The FINUDA

Pre-analysis monitor

Author: Diego Faso (faso@to.infn.it)

Last update: November 24, 2005

Contents

Introduction

The FINUDA pre-analysis monitor has been conceived in order to allow an on-line evaluation about the quality of acquired data. The main issues related to the *P.A.M.* are:

- Scalers reading
- Luminosity evaluation
- Noisy events detection
- Beam/Interaction point status evaluation
- Communication with the DAΦNE Control Room
- Interface to the FINUDA-online web page

Values of scalers and luminosity are very important parameters, since they are sent to the DAΦNE monitoring system: they represent the most important feedback from the FINUDA spectrometer to the DAΦNE machine (concerning the quality of the accelerator/detector tuning). These informations are stored into text files, and regulary updated (scalers are updated every 15 seconds and luminosity every minute).
Several further parameters related with the status of the interaction poin and with the spectrometer detection reliability are also computed and stored into dedicated files. Many of these parameters are results of an offline event-processing: for this reason the pre-analysis monitor structure was designed in order to allow both online and offline processing, reading data from UDP socket and from file respectively.
The whole pre-analysis software is developed within the *fROOT* package, which is based on the *ROOT* framework.
The Object-Oriented structure of fROOT, allowed to recycle a lot of code and to implement the *P.A.M.* as an extension of the already existing code.

Chapter 1

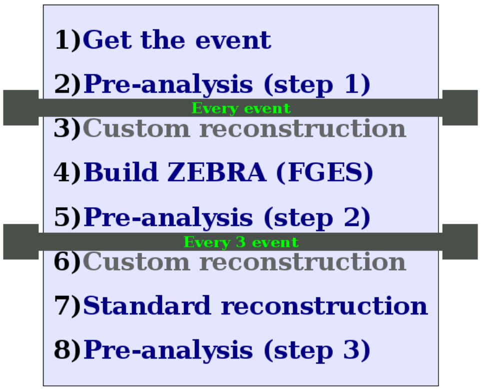
pre-analysis monitor structure and facilities

Every information provided by the *P.A.M.* is computed on raw-data, nevertheless not all the evaluated parameters require a complete reconstruction of the event. The pre-analysis proces must then run in a *smart way*, thus providing required results without performing unuseful operations. Furthermore the *P.A.M.* development started within a complex software structure, mainly composed of a C++ ROOT-based monitoring and a FORTRAN offline event reconstruction.
The main goal for the *P.A.M.* project was then to join all the facilities provided by the existing software, avoiding duplications of code, adding all missing features, and providing an fast and easy way for reading and reconstructing events.
Since the FORTRAN code (fidarc) can not be directly included inside the C++ structure, it is loaded by the *P.A.M.* as a library, hence the *P.A.M.* can control and steer it thanks to a software interface, which is implemented in the ‘‘TFndProcessRec” class [1].
The *preanalysis manager* (the ‘‘TFndPreanMan” class) provides the possibility of getting events bothe from UDP socket (online mode) and from a raw-file (offline mode); then the event processing is splitted into the following steps:

1. Get the raw-event (*optionally fill the fROOT HDT strycture*);
2. Get the required information directly from the raw-event (*1^{th} step of the pre-analysis process*);
3. Perform a custom reconstruction of the raw-event (optional);
4. Fill the FINUDA ZEBRA-FGES structure (*the FGES address is stored*);
5. Get the required information from the content of FGES (*2^{th} step of the pre-analysis process*);
6. Perform a custom reconstruction starting from the information stored in FGES (optional);
7. Perform the fidarc complete reconstruction and fill the FINUDA ZBRA-FDST structure;
8. Get the required information from the content of FDST (*3^{th} step of the pre-analysis process*);

This process of every event can be stopped at any time, thus allowing users to tune the pre-analysis performances according to their need. The example in Fig. 1.1 shows a configuration in which a fast event processing is obtained by performing the pre-analysis first step at every event, the second step every three events, and the third step every nine events. This configuration could be used for a fast online pre-analysis process, while an offline process allows to perform the three steps at every event, thus collecting the whole statistics on recontructed data.
In the end, the pre-analysis software inherits all ROOT facilities, such as data I/O management, GUI support, platform independence and virtual instruments for data storage and analysis.

[1] The structure of the ‘‘TFndProcessRec” class was directly derived by the ‘‘TProcess” class, written by V. Filippini and A. Panzarasa (2000)



1)Get the event

2)Pre-analysis (step 1)

3)Custom reconstruction

4)Build ZEBRA (FGES)

5)Pre-analysis (step 2)

6)Custom reconstruction

7)Standard reconstruction

8)Pre-analysis (step 3)

Figure 1.1: Pre-analysis event processing: process splitting example.

Chapter 2

pre-analysis monitor tasks

During the data taking, a lot of processes must run continuouesely in backgroung in order to control the status of the DAQ, check the quality of the events that are being collected and send feedback informations to the FINUDA DAQ and to the DAΦNE control. Some of the most important processes are:

- scalers reading;
- luminosity calculation;
- monitoring of the beam quality;
- evaluation of the quality of events;
- monitoring of the reconstruction program outputs.

The *P.A.M.* has been designed in order to join these backgroud processes together, thus improving their performances. In fact, the use of stand alone processes leads to:

- a lot of requests for events to the DAQ;
- a lot of waisted cpu time (*common tasks are performed by every process*);
- difficulty to control all processes (*they run on different machines and they must be started individually*);
- third parts dedicated to the communication between different processes (*it is often necessary to share some information by using text files or dedicated tables of the database*).

There is at list one operation required to all online processes: the request of events to DAQ via UDP socket...