A ROOT based monitoring system for the KLOE Experiment

A.Doria [a]
for the KLOE Collaboration [†]

[a] *INFN Sezione di Napoli, Via Cintia, Napoli, Italy*

The monitoring of the KLOE experiment requires filling and presenting a large number of histograms. A client-server system, based on the ROOT framework, has been developed for this task; the sistem is designed in such a way that network traffic and online machines load are optimized.

[†] The KLOE collaboration: M. Adinolfi, F. Ambrosino, A. Aloisio, A. Andryakov, A. Angeletti, A. Antonelli, C. Bacci, R. Baldini-Ferroli, G. Barbiellini, G. Bencivenni, S. Bertolucci, C. Bini, C. Bloise, V. Bocci, F. Bossi, P. Branchini, G. Cabibbo, A. Calcaterra, R. Caloi, P. Campana, G. Capon, G. Carboni, M. Carboni, A. Cardini, C. Carusotti, G. Cataldi, F. Ceradini, F. Cervelli, F. Cevenini, G. Chiefari, P. Ciambrone, S. Conticelli, E. De Lucia, G. De Robertis, R. De Sangro, P. De Simone, G. De Zorzi, S. Dell'Agnello, A. Denig, A. Di Domenico, S. Di Falco, A. Doria, F. Donno, E. Drago, V. Elia, L. Entesano, O. Erriquez, A. Farilla, G. Felici, A. Ferrari, M. L. Ferrer, G. Finocchiaro, D. Fiore, C. Forti, G. Foti, A. Franceschi, P. Franzini, M. L. Gao, C. Gatti, P. Gauzzi, S. Giovannella, V. Golovatyuk, E. Gorini, F. Grancagnolo, E. Graziani, P. Guarnaccia, U. V. Hagel, H. G. Han, S. W. Han, M. Incagli, L. Ingrosso, Y. Y. Jiang, W. Kim, W. Kluge, V. Kulikov, F. Lacava, G. Lanfranchi, J. Lee-Franzini, T. Lomtadze, C. Luisi, C. S. Mao, A. Martini, W. Mei, L. Merola, R. Messi, S. Miscetti, A. Moalem, S. Moccia, F. Murtas, M. Napolitano, A. Nedosekin, P. Pagès, M. Palutan, L. Paoluzi, E. Pasqualucci, L. Passalacqua, M. Passaseo, A. Passeri, V. Patera, E. Petrolo, G. Petrucci, D. Picca, M. Piccolo, G. Pirozzi, L. Pontecorvo, M. Primavera, F. Ruggieri, P. Santangelo, E. Santovetti, G. Saracino, R. D. Schamberger, C. Schwick, B. Sciascia, A. Sciubba, F. Scuri, I. Sfiligoi, S. Sinibaldi, T. Spadaro, S. Spagnolo, E. Spiriti, C. Stanescu, L. Tortora, E. Valente, P. Valente, G. Venanzoni, S. Veneziano, D. Vettoretti, S. Weseler, Y. G. Xie, C. D. Zhang, J. Q. Zhang, P. P. Zhao

1 The KLOE experiment, detector and data acquisition

The main physics goal of the KLOE [1] experiment, at the DAΦNE Φ -factory in INFN Frascati Laboratories, is the search for the direct CP-violation in the neutral kaon system. The KLOE detector design is optimized to measure the decays $K_{S,L} \rightarrow \pi^+\pi^-, \pi^0\pi^0$ to determine the ratio ϵ'/ϵ at sensitivity of the order of 10^{-4}. The rate from the detector at full luminosity is estimated to be $\sim 10^4$ events/s, corresponding to a total throughput of 50 Mbytes/s.

The detector components are, proceeding inwards, an electromagnetic calorimeter, made of one barrel and two encaps, a cylindrical drift chamber surrounding the beam pipe and two small calorimeters on the quadrupoles.

The detector read-out contains about 25000 electronic channels that are acquired using specialized readout controllers and Alpha based VME processors connected to a FDDI switch. A farm of AIX 4 way servers, connected to the switch, receives, builds, formats and records events. Destination assignement to the farm components for each group of events is accomplished by a dedicated data flow controller interconnected with the VME processors through VME mirrored memories.

Monitor processes can "spy" events from the main data stream, at farm level, allowing the online monitoring of the different system components.

2 The ROOT System

The ROOT system [2], developed since 1995 in the context of the NA49 experiment at CERN, provides a complete Object Oriented framework, with all the functionalities needed to handle and analyze large amounts of data. A wide range of HEP tasks is covered, with histogramming methods in 1, 2 and 3 dimensions, curve fitting, function evaluation, minimization, graphics and visualization classes.

C++ is the language used for ROOT command language, macros and programming. The system has a built-in C++ interpreter, named CINT, that allows fast prototyping of C++ macros since it removes the time consuming compile/link cycle. To achieve higher performances and to have complete syntax checking of the code, compiled C++ code can also be used in the ROOT framework.

The backbone of the ROOT architecture is a layered class hierarchy, where most of the classes inherit from a common base class TObject, which implements common behavior for ROOT classes. All ROOT objects can be uniformly stored using a proprietary file format.

3 Monitoring System Architecture

The goal of our system is to provide a common tool for online and offline monitoring of the KLOE detector readout, Trigger System and Front End Electronics. The system is used both by the experts of the sub-systems, who participated as well to its development, and by the shifters, who need an easy-to-use tool to continuously check the experiment status.

The architecture is based on the client-server model, where the server produces and fills histograms and the clients are in charge of presenting them. The system is highly flexible, since the clients load in a startup phase the histogram tree produced by the server, and configure their Graphical User Interface according to it.

Clients can work in two different modes:

- in online mode – get histograms from a server through a TCP/IP connection;
- in offline mode – read from a ROOT file previously created by a server.

The client and the server are implemented as C++ compiled programs, called Kbrowser and Kserver, using the ROOT class library for histogram production and visualization, file access, network communication and GUI management. Working in the ROOT framework adds to our program many useful features.

The possibility of filling and visualizing histograms on different machines offers important advantages. The histogram online production can be centralized, while many clients can simultaniously monitor histograms from different sites on the network. The system load can thus be subdivided on different machines. The exchange mechanism of updated histograms between server and clients is designed to minimize the network traffic.

4 The KLOE histogram server - Kserver

At the startup the Kserver creates all the histograms, organizing them in a tree structure according to physical or logical partitions for each sub-system to monitor (ECal, QCal, Drift Chamber, Trigger, FEE), then it enters a loop where the histograms are continuously updated with the incoming data. For each iteraction of the loop, Kserver checks its network connections for new requests or commands and eventually performs one of the following actions:

- send the tree structure to a client;
- send a single histogram content;
- send the contents of a set of histograms;
- reset the contents of all histograms;
- save all current histograms into a ROOT file;
- set or send the value of an internal parameter (e.g. enable disable the filling of a tree branch);
- exit.

To produce histograms from the online data stream, Kserver reads events from a buffer, residing on one of the online farm nodes or on a remote monitoring machine. In the first case, when Kserver runs on a node of the farm, the buffer is directly filled by the DAQ process which builds the complete events. When Kserver runs in a remote monitoring machine, its buffer is filled by a dedicated process that gets samples of events through a TCP/IP connection from a dedicated "spy" daemon in the farm.

To make offline analysis of past runs, Kserver can produce and serve its histograms getting data from a raw data file. An other facility offered by Kserver is to perform a batch conversion of raw data into ROOT histogram files.

5 The KLOE histogram browser - Kbrowser

The visualization of the monitoring histograms is achieved using Kbrowser. Working in online mode, Kbrowser opens a TCP/IP connection to a remote node where a Kserver is running and gets the current histogram tree structure. Each node in the structure corresponds to a browsable directory, while each leaf is a histogram name.

The user can browse the tree using Kbrowser control bar, starting from the first level; when a histogram is selected, its current content is requested to Kserver and, when received, displayed in a canvas. The control bar also contains a button for the drawing of all the histograms of the current directory, that are retrieved from the server in one single request. In this way the network data transfer is limited to the contents of the histograms actually displayed.

In offline mode, Kbrowser can navigate through the file system directories and in the directories of ROOT files previously saved by Kserver. This feature is particularly useful for detector experts, to perform offline checks of the working status of specific parts of the subdetectors. On the other side, online mode is currently used by people on shift to monitor some sets of global histograms.

Kbrowser uses the ROOT classes for networking (similarly to the server), for the creation of its GUI and for the histogram drawing. In the GUI implementation, some ROOT classes are extended in order to allow the compilation of the callback code.

6 The state of the art and some performance measurements

In KLOE the ROOT version 2.20/06 is currently used and our monitor programs run on HP-UX 10.20, OSF1 4.0 and AIX 4.2.

At present the monitoring system is working with a set of $\sim$ 30000 histograms, organized in $\sim$ 1300 directories and partitioned in 5 main classes. The filling of each of the classes can be disabled or re-enabled at run time. When the filling of all the histograms is enabled, Kserver processes $\sim$ 300 events/s on a 10 SpecInt95 CPU, consuming $\sim$ 60 Mbytes of memory.

The most time-consuming operation is the saving to a ROOT file: in our first trials it took up to 4.5 minutes for the complete set of histograms and this time didn't improve changing the compression level. We observed that our system have many levels of directories and that, for every directory to be written, a buffer flushing was performed. Overriding the flushing method, the writing performance of Kserver increased about 4 times, depending on the compression level. The values are reported in Table 1.

We foresee that the number of histograms produced by Kserver can be increased in the future without serious performance problems; however the use of more servers, running on different computers to produce the different classes of histograms, is advisable to optimize the load on the online system. We plan to use an analogous system for other kinds of monitoring, such as slow controls and to extend Kbrowser interactive functionalities.

compression	file size	time
0	33 MB	$\sim$ 3.5 min
1	9.5 MB	$\sim$ 50 sec
9	9.1 MB	$\sim$ 1 min

Table 1
Writing performance

References

[1] The KLOE collaboration, *KLOE, a general purpose detector for DAΦNE*, **LNF**-92/019 (1992);

[2] http://root.cern.ch