VEGA, an environment for gravitational waves data analysis

D. Buskulic, L. Derome, R. Flaminio, F. Marion, L. Massonet,
B. Mours, R. Morand, D. Verkindt, M. Yvert

LAPP Université de Savoie, B.P. 110
74941, Annecy-le-Vieux Cedex, FRANCE

A new generation of large scale and complex Gravitational Wave detectors is building up. They will produce big amount of data and will require intensive and specific interactive/batch data analysis. We will present VEGA, a framework for such data analysis, based on ROOT. VEGA uses the Frame format defined as standard by GW groups around the world. Furthermore, new tools are developed in order to facilitate data access and manipulation, as well as interface with existing algorithms. VEGA is currently evaluated by the VIRGO experiment.

1. General Overview

VEGA is an acronym for ”Visual Environment for Gravitational waves data Analysis”. This environment is based on the ROOT framework [1] developed at CERN. It is designed to ease gravitational data access and management, by integrating and using the Frame format that was agreed upon the two experiments VIRGO and LIGO. It integrates the FrameLib[2] software developed to handle Frame format data and defines new ways of accessing individual information, such as frames or vectors, by means of a database. Various pieces of software have been integrated as modules, such as the GRASP[3] package.

Furthermore, the ROOT environment was extended with specific code written for gravitational waves data analysis. We foresee to extend VEGA with external signal analysis libraries.

2. The ROOT framework

2.1. Presentation

The ROOT framework was initially designed for High Energy Physics data handling, analysis, simulation and online work. It is object oriented, includes a re-implementation of most of the CERNLIB in C++, inheriting for example it’s statistical tools. It represents around 500 000 lines of code.

*Submitted to International Journal of Modern Physics D
The characteristics of ROOT are the following:

- a C/C++ interpreter is used as a command and macro language,
- it includes full featured graphical and interactive analysis and presentation tools
- it is primarily designed to handle very big amounts of data, at the Peta-Byte scale
- it has networking tools to allow remote access of files
- parallel computing facilities are some of the important features in development
- it includes code documentation facilities
- it is free and open source

ROOT is running on most Unix platforms, it was tested on around 30 platform/compiler combinations, as well as on Windows.

2.2. The CINT Interpreter

The interpreter that is included into ROOT is called CINT. It interprets 95% of the C standard and 90% of C++. Though not meant to be fully compliant to the standards, it is able to interpret itself (around 90000 lines of code).

Using a C interpreter as a command and macro language means that the interactive mode language is the same as the macro language, which in turn is the same as the programming language. One can do a fast prototyping with macros, and if they are too slow when interpreted, just compile it.

Integrating an existing library in the framework is easy by construction. All the variables/classes/functions defined by the user may be accessible at the command line or in a macro. All this, obviously, means an open architecture of the software. Optionally, this is an easy way to learn C and C++.

3. The Database

From the user’s point of view, the analysis environment should allow her/him to access simply any vector or frame that is contained in a set of frame files, even remotely. One should also be able to access vector not caring about frame boundaries. It should also be possible to make accesses conditioned by some trigger, slow monitoring value or user-defined condition.

Furthermore, one should be able to manage a set of files that is as big as possible (of the order of 1 TB or more)
The option chosen in VEGA to solve this problem is to build a database that contains metadata about frames and indexes these frames. The structure of the database is drawn on Fig. 1.

The complete frame information is kept in the frame files, while the database serves as an index to access them. The access is a two step process. First, one gives the starting time of the vector one wants. This starting time is used to determine, with the help of what is called a ”time hash table”, the frame files that possibly contain the desired information. Then, only the metadata corresponding to these files are searched and finally, the desired frame is accessed in the relevant file or the desired vector is reconstructed and given to the user.

The metadata is kept in a container, called Tree, introduced in ROOT and specifically designed to handle big amounts of data and access it quickly. The tree structure can also contain conditions for frame access.

4. User interaction with the environment

In VEGA, the user’s interaction with the environment is made through the CINT interpreter. Almost all the commands are C/C++ statements of the type:

\[
\text{object.action(parameters)}
\]
For example, building a database is made with a statement like:
```
db = new VFrDataBase("DBName.db","CREATE","path/files*")
```
Thanks to the ROOT build-in tab completion, one doesn’t have to type long names of objects or classes. One just types VFrD<tab> and gets VFrDataBase.

Extracting a vector of any length from the set of files indexed by the database is done with:
```
vec = db.GetVect("adc.ADCNAME",starttime,length)
```
Where `vec` is a structure of type FrVect* which is automatically declared. One can extract any type of vectors (adc, proc or sim, different FrameLib types), which can be of any length, not caring about frame boundaries.

The performances of the database were tested and found to be satisfying to handle 10 million frames, which is around 1 TB, without putting undue overhead from the computing time point of view (around 2 % of the frame access time).

5. Slow monitoring data management

Slow monitoring (or environmental) data spans a large number of frames, and may even be missing in some of them. A special treatment was foreseen for this type of data.

The slow monitoring data is transferred to a standard container used in high energy physics, the ntuple. This is a special simplified tree which contains simple (float, double) data. Tools are available to handle, draw, select, parts of the data contained in the ntuple.

For example, drawing some variables from an ntuple can be made with statements such as:
```
nt.DrawGraph("t:sqrt(MYSMS.VF1-MYSMS.VI2)",
"MYSMS.VI2<0 && sqrt(MYSMS.VF1-MYSMS.VI2)>50","APT")
```
It should be noted that the only parts of the statement imposed by the analysis environment are the ones in bold. The rest is physics requirements. Here, one has two variables MYSMS.VF1 and MYSMS.VI2 and one plots
```
\sqrt{MYSMS.VF1 - MYSMS.VI2}
```
with the condition
```
MYSMS.VI2 < 0 && \sqrt{MYSMS.VF1 - MYSMS.VI2} > 50
```

6. Reference time

GPS time, used by gravitational wave detection experiments is 9 digits. This is not very handy to manipulate for everyday use! So a reference time was introduced to access, draw and manipulate the data.
Furthermore, the axis of the plots can be drawn with readable time labels.

7. Perspectives and conclusion

The next foreseen developments concern a conditioned or triggered frame or vector access, as well as remote files access. A connection to the Matlab tools is also foreseen.

VEGA’s strengths are the simplicity of interaction for the end user, the C/C++ interpreter, the ROOT input/output system and the direct link to the Framelib.

The simplicity of interaction is illustrated in some ROOT characteristics, namely the tab completion and the interactivity with the drawn objects, like for example interactive zooming.

The C/C++ interpreter allows a simplified connection to the batch system. Transforming a macro into compiled code is a matter of a one line command.

The ROOT input/output system is well suited to build big and complex OO databases.

Finally, the Framelib is included into VEGA, allowing to keep the frame file format.

Consistency across the whole data analysis is the main VEGA strength.

Software, documentation, tutorials, examples of plots, can be found on the following web address: http://www.lapp.in2p3.fr/virgo/vega

References