The CDF Run II Data Catalog and Data Access Modules

P. Calafiura[1⋆], *J. Kowalkowski*[2], *S. Lammel*[2], *M. Lancaster*[3], *F. Ratnikov*[5]
E. Sexton-Kennedy[2], *I. Sfiligoi*[4], *T. Watts*[5], *E. Wicklund*[2]

[1] Lawrence Berkeley National Laboratory, USA
[2] Enrico Fermi National Laboratory, USA
[3] University College, London, UK
[4] Istituto Nazionale di Fisica Nucleare, Frascati, Italy
[5] Rutgers University, USA

Abstract

The data access components of the CDF experiment Data Handling systems, together with the Data File Catalog, provide both the physicists running an analysis job and the production system running reconstruction jobs with a logical view of their input and output data, and transparent interaction with the storage management components.

Keywords: CDF data handling organization database schema input output

Overview

The next collider run of the Tevatron, Run II, will start in March of 2001. The CDF experiment will record and analyze proton-antiproton interactions at a center-of-mass energy of 2 TeV.

CDF will organize data by datasets which will contain events of the same physics properties and reconstruction version. A CDF Data File Catalog (DFC) will store information about the dataset. The Catalog will be implemented in a relational database; a prototype has been tested.

A user will analyze the data by selecting one or more datasets. A special data handling input module in the BaBar/CDF analysis framework AC++ [1], will coordinate the reading of all files belonging to a requested dataset using the Catalog and Disk Inventory Manager (DIM) software. An output module will write files of approximately 1 GB; data handling software will form filesets and transfer data to tape[2].

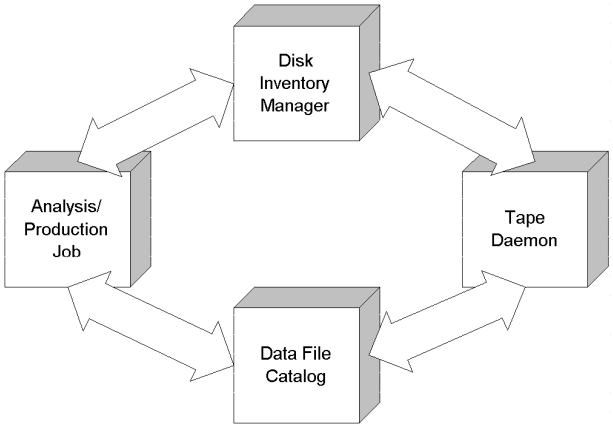

Figure 1: The Data Handling system components

⋆ mailto:PCalafiura@lbl.gov

The Data Access Hierarchy

CDF Data are organized in a hierarchical fashion: data streams store datasets, datasets are lists of filesets, filesets are groups of files, and files contain data from a range of luminosity sections.

A data stream is a stream of events that are stored together on disk files or tape partitions. A stream can be either primary (Raw Data) or derived (reconstructed data, mini-DST, etc) and will typically contain events belonging to several datasets.

Logically a dataset is a set of events with common physics properties (e.g top, W/Z, jets, etc.) From the Data Handling point of view, a dataset is also identified by the stream it belongs to and by a reconstruction version. Of course, an event may belong to more than one dataset, but datasets are grouped in data streams to minimize the number of events that are stored in more than one stream. We estimate to have 60-70 datasets grouped in about 8 streams coming out of production. Besides we expect to have to manage about 100-200 more derived datasets.

A fileset is a group of data files that are stored together in a tape partition. The number of files in a filesets is determined by the size of the resulting partition which in turn is chosen by optimizing the tape I/O performance. Typically a fileset will contain 5 files with a target size of 1 GB each.

Finally, a luminosity section (or run section) is a time-interval in data taking that CDF uses as unit for the integrated luminosity bookkeeping. The Data Handling must keep output files of derived data streams aligned to section boundaries (in other words they should keep all events of a given section in a single file).

It is interesting to notice how this hierarchy can also be seen as the combination of two interlacing hierarchies: the data handling hierarchy of streams $\rightarrow$ filesets $\rightarrow$ files and the physics hierarchy of datasets $\rightarrow$ luminosity sections

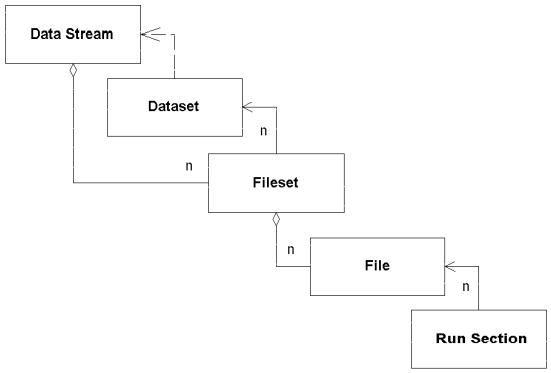

Figure 2: The Data Access Hierarchy

The Data File Catalog

The Data File Catalog[3] is based on a relational database and manages the information necessary to locate the data belonging to any CDF dataset (primary or derived). It also contains physics-related bookkeeping information such as data quality, trigger and filter used, average and integrated luminosity, dynamic trigger pre-scale values, etc. The data access granularity of the DFC is the luminosity section, in other words individual events are not described in the catalog, only groups of events, or luminosity sections.

The four core tables of the DFC schema correspond to four elements of the data access

hierarchy (Fig. 2): datasets, filesets, files and run sections. The DFC also has a table describing the tapes known to the system.

The DataFileDB package that performs the actual queries and updates to the DFC is built on top of another CDF package called DBManager[4]. DBManager provides a DBMS-independent layer to map relational table rows to transient C++ objects and to perform predefined queries and updates using composite key objects. DataFileDB currently supports interfaces to Oracle and mSQL.

The DataFileDB package provides classes and utilities for selecting information out of the database, and for writing information into the database. Using this package, one can make requests like, for example:

- get all the files from a dataset in a time (event/run number) range
- get all the files from a dataset in a luminosity section range
- find all the tapes used in this dataset
- find all the filesets of this dataset
- find the luminosity sections in this time range
- find all the files in this fileset or dataset

A set of auxiliary tables are available to log the progress of production jobs. This will make restarting a job after a crash more efficient. It will also be possible to implement a job checkpoint-restart mechanism.

Besides the C++ API used by the data handling AC++ modules, both command-line and web-based tools are provided to query and update the DFC. The command-line interface has been developed to allow the production farm control software (mainly python scripts) to interact with the Catalog.

The DFC component is accessed by almost every platform of the data handling system: the Level-3 farm, the reconstruction farm and the data logging system. Indeed one advantage of using a RDBMS like Oracle is that it provides an essentially transparent mechanism to share information across a distributed system.

The AC++ Data Handling Modules

Input Module

The input module allows the user to select her/his data sample in a logical fashion, in terms of datasets, runs, and luminosity sections. This selection criteria is used by the DFC to retrieve the list of the relevant filesets and files from the database. The input module will then ask the DIM to stage in the missing filesets and read through the staged files as they appear on disk.

Output Module

The output module will be mainly used by the production jobs to log their output files to tape. The reconstruction will filter and classify events that will be written to multiple output streams. Each output module requests blocks of DIM managed staging disk space of size matching the target file-set size. This is where it will write its output files. Once a block is completed (filled), a description of the block and the files within it will be inserted into the DFC. This action will trigger the staging of the completed output files to tape by the Tape Daemon, along with the request for a new block from the DIM. Once all the files in the block are staged out, the DIM reclaims the completed block space.

What make matters slightly more complex is the aforementioned requirement that output files must be aligned to luminosity section boundaries and the fact that events in the input streams

are not in general strictly time-ordered. This prevents us from closing a file when it reaches its target size and requires some extra bookkeeping to keep track of the outstanding luminosity sections. Even after the file target size has been reached one has to write all events belonging to the outstanding luminosity sections in the existing output file while the other events are written into a new file. Because the input file is aligned to section boundaries, we can safely close output files that have reached their target size once we close the current input file.

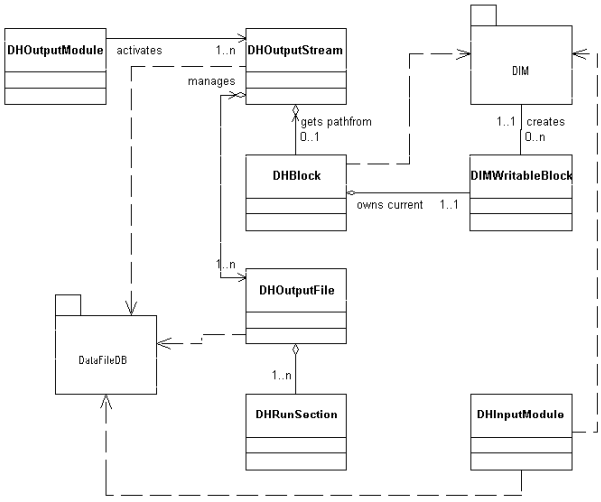

Figure 3: The Data Access Classes

Status and Outlook

As described in more detail elsewhere[5], all the Data Handling system components and their interactions with the Level-3 and reconstruction farms have been tested recently during CDF Mock-Data Challenge 1.

Although most of the data access components were at the prototype stage, the test validated their integration into a working system. Having established the component interfaces we can now work on improving the reliability and performance of the implementations.

References

1 Sexton-Kennedy E., "A Users Guide to the AC++ Framework" `http://www-cdf.fnal.gov/upgrades/computing/projects/framework/`

2 Lammel S. et al., "Overview of CDF Run II Data Handling System", this conference

3 Wicklund E., "Data-File Catalog for Data Analysis in CDF Run II" `http://www-cdf.fnal.gov/upgrades/computing/projects/catalog/data_file_catalog.ps` Kowalkowski, J., `http://ncdf16.fnal.gov/DataFileDB/doc/DFC_main.html`

4 Kowalkowski J. et al., "Mapping Auxiliary Persistent Data to C++ Objects in the CDF Reconstruction Framework", this conference

5 Watts T. et al., "Resource Management in, and Tests of, CDF Run II Data Handling", this conference