Design and First Tests of the CDF Run 2 Farms

Version 2, March 27, 2000

Jaroslav Antos [a], M. Breitung [b], Tony Chan [c], Paoti Chang [c], Yen-Chu Chen [c], Troy Dawson [b], Jim Fromm [b], Lisa Giacchetti [b], Tanya Levshina [b], Igor Mandrichenko [b], Ray Pasetes [b], Marilyn Schweitzer [b], Karen Shepelak [b], Miroslav Siket [a,d], Dane Skow [b], Stephen Wolbers [b], G.P.Yeh [b], Ping Yeh [c]

[a] *Slovak Academy of Science, Slovak Republic*

[b] *Fermi National Accelerator Laboratory, Batavia, IL, USA*

[c] *Academia Sinica, Taiwan*

[d] *Comenius University, Slovak Republic*

Abstract

CDF will write data in Run 2 at twenty times the rate of Run 1. The offline production system must be able to handle this rate. Components of that system include a large PC farm, I/O systems to read/write data to and from mass storage, and a system to split the reconstructed data into physics streams which are required for analysis. All of the components must work together as seamlessly as possible to ensure the necessary throughput. A description will be given of the overall hardware and software design for the system. A small prototype farm has been used for about one year to study performance, to test software designs and for the first Mock Data Challenge. Results from the tests and experience from the first Mock Data Challenge will be discussed. The hardware for the first production farm is in place and will be used for the second Mock Data Challenge. Finally, the possible scaling of the system to handle larger rates foreseen later in Run 2 will be described.

1 Introduction

The CDF detector will take enormous amounts of data in Collider Run II. The Higgs and SUSY searches, among others, will require that data be analyzed as quickly as possible. To keep up with data taking, the data production will use large amounts of CPU power, therefore a large offline farm of computers is required. CDF and Fermilab have decided to use as much commodity hardware as possible to build the farm to reduce the cost. The choice is hundreds of off-the-shelf PCs running the Linux operating system.

The main usage of the offline farm is for data production in quasi-real time, which includes reconstructing raw data, splitting them into physics datasets and storing physics data on tape for physicists to analyze. It will also be used for data reprocessing should reprocessing be requested. Monte Carlo generation is also possible when there are CPU cycles to spare.

This paper is organized as the following: In section 2 we state various requirements about Run II data and how they affect the farms. Section 3 shows the conceptual model of the farm that was built upon the facts and requirements. The efficiency of the farm is an important factor for the success of data production, thus it is the main focus when designing the production software system. Many factors contribute to the efficiency: error rate, error recovery time, idle time, etc. Their effects on the efficiency, as well as many other aspects of the system, are analyzed in section 4. Initial use of the software and the prototype farm hardware is given in section 5. A discussion of possible extension of capability beyond initial Run 2 rates is given in section 6. Conclusions and next steps are given in section 7.

2 Some Facts and External Requirements

There are many goals for CDF production. Some of the major goals are the following:

- Sufficient capacity to reconstruct raw data.

  Clearly the first goal of a production system is to have the capacity to reconstruct all of the raw data from the experiment in "quasi-real time", meaning that the capacity is there to process data taken in a given time period (e.g. a month) in that same time period, with a small delay which is required for calibration constants preparation. There should be no design constraints in the system which preclude reconstruction with essentially no delay.
- Sufficient capacity to reprocess some data.

  In addition to processing data for the first time there is a requirement that data be reprocessed in certain instances. Early processing of Run II data will likely need to be reprocessed once the detector is well-understood and the reconstruction packages are tuned. We need to plan capacity to be able to meet that need. There may be smaller subsets of the data (specific triggers, for example) that may require reprocessing to achieve consistent datasets. It may also be the case that reprocessing is accomplished on the output and not the raw data files. The system should be able to schedule this.

- Ability to easily configure farm processing to control priorities.

  The ease of use requirements include the ability to control priorities on the farms. This is important to allow the farm to be configured quickly to handle rapidly changing needs of the collaboration. There may be a new trigger which needs to be understood and tuned, a physics result which needs rapid processing, or other conditions under which the farm has to be configured to process specified sets of data using certain code and constants.

- Excellent bookkeeping.

  The bookkeeping that is required on the farms includes the details of versions of code, constants, etc. which were used as well as a clean and clear way to correlate events or files with those version numbers. It also may include the capability to handle various requests about the number of events run through each version of the reconstruction package, etc. The exact information which needs to be kept will be driven by the needs of the people who run the farms, as well as the needs of the physicists who will analyze the data.

- Excellent error handling and error prevention.

  One additional goal of such a system is to have excellent error handling. This makes such a system much easier to run and to maintain, and good attention to this will pay off over the long run. Another aspect of this is to have a well-designed system in which errors are less likely to occur. In other words, attention will be paid to the experience of Run I farms and other experiences as the system is designed.

- CPU and I/O Capacity

  The capacity required for the farms is driven by the rate events are to be taken by the experiment and by the time it takes to reconstruct each event. Additional time will have to be added to handle reprocessing, Monte Carlo, splitting and any other anticipated needs.

  The required throughput is 28 Hz, and the peak rate of the level 3 data logger is 75 Hz (about 20 MByte/s). For output, the throughput requirement is 11 MByte/s, given the ratio between output and input event sizes.

  The CPU estimates (given in 500 MHz PIII processor units) are 5 seconds per event which translates into a requirement of 140 PIII/500 processors if the system runs at 100% efficiency. Choosing a more realistic 70% overall efficiency one calculates a requirement of 200 PIII/500 processors to keep up with the DC rate. This does not include reprocessing, peak demands, Monte Carlo, etc. Therefore it is expected that the Run 2 CDF farm will be required to have approximately 300-400 PIII/500-equivalent processors. This translates to 6180-8240 SpecInt95.

3 Conceptual Model

The hardware of the data production system consists of

- One or a few input/output nodes which are connected to the mass storage system with large disk buffers, and
- many worker nodes for data reconstruction.

The I/O node(s) and the worker nodes are connected by the network. The number of input nodes and output nodes are not yet specified.

It is expected that the system that processes new raw data will function as follows: The raw data files, stored in several input streams, reside on the mass storage system or external disks. A process stages raw data to disks on the input node. Each raw data file is then dispatched to one worker node for reconstruction. The reconstructed inclusive data along with monitor files (log files, bookkeeping records, validation histograms, etc) are collected to an output node. Inclusive data are split based on triggers into different physics datasets and concatenated to meet the 1 GB target file size. The physics data are then spooled to tapes. Some physics data may be copied to external disks. The monitor files are exported to an external production monitor computer for shift-takers to examine.

A conceptual model of the production system is shown graphically in figure 1. It shows the data flow. A UML activity diagram in figure 2 shows the activities in the system, as well as transitions and synchronizations in the same conceptual model. The input node and output node are separated in figure 2 for clarity. The notations are described below.

The diagram is divided into 3 areas (so-called *swimlanes*): Input Node, Worker Node and Output Node.

The activities start at a solid black circle. There are two starting points on the input node. They are parallel to each other. An arrow goes out of a circle. Such an arrow is called a *transition*. The first transition leads to the "Input" *activity*. The texts "raw data" on the outgoing transition is for description purpose only.

This transition ends on a *synchronization bar*. Another transition coming from the "Retrieve Database" activity also ends on the same synchronization bar. This means that both transitions end here and no further transition may occur until the *synchronization condition* is met. In the diagram the condition is that the "Dispatch" activity starts only when raw data and the database valid for the raw data are both present on the input node.

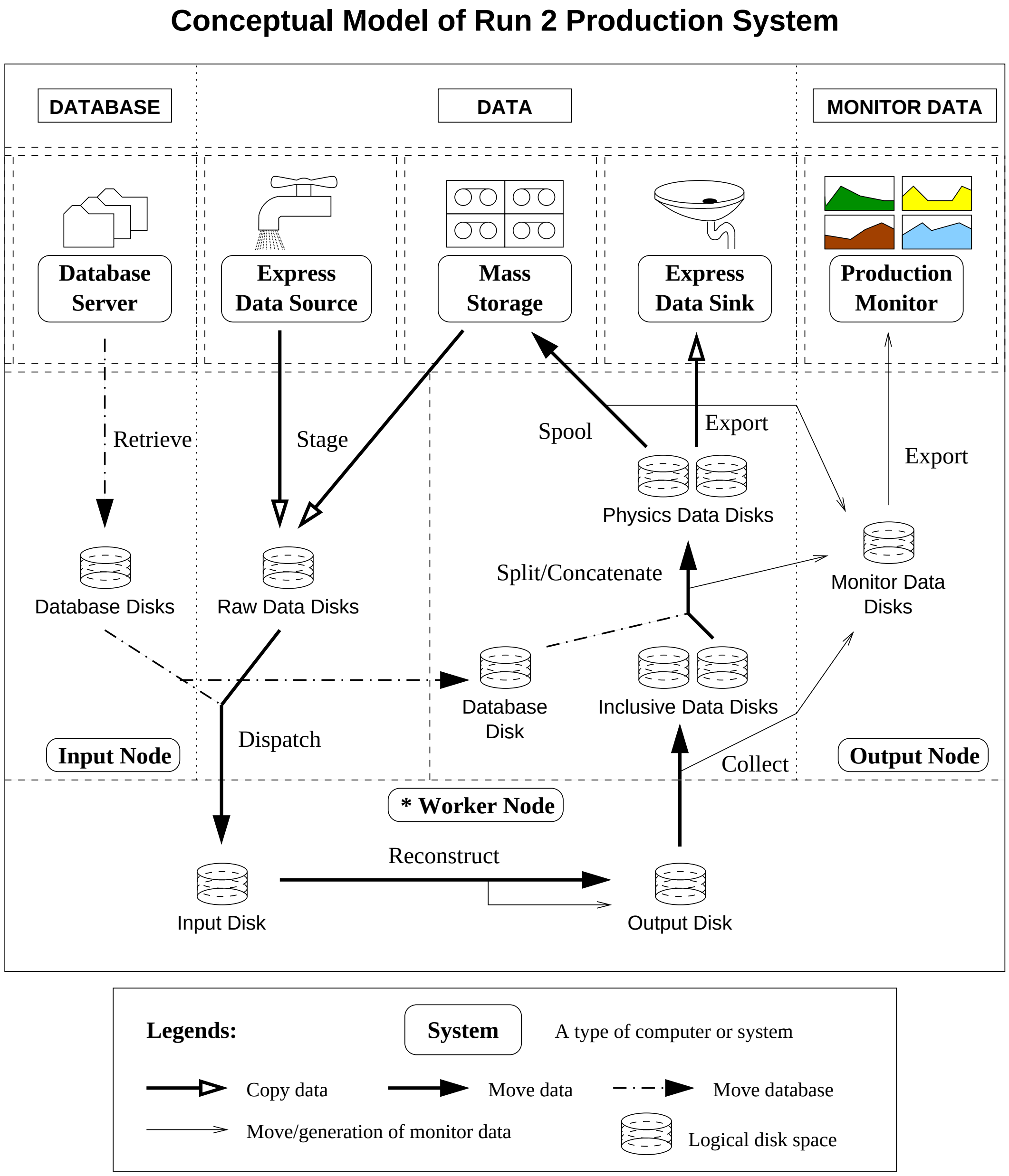

Fig. 1. The conceptual model of the Run 2 data production software system. External systems are enclosed in double dashed lines. Internal systems are separated by single dashed lines. Note that the separation between input node and output node is only logical. They could be the same physical computer.

The outgoing transition of “Dispatch” activity is a *multiple trigger*, indicated by the presence of a ‘*’, which shows that there may be many such transitions occurring in parallel.

A synchronization bar is used at the outgoing transition of “Collect” activity, which signals the end of the multiple trigger started on the dispatch activity.

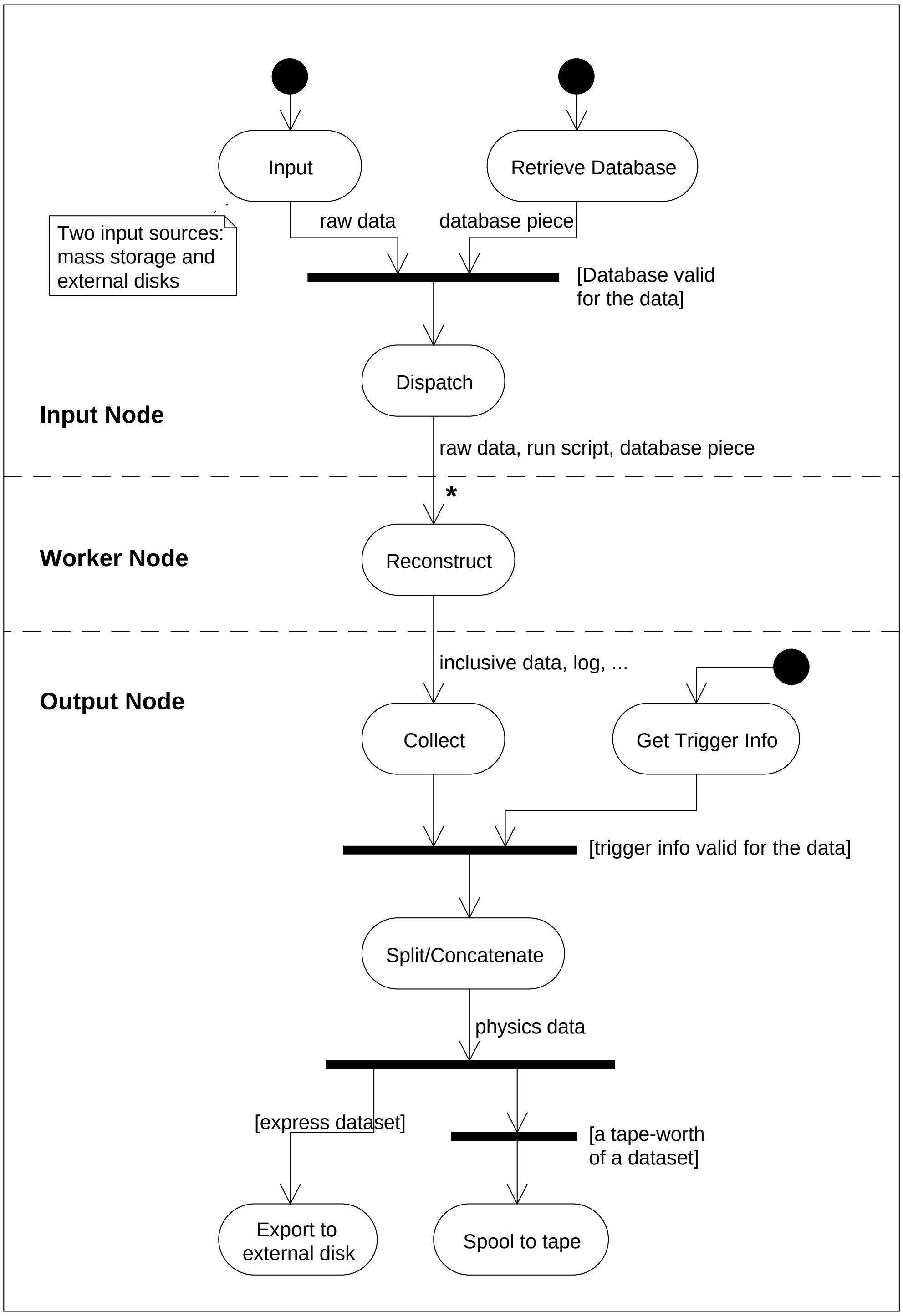

Fig. 2. The conceptual model of the Run 2 data production software system as an UML activity diagram. See texts for explanation of notations.

Another synchronization bar is present at the outgoing transition of "Split/Concatenate" activity. In this case there is only one incoming transition on the synchronization bar, but there are multiple outgoing transitions. This indicates that any of the outgoing transition may occur independently with no ordering requirements. In this example it shows that the physics data may be spooled or exported to external disks.

The transition to “Export to external disks” activity has a *guard* — a logical expression — enclosed in brackets. A transition occurs only when the guard is evaluated to true. Here the physics data is exported only when express_output is true, giving a way to turn it on/off. A transition without guard has a default guard of [true].

Figure 1 illustrates all key points of the production system. We'll analyze this conceptual model in section 4.

4 System Analysis

There are 7 requirements to fulfill. All tasks in the system as well as the system as a whole are analyzed to arrive at a software design to fulfill the requirements.

4.1 Pipeline and Buffers

The data production system can be viewed as a 6-stage data pipeline (input, dispatch, reconstruction, collect, split/concatenate, output), plus peripheral input/outputs. But it is not a usual straight-forward pipeline because

(1) the parallel processing of worker nodes which breaks the sequence of files, especially when errors occur; and
(2) the staging and spooling stage expect a collection of files as input while other stages operate with one file at a time.

It is mandatory to have data buffers at the input end of the split/concatenation and spool stage to accumulate files until they can be processed. The size of the data buffers is estimated to be**?**)

- 200 GB for input staging
- 200 GB for output staging
- 750 GB for physics dataset output.

4.2 Parallel Processing and Synchronization

There are 4 synchronization bars in the conceptual diagram (figure 2). Three of them are ordinary synchronizations, and the remaining is a parallel transition. Synchronization is one main focus of the system because if the system is out of sync buffer filling may occur and the efficiency may be seriously degraded.

4.2.1 Database Synchronization

The first synchronization is between raw data and calibration/trigger database on worker nodes. To avoid the raw data files stacking up in worker nodes when database is out of sync, it may be desirable to move this synchronization ahead to earlier stages in the pipeline. Since all disk space inside the production system is limited while the mass storage can be considered infinite in capacity, the best way is to trigger the "Input" activity for given raw data files *only when corresponding databases are ready.* This minimizes the probability of raw data disk congestion and/or waste of CPU time.

The output of split/concatenation stage are data of various physics datasets. Since the data of the same dataset aren't spooled to tape until a tape-worth (or possibly a file-set, approximately 10 GByte) of them appear on the disk, there is no clever solution for reducing the chance of congestion on physics disks. To avoid disk congestion, at least one of the following must hold:

(1) there are not too many physics datasets,
(2) the tape size is small,
(3) the physics data buffer is very large.

At this point it seems we have to go for option 3.

If we are allowed to use more tapes than necessary, we may design a post-production system such that data of a few datasets are stored on the same set of tapes, then files of the same dataset are gathered one by one into single-dataset tapes in the post-production system. If not every physics dataset needs to have this post-production, the hardware requirement on the post-production is not very high. A single computer with a few tape worth of disk space and a few tape drives shall be sufficient for the post-production.

The total throughput of a pipeline is determined by the slowest stage. Therefore we only need to analyze the rate and efficiency of each stage, and identify the slowest one.

The throughput of a stage is

$$R = B \times \epsilon$$

where

$$B = \text{available hardware bandwidth in MB/s,}$$
$$\epsilon = \text{efficiency of the stage.}$$

The available hardware bandwidth B in each stage is different and is not precisely known yet. Actually B may grow as time goes on. We'll concentrate on the efficiency, then use our current estimate of available hardware bandwidth to find the requirement on efficiency on each stage.

The bandwidth of various parts of the production system are listed below:

- Tape drive:
  The tape drive hardware is not decided. Before the start of Run II it is quite likely that newer technology with 6 MB/s speed or higher with capacity around 50 GB will mature. In that case we'll need 5 input tape drives and 3 output tape drives, plus some more for buffering and spare.
- Disk:
  The disk speed from the 14 node test is approximately 20 MB/s**?**). With RAID disks, the disk speed is expected to reach 50 MB/s or more, which is well beyond 30 MB/s. However, it will drop considerably if multiple read/write occurs on the same disk array whether or not the disk head contention costs much performance. Therefore it may be desirable to avoid multiple read/write on the same physical disks.
- I/O bus:
  The fibre channel on input/output nodes should have bandwidth around 1 Gb/s, or $\geq$90 MB/s. The SCSI bus on the worker nodes will probably have bandwidth of 20-40 MB/s. The bandwidth on worker nodes is not a problem because there are multiple worker nodes. The bandwidth on the output node is also suitable to sustain 5 concurrent read/write tasks of 11 MB/s each.
- Network:
  The network on input/output nodes will be gigabit ethernet ($\approx$90 MB/s wire speed), and on worker nodes are 100BaseT ($\approx$9 MB/s). With multiple worker nodes the bandwidth is sufficient.

Overall there are known no show-stoppers on bandwidth issues.

4.3 Worker Node Management and Utilization

The farms group of computing division has been developing the Farm Batch System (FBS) with the concept of clean worker nodes built in — all input files and output files can be deleted automatically when a job is done. It also provides worker node monitoring, among other features.

With FBS, one can submit job scripts in advance. FBS will start the job on a processor when the processor is available, and perform clean-ups after the job is done. In the job script, one can copy the file from input node to worker node, run the executable, copy the files to output node.

One caveat of this scheme is that it couples network input/output with CPU, so the CPU is idle when the worker node is doing network I/O. This increases f_{idle}. However, we expect f_{idle} to be fairly small because

(1) It takes about 8 hours to reconstruct a file, but it only takes about 3 minutes to copy it on input and output combined. That contributes 0.6% to f_{idle}.
(2) The input/output node will allow more than one network copy at the same time (limited by the disk head contention and network speed). That decreases the chance that the worker node has to wait.
(3) In case the worker node really has to wait, the average waiting time is small.

4.4 Bookkeeping and Catalog

When data files go through the production pipeline, a bookkeeping database should be used to keep track of the data files that are in the system. The database may not need to be implemented with a commercial database system, but it has to allow reliable multiple access and remote access since both input node and output node will perform reading/writing.

Another database may be used to keep track of errors/failures for further error analysis. It is not known yet how the catalog system will be for the files exported to the external systems, a fourth database may be needed to keep track of exported files.

4.5 Error Handling

Error handling is extremely important in such a large and dynamic system. It is not sufficient to fix problems "by hand", as the detailed work involved could potentially overwhelm the people involved and the farm could become idle for substantial fractions of time. Both of these situations are to be avoided.

The first level of error handling should be to avoid errors whenever possible. This will be built into the systems that comprise the farm system. It is hoped that the CDF executable will be robust. The farms software itself, including FBS, should be as error-free as possible. Finally, the hardware should be purchased with the avoidance of errors in mind. This may mean such simple things as specifying proper cooling in the PC's, redundant hardware or hot-swap capability, etc.

However, there will be errors at some rate and the next level of error handling will be the ability to automatically detect errors and take appropriate action. This can take many forms. If a worker node crashes, the reconstruction job could be automatically moved to another worker node, updating all of the logging databases appropriately. If the CDF executable crashes, the job could automatically restart. After a certain number of repeated failures the job could restart but skip the offending event (keeping a record of what was done of course). There are other approaches that could be taken. Other problems such as tape drive problems, disk failures, disks filling, etc. all should be handled automatically as much as possible.

The next level of error handling involves getting shifters to look at the system and take action to fix the problem, clean off disk space, call operations to get systems reset, or whatever is required to get the farm running properly. The shifters could be paged automatically, be sent email, or be contacted when they monitor the farm as part of their shift responsibilities.

The higher levels of error handling involve experts and could involve rebooting machines, replacing bad hardware, helping to debug scripts or other control software, contacting CDF code experts, etc. This level of error handling should be invoked rarely in a smoothly operating system.

4.6 Flexibility Requirement

To fulfill the requirement of flexibility, several global features must be present in the system:

- There should be a priority system for input streams, runs, or even files.
- Most, if not all, stages in the pipeline must be configurable before running and at run time.
- There should be a centralized sub-system to accept requests like change priority, add/remove stream, reprocess a run, ... etc. This sub-system should be smart enough to know how to re-configure the pipeline in response of the request.

5 Prototype and Mock Data Challenge Experience

CDF has been using a prototype Run 2 farm for over one year with great success. The prototype consists of 4 "I/O" nodes (dual PII/400's) with Gbit ethernet interfaces, 512 MByte memory and large disk storage, 14 "worker" nodes (dual PII/400's) with 100 Mbit ethernet interfaces, 512 GByte memory and 18 GByte local disk storage. This system has been used by the two collider experiments CDF and D0, the Computing Division, and by lattice QCD collaborators. The prototype farm has been used to study the CDF executable behavior, the batch scheduling system, the CDF farms control software system, I/O rates and was used for the first Mock Data Challenge. Experience from this system was used in the specification of the first large purchase of worker and I/O nodes for Run 2.

Tests on the prototype showed that the computing to be done could easily scale from one to as many of 14 (the maximum available) nodes, consisting of 28 processors. The network and disk susbsystems were tested and no serious limitations were found to the transfer of data. Given the disks available and the network connections, the rate of 20 MByte/sec from the 14 worker nodes to and from the I/O node was as expected.

A Mock Data Challenge was performed to study the entire CDF offline system. The study was termed a "connectivity test" in that the main goal was to connect the various CDF offline systems for the first time and to run a modest number of simulated events through the system. The Challenge consisted of a simulated dataset of about 500,000 events, the level 3 trigger system, the data logging system, the data file catalog, a tape robotic system with tapedrives, 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 farms and eventually onto separate "physics" streams, which would then be ready for analysis by physicists on the experiment.

The first Mock Data Challenge (MDC1) was successfully run in December of 1999. The goals of connectivity were met and all systems were able to communicate and move data properly through the entire system. Due to technical issues having to do with dataset definitions part of MDC1 was delayed until February of 2000.

The second Mock Data Challenge (MDC2), to be run in late spring 2000, is meant to stress the systems at rates comparable to rates expected during datataking for a long enough period of time that bottlenecks, both serious and subtle, are identified and possibly fixed. Doing so will go a long way to preparing the offline systems, including the farms, for full data taking in spring of 2001 and extending for about 5-6 years after.

6 Scalability beyond Run 2

The initial goals for Run 2 include a total integrated data sample of 2 fb^{-1} in approximately two years of running. This is accomplished with the L3 output rates (and consequent farm rates) of a maximum of 75 Hz (approximately 20 MByte/s) and a DC rate of 28 Hz. Run 2 will not end at 2 fb^{-1}. Though the exact parameters are not yet known it is clear that the luminosity will increase substantially after the first two years. It is assumed that the data-taking rate will also increase. The farms and production systems design should scale to substantially higher rates without a major redesign of the system. The following components should scale easily, mainly by improvements in the computing industry which are expected:

- CPU
  The Pentium and followon commodity computing chips will continue to improve. This will automatically lead to increased CPU power as older boxes are replaced. In addition, the farm should scale with the addition of boxes. The system is modular enough that the increase in boxes can be quite large without reaching any serious bottlenecks or those bottlenecks can be eliminated by a simple change in toplogy somewhere else in the system.
- I/O
  The increase I/O required through the system can be addressed in many ways. First, the tape I/O can be improved by using newer, faster tapedrives which are expected. More tapedrives can be added to the system as well. Disk I/O rates and disk storage requirements will increase and can be accommodated with expected improvements in disk and disk interface technology. The CPU necessary to split the data into final physics datasets can be increased by the upgrade and/or addition of processors to an I/O node or by adding a second (or third or fourth) I/O node to the system. The network can be made more capable by upgrading links from 100 Mbit to 1 Gbit or 1 Gbit to 10 Gbit or in some cases by multiple interfaces. The farm switch can be replaced with a more capable switch or if that is not possible by adding a second switch.

  In all of these cases the improvements required can be achieved because of the modular design of the system.
- Software
  A more serious concern is the capability of the software system to manage the systems and to function properly with a large scale-up of the systems. There are a few places (e.g. database access) where the software is not modular and will require capabilities substantially greater than those achieved up to now or during the first two years of Run 2. This will have to be studied and understood before such an upgrade is attempted. The rest of the software is modular already and should have little or no trouble scaling substantially beyond that which will be achieved in Run 2.

7 Conclusion

Run 2 will produce large amounts of data which will have to be processed quickly. The ability to handle the data quickly and easily will have a direct impact on the ability for CDF to analyze the data and to produce physics results in a timely fashion. The system described in this paper does have the necessary software design and hardware and software implementation to keep up with Run 2 data rates. Further improvements to the system to scale up to expected increases in luminosity and data-taking rate are possible and will allow the production systems to function for many years.