xxxx
Draft v.0.4
October 31, 2003

Design, Test and Operation of a large Linux PC Farm for CPU-intensive computing for the CDF experiment at Fermilab

Jaroslav Antos, Marian Babik, Antonio Wong Chan, Yen-Chu Chen, Paoti Chang, Roman Lysak, Miroslav Siket, Stephen Wolbers, G.P. Yeh, Ping Yeh

Abstract

The CDF offline processing system was designed and constructed to meet the needs of the CDF experiment for processing data from Run 2 at the Tevatron Collider. The system is composed of a large cluster of PCs with a high-speed network interconnect and a sophisticated software system to control the flow of data and the scheduling of tasks on this PC farm. This approach was natural, given the explosion in the use of the internet and PCs. A robust, conservative approach was used in the design and integration of the farm. Nevertheless, the farm explores and exploits advances in computing and communication technology. The system has been successful in providing large computing resources to the experiment.

1 Introduction

High-Energy Physics (HEP) has advanced over the years by the use of higher energy and higher intensity particle beams, more capable detectors and more computing to analyze larger and larger volumes of data. At the Fermilab Tevatron, recent advances in the accelerator have increased the intensity and energy of the proton and anti-proton beams (Run 2)[1]. The CDF detector has been upgraded to take advantage of the improvements in the accelerator[2]. Part of the improvement is an upgrade of the computing systems that process the larger volumes of data collected in Run 2.

The type of computing required for initial data processing can be characterized as very loosely-coupled parallel processing[3]. The data consists of a group 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 only the most interesting collisions. An event consists of approximately 250 Kbyte of data and each event is independent. Events of a similar type are collected into files of approximately 4000 events (1 Gbyte). Each file is processed through a software program (ProductionExe). This program is the full CDF reconstruction program that is used to transform digitized electronic signals from the CDF subdetectors into information that can be directly used for physics analysis. The quantities calculated include particle trajectories and momentum, vertex position, energy deposition, and particle identities.

A computer "farm" of many cheap processors is an ideal computing system for processing this type of data. Fermilab has a long history of using inexpensive processors to provide large computing power. The first large systems used dedicated processors (Motorola 68030) [4]; later, large arrays of commercial UNIX workstations were used [5]; and recently commodity PC's have replaced the UNIX workstations [6]. A challenge in realizing a successful system is to build and operate a system that can manage the large flow of data in addition to providing the computing power. Such a system has been designed, prototyped, built, integrated and operated for the CDF experiment for Run 2 at Fermilab.

This paper will describe the software, hardware, integration and operation of the CDF production computer farm. The first section will describe the requirements and design goals of the system. Next, the design of the farm, hardware and software, will be given. The software system will be described in the next section. Next, the performance and experiences with the system, including prototypes, will be described. Finally, a discussion of future possibilities for the farm for higher rates and future runs of the experiment will be explored.

2 Requirements and Design Goals

2.1 CPU and Input/Output(I/O)

To achieve the physics goals of the CDF experiment at the Fermilab Tevatron, the production computing system has to be able to process the data collected by the experiment in a timely fashion. In addition, some or all of the data must be processed multiple times. This is necessary because newer software and calibrations are developed as the data and detector are studied and become better understood. Reprocessing the data allows the maximum physics output from detector and data to be achieved. In addition, event simulation can easily be run on this same set of computers.

The requirement of processing data through the CDF off-line reconstruction program as quickly as possible through the computing farms leads to requirements on the total amount of data that has to be handled by the farm I/O systems, and the amount of CPU required by the farms. The CDF experiment expects to collect a maximum of 75 events/second, with each event requiring 2.5-5 CPU seconds on a PIII/1 GHz PC to process. The exact number depends on the type of event, the version of the reconstruction code, and the environment of the collision. The CPU time per event is not a constant. Two factors lead to changes in the CPU required to process events. One is the intensity of the beams during collisions. More intense beams lead to multiple events per beam crossing which in turn lead to more CPU time per event. The second factor is the version of CDF code. Therefore, the numbers quoted here apply to a range of versions of the code and to a range of intensity of the accelerator. Events average 250 Kbyte in size. These numbers leads to requirements of 18.75 Mbyte/seconds input data rate and the equivalent of 190-375 PIII/1 GHz CPUs, assuming 100% utilization of the CPUs. A somewhat more general statement is that primary processing leads to a requirement of 7,500-15,000 SpecInt95 or 190-375 GHz CPU power.

The output event size is planned to be 60-100 Kbyte/event and each event will be written 1.2 times on average because some events will be written to more than one output data set. The output rate is therefore 5.4-9.0

Mbyte/sec. In early running of the experiment the output size was closer to 250 Kbytes/event because extra information was kept for debugging purposes. Therefore the system needs to have an output capacity of approximatelly 20 Mbytes/sec.

Re-processing of data adds to both the I/O and to the CPU requirements of the farms, unless it is interleaved with the initial processing of data. Interleaving is possible when the experiment is not collecting data at full rate, e.g., when the accelerator is not running. The estimated size of re-processing, based on run 1 experience, is approximately 1/3 of the initial processing. Taking into account re-processing the required computing power will be 10,000-20,000 SpecInt95 or 250-500 GHz. Reprocessing requirements are expected to grow as the experiment collects data and requires reprocessing of ever-larger datasets. Therefore, the farm CPU requirements will increase with time.

2.2 Design Goals

In addition to providing sufficient data flow and CPU capacity for processing of data, the CDF production farms system must be relatively easy to manage, fault-tolerant, scalable, with good monitoring and diagnostics. Specifically, the goals can be listed as follows:

- Processing must be straightforward to establish and to maintain. In particular this must be true for those situations where processing needs are fairly complicated, with changes to the type of input data or other modifications of the system. This includes tests of new code and running multiple versions of code simultaneously.
- The farm should be stable but also able to easily recover, preferably automatically, to various common errors.
- The farm should scale fairly well to a larger size, a mix of different processors, or a change of I/O (type of machine or network).
- The farm should be easy to monitor.

These design goals have had a large impact on the system that was prototyped and the final production system that was built and operated for Run 2.

3 Design of Farms, H/W and S/W

As was stated in the previous section a computer system that initially satisfies the needs for the off-line data processing of the CDF experiment in Run 2 should meet three requirements: computing power on the order of 250-500 GHz, a transfer rate of input data of 20 Mbyte/s (in multiple streams) and an output data rate of 20 Mbyte/s (in multiple data sets). Off-line data processing consists of processing quasi-independent (or loosely coupled) units called "events". Each event records the response of the CDF detector systems to a given interaction triggered by a set of conditions. It follows that events can be processed independently and at the same time (in parallel) on different processors. It was shown [6] that the required computing power can be built from personal computers, i.e. a PC Farm. Using PC's as basic units for a compute farm has the following advantages:

- Low price. One cannot beat the price for commodity components (PC's).
- The possibility to easily increase the power of a PC Farm either by replacing less powerful units by faster PC's or by increasing the number of PC's, or both.

A very important requirement is the stability of performance of the farm over a period of many years. Therefore the choice of hardware and software should be made with this condition in mind. In the case of the hardware it means a careful choice of components of PC's with a "burn-in" test (which takes about one month at Fermilab) before a set of PC's is included in the PC Farm.

For software the choice of operating system is very important. Microsoft Windows and Linux have been considered. After some discussions and several tests Red Hat Linux was chosen. The programming environment of Linux is much better suited to HEP applications, given the long history of UNIX use in the field.

After making a choice of PC's for the farm and an operating system, a choice of the number of processors on each PC must be made. The Linux operating system supports in excess of 32 processors on a single PC in symmetric multiprocessing mode (SMP). We chose the most cost-effective option

– two CPU's per node. This choice is strongly favored by price to performance considerations (at least at the time of the design). But one should not underestimate the robustness and flexibility of design in this case. New node(s) can be added or released from the system without much impact on the system (just directly proportional to amount of added or released nodes).

For job scheduling on the PC farm there do exist sophisticated batch manager system for computer farm control (e.g. MOSIX, LSF, CONDOR, gNQS, PBS etc.). Some of them are commercial and their price depends on the number of processors in the PC Farm. The Computing Division at Fermilab has experience in the development of a job manager system for computer farm control. Their product CPS [8] was used for many years on the farms at Fermilab. The developers of a new manager system called FBSNG [9] built their product based on previous experience but they also very carefully listened to recommendations and requests of users [10]. So it is no surprise that this system is a success! The batch system was optimized for the type of processing that was planned for the PC Farm.

A decision was taken early to split some responsibilities for the CDF Farm. The CDF Data Handling group took responsibility for the software, hardware and operation of the I/O [7] for the farms. Well-defined interfaces were used to provide the input data for farms and to write output to mass storage. This arrangement changed when the CDF data handling system was modified to use the Enstore tape handling system[11]. The interface remained essentially the same but the tape I/O is handled by a Computing Division supported package (Enstore).

The structure of the initial hardware design is shown in Figure 2 The CDF farms group developed a dedicated specialized system for CDF off-line processing. The kernel of the system should be well designed to accept various changes and requests from the collaboration for the processing of data. The development of tools on the Internet made it relatively easy to define two capabilities for the system:

- monitoring of the offline processing on the web
- control of the PC Farm, in a platform independent way with very low requirements for Internet bandwidth connection

As was stated above the PC Farm for CDF offline processing will run for several years. It is expected that there will be someone on shift to check the performance of the processing and to act in case of problems. The CDF collaboration has members in many countries. Making use of the collaborators in many time zones could help to increase the efficiency of running the farms and the design will hopefully promote that.

4 CDF Farm Processing System (FPS)

FPS has been designed in a general way that allows the CDF Farm to manage a variety of tasks such as the processing of raw data, monte carlo event generation, beam-line processing, or re-processing simultanously and seemingly independent of each other. In order to accommodate this design the concept of farmlet was created. A farmlet is a logical unit, a subset of the Farm. It is defined by the input data information, an executable and the location of the output. Since a farmlet is an independent entity, it is treated as such, i.e., its execution is handled by its daemons that try to provide smooth operation. Since all of the farmlets run on the same sets of computers inside of the farm and they use the same network and data storage (either disks or tapes), a vital part of the FPS deals with distribution and prioritization of these resources among the farmlets.

A very important part of FPS is the internal database. It stores information of each individual file, process or history of the previous processing. That information is needed in order to control the flow of data and to make sure that files are not skipped or processed more than once. It also includes detailed information about the status of each file at every point as it passes through the system. This information is available through the web interface to the collaboration in real time.

FPS has been written in python[13], which is highly modular interpreted language with object oriented design. It allows fast development and is very convenient with its broad set of standard modules. For the internal database MySQL[14]was chosen for its ease of setup and reliability. MySQL cooperates very nicely with python as well as with PHP[16], that was chosen as the basis for web development.

Emphasis is put on automatic error recovery and minimal human interaction in the course of processing. 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 well known state from which it can safely continue to operate.

4.1 Kernel of the system

The system consists of several daemons that take care of different parts of the processing chain for each farmlet:

- **Stager** is a daemon that is responsible for finding and delivering data based on user input. The user input is usually a logical description of a set of data or files. Its task is also to determine whether the data are suitable for processing, e.g. are the proper calibration constants available.
- **Dispatcher** submits jobs through the batch manager to the worker nodes and controls their execution. Should any worker node stop functioning, it resubmits the jobs to another worker node.
- **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 (either tape or disk) in a timely organized fashion.

Besides these daemons that govern the flow of the data of each farmlet, there are additional daemons:

- **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 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.

Outside of the FPS framework there are other services and components needed for the proper functioning of the system, namely FBSNG as the batch system, FIPC (Farm Interprocess communication) as the storage of the persistent commmunication between the daemons and Dfarm (Disk Farm) as a distributed cache system that governs 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 (Data File Catalog) and the Calibration Database.

Data from the experiment is stored in the Enstore mass storage system and FPS reads input data and writes output data to it.

4.2 Bookkeeping

With hundreds of files being processed at the same time it is important to track the status of each file in an internal bookkeeping system. The bookkeeper is based on a MySQL database. This database server was designed to serve thousands of customers on the web. For our application it is a perfect match.

In the CDF Farms the MySQL server is running on an SGI SMP machine. All the requests from daemons are made remotely. Figure **??** shows the activity of the MySQL database while the farms are running. The x axis marks the time in units of minutes. The y axis is the counts per ten minutes. Each count means one message logging into the database. The spikes correspond to load tests where hundreds of requests were made in a few minutes. MySQL is able to accept and record the information without any difficulty. Normal farm activity requires significantly fewer requests in any period of time because all of the requests from the Reconstructor and other daemons are spread out in time. The system easily has the capcaity to handle this load.

4.3 Web interface

The natural place to make crucial information accessible to all of the CDF collaboration (in fact to all the world) is a web page. By using a web interface (a complete description can be found in[15]) to the FPS system the collaboration is informed in real time the status of data processing, flow of data, and other useful information about the farm and data processing (including an event display of interesting processed data in real time!).

From a design point of view the web interface for FPS is shown schematically in Figure **??**. Crucial tools in the creation of the web interface were the PHP language (for efficient design of web page) and RRDtool [17] for efficient storage and display of time series plots. The structural elements in the schema have the following meaning: *Module output* represents output from FPS modules; *Parser layer* transforms data into a format suitable for RRDtool; *RRDtool cache* stores this data in a compact way; and the *Web layer* accesses RRD files, special files or queries from MySQL (most of information about the current status of the PC farm is stored in MySQL) and is responsible for real time display of this information on web page.

The web interface has been in in use for greater than three years. Significant functionality was added in comparison with original design. Changes made were based on experience with this system and also on feedback from collaborators.

4.4 Java FPS control interface

A full description of the system is given in reference [18]. Design considerations were concentrated on the goal of providing for shifters and experts a tool for platform independent access to PC farm control. In addition, access should be robust in a wide range of quality of network connections and network bandwidth.

Shifters and experts when alerted by FPS (through a paging system or sms message) just need to find a computer connected to the Internet (even through some low end modem) equipped with at least a basic Internet browser (like Netscape or Internet Explorer) and they can obtain full information and full access to take appropriate actions. This system loosens constraints on the physical presence of shifters and experts at Fermilab. The architecture of the design is schematically expressed in Figure **??** the for server side and in Figure **??** for the client side. Certain features of the design are as follows.

Information transfer between the client and server over the network is done by using IIOP (Internet Inter-ORB protocol) which is part of CORBA[19]. It proved to be stable, and there have been no problems with short term disconnections and reconnections. An XML processor[20] is used to generate and interpret the internal representation of data. Abstract internal representation of data is important to cope with (expected) frequent changes in the FPS system. A Java programming language, Java Web Start technology[21] was used for implementation of a platform independent client. The designed system was tested on SGI running IRIX 6.5 and Redhat Linux 6.2,7.0,7.1 (server) and Windows 98, Windows 2000, HP-UX, Solaris8, Redhat Linux 6.2, 7.0, 7.1 (client). Monitoring capabilities and control of FPS was tested for clients connected through various bandwidth network (56 kbps modem, 10 Mbit and Mbit and 100 Mbit Internet connections) with very satisfactory results!

5 Performance of the system

5.1 First tests

The first tests of the farm concentrated on some bacic principles of operation for CDF off-line production processing. A prototype PC Farm consisting of 14 nodes was used to test I/O throughput and the stability of execution of the production executable. A detailed account of these tests can be found in reference[23]. Several important conclusions were drawn from these tests, with data transfer between the I/O node and the worker nodes in a realistic scenario measured to be $\approx$ 20 MB/s. This result was encouraging because the prototype farm was still far from the final choice of hardware for the I/O node or network switches. The average CPU utilization when running the CDF Production Executable on the prototype farm was $\approx$ 95%, including the time needed by FBS to schedule, start and end a process.

Another positive result of these tests was a recognition of that rcp time out problems in cases of simultaneous copies of several files. We included in our software design a special strategy to avoid this problem. Later, tests were carried out to test the throughput of the final design of I/O nodes and worker nodes (details are in reference[22]). Results of the tests are displayed in Fig. **??**, **??**. It was clearly demonstrated that in the proposed hardware configuration throughput a factor of two better than required can be achieved.

5.2 Mock data challenges and Commissioning run

The first Mock Data Challenge was performed to study all of CDF offline system. The study was termed a "connectivity test" and the main goal was to connect the various CDF offline systems for the first time and to run modest number of simulated events through the system. The Challenge consisted of a simulated dataset of about 500,000 events that were first passed through the level 3 trigger system, the data logging system, the data file catalog, the data handling system[27], a tape robotic system with tape drives, and the prototype farm. The goal was to run all the systems simultaneously and to have the data move seamlessly from the level 3 systems through the farm and eventually onto separate physics streams, which would then be ready for analysis by physicists on experiment. The first Mock Data Challenge was conducted between December 1999 and February 2000.

The second Mock Data Challenge (MDC2) was meant to stress the systems at rates comparable to real situation.

The results from the data challenges were the following. Clear problems were found and fixed, missing or poorly specified interfaces and formats were identified, bottlenecks were found, and work was identified for future improvements of the system. Each iteration of the test led to further increases in performance and functionality and led to increased confidence that the system was able to perform at levels required by CDF.

5.3 Experience with the system from three years of data processing in Run 2

5.3.1 Commissioning Run

5.3.2 Early Run 2 Experience

5.3.3 Change of design

A modification to the farm to implement a different model of access to the input and output data from tape was accomplished in 2002. The new architecture of the farm is shown in Figure 1.

5.3.4 Reprocessing and operational issues

5.3.5 CPU time as a function of luminosity

6 Future

A natural question arises concerning the longevity of this design and possible hardware expansion/replacements that will be needed to satisfy the needs of the CDF experiment in the future. Data rates and operational issues experienced during the first three years of Run 2 were handled by the original design with one major modification to the I/O system. As the data-taking rates increase and the computing demands increase more fundamental changes may be required to handle the changing needs through the end of Run 2 data-taking (2009) and beyond. By Moore's law (which is expected to hold for at least the next 15 years) the density of transistors in a chip is doubled every 1.5 years. Remarkable justification of Moore's law over the last 30 years is displayed in fig. **??**. The CPU technogical advances have been matched by advances in the overall PC performance.

6.1 CPU Power

A clear strategy for increasing CPU power on the farm is a gradual replacement of old (3-4 years) machines with new ones every year. This gradual replacement will naturally lead to large and necessary increases in CPU power which will go a long way in providing sufficient CPU power for data processing on the CDF farms. The long-term planning model assumes this yearly replacement will occur. The first such replacement occurred in 2003 when the first 500 MHz CPUs were decommissioned.

In table 1 the current status of the PC Farm from the point of view of CPU power and its evolution from the beginning of the CDF PC farms is shown. A proposed strategy for future acquisitions and replacements for the years 2004-2006 is shown in Table 2.

6.2 Data Rates (I/O)

Currently the farms handle 20 MByte/s input and the equivalent output. To increase to larger rates of I/O is straightforward. More I/O systems can be installed, higher rate tapedrives can be used, and more parallel access to the

Fiscal Year	Numbers	Type	GHz	Total GHz
1999	50	PIII/500 duals	50	50
2001	23	PIII/800 duals	37	87
2001	64	PIII/1000 duals	128	215
2002	32	PIII/1260 duals	81	286
2002	32	AMD/1666 duals	107	393
2003	-50	PIII/500 duals	-50	343
2003	64	PIV/2.2 duals	282	625

Table 1: Evolution of CDF PC Farm for offline data processing in Run 2

Fiscal Year	Numbers	Type	GHz	Total Ghz
2004	-23	PIII/800 duals	-37	588
2004	64	3500 MHz duals	448	1026
2004	-64	PIII/1000 duals	-128	898
2005	64	5600 MHz duals	717	1615
2005	-32	1260 MHz duals	-81	1534
2005	-32	1666 MHz duals	-107	1427
2006	64	8800 MHz duals	1126	2553
2006	-64	2200 MHz duals	-282	2271

Table 2: Proposed strategy for evolution of CDF PC Farms for offline data processing for the years 2004–2006

data can be included in the system. A change in strategy that puts a disk cache system (dcache) into the input and output data paths also promises to increase the speed of I/O of the farms. This system can provide "tapeless" data paths from the online/L3 system through the farms.

6.3 Distributed processing and the GRID

From the above discussion it follows that the current software design together with natural upgrade of hardware satisfies needs of CDF collaboration for offline data processing for the near future. In the future it may be desirable to process data more quickly and to make use of other resources at Fermilab and/or all over the world. This will include using the SAM[**?**] system and the grid[26].

7 Conclusion

The CDF production PC farms have been successfully prototyped, installed, integrated, commissioned and operated for many years. They have been and continue to be successful in providing the computing processing capacity required for the CDF experiment in Run 2. The success of this system has in turn enabled successful analysis of the wealth of new data being collected by the CDF experiment at Fermilab. The system has been modified and enhanced during the years of its operation to adjust to new requirements and to enable new capabilities.

The system will continue to be modified in the future to continue to serve the CDF collaboration as required. Some of the modifications are simple upgrades of components with more capable (faster, more capacious) replacements. Other modifications will affect the architecture of the system and quite likely will embrace distributed processing and the grid in some way. These developments will allow CDF to continue to process and analyze data through the end of the life of the experiment.

References

[1] www-bd.fnal.gov/run2upgrade

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

[3] T. Nash, Comp. Phys. Comm. 57 (1989) 47.

[4] I.Gaines and T. Nash, "Use of New Computer technologies in Elementary Particle Physics", Ann.Rev.Nuc.Part.Sci.37:177 (1987)

[5] Frank Rinaldo, Stephen Wolbers, "Loosely Coupled Parallel Processing at Fermilab" Comput.Phys.7:184-190 (1993)

[6] G.-P. Yeh,et al., PC Farms for Parallel Computing, Contribution to CHEP 98'

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

[8] http://cddocs.fnal.gov/cfdocs/products/productsDB/ProdDetail.CFM?ProdNum=SU0037.

[9] http://www-isd.fnal.gov/fbsng

[10] CDF recommendations for FBS, cdfnote 4942

[11] http://hppc.fnal.gov/enstore/

[12] Modifying the CDF production farms to use Enstore, cdfnote 5890

[13] http://www.python.org

[14] http://www.mysql.com

[15] Design of the Production Farm web interface, cdfnote 5556

[16] http://www.php.net

[17] http://ee-staff.ethz.ch/˜ oetiker/webtool/rrdtool/

[18] Java Farms Processing System Control Interface, cdfnote 5933

[19] http://www.corba.org

[20] http://www.w3.org/XML

[21] http://www.sun.com

[22] Performance Tests of the CDF Run 2 Production Farm, cdfnote 5467

[23] 14 Node Test of the Prototype Farm, cdfnote 4941

[24] The CDF Detector for Tevatron Run II. Technical Design Report, cdfnote 3702

[25] http://www-cdf.fnal.gov/internal/run2b.html

[26] http://eu-datagrid.web.cern.ch/eu-datagrid/

[27] R. Colombo, *et al.,* The CDF Computing and Analysis System: First Experience, Proceedings of the Computing in High-Energy and Nuclear Physics Conference, Beijing, China, 3-7 September, 2001.

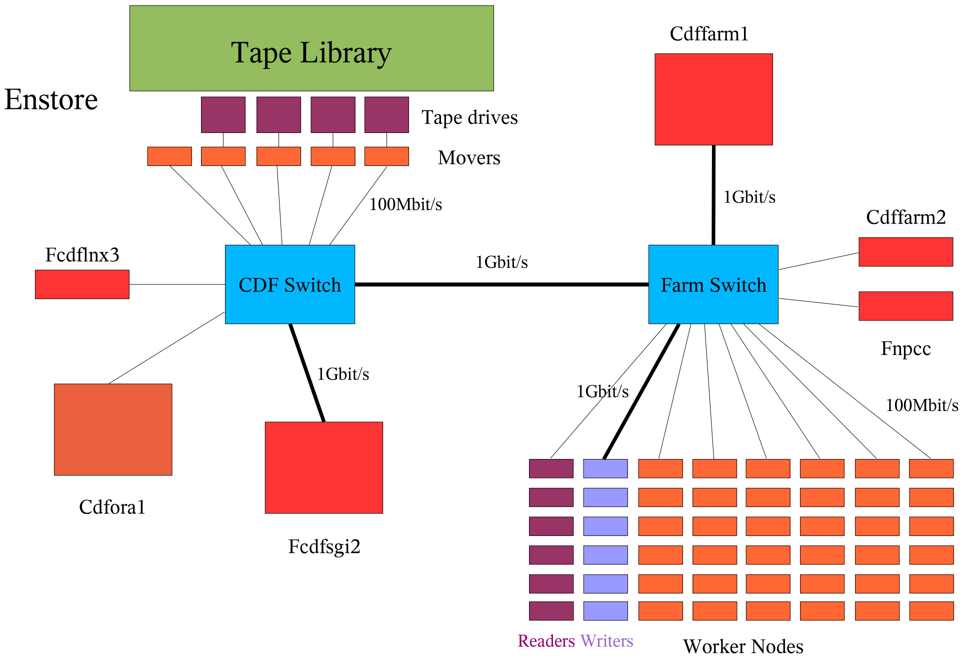

Figure 1: Design of the CDF Farm with Enstore mass storage system

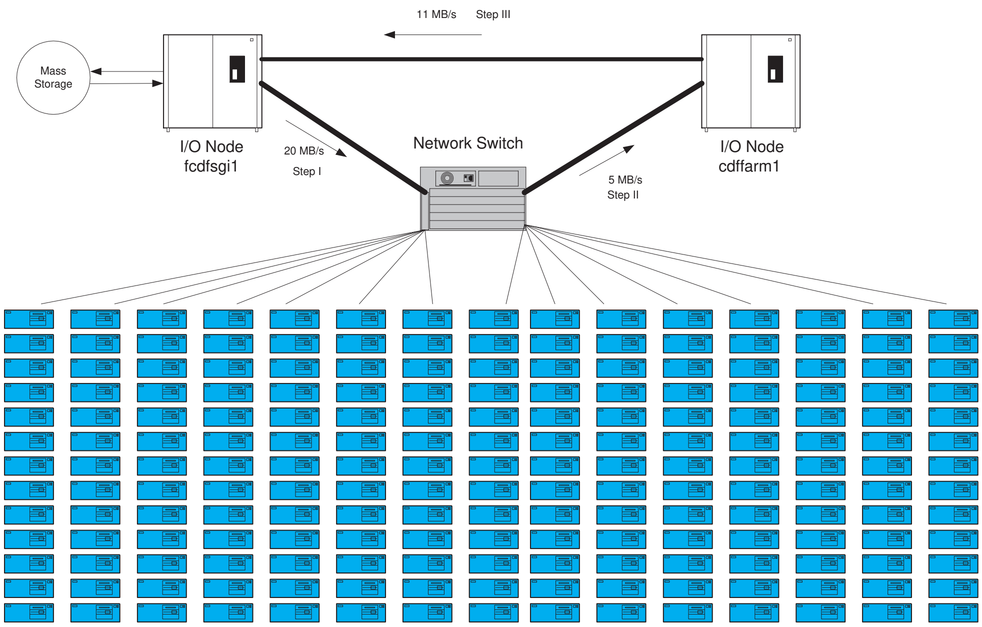

Figure 2: Original CDF farm design