The Online System of the ARGO Experiment

Huihai He for ARGO Collaboration
Institute of High Energy Physics, CAS, Beijing 100039, P. R. China
† **email:** *hhh@ihep.ac.cn*

Abstract

The ARGO experiment has been designed to detect shower events at an energy threshold as low as a few hundreds of GeV. To achieve this goal, the data acquisition system has been implemented to handle a high data flow up to 7Mbytes/s for a single chain. In such conditions the software architecture plays an important role. We report on the software structure and on the algorithms implemented to control the data flow to the event builder and storage system.

1. Introduction

The Sino-Italian collaboration YBJ-ARGO experiment (YangBaJing Astrophysical Radiation with Ground-based Observatory)[1] located at Yangbajing ($90^o31'50"E, 30^o6'38"N$, 4300m a.s.l.) is under its way. The goal of the experiment is to study cosmic rays, mainly γ ray point sources ($> 100GeV$) and γ ray burst ($> 10GeV$) from the northern hemisphere in the declination band $-10^o < \delta < 70^o$, by means of detecting small size EAS (Extensive Air Shower) using a full coverage RPC (Resistive Plate Chamber)[2] carpet.

The carpet consists of 1848 RPCs, $1258 \times 2850mm^2$ each, covering a total area of about $10,000m^2$. One chamber consists of 2×5 the smallest detector units so-called PADs (each with 8 readout strips).

YOLA (the Yangbajing OnLine system of Argo) is based on the Level 1 DAQ (hardware DAQ) system[3]. It works on multi-bus, multi-interface, and multi-platform, producing huge amount of data, thus needs high speed DAQ, high CPU power, wide network bandwidth, high speed mass storage system, etc

2. Level 1 DAQ

Table. 1 shows the offline estimation of ARGO event rate and data rate[4], which needs high speed DAQ and wide bandwidth for data transfer. Signals from the front end electronics are digitalized and buffered at the Local Station which serves 1 CLUSTER (consisting of 120 PADs), then data from Local Stations are transferred to the AMB (Argo Memory Board) where sub-events are built

pp. 773–776

Table 1. Offline estimation of ARGO event and data rate

Event Multiplicity	Event Rate(MHz)	Data Rate(Mbytes/s)
Low (< 30)	17.2	3.5
Middle ([25,70])	7.9	2.7
High (> 60)	1.3	1.1
Total	21.7	6.4

and buffered again. The ROCK (Read-Out Controller) gathers all the data from its slave AMBs and also sub-events are built and buffered. At last, a ROCKM (ROCK Manager) acquires all the data belonging to 1 event from all the ROCKs, the event is built and buffered there waiting for the online system to take it. All the above are done by hardware under an event-driven mechanism, without software penalties and CPU overhead, thus achieving high speed.

3. Software Architecture

The online system consists of the data acquisition core, the controlling system, the monitoring system, error logging, user interface, etc. Fig.1 shows the software architecture in detail.

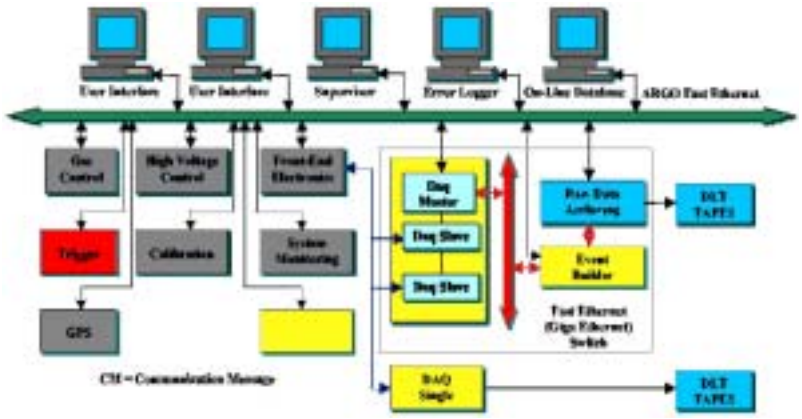

Fig. 1. YOLA software architecture.

Fig.2 shows the software package tree. Since the whole DAQ system is based on VME bus, we need the VME bus driver (the Vme package) above which all the VME boards are driven (AMB, ROCK, ROCKM, TRIGGER, etc). YOLA is considered as the level 2 DAQ (L2 package), which is based on the hardware DAQ, and takes the responsibility of collecting data from ROCKM and sending them to the mass storage system (FARM package), while the latter receives data

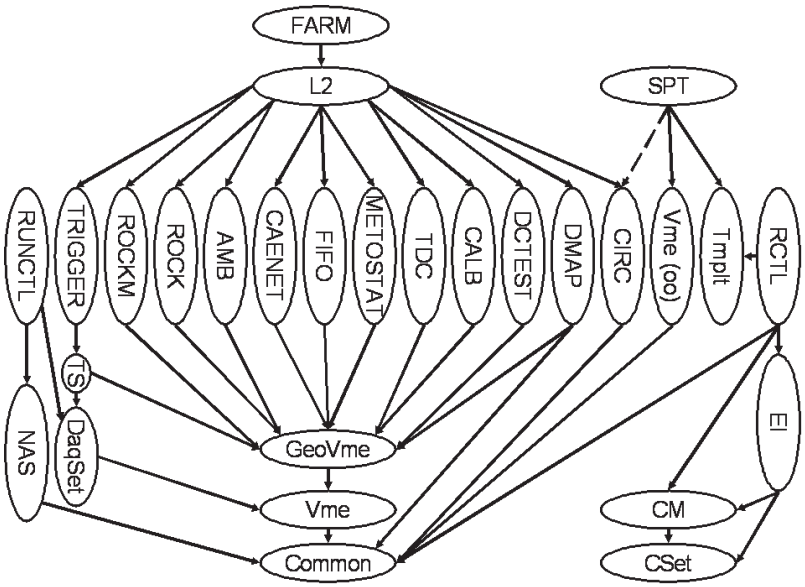

Fig. 2. YOLA package tree.

from the L2 system and stores them to tapes. Other packages provide slow control, run control, remote control, monitoring, etc.

4. YOLA Template

A template is designed to keep as much as possible the peculiar code of a process separated from the pieces of code that are common to all DAQ processes. It provides the model adopted for all the processes that take part in the YOLA system, working on all UNIX-like systems, including Linux, LynxOS, etc. The Template, together with its associated libraries, provides any process with the mechanisms to handle commands sent by another process (e.g. Run Control) and at the same time to deal with event data (see Fig.3).

Running a DAQ process, we want to make available to the external world some of its features and a number of its relevant parameters. This is made by storing them in a Shared Memory (called registry) which holds on each node the complete information about DAQ processes running on the node. The first step of a process initialization is to register itself. Every process can access that registry and acquire information of other processes running on the same node.

Communication between processes running on one node is done through message queue. Every process has a peculiar message queue ID stored in the registry. Other processes on the same node can get that ID and send messages

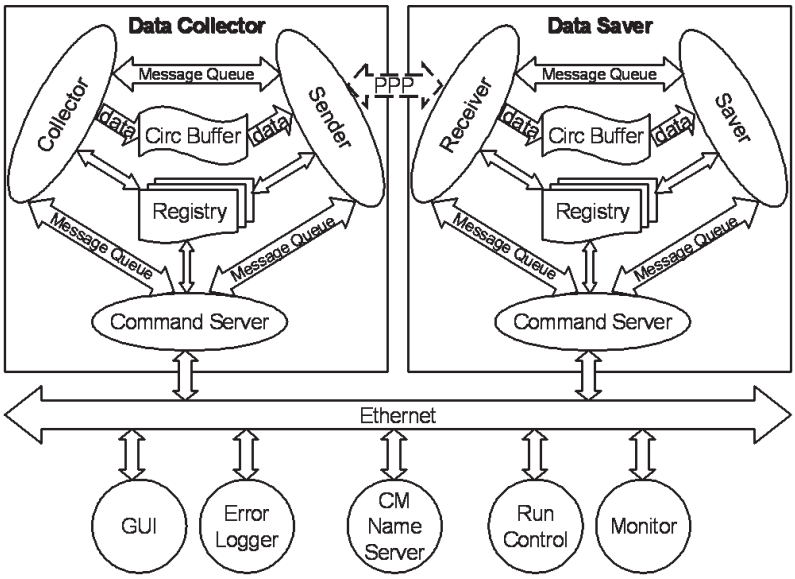

Fig. 3. YOLA mechanism.

to or receive messages from it. When one process receives a message, it puts the execution results of that message in the registry for the message-sender to read.

A command server running on each DAQ node takes the response of inter-node process communication. It receives commands from the network and update the registry or send a message to one or more processes on its node.

5. Conclusion

The ARGO online system is running with 40 CLUSTERs mounted. Other parts of the online system are under development, such as the GUI (Graphics User Interface), the Error Logger, etc.

6. References

1. Abbressics M. et al., Astro-particle Physics with ARGO Proposal, 1996
2. Santonico R. et al., Nucl. Inst. & Meth., 1981, A187:377
3. YBJ-ARGO collaboration, Proc. of 27th ICRC, Hamburg 2001, Vol. 7, p.2887
4. G. Mancarella et al., ARGO note 005/00