THE VIRGO ONLINE DATA SELECTION, A PROPOSED OFF-LINE DATA ANALYSIS ENVIRONMENT AND AN ALGORITHMS INTERFACE FOR GW DATA ANALYSIS

F. BELLACHIA, D. BUSKULIC, L. DEROME, F. MARION,
A. MASSEROT, B. MOURS, D. VERKINDT
LAPP, Chemin de Bellevue, BP 110
74941 Annecy-le-Vieux, France , email: verkindt@lapp.in2p3.fr
presented by D. VERKINDT

In Interferometric Gravitational Waves Experiments like VIRGO or LIGO, developing algorithms for off-line or online search of a gravitational wave signal needs data analysis software environments. It would also benefit of a common data format, already available, and of a common interface between algorithms and data analysis environments. We describe here an interactive data analysis environment based on ROOT, a first version of the VIRGO online data selection and a preliminary version of algorithms interface common to these two types of data analysis environment.

1 Introduction

In the near future, many physicists working on gravitational waves interferometric detectors will be involved in the development of algorithms searching for gravitational wave signal in off-line analysis and online data selection. They will use different data analysis software environments: interactive like Matlab or ROOT [1], using simulated data or not, or non interactive like heavy computing tasks (production mode) or online processing tasks (data selection, data quality). To make different algorithms written by different people from different scientific communities speak the same language, a convenient way could be the definition of a standard interface between the algorithms and the different data analysis environments.

We first describe here the scheme of the VIRGO online data selection, then a proposed interactive data analysis environment: VEGA, and finally some proposed guidelines for an algorithms interface common to those two types of data analysis environments.

2 The VIRGO online data selection

The VIRGO experiment will face a huge data flow from data acquisition [2] of, at least, 2.6 MBytes per second (see figure 1). This is about 225 GBytes per day and it makes data analysis quite unmanageable. To solve this problem, we would like to reduce the amount of data stored by the Data Distribution system and to improve the off-line data analysis performances, without rejecting any data needed for off-line analysis [3]. This data reduction, of hopefully two orders of magnitudes, will be the task of the Online Data Selection.

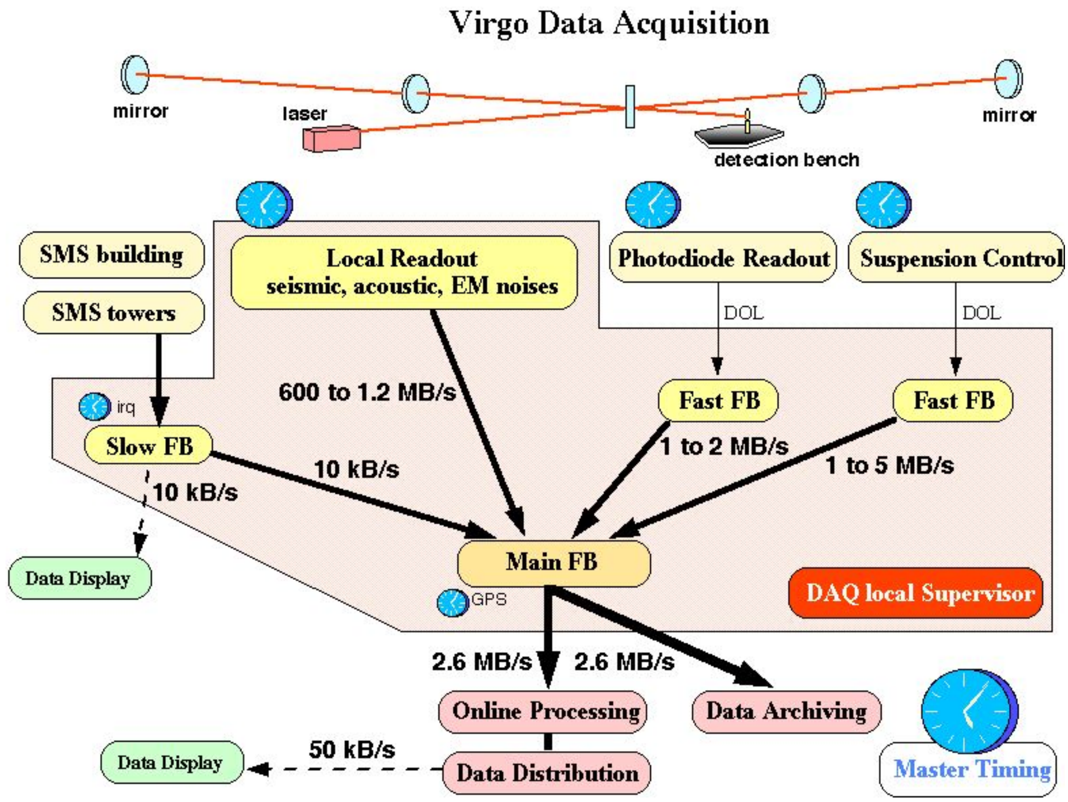

Figure 1: *Simplified scheme of the VIRGO data acquisition system. The VIRGO data acquisition is mainly composed of 3 parts: a front end acquisition (Photodiodes Readout, Slow Monitoring Stations, Local Readouts, etc...), a formatting acquisition (Fast Frame Builder, Slow Frame Builder, Main Frame Builder) in charge of putting data into the frame format, and an online processing part (Data Archiving, Data Selection, Data Distribution,...). A timing system is in charge of synchronizing mainly all the front-end acquisition.*

The VIRGO data acquisition is mainly composed of 3 parts:

1. **front end acquisition:** Photodiodes Readout recording the output signal of the interferometer, Slow Monitoring Stations for central building and vacuum control, Local Readouts for environment monitoring (ascoutic, electromagnetic, seismic noises), etc... in charge of taking data coming out of the ADC's.

2. **formatting acquisition:** Fast Frame Builder, Slow Frame Builder, Main Frame Builder) in charge of putting data into the frame format [4,5] (the common data format adopted by the VIRGO and LIGO experiments)

3. **online processing:** Data Archiving, h Reconstruction, Data Selection, Data Distribution, etc..., in charge of data storage, data pre-selection and distribution.

The online data selection task will be implemented as shown on figures 2 and 3. The Main Frame Builder is the central part of the VIRGO data acquisition system. It sends frames to the data archiving and to the data quality checker (first task of the online processing). The resulting frames are sent to the reconstruction task that builds calibrated data (h strain), which is the main input of the algorithms used in the online data selection system.

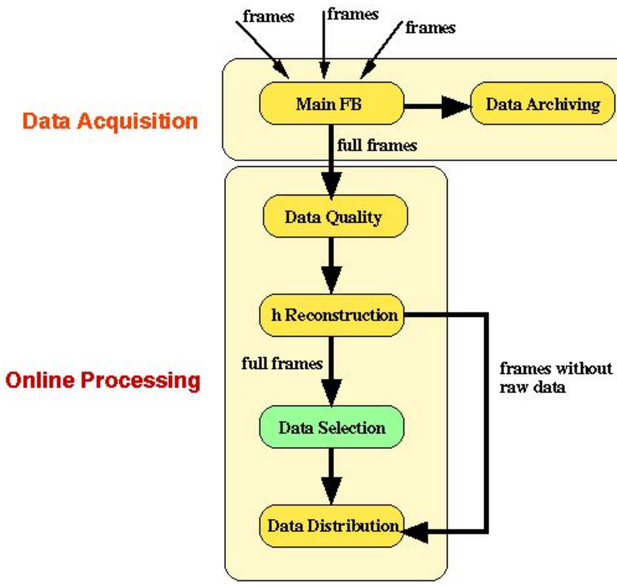

Figure 2: *The online processing tasks and the online data selection working principle.*

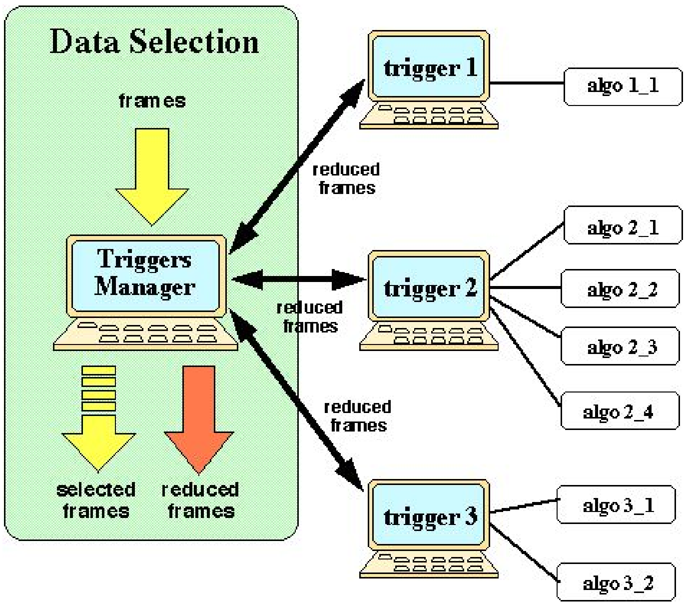

Figure 3: *The "Triggers Manager" in charge of the online data selection exchanging reduced frames with the different trigger applications in charge of the algorithms.*

As shown on figure 2, the online data selection working principle is to convey raw data in the case one of the data selection algorithms has triggered, and a reduced set of data otherwise.

Figure 3 shows that a workstation, the "Triggers Manager", is in charge of taking input frames, making a reduced copy of them (that does not contain any raw data) and sending those reduced frames to several workstations (trigger1, trigger2, trigger3, etc...), each in charge of one or several algorithms. Frames containing only the results of the algorithms are sent back to the "Triggers Manager" that, according to those results and to the data quality results, selects the full frames to be sent to the Data Distribution system.

This selection of data subsets should be sufficiently loose to keep all the signals that could be seen by off-line analysis and sufficiently tight to reduce by at least a factor 10 the amount of data stored in the Data Distribution system and off-line analysed.

Two ways to exchange data between a trigger application and one of its algorithms are foreseen for now: frame (a more convenient way for complex data analysis algorithms) or vector (more user friendly for algorithm developers), that is a n-Dim array of data associated with some useful information (typically: a FrVect structure of the frame format [4]).

3 A proposed interactive environment: VEGA

Before running signal search algorithms in online data selection or off-line production mode, we need to prototype and test those algorithms. For this purpose, an interactive off-line software environment is needed, that allows to visualize data, to manipulate data quite easily and to give the possibility to call algorithms with simple "plug-ins".

ROOT is a Data Handling Environment for High Energy Physics. It is developed at CERN, used officially by many experiments like NA49 and ALICE at CERN, CDF at Fermilab, STAR and PHENIX at BNL, and has user support sites at CERN, Fermilab and BNL. It uses an object oriented language (C++), it has a lot of convenient tools for graphical and interactive data analysis. And, key feature, it uses a C/C++ interpreter that allows C or C++ to be used as command language: this makes the ROOT macros or the interactive commands identical to C or C++ programs.

We have initiated the development of a custom version of ROOT [1]: VEGA (Visual Environment for Gravitational waves data Analysis), This version makes ROOT able to read frames (using the functions provided by the Frame Library [4]). The few lines below show an interactive session where data under frame format are read and plotted, including the result of an algorithm that was interfaced to VEGA. The result of those VEGA command lines is shown on figure 4.

```
TCanvas* canvas = new TCanvas("canvas","Vega Example");
can->Divide(2,2);
FrFile f("demo.dat")                 /* open the file containing frames */
FrameH* frame = FrameRead(f)         /* load first frame of the file */
VSeries v(frame,"adc.ADC0")          /* extract a signal from this frame */
canvas->cd(1);
v.Draw()                             /* draw content of signal v */
canvas->cd(2);
v.Draw(myAlgoFFT,120,500)            /* draw result of an algorithm applied on signal v */
canvas->cd(3);
v.DrawHist()                         /* draw distribution of signal v */
```

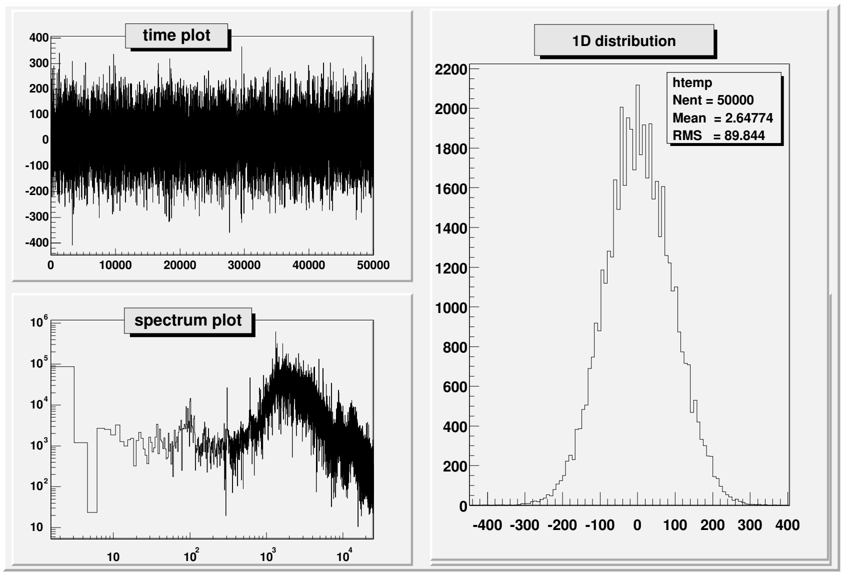

Figure 4: *Some plots showing the result of an interactive session of VEGA.*

4 A preliminary algorithms Interface

We try to define here a standard interface giving the possibility to develop data analysis algorithms the same way for different environments like online data selection, off-line production mode, off-line interactive mode, tests of algorithms using simulated data, etc...

This standard interface should give the possibility to easily plug an algorithm to those various environments, that means the possibility to re-use it in different places.

4.1 Guidelines and API's

Trying to define a "standard" interface faces the problem of defining some rules. Some preliminary guidelines for the algorithms developers are asserted here:

- **Language:** algorithm should be written in Object Oriented style (for instance C or C++), in order to have flexibility and a simplified maintenance.
- **Configuration:** configuration parameters are passed to the algorithm using a easy handling way (for instance a character string).
- **Data exchange:** algorithm's inputs and ouputs should be limited to some standard types (like frames or vectors).
- **Isolation:** algorithm should not change the input data, in order to keep an isolation between the input and output data.
- **Output structure:** algorithm's results, when put back into a frame, should be stored in a specific structure (like the FrTrigData structure [4]).

A basic C structure is used by the algorithms interface. Some utilities and API (Application Programming Interface) are provided to extract the algorithm configuration parameters, to create and fill this basic C structure and to exchange data between the algorithm and external environment. Moreover, some utilities to do statistics on the algorithm results and a short library of basic functions (FFT, FWT, Transfer Function, Correlation Function...) will be provided.

4.2 Some specifications to plug an algorithm

The algorithms interface requires C or C++ and, for C case, is mainly based on a basic C structure. But to use an algorithm, we first need to fill this C structure i.e. to get a configuration. Then, we must define input data format and output data format. More informations can be found in a Virgo Note [6].

Algorithm Configuration

The algorithm's configuration is set by parameters stored in a basic C structure provided by the algorithms interface and completed by the algorithm developer. From the user point of view and for what concerns the online data selection, the configuration is a character string passed to the interface (which decodes it and puts the parameters into the basic C structure). This character string is made of keywords, each followed by at least one parameter, like for instance:

```
"name BinaySearch input h massmin 0.8 massmax 4.0 freqmin 35."
```

The API provided to create the basic algorithm structure: `UserAlg *alg = UserAlgNew(char *paramString)`, is also the one which decodes this character string.

Input and Output Format

The input and output data format of the algorithm should be quite standard and we have chosen for now two possibilities: frame or vector. In the "frame" case, the algorithm is embedded in a function that takes a frame as input and returns its results in a frame:

```
FrameH *frameOut = UserAlgProc(UserAlg *alg, FrameH *frameIn,...)
```

The same type of function is available for the "vector" case (see documentation [6]).

5 Conclusion

The VIRGO experiment has began to look at the implementation of signal search algorithms for off-line or online use, and LAPP has been developing an off-line data analysis environment, VEGA, a preliminary VIRGO online data selection system and a preliminary algorithms "standard" interface for this purpose. LIGO is also working on such algorithms interface specifications and the hope is that future will be devoted to exchange of ideas between the two experiments in order to have algorithms interfaces in VIRGO and LIGO quite similar.

Acknowledgments

We would like to thanks people from VIRGO and LIGO experiments for fruitful discussions.

References

1. Proceedings AIHENP'96 Workshop, Lausanne, Sept. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch
2. "Virgo DAQ system", C. Arnault et al., presented by F. Chollet, Proceedings Xth Real Time Conference, Beaune, Sept. 1997
3. "Fast trigger algorithms for binary coalescences", R. Flaminio et al., Astroparticle Physics 2 (1994) 235-248
4. "Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors", Virgo Note VIR-SPE-LAP-5400-102 and Ligo Note LIGO-T970130-B. "Frame's Library User's Manual", Virgo Note VIR-MAN-LAP-5400-103 (v3.72). See also http://wwwlapp.in2p3.fr/virgo/FrameL
5. B.Mours, Gravitational Wave Detection, Proceedings of the TAMA International Workshop on Gravitational Wave Detection, Saitama, Japan, November 12-14 1996, edited by K.Tsubono, M.K.Fujimoto and K.Kuroda (1997)
6. "A Proposal for Data Analysis Algorithms Software Specifications", Virgo Note VIR-SPE-LAP-5400-106