EPJ manuscript No.
(will be inserted by the editor)

FERMILAB-CONF-05-491-CD

CDF computing and event data models

F.D. Snider[1] for the CDF Collaboration

Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

Received: date / Revised version: date

Abstract. We discuss the computing systems, usage patterns and event data models used to analyze Run II data from the CDF-II experiment at the Tevatron collider. A critical analysis of the current implementation and design reveals some of the stronger and weaker elements of the system. which serve as lessons for future experiments. We highlight a need to maintain simplicity for users in the face of an increasingly complex computing environment.

PACS. 07.05.-t Computers in experimental physics – 07.05.Bx Computer systems – 07.05.Wr Computer interfaces

1 Introduction[1]

High performance computing is an essential component of most modern high energy physics experiments. In general terms, the computing resources required to produce physics results scales linearly either the total number of events, the total data volume or the average event logging rate. For many experiments such as CDF[1], the resulting demand for computing increases dramatically over the life of the experiment. The maximum data logging rate, for instance, will increase by a factor of three between 2004 and 2006; the volume of archived data during the final four years of operation will more than quadruple the volume of data from the first three. In short, the computing problem becomes more difficult with time and requires increasingly sophisticated and robust solutions to ensure that the systems scale stably. At the same time, users must be provided tools that hide the underlying complexity and make computing on increasingly large scales appear "simple".

The general computing strategy revolves around three basic principles. First, centralize and automate control of common and repetitive computing tasks, such as full event reconstruction, Monte Carlo (MC) production and major dataset stripping. Second, distribute computing hardware as needed to perform user analysis and MC production since no single facility can host the required computing. Finally, provide users with stable, simple interfaces to the broad range of resources deployed, thereby allowing them to focus on what they are computing rather than on the mechanics of how to do it.

In this talk, we describe the major elements of the CDF computing model, the current migration toward grid-based distributed computing to meet the increasing computing demands of the experiment and the mechanisms by which users access this computing power. We will then discuss some details of the Event Data Model (EDM), the principal interface between users and the data, the deficiencies of the EDM and several general ways to increase productivity through a better approach to the EDM. Finally, we review some of the strengths of the current computing model.

[1] Work supported by the U.S. Department of Energy under contract No. DE-AC02-76CH03000.

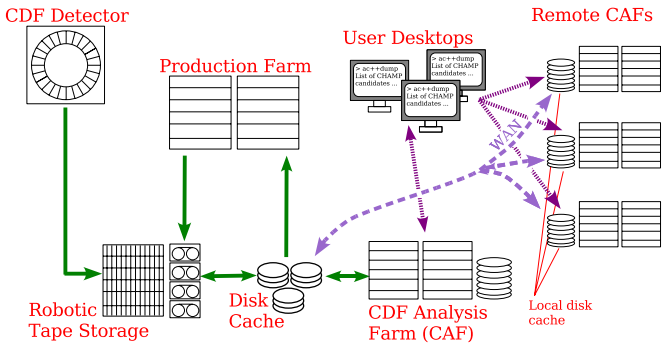

Fig. 1. CDF computing systems and data flow

2 Computing model and data flow

The central CDF computing systems consist of five major hardware components, as shown in Fig. 1: robotic data archive and disk cache; the "production farm"; local and remote CDF Analysis Farms (CAF)[2]; and interactive systems both at Fermilab and at remote institutions. Various collaborating institutions host the remote CAFs.

The arrows in Fig.1 represent the data flow, which proceeds as follows. Raw data from the detector is first logged

in the tape archive. It is then read into a disk cache, served to the production farm where full event reconstruction takes place and written back into the tape library. The production output is copied into another disk cache where it is served to the CAF or distributed to remote CAFs for further reduction and analysis. Users submit analysis jobs from interactive computers (primarily desktops and laptops) located anywhere to any of the CAFs. Results from the CAFs are written to the disk cache, into the tape archive or returned directly to the user.

A Data Handling (DH) system automates the cataloging and movement data between various elements of the computing system as well as delivery to analysis jobs. All data stored in the tape archive are under control of the DH system; a significant fraction of data in the disk cache is not.

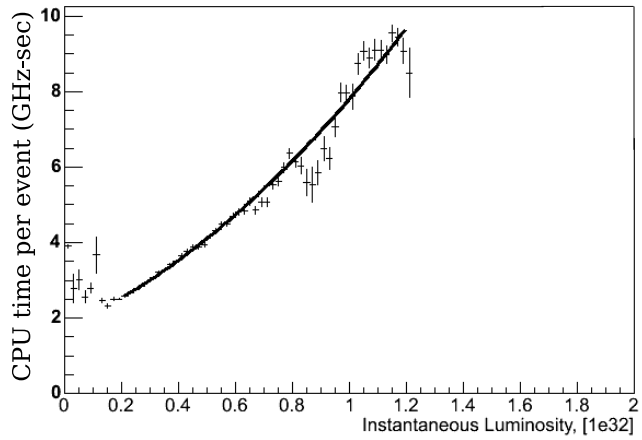

Fig. 2. Reconstruction CPU time per event. The time is approximately linear to luminosities of 2×10^{32} cm^{-2}s^{-1}.

3 Computing systems

In this section, we briefly describe salient details of the major computing systems.

3.1 Data archive and storage systems

Data from the experiment is archived in a robotic tape library with a total capacity of 2.2 PBytes using the existing tape technology. The total will grow to about 6 PBytes through the life of the experiment. A bank of 18 tape drives, each with a bandwidth of about 30 MBytes/sec, provides I/O to the contents of the library. A disk cache of approximately 370 TBytes stages data from the archive and from analysis jobs.

3.2 Data handling system

The Data Handling system is perhaps the most important and technically demanding of the various computing systems and consumes by far the largest development effort. Since all interactions with data involve the DH system, performance, robustness and scalability through the duration of the experiment are the paramount concerns.

The DH system fills several important roles. First, it archives the data and provides a metadata catalog by which the data can be managed. Upon request, the system provides high bandwidth channels to individual data files, moving data as needed to deliver a file or achieve high throughput, and organizing access to multiple files in an efficient manner. Throughout these operations, the DH system ensures the integrity of all delivered data. Details of these transactions are transparent to the user. The two major components providing these services are "SAM" (Sequential Access via Metadata[3]) and "dCache"[4].

The dCache system distributes data across an arbitrary number of local file-servers and mediates tape transactions. Clients specify files using fixed, location-independent identifiers. The system associates these identifiers to the physical locations and returns a direct I/O channel to the client. Each day, upward of 100k such file transactions are processed, routinely delivering up to 75 TBytes. More typical volumes are in the range of 10–25 TBytes per day, of which 5–10 TBytes are restored from tape.

SAM expands upon the capabilities of dCache by providing support for highly distributed data. SAM also includes features designed to aid handling of large datasets, such as a simple mechanism to create datasets based upon metadata queries, storage of file tracking information to facilitate data processing and tools to automate certain processing tasks. The production farm uses SAM to automate most tasks. At the time of writing, SAM is in the final stages of deployment to general users.

3.3 Production farm

The objective of the production farm is to reconstruct all raw data from the experiment as soon as possible after data taking. As the most predictable of the computing problems, all production tasks can be fully automated. The required computing is easily calculated from the event logging rate (130 Hz in FY2005 and 220 Hz for FY2006 and beyond), the average instantaneous luminosity (upward of 10^{32} cm^{-2}s^{-1} in FY2006 and beyond) and the required CPU time per event (Fig. 2).

The processing strategy for the production farm has three steps. In the first, a fraction of the data is processed within three days of data taking to provide data quality monitoring. The accumulated monitoring data from the previous month is analyzed in the second step to produce final calibration constants. Finally, all the data from that month is processed using final calibrations, typically within four to eight weeks after data taking. Using this scheme, the data from a given year is processed about 1.3 times with no later re-processing required.

With 150 dual processor PC's providing the equivalent of about 1.2 THz of PIII CPU's or 48k SpecInt2k's, the current farm can process about 18 million events per day.

Processing can expand into the CAF as needed in order to provide short term increases in throughput.

3.4 CDF Analysis Farm

The CAF[2] is the primary analysis platform for the experiment and contains the bulk of the available computing capacity. Activities on the CAF fall into three major categories. The first is user analysis, consisting of ntuple creation and analysis, and all other CPU-intensive calculations. Semi-coordinated activities run by the physics groups represent the second major set of tasks, and include secondary and tertiary dataset production and experiment-wide ntuple production. The final category includes production of large-scale MC datasets. Monte Carlo production and experiment-wide ntuple creation present reasonably predictable loads on the system, some of which scale with the rate of data taking. User analysis is less predictable and typically scales with the full dataset size.

CAF installations at nine remote institutions[2] contain approximately 40% of the 5.6 THz (PIII equivalents) of analysis computing at CDF. All installations exhibit near 100% utilization. The CAF at Fermilab typically processes in excess of 10k jobs per day for about 100 users.

3.4.1 CAF usage

Over the first 10 months of 2005, about 50% of the load on the CAF at Fermilab was in analysis of production output data, 20% in analysis of ntuples, 20% in MC generation and the balance in unidentified activities (see Fig. 3). At remote CAFs, the load fractions were about 40% in MC generation and 50% in either production data or ntuple analysis. Among organized analysis activities, the B-physics group consumed the majority of CPU cycles (Fig. 4).

Analysis jobs on production data required an average of 0.75 seconds of CPU per event (Fig. 5). About 20% of these jobs consumed more that one second per event and represented about 40% of the total CPU used in production data analysis. For comparison, event input and unpacking requires about 60 msec per event.

The long tail in Fig. 5 results from track re-fitting and vertex finding. These tasks are integral parts of a typical B-physics analyses and others that exploit the precision tracking available at CDF. Both of these tasks require the full analysis framework and must therefore occur in jobs that process production data rather than ntuples.

3.4.2 CAF user-level features

Simple graphical and command-line interfaces provide access to the resources within the CAF. Users specify a file to execute, the number of parallel segments that should be run and the site at which the jobs should be executed. The system creates a tarball from the input directory, ships it to the appropriate nodes and executes the specified file. Output can be directed to any location to which the user has write access. A number of quasi-interactive features are also available, such as the ability to look at log files or a directory listing on a specific worker node, or to connect a debugger to a running process on a worker node. Monitoring of CPU, memory and status for individual processes is also available.

[2] The institutions hosting remote CAFs are CNAF, Bologna, Italy; KNU, Korea; Academia Sinica, Taiwan; University of California at San Diego; Rutgers University; University of Toronto; Tsukba, Japan; Cantabria, Spain; and Massachusetts Institute of Technology.

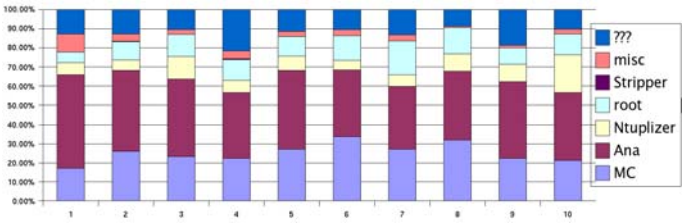

Fig. 3. CAF usage by task for the first 10 months of 2005. The task classifications include: Monte Carlo generation ("MC"); three categories of production output analysis ("Ana", "Ntuplizer", "Stripper"); analysis of ntuples using root ("root"); various small categories ("misc"); and unknown ("???").

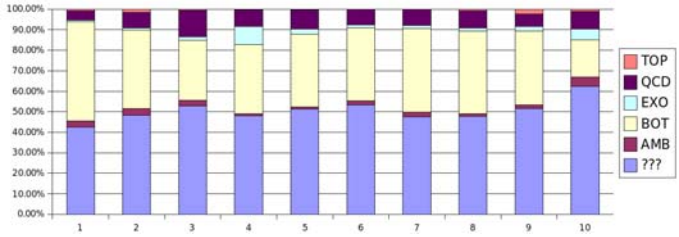

Fig. 4. CAF usage by physics group for the first 10 months of 2005. The group classifications correspond to top physics, QCD, searches for exotic particles, and bottom quark physics. The physics group affiliation could not be determined for jobs in the "???" category.

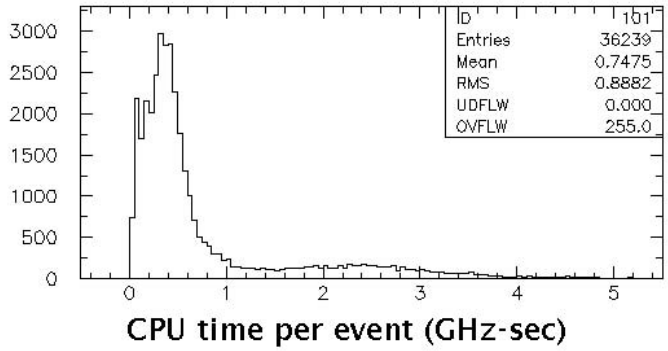

Fig. 5. CPU time per event for randomly selected analysis jobs that read production output files.

4 Grid migration plans

As collaborating institutions invest in large common pools of computing for the LHC, they will become increasingly reluctant to expand pools of computers dedicated to CDF. Indeed, some institutions will require that CDF computers be consolidated into shared, grid-based pools just to maintain access to existing computing capacity. In order to meet the increasing demand for off-site CPU, CDF must therefore adopt the grid-based technologies and infrastructure that will enable utilization of computing in these shared pools. Grid technologies will also allow opportunistic use of more general Open Science Grid[5] (OSG) and LHC Computing Grid[6] (LCG) resources, which could deliver significant computing to CDF until LHC data taking begins.

The basic approach toward achieving interoperability with both OSG and LCG will be to adopt incremental changes, deploying functionality in stages and, if necessary, using partial solutions in the short term until more complete deployments are possible. Throughout this migration, the existing user interfaces will remain unchanged.

Recently, CDF has utilized the Condor "glide-in" [7] technique to gain access to computers within OSG and LCG shared pools. In this scheme, a small job submitted via the grid gatekeeper of the remote pool installs client software for a batch system on one or more worker nodes within the pool. This software contacts a locally maintained CAF headnode and registers the worker node as a worker in an associated virtual CAF. Jobs submitted to the headnode can then be routed to registered workers as if they belonged to a dedicated CAF. Upon termination of a job segment, the worker node can be returned to the common pool. Using a glide-in CAF effectively reduces the remote dedicated resources from a pool of computers to a single headnode.

At present, CDF operates glide-in CAFs at the LCG tier-1 center at CNAF in Italy and at OSG-based farms at Fermilab. Future worker node procurements for CDF at Fermilab will be added to the common OSG pool. Negotiations are now under way to deploy glide-in CAFs at several other sites.

The use of glide-in CAFs will suffice to meet our needs for at least the next year. In the mean time, efforts are under way to re-implement the CAF functionality using native grid tools under both LCG and OSG. When successful, no dedicated computing resources will be required at the remote site beyond local disk, which at the present time is needed in all grid implementations.

5 Event data model

The Event Data Model is the set of structures for raw and reconstructed data, typically stored within some larger shared data structure, and the associated user interfaces and utilities needed to locate, manipulate and store the data. Since the EDM provides the lowest-level contact with the data for all users, a well-designed interface and feature set is essential.

A simple example of an EDM is an ntuple. CDF physics groups maintain several standardized ntuples, which are often created in some organized fashion. The CDF offline framework, however, uses a more complex EDM designed to address certain analysis issues and support specific features of the data from the experiment. A few of the basic features of the EDM are summarized below:

- EDM objects consist of fully-featured C++ objects with the following common attributes: a unique ID number, two independent description strings, a print method, an equivalence operator and a serialization function used to store (and restore) the object within a file.
- Once entered into the event record, data objects can no longer be modified, thereby preserving the processing history of the event.
- The EDM provides a number of standard containers so that arrays of objects or references to objects can be stored in the event record.
- Search utilities allow objects to be located based upon any combination of criteria, including the common attributes or internal data.

Several design features help prevent some of the most common programming errors encountered in earlier generations of the experiment. Making objects in the event read-only, for instance, prevents unrequested and unexpected modification of existing data within the event. Many of these features are disliked by a large fraction of the users, despite the benefits offered from a programming perspective.

5.1 Deficiencies of the EDM and advice for future EDM designers

In the opinion of the author, the EDM at CDF suffers from two significant problems.

1. There is far too much functionality built into data objects. As new C++ developers and novices in object-oriented design, many code authors created data objects that provided a number of complex features. An important example is the track object (of which the present company was a co-author), which includes a topological fitting interface and a rich class hierarchy. While both could in principle be useful, neither are used as intended. The class hierarchy, in fact, is rendered largely inaccessible due to the technical requirements of EDM-based containers. In both cases, however, all the associated code must be carried along even if the only quantities of interest are the track parameters or the number of hits within the tracking chamber.

   Conclusion: All data objects should be very simple structures. Any extra functionality, such as that noted above for tracks, should be placed within auxiliary or helper classes that use the data objects as input. This design pattern adds flexibility and clarity, yet makes it no more difficult to use the more complex features.

2. The EDM is effectively tied to the offline analysis framework due in large measure to the integration of the serialization methods within the low-level objects. Since the reconstruction algorithms are written to use EDM objects, the entire body of primary reconstruction code, along with many useful tools and utilities, are held hostage to that same analysis framework, despite the fact that neither the data nor the reconstruction algorithms have any intrinsic connection to the offline analysis framework.

   Conclusion: The serialization methods should be separated from the EDM. A separation of the serialization methods would allow the EDM to be highly portable. An entirely root-based representation, for instance, could be rather trivially achieved. Then, by taking care to write reconstruction algorithms with respect to the low-level data structures alone, the entire reconstruction framework becomes portable as well. All the tools and utilities that are useful for analysis become liberated for use within an arbitrary ntuple environment, for instance. Such a scheme provides vastly more flexibility while retaining user-level simplicity.

6 Successes

Among the better aspects of the CDF computing model, the author notes the following.

- Development of the CAF and simple submission and monitoring tools, all of which have made using large computing resources"easy".
- Adopted a computer language that supports highly structured data representations.
- Established and maintained a good physical design for offline software.
- Defined many sensibly defined datasets that are produced as output from the production farm. This step has greatly simplified many processing steps.
- Wrote a fast reconstruction program.

7 Summary

CDF has deployed over 5.6 THz of CPU deployed around the world. A simple job submission interface has led to very high utilization of these resources by a large number of collaborators. The C++ based EDM has facilitated the development of structured data representations, although many times at the cost of unnecessary complication stemming from a lack of discipline in low-level design. Finally, the continued need to increase the computing resources available to the experiment will drive the adoption of grid technologies.

The most important issue facing the future development of computing at CDF is the increasing complexity that accompanies growth in the scale of the computing problem. Maintaining robust and scalable data handling systems throughout this growth, for instance, is difficult. The need for other new technologies potentially further complicates this picture. Given these realities, it is imperative that we develop the tools and automation that make it simple for the user to marshal the resources needed to analyze data. Users should be well insulated from the underlying complexities and system details that have little value to them. User interfaces must remain simple and stable. Ideally, the EDM, reconstruction and analysis tools should be context independent. In short, users should be able to focus on the physics rather than the computing problem. Success in this goal will require a determined effort, particularly by those physicists with a computing specialization.

8 Acknowledgments

The author thanks Mircea Coca, Ashutosh Kotwal, Elliot Lipeles, Pasha Murat, Liz Sexton-Kennedy and Igor Sfiligoi for valuable input during the preparation of this report.

References

1. A description of the CDF-II experiment can be found in D. Acosta, *et al.*, Phys. Rev. **D71**, (2005) 032001.
2. T.H. Kim *et al.*, IEEE Trans. Nucl. Sci. **51**, (2004) 892.
3. A. Baranovski *et al.*, FERMILAB-TM-2175 (2002); D. Bonham *et al.*, eConf **C0303241**, (2003) TUAT004.
4. See http://www.dcache.org/.
5. A. Pordes, FERMILAB-CONF-04-466-CD; the home page for the organization is at http://opensciencegrid.org/.
6. A. Pfeiffer *et al.*, Nucl. Instrum. Meth. A **534**, (2004) 106; the home page for the organization is at http://lcg.web.cern.ch/LCG/.
7. Details on Condor glide-in can be found at http://www.cs.wisc.edu/condor/.