THE ATLAS SIMULATION: AN LHC CHALLENGE

A. Rimoldi (University of Pavia & INFN), A. Dell' Acqua, M.V. Gallas, A. Di Simone (CERN, Geneva Switzerland), J. Boudreau, V. Tsulaia (University of Pittsburg –USA), D. Costanzo (University of Sheffield – UK)

Abstract

The simulation program for the ATLAS experiment at CERN is currently in a full operational mode and integrated into the ATLAS common analysis framework, ATHENA. The OO approach, based on GEANT4, and in use during the DC2 data challenge has been interfaced within ATHENA and to GEANT4 using the LCG dictionaries and Python scripting. The robustness of the application was proved during the DC2 data challenge. The Python interface has added the flexibility, modularity and interactivity that the simulation tool deserves to implement in a common way different full ATLAS simulation setups, test beams and cosmic ray applications. Generation, simulation and digitization steps were exercised for performance and robustness tests. The comparison with real data has been possible in the context of the ATLAS Combined Test Beam (2004) and ongoing cosmic ray studies.

INTRODUCTION

ATLAS is a general-purpose detector for studies of fundamental particle properties presently in assembly phase and on its way to be operational in the LHC tunnel at the CERN laboratory in Geneva (Switzerland) after about 15 Years of preparation, development and construction.

34 countries, 152 different Institutions and about 1800 scientific authors contributed along this time making its offline software being handled by a common framework so that all the different phases (generation, simulation, digitization, reconstruction and physics analysis) are approached and handled in the same way.

G4ATLAS: THE SIMULATION APPLICATION

The common framework for event processing (ATHENA) is the place where to run applications (ref [1]) in ATLAS.

An application is a set of services and algorithms assembled and configured at runtime, steered using jobOption scripts written in Python language.

Python is an OO-scripting language which is simple and intuitive, robust when used in interactive way, allows introspection mechanisms so that any user can interrogate the object about type and internals adding therefore introspection and interactivity to the C++ layer.

The application used in ATLAS to setup the simulation is named G4Atlas. It is the only application available and supported by the experiment for simulation, it is written entirely in C++, being a full featured OO GEANT4 simulation suite based on dynamic loading and action on demand, so that all user requested functionality is added using plug-in modules. The Python application to setup the appropriate conditions to run the simulation is named PyG4Atlas and its role is to add flexibility for configuring the different setups, interactivity for settings at runtime and introspection.

With these tools we are now able to handle daily user requests and different geometry configurations that could be set at runtime.

The resulting robustness was proved with negligible failures in many productions on the GRID since now. Table 1 shows the present amount of data simulated in big productions since 2004.

The different setups implemented (ref. [2]) are handled similarly:

- full ATLAS Simulation
- cosmic-ray setups
- combined test beams and standalone test beams.

Table 1: ATLAS event productions

Year	**Millions of Events**	**Production Type (full simulation)**
2004	12	Large scale production (DC2)
2005	4	Combined Test Beam for performance studies
2005	8.6	Latest ATLAS workshop
2006	just started	New test production

Consistency and validation effort is kept throughout all the applications so that the user can switch among applications with minimal effort. The non-ideal detector description is in progress to describe the geometry for detector as installed, to introduce the inavoidable misalignements, material services, etc. In addition to that algorithms and tests tools are in place (e.g. G4AtlasTest application) to access the detector "hits", to perform material scans and to allow computations of radiation and interaction lengths along a selected slice of the detector.

These tools are useful to test the correctness of the application as shown for the sample histograms obtained using simulated hits from full 550 Z -> ee events. Fig. 1

shows in a plane R Z (in mm) the hit distribution in the Inner Detector. Fig.2 shows the middle compartment hit map in the Electromagnetic Barrel LAr Calorimeter, while the detector thickness for an η slice < 0.5 as a function of R (cm) is shown in Fig.3.

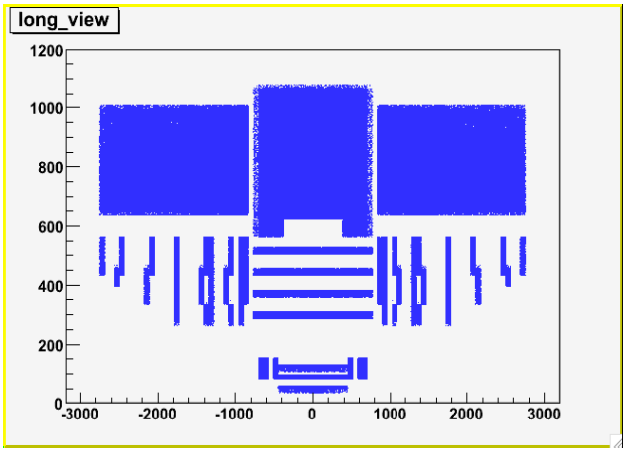

Figure 1: Hit distribution for Z->ee events in the Inner detector in a R Z plane. Units are mm, horizontal axis Z(mm), vertical axis R(mm)

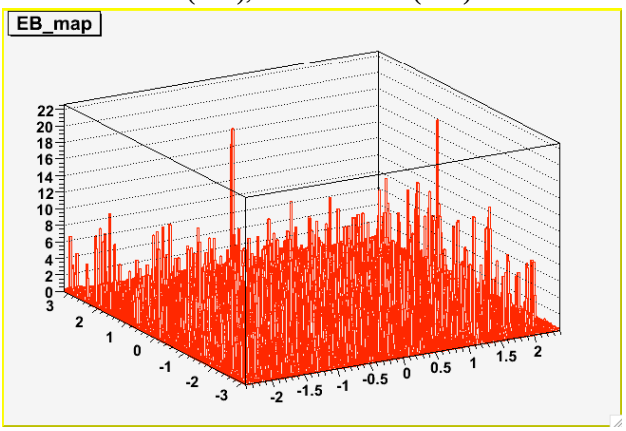

Figure 2: LAr barrel calorimeter hit map -middle compartment (η ϕ distribution, |η| < 2, all ϕ).

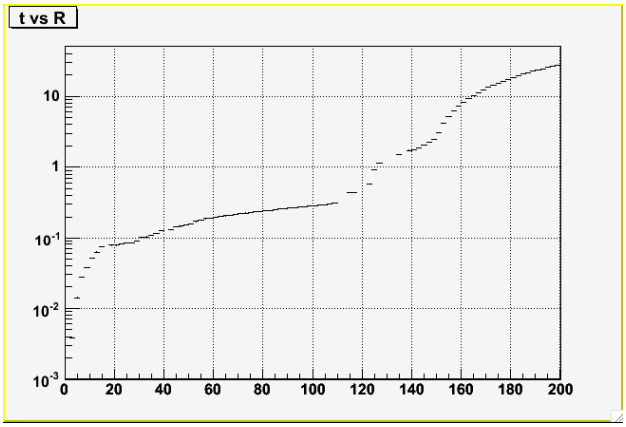

Fig.3: detector thickness X_0(%) for η <0.5 vs R(cm)

ATLAS DETECTOR DESCRIPTION

The description of the complex geometry of ATLAS is decoupled from the simulation framework (G4Atlas) and two hierarchical trees are present in memory at the same time ("GeoModel" and "Geant4). GeoModel provides a transient geometry representation built from primary numbers and alignment constants (Fig.4). The database solution adopted is Oracle and versioning is in place.

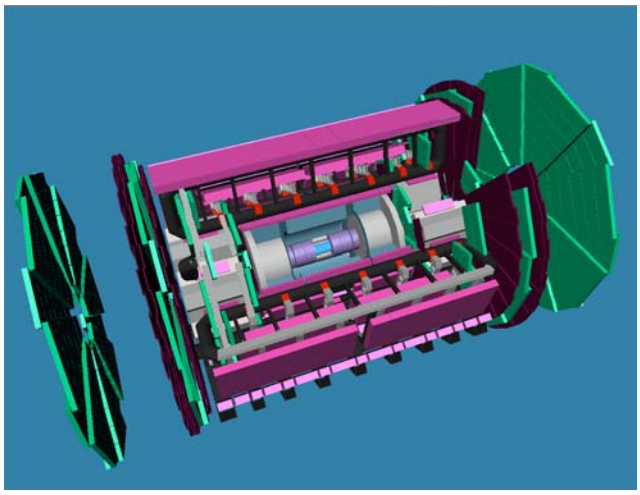

Fig.4 – 3D view of the simulated ATLAS detector

As a consequence of that the simulation, digitization and reconstruction applications are all using the same geometry built at runtime.

The GeoModel description is optimized for a large numbers of volumes (order of few x 10^6) with extensive use of parameterized volumes solutions. At the initialization phase this geometry is translated into the GEANT4 geometry and placed into resizeable and moveable GEANT4 envelopes. Despite all major optimization the total amount of memory required is exceeding today 90 MB. The single contributions from the different detectors are shown in Table 2.

Table 2: breakdown of ATLAS memory allocation at runtime for the different subdetectors (MB)

Subdetector type	**Memory consumption (MB)**
Pixel	5.6
SCT	9.1
TRT	3.1
Inner Detector material	1.0
LAr	54.4
Tile	1.1
Muon System	21.3

GeoModel is also used for the Combined Test Beam description (2004 setup). From 2005 onwards, since the past productions were carried out using ideal detectors with nominal positions, the new productions will include a revised description of the detector "as installed". All information about deformation will be imported into GeoModel with time variation associated to the run number information.

JOB CONTROL

The ATLAS offline software is handled by the ATLAS ATHENA-based framework. This framework uses the Python language as a front-end language to the final users. JobOption files handle jobs via Python scripts configuring input/output, setting configurations and algorithms to run.
Since GEANT4 has no native interface to Python, simulation configurations are passed to GEANT4 through specific command lines.
Users control the application through sets of macro-files: these files indeed are easily diverging in number in a complex application as ATLAS, giving problems of maintenance on the long time life of the experiment.
To avoid these problems a Python layer was recently added and developed in order to allow the user for a direct control of simulation from the ATHENA prompt at runtime. The Python interface application in ATLAS (PyG4Atlas) provides enormous flexibility for configuring and maintain different setups, improving usability by adding interactivity and introspection.
PyG4Atlas is a Python module that uses the PyLCGDict binding to the LCG C++ dictionaries.
As a consequence of this development in the Python environment all macro-files are dropped and substituted by two new classes (G4AtlasControl and G4AtlasInterface) with both public interfaces exported to Python with the result of guaranteeing all functionality and availability of all GEANT4 commands.
G4AtlasApps uses the created dictionaries as well as additional ATLAS or external dictionaries.

The Python G4AtlasInterface

LCG dictionary exports the user interface of the G4Atlas building blocks and needed G4Commands. A thin Python layer reproduces the G4Atlas blocks and the user can access the simulation engine profiting from the introspection and being able to customize it. PyG4Atlas always selectively import Python modules, libraries and dictionaries depending on the user requirements.

G4ATLAS APPLICATIONS

The simulation framework itself offers a set of pre-configured simulation applications (full ATLAS simulation, Combined test beam, cosmic ray setups and old standalone test-beams). For each application several layouts are available so that sub-detector specific studies and user customizations can be easily achieved from the pre-configured applications. These applications are exercised daily through automatic nightly tests. The feedback from users as well as improvements and new features are all included in the preconfigured applications and they are maintained centrally for the community.

Example 1 – Cosmic Ray simulation

During 2006 priority is put in ATLAS in the cosmic ray data. Full support for the cosmic ray simulation is in place, from the description of the experimental area (rock overburden and surface buildings as well) to primary cosmics using a dedicated CosmicGenerator able to produce cosmic muons.
Each detector envelope is used as a scoring layer so that particles at its entrance are recorded. The more external envelope (Muon System) saves the particles propagated through the rock overburden before entering the ATLAS detector so that at the next time the simulation could be restarted from that point.
The ATLAS cavern description, with shafts and muon system , is completely described by the simulation application.

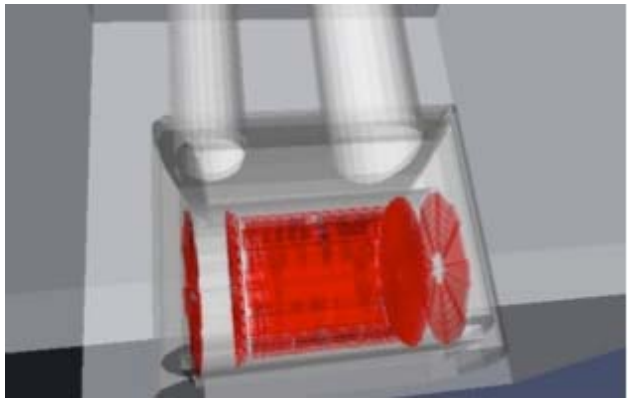

Fig.5: 3D view of the ATLAS "cavern" setup. In red the muon system positioned inside the ATLAS cavern

Example 2 – Combined Test Beam

The Combined test Beam environment is a big and natural source for performance studies and physics validation at LHC (Fig.6). It has been completely simulated with all active and passive components in place.
The simulation infrastructure deals with all the following different configurations:

- Combined mode
- Photon beams
- Material studies
- η scans
- Calibration
- Ancillary detectors

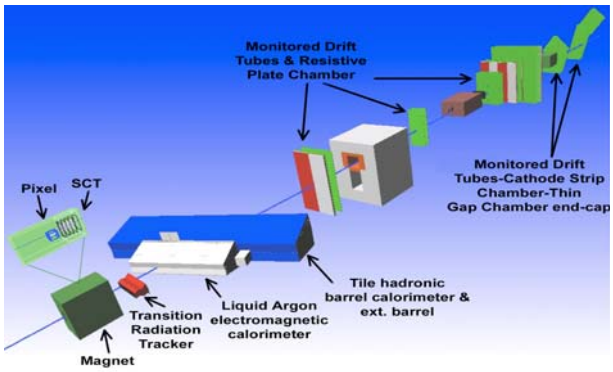

Fig.6: 3D view of the simulated ATLAS Combined Test Beam Setup. In the picture all detector components are described.

In the data taking period (24 weeks in 2004) the layout had frequent and sudden evolutions: simulation of these different and time-dependent layouts was handled by specifying the run number when needed a particular layout version. Single particle generator was used in most cases while Hijing generator was used to speed-up material studies (jet and hadronic processes).

The total data available after the data-taking period is 90Mevents (4.5 TB) and 22Mevents in combined mode. With simulation we produced 4 M events (electrons pions and muons) in a momentum range from 1 to 350 GeV, GRID facilities were extensively used through production of 200 validated runs.

VALIDATION

This process is parallel to the simulation development. The aim of validation is to spot as soon as possible any non-optimal performance or internal inconsistency or even inaccurate description of the detectors or physical processes.

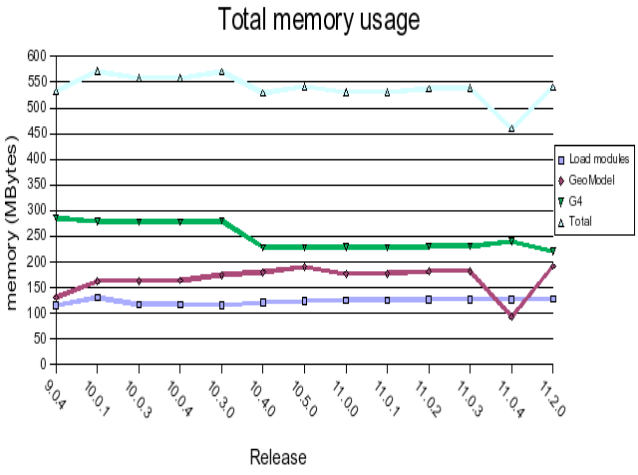

Fig.7 - total memory allocation vs. ATHENA release number (last Year 2005)

Figure 7 shows the memory consumption at runtime (MB) as a function of the ATHENA release number (2005/2006). In the histogram is shown the total amount of memory needed to run the simulation (about 550 MB) with the breakdown of the contributions from GeoModel, Geant4 itself and for the module-loading phase.

The overall approach for validation is threefold:

- Continuous measurement of the performance in terms of the CPU time and memory consumption.
- Comparisons with real data from old standalone test beams for the different subdetectors, ATLAS combined testbeam and cosmic ray tests.
- Physics performance studies by reconstruction of full physical events.

CPU time per event and memory consumption at runtime is daily monitored. Detailed measurements of these quantities in the case of single particle and for full physics events are performed in each new release (ref [3] and ref [4]).

CONCLUSIONS

G4Atlas is the GEANT4-based simulation framework of the ATLAS experiment. It has been successfully and largely used in several massive GRID productions. In this process performance of the Geant4 simulation is systematically monitored, memory usage and normalized CPU time per event are measured for different event types. All performance results are coming as soon as a new release is distributed after a fast phase of automatic testing for pre-validation purposes. The crashing rate resulted to be set around 1/10000 events.

The detector geometry is described by GeoModel and automatically translated to GEANT4. The detector description is being described according to reality (detector as installed and misalignement). PyG4Atlas interface provides the flexibility and configurability required for the full ATLAS and test beam setups (maintenance and usability were achieved).

PyLCGAtlas uses the LCGDictionaries through PyLCGDict Python binding to connect Python to the C++ layers.

The data from the combined test beam is a good source for the study of detector performance and GEANT4 physics validation.

The long-term success will be measured by the user-friendliness and performance of the ATLAS simulation both in large-scale coordinated data challenges and on the desktop.

There is a lot more that will go into this program over the course of its very long life. A user community of about 2000 people will soon make this one of the most scrutinized computer programs in HEP checked with real-data in every corner of the phase space.

Optimization will play an important role in the next future and in this direction our next major development efforts are concentrated.

ACKNOWLEDGEMENTS

I wish to thank all my colleagues of the ATLAS Simulation Core Team: I presented this work on their behalf. A particular thank is also due to the developers in the subdetector groups for the constant improvement of the quality of their implementation and to the Geant4 and LCG people for the continuous help in the phase of tuning of this complete tool to perform the ATLAS detector simulation.

REFERENCES

[1] Athena framework: *http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/Documentation/AthenaDeveloperGuide-8.0.0-draft.pdf*.

[2] *ATL-SOFT-2004-006 . The Simulation of the ATLAS EXperiment: present Status and Outlook. by:Rimoldi, A; Dell'Acqua, A; Gallas, M; Nairz, A; Boudreau, J; Tsulaia, V; Costanzo, D;; Geneva : CERN, 15 Nov 2004*

[3] ATL-SOFT-PUB-2005-004, "Validation of the GEANT4-Based Full Simulation Program for the ATLAS Detector" by D. Costanzo, A. Dell'Acqua, M. Gallas, A.Nairz, N. Benekos, A. Rimoldi, J. Boudreau, V. Tsulaia

[4] Simulation validation page:
http://atlas-computing.web.cern.ch/atlas-computing/packages/simulation/geant4/validation/Validation.html