GENSER status

Ivan Razumov, Grigory Latyshev, Dmitri Konstantinov

GENSER SFT Team

October 19, 2015

SFT Group Meeting

What is GENSER?

GENSER

The Generator Services project:

- work in collaboration with Monte Carlo (MC) generators authors and LHC experiments in order to prepare validated LCG compliant code for both the theoretical and experimental communities at the LHC sharing the user support duties
- provide assistance for the development of new object-oriented generators and guarantee the maintenance of the older packages on the LCG supported platforms

Generator Service

The Generator Service project purpose is to provide the following components for LHC community and experiments:

- MC generator libraries installed on CERN AFS and CVMFS
- compressed source files (tarballs)
- various tests of Monte Carlo generators
- service support for LCG release users

Team

The idea is to cover entire year with 2–3 months long shifts

GENSER team **(~ 1 FTE):**

- Mikhail Ilyushin (IHEP, Protvino)
- Mikhail Kirsanov (INR, Troitsk)
- Dmitri Konstantinov (IHEP, Protvino)
- Grigory Latyshev (IHEP, Protvino)
- Ivan Razumov (IHEP, Protvino)

Funding: 50% CERN, 50% Russian

We have been providing this service successfully for the last ~10 years!

How GENSER works?

GENSER software stack

- GENSER code is fully integrated in SFT LCGCMake system:
  - same GIT repository – https://gitlab.cern.ch/sft/lcgcmake
  - generators are usual ‘packages’ in terms of lcgcmake
- all generators requested by experiments are part of official LCG release (central installation managed by Patricia)
- a life cycle of LCG releases is too long and we are often requested to update a set of existing LCG installations with newly released generator/tool (managed by GENSER)

Generator integration

- for new generator:
  - add/update instructions for generator in lcgcmake
  - download the tarfile to SFT/GENSER local source repository.
  - add new version into steering tool chain file.
- for new LCG release:
  - prepare new toolchain file with all requested versions

Build instructions in lcgcmake

```
LCGPackage_Add(
  evtgen
  URL http://cern.ch/service-spi/external/tarFiles/MCGeneratorsTarFiles/evtgen-<evtgen_<NATIVE_VERSION>_tag>.tar.gz
  UPDATE_COMMAND <VOID>
  CONFIGURE_COMMAND ./configure --prefix=<INSTALL_DIR>
                                --hepmcdir=${HepMC_home}
                                --pythiadir=<pythia8-<evtgen_<NATIVE_VERSION>_pythia8>_home>
                                --photosdir=${photos++_home}
                                --tauoladir=<tauola++-<evtgen_<NATIVE_VERSION>_tauola++>_home>
  BUILD_COMMAND ${MAKE} -j1 "${evtgen-build-options}"
  INSTALL_COMMAND make install
          COMMAND ${CMAKE_COMMAND} -E chdir <INSTALL_DIR>/../share ${CMAKE_COMMAND} -E create_symlink  sources/evt.pdl
evt.pdl
          COMMAND ${CMAKE_COMMAND} -E chdir <INSTALL_DIR>/../share ${CMAKE_COMMAND} -E create_symlink  sources/
DECAY_2010.DEC DECAY.DEC
  BUILD_IN_SOURCE 1
  DEPENDS HepMC pythia8 photos++ tauola++
)
```

- A few lines in CMakeLists.txt required to describe installation steps and dependencies of a new packages
- Explicit handling of dependencies

Steering tool chain

- All packages(externals + generators) are kept in one steering ‘toolchain’ file:

```
#---Additional External packages-------(Generators)-----------------

set(MCGENPATH  MCGenerators_lcgcmt${heptools_version})

LCG_external_package(powheg-box          r2092         ${MCGENPATH}/powheg-box    )
LCG_external_package(lhapdf              5.9.1         ${MCGENPATH}/lhapdf        )
LCG_external_package(lhapdf6             6.0.5         ${MCGENPATH}/lhapdf6        )
LCG_external_package(pythia8             175.lhetau    ${MCGENPATH}/pythia8      author=175      )
LCG_external_package(sacrifice           0.9.9         ${MCGENPATH}/sacrifice  pythia8=183 )
LCG_external_package(thepeg              1.9.0         ${MCGENPATH}/thepeg        )
LCG_external_package(herwig++            2.7.0         ${MCGENPATH}/herwig++     thepeg=1.9.0)
LCG_external_package(tauola++            1.1.4         ${MCGENPATH}/tauola++      )
LCG_external_package(pythia6             428.2         ${MCGENPATH}/pythia6        author=6.4.28 hepevt=10000   )
LCG_external_package(agile               1.4.1         ${MCGENPATH}/agile         )
LCG_external_package(photos++            3.54          ${MCGENPATH}/photos++      )
LCG_external_package(photos              215.4         ${MCGENPATH}/photos        )
```

Flexibility: syntax allows to pass ‘key=value’ pairs to package build function

Each generator has 1 main and N extra versions in release

Build instructions

- Simple steps like ‘./configure; make; make install’

```
git clone https://gitlab.cern.ch/sft/lcgcmake.git

mkdir build; cd build

source /afs/cern.ch/sw/lcg/contrib/gcc/4.8/x86_64-slc6/setup.sh

cmake \
  -DLCG_VERSION=78 \
  -DCMAKE_INSTALL_PREFIX=../install \
  -DLCG_INSTALL_PREFIX=/afs/cern.ch/sw/lcg/releases \
  ../lcgcmake

# make help will prints all available targets
make -jN package ...
```

Requirements: Unix (or MacOS), cmake >=2.8, autotools (for some packages

https://gitlab.cern.ch/sft/lcgcmake/blob/master/documentation.markdown/How-to-build-generators-with-LCGSoft-CMake-system.md

Tests: making sure everything works as it should

Nightly builds(Jenkins) + monitoring (CDash)

- Automated building of the entire software stack everyday
  - problems immediately visible
  - several platforms are tested

LCGSoft

Dashboard Calendar Previous Current Project

No file changed as of **Monday, March 31 2014 - 03:00 CEST** Show Filters Advanc

Experimental

Site	Build Name	Update	Configure		Build		Test		
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass
ec-slc6-x86-64-spi-8	x86_64-slc6-gcc48-opt	0	0	0	4^{+4}	1^{+1}	0	16^{+14}	72^{+1}_{-13}

Preview

Site	Build Name	Update	Configure		Build		Test		
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass
ec-slc6-x86-64-spi-9	x86_64-slc6-gcc48-opt	0	0	0	0	0	0	0_{-2}	88^{-2}
ec-slc6-i686-spi-1	i686-slc6-gcc48-opt	0	0	0	0	0	0	0_{-1}	69^{-1}
lxbsp0516.cern.ch	x86_64-slc5-gcc43-opt	0	0	0	0	0	0	0	69
ec-slc6-x86-64-spi-7	x86_64-slc6-icc14-dbg	0	0	0	27	1	0	38	31
macitois17.cern.ch	x86_64-mac108-gcc42-opt	0	0	0	10_{-4}	1	0	15_{-4}	32^{-4}

Release

Site	Build Name	Update	Configure		Build		Test		
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass
ec-slc6-x86-64-spi-8	x86_64-slc6-gcc48-opt	0	0	0	0	0	0	0	91^{+4}
lxbsp0516.cern.ch	x86_64-slc5-gcc43-opt	0	0	0	0	0	0	0_{-1}	71^{+1}
macitois17.cern.ch	x86_64-mac108-clang34-opt	0	0	0	19	1	0	7	41
macitois18.cern.ch	x86_64-mac108-gcc42-opt	0	0	0	21^{+2}	3^{+2}	0	7	41
ec-slc6-x86-64-spi-9	x86_64-slc6-gcc47-opt	0	0	0	0	0	0	0	71

Testing

- the generator tests are run after each nightly build
- all generator are covered at least by a simple tests – compile simple code with the library and run it
- A few popular generators have rivet-based tests for comparing results with previous versions

Testing started on 2015-10-17 13:03:49

Site Name: lcgapp-slc6-physical1.cern.ch
Build Name: experimental-x86_64-slc6-gcc48-opt
Total time: 3h 52m 47s 590ms
OS Name: Linux
OS Platform: x86_64
OS Release: 2.6.32-504.23.4.el6.x86_64
OS Version: #1 SMP Wed Jun 10 07:47:59 CEST 2015
Compiler Version: unknown

311 tests passed.

Name	Status	Time
gg2VV-3.1.7.native	Passed	9m 48s 730ms
debug-symbols-test	Passed	7m 2s 410ms
herwigpp-2.7.1.genser-LHC-W	Passed	6m 28s 590ms
sherpa-2.2.0.blackhat.genser-zjetsgenser-rivet	Passed	5m 28s 350ms
sherpa-2.2.0.genser-zjetsgenser-rivet	Passed	5m 25s 350ms
sherpa-2.1.1.native-LHC_ZJets	Passed	5m 12s 510ms
sherpa-2.1.1.genser-zjetsgenser-rivet	Passed	5m 1s 290ms
coral-tests	Passed	4m 4s 620ms
env-scripts-test	Passed	3m 55s 630ms

Implementation of tests

Adding tests by ‘add_test’ function in lcgcmake

```
#---test of the test infrasgtructure-----------------------------------
LCG_add_test(test-test PRE_COMMAND /bin/echo pre-comand 1
                            COMMAND /bin/echo pre-comand 2
                            COMMAND /bin/echo pre-comand 3
                            COMMAND /bin/echo pre-comand 4
                            COMMAND /bin/echo pre-comand 5
                       TEST_COMMAND /bin/echo test-command 1
                            COMMAND /bin/echo test-command 2
                            COMMAND /bin/echo test-command 3
                       POST_COMMAND /bin/echo post-command 1
                            COMMAND date )
```

Simple SHERPA test:

```
# $PWD is needed because Sherpa uses this variable to know current directory
LCG_add_test(sherpa_orig_test1
  PRE_COMMAND ${CMAKE_COMMAND} -E remove_directory sherpa/tests_orig1
      COMMAND ${CMAKE_COMMAND} -E make_directory sherpa/tests_orig1
  TEST_COMMAND ${CMAKE_COMMAND} -E chdir sherpa/tests_orig1 ${sherpa_home}/bin/Sherpa -j20 -f ${CMAKE_SOURCE_DIR}/generators/sherpa/tests/LHC_Z.dat
  ENVIRONMENT
    ${library_path}=${sherpa_home}/lib/SHERPA-MC:${HepMC_home}/lib:${lhapdf_home}/lib:${fastjet_home}/lib:${GSL_home}/lib:$ENV{${library_path}}
    PWD=.
)
```

What is Rivet

- Rivet is a generator-agnostic validation system for MC generators.
- More simply, it's a tool to produce physics plots from an MC generator code which can produce HepMC events. All the "major" generators can do this one way or another: C++ Pythia 8, Sherpa, Herwig++ out of the box, Fortran PYTHIA 6, HERWIG+JIMMY, etc. via AGILe.
- This is useful for validating generators – only need to write the analysis once and it can be used to validate and compare every generator that should be able to simulate it.

Validation tests based on Rivet

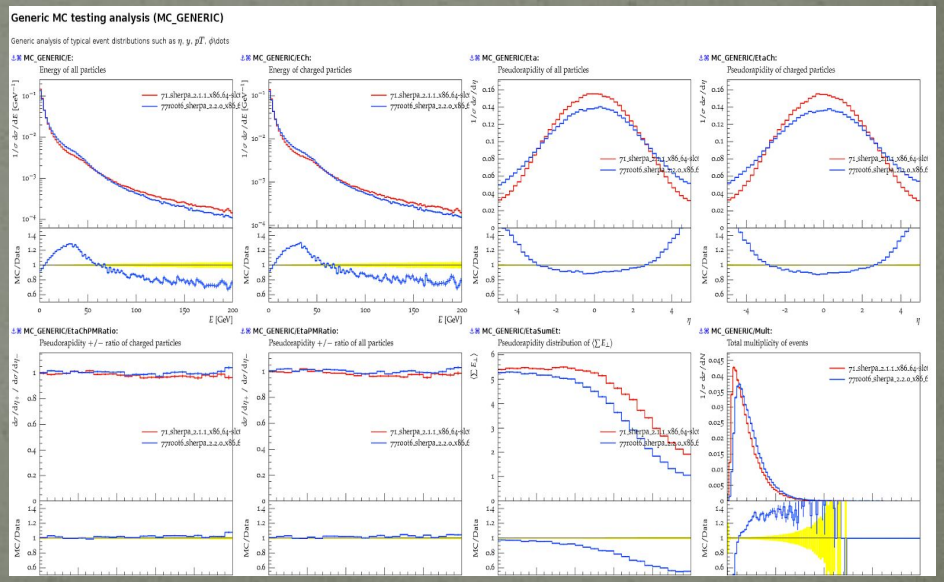
Exploiting the power of Rivet we have implemented:

- simple fast (~1000 events) tests – run every day
- validation website* running regression tests (~ 10^6 events) – run on demand

Displayed analyses:

- MC_GENERIC
- MC_IDENTIFIED
- MC_XS
- MC_TTBAR
- MC_ZJETS

  (+GENSERZJETS)



* http://genser.cern.ch

A new tool: LCGEnv

Motivation

- To make the generators and other packages portable, all the “RPATH”s are stripped from the produced executables/libraries
- This is not a problem for experiments, which have dedicated environments for using LCG software
- This **is** a problem for those who want to use LCG software outside that environment

rpath is a term in programming which refers to a run-time search path hard-coded in an executable file or library, used during dynamic linking to find the libraries the executable or library requires.

Solution

The “lcgenv” is a tool for advanced users who want to use individual tools from the LCG release without setting up the entire release:

- sets up all the dependencies needed for a package
- works with official LCG releases as well as local user builds
- optimized to work CVMFS installations
- supports sh- and csh-like shells
- is used in per-package environment scripts

The script is available on CERN AFS and SFT CVMFS.

It was tested by ATLAS and included in ATLAS UI

```
lcgenv -p /cvmfs/sft.cern.ch/lcg/releases/LCG_79 86_64-slc6-gcc48-opt ROOT
```

Status and Plans

Status and plans

- fulfilling requests from experiments and users for addition of new generators and new versions
- contribution to “lcgcmake” build system development to meet latest requirements (users and ours)
- preparation of LCG stack for “exotic” platforms:
  - MacOS - clang, Ubuntu-gcc, SLC-icc
    (never ending effort…)

Generators provided in 2015

- chaplin 1.2
- CRMC 1.5.3, 1.5.4
- cython 0.22
- Dpmjet 3.0-6
- evtGen R01-04-00
- FeynHiggs 2.10.2
- gg2VV 3.1.7
- HepUtils 1.0.0, 1.0.6, 1.0.7
- HYDJET++ 2.1
- JHU 4.8.1, 5.2.5, 5.6.3
- KKMC 4.22
- LoopTools 2.8
- MCUtils 1.1.0, 1.1.1, 1.1.2
- aMCatNLO 2.2.1, 2.2.2
- photos++ 3.60, 3.61
- powheg-box-v2 r3033, 3043
- professor 2.0.0
- pythia 8.204, 8.205, 8.210, 8.212
- Rivet 2.2.1, 2.3.0, 2.4.0
- SHERPA 2.2.0
- vinicia 1.1.00, 1.1.01, 1.1.02, 1.1.03, 1.2.00, 1.2.01, 1.2.02
- yoda 1.3.1, 1.4.0, 1.5.5

Conclusion

- There is a synergy between GENSER and SPI teams
- GENSER has evolved from a standalone project and has been contributing into a common infrastructure
- Because we have more established and experienced manpower this year, we plan to extend GENSER team involvement in HepMC3 and Geant validation.
- The GENSER project is going well! Its results are essential for experiments and HEP users!
- We have established direct contact with experiments: they can rely on us

backup

Rivet validation

Based on GENSER Rivet tests, but has larger statistics:

- 1000 events are generated in **nightly** tests
- 100000 events are generated in **validation** tests

Needs to be ran manually, takes about 1–2 days.

Results can be viewed on

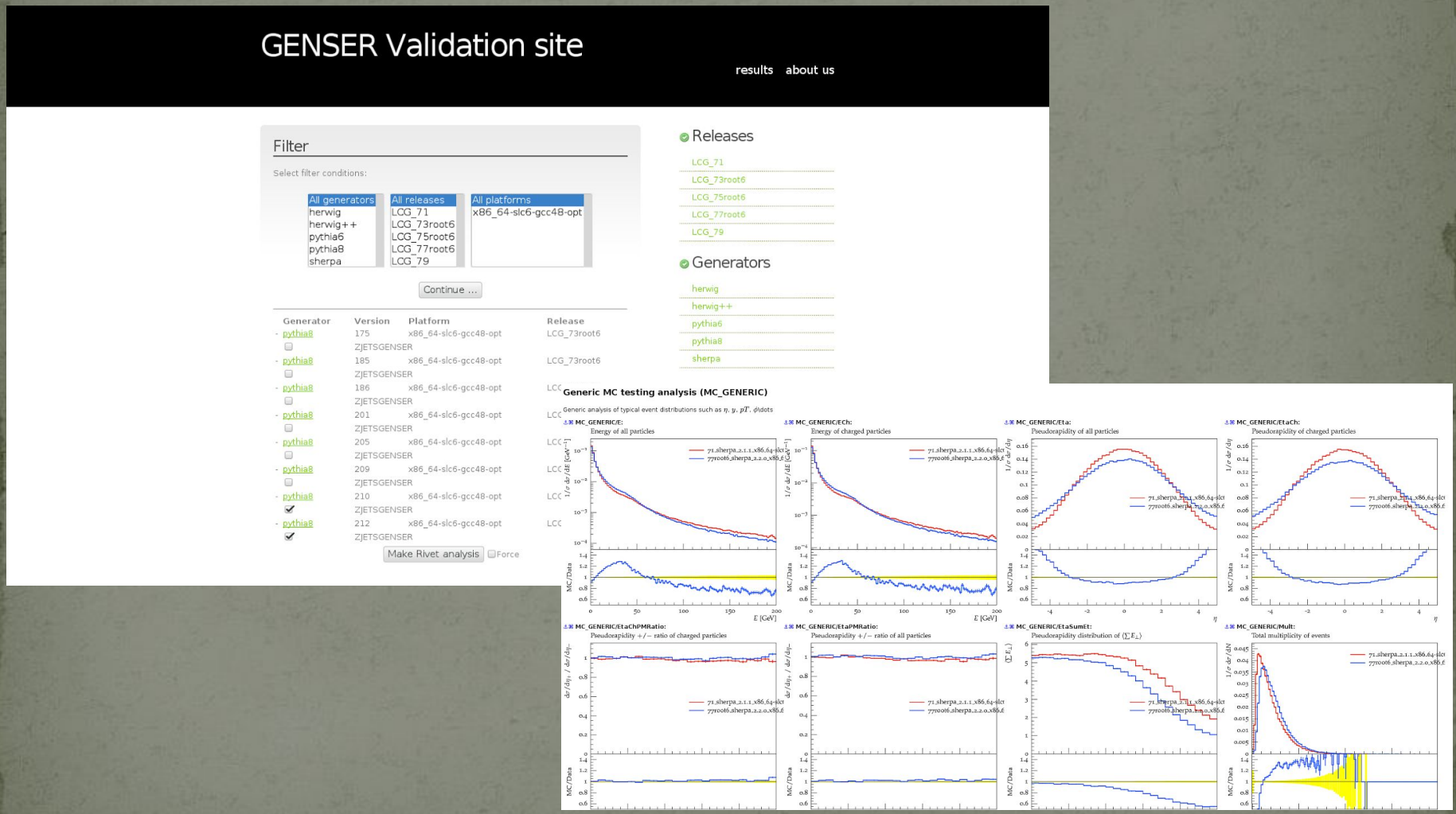
http://genser.cern.ch

Build instructions

- Requirements:
  - Unix
  - CMake 2.9
  - Compiler, autotools (for some packages)
- Simple steps:
  - checkout lcgcmake from git
  - create build area
  - configure with cmake
  - build with ‘make’

https://gitlab.cern.ch/sft/lcgcmake/blob/master/documentation.markdown/How-to-build-generators-with-LCGSoft-CMake-system.md

Rivet validation (2)



Rivet-based tests

GENSER maintains Rivet-based tests for

- herwig++
- Sherpa
- pythia8
- pythia6
- herwig

reference.py -- ROOT-based tool for comparing test results and reference results