Proceedings of ICRC 2001: 1–4 © Copernicus Gesellschaft 2001

ICRC 2001

An Object Oriented approach for the data management of the ARGO-YBJ experiment

S.M. Mari and S. Bussino for the ARGO-YBJ collaboration

Department of Physics, University of Roma Tre and INFN sezione di Roma3 - Via della Vasca Navale, 84 00146 Rome ITALY

Abstract. The Argo-YBJ detector will be able to continuously monitor the sky with a low threshold, an high duty cycle and a large solid angle. The experimental and Monte Carlo data flow will be some orders of magnitude larger than the data flow of the past experiments and it requires a strong computational effort to face out the management, the reconstruction and the distribution of the data for the end user analysis. In the paper an Object Oriented framework for the data management based on Object Oriented Database (Objectivity) and on the ROOT package will be presented. Hundreds thousands events were stored in order to test the performance and the flexibility of the model.

1 Introduction

The FORTRAN programs and libraries have now reached their limits: it seams that this software doesn't scale up to the needs of the new generation experiments with an experimental and Monte Carlo data flow some orders of magnitude larger than the data flow of the past experiments. The Argo-YBJ experiment (C. Bacci, 1999), (C. Bacci, 2000) is a full coverage carpet realized by means of RPC chambers devoted to the study of cosmic rays radiation, mainly γ in the energy region $\sim$ $(100\ GeV \div 10\ TeV)$ and it is located at YangBaJing (Tibet - China) at 4300 meter a.s.l. The carpet will cover an area of $\sim$ $(74 \times 78)\ m^2$ and will perform an electronic image of the shower measuring arrival time and position for each ionizing particles of the shower. The experiment requirements for the software and for the computing exceed those of the existing EAS experiments not only because of physics items but also because of the expected data flow. The storage and management of the data of the Argo-YBJ experiment represent a challenge mainly because of the large data flow due to the high trigger rate of the apparatus. The data flow and data volume of Argo-YBJ lead toward OO programming technologies for developing the software for the experiment: the storage and management of a huge amount of data and the maintenance of software code and libraries composed by many source lines is a very difficult task using old software technologies. The advantages of using C++ language (encapsulation, inheritance, polymorphism, etc.) strongly recommend it as the natural choice to develop the software needed by the Argo-YBJ experiment. It must be also considered that the HEP community it is now strongly involved migrating toward these new software technologies. The Object Oriented approach is a powerful tool to face out the challenges of the new experiments. In an Object Oriented environment data should be stored as class objects either using specific software ROOT (Rene Brun et Fons Rademakers, 1997) or using an OO database, like Objectivity (Objectivity Inc., 1997) and its OODBMS. A preliminary test setting up an OO environment using C++ structures and strategy patterns (E. Gamma, 1994) was performed. The use of a database is strongly suggested by the advantages concerning the great flexibility in data-management and by the possibility of data backup, replication and recovery which are peculiarities of the databases. It should be also pointed out that the informations concerning the data, needed for the physics analysis, are completely contained in the database and available simultaneously to all end users. Also the ROOT package has been taken into account to implement the storage and management of the data. The ROOT package is not a database, therefore it doesn't have the advantages peculiar to database as the great flexibility in data-management, the possibility of data backup, replication and recovery. ROOT provides a set of OO structures and functionality allowing the storage and analysis of a large amount of data, its structures include also histograms, graphics and visualization classes. It should be considered that ROOT allows interactive computing and also data compression which is an important point because it reduces the cost for disks needed to store data. ROOT is developed at CERN so it doesn't suffer of problems related to the small companies producing commercial databases.

The Argo-YBJ software has been designed taking into ac-

Correspondence to: mari@fis.uniroma3.it

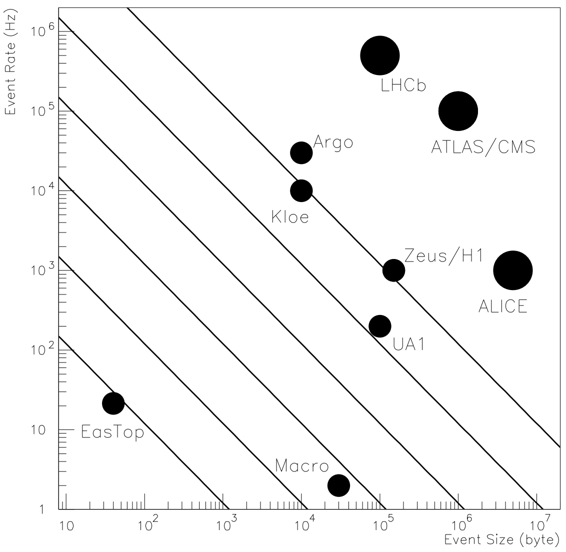

Fig. 1. Data flow of the next generation experiments compared to the past experiments: the lines are drown for constant value of event size times event rate

count the following rules:

- the software should be able to run on different platforms: the software/hardware commodities market is changing so fast that migration between different platform has to be foreseen and must be easy to be implemented
- it should be flexible enough to face out the new requirements coming from the developing of the physics analysis and should be easy to use
- it should be able to support the large data throughput of the experiment
- the software should be insulated from the persistency tools

2 The Argo-YBJ data flow

The throughput of the new generation experiments is orders of magnitude larger than the past experiment, in figure 1 is reported the trigger rate versus the event size for the experiments now in data acquisition and for the experiments of the next generation in the field of Cosmic Ray Physics and High Energy Physics. It can be seen that the trigger rate times the event size for the next generation experiments is of the order of $10^9 \div 10^{11}\ byte/sec$.

Planning the computing model for the Argo-YBJ experiment, it should be considered that it must be structured around the data flow generated by the experiment and that it is strongly related to the software environment chosen to realize data

trigger	**rate**	**mean multiplicity**	**events/year**
	kHz	$hits$	
LMT	$17,2$	24	5.4×10^{11}
MMT	7.8	43	2.5×10^{11}
HMT	1.3	164	0.4×10^{11}

Table 1. Trigger rate of the Argo-YBJ experiment.

persistency and data management. The Argo-YBJ trigger rate was separated in three main branches, following a simple approach that lists the events on the base of the hits multiplicity:

- LMT low multiplicity trigger ($hits\ multiplicity < 30$)
- MMT medium multiplicity trigger ($25 < hits\ multiplicity < 70$)
- HMT high multiplicity trigger ($hits\ multiplicity > 60$)

The characteristics of these trigger branches are reported in table 1 where it is also shown the events expected in one year of data taking for each trigger. The full design of the software was developed taking into account these trigger rates.

3 The computing model

The requirements of the Argo-YBJ experiment related to the trigger rate, to the data flow and to the physics goals define the software architecture. Moreover it should be also considered that in the Argo-YBJ experiment there is no direct coupling between the ON-line equipment located at YangBaJing (Tibet) and the OFF-line equipment located in Italy: the data will be available for the physics analysis in Italy as a set of tape cartridges so that the OFF-line data structure and data management are clearly independent by the ON-line data. This peculiar characteristic of the Argo-YBJ experiments should be considered in the design of the software. The following considerations were taken into account in order to define the software framework for the Argo-YBJ experiment:

1. clear separation between Persistent Data and Transient Data. Persistent data are low-level data which will be stored on disks in such a way to minimize disk space and in order to realize an easy data access. The end user physics analysis or the reconstruction software need of transient data which are a high-level data and which couldn't be stored on disk.
2. the software architecture should realize separation between the Data and Methods and Algorithms. The data should be organized in very simple and robust structures, algorithms and methods should be considered as data producers: they transform data reading raw data and producing reconstructed data or filtered data.

structure	type	data members	member functions
Hit	**C++**	**hit-address**	**get**
		hit-time	**set**
		hit-weight	

Table 2. Structure of Hit objects.

3. four data types have been selected for the Argo experiment:

   - MC Data: the Monte Carlo data
   - Detector Data: the Real Data from the apparatus
   - Reconstructed data: data created out of low-level informations representing high-level data needed by the physics analysis
   - Statistical Data: the results of the analysis process

4. the interaction between data and algorithm and the behavior of the objects and their collaboration processes are managed by dedicated software structures, called Managers

4 The software structure

A prototype of the computing model for the Argo-YBJ experiment was realized according to what we have shown in the previous sections. The software was organized in three main layers: the first one was devoted to the basic, lowest level objects, designed to store the elementary informations produced by the apparatus. A second layer was planned in order to manage the basic objects and to produce higher level ones by means of reconstruction and analysis algorithms. The third layer was completely dedicated to persistency, with a very week coupling to the previous layers. In this way it is easy to change persistency tools (ROOT package, Objectivity DB). The following basic objects were defined in the first layer:

- **Hit** → it is the basic data object, it contains the informations related to the experimental hit
- **Event** → an abstract class which should be specified in different real types (RawEvent, RecEvent, etc...). It contains the informations of the physical event and the links to the corresponding hits collection
- **Run** → the events container. It is the entry point to access the data, the calibrations and geometry informations of the apparatus
- **Catalog** → the catalog contains the information on each run, it should be looked up to select the runs for the reconstruction and/or analysis.

An example of the variables implemented in each persistent object are shown in tables 2, 3, 4.

structure	type	data members	member functions
Event	**C++**	**event number**	**get**
		pad multiplicity	**set**
		strip multiplicity	
		energy estimate	
		x of core shower	
		y of core shower	
		theta of the shower	
		phi of the shower	
		cone index	
		type of fit	
		χ^2 **of the fit**	
		pads in the fit	
		photon/hadron probability	
		trigger pattern	
		date	
		noise subtraction	

Table 3. Structure of the Event objects.

5 Managers structure

The managers were realized to take responsibility of:

1. input/output of data
2. evolution of data
3. fulfill end user analysis requirements

Data will be stored on disk as raw-data and will become reconstructed-data and statistical-data by means of the dedicated manager objects. Also the end user interaction with the data stored on disks was implemented through manager. The following manager objects have been implemented:

- **MManager:** the main code manager
- **IOManager:** it is responsible for the data input/output
- **RecoManager:** it is responsible for the events reconstruction
- **StManager:** it is responsible for creating statistical objects
- **UserManager:** it is responsible for the management of the end user code
- **DMManager:** it is responsible for data monitoring and data validation

structure	type	data members	member functions
Run	**C++**	**run number**	**get**
		number of events	**set**
		date	
		run type	
Catalog	**C++**	**name**	**query functions**
		list of variables	

Table 4. Structure of the Run and Catalog objects.

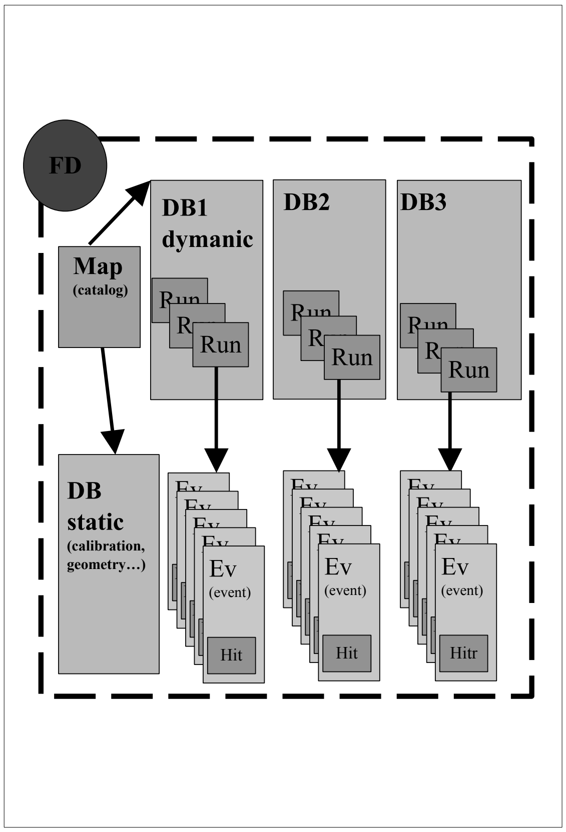

Fig. 2. Objectivity DB model for the Argo-YBJ experiment.

6 Prototype results

A prototype of the Argo-YBJ computing model was realized, the code was installed in a PC cluster made of one DPII 300 MHz and two PII 300 MHz in a client-server configuration. Hundreds thousands events were generated using the Corsika Monte Carlo and they were used to test the code and the performance of the model. Persistency was implemented either by means of Objectivity database or by means of the ROOT package. A prototype of Objectivity DB was designed and the schema is shown in figure 2. It consists of a Federation (FD) containing a collection of database (DB) of two different kinds: the dynamic ones dedicated to store the data collected by the apparatus and the static ones dedicated to store calibration constants, geometry data and so on. Each DB includes basic containers which consist of the data runs (Run) taken at YangBaJing. The Run is the basic unit of the federation: it is a collection of events (Ev) structured as C++ objects. The Run is an evolving object because the Run content can be changed by member functions which access the DB to reconstruct events from raw hits, to perform noise subtraction etc. The Run is also the basic unit to access the federation. The event objects (Ev) are linked by means of dedicated connections to the hits (Hit) that are the elementary bricks of the federation. A map object (Map) was implemented to realize the DB catalog, also devoted to link the Run objects to the static DB containing the geometry data and of the calibration constants. The Objectivity database was successfully installed and run, the model was implemented and a Federated DB was created. The federation was filled-up by using Monte Carlo data: hundreds thousands events were written and read out successfully. A second test was performed by using the ROOT package which was successfully installed on the same PC cluster used to test the Objectivity database. ROOT files were created and filled-up by using the Monte Carlo events generated to test Objectivity. Also the ROOT data compression was tested: data can be compressed on the flight before to be stored on disk. The ROOT test was performed using a compression level = 0 (which means no data compression) and a compression level = 1 (which is the recommended one by authors of ROOT).

7 Conclusion

The data flow and the complexity of the data management of the Argo-YBJ experiment requires to develop robust software, flexible and user friendly. In these years the Object Oriented technologies appear as the best candidate to fulfill the requirements of the new generation experiments. A prototype of the Argo-YBJ computing model was realized in an Object Oriented framework, data persistency was achieved by means of Objectivity database or by means of the ROOT package. Some hundreds thousands Monte Carlo events were used to successfully test the computing model.

References

C. Bacci et al.,The use of RPC in the ARGO-YBJ project, Nucl. Phys. B (Proc. Suppl.) 78 (1999) 38-43.

C. Bacci et al.,High altitude test of RPCs for the ARGO-YBJ experiment, Nucl. Instr. Meth. A443 (2000) 342-350.

Rene Brun et Fons Rademakers, ROOT-An Object Oriented Data Analysis Framework, Nucl. Instr. Meth. A389 (1997) 81-86

Objectivity Inc., The Objectivity Database Standard R.G.G. Cattell (1997)

E. Gamma et al., Design Patterns Addison Wesley (1994)