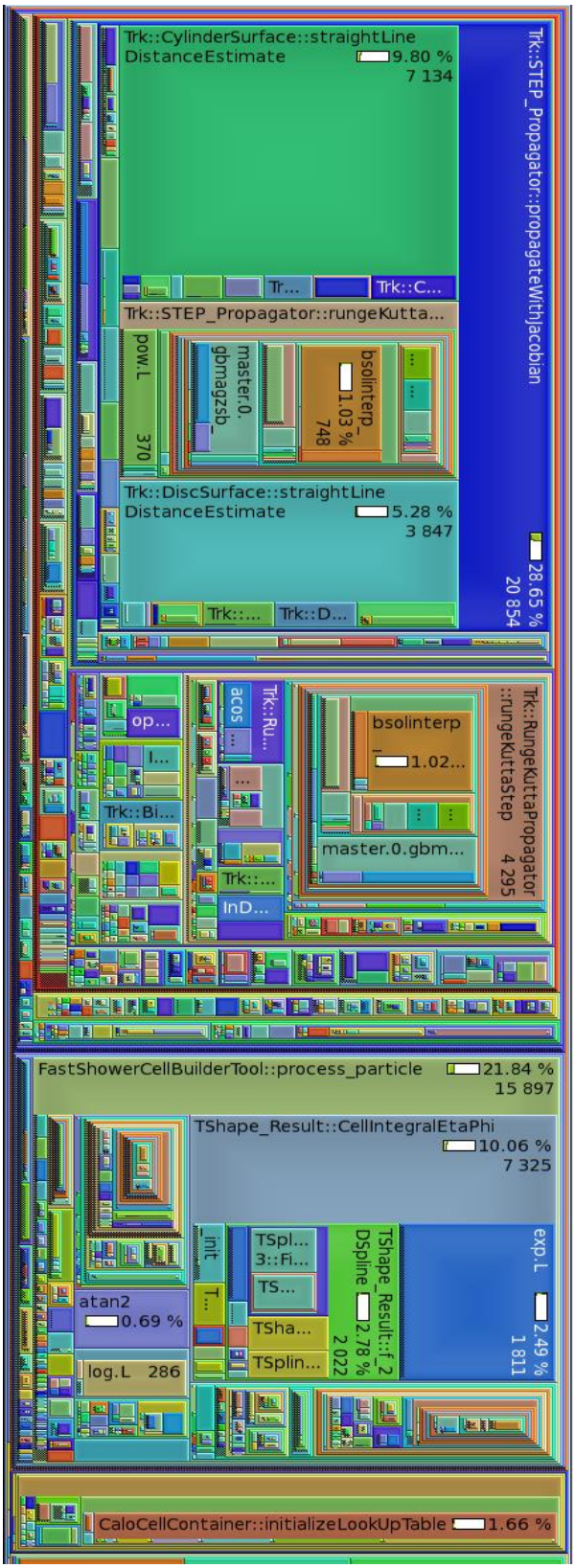
1 ATLFASTIIF with Intel® Math Kernel Library 10.3 update 2



4 ATLFASTIIF with Intel® Math Kernel Library 11.0 update 3

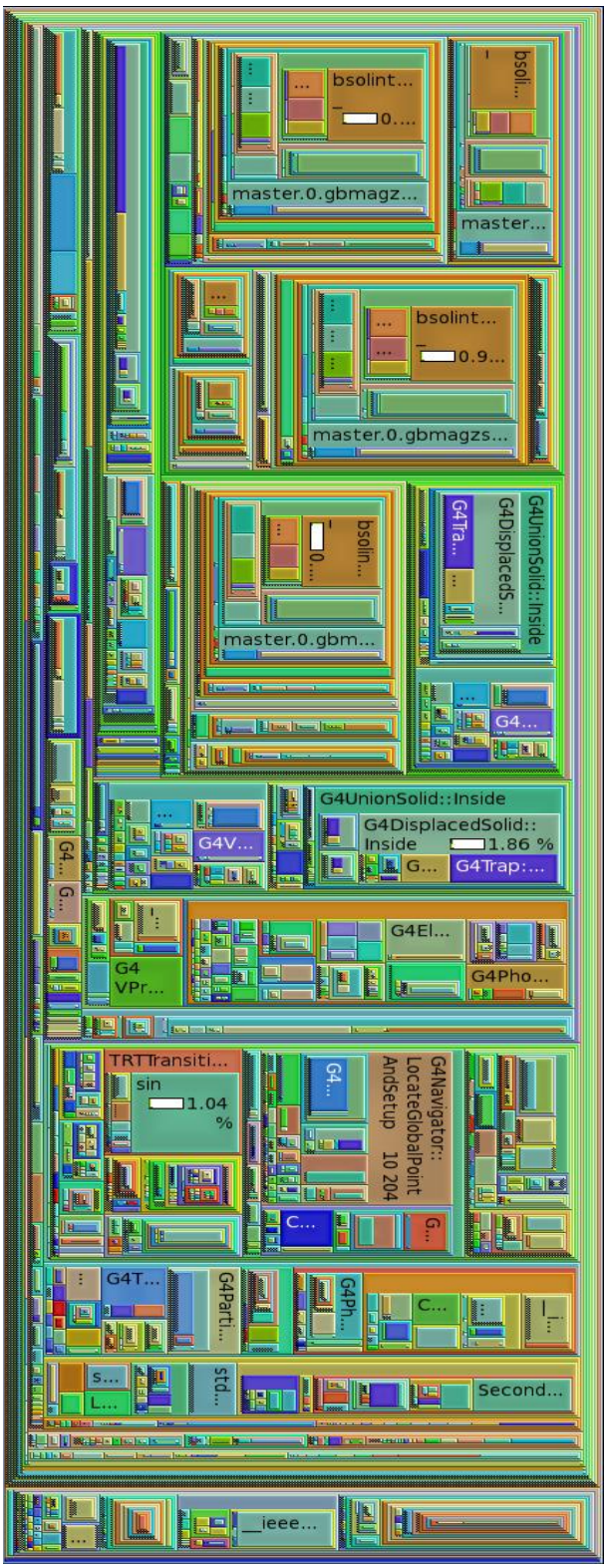


3 ATLFASTIIF with Intel® Math Kernel Library 11.0 update 3 and new ATLAS magnetic field interpolation

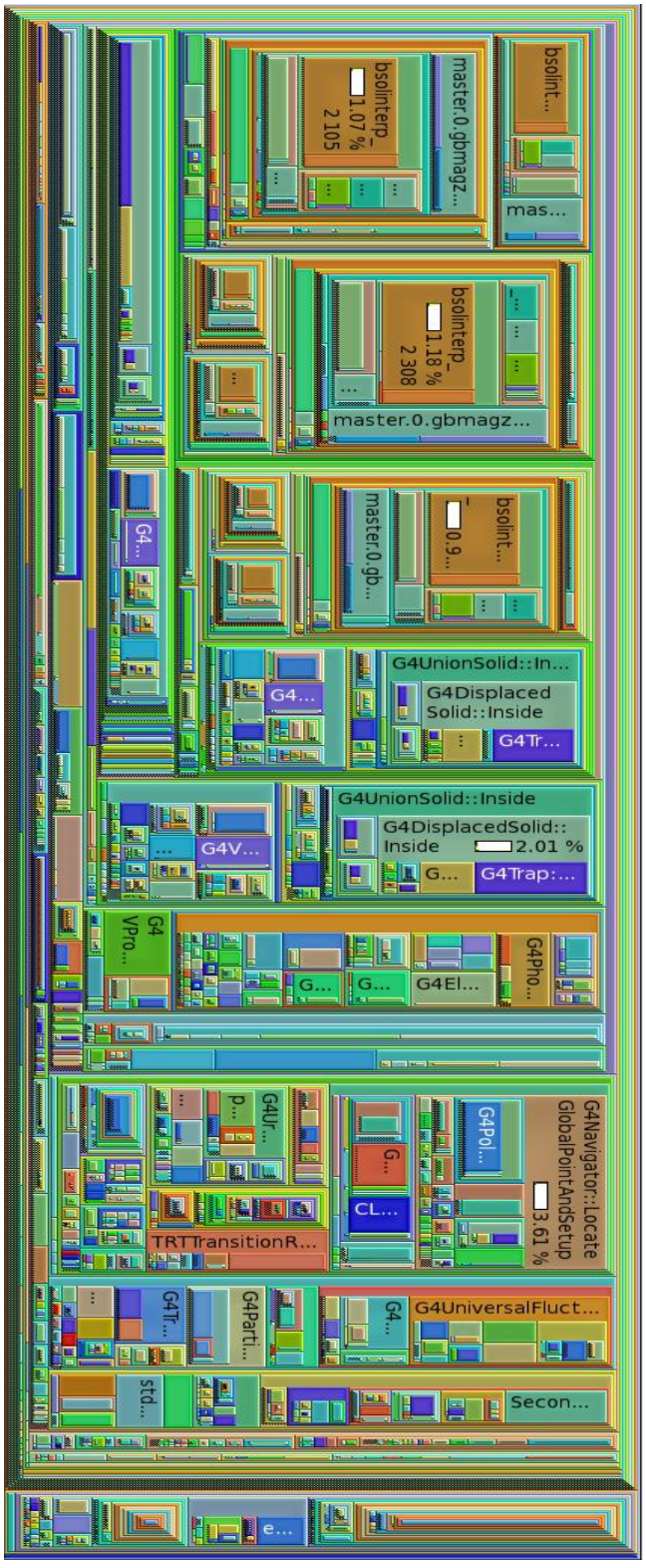


7.2 ATLFASTII CPU Profiling Results

6 Original ATLFASTII Setup



5 ATLFASTII with Intel® Math Kernel Library 10.3 update 2



7 ATLFASTIIF with Intel® Math Kernel Library 11.0 update 3