EPJ Web of Conferences **214**, 02009 (2019) https://doi.org/10.1051/epjconf/201921402009
CHEP 2018

HDF5 and row-wise ntuple in analysis tools in Geant4 10.4

Guy Barrand[1,*], *Ivana* Hřivnáčová[2,**]

[1]Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud, CNRS-IN2P3, Orsay, France
[2]Institut de Physique Nucléaire (IPNO), Université Paris-Sud, CNRS-IN2P3, Orsay, France

Abstract. The analysis category was introduced in Geant4 release 9.5 to help users capture statistical data in the form of histograms and ntuples and store these in files in various formats. Up to release 10.3 the following formats had been introduced: csv, AIDA/XML and the binary ROOT file format. We present here the work done to handle, in Geant4 10.4, the binary HDF5 file format, a format/library widely used in other domains of science but quite ignored in HEP for the moment. Work has been done also to support the management of a single file in a multi-thread or MPI parallel environment for the ROOT format; we present the introduction of a row-wise way to manage paginated ntuples in order to restore an "event view" lost by our today column-wise implementation for this format.

1 Introduction

We remind the reader that the Geant4 [1] analysis category is organized in two layers; first the g4tools "basic layer" (agnostic of Geant4 core) that provides histograms, profiles, ntuples (and plots since Geant4 10.3) along with the mechanisms to store these in file in various formats by bringing only what is needed to do that; and second the Geant4 analysis "upper layer" to help users manage collections of these, putting the accent on a uniform, user-friendly interface integrated in the Geant4 user API, by offering, for example, interactive commands, units and (in)activation of selected objects. We refer to [2], [3], [4], [5], and [6] for an overall presentation of these layers. Releases of the g4tools are available, since June 2018, on GitHub [7].

As Geant4 is related to domains of science other than HEP, where the HDF5 [8] library is used, it seemed interesting to us to add this file format to the ones already handled (ascii csv, AIDA/XML and binary ROOT [9]). About HDF5 itself and for a breakdown of users, we refer to [8] but quote the short presentation in the support section: HDF5 is a data model, library, and binary file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analysing data in the HDF5 format.

*e-mail: barrand@lal.in2p3.fr
**e-mail: ivana@ipno.in2p3.fr

2 File organization

With g4tools/HDF5, a file is organized in a tree like structure similar to a file system. We use the `H5group` entity to build this hierarchy. `H5group` can have `H5group` children and the name of a `H5group` can be then as the name of a directory in a file system. The HDF5 library provides tools to find, at read time, an `H5group` given the access path to it given in a UNIX-like "slash separated" syntax. Some `H5groups` are considered "leaf groups" and contain a set of the `H5dataset` storing data of a given object. If the user data is an instance of a C++ class, in general, we have one `H5dataset` per field. A leaf `H5group` has a name which is considered as the "storage id" of the object. This file organization is similar to the ROOT file format, where `TDirectory` can be used to organize data (`TKeys`) in a tree like structure. Note that in g4tools, histograms are not named (but have a title), and the storage id is given when writing/converting the object to a pair (`H5group`, `H5datasets`) when written to file. This differs from ROOT where the storage id is the `TName` of the object.

For a given object (for example an histogram), there is a lot of ways to store it in HDF5. We could have defined a "compound datatype" (`H5compound`) that does a one to one mapping of the class/fields to a structure and then writes one histogram in a single `H5dataset` at once by writing the compound. This would be more effective, but at the loss of a more transparent visibility from reader tools. For the moment, in this first approach to HDF5, we prefered to give priority to simple data schemas.

3 More on ntuples

For us, an "ntuple" (or HEP-ntuple) is defined as a paginated table where each column can be then attached to a pagination mechanism. Contrary to a histogram, a storage system is attached to the ntuple when created. At fill time, leaf column data (of a simple type such as `int, float, double`, or vector of simple types) is written in a page, and when full, the page is written on disk by using the attached IO package. This simple mechanism, using a fixed amount of memory (one page per column), allows a large amount of data to be collected on disk.

The ROOT file format has the `TTree/TBranch/TLeaf` which implement this kind of paginated table (with a `TBranch` managing data pages which are implemented as `TBaskets`) and the g4tools ntuple writter for the ROOT file format arranges to map this structure in files. For the g4tools HDF5 ntuple, we use a similar mechanism with the ntuple attached to a master `H5group`, columns attached each to a `H5group` within the master group and pages stored in `H5datasets` put within each column `H5group`.

4 Geant4 analysis and HDF5

In order to support HDF5, the Geant4 analysis category was enhanced with the hdf5 sub-category, providing the implementation of the `G4AnalysisManager` and `G4AnalysisReader` types for this new format. Building these sub-category classes requires the HDF5 libraries installation as well as Geant4 libraries built with the CMake option:

```
-DGEANT4_USE_HDF5=ON
```

User can then choose the HDF5 format by including the provided `g4hdf5.hh` file and use the `G4AnalysisManager` and `G4AnalysisReader` in the same way as for the other supported formats.

5 Performance

I/O performance to store histograms and ntuples is not an issue in the context of Geant4 where writing these on file is not a highly intensive operation compared to an overall simulation. If needed, for histograms, we could introduce a "compound" mode which would be more effective.

For ntuple, the pagination and compression logic being similar to the `TBasket` of ROOT, if choosing the same size and compression level for them, we observed similar file sizes and write times since, in both cases (HDF5 versus ROOT/IO), we are anyway dominated by the speed of writing pages on disk. (For compression, with HDF5, it is done on the `H5dataset` of a page by using the same zlib than for ROOT/IO which is applied on the `TBasket` inplementing a data page).

We emphasize that this conclusion is valid for us for histograms and paginated ntuples, but that it may be different in attempting to store the event model of a complex HEP detector. So far we did not have the opportunity to attempt storing this kind of information, but HDF5 looks sufficiently flexible to be not discarded as a candidate package for such purpose.

6 Parallelisation

For the ntuple at ROOT format, in order to manage one file for multiple threads or workers, we introduced [2] a mechanism to pass pages of data from one thread/worker to the one managing the file. It works with multi-threads and MPI. For the moment, we did not attempt to do the same for HDF5. HDF5 having some logic of its own for parallelisation, it is not so clear for us if we have to do something specific here. This is to be explored.

7 Examples and tests

In `g4tools/test/cpp`, available on GitHub [7], we provide :

- `hdf5_histos.cpp, hdf5_ntuple.cpp` to show how to write/read histograms and ntuples.
- `hdf5_threads.cpp` to show how to write ntuples in threads (by using a thread safe build of the HDF5 library).
- `cern_root_hdf5_ntuple.cpp` to show how to read an ntuple, do a projection and plot by using a ROOT TH1D and a TCanvas.

In the Geant4 B4 and B5 examples, the selection of the output format takes place by including one of dedicated header files, `g4root.hh` being used by default. Using `g4hdf5.hh` will then switch from the default ROOT output to HDF5, other options are `g4csv.hh` for CSV and `g4xml.hh` for AIDA XML. The Geant4 test code for the analysis category was enhanced with the HDF5 option and a dedicated test using this option was added in the standard Geant4 regular testing.

8 Applications to read files

The following applications can be used to read the files: HDF5/h5ls, HDFView, ioda(1.14x) and gopaw.

- HDF5/h5ls permits one to have a glance at the `H5group` tree structure in a file.

- HDFView also has the same functionality. In addition, HDFView is agnostic of g4tools histograms but has a plotting feature that, when applied on the `H5dataset` of bin weights, permits to have a glance of the bins. See figure 1. HDFView is available at [8].
- ioda (1.14.x) can read histograms and ntuples of g4tools HDF5 files. It offers more plotting options and styles than h5ls and HDFView.
- gopaw, for the "Good Old PAW", was introduced in June 2018, in time for the CHEP 2018 conference (see related poster). It is an application which can read the ROOT and HDF5 files produced by g4tools, plot histograms and do ntuple projections in a way similar to CERN PAW. See figure 2.

For ntuple, ioda and gopaw use an implementation that permits to read data in an effective way by importing in memory, at need, one page of data per column (a page being the same concept as described in 3 for writing). ioda and gopaw are also now available on GitHub [10], [11].

9 Row-wise ntuple for parallelisation for the ROOT file format

On users requests we handle one file when filling an ntuple (with same booking) from different workers (threads or MPI/ranks). It is done with column-wise ntuples in the ROOT format in 10.3 by exploiting the page/`TBasket` logic of a column/`TBranch`. For the moment, a page is sent to the master worker (the one handling the file) immediately after being full. But, as the pages arrive from workers in random order for a given column, we loose an "event point of view" when reading back the data. (See figure 3)

We then introduced a row-wise mode to sort out this case. In this mode, columns are leaves of a single `TBranch` attached to each ntuple per worker. A page contains then a set of "row" which could be interpreted consistently as an "event" when reading data from the single file. (See figure 4). This mode had been introduced for multithreads for the ROOT format in Geant4 10.4. Row-wise merging with MPI is foreseen for 10.5.

This approach has the inconvenience that if column-wise scheme is used in sequential mode, but row-wise is used in parallel mode, the user will obtain files with different schema (organizations of `TBranches` and `TLeaves`), which may complicate to read the data. To resolve this we intend, in the future, to consider a solution proposed by the ROOT team: mainly arrange to stage pages per worker, await a set of "consistent filling of few rows" and then send at once worker pages to the master. This restores an "event view", even in column-wise writing mode.

10 Conclusions

Having introduced the HDF5 library, dedicated to I/O and already used in other domains of science, allows for an effective treatment/representation of statistical data within the Geant4 context, by providing a clean, light and portable engineering solution to this problem.

References

[1] Agostinelli S et al, *Nucl. Instrum, and Methods*, **A506**, 250-303 (2003)
Allison J et al, *IEEE Transactions on Nuclear Science*, **53 No. 1**, 270-278 (2006)
Allison J et al, *Nuclear Instruments and Methods in Physics Research*, **A835**, 186-225 (2016).

[2] Hrivnacova I, Barrand G, *J. Phys.: Conf. Ser.*, **898**,042018 (2017)

[3] Hrivnacova I, *J. Phys.: Conf. Ser.*, **513**,022014 (2014)
[4] Barrand G, *J. Phys.: Conf. Ser.*, **513**, 022002 (2014)
[5] https://geant4.web.cern.ch. See the User's Guide for Application Developers section.
[6] http://softinex.lal.in2p3.fr
[7] https://github.com/gbarrand/g4tools
[8] https://portal.hdfgroup.org
[9] http://root.cern.ch
[10] https://github.com/gbarrand/ioda
[11] https://github.com/gbarrand/gopaw

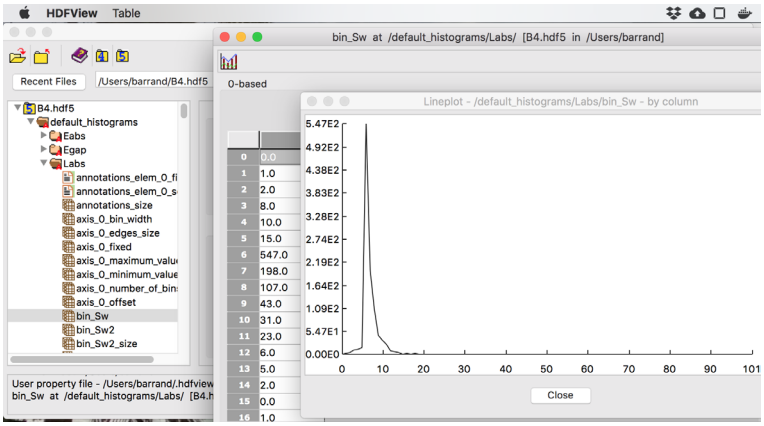
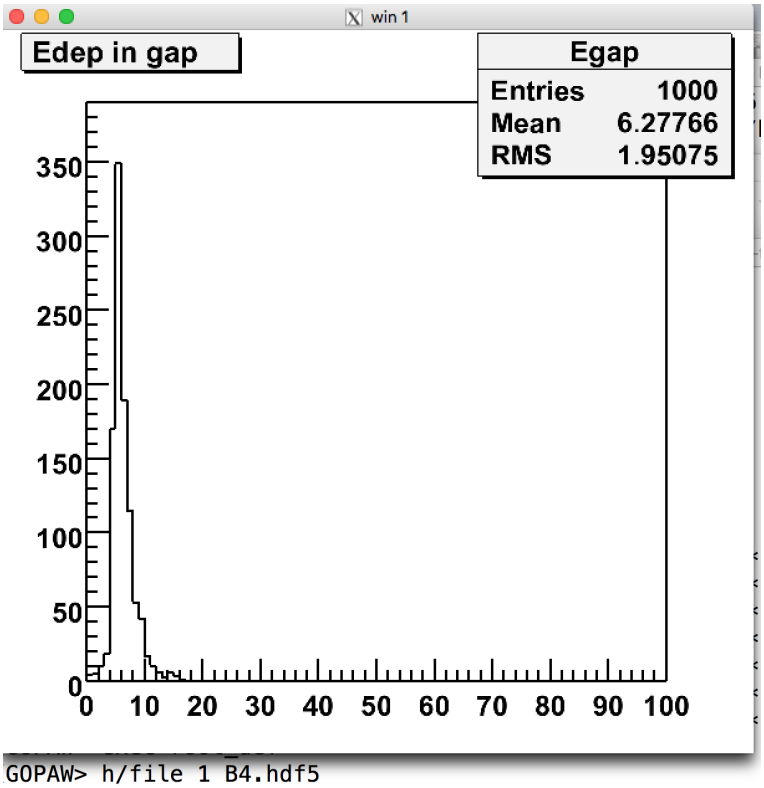


Figure 1. HDFView of the B4 HDF5 file




```
GOPAW> h/file 1 B4.hdf5
GOPAW> cd //lun1/default_histograms
GOPAW> opt stat
GOPAW> set *wid 5
GOPAW> h/plot Egap
```

Figure 2. gopaw on the B4 HDF5 file

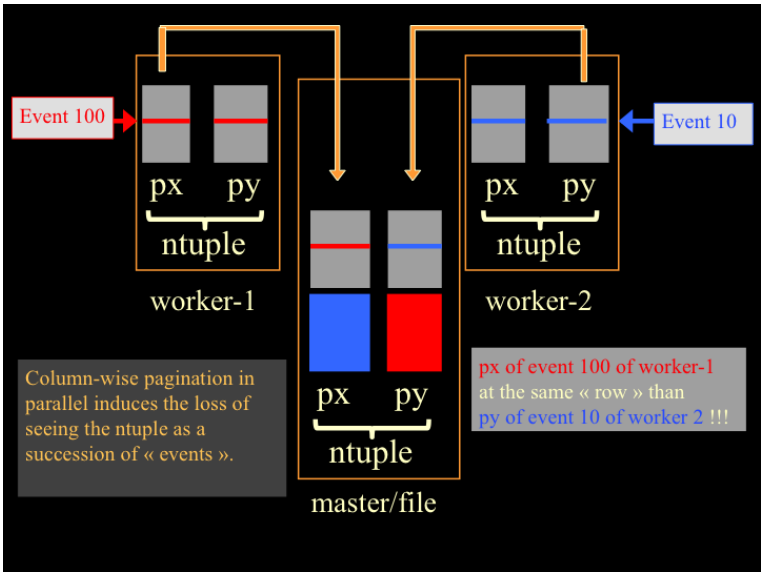


Figure 3. // pages column-wise

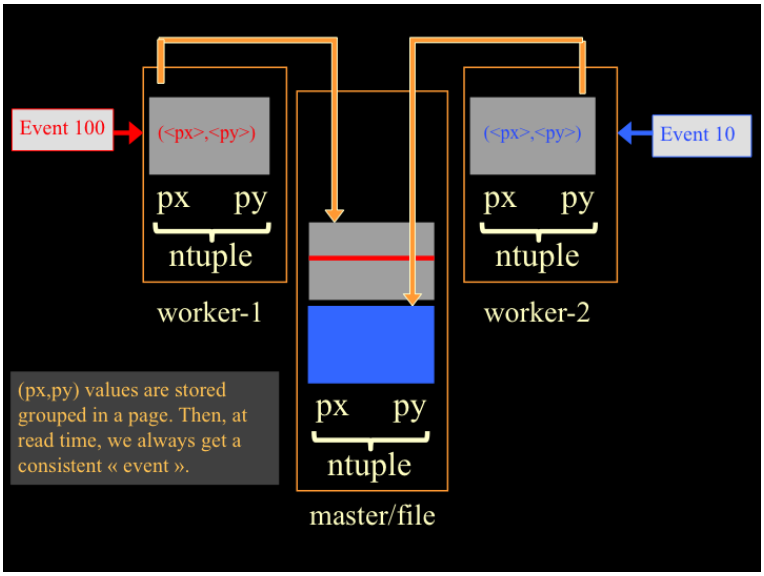


Figure 4. // pages row-wise