Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Cross-platform validation and analysis environment for particle physics

S.V. Chekanov [a,*], I. Pogrebnyak [b,a], D. Wilbern [c]

[a] HEP Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA
[b] Department of Physics and Astronomy, Michigan State University, 567 Wilson Road, East Lansing, MI 48824, USA
[c] Northern Michigan University, 1401 Presque Isle Avenue, Marquette, MI 49855, USA

ARTICLE INFO

Article history:
Received 27 October 2015
Received in revised form 16 May 2017
Accepted 25 June 2017
Available online xxxx

Keywords:
Monte Carlo
Format
IO
Analysis software

ABSTRACT

A multi-platform validation and analysis framework for public Monte Carlo simulation for high-energy particle collisions is discussed. The front-end of this framework uses the Python programming language, while the back-end is written in Java, which provides a multi-platform environment that can be run from a web browser and can easily be deployed at the grid sites. The analysis package includes all major software tools used in high-energy physics, such as Lorentz vectors, jet algorithms, histogram packages, graphic canvases, and tools for providing data access. This multi-platform software suite, designed to minimize OS-specific maintenance and deployment time, is used for online validation of Monte Carlo event samples through a web interface.

Published by Elsevier B.V.

1. Introduction

Development of data-analysis software is one of the most effort consuming components of nearly every project in high-energy physics. Consequently, engineering of analysis packages which require minimum time for deployment, maintenance, and porting to new platforms, is an important direction, especially for long-term projects and projects without dedicated funding for computer support. This is precisely the category that includes detector performance and physics studies for next-generation circular colliders. Currently, these projects are community driven and often do not have dedicated computer infrastructure needed for detailed exploration of physics using large samples of Monte Carlo (MC) events. Therefore, a software environment and deployment model that can help reduce maintenance effort present an unambiguous advantage. Conversely, the exploratory nature of future collider studies provides for a good setting for testing of innovative computation models.

This paper discusses a validation and analysis framework that is built from the ground to work with the HepSim data repository [1] that stores Monte Carlo predictions for HEP experiments. To ensure platform independence and support data processing using a web browser, the analysis environment runs on the Java platform, which also allows for the end-user analysis scripts to be written in Jython, a Java implementation of Python. This setup creates a familiar analysis environment, since Python is often used by the current LHC experiments, and Java syntax is very similar to that of C++. Therefore, such a framework has a potential to target a larger audience within HEP community, simplifying and facilitating exploration of physics for the next generation of collider experiments.

It should be pointed out that the same approach has been proposed for studies of e^+e^- collisions back in 2000, when the first version of FreeHep and AIDA (Abstract Interface for Data Analysis) was developed [2]. Unlike FreeHep which is primary designed for linear e^+e^- collider studies, HepSim software is also used for circular colliders, including pp colliders. HepSim analysis framework is similar to PyROOT and is fitting for Python programming, with classes named conveniently to reduce code verbosity. As discussed in this paper, HepSim also includes jet algorithms which are popular for pp collisions.

The choice of Java and Python for HepSim framework offers some key advantages, such as

1. hardware, operating system, and library independence (except Java itself), making deployment trivial and effortless,
2. automatic memory management, eliminating frustratingly common sources of errors, and
3. simple, compressed, and lossless data storage format.

* Corresponding author.
E-mail addresses: chekanov@anl.gov (S.V. Chekanov), ivanp@msu.edu (I. Pogrebnyak), dwilbern@nmu.edu (D. Wilbern).

http://dx.doi.org/10.1016/j.cpc.2017.06.017
0010-4655/Published by Elsevier B.V.

These features eliminate all portability issues and give HepSim an edge in terms of programmer productivity, allowing researcher to concentrate more on physics and less on bookkeeping. Due to optimization provided by Java Virtual Machine employing just-in-time compilation technology, HepSim remains competitive in terms of data processing speed with natively compiled analysis software.

To access data from the HepSim Monte Carlo event catalog [1], one needs to download a small (25 MB) software package. The software requires Java version 7 or above. The package includes Java classes for data access (`browser_promc.jar`), physics and graphical classes (`hepsim.jar`), and a complete Jython distribution (`jython.jar`) with required modules (`modules.jar`). To minimize the package size for the online usage, only essential Python modules are used. The software framework can be used on Windows computers, but here will discuss Linux/Mac OS for simplicity. Bash shell users can download and setup the framework by running the following commands:

```
wget http://atlaswww.hep.anl.gov/hepsim/soft/hs-toolkit.tgz -O - | tar -xz
source hs-toolkit/setup.sh
```

The last command sets up the working environment. One can find HepSim commands by using the `hs-help` command that prints this help message:

```
HepSim toolkit version: 2
Commands:
=========
hs-find      - find a URL of a MC sample
hs-get       - download MC samples in multiple threads
hs-info      - validate the ProMC file or dump separate events
hs-meta      - read meta information
hs-ls        - list all files from a given data sample
hs-ide       - run an editor and process analysis *.py script
hs-run       - process analysis *.py script in a batch mode
hs-view      - view ProMC file from the HepSim database
hs-exec      - execute external commands
hs-extract   - extract N events from a file and write them to other file
hs-pileup    - create a new file mixing signal events with pile-up events
hs-distiller - convert ProMC file to zip64 format (distiller)
hs-help      - shows this help
```

As an example, one can run Python code using the command `hs-ide test.py`, or in a batch mode using `hs-run test.py`.

Alternatively, the framework can be used through a Java Web Start interface. In this case, it uses a minimal version of the HepSim package with simplified Jython and reduced sizes of Java jar files.

This paper discusses how to access Monte Carlo (MC) datasets stored in HepSim using the Python interface or Java, manipulate data containers and perform a full scale analysis using local files, or streaming the files over the network. Our discussion mainly focuses on *pp* collision events, but the examples given in this paper can also be used for other particle collision experiments.

2. Data access

The main Java package for data access is called `hepsim.HepSim`. It includes static Java classes for retrieving MC data lists from the HepSim catalog, and streaming event records over the network (when reading data using URL).

Here is a simple Python script example to list HepSim MC files using the name of a dataset.

```
from hepsim import HepSim
url   = HepSim.urlRedirector("tev100pp_ttbar_mg5")
flist = HepSim.getList(url)
print flist                         # print list of files for this dataset
```

Run this example as `hs-run example.py`. Alternatively, use the editor `hs-ide example.py`. If you know the direct URL link to the MC dataset, use this example:

```
from hepsim import HepSim
flist = HepSim.getList(url)         # url points to dataset
print flist
```

If MC datasets have been downloaded to the local file system as discussed in [1], one can build a list of files with:

```
from jhplot.utils import FileList
flist = FileList.get("./data","promc")
```