ALEPH 92-170
DATACQ 92-01
M. Cattaneo et al.
23.11.1992

Online monitoring in the Aleph experiment at LEP

M.Cattaneo[1], *O.Callot*[2], *J.Harvey*[3], *B.Jost*[3], *J.-P.Lees*[4], *E.Veitch*[5]

[1] Department of Physics, Imperial College, London SW7 2BZ, United Kingdom
[2] LAL, Université de Paris-Sud, IN^2P^3-CNRS, 91405 Orsay Cedex, France
[3] European Laboratory for Particle Physics (CERN), 1211 Geneva 23, Switzerland
[4] LAPP, IN^2P^3-CNRS, 74019 Annecy-le-Vieux Cedex, France
[5] Dept. of Physics, University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom

A data driven system which manages all the histograms used for online monitoring of the Aleph experiment is described. A relational database containing information about the histograms is used by monitoring tasks to book and save logical groups of histograms. The same database describes the contents and layout of pages displayed by a presenter, which can retrieve histograms either online or from archive, hiding from users details of their physical location. A variety of analysis techniques automatically search for anomalous histograms at regular intervals. Anomalies, filtered by databases of known problems, are reported to an error logger for display either as error messages or as histograms overlaid with reference copies of themselves. This system, which has been in use since April 1991, has improved the problem detection efficiency and reduced training requirements for shift personnel.

1 Introduction

The quality of data produced by any scientific experiment depends on the proper functioning of the apparatus throughout the data taking period. It is desirable to detect problems as early as possible, so that they can be fixed without compromising a significant fraction of the data. This is best done by monitoring the data online, but is only effective if the shift crews are able to detect anomalies quickly and reliably.

The Aleph Collaboration, consisting of ~400 physicists from 31 institutes, operates a detector assembled from a dozen independent components, with a total of ~500,000 readout channels and which relies on complex services and safety systems [1]. Data taking is continuous for typically seven months each year, with a few short breaks scheduled for accelerator studies and hardware maintenance. The complexity of the detector means that no one person is familiar with all its components; the length of the data taking periods makes it impossible to operate the detector with teams made up exclusively of all the necessary experts.

The monitoring system described here automatically filters the vast amount of information available online so that the attention of the small, non-expert, shift crews is drawn to real and potential problems. Tools with simple user interfaces and extensive online help minimise the amount of training required. The system makes full use of the distributed architecture of the Aleph readout [2] and is largely database driven, making it open to additions and modifications.

2 Aleph DAQ architecture

Details of the Aleph readout tree can be found in reference [2]. Subevents from different subcomponents of the detector are assembled in various stages of event building into a full event, two copies of which are read out into the online VaxCluster via two independent channels. The "main" channel, into a Vax 8700, is dedicated to data acquisition and logging; the "spy" channel, into a Vax 6510, is available for monitoring, which can thus be performed independently of the

main data stream. 25 Vaxstation 3000 workstations, also members of the same VaxCluster, are available for slow control, system initialisation, and aspects of monitoring such as histogram presentation and histogram analysis which do not require online access to the event data. Tasks, which can either run permanently or associated to a data taking run, can be added, suppressed or moved between VaxCluster nodes via database edits, making it easy to expand the system and to optimise the loading of the CPUs.

The Vax 8700 writes the raw data onto one of four dual ported disks. At the end of a run (every one to two hours) the disk is "switched" to the Aleph event reconstruction facility FALCON [3], also a cluster of Vaxstations, where the data is reconstructed for physics analysis. Production output tapes are usually available three to four hours after the end of a run.

3 Monitoring architecture

Figure 1 gives an overview of the Aleph online monitoring system. A variety of monitoring tasks process the raw data into histograms, which can be displayed online or offline and are analysed at regular intervals by tasks distributed throughout the VaxCluster, reporting problems to a single error logger. A histogram database implemented as a disk based global section and accessible from any machine in the cluster describes and drives the system.

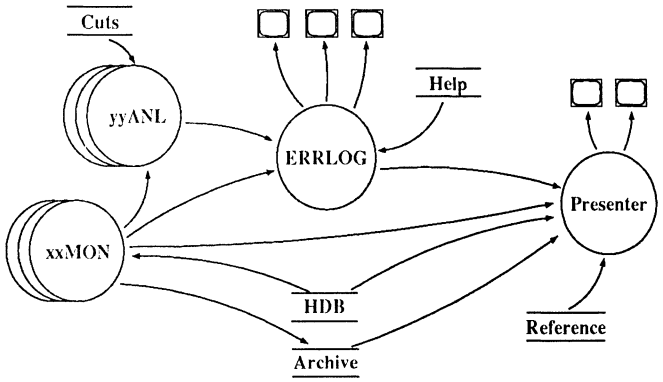

Figure 1: The Aleph online monitoring system

3.1 The histogram database

The Histogram Database (HDB) keeps a record of all the histograms being used in the Aleph Online System. It contains a full description of more than 3000 histograms in terms of the attributes specified in the standard CERN package, HBOOK [4]. It also contains details on how histograms are logically grouped. Groupings of histograms are made for a number of reasons:

- to find the set of histograms belonging to a particular "Task". This set is typically used when defining (i.e. booking) histograms.
- to find the set of histograms to be displayed together on the same "Page" by the histogram presenter. Information on the layout of each page can also be specified.
- to find the "Saveset" of histograms to be saved, reset or analysed at the same time.

The database (figure 2) was designed using the entity-relationship model following the ADAMO convention [5]. Groupings are inferred by navigating from one table to the next using the relationships. Thus the set of histograms belonging to a particular page is found by scanning

the group table for all groups associated with the page and then the histogram table for all histograms associated with each group. In practice the efficiency of this navigation has been increased substantially by introducing general attributes in each table which are used to chain elements of a table together. For example the list of all groups belonging to each page is actually maintained in the database and can therefore be projected out immediately.

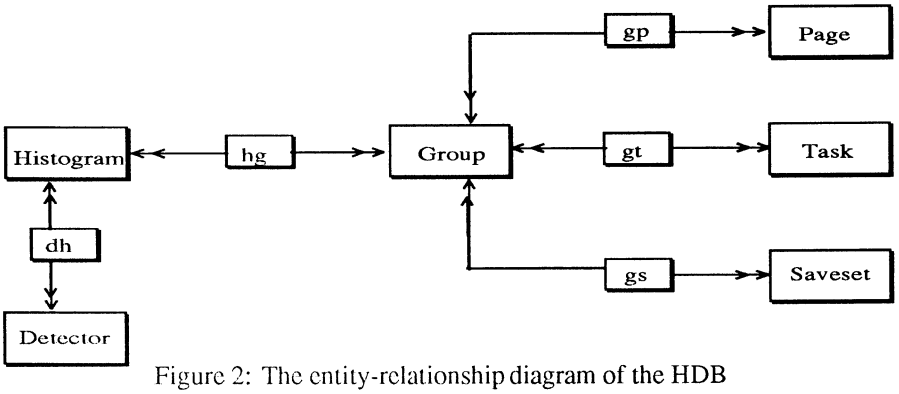

Figure 2: The entity-relationship diagram of the HDB

A set of access routines can be called by applications to modify the database contents or to retrieve information. This approach is required to avoid applications coupling directly to the data and for maintaining the integrity of the database contents. A package of histogram management routines layered above the HDB access routines is used to perform operations on sets of histograms, such as booking all histograms associated to a task, or saving a group of histograms to disk before resetting them. Whenever a saveset is written to disk, a bookkeeping record is added to a direct access log file. These bookkeeping files can be accessed via additional routines to retrieve histograms according to selection criteria such as run number, time of save, or histogram presenter page. Use of this package simplifies the application programmer's code and makes the system truly data-driven.

3.2 The error logger

The Aleph error logger receives messages from monitoring and analysis tasks, slow control, safety systems etc. Each message consists of a severity code, a three field error keyword, some descriptive text and an optional histogram identifier. The message is displayed on one or several alarm screens and/or logged according to the severity, and dispatched to any task which has declared an interest in errors with a particular keyword field. Examples of such tasks are the histogram presenter, which can immediately display anomalous histograms overlaid with normal reference copies, or the Expert System in charge of running the DAQ [6]. Messages can be retrieved offline using selections based on the keyword fields. Help information, linked to the error keyword, can be called up with a few key strokes for every displayed alarm.

3.3 Online monitoring tasks

All monitoring tasks are based on a template which hides details of run control and histogram management from normal users, who just tailor a set of user routines called by the template. The tasks are layered above the histogram management package, which they use at run time to book their associated histograms. These are stored in a section of shared memory, allowing online access from other tasks. The same package is used to save and/or reset savesets of histograms at the regular intervals specified in the HDB. The reset frequency of individual histograms is easily modified to optimise their usefulness: histograms need to be reset often enough to detect

new problems quickly, but rarely enough to smooth out statistical fluctuations in bin contents. For most applications one run (~800 multi-hadron Z decays) is a convenient interval; other applications need a much larger number of events of a specific type (e.g. 10000 Z's).

After an initialisation phase, during which it declares itself to the event buffer manager [7], a monitoring task waits for interrupts generated by run control messages, by the operator via a menu interface, or when a new event appears in the event buffer. On run control messages the template calls user specific initialisation or start or end of run routines. Menu input can change the control flow of the program, or be used to alter program parameters at run time. On event interrupts the template locks the new event in the event buffer, sets event type flags according to simple algorithms based on trigger bits and calorimeter energy deposits, increments event type counters and finally calls a user-written event processing routine which locates the appropriate data structures in the event and fills histograms; this routine may also generate and clear alarms via calls to error logger routines. When the event processing is finished, the lock on the event is released. Before returning from each interrupt a check is made on whether a condition for saving or resetting a saveset of histograms has been reached; if so the template automatically takes the appropriate action. The task then goes back into a waiting state until the next interrupt (which may be immediately if the next event is already queued). If the processing time per event is large compared to the data rate, only a fraction of the events will be queued for processing, to ensure that buffer space is always available for new incoming data.

The monitoring template is also used by applications not interested in histogram management, in particular trigger verification tasks which for each event calculate the expected trigger decision from the inputs to the various trigger registers. The expected decision is compared to the actual trigger decision and an alarm generated if the two don't match - one such program detected a hardware problem that was causing the loss of one trigger bit in a million events.

3.4 Presentation of histograms

Information contained in the HDB is used by a data driven histogram presenter to set up a tree structure of ~200 pages of histograms which can be called up by single mouse actions on an X-windows terminal. All information describing the contents, layout and labelling of each page is obtained from the database. The histograms are displayed using the HIGZ graphics package [8], with a special GKS emulation which calls the X11 library routines.

Several modes of operation are available. In online mode, the presenter searches the HDB for the name and VaxCluster node of the task filling each of the histograms in the requested page. If the task is on the same node, the presenter maps the histogram global section of the task, and uses the monitoring histogram directly. If the task is on another node, a message is sent to a dedicated server on the remote node, which maps the histogram global section, reads the requested histogram, and sends it as a message to the presenter. Reference copies of all histograms in the current page can be called up with a single mouse stroke and overlayed in a different colour on the online histograms. This allows inexperienced shift crews to decide whether histograms have a normal shape or whether they should call an expert because something is wrong. In history-by-run and history-by-time mode, routines from the histogram management package are used to retrieve the required histograms from disk; if more than one file was saved during the selected run range or time range, their contents are added together before display, allowing study of subtle effects that require more statistics than is available in a single file. Online mode is used during data taking by the shift crew. History mode is typically used every morning by subdetector experts to monitor the performance of their apparatus during the previous 24 hours, or to investigate problems which occured in their absence. In both cases ease

of use was a major design consideration - people can be trained to use the main features of the presenter in a few minutes and do not need to know any details of the physical location of online histograms, history files or reference files. A file mode is available, mainly for debugging by experts, where the operator can choose to view only the histograms stored in the specified file.

The presenter can also display a special type of HBOOK ntuple, in which we store the instantaneous value of several variables at regular intervals of, typically, one minute to one hour. These "time charts" can be viewed online for the most recent history and are regularly saved on disk so that the history of each variable versus time can be inspected several days or weeks later.

3.5 Automatic analysis of histograms

One aim of the Aleph online monitoring system was that the detection and reporting of anomalies should be as automatic as possible. A large body of code has been written which analyses the shape of histograms automatically, producing alarms if anomalies are found. When the monitoring task template saves a set of histograms to disk, it checks in the HDB whether a histogram analysis task is associated to this saveset. If so, it sends a message to that task. Histogram analysis tasks are also based on a template, which handles the communication with monitoring tasks. When a message arrives, the template reads into memory the appropriate saveset, then calls a user-written analysis routine.

A variety of methods have been used to analyse histograms, such as simple calculations of means, channel by channel comparisons with expectations (using reference files) or spike detection by fitting the histogram profile with a polynomial and flagging channels which give an anomalously large contribution to the chi squared. Any anomalies are tested against a database of known problems; a message is sent to the error logger if the problem is new. If the message contains the identifier of the offending histogram, the error logger forwards it to the histogram presenter, which adds the histogram to a menu of pending problems for verification by the shift crew or the appropriate expert. Analysis cuts are stored on disk, and can be changed via a menu interface to the analysis task. The same interface can be used to manually trigger the analysis of a set of histograms (useful for debugging) and to update the list of known problems.

3.6 Offline monitoring

The tools described so far are ideal for the early detection of electronic or control failures at the sub-detector level. It is also of interest to have the earliest possible warning of more subtle, but no less serious, failures which can only be detected at a later stage. For example, the tracking quality and efficiency, the stability of alignment, timing and calibration, and the position of the interaction vertex, can only be investigated after analysis of completely reconstructed events.

Two levels of monitoring have been provided in the FALCON processor farm in order to produce this information within a few hours of the end of a run. The first derives the data from the reconstruction program JULIA, which records $\sim$100 quantities for every event processed. These data can then be analysed by means of a purpose-written, flexible tool which permits event selection, the implementation of cuts, and statistical analysis [9]. The second is a Fortran program running on the data-set output from JULIA, which is activated immediately the reconstruction is complete. The results obtained are available directly in the Aleph control room for two important purposes: to warn of otherwise unsuspected problems and to make an assessment of the quality of the data for physics analysis. The tests which can be performed at this stage are very sophisticated, and the statistics limitation given by the size of any one data-set can be overcome by summing histogrammed results over many runs. In future high luminosity

running such automatic checks will become increasingly important.

4 Conclusion

Shift crews have found this monitoring system easy to learn and easy to use. Even very inexperienced shift crews are able to detect problems efficiently (though they may sometimes rely too heavily on the automatic features of the system!); in particular the histogram presenter has proved a very effective interface to the wealth of monitoring information available online and offline, and the error logger, together with its associated help files, a rapid way of alerting shift crews to new problems, allowing them to take the correct action promptly.

One problem is to keep a correct balance between early detection of problems and spurious errors. Also, it took some time to convince experts not to alarm the shift crew for failures which can not be handled before the next detector opening. A constant effort is needed to keep the help information and the list of known problems up to date, to implement new checks when new sources of failures are identified in the system, to suppress or modify histograms and presenter pages, etc. The abundant use of databases everywhere is very useful for this maintenance.

It is perhaps a law of online computing that the amount of monitoring required expands to fill the available CPU; the distributed nature of our system has made it possible to optimise the distribution of tasks so that virtually all the events can be monitored online, while maintaining a degree of sophistication which has allowed us to greatly improve the efficiency and speed of problem detection and to reduce by 30% the number of people required to run the experiment while also reducing their training needs.

Acknowledgments

We thank our colleagues in the Aleph Collaboration, and in particular W.Cameron, M.Pepe-Altarelli, J.Rothberg and M.Wunsch who made many useful suggestions during the design and implementation of this system and who provided some of the examples for this paper.

References

[1] D.Decamp et al. (ALEPH Coll.), Nucl.Instrum.Methods A294 (1990) 121.

[2] W.von Rüden, IEEE Transactions on Nuclear Science 36 (1989) 1444.

[3] M.Delfino et al., Computer Physics Communications 57 (1989) 401.

[4] R.Brun, D.Lienart, CERN Program Library Y251.

[5] S.Fisher, P.Palazzi, "The ADAMO Data System", CERN Computer Newsletter 207.

[6] P.Mato, T.Wildish, "DEXPERT: an expert system for read-out error recovery in the Aleph data acquisition system", these proceedings.

[7] D.Quarrie, CDF note 166, Fermilab.

[8] R.Bock et al., CERN Program Lbrary Q120.

[9] A. Bonissent et al., "New Computing Techniques in Physics Research", Proc. 1st Intl. workshop on software eng., A.I. and expert systems in HENP, 1990, Editions CNRS.