CDF/PUB/PRODUCTION/PUBLIC/7777

Data production models for the CDF experiment

S. Hou
Institute of Physics, Academia Sinica, Nankang, Taipei, Taiwan

***Abstract*— The data production farm for the CDF experiment is conducted by a large Linux PC farm designed to meet the needs of data collection at a maximum rate of 20 MByte/sec during the run. We present two data production models that exploits advances in computing and communication technology. The first production farm is a centralized system that has achieved a stable data processing rate of approximately 2 TByte per day. The recently upgrade is migrating to the SAM (Sequential Access to data via Metadata) data handling system. The software and hardware of the CDF production farms has been successful in providing large computing and data throughput capacity to the experiment.**

***Index Terms*— PACS: 07.05-t. Keywords: Computer system; data processing**

I. Introduction

The Collider Detector at Fermilab (CDF) detector is a large general purpose detector for experiment of proton-anti-proton collision at the Fermilab Tevatron Collider. The CDF detector has been upgraded to take advantage of the improvements in the accelerator [1]. Computing systems were also upgraded for processing larger volumes of data collected in Run II since 2000.

Data processing required for CDF is a loosely-coupled parallel processing of "events", where each event is the result of a collision of a proton and an anti-proton. A hardware and software trigger system is used to store and save data of interesting collisions. Each event is processed independently through the offline code without use of information from any other event.

Events of a similar type are collected into files of a data stream. Data is logged in parallel to eight data streams for final storage into a mass storage system. Each file is processed through an event reconstruction program that transforms digitized electronic signals from the CDF sub-detectors into information that can be used for physics analysis. The quantities calculated include particle trajectories and momentum, vertex position, energy deposition, and particle identities.

The production farms are collections of dual CPU PCs running Linux, interconnected with 100 Mbit and gigabit ethernet. The hardware architectures are cost-effective. The challenge in building and operating PC farms is in managing the large flow of data through the computing units.

In this report we describe the hardware integration and software for operation of the CDF production farms. We first describe the requirements and design goals of the system. An early centralized model is described for its design of hardware and software control systems. The performance and experiences with this system is presented. The upgrade using the Fermilab developed "Sequential Access via Metadata" (SAM) system [2] for data handling is discussed. It is portable for distributed computing through network to PC farms. Performance of data production with the SAM production farm are also presented.

II. Requirements

To achieve the physics goals of the CDF experiment at the Fermilab Tevatron, the production computing system is required to process the data collected by the experiment in a timely fashion. In 2001 through 2004 the CDF experiment collects a maximum of 75 events/second at a peak throughput of 20 MByte/sec. The output of event reconstruction is split into many physics data-sets. The splitting operation is required to place similar physics data together on disk or tape files, allowing faster and more efficient physics analysis. The output event size is currently approximately the same as the input. Each event is written 1.2 times on average because some events are written to more than one output data set. Therefore the system output capacity is also required to be approximately 20 MByte/sec.

To accomplish rapid data processing through the farms, adequate capacity in network and CPU is required. The event processing requires 2-5 CPU seconds on a Pentium III 1 GHz PC. The exact number depends on the type of event, the version of the reconstruction code, and the environment of the collision. These numbers lead to requirements of the equivalent of 190-375 Pentium III 1 GHz CPUs, assuming 100% utilization of the CPUs.

In addition to providing sufficient data flow and CPU capacity for processing of data, the production farm operation is required to be easily manageable, fault-tolerant, scalable, with good monitoring and diagnostics. Hardware and software options were explored to meet the requirements for the system. These include large symmetric multiprocessing (SMP) systems, commercial UNIX workstations, alternative network configurations. Prototype systems were built and tested before the final design was chosen and production systems built.

III. Architecture

The first developed CDF data production farm consists of a large number of PCs that run the CPU-intensive codes (workers), PCs that buffer data into and out of the farm (readers and writers) and PCs providing various services (servers). The hardware architecture is shown in Fig. 1. It has two server nodes cdffarm1 and cdffarm2. cdffarm1 is a SGI O2000 machine that host a batch submission system and a database server. cdffarm2 is a dual Pentium server running control daemons for resource management and job submission. Monitoring and control interfaces for farm operation includes a java server to the control daemons and and a web server

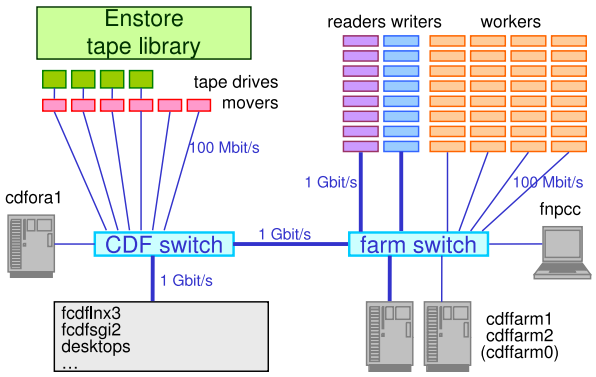

Fig. 1. CDF production farm architecture.

for monitoring. The disk space is a "dfarm" file system [3]. It is a distributed logical file system using a collection of IDE hard-disks of all dual Pentium nodes. The job scheduling on the production farm is controlled by a batch management system called FBSNG developed by the Computing Division at Fermilab [4]. The CDF Data Handling group has well-defined interfaces and operation [5] to provide input data for the farm and to write output to a mass storage system (Enstore) [6].

The dual Pentium nodes were purchased over the years. Old nodes were replaced after three years service. At its peak in mid-2004, there were 192 nodes in service. The dfarm capacity of the collected worker hard-disks was as large as 23 TByte including three file-servers each having 2 TByte. The IDE hard-disk size varies from 40 to 250 GByte.

The input and output (I/O) nodes are configured to match the data through-put rate. A total of 16 nodes equipped with optical giga-links are configured with the pnfs file system [7] for access to the Enstore storage. A 48 port Cisco switch module was added recently to provide gigabit Ethernet over copper switching. Additional I/O nodes may be added if needed. The number of workers can be scaled to as large a number as is required. However, the total data through-put capacity to Enstore storage is limited by the number of Enstore movers (tape-drives) available.

Raw data from the experiment is first written to tape in the Enstore mass storage system. Raw data are streamed into eight data-sets listed in Table I. These tapes are cataloged in the CDF Data File Catalog (DFC) [8] as a set of tables in an Oracle database (accessed via cdfora1 in Fig. 1). After the data is written to tape and properly cataloged, and once the necessary calibration constants exist, the data is available for reconstruction on the farms.

Stream	data-sets	events/GByte	total event (%)	total size (%)
A	aphysr	2720	3.8	7.7
B	bphysr	5470	9.9	5.5
C	cphysr	6770	9.2	7.5
D	dphysr	2570	3.7	7.9
E	ephysr	5930	17.0	15.7
G	gphysr	6140	26.4	23.5
H	hphysr	6050	19.6	17.7
J	jphysr	5520	10.3	10.3

TABLE I

STATISTICS OF DATA STREAMS OF A TYPICAL RUN TAKEN IN JUNE 2004 CONTAINING ALL SUB-DETECTORS. THE RAW DATA FILES ARE 1 GBYTE IN SIZE. LISTED ARE THE NUMBER OF EVENTS PER GBYTE, RATIO OF TOTAL EVENTS AND TOTAL FILE SIZE.

IV. FARM PROCESSING SYSTEM

The production farm is logically a long pipeline with the constraint that files must be handled in order. The input is fetched directly from Enstore tapes and the outputs are written to output tapes. The data flow is illustrated in Fig. 2 for the files moving through dfarm storage controlled by four production daemons. The daemons communicate with the resource manager daemon and the internal database to schedule job submission. The internal database is a MySQL [9] system used for task control, file-tracking, and process and file history. The DFC records are fetched at the beginning of staging input data. Output files written to tapes are recorded in the DFC. Job log files and other logs and files are collected to the user accessible fcdflnx3 node. Operation status is monitored by a web server fnpcc.

The operation daemons are configured specifically for production of a input "data-set". For raw data, each data stream is a data-set. The input files are sent to worker nodes for reconstruction. Each worker node (dual-CPU) is configured to run two reconstruction jobs independently. An input file is approximately 1 GByte in size and is expected to run for about 5 hours on a Pentium III 1 GHz machine. The output is split into multiple files, with each file corresponding to a data-set defined by the event type in the trigger system. An event may satisfy several trigger patterns and is consequently written to multiple data-sets that are consistent with that event's triggers. Each data-set is a self-contained sample for physics analysis. The total number of output data-sets is 43 for the eight data streams used in the most recent trigger table.

The Farm Processing System (FPS) is the software that manages, controls and monitors the production farm. It is flexible and allows configuration for production of data-sets operated independently in parallel farmlets. A farmlet contains a subset of the farm resources specified for the input data-set, the executable and the output configuration for concatenation. Its execution is handled by its own daemons taking care

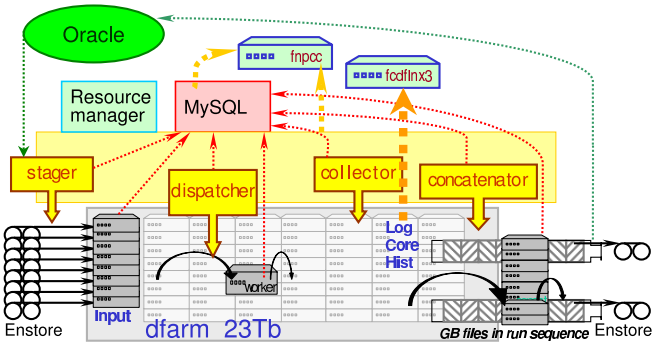

Fig. 2. Flow control in the CDF production farm. Congestion is observed in concatenation waiting for lost files and in part caused by slow MySQL service.

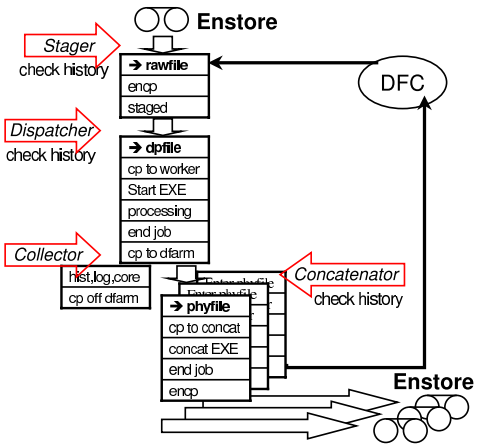

Fig. 3. Task control for a farmlet. Status is recorded for each input file in MySQL database.

of consecutive processing in production and its records are written in the internal database. The task control by FPS for a farmlet is illustrated in Fig. 2 and Fig. 3. The daemons of the farmlets are :

- **Stager** is a daemon that is responsible for finding and delivering data from tapes based on user selection for a set of data files or run range in the data-set. Jobs are typically submitted one "file-set" at a time. A file-set is a collection of files with a typical size of 10 GByte. The stager fetches DFC records for input and checks that proper calibration constants are available. The staging jobs are submitted to the input I/O nodes and the file-sets are copied to their scratch area, and afterward to dfarm.
- **Dispatcher** submits jobs through the batch manager to the worker nodes and controls their execution. It looks for the staged input file, which is then copied into the worker scratch area. The binary tarball (an archive of files created with the Unix tar utility) containing the executable, complete libraries, and control parameter files are also copied. This allows the reconstruction program to run locally on the worker nodes and the output files, of various sizes from 5 MByte to 1 GByte, are written locally. At the end of the job the output files are then copied back to dfarm. In case of abnormal system failure, job recovery is performed and the job is resubmitted.
- **Collector** gathers any histogram files, log files and any additional relevant files to a place where members of the collaboration can easily access them for the need of validation or monitoring purposes.
- **Concatenator** writes the output data that is produced to the selected device (typically the Enstore tape) in a timely organized fashion. It checks the internal database records for a list of files to be concatenated into larger files with a target file size of 1 GByte. It performs a similar task as the dispatcher, with concatenation jobs submitted to output nodes. The output nodes collect files corresponding to a file-set size ($\approx$ 10 GByte) from dfarm to the local scratch area, execute a merging program to read events in the input files in increasing order of run and section numbers. It has a single output truncated into 1 GByte files. These files are directly copied to tapes and DFC records are written.

Since all of the farmlets share the same sets of computers and data storage of the farm, the resource management is a vital function of FPS for distribution and prioritization of CPU and dfarm space among the farmlets. The additional daemons are:

- **Resource manager** controls and grants allocations for network transfers, disk allocations, CPU and tape access based on a sharing algorithm that grants resources to each individual farmlet and shares resources based on priorities. This management of resources is needed in order to prevent congestion either on the network or on the computers themselves and to use certain resources more effectively.
- **Dfarm inventory manager** controls usage of the distributed disk cache on the worker nodes that serves as a front-end cache between the tape pool and the Farm.
- **Fstatus** is a daemon that checks periodically whether all of the services that are needed for the proper functioning of the CDF production farm are available and to check the status of each computer in the farm. Errors are recognized by this daemon and are reported either to the internal database which can be viewed on the web or through the user interfaces in real time. Errors can also be sent directly to a pager with a copy to an e-mail address that is registered as the primary recipient of these messages.

The system control framework of FPS is primarily coded in python [10]. It runs on one of the server computers (cdffarm2) and depends on the kernel services provided by cdffarm1, namely the FBSNG batch system, the FIPC (Farm Interprocess communication) between the daemons and dfarm server governing available disk space on the worker nodes. Daemons have many interfacing components that allow them to communicate with the other needed parts of the offline architecture of the CDF experiment. Those include mainly the DFC and the Calibration Database.

The FPS status in data production is shown in real time on a web page that gives the status of data processing, flow of data, and other useful information about the farm and data processing. The web page is hosted on a dual Pentium node (fnpcc on Fig. 1) connected to the farm switch. The web interface was coded in the PHP language [11] and RRDtool [12] for efficient storage and display of time series plots. The structural elements in the schema include output from each FPS modules, a parser layer that transforms data into a format suitable for RRDtool, a RRDtool cache that stores this data in a compact way, and finally the web access to RRD files and queries from MySQL for real time display of file-tracking information.

The java control interface was designed for platform independent access to production farm control using an internet browser. Information transfer between the client and server over the network is done using IIOP (Internet Inter-ORB protocol) which is part of CORBA[13]. It has proved to be stable, and there have been no problems with short term disconnections and re-connections. An XML processor[14] is

used to generate and interpret the internal representation of data. Abstract internal representation of data is important to cope with changes in the FPS system. A Java programming language, Java Web Start technology [15] was used for implementation of a platform independent client.

V. Bookkeeping

With hundreds of files being processed at the same time it is important to track the status of each file in the farm. File-tracking is an important task of FPS and the bookkeeping is based on a MySQL database. The database stores information about each individual file, process and the history of earlier processing. Three tables are implemented for each farmlet: for stage-in of input files; reconstruction and output files; and the concatenation. The processing steps tracked by the bookkeeping and records in each table are illustrated in Fig. 3. Once a file is successfully processed, its records are copied over to the corresponding history tables. The file status is used in order to control the flow of data and to make sure that files are not skipped or processed more than once. The MySQL database also includes detailed information about the status of each file at every point as it passes through the system. This information is available through a web interface to the collaboration in real time. This database server was designed to serve thousands of simultaneous connections.

With the help of information that is stored in the internal database, the system is able in most cases to recover and return to the previously known state from which it can safely continue to operate. The daemons checking the file history in the database are not instrumented to detect an abnormal failure for a job in process or a file lost to network or hardware problems. The concatenator often has to wait for output file in order to combine files in order. This bottleneck can be a serious problem and is a major consideration for relaxing strict ordering of files to improve overall system performance.

VI. Data processing capacity

A major reprocessing of all CDF data (with code version 5.3.1) was launched in March 2004 and the production farm operated at full capacity for a six week period. The main characteristics and performance of the farm is described for the production capacity. The CPU speed and data through-put rate are the factors that determine the data reconstruction capacity of the production farm. The computing time required for an event depends on the event characteristics determined by the event trigger in different data streams. In addition, the intensity of the proton and antiproton beams matters. More intense beams lead to multiple events per beam crossing which in turn lead to more CPU time per event. Inefficiency in utilizing CPU comes from the file transfer of the executable and data files to and from the worker scratch area.

The event size increases with beam intensity from 140 to 180 kByte. The CPU time per event, in reconstruction of cdf software version 5.3.1 time on a dual Pentium III 1 GHz machine, is around 2 seconds per event and increase for beam intensity and event size. The input data files are staged from Enstore tapes. The rate of staging data depends on how fast the link to Enstore movers is established. Once a mover is allocated, staging a file-set of 10 GByte takes about 20 minutes. The data transmission rate varies file by file, the commonly observed rate is around 10 MByte/sec.

Output of concatenated files are copied to tapes. The effectiveness in staging data to a tape is a concern because of the limited dfarm space and output bandwidth. A concatenation job on the output node collects files of a data-set with close to 10 GByte at a speed that may reach the maximum IDE disk transfer speed of 40 MByte/sec. It takes an average 10 minutes to copy all the files requested. The concatenation program reads the numerous small files and writes output that is split into into 1 GByte files. On a Pentium 2.6 GHz node the CPU time is about 24 minutes for processing 10 GByte. The job continues by copying the output to Enstore at an average rate of close to 20 MByte/sec. It takes about 10 minutes for writing 10 GByte. Further delay may be caused by having more than one job accessing the same hard disk in dfarm, or waiting to write to the same physical tape.

The tape writing is limited to one mover per data-set at a time, to ensure that files are written sequentially on tape. A tape is restricted to files of the same data-set. The instantaneous tape writing rate is 30 MByte/sec. However, the average rate drops to below 20 MByte/sec because of latency in establishing connection to the mass storage system (this includes mounting and positioning the tape establishing the end-to-end communication). Running only one data-set on the farm limits the capability of the farm. Running a mix of jobs from different data-sets in parallel increases the through-put of the farm by increasing the output data rate.

To maximize the farm efficiency the data reprocessing was performed on five farmlets with each farmlet processing one data-set. The tapes were loaded one data-set at a time, therefore farm CPU usage came in waves shared by a couple data-sets at a time. The CPU usage for the week of March 18 is shown in Fig. 4. A lag in CPU utilization was observed when the farm switched to a new data-set, seen as the dips

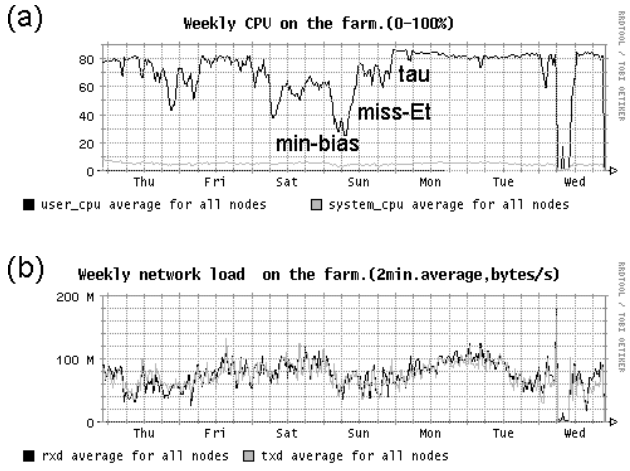

Fig. 4. (a) CPU load and (b) dfarm traffic of the week of March 18-25, 2004.

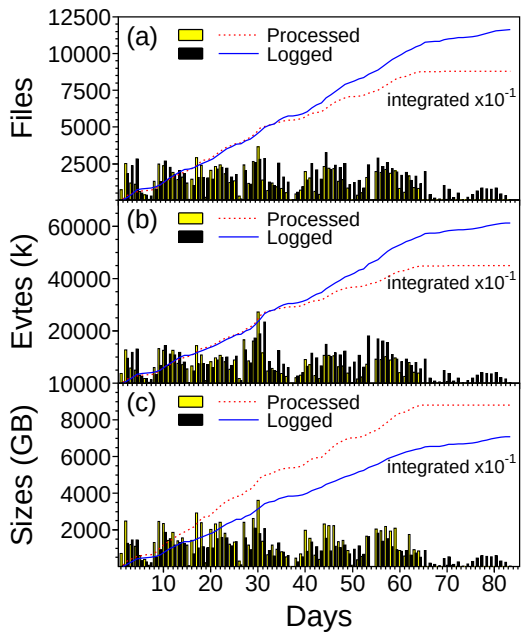

Fig. 5. Daily processing rates are shown in histograms for (a) number of files, (b) number of events, and (c) data size. The integrated rates are shown in lines. Compressed outputs were created for selected data-sets (about a quarter of the total). Event size is reduced by about 30% and thus a net reduction in output storage.

in CPU in Fig. 4.a, because of lack of input files. File-sets are distributed almost in sequence on a tape The lag at the beginning of staging in a data-set is because the files requested are stored on the same tape, causing all the stage-in jobs to wait for one tape. Overall the stage-in is effective in feeding data files to dfarm. The CPU usage varies for data-sets. The "minimum bias" data-set has smaller file sizes and the CPU per event is about 40% less than the average. When this data-set was processed, the stage-in rate was not able to keep up with the CPU consumption.

The output data logging rate is shown in Fig. 5 for the number of files, number of events, and total file size written to Enstore tapes. Compressed outputs were also created for selected data-sets. Therefore the total events in output was increased by about 25%. The event size was reduced and resulted to a net reduction in storage by about 20%. On average we had a through-put of over 2 TByte (10 million events) per day to the Enstore storage. The data logging lasted two extra weeks for a large B physics data-set that accounted for about 20% of the total CDF data. It was the latest data-set processed and the tape logging rate was saturated at about 800 GByte per day.

VII. SAM PRODUCTION FARM

Upgrade of the production farm is required for the increasing demand in computing capacity. Also the FPS system, for having many unique configurations, has become more difficult to be compatible with newly developed computing facilities. Among the recent development, the CDF Analysis Farms (CAF) [16] is deployed in many CDF collaboration institutes all over the world. The CAF is a Linux PC farm with access to the CDF data handling system and databases running batch analysis jobs. The software interface provides job submission to batch systems like FBS and Condor [17] in a uniform manner.

The CDF data management is migrated to the SAM data handling system. SAM is organized around a set of servers communicating via CORBA to store and retrieve files and associated metadata. File information is stored in the SAM database as file metadata. A task for processing many files is launched as a SAM project. A project is organized for a user dataset, with a consumer process established to receive data files. File delivery is coordinated such that the events are read only once to all the analyses programs of the project.

Illustrated in Fig. 6 is the hardware architecture and applications for data production with SAM. With data handling taken care by SAM, the disk storage required is a durable cache for program output to be concatenated before being stored to SAM. The communication with SAM database is conducted by the farm servers configured as SAM stations. CPU and durable storage are modular entities easily specified in the job submission. The CAF is specified in the job submission, therefore it is flexible to be any facility available on the network. To optimize bandwidth and file usage, the SAM production farm is mounted to the dCache [18] file system where input files are delivered to. Concatenated output files are transferred directly to Enstore.

Job submission is controlled by applications scheduled on a SAM station. The usage of file metadata is generalized for bookkeeping purpose. The tasks in preparing input datasets and data processing in a CAF worker node are illustrated in Fig. 7. The tasks are:

- **Prepare input datasets :**
  The input data to be processed are selected with queries made to online DFC records for data quality (good-run) and detector calibration. A run is a data taking period

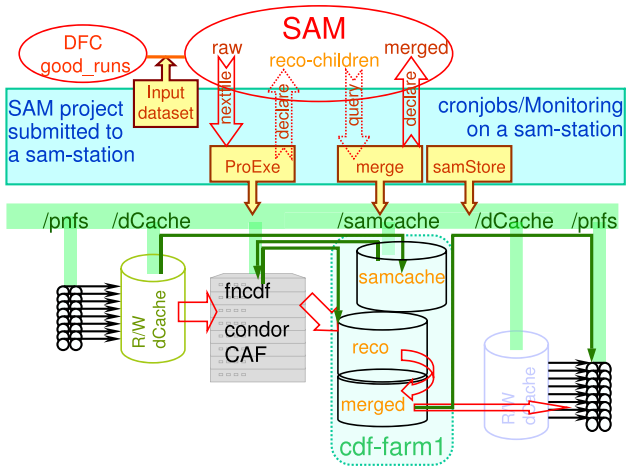

Fig. 6. Data flow and control of production on a SAM farm. Data are transported by SAM to a file cache accessible to the Condor CAF facility. Output is sent to a durable storage where concatenation is operated. Merged outputs are declared to SAM and stored to Enstore.

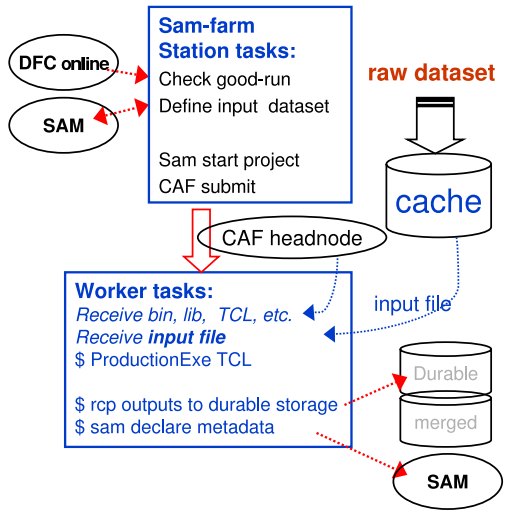

Fig. 7. Task flow for a SAM project submitted to CAF workers. A worker node receives the executable tarball, copies input data file, after processing outputs are copied to durable storage with metadata registered to SAM.

of continuous proton-anti-proton collisions specified by a run number. The input datasets are organized in run sequence of more than 20 files of one or multiple runs of a raw data stream.

- **Start SAM project, and CAF submission :** A SAM project is started for a dataset not yet fully consumed. It is submitted to a CAF as a batch job. SAM establishes a consumer process to deliver files to CAF workers. The CAF workers receive an archived (tar) file containing program binary, library and control TCL cards. Data file location is given by SAM. It is copied to the local scratch area. Files are delivered according to the file consumption status, till all files are delivered. Output of the program are then copied to dedicated durable storage area, and the associated metadata are declared to SAM.

The dataset preparation and job submission are all issued periodically by cron jobs. To prevent exhausting the computing recourses, permission is required by a resource template recording the latest status of them. A monitoring graph on the consumption of data files are plotted in Fig. 8

In comparison with the FPS system, the SAM farm management is attending operation in dataset movement. Tracking on individual file is taken care by the SAM consumer process. The operation is therefore reduced to detect incomplete projects and debug. The bookkeeping tasks is reduced from tracking thousands of files in an instance to a few dozens of projects. The monitoring is concentrated on the usage of durable storage, where outputs from CAF are checked and merged in the concatenation process.

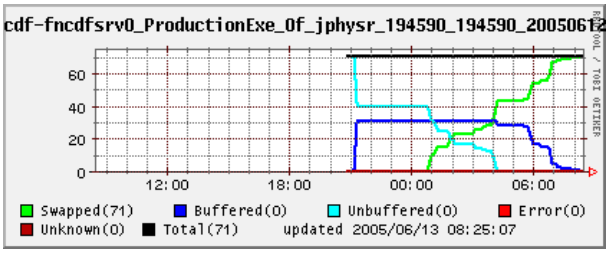

Fig. 8. Consumption of files by a SAM project is plotted. Tho total of 71 files in a dataset is requested. Files are quickly ”buffered” to CAF workers. The CAF job is configured to use 30 CPU segments. After approximately 4 hours, consumed files are being ”swapped”. The project is terminated after all files are swapped.

VIII. Durable storage

Output of CAF jobs are buffered in durable storage provided by 2 TByte file servers. With the total number of files above a threshold (for example, 20 GByte), concatenation tasks are launch merging small files into output of size close to 1 GByte. Previously, the output of concatenation in the FPS system is truncated to 1 GByte exact, therefore an input file can be split into two concatenated files. This is changed in the new algorithm. We have relaxed the contraint on output file size such that the output is a merged file of a complete set of files. This algorithm simplifies bookkeeping for unique parentage recorded in metadata. The details of concatenation on the durable storage node are illustrated in Fig. 9, and are described in the following:

- **Durable cache :** the durable cache is a directory on a large file server where CAF output of the same dataset are stored. In total 41 directories are used for all reconstructed datasets. The files are buffer till a threshold (for example 100 files). A cron job sort them into lists of files in run sequence. The size of each list is within the desired concatenation file size. And the control TCL is prepared to include these files for inputs to the concatenation binary (AC++Dump).
- **Concatenation :** Concatenation is running on the file server with the output is stored in the ”merged” directory ready to be stored to SAM.
- **SAM store :** The merged files are scanned periodically for a total over a threshold (for example 10 GByte) and the SAM store is conducted to copy files to Enstore and declare metadata. The threshold size is used to reduce Enstore operation cycle to improve bandwidth.

The concatenation job is mostly just moving blocks on disks, therefore we choose to do it locally on the file server to prevent network hurdle and job management. The CPU time

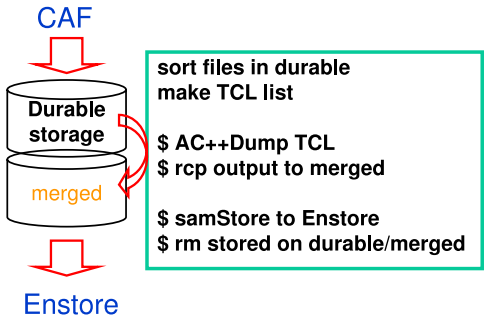

Fig. 9. Files in a durable cache are sorted into lists in TCL cards read by the concatenation binary (AC++Dump). The merged files (of size close to 1 GB) are stored to SAM.

is roughly 3 minutes per GByte on a P3 2.6 GHz file server for 7200 rpm IDE hard drives. This is the slowest process operated on the durable cache. The network giga-link ethernet speed is commonly running at 20 MByte/sec and the Enstore logging rate by a single mover can accomplish 1 TByte tape writing a day.

The new system is more tolerable to errors. We impose parentage in metadata listing input raw data parents and output children. With a SAM query we find files not yet produced. If a merged output file should be reprocessed, we look for its parents and submit a new project for them.

IX. Scalability

The FPS system uses dfarm file system which is the collection of IDE hard disks on workers. With a total 200 workers, the chance of losing file increases whenever an worker is not accessible. Meanwhile the MySQL database requires fast CPU speed for thousands of query in an instance. The architecture of the FPS system is constraint to the data handling with direct access to Enstore. This, however, also prevents usage other than the dedicated production operation.

The SAM production farm exploits the advantage of data handling system provided, meanwhile uses file metadata for bookkeeping. The overhead is the durable cache management for concatenation which is specific for CDF data production. The software contains applications of SAM clients, therefore the constrains on hardware facilities are minimized. The CPU facility can be any of CAF facility of the CDF collaboration. The durable storage file servers can also be located anywhere accessed by the CDF data handling system. Production tasks are operated on a SAM station communicating with the SAM with file metadata tailored for bookkeeping and monitoring purpose.

The prototype SAM production farm was tested with SAM station and file servers located at Fermilab and jobs sent to CAF facilities in Japan and Taiwan. We were able to accomplish a few MByte/sec bandwidth in operation. The dedicated SAM production farm is constructed in the spring 2005 at Fermilab. It has gigabit network links for a CAF facility of 70 workers and four file servers each having 2 TB durable storage space. The data input is configured for direct copy from dCache read pool and SAM store to Enstore. Each file server can manage about half TByte throughput running two concatenation jobs. This system has accomplished a stable operation for production of CDF data collected in 2005.

X. Conclusion

The CDF production farms have been successfully prototyped and commissioned. They have provided the computing capacity required for the CDF experiment in Run II. The system has been modified and enhanced during the years of its operation to adjust to new requirements and to enable new capabilities. The production facility is recently upgraded to adapt to the SAM data handling system. It is migrated from a customized central computing model to to portable for operation on distributed computing facilities. The system will continue to be modified in the future to continue on serving the CDF collaboration as required. These developments will allow CDF to continue to process and analyze data through the end of the life of the experiment.

References

[1] R. Blair, *et al.*, The CDF-II Detector: Technical Design Report, Fermilab-Pub-96/390-E, Nov, 1996.

[2] The SAMGrid Project, Fermi National Accelerator Laboratory; *http://projects.fnal.gov/samgrid/*.

[3] I. Mandrichenko et al., "Disk Farm Installation and Administration Guide ", v1.6, Nov 11, 2001; *http://www-isd.fnal.gov/dfarm/*.

[4] Farms and Clustered Systems Group, Fermi National Accelerator Laboratory, "Farm Batch System (FBS) Users Guide" v 1.5, Aug 25, 2003; *http://www-isd.fnal.gov/fbsng/*.

[5] R. Colombo, et al., "The CDF Computing and Analysis System: First Experience", FERMILAB-Conf-01/300-E, November, 2001

[6] Computing Devision, Fermi National Accelerator Laboratory, "Enstore mass storage system"; http://www-isd.fnal.gov/enstore/.

[7] Information Technology Group, Deutsches Elektronen-Synchrotron DESY, "The Perfectly Normal File System"; http://www-pnfs.desy.de/.

[8] P. Calafiura et al., "The CDF Run II Data Catalog and Access Modules", "CHEP 2000, Computing in high energy and nuclear physics", p494.

[9] MySQL AB, Uppsala, Sweden; *http://www.mysql.com/*

[10] Python Software Foundation; http://www.python.org.

[11] The PHP Group; http://www.php.net.

[12] T. Oetiker, "RRDtool"; http://people.ee.ethz.ch/~oetiker/webtool/rrdtool/.

[13] Object Management Group, Inc., Needham, MA 02494, USA; http://www.corba.org/.

[14] World Wide Web consortium, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; http://www.w3.org/XML.

[15] Sun Microsystems, Inc., Santa Clara, CA 95054, USA; http://java.sun.com/products/javawebstart/.

[16] M. Casarsa et al., CDF CAF User's Manual, CDF/DOC/COMP_UPG/PUBLIC/6092, June24, 2005; http://cdfcaf.fnal.gov/.

[17] Condor Manual, Condor Team, 2003; http://www.cs.wisc.edu/condor/manual/.

[18] dCache, a distributed storage management data caching system; http://dcache.desi.de/.