Online monitoring system at KLOE

Alessandra Doria[1]
*for the KLOE collaboration**

1 INFN sez. di Napoli - Via Cintia - 80125 Napoli ITALY

Abstract

A simple and efficient mechanism to perform distributed online monitoring of detector, physics and data taking has been designed and implemented at KLOE.

Monitoring tasks, running on distributed nodes, are able to process online data without DAQ system overload and without redundant data transfer and duplication. The main monitoring tasks include event display, histogram producers and data quality monitors based on fast event selection. The presentation of monitoring results is performed by means of graphic facilities.

The basic principles of the KLOE online monitoring and an overview of the system main components are presented.

Keywords: DAQ, monitoring

1 The KLOE experiment and its data acquisition system

The main physics goal of the KLOE experiment [1], running at the DAΦNE Φ -factory at the INFN Frascati Laboratories, is the search for the direct CP-violation in the neutral kaon system at sensitivity of the order of 10^{-4}.

The KLOE data acquisition system [2] has been designed to sustain a rate of $\sim 10^4$ events/s, and a data throughput of 50 Mbytes/s. In the DAQ system 10 readout chains are connected to an online farm of computing servers through a FDDI switch. In each chain, data coming from the detector are read out by a set of custom devices while a commercial VME-CPU board collects packets of sub-events and sends them through the switch to a given processor in the online farm. The packets of sub-events related to the same trigger numbers are sent to the same farm processor,

* M. Adinolfi, A. Aloisio, F. Ambrosino, A. Andryakov, A. Antonelli, M. Antonelli, F. Anulli, C. Bacci, A. Bankamp, G. Barbiellini, G. Bencivenni, S. Bertolucci, C. Bini, C. Bloise, V. Bocci, F. Bossi, P. Branchini, S. A. Bulychjov, G. Cabibbo, A. Calcaterra, R. Caloi, P. Campana, G. Capon, G. Carboni, A. Cardini, M. Casarsa, G. Cataldi, F. Ceradini, F. Cervelli, F. Cevenini, G. Chiefari, P. Ciambrone, S. Conetti, 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, E. Drago, V. Elia, O. Erriquez, A. Farilla, G. Felici, A. Ferrari, M. L. Ferrer, G. Finocchiaro, C. Forti, A. Franceschi, P. Franzini, M. L. Gao, C. Gatti, P. Gauzzi, S. Giovannella, V. Golovatyuk, E. Gorini, F. Grancagnolo, W. Grandegger, E. Graziani, P. Guarnaccia, U. v. Hagel, H. G. Han, S. W. Han, X. Huang, 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, M. Martemianov, M. Matsyuk, W. Mei, L. Merola, R. Messi, S. Miscetti, A. Moalem, S. Moccia, M. Moulson, S. Mueller, F. Murtas, M. Napolitano, A. Nedosekin, M. Panareo, L. Pacciani, P. Pages, M. Palutan, L. Paoluzi, E. Pasqualucci, L. Passalacqua, M. Passaseo, A. Passeri, V. Patera, E. Petrolo, G. Petrucci, D. Picca, M. Piccolo, G. Pirozzi, C. Pistillo, M. Pollack, 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, J. Shan, T. Spadaro, S. Spagnolo, E. Spiriti, C. Stanescu, G. L. Tong, L. Tortora, E. Valente, P. Valente, B. Valeriani, G. Venanzoni, S. Veneziano, Y. Wu, Y. G. Xie, P. P. Zhao, Y. Zhou

where the complete events are built. The assignement of the event data to the online farm processors is managed the Data Flow Control [3], that optimises data distribution according to the load of the farm processors. After the building phase, formatted events are written in a circular buffer where they wait to be extracted by an event recorder, which eventually writes them to disk.

2 Event monitoring and filtering

Readout electronics and detector response, beam parameters and data quality are monitored during KLOE data taking so that actions can be taken when the conditions are outside of a defined range or in case of malfunctioning. Moreover, online event selection based on fast event reconstruction is performed by a set of event filters, that produce selected event samples used for physics consistency checks nad for calibrations.

All these tasks are performed using formatted events as soon as they are available in the farm processors, i.e. when they are written in the output buffer of the builder. There are many kinds monitoring tasks: some of them are CPU bound, some need most of the acquired events, while others may work on reduced data samples. In order to avoid the overload of the online processors, we chose to implement a distributed system, where monitors and filters run either in the online farm or in remote nodes, according to their resource requirements.

3 Spying online event data

The mechanism of reading packets of formatted events from a circular buffer has been called *spying*, since its main requirement is to get events without disturbing the main data flow.

When a monitoring task runs in a farm node it can directly spy events, reading from the buffer as much data as it's able to process.

To allow monitoring tasks to run on remote nodes, a *spy daemon* has been implemented. It runs in each node of the farm, acting as an event server and distributes events via TCP/IP connections. The *spy daemon* is multithreaded: a thread is started for each established connection, so that more clients can be served contemporanely.

Three data serving modes are available, depending on client requirements. The method actually used is connection based and it is chosen by the client during the initial handshaking.

- **pop mode:** the client requests a new packet when the processing of the previous packet is completed and sends with the request the last processed event number. The *spy daemon* seeks in the buffer the oldest available packet following the already processed one and sends it back.
- **pop in advance mode:** the client issues a new request, containing the last received event number, as soon as a packet is received. If client processing takes longer than data transfer (this is often the case) a new packet is always ready to be processed.
- **push mode:** the *spy daemon* serves packets asynchronously to the client, until the TCP buffer is full. This mode minimize "dead times" due to data transmission, because of the extended overlap between data sending and processing.

Pop mode is the least efficient, but guaratees that the client always reads the latest data. It is best suited when only a small sample of events get analyzed, for example in user driven and interactive tools such as the event display. Tasks that require to sample the DAQ input at the highest synchronous rate use pop in advance mode, while push mode is useful if data processing rate is comparable with input event rate, otherwise it can produce a large delay between data transmission and processing.

The functions that perform either local or remote spying have been grouped in the *KID*

(KLOE Integrated Data Flow) library [4], together with the functions for offline access to raw data files and archived data. Processes using the *KID library* may easily change their data source and the way data are passed to the processing/analysis phase is uniformly managed by KID.

4 The Event Filtering tasks

The use of the *spy* mechanism on the main data stream allows to have a set of online tasks performing fast event classification, based on simple selection criteria. On the basis of information about position, energy and time, events as e^+e^-, $\gamma\gamma$, cosmic rays are selected and allow to calculate and monitor online important parameters with a minimum amount of CPU power.

The *Trigger monitor*, using the contents of trigger banks, selects Bhabha and cosmic events to calculate machine luminosity, trigger dead time, DAQ rates and more. The values of the monitored parameters are displayed by a simple tcl/tk graphic interface and are continuously updated during the run.

A set of *Event filters* select "interesting" events and, for each selected event type, fill an output circular buffer: it can be used as data source for other processes, with the same mechanisms used for the main DAQ event stream. Monitoring processes, such as the Event Display and the Physics monitor that we'll describe later, spy the filtered output buffer and perform their task only on selected types of events. A recorder process gets data packets from each filtered stream and stores files of selected events, that are used for quasi-online calorimeter calibration and for other offline analysis.

5 The Monitoring tasks

5.1 Histogram producer and browser [5]

The monitoring of front end electronics and detector readout requires filling and presenting several thousands of histograms.

A central histogram producer, based on the ROOT framework [6], has been implemented. It creates all the histograms in the ROOT format, organized in a tree structure according to physical or logical partitions (e.g. subdetector, sector, layer, chain), and updates the histogram contents with the incoming data.

Another ROOT based task, namely the *KLOEbrowser*, is in charge of the histogram visualization. The producer works as histogram server sending data through TCP connections to *KLOEbrowser* clients. The client graphic interface allows users to navigate through the histogram tree structure; each node in the structure corresponds to a browsable directory, while each leaf is a histogram name. When a histogram is selected its current content is requested and displayed, while an option allows the drawing of all the histograms of the current directory, that are retrieved from the server in one single request.

While the producer fills the whole histogram set, distributed *KLOEbrowsers* can display different subsets, but the network data transfer is limited to the contents of the histograms actually displayed. Produced histograms can also be saved to file, to be re-analized offline.

5.2 Event Display

The KLOE *Event Display* performs the full event reconstruction that has been developed for offline analysis, using the Analysis_Control (A_C) [7] Fortan package. To integrate the offline tools into the online environment, the A_C modules have been modified in such a way that input data can be obtained with the *spying* mechanism.

The *Event Display* graphic components are based on the *O-package* [8] graphical objects. Detector geometry, tracks, hits, energy deposits, displayed in a graphic user interface, allow an immediate visual analysis of the reconstruction results.

5.3 Physics Monitor

The *Physmon* task performs A_C full event reconstruction on e^+e^- and $\gamma\gamma$ streams obtained from the online event filtering. Processing these events, it monitors the state of a variety of relevant physics quantities, like times of flight, total energy, momentum, position of the interaction vertex and beam dimensions.

Physmon stores mean values and produces an histogram file (in HBOOK format) every time enough statistic is accumulated. A program similar to the KLOEBrowser converts histograms in ROOT format and displays them; the trend of mean values vs. time is plotted as well.

6 Task optimized distribution

The filtering tasks require the processing of events at a high rate, without an high CPU load. Since events collected by each online farm node are statistically equivalent, the most suitable configuration is to run filters locally on a farm node, avoiding intense network data transfer.

The histogram producer and the monitors based on full reconstruction are quite "heavy" tasks and their event processing rate is much lower then the DAQ rate, so it is not convenient to run them on a farm node. When only selected event streams are processed, the input rate is reduced and statistics can be increased merging data coming from more farm nodes.

However, filters and monitors distribution can easily be adjusted every time new tasks are added to the system and according to the current DAQ rates, CPU load and network traffic.

7 Conclusions

The *spying* mechanism described above is the core of the online data access method and its application to the different tasks has shown to be powerful and flexibile. The introduction of the event filtering level allowed to perform online most of the checks needed to determine data quality, that will be used in the next future to automatically activate detector calibrations and offline reconstruction. Moreover, the integration in the *KID library* of online and offline data access methods allows all KLOE software tools to manage all possible event sources in a simple and uniform way.

References

1. The KLOE collaboration, *KLOE, a general purpose detector for DAΦNE*, **LNF**-92/019 (1992);
2. The KLOE collaboration, *The KLOE Data Acquisition system, Addendum to the Tecnical Proposal*, **LNF**-95/014 (1995);
3. E.Pasqualucci for KLOE Coll., *Process and Data Flow Control in KLOE*, Proceedings of CHEP 2000, Padova;
4. I.Sfiligoi for KLOE Coll., *Data Handling in KLOE*, Proceedings of CHEP 2000, Padova;
5. A.Doria for KLOE Coll., *A ROOT based monitoring system for the KLOE Experiment*, Proceedings of AIHENP 1999, Crete;
6. http://root.cern.ch;
7. M.Shapiro et al.,*A Beginner's Guide To Analysis_Control And Build_Job*, CDF Note 384;
8. G.Barrand, OnX, presented at HEPVis95, FNAL (1995)