Design and Implementation of the CLEO III Data Access System

Chris Jones [a], Simon Patton [b], Martin Lohner [a], Paul Avery [a]

[a] *Department of Physics, University of Florida, Gainesville, FL 32611 USA*

[b] *School of Physics and Astronomy, University of Minnesota, Minneapolis, MN 55455-0112 USA*

The CLEO III data access system is the "glue" between the user's analysis code and the data sources (files, databases, etc.) that hold the data. It is based on the concept of a Frame which describes the state of the CLEO experiment at any moment in time. The system is designed to provide Frames requested by a physicist. This design allows for data to be accessed from different types of sources (files, databases, persistent object stores, etc.) in a method that is transparent to the user. This paper explains the Frame model, describes the design of a system to serve Frames, and details the implementation of a data access system prototype.

Key words: Data Model; Data Access; Database; CLEO

1 Introduction

The CLEO III experiment will collect approximately 100 TB of data during the first few years of its lifetime. The challenges facing CLEO III are how to analyze such a large dataset efficiently and flexibly while providing a smooth transition from the current CLEO II environment to the new environment. The data access system is the "glue" between the user's analysis code and the data sources (files, databases, etc.) that hold the data. The data access system allows the different analysis frameworks (C++, FORTRAN, script based analysis language, etc.) to access data from the different types of data sources in a uniform manner. This paper will present the underlying conceptual (i.e. mental) model of the data access system, a design of the system, and results of a system prototype.

2 Conceptual Model

When a physicist wishes to process some of CLEO's data she wants to know the state of part, or all, of CLEO at one or more instances in time. Some parts of that state, such as geometry, are fixed over long periods of time, while other parts, such as ADC counts, have a very short lifetime. Data which are related by lifetime and by concept are parceled up into packets of information, which are called Records. Some examples of a Record are an event, information about the beginning of a run, and the explicit particle decay chain found by a user's analysis code.

Clearly as time progresses and the state of CLEO changes new Records need to be created. The set of Records describing the same portion of the state, but at different times, is called a Stream. A number of different Streams are required to build up the complete information about the state of CLEO. This complete set of Streams is called the Datastream. A complete picture of the state of CLEO can be constructed by collecting the most recent Records in each Stream. This set of "most recent Records" is called a Frame and is the structure through which data is accessed. Figure 1 summarizes this graphically.

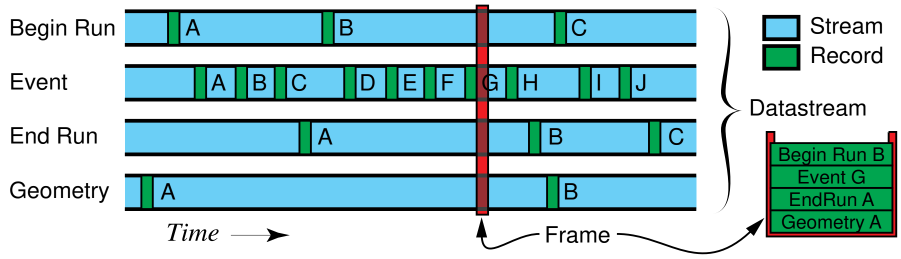

Fig. 1. Graphical representation of the Frame model.

Normally a physicist wants to perform an action on CLEO data whenever a particular type of Record appears in its Stream, e.g. a new Event Record (so they can analyze it), a new End-Run Record (so they can count the accumulated luminosity), a new Hardware Record (so they can monitor changes in high voltage), etc. To accomplish this the physicist can specify a list of one or more Streams. Every time a new Record appears in one of these Streams the relevant Frame for that time is created and passed to the physicist for analysis. The Streams specified by the physicist are call Active Streams.

The Frame model has several nice characteristics. It mimics how data is collected from the detector so physicists already understand how it works. In addition, new types of data ,e.g. user specific data. can be accessed simply by adding a new Stream to the Frame. Moreover, since all Records are treated equally, it is as easy to do analyses using Hardware records as it is using Event records. Finally, it is easy to study correlations of data with time, e.g. silicon detector noise versus time between beam crossing and trigger.

3 System Design

In the following discussion, it is important to remember that this data access system is meant as a "glue" layer between the data sources and the physicist's analysis framework. Therefore the direct user of this system is the analysis framework developer, not the average HEP physicist.

The system model is designed to follow the conceptual model. Therefore in the design we have an object called a Frame which is a container of Records. Those Records have methods which can retrieve CLEO data. The biggest difference between the conceptual model of the data and the system model is availability of data. In the conceptual model, all the data is available and properly ordered in a continuously flowing Datastream. In actual use, a physicist may want to start his analysis anywhere in the Datastream, not just at the beginning. This means there must be a facility that can create a Frame at an arbitrary point in the Datastream. In addition, the data will be stored in multiple data sources (user data in a file, hardware data in a database, etc.) and all of the data sources may not be available at every CLEO collaborator's site.

The data access system must handle the following tasks: connect Streams to data sources in order to read the the Records in those sources, synchronize the different Streams so that they return Records that are related in time, and assemble a Frame from the retrieved Records. The following objects are used to implement this behavior:

FrameDeliverer coordinates the other classes as well as creates each Frame required for analysis.

SyncValue designates the position of a Record within the Datastream. SyncValues monotonically increase over the lifetime of the experiment.

DataSourceBinder binds one or more Streams to a single data source.

DataSourceController manages a data source, determines the next available record in each Stream bound to that data source and finds the "most recent Record" in each Stream for a given SyncValue.

The analysis framework must tell the FrameDeliverer which data sources to use and which Streams should be read from which data sources. This is accomplished by creating the appropriate DataSourceBinder, e.g. a Zebra file DataSourceBinder, with the appropriate information, e.g. the file name and to return Event Records. These DataSourceBinders are then passed to FrameDeliverer and are used to create the necessary DataSourceControllers. The analysis framework must also provide the FrameDeliverer with a list of Active Streams. The framework may change the data sources and Active Streams at any time.

Once a FrameDeliverer has been set up, the only action an analysis framework needs to do is to request another Frame of interest. To get that Frame, FrameDeliverer queries each DataSourceController associated with an Active Stream to find the SyncValue of the next Record in that Stream. The FrameDeliverer then chooses the lowest-value SyncValue and tells all DataSourceControllers to synchronize to that value. The FrameDeliverer then gets the necessary Records from each DataSourceController to build a Frame which is then returned to the framework. The framework developer must write code to translate the data stored in the Frame to a structure that can be read by her framework, e.g. to COMMON blocks for a FORTRAN framework.

4 Prototype System

A prototype of the data access system has been written using C++. The prototype can access a subset of present CLEO II data either by directly reading a file or through a fast data server described elsewhere [1]. In the near future we will be able to read data from a database [2] and a persistent object store [3]. In the initial version of the prototype, data was stored in tables held by the Records. This was chosen since FORTRAN and C++ can both "easily" read tables. On the next iteration of the prototype, the Records were changed to contain C++ objects. That way, we can do any necessary data corrections in C++ and then the FORTRAN data can be filled from those objects. The data access system is presently being used by the prototype CLEO III event display [4], to develop CLEO III tracking code, as well as to rewrite calibration code for CLEO II.

This work is funded by the Department of Energy, grant DEFG0586ER40272.

References

[1] Paul Avery *et.al.*, *A High Performance Data Server Optimized for HEP Data*, CHEP97 Conference Proceedings.

[2] Michael Athanas *et.al.*, *(Object) Relational Databases for HEP Data*, CHEP97 Conference Proceedings.

[3] Michael Ogg *et.al.*, *Experience with a Persistent Object Manager for HEP data*, CHEP97 Conference Proceedings.

[4] C.D. Jones, P.R. Avery *A Novel Approach to Designing an Event Display*, CHEP97 Conference Proceedings.