Journal of Physics: Conference Series **513** (2014) 022037 doi:10.1088/1742-6596/513/2/022037

The STAR "plug and play" event generator framework

J. Webb[1], J. Novak[2], J. Lauret[1] and V. Perevoztchikov[1]
Brookhaven National Laboratory[1]
Michigan State University[2]

E-mail: jwebb@bnl.gov

Abstract. The STAR experiment pursues a broad range of physics topics in pp, pA and AA collisions produced by the Relativistic Heavy Ion Collider (RHIC). Such a diverse experimental program demands a simulation framework capable of supporting an equally diverse set of event generators, and a flexible event record capable of storing the (common) particle-wise and (varied) event-wise information provided by the external generators. With planning underway for the next round of upgrades to exploit ep and eA collisions from the electron-ion collider (or eRHIC), these demands on the simulation infrastructure will only increase and requires a versatile framework. STAR has developed a new event-generator framework based on the best practices in the community (a survey of existing approach had been made and the "best of all worlds" kept in mind in our design). It provides a common set of base classes which establish the interface between event generators and the simulation and handles most of the bookkeeping associated with a simulation run. This streamlines the process of integrating and configuring an event generator within our software chain. Developers implement two classes: the interface for their event generator, and their event record. They only need to loop over all particles in their event and push them out into the event record. The framework is responsible for vertex assignment, stacking the particles out for simulation, and event persistency. Events from multiple generators can be merged together seamlessly, with an event record which is capable of tracing each particle back to its parent generator. We present our work and approach in detail and illustrate its usefulness by providing examples of event generators implemented within the STAR framework covering for very diverse physics topics. We will also discuss support for event filtering, allowing users to prune the event record of particles which are outside of our acceptance, and/or abort events prior to the more computationally expensive digitization and reconstruction phases. Event filtering has been supported in the previous framework and showed to save enormous amount of resources – the approach within the new framework is a generalization of filtering.

1. Introduction

As the STAR collaboration plans for its evolution into the eSTAR experiment at an anticipated electron-ion collider at RHIC, we are evaluating the suitability of our software framework to perform these studies and eventually support operations. Of immediate concern was our ability to handle the larger set of event generators required to support both ongoing operations and the upgrade studies. To date, our simulation framework[1] has only needed to support a number of pp, pA and AA generators. The requirement to support ep and eA collisions demands additional flexibility and consistency in both the interface to the event generators and the persistency of events in the event record.

Frameworks in other experiments were studied in order to evaluate their suitability for use in our environment, and their compatibility with our simulation and embedding workflows. The ALICE[2], ATLAS[3], CMS[4] and FairRoot[5] frameworks were examined, and we considered the HepMC package[6] for event persistency. In all of the frameworks, there are clear albeit different divisions of responsibility for steering the simulation, interfacing the generators to the Monte Carlo, and I/O models. FairRoot's model for steering the simulation, which assigns the responsibility to a dedicated class, was thought to be more compatible with our existing framework than ALICE's approach of building a facade around each event generator. While we liked the capabilities to replay events from persistent files provided by FairRoot, and HepMC provides a flexible and robust model for storing particle-wise information, we preferred ALICE's I/O model whereby the event-wise (specific to each generator) and particle-wise information is saved into ROOT[7] TTrees. Rather than accept the trade offs in adopting an existing framework, the decision was undertaken to develop our own. Based on the lessons learned during our survey, we developed several design goals:

Interface The end-user should be presented with a single-pass "plug-and-play" experience. Configuration of the event generators, and steering the simulation workflow, should be done using ROOT macros. The configuration of common behaviors across event generators, e.g. beam species and energies, should be through a common interface.

Integration The task of integrating new event generators in the framework should be as streamlined as possible. The number of classes a developer needs to implement should be small, and they should derive from well documented base classes. These base classes should define common base functionality for all event generators: definition of collision configuration, adding particles to the event record, handling event persistence, etc...

Upgrade Path The new event generator framework should facilitate developing a new Virtual Monte Carlo (VMC)[8] application for the STAR. It should utilize VMC base classes for establishing the particle stack, while simultaneously supporting our current FORTRAN simulation package.

Event Record Particle-wise and event-wise information should be stored in a persistent event record using ROOT TTrees. End-users should be able to analyze the event record without having to load additional libraries. Event records defining the pp, pA, AA, ep and eA kinematics should be provided, which developers can extend to include additional information from specific event generators.

Event Mixing Users should be able to mix events from different event generators, read TTree files and replay events from previous runs, and modify particle lists before stacking particles for simulation. The event record should provide bookkeeping data so that the particles can be traced back to their parent event generator.

Filtering Our current framework provides for filtering of particles and events. We should retain and improve this functionality, allowing users to implement filters which remove particles which are outside of the detector's acceptance and to reject events which are unlikely to satisfy trigger conditions and/or offline analysis cuts.

2. Design

The responsibilities for steering, event persistency and interfacing with external generator packages is divided between three main classes, and several support classes, in the STAR framework. These classes are summarized below.

2.1. StarPrimaryGenerator

The event generation process is steered by the `StarPrimaryGenerator` class. It estabilishes the interface between the concrete, user-defined event generators and the STAR simulation chain.

When control is passed to this class, it calls all declared event generators in turn. Particles are associated to one or more event vertices as required for the simulation. Event records from the various event generators are aggregated, with additional bookeeping information appended to trace the origins of the particles back to the parent generator. One or more user-defined filters may be applied at this point, pruning the event record of particles outside of the acceptance, or even aborting events which are unlikely to satisfy selective triggers. Final state particles are then pushed out to GEANT[10] for simulation of the detector response.

2.2. StarGenerator

The interface between the primary generator, the concrete event generators, and the generator-specific event record is completely established by the abstract base class `StarGenerator`. To integrate a new event generator, the developer only needs to provide a concrete implementation of `StarGenerator`, provision it with the necessary functionality to exercise the event generator's machinery, and fill in the event record. Specifically, the developer must implement the following:

Configuration The `StarGenerator` class provides base functionality for configuring the behavior of the event generator. Specifically, standard methods are provided for the definition of the particle species, energies and interaction frame. The developer should either override the default behavior of these methods, or configure the base functionality when the generator is initialized by the chain. Any configuration methods beyond the base functionality will need to be implemented as additional methods.

Generate A Generate() method must be implemented by the developer. It is responsible for exercising the event generation machinery of the concrete event generator, looping over all particles, and filling the event record.

Event Record The base class specifies four methods for filling event records which correspond to pp, ep, AA and eA events. The developer provides code to fill these event records where it is appropriate, e.g. PYTHIA 6[11] implementations are expected to fill pp and ep event records, whereas HIJING[12] implementations would be expected to fill pA and AA records.

Particle Record The developer should also push particles out to the event record, taking care to translate the event generator's private particle IDs to the PDG convention adopted by the framework. The status of each particle, e.g. final state vs undecayed, should also be registered according to the HepMC standard.

Interface The most difficult task faced by the developer will be implementing an interface to FORTRAN-based event generators. In general they will need to expose the FORTRAN common blocks and subroutines defined by the event generator to C++.

2.3. StarGenEvent

Event records are responsible not only for storing the list of particles and their mother-daughter relationships in an event, they are also responsible for recording the diverse event-wise information provided by each generator. End-users may wish to study, for instance, how their triggers bias the contributions of different parton-level processes to the total cross sections for jet production in pp collisions, or in the number of spectator nucleons produced in an eA collision. Event records should be able to provide this information for analysis.

STAR has implemented a model for storing the event record which allows the developer the maximum flexibility in saving event-wise information, and permits the user to analyze the particle-wise information without having to load additional shared libraries. The main classes in the event record are illustrated in figure 1. `StarGenEvent` is the base class, which provides common functionality across all event records. It stores, for instance, the list of particles created by the event generator(s), provides information on the collision configuration and subprocess created by the generator, and adds bookkeeping members enabling better tracking of filtering

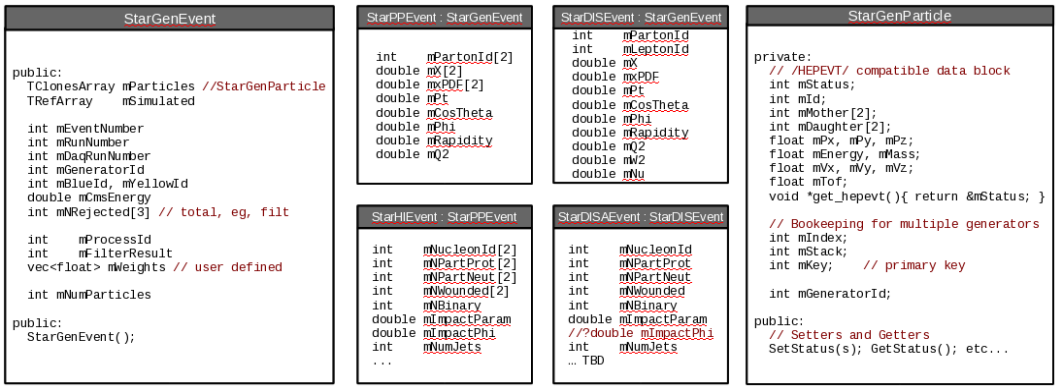

Figure 1. The `StarGenEvent` event record provides persistency for common header information and particles. Derived classes represent the different kinematics of pp, pA, AA, ep and eA collisions. A lightweight particle class is also provided, extending the HEPEVT standard with additional bookkeeping values which enable us to trace the particle back to the parent generator.

results. Particle-wise information is saved in a lightweight class based on the HEPEVT[13] standard, with additional bookkeeping members added. This enables the particles with their mother-daughter relationships to be analyzed by reading the TTree, without loading additional libraries. When integrating a new event generator, it is anticipated that the developer will derive their event record from one of the four records provided, which adds the relevant kinematics of the simulated process to the event record. Thus, all of the information which a user may be interested in in studying a simulated event can be provided.

3. Filtering

High energy physics cross sections feature dramatic variations in scale, for example falling many orders of magnitude within the transverse momentum range measured at STAR. Comprehensive sampling of these cross sections becomes prohibitively expensive: the majority of computational resources are expended drawing events from low p_T. The majority of these events will be discarded by offline analysis as they try to compare the simulated data samples with real data where high-p_T thresholds are imposed by the online trigger.

Filtering enables us to examine the generated event at early stages in the simulation, and stop processing it if it is unlikely to be of use in offline analysis[9]. This saves us from passing unnecessary events to the more computationally demanding parts of the simulation chain. This is illustrated in figure 2. Events are passed from the primary generator to the first filter, labeled filter A. Here, loose cuts can be applied to the event record to determine whether the event may pass the offline trigger cuts. If it does, the event is fully simulated by GEANT and a full simulation of the trigger response performed. Filter B is then applied to determine whether the event passes the offline trigger. If and only if the event passes the offline trigger is it passed along to the most time consuming elements of the chain. Filtering has been used in several STAR simulations to enhance high p_T jet and dijet yields, and to provide high-statistics background simulations for prompt-photon, W and Z boson production.

4. Summary and Conclusion

The STAR experiment at Brookhaven National Lab continues to pursue a rich and diverse experimental program utilizing the Relativistic Heavy Ion Collider. To better support these

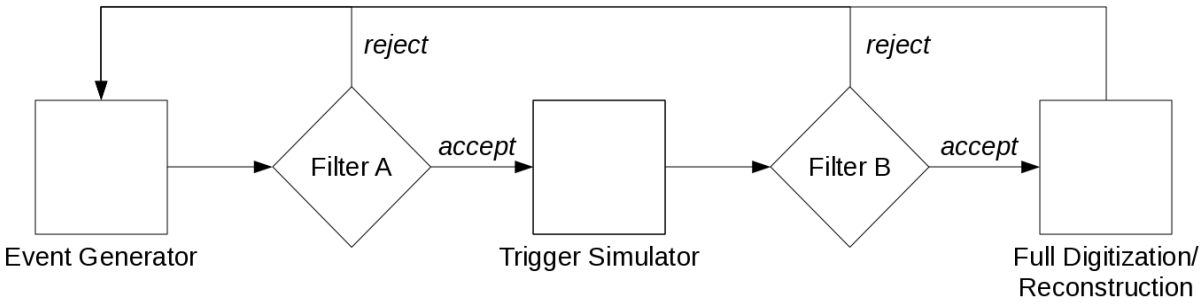

Figure 2. Illustration of filtering. Two filters are applied. Filter A runs immediately after the event generator, examining the event record and performing a loose cut on events which are unlikely to satisfy trigger conditions. This significantly reduces the number of events which the Monte Carlo is required to simulate. Filter B executes after the MC runs, and the detailed response of the trigger detectors have been simulated. Now tighter cuts can be applied which closely emulate the behavior of the online trigger. Events which do not satisfy the trigger are aborted, saving the time required for the detailed simulation of tracking detectors, and the reconstruction of tracks in STAR.

programs, and with the plans for eSTAR at eRHIC in mind, we have developed a new framework for event generation which leverages our existing simulation package and lays the groundwork for the development of a new virtual MC application. The "Plug-and-Play" framework streamlines for developers the process of integrating new event generators into the STAR software stack by establishing a simple and well-defined interface to the simulation package. Event generators can then be added to the simulation chain by users, who need to run only a single pass of a single application to obtain their results. More complicated use cases are supported. Events from multiple event generators can be mixed, either online or from ROOT files. Filters can be used to reject both particles and events, to save time and disk space from filling events which will never be considered in an analysis. And a complete record of the simulated event is stored in a ROOT TTree, which can be analyzed without dependence on external libraries.

References

[1] P. Jacobs and D. Irmscher, *GSTAR: A Geant-based Detector Simulation Chain for STAR*, STAR internal note 0235, see also https://drupal.star.bnl.gov/STAR/starnotes/public/sn0235.
[2] http://aliweb.cern.ch/Offline/Activities/Simulation/index.html
[3] The ATLAS Collaboration 2005 ATLAS computing : Technical Design Report (Preprint ATLAS-TDR-017, CERN-LHCC-2005-022) Athena Core software http://cern.ch/atlas-computing/packages/athenaCore/athenaCore.php
[4] https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideSimulation
[5] M Al-Turany et al 2012 J. Phys.: Conf. Ser. 396 022001.
[6] M. Dobbs and J.B. Hansen, Comput. Phys. Commun. 134 (2001) 41.
[7] Rene Brun and Fons Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Nucl. Inst. Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/.
[8] I. Hrivnavova, et al., *The Virtual Monte Carlo* 2003 Proc. of Computing in High Energy and Nuclear Physics (La Jolla) p THJT006 (arXiv:cs/0306005)
[9] M. Betancourt, et al., *Targeted Monte Carlo: Filtered Simulations at STAR*, STAR internal note 523.
[10] *GEANT - Detector Description and Simulation Tool*, CERN, Geneva 1993.
[11] T. Sjostrand, Stephen Mrenna, and Peter Skands, *PYTHIA 6 Physics and Manual*, JHEP 0605:026,2006.
[12] X. Wang and M. Gyulassy, *HIJING: A Monte Carlo model for multiple jet production in p p, p A and A A collisions*, Phys.Rev.D 44, 3501 (1991).
[13] T. Sjstrand et al., in 'Z physics at LEP 1', eds. G. Altarelli, R. Kleiss and C. Verzegnassi, CERN 89-08 (Geneva, 1989), Vol. 3, p. 143