H1OO - an analysis framework for H1

Judith Katzy for the H1 collaboration, DESY, Hamburg, Germany

Abstract

During the years 2000 and 2001 the HERA accelerator and the H1 experiment performed substantial luminosity upgrades. To cope with the increased demands on data handling, an effort had been made to redesign and modernize the analysis software. Main objectives were to unify the physics related software and to lower turn-around time for physics analysis by providing a single framework for data storage, event selection, physics analysis and event display. A new object oriented analysis environment was developped using C++ and the RooT framework. Additional persistent data layers for physics particles, event summary information and user specific information were defined. Links between all data layers and partial event reading allow for correlating quantities of different abstraction levels with high performance. Binding on demand of existing FORTRAN based libraries allows for (a) reuse of existing utility functions and (b) access of the existing data base. On this basis tools with enhanced functionality are provided. This framework has become standard for data analyses of previously and currently collected data.

MOTIVATION

The H1 experiment at the HERA electron proton collider at DESY started taking data in 1992. Until 2000 H1 collected 120 pb^{-1} luminosity corresponding to 35TB raw data and 3.5TB of reconstruction output. Analysis was done based on FORTRAN and FORTRAN based packages BOS[2], fpack[3], hbook[4] and PAW[5]. In order to reach higher luminosities a major accelerator and detector upgrade was done in the years 2000-2002. This upgrade provided a good opportunity for a software upgrade.

Main objective of the software upgrade is lowering the turn-around time for physics analysis by

- modularisation and unification of the analysis software
- centralisation of expert knowledge on physics algorithms
- providing a common particle concept
- improved data access

Because of the large amount of data already recorded it was required that the software is able to read previous data formats and that established FORTRAN code may still be used.

Since an object oriented framework suits best the requirements H1OO is written in C++ and based on the RooT analysis framework [1].

H1 DATA STREAM AND STORAGE MODEL

Fig. 1 shows the data stream and storage model of H1 data. H1 data are recorded and directly reconstructed using the FORTRAN reconstruction codes. The data are stored in BOS/fpack format on raw data tapes, subsets are copied on Data Summary Tapes (DST).

The DST files are converted into an Object Data Store (ODS) containing cluster, track and BOS bank objects as a 1-1 copy of the DST in RooT format to be accessed in the H1OO framework. The average event size in ODS format is 13kB.

The H1OO physics algorithms run on the ODS data to define physics particles that are stored in the μODS data layer. This is the main data layer for physics analysis with a size of 3 kB/event.

Event summary information that is used for event selection ("tagging") such as the number of particles of a given type, event kinematics, detector status etc. is stored as basic types on the H1 Analysis Tag (HAT) data layer. It contains 0.4 kB/event.

If specific information is needed for analysis it can be stored in so called User Trees that are filled with user specific code. ODS, μODS, HAT and UserTree are stored in RooT format.

Due to the small event sizes the μODS and HAT data layers are stored on a file server with fast access using rfio[1]. In general, ODS files are not stored persistently, since the access time to read ODS objects from disk is only increased minimally in comparison to reading the DST files and converting the information in memory. Due to its larger size, DST data are stored on tapes and are accessed using dcache [6].

DATA ACCESS

The different data layers are linked by the class H1Tree that provides access to all data layers in a single event loop transparent to the user. Since the DST file location is stored together with the μODS and HAT data it is not needed as input for the H1Tree.

Events can be selected using the information stored on HAT in basic logic operations. However, if information from other data layers or more complex calculations are

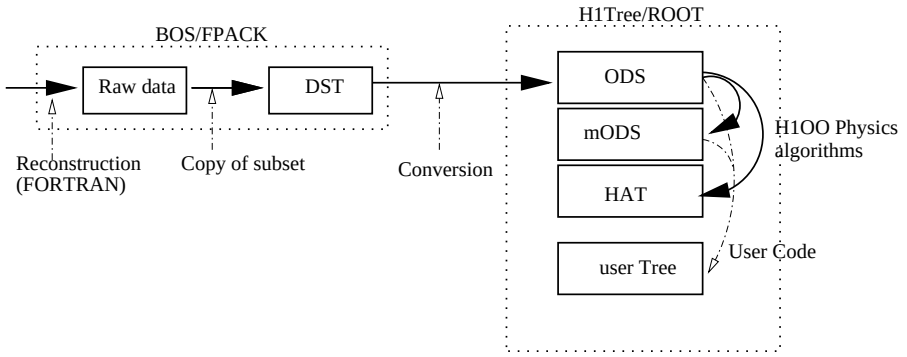

Figure 1: H1 data stream and data layers. The reconstruction of tracks and clusters is done with the established FORTRAN codes and stored in BOS/fpack format. Physics algorithms and particle finders are implemented in H1OO and the output is stored in the new data layers μODS and HAT in RooT format.

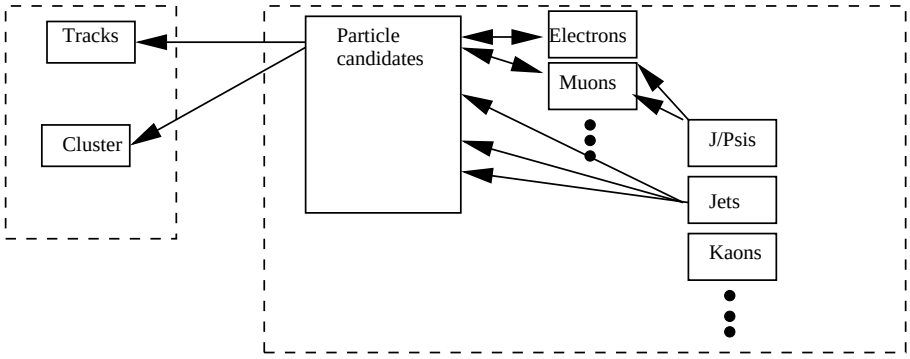

Figure 2: H1OO physics particles and their relation. Arrows indicate links between the different particle classes implemented as smart pointers. Even the reconstruction output stored on ODS and used to build the physics particle can be accessed transparent to the user.

needed for the selection, a separate selection program has to be written. The output can then be stored as a list of selected events, a so-called H1EventList. The list contains the link to the selected events, the selection string, and name and location of the associated files. This way subsamples of events can be stored without duplication of the data files. For example, eventlists for standard H1 data sets are provided together with the data.

The H1OO data are stored in RooT trees with one branch for each class. H1OO software provides smart pointer classes with a simple user interface that reads only the demanded branches transparent to the user. In addition, the pointer relation between objects stored in different branches is made persistent by identification of objects and tree branches with a unique integer ID. To keep the partial event reading capability, the referred object is only loaded when it is dereferenced.

With the H1OO extension to RooT I/O partial event reading is possible (via smart pointers) within one data layer and across different data layers which is crucial for the particle concept discussed below.

PARTICLE CONCEPT

The particle information on μODS is available as particle candidates, identified particles and composed particles see Fig 2.

The particle candidates consist of a track, clusters or a track-cluster link associated with one or more identified particles or the hadronic final state. The 4-Vector of the particle candidates correspond to the most probable particle hypothesis to avoid double counting of energy. The sum of the energies of the particle candidates corresponds to the energy of the final state. Each particle candidate stores the link to the tracks and clusters on ODS that they're build from. In addition, a list of identified particles that are based on this particle candidate is provided to allow for studies on multiple particle hypothesis.

Identified particles store their 4-Vector and keep detailed information related to the particle as used by the physics finder or as needed for further classification. In addition, they contain a link to the associated particle candidate that allows to retrieve the reconstruction level information of the particle. Composed particles store their 4-Vector and the link to the particles they're composed off. Among the identified and the composed particles multiple particle hypothesis based on the same tracks and clusters are allowed.

This way dependencies among the identified particles are avoided and the list of identified particles is extendable i.e. new particles can be added without effecting existing ones.

The links between the different particle classes are implemented as smart pointers and allow to collect all information related to the analysed particle even across the data layers with partial event reading.

PHYSICS ALGORITHMS

The H1OO framework provides physics algorithms for all aspects of physics analysis: particle finders used to fill the different data layers, tools to calculate event kinematics and luminosities, statistic tools such as neuronal networks, 3D event display with GUI, etc.. All physics algorithms have all been re-written in H1OO based on the current expert knowledge. During the development of the algorithms, the new H1OO implementations were first compared event-by-event to the established FORTRAN. In the meantime, the algorithms have been further developped and new algorithms are added to the H1OO analysis.

The code has been divided into modules with special care to keep it modular and extendable. The code is split into 45 packages that compile into a shared library each. Circular dependency between the core framework packages are avoided. Steering of the physics algorithms and the data handling is based on the RooT command line interpreter CINT [7].

SUMMARY

The H1OO analysis framework has successfully been established as the standard tool for physics analysis in H1. The key of the success was the clear definition of the scope of the project:

- A code development group takes care of the technical challenges, such as encapsulation of the data handling. The physics working groups develop algorithms and add their code via well defined interfaces. Special workshops are organised on the unification and implementation of physics algorithms across different working groups.
- The end users obtain a nice and easy-to-use product integrating all analysis specific tools into one single framework.

The physics analysis all greatly profit from the new and enhanced analysis environment. The framework is widely accepted within the Collaboration: most of the results presented in recent physics conferences were obtained using the H1OO framework.

REFERENCES

[1] Rene Brun and Fons Rademakers, “ROOT - An Object Oriented Data Analysis Framework”, Proceedings AIHENP96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. Meth. in Phys. Res. A 389 ,81-86(1997).
See also http://root.cern.ch/.

[2] V.Blobel, “The BOS System - Dynamic memory management”, DESY Internal Report R1-88-01 (1988).

[3] V.Bloble, ”FPACK - a general standalone package for machine independent data input/output”, H1 internal report (1991)

[4] http://wwwasdoc.web.cern.ch/wwwasdoc/hbook_html3/hboomain.html

[5] http://wwwinfo.cern.ch/asd/paw/

[6] http://www.dcache.org

[7] M. Goto, “C++ Interpreter - CINT”, CQ publishing, ISBN4-789-3085-3 (Japanese)
Rene Brun and Fons Rademakers, “ROOT: An object oriented data analysis framework”, Linux Journal 998July Issue 51, Metro Link Inc, (English)