EVENTSTORE: MANAGING EVENT VERSIONING AND DATA PARTITIONING USING LEGACY DATA FORMATS

C.D. Jones, V. Kuznetsov, D. Riley, G.J. Sharp,
Cornell University, Ithaca, NY 14853, USA

Abstract

HEP analysis is an iterative process. It is critical that in each iteration the physicist's analysis job accesses the same information as previous iterations (unless explicitly told to do otherwise). This becomes problematic after the data has been reconstructed several times. In addition, when starting a new analysis, physicists normally want to use the most recent version of reconstruction. Such version control is useful for data managed by a single physicist using a laptop or small groups of physicists at a remote institution in addition to the collaboration-wide managed data.

In this paper we will discuss our implementation of the CLEO-c EventStore, which uses a data location, indexing and versioning service to manage legacy data formats (e.g., an experiment's existing proprietary file format or Root files). A plug-in architecture is used to support adding additional file formats. The core of the system is used to implement three different sizes of services: personal, group and collaboration.

INTRODUCTION

Over the next five years, the CLEO-c experiment will collect hundreds of Terabytes of data focusing on the decays of the charmed quark. These data will be used for precision measurements of CKM matrix elements, glueball searches, and precision tests of QCD predictions calculated via Lattice QCD and Heavy Quark Effective Theory techniques, among other topics. From experience, we expect the reconstruction process will be performed several times, as the reconstruction software evolves and improves. One of the challenges for physicists analyzing such data is maintaining a consistent set of data and Monte Carlo simulated data, despite the introduction of new versions of the reconstructed data.

The CLEO III data storage system[1], while highly flexible, did not provide good support for managing multiple versions of the reconstructed data and Monte Carlo. The CLEO-c[4] EventStore was designed as a general purpose framework to manage the event data from the CLEO-c experiment, simplifying many common tasks of data analysis by relieving physicists of the burden of data versioning and file management. Rather than invent a new data format, EventStore was designed to manage data stored in a variety of formats, including the CLEO-c raw data and reconstructed object formats, and ROOT[3]. Data stored in the various formats are versioned as a group, so that physicists conducting analyses are always presented with a consistent set of data, and can recover the version of the data used previously. With EventStore, data in different formats, including overlapping skims, can be accessed at the same time with a common interface. It provides a physics-oriented interface for selecting data based on "run conditions", such as the beam energy or status of various subdetectors. Adding new data to EventStore is simple, as is superseding old data while retaining the ability to recover the data superseded. It was designed to add minimal overhead to sequential access to the data, and to speed random access to data formats that do not have a native index.

ARCHITECTURE

The CLEO-c EventStore holds the raw data produced by the CLEO-c detector, reconstructed data, user-generated skims, and Monte Carlo simulated data. Data can be stored in native file formats, which can be relocated and replicated for load balancing and staged in and out of HSM storage. The EventStore consists of several components (see Fig. 1):

- An EventStoreModule, which is loaded into the CLEO III/c analysis framework, SUEZ[2].
- A management layer, which provides versioning data and the locations of the indexing data.
- A data delivery layer, which uses the indexing data for rapid access to the data.

The management layer uses a relational database to hold the run conditions and version data, and the locations of the indexing data. Where each of these components is implemented depends on the EventStore size, as discussed next.

EventStore Sizes

The CLEO-c EventStore comes in three sizes, tailored to the scale of the application: personal, group and collaboration. The only user interface differences between the three sizes is the name of the module loaded, which is also the first word of all EventStore commands. The difference in names is required for SUEZ to load different EventStore sizes simultaneously.

The "personal" EventStore is meant to manage user skims on a personal system such as a laptop or desktop. It is designed to provide the versioning and metadata query facilities of the EventStore with minimal overhead. The relational database is implemented using the embedded SQLite database, making the personal EventStore completely self-contained in the SUEZ EventStoreModule. No daemons

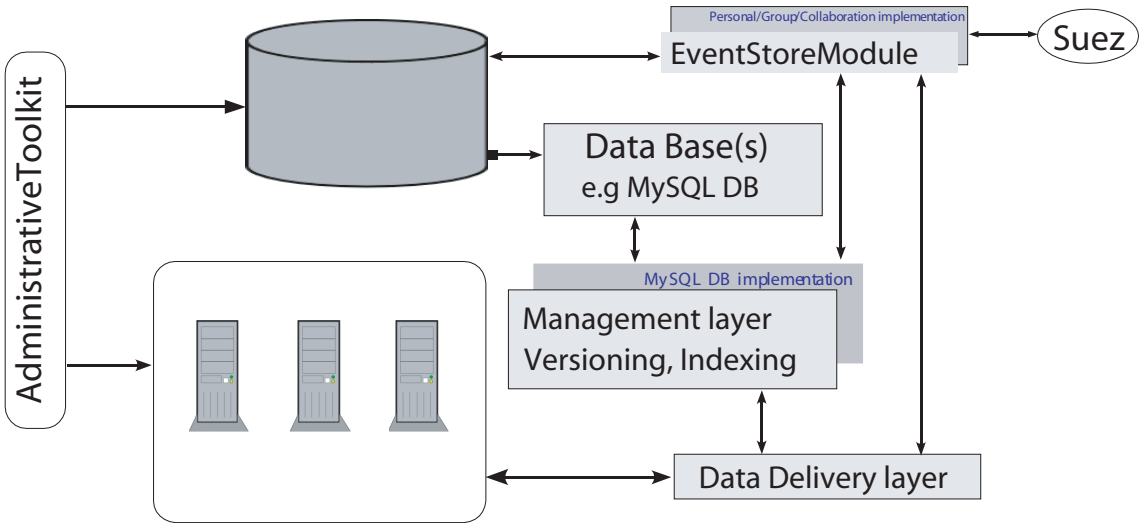

Figure 1: The EventStore Architecture. The shaded boxes represent a particular implementation.

or other system administration tasks are required beyond installation of the CLEO-c software packages.

The "group" EventStore is designed for managing substantial amounts of data residing on a pool of disks. A relational database server (currently MySQL) is used for better performance and scalability. More complex physics queries are supported. The typical application for the group EventStore would be a second tier analysis facility.

The "collaboration" EventStore adds more sophisticated data management facilities to the group scale system, including support for hierarchical storage, data replication and relocation, and a distributed directory service. The collaboration scale EventStore would typically be used at a first tier facility holding copies of all the CLEO-c data.

The CLEO analysis framework can load more than one size EventStore at the same time, synchronizing data access between all the modules loaded. This allows a physicist to begin an analysis on a collaboration or group scale facility, export a skim to a personal EventStore to continue the analysis, and later reconnect to the larger facility and access additional data for the events while retaining access to the data in the personal EventStore. We believe this capability to easily resynchronize a personal store with a larger scale store is essential to reducing the "I can't check that, it isn't in my ntuple" phenomenon.

Data Organization

The data in the EventStore is presented to physicists organized into "grades", representing phases in the data management lifecyle. Examples of grades are "raw", "reco", and "physics", representing raw data directly from the detector, recently reconstructed data, and reconstructed data approved for physics analysis, respectively. Skims are defined to be subsets of the events in a grade. A skim is implemented as an index, independent of the clustering of the data on the storage system. Multiple skims can reference the same event data, and it is easy to create additional skims. Data of additional types can also be added to a grade. This can be used to avoid common run-time calculations, such as π^0 identification or shower energy corrections, by adding the results of the calculation to a grade. In combination with the EventStore versioning system (discussed below), storing the results ("materializing", in database language) of common run-time calculations guarantees consistency when the data are reprocessed, by decoupling the analysis from subsequent changes to the code and calibrations that are inputs to the run-time calculation.

Once the EventStoreModule is loaded into the framework, the interface to specify the grade and date is simply "eventstore in 20040501 physics". "physics" is the default grade, so it need not be specified, and there are additional optional arguments to specify a specific skim or range of runs. The physicist does not need to know any paths or filenames to access the data, as that is all managed by EventStore.

Metadata

Metadata about the data are stored in a relational database supporting the standard SQL query language. All but the lowest layers of the database interface code are independent of the database implementation, allowing us to

easily use different implementations for the different scale systems as appropriate. The metadata stored include the mapping from EventStore internal logical file IDs (64-bit numbers) to the file path, the assignment of sets of data to grades and skims, the run conditions data used for physicist queries identifying data to analyzed, and the versioning and provenance data discussed later. The metadata are accessed as a web service, allowing the personal EventStore to take advantage of the more sophisticated physics queries supported by the full metadata database by accessing it over the Internet.

Examples of simple physics queries for run selection include:

- run range: "runs 202000 203000"
- dataset: "datasets 31-34"
- energy:
  - energy 1.89
  - energy psi(3770)
  - energy psi(3770)-off

Queries on the energy return the runs with energies clustered around the requested beam energy. Named energy ranges include the energy range below the resonance used for continuum background, with "-on" and "-off" names specifying the individual ranges.

Indexing

To provide efficient data access, EventStore creates two kinds of additional index files. The location independent index file maps a run and event number (and an optional Monte Carlo ID tag) to a record number in the format dependent location files. A location file includes a header indicating which data sources (e.g., which data files of a particular format) are indexed by the location file, and what objects within that source are indexed by that location record. The subsequent fixed-length records in the location file contain, for each event, "accessors" for rapid access to each of the objects indexed by that location file. A typical accessor is an offset within a flat file, but it could also be a key within a relational database or an object ID in an object database. There may be multiple sets of indexing data for a given data set, corresponding to different skims. For typical CLEO-c data, the index and location files add roughly 5% disk space overhead.

Versioning

The tradition at CLEO has been to continue to use the version of the data an analysis was started with throughout the lifetime of that analysis, unless there is a compelling reason to repeat the analysis with a newer version of the reconstructed data. With multiple newer versions of the reconstructed data appearing during the lifetimes of some analyses, this can impose a substantial burden on the physicist to track which versions of of the data were used over the lifetime of the analysis, particularly as data collected after the analysis began are added to the dataset used.

At the lowest level, the EventStore solution to this problem attaches versioning information to every piece of derived data, identifying how that data were produced. As an example, the version identifier Recon-20040312-Feb13_04_P2 indicates that the data were produced by the Feb13_04_P2 release of the reconstruction software, and that March 12, 2004 was the date of the most recent change to the software or inputs (e.g., calibration data) to the reconstruction that might affect the results. Similar tags identify later processing steps, such as π^0 identification. The full version identifier for a given piece of derived data is the union of the identifiers of all the inputs to the computation. While version identifiers conceptually apply to individual objects, in our current implementation version identifiers are associated with files, with the additional restriction that all the objects in a file must be derived from the same computational chain. We hope to relax this restriction in the future, so that sparse skims may be properly versioned.

Version identifiers are organized into grades by associating a grade with a list of run ranges and a list of version identifiers for each run range. Assignment of data to grades, particularly to the "physics" grade, is an administrative procedure performed by the CLEO officers. The evolution of a grade over time is recorded, so a consistent set of data is fully identified by the name of a grade and a time at which to "snapshot" that grade. For an analysis project, a physicist will usually specify "physics" grade data (the default) and use the date the analysis project started (e.g., 20040501) as the timestamp, so that the same consistent version will always be used. EventStore finds the most recent snapshot as of the specified date, so the date specified is not limited to a set of 'magic" dates.

If some of the data are reprocessed, such as redoing the π^0 identification, that change will not appear in the snapshot used by the analysis unless the physicist changes to a date after that change was made. However, data added to the dataset for the first time, such as data recently taken and reconstructed for the first time, or the addition of a new object, will appear in the snapshot. This is done so that a physicist can add data collected after the beginning of the analysis without having to change to a later timestamp.

Our future plans for the versioning system include embedding more of the versioning data in all output files, so that files are "self-identifying", and implementing a tool which will automatically generate Monte Carlo simulated data to match the data used in an analysis.

PERFORMANCE

Since EventStore does not define a new storage format or require a particular storage format, performance accessing the data is determined primarily by the underlying storage formats used, plus any overhead added by the EventStore indexing. Two tests of the indexing overhead have been performed, comparing the performance of data in the CLEO-c native object storage format, PDS, to accessing the same data files via EventStore. PDS is an unin-

dexed sequential file format, so in both cases the direct access to the PDS files is sequential. The only object stored in these files is the event header, which takes 12 bytes per record plus a small amount of overhead for the storage format.

Figure 2 shows the performance of strict sequential access, with EventStore approximately 12% slower than direct access to the file. The EventStore indexing is of no benefit for sequential access, so this is a worst-case measurement of the EventStore overhead.

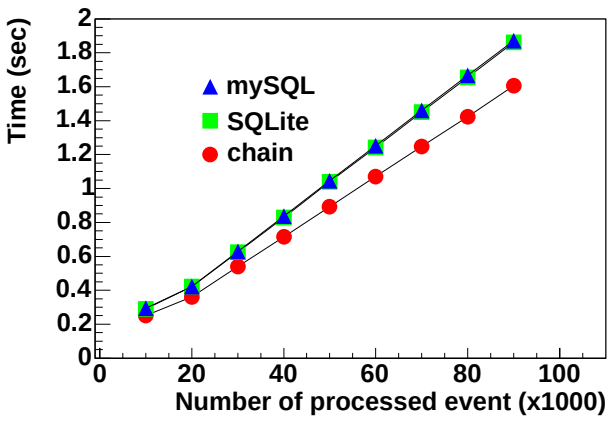

Figure 2: Sequential access, with and without EventStore

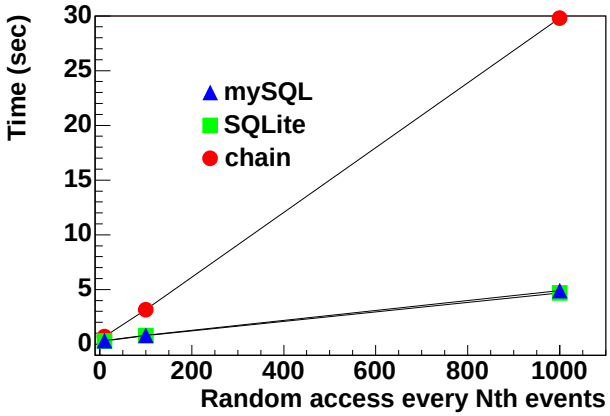

Figure 3: Random access via EventStore compared to file format native sequential access

Figure 3 shows the performance when driven by an event list skipping n-1 out of every n events, for n of 10, 100 and 1000. Since direct PDS access is sequential, the times for direct access are simply n times the time for sequential access to the same number of events. The factor of 6 improvement of EventStore is due to the additional indexing information allowing random access. Because of the small size of the records, both tests read the same amount of data from the PDS file, so the difference in performance is entirely due to the processor time required to decode the n-1 intervening records in the sequential access case, minus the overhead of processing the EventStore indexing information. This is also a worst-case measurement, as the sequential access performance will degrade as the event size is increased, while the EventStore performance will not.

CONCLUSION

The initial version of the personal and group event store have been deployed, providing several benefits. The system

- is adaptable to a wide range of existing data formats, including flat files and relational or object databases;
- provides random access to events, even with unindexed file formats;
- allows incremental addition of data and data lifecycle management;
- seperates event indexing from data clustering
- provides a powerful versioning system to maintain consistent views of the data
- has a simple user interface supporting physics oriented queries

Reception has been positive, largely due the simpler interface it provides, as the system has not been deployed long enough for the benefits of the versioning system to show. Work is proceeding on expanding the scope of the metadata available for physics queries, user-friendly administrative interfaces, and implementation of the more sophisticated data management features of the collaboration scale event store.

ACKNOWLEDGEMENTS

This work was funded by the National Science Foundation of the United States of America.

REFERENCES

[1] M. Lohner and C.D. Jones and D. Riley, CLEO III Datastorage, International Conference on Computing in High-Energy Physics and Nuclear Physics (CHEP 2000), Padova, Italy, February 2000

[2] M. Lohner and C.D. Jones and P. Avery, SUEZ: Job Control and User Interface for CLEO III, International Conference on Computing in High-Energy Physics and Nuclear Physics (CHEP 1998), Chicago, Il, August 1998

[3] R. Brun and F. Rademakers, ROOT Reference Guide, http://root.cern.ch/root/Reference.html

[4] R.A. Briere and others, CLEO-c and CESR-c: A new frontier of weak and strong interactions, CLNS-01-1742, http://www.lns.cornell.edu/public/CLEO/spoke/CLEOc