Fermilab FERMILAB-Conf-01/301-E November 2001

The CDF Run II Disk Inventory Manager

Paul HUBBARD[1], Stephan LAMMEL[1]
[1](Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA)

Abstract

The Collider Detector at Fermilab (CDF) experiment records and analyses proton-antiproton interactions at a center-of-mass energy of 2 TeV. Run II of the Fermilab Tevatron started in April of this year. The duration of the run is expected to be over two years.

One of the main data handling strategies of CDF for Run II is to hide all tape access from the user and to facilitate sharing of data and thus disk space. A disk inventory manager was designed and developed over the past years to keep track of the data on disk, to coordinate user access to the data, and to stage data back from tape to disk as needed.

The CDF Run II disk inventory manager consists of a server process, a user and administrator command line interfaces, and a library with the routines of the client API. Data are managed in filesets which are groups of one or more files. The system keeps track of user access to the filesets and attempts to keep frequently accessed data on disk. Data that are not on disk are automatically staged back from tape as needed. For CDF the main staging method is based on the mt_tools package as tapes are written according to the ANSI standard.

Keywords: data handling, data storage, data access, HSM, CDF, Run II

1 Introduction

The Collider Detector at Fermilab (CDF) [1] is a large multi-purpose particle detector at the Fermilab Tevatron. The experiment records and analyses proton anti-proton interactions at a center-of-mass energy of $\sqrt{s} = 2\,\mathrm{TeV}$. With the Tevatron being the world's highest energy collider, searches for new particles, like the Higgs boson, supersymmetric particles, and new vector bosons predicted by string compactification are on the experimental program together with precision measurement of electroweak parameters like the mass of the W boson, top quark parameters, and B physics.

The current collider run of the Tevatron, Run II, started in April 2001. During the over two year duration the CDF collaboration plans to record 1 PByte of data. This represents more than 20 times the data volume of the previous experiment. All data needs to be accessed multiple times, stretched over several years in which the data will be actively analysed. Direct access storage media is not affordable for this data volume. Instead, the data will be archived on a sequential media with the part of the data that is being actively analysed being cached onto magnetic disks.

To share data in a heterogeneous analysis cluster, data analysed frequently for a long time will be put into read-only filesystems and shared by all compute nodes in the analysis cluster. A software package, the disk inventory manager, manages the flow of data between disk, tape, and user analysis programs.

2 CDF Run II Data Analysis

The experiment is organizing its data in Run II hierarchically [2]. Users will access data mainly by datasets which are collections of events with similar physics properties and same information per event. The event data are stored in files, with files of the same dataset being grouped into a 5 to 10 GByte fileset.

The framework of the CDF Run II analysis programs was jointly developed with the BaBar experiment at SLAC. Special input and output modules [3] provide a convenient inferface to the CDF Run II data handling system.

The input module allows a user to select data by dataset. The module uses the CDF data catalogue [4] to translate the dataset into a list of filesets. It then passes the list to the client API of the disk inventory manager to be made available for read access. The client software contacts the server process to query the status of the filesets in the list. Filesets available on disk are processed while separate staging jobs are launched to copy "missing" filesets from tape onto disk. To process a fileset, a read reservation is registered with the server process (to prevent evicting the fileset from disk while it is in use), the location and list of files in the fileset is passed back to the application, i.e. input module of the analysis job, and when all files are analysed the reservation is released. A tuneable look-ahead window is used to provide uninterrupted data flow to the analysis program and to launch staging jobs just-in-time. Tape to disk staging of data is thus under control and coordination of the disk inventory manager. Tape jobs are submitted to the same batch system used to schedule the analysis jobs but using separate queues and accounting. With this approach a rather simple way of I/O resource management independent of the disk inventory manager is available.

3 Disk Inventory Manager

Disk space is managed through quotas, on the login, scratch, and spool areas and data disk space through a disk inventory manager.

Some datasets are accessed more frequently than others and, if small in size, copies can be fixed permanently on disk; such datasets are chosen by collaboration management. A disk can also be marked as read-only; such disks are useful to manually manage the cache.

Larger datasets and less frequented datasets are staged from tape to disk and reside in a disk pool for temporary use. A disk inventory manager coordinates the use of all the data disk space of static and cached data.

The quantum of data used for managing disk space is the fileset. Data transfers between disk and tape move all filesets on a tape in one job. It is filesets that are marked static or appear temporarily in shared space; the disk inventory manager manages filesets and keeps no knowledge of datasets as such. This serves to decouple the manager from the database system.

The code is split into a server process and client code. Commonly used client functions are compiled into a command- line client that is used by both users and administrators. Communications between client and server use TCP/IP sockets. Additionally, the analysis jobs use API functions via the library code.

With the above architecture the disk inventory manager can handle locally attached storage as well as network attached storage (NAS) and space inside a storage area network (SAN) for clustered configurations. The package uses the standard filesystems on each operating system and thus requires no kernel modification.

3.1 Server daemon

This is a multithreaded server program written to the POSIX C API, using the worker pool model. Its central task is to manage a cache for the tape system, meaning that it attempts to keep the most used filesets on disk and update the working set as required.

The server uses a dynamic threading system, where it will start more threads under load, and eliminate them when the load is reduced. Initially, the startup routine only launches one thread, which is then free to add others, up to a maximum, as required. The server tries to always have at least one thread waiting on the socket, and will attempt to start others if the socket is unattended. Additionally, it will start another thread if requests are coming in faster than a predefined rate. Threads will exit if they are idle and have been so for another predefined interval; this provides hysteresis and some defense against load transients.

Each thread attempts to get a client connection, and services the request from the client. As it progresses, the thread updates its status field so that other threads may track its progress, and also serves as a simple detection mechanism for hung or crashed threads. To handle signals, there is a dedicated thread; the worker threads inherit a signal mask that blocks all non-fatal signals.

User reservations and inventory changes are written to disk using internal transaction code that guarantees a valid saved state as long as the filesystem is intact. When combined with the command / respond communications, this allows restarts of the server without client disruption.

Every minute, a list of cleanup tasks is run to release expired reservations, save the reservation history, print out usage statistics, and other housekeeping minutae.

3.2 Cache algorithm

The filesets in the disk pool remain in place until space is needed to fetch more filesets. The code uses an order-N multiple pass algorithm that

- Vetos disks marked as read-only
- Vetos filesets that
  - are reserved by users
  - have just arrived on disk
  - are marked as static
  - are otherwise not removable
- Groups the remaining filesets by disk
- Scores each fileset based on the reservations, time on disk, and time since last usage.

Since space must be contiguous, we choose the disk with the lowest score for the space requested, and then evict as many low-scoring filesets as are needed.

Scoring of a fileset is based on the time since the last reservation, which yields a score linearly proportional to the time since the last usage. As the experiment proceeds, we anticipate that we will adjust this based on the use patterns that emerge from the collaboration.

3.3 User Identification and Quota

The disk inventory manager coordinates the use of disk space and I/O resources among users. It has to balance user requests in case the available resources are oversubscribed. To do this effectively, the disk inventory manager needs to be able to idenify users on both the local and other compute nodes in the cluster. External authentification based on UNIX user ids is being used. The identification information is encoded into a token that the client and server exchange.

To prevent run-away analysis jobs and individual users from using excessive resources, quota are being implemented. Quotas for fileset reservation, read-ahead staging jobs, space allocation to assemble new filesets, and allocation of temporary data space. Resources may be charged by a user either against her/his personal quota or against a group to which they belong.

3.4 Tape Handling Interface

The jobs transfering data from tape to disk and vice versa are rather independent of the disk space management process. However, before data can be copied onto disk, space needs to be allocated (and potentially filesets evicted by the server) and the status of the copy needs to be reported to the disk inventory manager after the operation is complete. For this handshake both an API and a command line interface exist, allowing easy setup of new staging methods. For CDF Run II ANSI file structured tapes are used. The CDF mt_tools package [5] provides convenient tools to read and write tapes according to this standard. A simple disk-to-disk copy is used as staging method for testing.

References

[1] F. Abe *et al.*, Nucl. Instr. and Meth. **A271** 387 (1988), see also `http://www-cdf.fnal.gov/pubcdf.html`.

[2] R. Colombo *et al.*, "The CDF Computing and Analysis System: First Experience", CHEP2001, Paper 4-036.

[3] F. Ratnikov *et al.*, "User Friendly Interface to the CDF Data Handling System", CHEP2001, Paper 4-027.

[4] D. Litvintsev *et al.*, "CDF Run II Data File Catalog", CHEP2001, Paper 4-037.

[5] L. Buckley-Geer *et al.*, "Overview of the CDF Run II Data Handling System", CHEP2000, Paper 366,
http://www.fnal.gov/fermitools/abstracts/ftt/abstract.html,
http://www.fnal.gov/fermitools/abstracts/ocs/abstract.html.