PROCEEDINGS OF SCIENCE

Development, validation and maintenance of Monte Carlo event generators and generator services in the LHC era

M. Kirsanov*
INR (Moscow)
E-mail: Mikhail.Kirsanov@cern.ch

A. Ribon
CERN
E-mail: Alberto.Ribon@cern.ch

O. Zenin
IHEP (Protvino)
E-mail: Oleg.Zenin@cern.ch

The LCG Generator Services project collaborates with the authors of Monte Carlo generators and with the LHC experiments in order to provide validated LCG compliant code for both theoretical and experimental communities at the LHC.
The libraries of the Monte Carlo generators maintained within this project are currently widely adopted by the LHC collaborations and are used in large scale productions. The existing testing and validation tools are regularly used and additional ones are being developed, in particular for the new object-oriented generators. The aim of the validation activity is also to participate in the tuning of the generators in order to provide appropriate settings for the proton-proton collisions at the LHC energy level.
This paper presents the current status and the future plans of the LCG Generator Services project. The approach used in order to provide tested Monte Carlo generators for the LHC experiments is discussed and some testing and validation tools are presented.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7 2008
Erice, Italy

*Speaker.

PoS(ACAT08)114

1. Introduction

The LCG Generator Services project was started in 2003 as a long term project. Its purpose is to provide support to the high-energy event-generation software and related services in the LHC era. The project consists of the following four subprojects:

- WP1: GENerator SERvices library (GENSER);
- WP2: Event formats and event interfaces;
- WP3: Shared event files: framework and Data Base (MCDB);
- WP4: Tuning and validation.

The following scientific centers and groups participate in the project: CERN, FNAL, LCG Russia.

More information on the LCG Generator Services project can be found at:
`http://lcgapp.cern.ch/project/simu/generator/`.

2. GENSER

The purpose of this subproject is to replace the obsolete CERN library for what concerns the MC event generators and related software and to maintain them during a long period of the LHC operation, in the conditions of deployment of new computer platforms and compilers. It should be noted that a significant part of the packages included in GENSER have been written in Fortran over a long period of time and thus makes use of some legacy Fortran features. At the same time, some of them are rather sophisticated, so that the development and validation of the corresponding modern object-oriented substitutions would take years to reach the same production quality.

The duties of the GENSER team are the following:

- Collaborate with MC authors in order to prepare validated LCG compliant codes
- Maintain the legacy software packages on the LCG supported platforms
- Integrate new packages upon requests from LHC experiments and other GENSER users.

In 2004 - 2005 GENSER reached a production quality. During this period ATLAS and LHCb started to use it for the Monte Carlo productions. In 2006 CMS also started to use GENSER.

During the period 2006 - 2007, without interrupting its services, GENSER underwent the evolution from the SCRAM [1] build system and a quarterly release scheme, which turned out to be insufficiently flexible, to the GNU Make build system and a continuous release scheme. After this evolution the version of GENSER changed and it became GENSER 2.

GENSER 2 develops and provides the GNU Make build system for MC generators distributed as a single source file or for generators with insufficiently developed build system (the latter is always agreed with the authors). A typical build system includes a `configure` script compliant to the adopted standards (arguments `-help`, `-with-package`, etc.). The build system should be

capable to build both static and shared libraries. While most LHC experiments use shared libraries, static ones can also be needed.

The generator subdirectory structure adopted in GENSER includes a subdirectory `/examples` with a similar `configure` script and a Makefile that can produce one or several executables.

The author's build system is used if it is sufficiently developed.

For each generator there are GENSER installation scripts invisible for users.

The main features of GENSER 2 are listed below:

- **Continuous release scheme**: a new version of a generator is installed in `/afs/cern.ch/sw/lcg/external/MCGenerators` as soon as it is released by the authors
- The **distribution** of the generators is performed through the tarballs kept on CERN AFS in the `MCGenerators/distribution` directory. For each generator a source tarball (with the author's or GENSER-provided build system) and binary tarballs for each LCG-supported platform are provided. Binary tarballs are usually sufficient to deploy a generator on a standalone machine without recompilation. The `examples/` directory is also included in the binary tarballs
- **Security**: old GENSER versions are never removed. Normally, new generator versions are installed to dedicated AFS volumes. After internal testing the volume becomes read-only. If a package needs a correction while authors did not yet release a new version, this correction is agreed with the authors and a new GENSER subversion of the package is installed. Only one person is allowed to create new volumes and set their permissions
- **Standard LCG platforms**, currently `slc4_ia32_gcc34` and `slc4_amd64_gcc34`, are fully supported

At present 28 MC generators and related packages are included in GENSER. ATLAS and LHCb directly use the GENSER libraries installed on CERN AFS while CMS builds generators from the GENSER source tarballs and installs them to the CMS AFS volumes.

3. LCG generator testing and validation: general considerations

Large scale MC productions in LHC experiments involve qualified manpower and a lot of computer resources. They are rather expensive. At the same time, a bug in a MC event generator, depending on its severity, can make such a production mostly useless. In order to avoid such problems, the MC generators must be thoroughly tested. There are, however, contradicting requirements to this validation: it should be thorough and detailed, but at the same time it should be performed quickly.

In the opinion of the LCG Generator team, several **levels** of testing and validation can help. These levels are seen as follows:

- **Level 0**: The generators table on the GENSER WEB page is always prepared by a special script. This script checks the existence of non-zero size libraries according do definite rules (one or several libraries should correspond to a package) and the existence of tarballs.

- **Level 1**: Dedicated package in GENSER. The test script compiles one or several applications for each generator to be tested and runs them one by one. Each application computes values of one or several observables to be compared with reference values. The latter are obtained by running the same applications for a well defined version of the considered generator, ususlly corresponding to the date of creation of the test

- **Level 2**: Dedicated projects (like WP4)

- **Level 3**: Tests inside the experiments

Note that levels 2 and 3 are outside WP1 GENSER. Additional levels can also be used.

4. LCG generator testing and validation: Level 1 testing

For this purpose the dedicated package `TESTS` in GENSER was created. It includes service routines and codes for a simple reconstruction and analysis, like jet reconstruction or checks of lepton isolation. The subdirectory `/examples` of this package contains test scripts and main modules for test applications.

The test procedure takes into account the fact that the results of the simulation are often **correct, but not identical to the reference results**. This is due to the fact that even small modifications in the Monte Carlo programs could change the sequence of random numbers (even if started from the same initial seed). For this reason all values of observables are compared with the reference ones taking into account statistical errors.

The comparison is performed by a special script which presents the results in a form of an HTML table. All significant deviations (above 3σ) are reported as "deviation". Missing results (this can happen if the script failed to compile an application or if an application crashed) are also reported as "errors" without stopping the comparison program. The comparison tables can be seen on the GENSER WEB page.

All generators available in GENSER have at least one Level 1 test. The work on the extension of tests proceeds according to the generators priority.

GENSER also included several tests developed by LHC experiments into the Level 1 test suite. There are plans to include other such tests.

The full GENSER testing (newest versions only) runs approximately 12 hours on 5000 MIPS x86_32 and x86_64 CPUs. This involves separate tests of shared and static libraries and takes into account that for generators using the HEPEVT common block there are always two subversions, with HEPEVT common block size set to 4000 and 10000.

Every year $\sim$ 1 - 2 bugs are found with the Level 1 GENSER testing.

From time to time models used in the generators change, which leads to permanent deviations from the reference values. After discussions with the authors, the reference values are eventually updated. These changes are described in a history file.

5. Preparation of the LCG generator to new compilers

The current version of the GNU compiler collection is GCC4. In GCC4 the `g77` fortran compiler is replaced by `gfortran`. The latter almost conforms to the Fortran 95 standard but does not support some Fortran 77 features.

Although gcc4-based platforms are not yet officially supported by LCG, GENSER team devoted some efforts to the migration to this new generation of compilers. Few minor problems were encountered in PYTHIA 6, several problems appeared in HERWIG (return to label, subroutine entry points, etc.). In collaboration with the authors new versions of these most popular general-purpose Fortran generators were prepared.

In November 2008 the testing of generators on the platforms slc5_ia32_gcc43 and slc5_amd64_gcc43 has started. The full support of these platforms is planned for the second half of 2009.

6. Package management

Currently the mature `pkgsrc` [2] package management system is being tested as a possible future OS-independent base for GENSER. This package comes from Berkeley UNIX and runs on all POSIX platforms: GNU/Linux, *BSD, Mac OS X, Solaris, HP-UX, AIX, Cygwin/NT, etc.

The essential features of the `pkgsrc` system are:

- MC generators and related packages can be transparently fetched, extracted from source tarballs, patched, configured, compiled and installed. This process is fully controlled by a compact set of variables and (optionally) "make" rules kept in a single file for each package
- There is a possibility of an automatic preparation of patched source tarballs that can be used in a standalone mode
- The `pkgsrc` provides an isolated POSIX environment to build the packages, thus reducing the dependency on an OS-specific environment. A possible insufficient portability of the package's native build system can be partially compensated for
- The system provides an automatic dependency tracking

The `pkgsrc` system can automatically build the entire GENSER on a standalone site. To simplify this, a special "bootstrap" script is provided by GENSER [3]. Several high priority generators are already migrated to the `pkgsrc` framework, a complete migration being expected by the middle of 2009. If `pkgsrc` is adopted as a base for GENSER, the latter will become GENSER 3.

7. Other possibilities for the build systems

Currently `Autotools` [4] are being evaluated as a GENSER - provided build system for generators distributed without one. This build system has the following advantages:

- `Autotools` correctly work on any POSIX platform: GNU/Linux, Mac OS X, etc.

- `Autotools` are an industrial standard *de facto*. This implies a fast adoption of any prospective general purpose OS which might be used by the physics community

However, in our opinion `Autotools` have also some disadvantages:

- The build system becomes quite sophisticated for the sake of portability: e.g, the `configure` script automatically generated from user-defined rules can be bigger than all the source code of a generator
- `Autotools` do not automatically trace dependencies on `includes` in a Fortran code
- Sometimes it is not easy to make a build system flexible enough
- There are some caveats with relocation of libraries (i.e. using libraries after moving them to a location different from the original installation) generated by `libtool`

One of the most popular generators, Pythia 6, is ported to this build system. The corresponding GENSER subversions 419.ac of the version 419 are available for tests by the LHC experiments (they exist in parallel to the subversions with a traditional build system).

8. WP2: Event formats and event interfaces

The goal of this subproject is to standardize interfaces, avoiding the proliferation of versions and duplication of work. This should improve the conditions for the development and validation of the new MC generators created using Object Oriented technology.

The most popular interface is HepMC [5]. GENSER regularly organizes the HepMC discussion meetings. The frequency of such meetings for 2009 is agreed to be 6 months. However, since a migration to a new HepMC version is rather painful due to a growing number of dependencies, the desirable frequency of this migration is one per year.

The HEPML - Meta-data format facilitating automated documentation - is being prepared by the CEDAR collaboration and the MCDB team (see below).

9. WP3: Production and maintenance of shared event files

The goal of this subproject is to produce and store certified generator level event files. This subproject is motivated by the increasing complexity of the event generation in the LHC era.

For the storage and maintenance of these event files the Monte Carlo Data Base (MCDB) is developed under the subproject WP3. The maintenance includes book-keeping and providing documentation for the files, using the competences of the Monte Carlo experts and HEPML as a language.

MCDB is currently used in the CMS MC production. There is a separate presentation on MCDB and HEPML at ACAT-2008.

10. WP4: Tuning and validation of the MC generators

This subproject deals with specialized tools for the detailed testing, comparison with data and tuning of the generators. The main tool adopted for this in the MC generators community is RIVET [6], developed by the CEDAR collaboration. RIVET is a replacement of HZTool. There is a presentation on RIVET at ACAT-2008.

The GENSER team was one of the first users of RIVET and developed several examples of RIVET analyses that compare the results of the simulation of processes at Tevatron by MC generators with experimental data taken from the the HEP databases. More validation examples are under development.

11. Conclusion

In order to provide the MC generator services in the LHC era, the LCG generator project was started in 2003. The subprojects GENSER and MCDB of this project are now in production, the former is already widely used by the LHC experiments. However, an intensive development of GENSER continues in order to make it more reliable, convenient and flexible when used on a variety of commonly used computing platforms.

To provide the Level 0 and Level 1 testing of the generator library GENSER a special script and a special GENSER package were created. The Level 1 testing covers now all generators and auxiliary packages. Its development continues in order to perform a more detailed testing. This is true also for the subproject of the LCG Generator Services devoted to the MC tuning and validation.

References

[1] `https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookScramV1Intro`

[2] `http://www.netbsd.org/docs/pkgsrc/`

[3] `http://lcgapp.cern.ch/project/simu/generator/genser-bootstrap.html` (*preliminary*)

[4] `http://www.gnu.org/software/autoconf/`
`http://www.gnu.org/software/automake/automake.html`
`http://www.gnu.org/software/libtool/libtool.html`

[5] `http://lcgapp.cern.ch/project/simu/HepMC/`

[6] `http://projects.hepforge.org/rivet/`