The LHCb Simulation Application, Gauss: Design, Evolution and Experience

M Clemencic[1], G Corti[1], S Easo[2], C R Jones[3], S Miglioranzi[1], M Pappagallo[4] and P Robbe[1,5]
on behalf of the LHCb Collaboration

[1] European Organization for Nuclear Research (CERN), Geneva, Switzerland
[2] STFC Rutherford Appleton Laboratory, Didcot, Oxfordshire, United Kingdom.
[3] Cavendish Laboratory, University of Cambridge, Cambridge, United Kingdom
[4] Sezione INFN di Bari and Universitá di Bari, Bari, Italy
[5] LAL, Université Paris-Sud, CNRS/IN2P3, Orsay, France

E-mail: `Silvia.Miglioranzi@cern.ch`

Abstract. The LHCb simulation application, Gauss, is based on the Gaudi framework and on experiment basic components such as the Event Model and Detector Description. Gauss also depends on external libraries for the generation of the primary events (PYTHIA 6, EvtGen, etc.) and on GEANT4 for particle transport in the experimental setup. The application supports the production of different types of events from minimum bias to B physics signals and particle guns. It is used for purely generator-level studies as well as full simulations. Gauss is used both directly by users and in massive central productions on the grid. The design and implementation of the application and its evolution due to evolving requirements will be described as in the case of the recently adopted Python-based configuration or the possibility of taking into account detectors conditions via a Simulation Conditions database. The challenge of supporting at the same time the flexibililty needed for the different tasks for which it is used, from evaluation of physics reach to background modeling, together with the stability and reliabilty of the code will also be described.

1. Introduction

Simulation applications are of major importance both in the design and construction phase of an experiment and during its operation. They allow to understand experimental conditions and performances. In LHCb [1] the task of modeling the behavior of the spectrometer for the different type of events occurring in the experiment is carried out by two separate applications called Gauss and Boole [2]. Gauss generates the initial particles and simulates their transport through the LHCb detector, whilst Boole reproduces the different subdetectors responses and their digitization converting the data in the same format provided by the experiment electronics and the DAQ system. After digitization real data and Monte Carlo data follow the same path through trigger, reconstruction and analysis procedures. The Gauss simulation application has been used extensively in massive production in the experiment since 2004 and has undergone various changes to support its evolving needs. The general structure of the Gauss simulation application will be described and a few selected aspects which are of interest for the functionality and evolution of the code will be addressed. The monitoring and validation of the application that are regularly performed to ensure the stability and reliability of the software will also be briefly described.

2. Gauss as a Gaudi Application

In the LHCb experiment all the event data processing applications are built on Gaudi [3], [4], an Object Oriented (OO) framework providing a common infrastructure and environment. Gaudi identifies components with specific purposes and well defined interfaces, interacting with each other to provide the whole functionality of an application. Gaudi provides to the application general purpose components, like messaging and configuration services.
By design, the framework decouples objects describing data from those describing algorithms. This ensures a longer stability for the data objects (the LHCb event model) as algorithms evolve much more rapidly. *Algorithms* and *Tools* (light-weight algorithmic objects) are the essence of the data processing applications, encapsulating in this case the functionality specific to the simulation. The Gaudi *Application Manager* takes care of instantiating and calling the algorithms. Gauss is customized by choosing and configuring the appropriate set of algorithms to execute in a given sequence and other suitable components for the various tasks. Some of these components, as the generator and the GEANT4 toolkit [5], [6] interfaces have been built specifically for the LHCb simulation application, but were kept general so they can be used by other experiments using Gaudi-based software. A schematic view of the Gaudi architecture can be seen in the object diagram shown in Fig. 1. where a snapshot of the state of the system

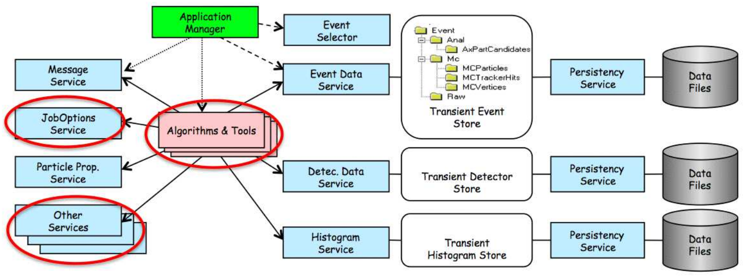

Figure 1. Object Diagram of the Gaudi Architecture. Place holders for the Gauss specializations are circled in red.

showing the objects (in this case component instances) and their relationships is represented.

3. Structure and Sequencing of the application

The Gauss application consists of two major independent processing phases:

(i) the generation of the primary event,

(ii) the tracking of the particles through the detector.

A schematic structure of the application can be found in Fig. 2. The format used for the generated event is the HepMC event record [7], wrapped into Gaudi data objects. The output of the detector simulation uses the LHCb Event Model format [8], which is then used as the input format for the Boole digitization application.
Both Generator and Simulation sequences have their own configuration and initialization. A dedicated algorithm takes care of the global initialization of each sequence. The random number seed is reset for each event both at the beginning of the Generator and of the Simulation sequences by using a combination of *Run, Event Number* and algorithm name. The *Run* and *Event Number* are stored in the generated event and read back when initializing the simulation

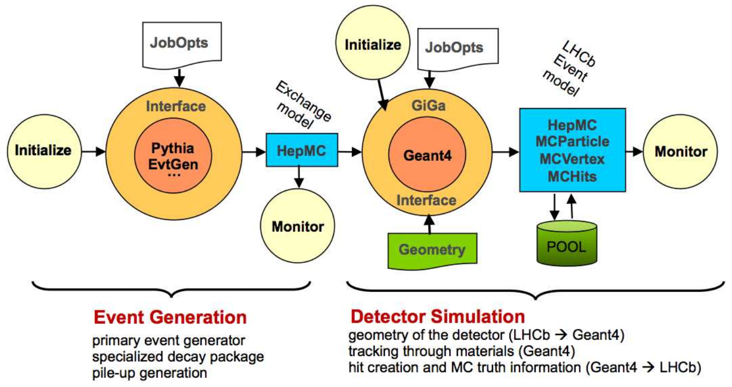

Figure 2. Schematic view of the structure of the Gauss Application.

phase to ensure the event reproducibility.
Two types of Gauss output are predefined, reflecting the two common ways of running the application:

(i) generator-only mode (*"gen"* output),
(ii) full simulation mode (*"sim"* output).

In the full simulation mode, together with the generator output, the information of the evolution of particles through the detector and of the hits produced is also stored.
The simulation application is independent from the underlying persistent technology. Currently, LHCb is using POOL [9] for the event data files and ROOT [10] for histograms and *ntuples* to save the information on disk, but this could be changed at any time, without affecting the actual simulation application.

3.1. Generator Sequence

In LHCb the production of particles coming out of the primary pp collision of the LHC beams is handled by default with PYTHIA [11], a general purpose event generator, whilst the decay and time evolution of the produced particles is delegated to EvtGen [12]. This package, originally designed for the BaBar collaboration, is specialyzed to accurately describe B decays with a special customization for LHCb needed to handle incoherent B^0 and B^0_s production in contrast to the coherent production at the B-factories.
The Generation algorithm uses tools that can be plugged in, realizing specific actions: generation of number of pile-up events (*Pile-Up Tool*), generation of a given event sample (*Sample Generation Tool*), production of N p-p interactions (*Production Tool*), generation of beam parameters (*Beam Tool*), decay of unstable particles (*Decay Tool*), cut at generator level on the decay (*Cut Tool*), cut on full event properties (*Full Event Cut Tool*) and smearing of the primary vertex (*Vertex Smearing Tool*).
Each tool has a generic interface and at least one specific implementation allowing to reuse a large amount of common code and resulting in a higher flexibility.
This tool structure allows to smoothly add new external generator libraries as Pythia8 [13] or HERWIG++ [14] for production of pp interactions or SHERPA [15] for particles decays.

3.2. Tracking through the detector

In Gauss the simulation of the physics processes undergone by the particles travelling through the detector, is delegated to the Geant4 toolkit. Gaudi specialized service components (GiGa [16]) encapsulate the Geant4 engine and provide a unified high level abstract interface. A set of abstract interfaces implemented within GiGa allows the conversion of the LHCb detector geometry from the Gaudi transient store into the Geant4 geometry and the possibility to use standard LHCb core software services by Geant4. A dedicated Gauss algorithm transforms the output of the Generator phase of Gauss to the Geant4 input format via the GiGa interface that trigger the Geant4 event manager to process the event. The output of Geant4 in the form of Monte Carlo truth history as well as hits produced in the sensitive detectors is then transformed back into the LHCb event model via GiGa dedicated algorithms. The complete behavior of the Geant4 simulation engine in terms of detectors to simulate, physics models to use, details of the Monte Carlo truth to be provided, is controlled at run time via configuration options.

4. Configuration
Gaudi provides a *Job Options Service* to configure the applications at run-time. The options, or properties, are set via a job options file, which is read in when the Job Options Service is initialised by the Application Manager. The structure of Gauss as described in Sec. 3 is realized at run time by providing a well defined list of options. This was initially implemented as a set of ASCII-Files containing statements read in at the very start of the execution of a job when the Job Options Service was initialised by the Application Manager. The files had a C++-like syntax with the data types recognized by a simple parser (*Job Options Compiler*). In 2009 it was decided to adopt a new feature of Gaudi for job configurations based on the Python language, allowing a high level configuration of the application. This Python configuration is based on *Configurables* [4], special python classes built from the C++ components (Services, Algorithms, Tools...). This structure allows to profit of the characteristics of the python language, not only ensuring basic option validation but also implementing high level configuration and checks.

5. Event Model and MC truth
By design all the simulation classes inherit from the LHCb *Data Object* and *containers*. The generator and simulation classes are kept separated, communicating between each other via algorithms and tools and no major upgrades have been implemented for them in the recent years.
The MC truth-level information is vital in order to understand efficiencies and physics effects therefore efforts have been made to save the LHCb event model MCHistory (MCParticles, MCVertices and their relationships) in a consistent way to avoid loss of informations during the tracking of the particles inside the detector. Since Geant4 does not have an easily accessible tree structure to keep history of the tracks, HepMC was internally introduced to Geant4 to mantain this structure.
During normal production operations, keeping track of the full history is not optimal (e.g. shower history in calorimeter can be skept); to solve this problem, an intermediate shower particle to mantain the otherwise disappearing links between saved daughters and their non-rejected ancestors, has been introduced.

6. Geometry modeling
The detector description is contained in an XML geometry database. The conversion of the LHCb geometry model to the Geant4 geometry tree is done at the beginning of each simulation job by a specialized geometry conversion service.
The geometry configuration can be changed easily via job options, without any need for recompilation and the addition, removal or changing of sensitive detectors can be performed by simply editing the geometry description file. These changes are then taken into account at run-time and the instantiation of the required sensitive detectors is done using the abstract factory approach.
In reality, the reconstruction algorithms deal with a detector whose alignment is not perfectly

known. The misalignment and other data taking conditions are provided via an XML LHCb condition database. Some of these information (e.g. magnet polarity) is given to the simulation via a similar XML simulation condition database. Work is in progress to understand how misalignment conditions data can be made available to the simulation to allow detailed understanding of systematic effects.

7. Handling of spill-over events

The digitization program was previously configured to read events from two distinct Gauss files: an in-time file containing simulated events for the channel under study (usually specific decay channels, or a generic mixture of B events, or a minimum bias mixture of events), and a spill-over file containing an independent mix of minimum bias events.
In 2009 the spill-overs simulation changed moving from the digitization application directly into Gauss, allowing to handle these events inside the same file in a single simulation job. The Gauss sequence is set to allow interactions to occur in the preceding and following beam crossing by replicating the whole event sequence for those beam crossings close enough for spill-over to occur. When initializing the processing of these spill-over sequences, Gauss populates these events according to the probability of their occurrence based on the instantaneous luminosity with which it has been configured. In this way only one instantiation of Pythia, EvtGen and Geant4 is needed which handles both the main event and the spill-overs. Moreover, the simulation and decay packages do not need to be reconfigured when dealing with the spill-overs events.

8. Monitoring, Output and Performance

Gauss releases are built on average with monthly frequence together with patches releases when needed. Upcoming releases and development versions are checked in the LHCb nightly build system [17] that provides the application binary products for their integration testing on different platforms. The nightly system provides fast feedbacks to the developers about their changes in order to spot any potential simulation problems early and take appropriate actions. A set of nightlies run-time tests have been developed for Gauss using the QMTest open-source software [18] in order to monitor specific simulation benchmark samples.
A monitoring sub-phase is available for each of the Gauss phases (Fig. 2). In production most histograms, ntuples and printout are turned off. They can be selectively turned on to study the simulation performances, in particular to verify new versions of the program against well-tested reference output. For this purpose, a complete set of Monitoring tools were re-engineered from the Online software [19]: a SQL-based database storing the display settings and configurations for all histograms, a graphical user interface (the Presenter) to display the ROOT-based histograms published by the monitoring algorithms and an histogram analyser package. In addition, a set of python scripts are periodically run on log files and ROOT histograms to produce statistics tables (generator efficiencies, comparison between versions, etc.) in html format that are needed by the analyzers.
The detector XML geometry description changes periodically due to continuous updates. Volume overlaps can cause the simulation jobs to crash, therefore it is vital to ensure that after applying all the misalignments, the geometry description is overlaps-free. The tool mainly exploited to detect geometry anomalies is the Geant4 DAVID visualization tool [20] which can find overlaps between volumes and convert them into a graphical representation for visualization purposes. Moreover, radiation length evaluations are performed when changes are introduced to compare differences in the material budget encountered by the particles in the spectrometer.
Every time a new Geant4 version is adopted, distributions of the main Physics quantities (track multiplicities, cross sections, etc.) and process-related issues (description of the dE/dx in thin Si detectors, Multiple Coulomb Scattering simulation, etc.) which are affecting the simulation, are kept under control. Different sets of Geant4 Physics Lists [21] are also periodically investigated

to ensure the more suitable physics models to be chosen for the description of electromagnetic and hadronic physics occurring inside the detector.

9. Conclusion

The LHCb simulation application Gauss has been described, focusing on its evolution steps recently needed to cope with crucial requirements as the handling of the MCHistory and the treatment of the spill-overs. Different monitoring tools for both geometry and Physics simulation have been introduced which have been essential to guarantee the good quality of the latest years MonteCarlo productions.

Acknowledgments

Special thanks go to I. Belyeav for its ideas on the interface service to Geant4 and the development of GiGa and to W. Pokorski who developed and implemented the first completely functional version of Gauss. Their work has provided the foundations on which the current application was built.

References

[1] The LHCb Collaboration, Alves A A et al., The LHCb Detector At The LHC, *JINST* **3** 2008 S08005

[2] The LHCb Collaboration, Antunes Nobrega R et al., LHCb Computing Technical Design Report, *CERN-LHCC-2005-019* 2005

[3] Barrand G et al., GAUDI - A software architecture and framework for building HEP data processing applications, *Comput. Phys. Commun.* 140 2001 45

[4] Clemencic M, Degaudenzi H, Mato P, Binet S, Lavrijsen W, Leggett C and Belyaev I, Recent Developments In The LHCb Software Framework GAUDI, *J. Phys. Conf. Ser.* 219 2010 042006

[5] The GEANT4 Collaboration, Agostinelli S et al., GEANT4, a simulation toolkit, *Nucl. Instr. and Meth. Phys. Res.* **A** 506 2003 250-303

[6] The GEANT4 Collaboration, Allison J et al., Geant4 developments and applications, *IEEE Transactions on Nuclear Science.* **v. 53**, 1 2006 270-278

[7] Dobbs M and Hansen J B, The HepMC C++ Monte Carlo event record for High Energy Physics, *Comput. Phys. Commun.* 134 2001 41

[8] *https : //twiki.cern.ch/twiki/bin/view/LHCb/LHCbEventModel*

[9] Duellmann D, Frank M, Govi G, Papadopoulos I and Roiser S, The POOL data storage, cache and conversion mechanism, *Proceedings of 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 24-28 Mar 2003 *arXiv:physics/0306084*

[10] Brun R and Rademakers F, ROOT - An Object Oriented Data Analysis Framework, *Proceedings AIHENP96 Workshop, Lausanne*, Sep. 1996, *Nucl. Inst. and Meth. in Phys. Res.* A 389 1997 81-86. See also *http : //root.cern.ch/*.

[11] Sjostrand T , Mrenna S and Skands P Z, PYTHIA 6.4 Physics and Manual, *JHEP* 0605 026 2006

[12] Lange D J, The EvtGen particle decay simulation package, *Nucl. Instrum. Meth.* **A** 462 2001 152

[13] Sjostrand T, Mrenna S and Skands P Z, A Brief Introduction to PYTHIA 8.1, *Comput. Phys. Commun.* 178 2008 852

[14] Bahr M et al., Herwig++ Physics and Manual, *Eur. Phys. J.* C 58 2008 639.

[15] Gleisberg T, Hoeche S, Krauss F, Schonherr M, Schumann S, Siegert F and Winter J, Event generation with SHERPA 1.1, *JHEP* 0902 2009 007

[16] Belyaev I, Frank M, Gracia G, Mato P and Ranjard F, Integration of GEANT4 with the GAUDI framework, *Proceedings of 2001 Conference for Computing in High-Energy and Nuclear Physics (CHEP01)*, Beijing P. R. China 3-7 Sep 2001, *http : //www.ihep.ac.cn/ chep*01/5 − 009*.pdf*

[17] Kruzelecki K, Roiser S and Degaudenzi H, The nightly build and test system for LCG AA and LHCb software, *Proceedings for CHEP 2009*, Prague, Czech Republic, March 21-27, 2009

[18] QMTest URL: *http : //www.codesourcery.com/qmtest*

[19] Graziani G, Histogram DB for Online Monitoring User's Manual, */afs/cern.ch/user/g/ggrazian/www/lhcb/histdbdoc.pdf*
Presenter URL: *https : //lbtwiki.cern.ch/bin/view/Online/HistogramPresenter*

[20] The GEANT4 Collaboration, Geant4 User's Guide for Application Developers, 2009 110

[21] The GEANT4 Collaboration, Physics Reference Manual, 2009